National Institute of Technology

Warangal

Lab 3 Assignment

Name: Ayush Rana

Roll Number: 24CSM2R05

Course Title: Data Privacy

Department: Computer Science and Engineering

Program: M. Tech in Computer Science and Information Security

Semester: 2

1. Implement any 3 Perturbation Techniques to preserve data privacy.

1. Additive Noise Perturbation

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
# Load dataset
df = pd.read_csv("adult.csv")

# Select numerical columns to perturb
numerical_columns = ["age", "hours_per_week", "capital_gain", "capital_loss"]

# Display original dataset before perturbation (first 5 rows)
print("Original Dataset (First 5 rows):")
print(df[df.columns.tolist()].head(100))
```

```
Original Dataset (First 5 rows):
    age         workclass  fnlwgt     education  education_num  \
0    39         State-gov   77516     Bachelors             13
1    50  Self-emp-not-inc   83311     Bachelors             13
2    38           Private  215646       HS-grad              9
3    53           Private  234721          11th              7
4    28           Private  338409     Bachelors             13
..  ...               ...     ...           ...            ...
95   29         Local-gov  115585  Some-college             10
96   48  Self-emp-not-inc  191277     Doctorate             16
97   37           Private  202683  Some-college             10
98   48           Private  171095    Assoc-acdm             12
99   32      Federal-gov  249409       HS-grad              9

         marital_status         occupation   relationship   race     sex  \
0        Never-married       Adm-clerical  Not-in-family  White    Male
1   Married-civ-spouse    Exec-managerial        Husband  White    Male
2             Divorced  Handlers-cleaners  Not-in-family  White    Male
3   Married-civ-spouse  Handlers-cleaners        Husband  Black    Male
4   Married-civ-spouse     Prof-specialty           Wife  Black  Female
..                 ...                ...            ...    ...     ...
95      Never-married  Handlers-cleaners  Not-in-family  White    Male
96  Married-civ-spouse     Prof-specialty        Husband  White    Male
97  Married-civ-spouse              Sales        Husband  White    Male
98            Divorced    Exec-managerial      Unmarried  White  Female
99      Never-married      Other-service      Own-child  Black    Male

    capital_gain  capital_loss  hours_per_week native_country income
0           2174             0              40  United-States  <=50K
1              0             0              13  United-States  <=50K
2              0             0              40  United-States  <=50K
3              0             0              40  United-States  <=50K
4              0             0              40           Cuba  <=50K
..           ...           ...             ...            ...    ...
95             0             0              50  United-States  <=50K
96             0          1902              60  United-States   >50K
97             0             0              48  United-States   >50K
98             0             0              40        England  <=50K
99             0             0              40  United-States  <=50K

[100 rows x 15 columns]
```

[3]:
```
# Define noise parameters (Standard deviation for each attribute)
noise_params = {
    "age": 5,
    "hours_per_week": 5,
    "capital_gain": 1000,  # Higher noise for financial data
    "capital_loss": 500
}
```

[4]:
```
# Apply noise to each selected column
for col in numerical_columns:
    df[col + "_perturbed"] = df[col] + np.random.normal(0, noise_params[col], df.shape[0])

    # Ensure values remain within reasonable bounds (e.g., no negative ages or financial values)
    df[col + "_perturbed"] = df[col + "_perturbed"].clip(lower=0)

# Extract perturbed columns and rename them to match original column names
df_perturbed = df[[col + "_perturbed" for col in numerical_columns]].copy()
df_perturbed.columns = numerical_columns  # Rename to match original column names
```

[5]:
```
# Identify affected rows where at least one value has changed
df["affected"] = (df[numerical_columns] != df_perturbed).any(axis=1)

# Extract only the affected rows
affected_rows = df[df["affected"]]
```

[6]:
```
# Display the first few affected rows with changes
print(affected_rows[numerical_columns + [col + "_perturbed" for col in numerical_columns]].head(100))
```

```
    age  hours_per_week  capital_gain  capital_loss  age_perturbed  \
0    39              40          2174             0      39.823239
1    50              13             0             0      61.816262
2    38              40             0             0      38.422463
3    53              40             0             0      56.128772
4    28              40             0             0      29.043660
..  ...             ...           ...           ...            ...
95   29              50             0             0      35.301448
96   48              60             0          1902      43.657491
97   37              48             0             0      32.461603
98   48              40             0             0      49.576797
99   32              40             0             0      24.967738

    hours_per_week_perturbed  capital_gain_perturbed  capital_loss_perturbed
0                  46.650190             1719.361055                0.000000
1                  16.157309                0.000000              440.091910
2                  33.112590                0.000000               15.872228
3                  40.499758                0.000000               80.947909
4                  31.633784                0.000000                0.000000
..                       ...                     ...                     ...
95                 53.971447             1247.073706              640.875339
96                 52.318290                0.000000             1649.529427
97                 41.882412              864.321769                0.000000
98                 39.209841                0.000000              206.205574
99                 40.975302                0.000000                0.000000

[100 rows x 8 columns]
```

']:
```
# Save the perturbed dataset
df.to_csv("adult_noisy.csv", index=False)

print("\n Additive Noise Perturbation Applied. File saved as 'adult_noisy.csv'.")
```

```
 Additive Noise Perturbation Applied. File saved as 'adult_noisy.csv'.
```

2. Randomized Response (Data Swapping)

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
# Load dataset
df = pd.read_csv("adult.csv")

# Normalize column names (fixes dashes, spaces, and case issues)
df.columns = df.columns.str.strip().str.lower().str.replace("-", "_")
```

[3]:
```
# Print available columns to verify correct names
print("Available columns in dataset:", df.columns.tolist())
```

```
Available columns in dataset: ['age', 'workclass', 'fnlwgt', 'education', 'education_num', 'marital_status', 'occupation', 'relationship', 'race', 'sex',
'capital_gain', 'capital_loss', 'hours_per_week', 'native_country', 'income']
```

[4]:
```
# Display original dataset before perturbation (first 5 rows)
print("Original Dataset (First 5 rows):")
print(df[df.columns.tolist()].head(100))
```

```
Original Dataset (First 5 rows):
    age         workclass  fnlwgt     education  education_num  \
0    39         State-gov   77516     Bachelors             13
1    50  Self-emp-not-inc   83311     Bachelors             13
2    38           Private  215646       HS-grad              9
3    53           Private  234721          11th              7
4    28           Private  338409     Bachelors             13
..  ...               ...     ...           ...            ...
95   29         Local-gov  115585  Some-college             10
96   48  Self-emp-not-inc  191277     Doctorate             16
97   37           Private  202683  Some-college             10
98   48           Private  171095    Assoc-acdm             12
99   32      Federal-gov  249409       HS-grad              9


         marital_status         occupation   relationship   race     sex  \
0        Never-married       Adm-clerical  Not-in-family  White    Male
1   Married-civ-spouse    Exec-managerial        Husband  White    Male
2             Divorced  Handlers-cleaners  Not-in-family  White    Male
3   Married-civ-spouse  Handlers-cleaners        Husband  Black    Male
4   Married-civ-spouse     Prof-specialty           Wife  Black  Female
..                 ...                ...            ...    ...     ...
95       Never-married  Handlers-cleaners  Not-in-family  White    Male
96  Married-civ-spouse     Prof-specialty        Husband  White    Male
97  Married-civ-spouse              Sales        Husband  White    Male
98            Divorced    Exec-managerial      Unmarried  White  Female
99       Never-married      Other-service      Own-child  Black    Male

    capital_gain  capital_loss  hours_per_week native_country income
0           2174             0              40  United-States  <=50K
1              0             0              13  United-States  <=50K
2              0             0              40  United-States  <=50K
3              0             0              40  United-States  <=50K
4              0             0              40           Cuba  <=50K
..           ...           ...             ...            ...    ...
95             0             0              50  United-States  <=50K
96             0          1902              60  United-States   >50K
97             0             0              48  United-States   >50K
98             0             0              40        England  <=50K
99             0             0              40  United-States  <=50K

[100 rows x 15 columns]
```

[5]:
```
# Choose categorical columns for swapping
categorical_columns = ["workclass", "education", "marital_status", "occupation"]

# Probability of swapping a value
swap_prob = 0.2  # 20% chance of swapping each categorical value
```

[6]:
```
# Function to perform randomized response (data swapping)
def swap_values(series, swap_prob):
    n = len(series)
    swap_mask = np.random.rand(n) < swap_prob  # Create a mask for swapping
    shuffled_series = series.sample(frac=1, random_state=42).reset_index(drop=True)  # Shuffle column values

    # Apply swapping based on the mask
    swapped_series = series.copy()
    swapped_series.loc[swap_mask] = shuffled_series.loc[swap_mask]

    return swapped_series
```

[7]:
```
# Apply swapping to selected categorical columns
for col in categorical_columns:
    df[col + "_swapped"] = swap_values(df[col], swap_prob)

#  Ensure index alignment for comparison
df = df.reset_index(drop=True)
```

[8]:
```
#  Create affected column correctly
df["affected"] = False
for col in categorical_columns:
    df["affected"] |= df[col] != df[col + "_swapped"]
# Extract affected rows for analysis
affected_rows = df[df["affected"]]
```

[9]:
```
# Display first few affected rows
print("\n Affected Rows (First 10 rows where values changed):")
print(affected_rows[categorical_columns + [col + "_swapped" for col in categorical_columns]].head(10))
```

```
 Affected Rows (First 10 rows where values changed):
          workclass     education      marital_status         occupation  \
2           Private       HS-grad            Divorced  Handlers-cleaners
3           Private          11th  Married-civ-spouse  Handlers-cleaners
4           Private     Bachelors  Married-civ-spouse     Prof-specialty
5           Private       Masters  Married-civ-spouse    Exec-managerial
7   Self-emp-not-inc       HS-grad  Married-civ-spouse    Exec-managerial
10          Private  Some-college  Married-civ-spouse    Exec-managerial
13          Private    Assoc-acdm       Never-married              Sales
16  Self-emp-not-inc       HS-grad       Never-married    Farming-fishing
17          Private       HS-grad       Never-married  Machine-op-inspct
18          Private          11th  Married-civ-spouse              Sales

   workclass_swapped education_swapped marital_status_swapped  \
2            Private           HS-grad               Divorced
3            Private         Bachelors          Never-married
4            Private         Bachelors     Married-civ-spouse
5            Private           Masters     Married-civ-spouse
7            Private           HS-grad          Never-married
10           Private      Some-college     Married-civ-spouse
13           Private        Assoc-acdm     Married-civ-spouse
16           Private           HS-grad          Never-married
17           Private       Prof-school          Never-married
18           Private        Assoc-acdm     Married-civ-spouse

   occupation_swapped
2     Exec-managerial
3    Handlers-cleaners
4         Craft-repair
5      Prof-specialty
7     Exec-managerial
10              Sales
13              Sales
16    Farming-fishing
17   Machine-op-inspct
18              Sales
```

[10]:
```
# Save the swapped dataset
df.to_csv("adult_swapped.csv", index=False)

print("\n Randomized Response (Data Swapping) Applied. File saved as 'adult_swapped.csv'.")
```

```
 Randomized Response (Data Swapping) Applied. File saved as 'adult_swapped.csv'.
```

3. Microaggregation

[1]:
```
#Microaggregation

import pandas as pd
import numpy as np
```

[2]:
```
# Load dataset
df = pd.read_csv("adult.csv")
```

[3]:
```
# Normalize column names (fixes dashes, spaces, and case issues)
df.columns = df.columns.str.strip().str.lower().str.replace("-", "_")

# Select numerical columns for microaggregation
numerical_columns = ["age", "hours_per_week", "capital_gain", "capital_loss"]

# Define group size for microaggregation (k-anonymity parameter)
k = 5  # Ensures each group has at least 5 records
```

[4]:
```
# Display original dataset before perturbation (first 5 rows)
print("Original Dataset (First 5 rows):")
print(df[df.columns.tolist()].head(100))
```

```
Original Dataset (First 5 rows):
    age         workclass  fnlwgt     education  education_num  \
0    39         State-gov   77516     Bachelors             13
1    50  Self-emp-not-inc   83311     Bachelors             13
2    38           Private  215646       HS-grad              9
3    53           Private  234721          11th              7
4    28           Private  338409     Bachelors             13
..  ...               ...     ...           ...            ...
95   29         Local-gov  115585  Some-college             10
96   48  Self-emp-not-inc  191277     Doctorate             16
97   37           Private  202683  Some-college             10
98   48           Private  171095    Assoc-acdm             12
99   32       Federal-gov  249409       HS-grad              9

       marital_status         occupation   relationship   race     sex  \
0       Never-married       Adm-clerical  Not-in-family  White    Male
1   Married-civ-spouse    Exec-managerial        Husband  White    Male
2            Divorced  Handlers-cleaners  Not-in-family  White    Male
3   Married-civ-spouse  Handlers-cleaners        Husband  Black    Male
4   Married-civ-spouse     Prof-specialty           Wife  Black  Female
..                 ...                ...            ...    ...     ...
95      Never-married  Handlers-cleaners  Not-in-family  White    Male
96  Married-civ-spouse     Prof-specialty        Husband  White    Male
97  Married-civ-spouse              Sales        Husband  White    Male
98           Divorced    Exec-managerial      Unmarried  White  Female
99      Never-married      Other-service      Own-child  Black    Male

    capital_gain  capital_loss  hours_per_week native_country income
0           2174             0              40  United-States  <=50K
1              0             0              13  United-States  <=50K
2              0             0              40  United-States  <=50K
3              0             0              40  United-States  <=50K
4              0             0              40           Cuba  <=50K
..           ...           ...             ...            ...    ...
95             0             0              50  United-States  <=50K
96             0          1902              60  United-States   >50K
97             0             0              48  United-States   >50K
98             0             0              40        England  <=50K
99             0             0              40  United-States  <=50K

[100 rows x 15 columns]
```

[5]:
```
# Function to apply microaggregation to a numerical column
def microaggregate(series, k):
    sorted_series = series.sort_values().reset_index()
    grouped_values = np.array(sorted_series[series.name])  # Extract values

    # Apply microaggregation (group mean)
    for i in range(0, len(grouped_values), k):
        grouped_values[i:i + k] = np.mean(grouped_values[i:i + k])

    # Assign back to original DataFrame
    series_aggregated = pd.Series(grouped_values, index=sorted_series["index"])
    return series_aggregated.sort_index()
```

[6]:
```
# Apply microaggregation to selected numerical columns
for col in numerical_columns:
    df[col + "_microaggregated"] = microaggregate(df[col], k)

#  Identify affected rows (where values changed)
df["affected"] = False
for col in numerical_columns:
    df["affected"] |= df[col] != df[col + "_microaggregated"]

# Extract affected rows
affected_rows = df[df["affected"]]
```

[7]:
```
# Display first few affected rows
print("\n🔷 Affected Rows (First 10 rows where values changed):")
print(affected_rows[numerical_columns + [col + "_microaggregated" for col in numerical_columns]].head(10))
```

🔷 Affected Rows (First 10 rows where values changed):

	age	hours_per_week	capital_gain	capital_loss	age_microaggregated	\
7	52	45	0	0	52	
8	31	50	14084	0	30	
9	42	40	5178	0	42	
12	23	30	0	0	23	
23	43	40	0	2042	43	
27	54	60	0	0	53	
29	49	40	0	0	49	
30	23	52	0	0	23	
31	20	44	0	0	19	
35	48	40	0	0	47	

	hours_per_week_microaggregated	capital_gain_microaggregated	\
7	44	0	
8	50	14084	
9	40	5088	
12	29	0	
23	40	0	
27	60	0	
29	39	0	
30	51	0	
31	44	0	
35	40	0	

	capital_loss_microaggregated
7	0
8	0
9	0
12	0
23	2034
27	0
29	0
30	0
31	0
35	0

[8]:
```
# Save the microaggregated dataset
df.to_csv("adult_microaggregated.csv", index=False)

print("\n✅ Microaggregation Applied. File saved as 'adult_microaggregated.csv'.")
```

✅ Microaggregation Applied. File saved as 'adult_microaggregated.csv'.