Annand_DSE6003_Final

October 20, 2023

1 DSE6003 Final Project Supplemental Code

1.1 Joseph Annand

1.2 10/20/2023

Import pandas and dataset. To ensure data quality, any NA values are removed from the dataset. A separate dataframe with only the five quasi-identifiers is created to focus on de-identification efforts.

[14]:
```
import pandas as pd

data = pd.read_excel("Merr.Customer.Survey (2).xlsx")
data.dropna()

quasi_data = data[['Region', 'Gender', 'Age', 'EducationYears',␣
 ↪'HouseholdIncome']]
```

A variety of functions were created in order to de-idnetify quasi-identifiers by converting values into ranges of values. This is done for Age, Household Income, Education Years, and Region.

[15]:
```
def de_id_age(value):
    if value >= 18 and value < 26:
        return "18-25"
    elif value >= 26 and value < 35:
        return "26-34"
    elif value >= 35 and value < 45:
        return "35-44"
    elif value >= 45 and value < 55:
        return "45-54"
    elif value >= 55 and value < 65:
        return "55-64"
    elif value >= 65 and value < 75:
        return "65-74"
    elif value >= 75:
        return "75+"


def de_id_income(value):
    if value <= 30:
```

```python
        return "0-30000"
    elif value > 30 and value <= 80:
        return "30001-80000"
    elif value > 80 and value <= 120:
        return "80001-120000"
    elif value > 120:
        return "120001+"


def de_id_education(value):
    if value < 11:
        return "0-11"
    elif value > 11 and value <= 15:
        return "12-15"
    elif value > 15 and value <= 19:
        return "16-19"
    elif value > 19:
        return "20-23"


def de_id_region(value):
    if value >= 1 and value <=3:
        return "1-3"
    elif value > 3:
        return "4-5"
```

New columns were created in the data frame to reflect the new generalized values for the quasi-identifiers that would be used in creating equivalence classes.

[16]:
```python
quasi_data['age_de_id'] = quasi_data['Age'].map(de_id_age)
quasi_data['income_de_id'] = quasi_data['HouseholdIncome'].map(de_id_income)
quasi_data['edu_de_id'] = quasi_data['EducationYears'].map(de_id_education)
quasi_data['region_de_id'] = quasi_data['Region'].map(de_id_region)
```

```
C:\Users\janna\AppData\Local\Temp\ipykernel_28120\3279607735.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  quasi_data['age_de_id'] = quasi_data['Age'].map(de_id_age)
C:\Users\janna\AppData\Local\Temp\ipykernel_28120\3279607735.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
```

```
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  quasi_data['income_de_id'] = quasi_data['HouseholdIncome'].map(de_id_income)
C:\Users\janna\AppData\Local\Temp\ipykernel_28120\3279607735.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  quasi_data['edu_de_id'] = quasi_data['EducationYears'].map(de_id_education)
C:\Users\janna\AppData\Local\Temp\ipykernel_28120\3279607735.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  quasi_data['region_de_id'] = quasi_data['Region'].map(de_id_region)
```

Initialize variables that are equal to or used in calculating the probabilities of each re-identification action ocurring.

[17]:
```python
# Probability of attempt based on Verizon DBIR - 83% are from external␣
↪adversaries
pr_attempt = 1 - 0.83

# Number of customers with Merrimac Communications before sale
merr_customers = 6000
# U.S. Adult Population
us_pop = 260836730

# Probability of an analyst knowing someone in the data set
pr_acquaintance = 1 - ((1-(merr_customers/us_pop))**150)

# Probability of a data breach is 60% for large companies
pr_breach = 0.60
```

A dataframe of all the equivalence classes is created. Each equivalence class has the same combination of all five quasi-identifiers.

[18]:
```python
# Create dataframe of equivalence classes
eq_classes = quasi_data.groupby(['age_de_id', 'Gender', 'region_de_id',␣
↪'edu_de_id', 'income_de_id']).size().reset_index()
eq_classes.rename(columns = {0: "count"}, inplace=True)
eq_classes['pr_re_id'] = 1 / eq_classes['count']
```

The risk for each scenario is calculated for each equivalence class and added to the data frame.

[19]:
```
# Calculate Pr(re-id) for all four scenarios
eq_classes['scenario_1'] = pr_attempt * eq_classes['pr_re_id']
eq_classes['scenario_2'] = pr_acquaintance * eq_classes['pr_re_id']
eq_classes['scenario_3'] = pr_breach * eq_classes['pr_re_id']
eq_classes['scenario_4'] = 1.00 * eq_classes['pr_re_id']
```

A function to assign each equivalence a range of re-identification risk it falls into.

[20]:
```
def determine_risk_range(value):
    if value < 0.05:
        return "<5%"
    elif value >= 0.05 and value < 0.10:
        return "<10%"
    elif value >= 0.10 and value < 0.20:
        return "<20%"
    elif value >= 0.20 and value < 0.33:
        return "<33%"
    elif value >= 0.33 and value < 0.5:
        return "<50%"
    elif value >= 0.5:
        return ">50%"
```

We apply the function to determine thr risk range to the equivalence class data frame.

[21]:
```
# Calculate risk for each equivalence class for each scenario
eq_classes['s1.risk'] = eq_classes['scenario_1'].map(determine_risk_range)
eq_classes['s2.risk'] = eq_classes['scenario_2'].map(determine_risk_range)
eq_classes['s3.risk'] = eq_classes['scenario_3'].map(determine_risk_range)
eq_classes['s4.risk'] = eq_classes['scenario_4'].map(determine_risk_range)
```

We determine the distrubtion of risk across each equivalence for each scenario.

[22]:
```
s1_prob = eq_classes.groupby('s1.risk').size().reset_index()
s1_prob.rename(columns ={0: "count"}, inplace=True)
s1_prob['percentage'] = s1_prob['count'] / s1_prob['count'].sum()
print(s1_prob)

s2_prob = eq_classes.groupby('s2.risk').size().reset_index()
s2_prob.rename(columns ={0: "count"}, inplace=True)
s2_prob['percentage'] = s2_prob['count'] / s2_prob['count'].sum()
print(s2_prob)

s3_prob = eq_classes.groupby('s3.risk').size().reset_index()
s3_prob.rename(columns ={0: "count"}, inplace=True)
s3_prob['percentage'] = s3_prob['count'] / s3_prob['count'].sum()
print(s3_prob)

s4_prob = eq_classes.groupby('s4.risk').size().reset_index()
s4_prob.rename(columns ={0: "count"}, inplace=True)
```

```
s4_prob['percentage'] = s4_prob['count'] / s4_prob['count'].sum()
print(s4_prob)
```

```
  s1.risk  count  percentage
0    <10%     66    0.176471
1    <20%     51    0.136364
2     <5%    257    0.687166
  s2.risk  count  percentage
0     <5%    374         1.0
  s3.risk  count  percentage
0    <10%     82    0.219251
1    <20%     74    0.197861
2    <33%     36    0.096257
3     <5%    131    0.350267
4    >50%     51    0.136364
  s4.risk  count  percentage
0    <10%     86    0.229947
1    <20%     68    0.181818
2    <33%     44    0.117647
3     <5%     59    0.157754
4    <50%     30    0.080214
5    >50%     87    0.232620
```

The maximum, average, and median risk was calculated for each scenario. Average risk is the same across all scenarios because all scenarios use the same number of equivalence classes and include the same total number of records.

[23]:
```
# Scenario 1
print("Scenario 1:")
print(eq_classes['scenario_1'].max())
print(eq_classes['scenario_1'].median())

# Scenario 2
print("Scenario 2:")
print(eq_classes['scenario_2'].max())
print(eq_classes['scenario_2'].median())

# Scenario 3
print("Scenario 3:")
print(eq_classes['scenario_3'].max())
print(eq_classes['scenario_3'].median())

# Scenario 4
print("Scenario 4:")
print(eq_classes['scenario_4'].max())
print(eq_classes['scenario_4'].median())

#Average risk across scenarios
```

```python
print("Average Risk")
len(eq_classes) / len(data)
```

```
Scenario 1:
0.17000000000000004
0.02428571428571429
Scenario 2:
0.0034445279415022956
0.0004920754202146137
Scenario 3:
0.6
0.0857142857142857
Scenario 4:
1.0
0.14285714285714285
Average Risk
```

[23]: 0.0748

[ ]: