The Titanic dataset is a well-known dataset in the field of data science and machine learning. It contains information about passengers who were aboard the RMS Titanic, which tragically sank on its maiden voyage in April 1912. The dataset is often used for various data analysis, visualization, and machine learning tasks. Here's a more detailed explanation of the dataset:

**Features/Columns:**

* ID:A unique identifier for each passenger.Numeric or alphanumeric.
* Name:The name of the passenger.Type: String.
* Gender:The gender or sex of the passenger.Type: Categorical (e.g., male, female).
* Age:The age of the passenger.Type: Numeric.
* Class:The ticket class or socio-economic status of the passenger.Type: Categorical (e.g., 1st class, 2nd class, 3rd class).
* Embarked:The port of embarkation for the passenger.Type: Categorical (e.g., C = Cherbourg, Q = Queenstown, S = Southampton).
* Country:The country of origin or nationality of the passenger.Type: Categorical or String.
* Ticket Number:The ticket number associated with the passenger.Type: Alphanumeric.
* Fare:The fare or ticket price paid by the passenger.Type: Numeric.
* SibSp (Siblings/Spouses):The number of siblings or spouses the passenger had aboard the Titanic.Type: Numeric.
* Parch (Parents/Children):The number of parents or children the passenger had aboard the Titanic.Type: Numeric.
* Survived:The target variable indicating whether the passenger survived (1) or did not survive (0).Type: Binary (yes or no).

The dataset is commonly used for tasks such as survival prediction, classification, and regression.

It is also used for exploratory data analysis (EDA) to understand patterns and relationships within the data.

Due to its simplicity and historical context, it is often used as a beginner-friendly dataset for learning and practicing data science skills.

The Titanic dataset is frequently used as a benchmark for testing machine learning algorithms and models.

It provides an opportunity to explore factors that may have influenced the survival of passengers, such as class, gender, and age.

Given the tragic nature of the Titanic disaster, handling the dataset requires ethical considerations.

Sensitivity to the historical context and the people involved is essential when working with such datasets.

**3.1 Pseudonymisation**

Function Definition:

pseudonymize\_data: A function that takes a DataFrame (df) and a list of attributes (attributes\_to\_pseudonymize) and pseudonymizes the specified attributes using fake names generated by the Faker library.

Faker Initialization:

fake = Faker(): Creates an instance of the Faker class to generate fake data.

Pseudonymization Loop:

for attribute in attributes\_to\_pseudonymize:: Iterates over each specified attribute (in this case, 'Name').

df[attribute] = df[attribute].apply(lambda x: fake.name()): Applies the fake.name() function to each value in the specified attribute, replacing the original names with pseudonyms.

Return Pseudonymized DataFrame:

return df: Returns the DataFrame with pseudonymous values.

Example Usage:

attributes\_to\_pseudonymize = ['Name']: Specifies that the 'Name' attribute should be pseudonymized.

titanic\_df\_pseudonymized = pseudonymize\_data(titanic\_df.copy(), attributes\_to\_pseudonymize): Calls the pseudonymize\_data function on a copy of the original Titanic DataFrame, pseudonymizing the specified attribute.

**3.2 Randomisation**

Function Definition:

randomize\_data: A function that takes a DataFrame (df) and a list of attributes (attributes\_to\_randomize) and randomizes the specified attributes by replacing the original values with randomly generated strings.

Randomization Loop:

for attribute in attributes\_to\_randomize:: Iterates over each specified attribute (e.g., 'Name').

df[attribute] = df[attribute].apply(lambda x: ''.join(np.random.choice(list('abcdefghijklmnopqrstuvwxyz'), len(x)))): Applies the randomization process to each value in the specified attribute. It generates a random string of characters of the same length as the original value.

Return Randomized DataFrame:

return df: Returns the DataFrame with randomized values.

Example Usage:

attributes\_to\_randomize = ['Name']: Specifies that the 'Name' attribute should be randomized.

titanic\_df\_randomized = randomize\_data(titanic\_df.copy(), attributes\_to\_randomize): Calls the randomize\_data function on a copy of the original Titanic DataFrame, randomizing the specified attribute.

**3.3 Aggregation**

Function Definition:

aggregate\_data: A function that takes a DataFrame (df) and a list of attributes (attributes\_to\_aggregate) and aggregates the specified numerical attributes by grouping them into predefined ranges.

Aggregation Loop:

for attribute in attributes\_to\_aggregate:: Iterates over each specified numerical attribute (e.g., 'Age').

df[attribute] = pd.cut(df[attribute], bins=[0, 20, 40, 60, 80, 100], labels=['0-20', '21-40', '41-60', '61-80', '81-100']): Uses the pandas.cut() method to create bins and assign labels, thereby transforming numerical values into categorical ranges.

Return Aggregated DataFrame:

return df: Returns the DataFrame with aggregated values.

Example Usage:

attributes\_to\_aggregate = ['Age']: Specifies that the 'Age' attribute should be aggregated.

titanic\_df\_aggregated = aggregate\_data(titanic\_df.copy(), attributes\_to\_aggregate): Calls the aggregate\_data function on a copy of the original Titanic DataFrame, aggregating the specified attribute.

**3.4 Perturbation**

Function Definition:

add\_noise: A function that takes a DataFrame (df) and a list of attributes (attributes\_to\_perturb) and perturbs the specified numerical attributes by adding controlled noise.

Perturbation Loop:

for attribute in attributes\_to\_perturb:: Iterates over each specified numerical attribute (e.g., 'Age', 'Fare').

mean = df[attribute].mean(): Calculates the mean of the original values in the specified attribute.

std = df[attribute].std(): Calculates the standard deviation of the original values in the specified attribute.

noise = np.random.normal(0, std, df[attribute].shape): Generates random noise from a normal distribution with mean 0 and standard deviation equal to the original standard deviation.

df[attribute] += noise: Adds the generated noise to the original values in the specified attribute.

Return Perturbed DataFrame:

return df: Returns the DataFrame with perturbed values.

Example Usage:

attributes\_to\_perturb = ['Age', 'Fare']: Specifies that the 'Age' and 'Fare' attributes should be perturbed.

titanic\_df\_perturbed = add\_noise(titanic\_df.copy(), attributes\_to\_perturb): Calls the add\_noise function on a copy of the original Titanic DataFrame, perturbing the specified attributes.

**3.5 Data Analysis**

Explanation:

Function Definition:

analyze\_information\_loss: A function that takes two DataFrames (original\_df and transformed\_df) and analyzes the information loss by comparing their memory usages.

Memory Usage Calculation:

original\_size = original\_df.memory\_usage().sum(): Calculates the total memory usage of the original DataFrame.

transformed\_size = transformed\_df.memory\_usage().sum(): Calculates the total memory usage of the transformed DataFrame.

loss\_percentage = ((original\_size - transformed\_size) / original\_size) \* 100: Computes the percentage of information loss by comparing the memory usages.

Print Information Loss:

print(f"Information loss percentage: {information\_loss\_percentage:.2f}%"): Prints the calculated information loss percentage.

Example Usage:

information\_loss\_percentage = analyze\_information\_loss(titanic\_df, titanic\_df\_pseudonymized): Calls the analyze\_information\_loss function to analyze the information loss between the original Titanic DataFrame and the pseudonymized DataFrame.

Prints the calculated information loss percentage.

**Conclusion:**

These operations collectively demonstrate various techniques for enhancing privacy in a dataset while preserving its utility for analysis. The Titanic dataset is pseudonymized, randomized, aggregated, perturbed, and then the information loss is analyzed. Each operation introduces a different layer of privacy protection, and the information loss analysis provides insights into the impact of these transformations on the dataset.