Illinois Institute of Technology

CS528 Data Privacy and Security

Project

About the Dataset

This dataset is loan application dataset, it has information about the applicant like age, experience, profession, house information, car information, current job experience, birth state, and based on this information there is the risk flag which determines if the applicant can get a loan or not. So, ID is the primary key for this dataset which is unique, and not null. But ID can be identified with other columns. These columns are called quasi-identifiers. Quasi-identifiers in this dataset are Age, Current_Job_Years, Profession, State. For most of the applicants ID can be identified with these four columns.
The dataset has over 200 thousand rows, but for the project I have taken 67210 rows for performing the analysis

Dataset Link

https://www.kaggle.com/datasets/rohit265/loan-approval-dataset?resource=download

Dataset code to convert into csv

```
import pandas as pd
import json

# File path to the JSON file
file_path = "C:/Users/yadap/Desktop/archive/loan_approval_dataset.json"

with open(file_path, 'r') as file:
    json_data = json.load(file)

# Create a DataFrame from the JSON data
df = pd.DataFrame(json_data)

# Save the DataFrame to a CSV file
csv_file_path = 'loan_applicants.csv'
df.to_csv(csv_file_path, index=False)
```

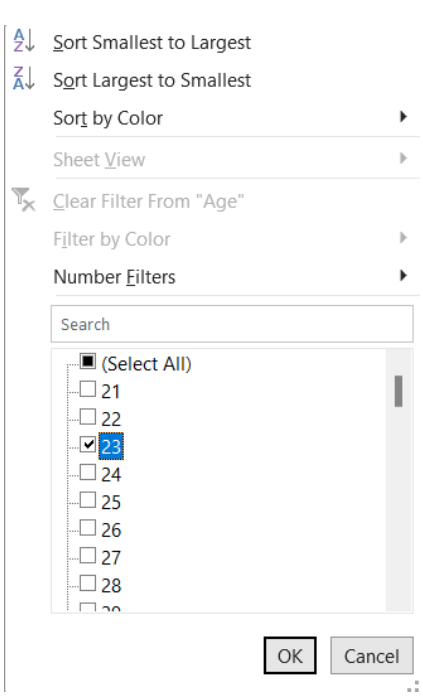
The dataset has 67210 rows

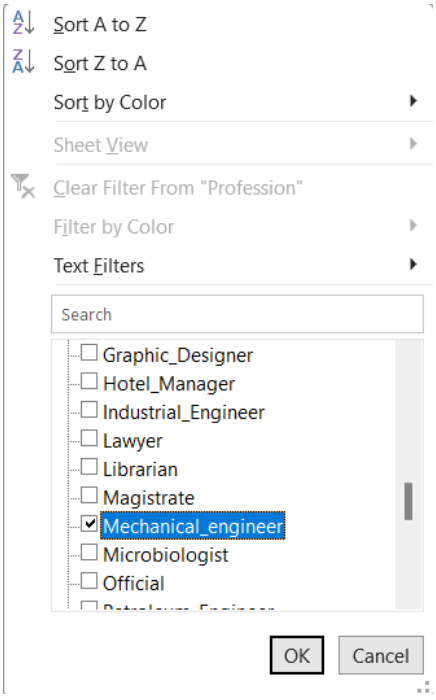
Privacy Assessment:

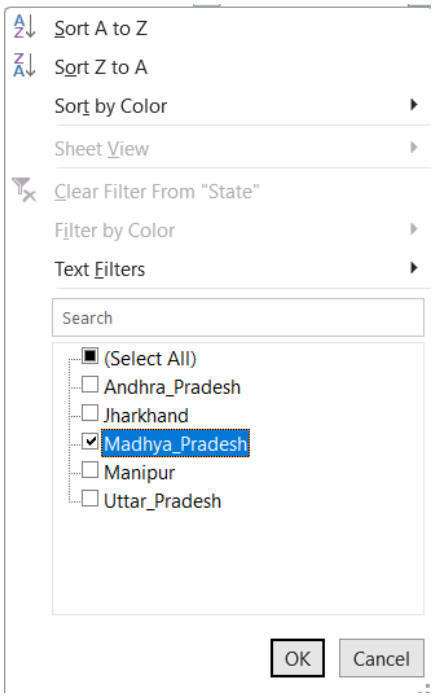
As discussed earlier the data can be identified using these four columns Age, Current_Job_Years, Profession, State and example cases shown below

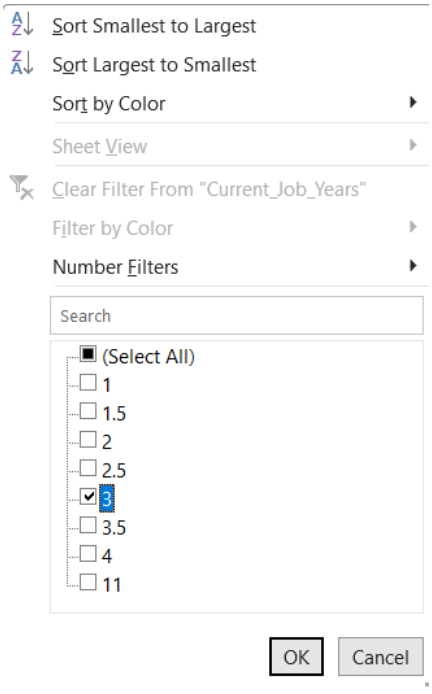
Example 1

Age: 23, Current_Job_Years: 3, Profession: Mechanical_Engineer, State: Madhya_Pradesh







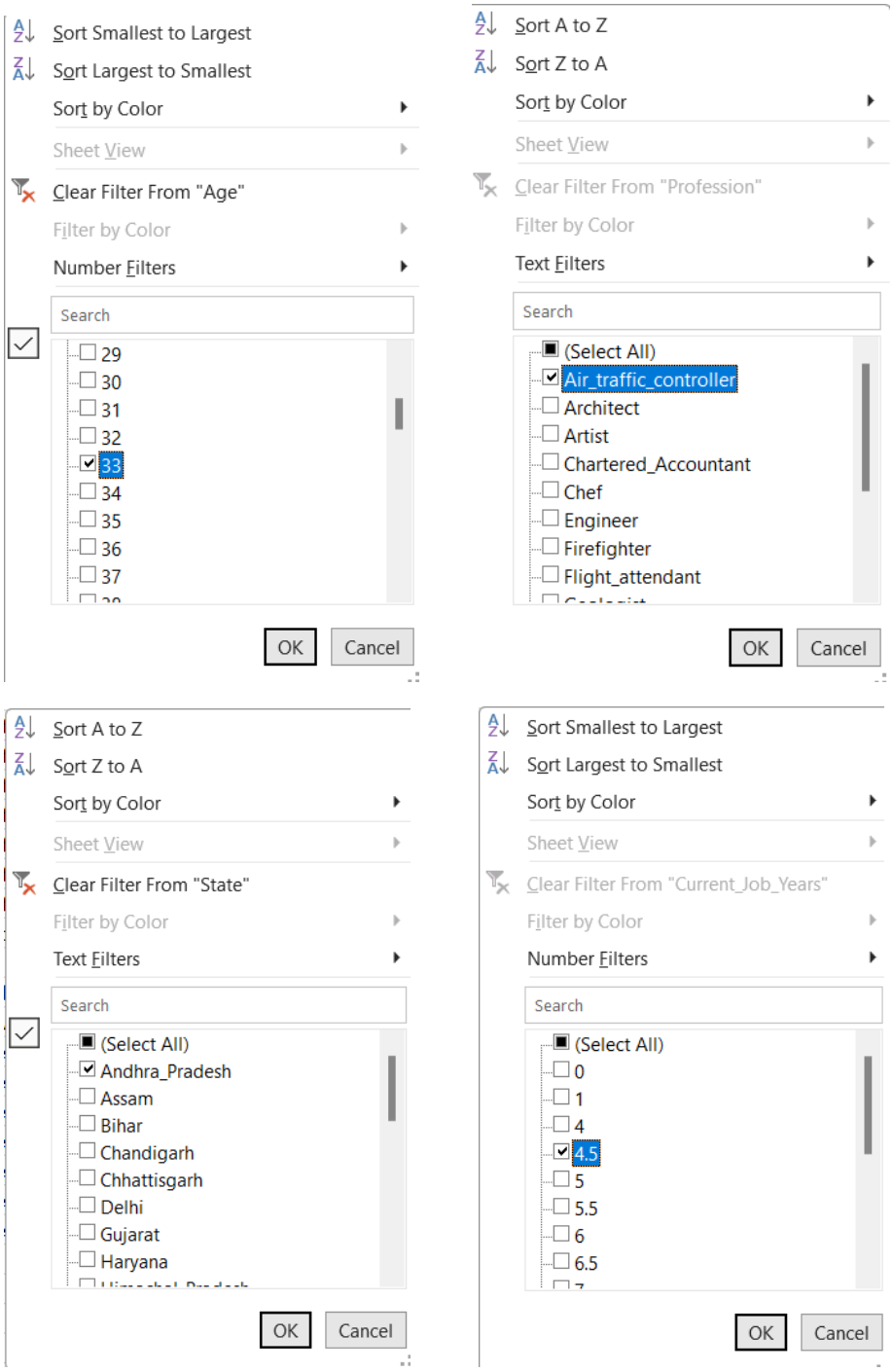


After Applying the filters

Id	Income	Age	Experien	Married/Sing	House_Ownersh	Car_Ownersh	Profession	State	Current_Job_Yea	Current_House_Yea	Risk_Fl
1	1303834	23	3	single	rented	no	Mechanical_engineer	Madhya_Pradesh	3	13	0

Example 2

Age: 33, Current_Job_Years: 4.5, Profession: Air_traffic_controller, State: Andhra_Pradesh

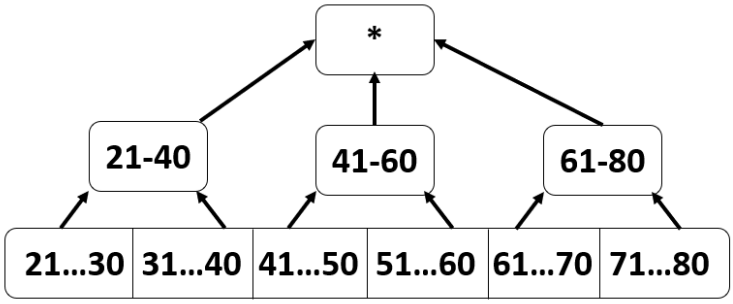


After Applying the filters

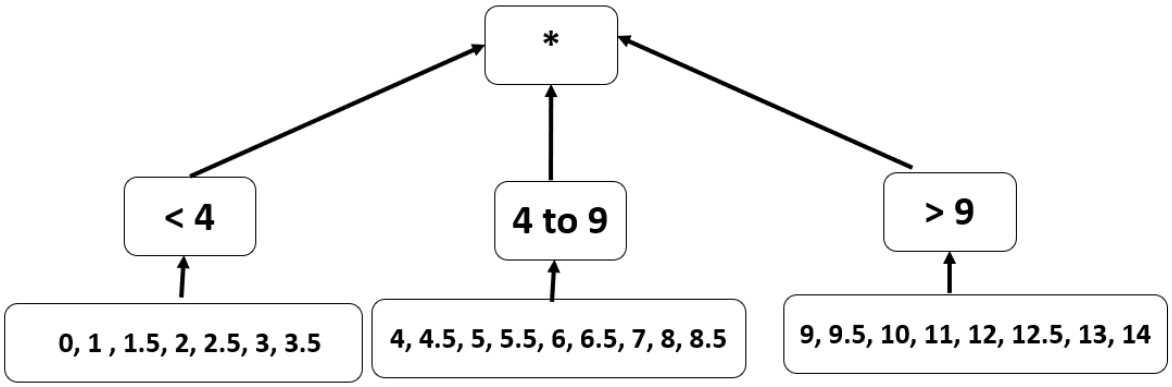
A	B	C	D	E	F	G	H	I	J	K	L
Id	Income	Age	Experien	Married/Sing	House_Ownersh	Car_Ownersh	Profession	State	Current_Job_Yea	Current_House_Yea	Risk_Fl
308	208537	33	6	married	rented	no	Air_traffic_controller	Andhra_Pradesh	4.5	12	1

GENERALIZATION HIERARCHY

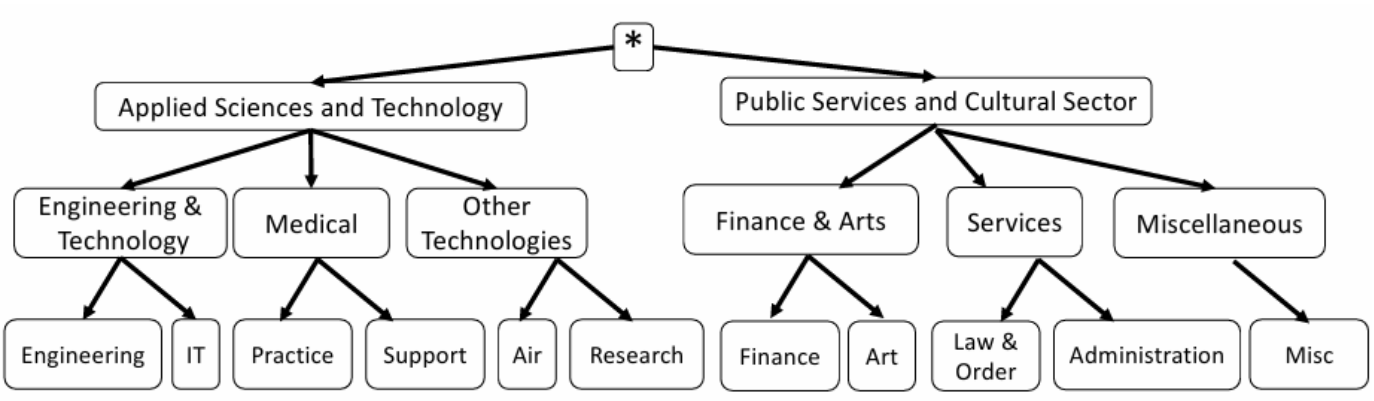
Hierarchy for Age: (Generalization)



Hierarchy for Current Job Years:



Hierarchy for Profession:



Since the profession is a wide category, the last level is shown in multiple images

ENGINEERING

'Mechanical_engineer' 'Technical_writer'
'Architect' 'Design_Engineer' 'Technician'
'Chemical_engineer' 'Engineer' 'Analyst'
'Civil_engineer' 'Industrial_Engineer'
'Technology_specialist, 'Petroleum_Engineer'

IT

'Software_Developer'
'Computer_hardware_engineer'
'Computer_operator'
'Web_designer'

MEDICAL PRACTICE

'Physician'
'Surgeon'
'Psychologist'
'Dentist'

MEDICAL SUPPORT

'Biomedical_Engineer'
'Microbiologist'

AIR

'Flight_attendant'
'Air_traffic_controller'
'Aviator'

RESEARCH

'Statistician'
'Scientist'
'Surveyor'
'Geologist'

FINANCE

'Economist'
'Financial_Analyst'
'Chartered_Accountant'

ART

'Comedian'
'Graphic_Designer'
'Artist'
'Drafter'
'Designer'

LAW & ORDER

'Police_officer'
'Magistrate'
'Lawyer'
'Consultant'

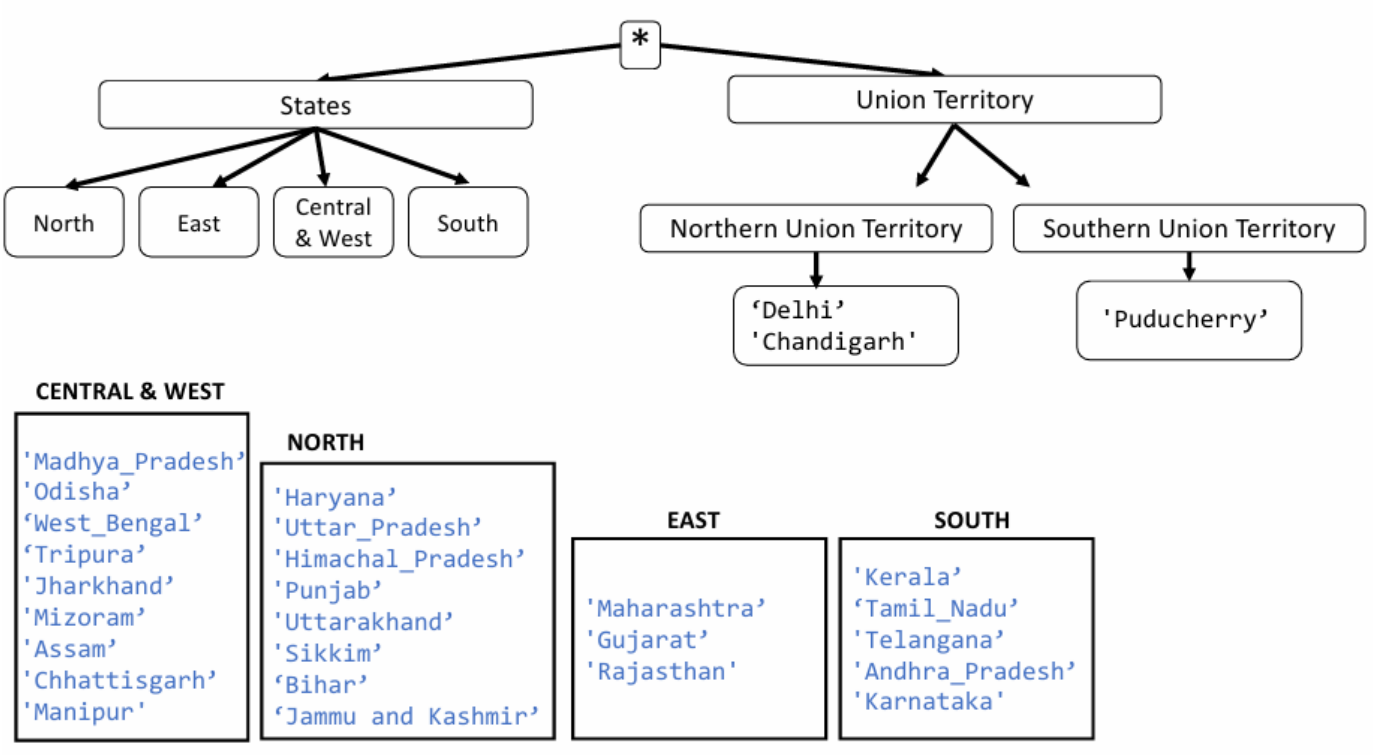
ADMINISTRATION

'Civil_servant'
'Politician'
'Secretary'
'Official'
'Army_officer'
'Firefighter'

MISC

'Fashion_Designer'
'Chef'
'Librarian'
'Hotel_Manager'

Hierarchy for States:



K - Anonymity Overview:

k-Anonymity is a privacy-preserving technique aimed at ensuring that an individual's data cannot be distinguished from at least k-1 other individuals' data within a dataset. It is a method used to protect against re-identification attacks.

Concepts:

- Quasi-Identifiers: Attributes that, when combined, can potentially identify an individual (e.g., birthdate, gender, ZIP code).
- k-Anonymity: A dataset achieves k-anonymity if each record is indistinguishable from at least k-1 other records concerning the quasi-identifiers.

Implementation Techniques:

- Generalization: Replacing specific values with broader categories.
- Suppression: Removing specific values or entire records that are hard to anonymize.

Benefits:

- Simplicity: Easy to understand and implement compared to more complex privacy models.
- Effectiveness: Provides a straightforward measure to prevent re-identification by ensuring each record is similar to at least k-1.

Major Applications:

- Healthcare: Anonymizing patient records to share data for research while protecting patient identities.
- Data Publishing: Ensuring published datasets do not allow re-identification of individuals.

Challenges:

- Data Utility: Generalization and suppression can reduce data accuracy and utility for analysis.
- Dimensionality: High-dimensional data makes achieving k-anonymity more complex and can lead to excessive data loss.

K - Anonymity

Load the adult data

```
import pandas as pd
import numpy as np

project_data = "C:/Users/yadap/Desktop/DPS/Project/dataset.csv"
dfm = pd.read_csv(project_data)
dfm.head()
```

output

	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	State	Current_Job_Years	Current_House_Years	Risk_Flag
0	1	1303834	23	3	single	rented	no	Mechanical_engineer	Madhya_Pradesh	3.0	13	0
1	2	7574516	40	10	single	rented	no	Software_Developer	Maharashtra	9.0	13	0
2	3	3991815	66	4	married	rented	no	Technical_writer	Kerala	4.0	10	0
3	4	6256451	41	2	single	rented	yes	Software_Developer	Odisha	2.0	12	1
4	5	5768871	47	11	single	rented	no	Civil_servant	Tamil_Nadu	3.0	14	1

we need only Age, Current_Job_Years, Profession, State columns from dfm

```
df = dfm[['Age', 'Current_Job_Years', 'Profession', 'State']]
df
```

output

	Age	Current_Job_Years	Profession	State
0	23	3.0	Mechanical_engineer	Madhya_Pradesh
1	40	9.0	Software_Developer	Maharashtra
2	66	4.0	Technical_writer	Kerala
3	41	2.0	Software_Developer	Odisha
4	47	3.0	Civil_servant	Tamil_Nadu
...	...	...	...	...
67205	35	3.0	Psychologist	Rajasthan
67206	63	13.0	Engineer	Bihar
67207	47	2.0	Technician	Maharashtra
67208	52	2.0	Drafter	West_Bengal
67209	60	9.0	Secretary	Maharashtra

67210 rows × 4 columns

we shall find all unique values to build hierarchies

```
# Get unique values for each column
unique_ages = df['Age'].unique()
unique_current_job_years = df['Current_Job_Years'].unique()
unique_profession = df['Profession'].unique()
unique_state = df['State'].unique()
```

```
print("Unique ages:", unique_ages)
print("\nUnique current job years:", unique_current_job_years)
print("\nUnique professions:", unique_profession)
print("\nUnique states:", unique_state)
```

output

```
Unique ages: [23 40 66 41 47 64 58 33 24 78 26 28 57 48 27 71 56 54 50 72 76 38 31 52
 59 21 55 45 77 63 30 25 73 75 74 67 46 68 36 79 42 60 32 49 65 39 35 22
 61 29 37 69 62 53 70 34 43 44 51]

Unique current job years: [ 3.   9.   4.   2.   0.   8.  11.   5.   7.   6.   4.5 12.  10.  13.
 14.   6.5  9.5  1.  12.5 10.5  8.5  5.5  3.5  1.5  2.5]

Unique professions: ['Mechanical_engineer' 'Software_Developer' 'Technical_writer'
 'Civil_servant' 'Librarian' 'Economist' 'Flight_attendant' 'Architect'
 'Air_traffic_controller' 'Physician' 'Financial_Analyst' 'Politician'
 'Police_officer' 'Surveyor' 'Design_Engineer' 'Hotel_Manager' 'Dentist'
 'Comedian' 'Biomedical_Engineer' 'Graphic_Designer'
 'Computer_hardware_engineer' 'Petroleum_Engineer' 'Secretary'
 'Computer_operator' 'Chartered_Accountant' 'Technician'
 'Chemical_engineer' 'Microbiologist' 'Fashion_Designer' 'Artist'
 'Aviator' 'Psychologist' 'Magistrate' 'Lawyer' 'Firefighter' 'Engineer'
 'Analyst' 'Geologist' 'Drafter' 'Designer' 'Statistician' 'Web_designer'
 'Consultant' 'Chef' 'Official' 'Army_officer' 'Surgeon' 'Civil_engineer'
 'Scientist' 'Industrial_Engineer' 'Technology_specialist']

Unique states: ['Madhya_Pradesh' 'Maharashtra' 'Kerala' 'Odisha' 'Tamil_Nadu' 'Gujarat'
 'Rajasthan' 'Telangana' 'Bihar' 'Andhra_Pradesh' 'West_Bengal' 'Haryana'
 'Puducherry' 'Karnataka' 'Uttar_Pradesh' 'Himachal_Pradesh' 'Punjab'
 'Tripura' 'Uttarakhand' 'Jharkhand' 'Mizoram' 'Assam' 'Jammu_and_Kashmir'
 'Delhi' 'Chhattisgarh' 'Chandigarh' 'Manipur' 'Sikkim']
```

```
len(unique_ages)
✓ 0.0s
59

len(unique_current_job_years)
✓ 0.0s
25

len(unique_profession)
✓ 0.0s
51

len(unique_state)
✓ 0.0s
28
```

```
df.head()
```

Output

	Age	Current_Job_Years	Profession	State
0	23	3.0	Mechanical_engineer	Madhya_Pradesh
1	40	9.0	Software_Developer	Maharashtra
2	66	4.0	Technical_writer	Kerala
3	41	2.0	Software_Developer	Odisha
4	47	3.0	Civil_servant	Tamil_Nadu

Maximum Hierarchy of each column

```
manageHierarchy = {'Age': 2, 'Current_Job_Years': 2, 'Profession': 4, 'State': 3}
```

Code to calculate Distortion and Precision

```
def calculateDistortion_And_Precision(quasiIdentifiers, level):
    distortion = 0
    precision = 0
    for i in range(0,len(quasiIdentifiers)):
        distortion = distortion + level[i] / manageHierarchy[quasiIdentifiers[i]]
    distortion = round(distortion / len(quasiIdentifiers), 2)
    precision = round(1 - distortion,2)
    return {'distortion': distortion, 'precision': precision}
```

Main Function to anonymize based on levels

```
def anonymizeBasedOnLevel(key, value, level):
    # ANONYMIZING AGE
    if(key == 'Age'):
        if level == 0:
            return value
        elif level == 1:
            if 21 <= value <= 40:
                return '21 to 40'
            elif 41 <= value <= 60:
                return '41 to 60'
            elif 61 <= value <= 80:
                return '61 to 80'
        elif level == 2 or level > 2:
            return '*'

    # ANONYMIZING CURRENT JOB YEARS
    if(key == 'Current_Job_Years'):
        if level == 0:
            return value
        elif level == 1:
```

```
            if 0 <= value < 4:
                return '< 4'
            elif 4 <= value < 9:
                return '4 to 9'
            elif 9 <= value:
                return '> 9'
        elif level == 2 or level > 2:
            return '*'

    # ANONYMIZING PROFESSION
    if(key == 'Profession'):
        if level == 0:
            return value
        elif level == 1:
            if value in [
'Mechanical_engineer','Technical_writer','Architect','Design_Engineer','Techni
cian','Chemical_engineer','Engineer','Analyst','Civil_engineer','Industrial_En
gineer','Technology_specialist','Petroleum_Engineer']:
                return 'Engineering'
            elif value in [ 'Software_Developer',
'Computer_hardware_engineer', 'Computer_operator', 'Web_designer']:
                return 'IT'
            elif value in ['Physician', 'Surgeon', 'Psychologist', 'Dentist']:
                return 'Medical Practice'
            elif value in [ 'Biomedical_Engineer', 'Microbiologist']:
                return 'Medical Support'
            elif value in [
'Flight_attendant','Air_traffic_controller','Aviator']:
                return 'Air'
            elif value in ['Statistician', 'Scientist', 'Surveyor',
'Geologist']:
                return 'Research'
            elif value in [ 'Economist', 'Financial_Analyst',
'Chartered_Accountant']:
                return 'Finance'
            elif value in [ 'Comedian','Graphic_Designer', 'Artist',
'Drafter', 'Designer']:
                return 'Art'
            elif value in [ 'Police_officer', 'Magistrate', 'Lawyer',
'Consultant']:
                return 'Law and Order'
            elif value in [ 'Civil_servant', 'Politician', 'Secretary',
'Official', 'Army_officer', 'Firefighter']:
                return 'Administration'
            elif value in [  'Fashion_Designer', 'Chef', 'Librarian',
'Hotel_Manager']:
                return 'Misc'
```

```
        elif level == 2:
            if value in [
'Mechanical_engineer','Technical_writer','Architect','Design_Engineer','Techni
cian','Chemical_engineer','Engineer','Analyst','Civil_engineer','Industrial_En
gineer','Technology_specialist','Petroleum_Engineer', 'Software_Developer',
            'Computer_hardware_engineer', 'Computer_operator',
'Web_designer']:
                return 'Engineering and Technology'
            elif value in ['Physician', 'Surgeon', 'Psychologist', 'Dentist',
'Biomedical_Engineer', 'Microbiologist']:
                return 'Medical'
            elif value in [
'Flight_attendant','Air_traffic_controller','Aviator', 'Statistician',
'Scientist', 'Surveyor', 'Geologist']:
                return 'Other Technologies'
            elif value in [ 'Economist', 'Financial_Analyst',
'Chartered_Accountant', 'Comedian','Graphic_Designer', 'Artist', 'Drafter',
'Designer']:
                return 'Finance and Arts'
            elif value in [ 'Police_officer', 'Magistrate', 'Lawyer',
'Consultant', 'Civil_servant', 'Politician', 'Secretary', 'Official',
'Army_officer', 'Firefighter']:
                return 'Services'
            elif value in [  'Fashion_Designer', 'Chef', 'Librarian',
'Hotel_Manager']:
                return 'Miscellaneous'


        elif level == 3:
            if value in [
'Mechanical_engineer','Technical_writer','Architect','Design_Engineer','Techni
cian','Chemical_engineer','Engineer','Analyst','Civil_engineer','Industrial_En
gineer','Technology_specialist','Petroleum_Engineer', 'Software_Developer',
'Computer_hardware_engineer', 'Computer_operator', 'Web_designer',
'Physician', 'Surgeon', 'Psychologist', 'Dentist', 'Biomedical_Engineer',
'Microbiologist', 'Flight_attendant','Air_traffic_controller','Aviator',
'Statistician', 'Scientist', 'Surveyor', 'Geologist']:
                return 'Applied Sciences and Technology'

            elif value in [ 'Economist', 'Financial_Analyst',
'Chartered_Accountant', 'Comedian','Graphic_Designer', 'Artist', 'Drafter',
'Designer', 'Police_officer', 'Magistrate', 'Lawyer', 'Consultant',
'Civil_servant', 'Politician', 'Secretary', 'Official', 'Army_officer',
'Firefighter',  'Fashion_Designer', 'Chef', 'Librarian', 'Hotel_Manager']:
                return ' Public Services and Cultural Sector'


        elif level == 4 or level > 4:
            return '*'
```

```
    # ANONYMIZING STATE
    if(key == 'State'):
        if level == 0:
            return value


        elif level == 1:
            if value in [ 'Madhya_Pradesh', 'Odisha', 'West_Bengal',
'Tripura', 'Jharkhand', 'Mizoram', 'Assam', 'Chhattisgarh', 'Manipur']:
                return 'North'
            elif value in ['Haryana', 'Uttar_Pradesh', 'Himachal_Pradesh',
'Punjab', 'Uttarakhand', 'Sikkim', 'Bihar', 'Jammu and Kashmir']:
                return 'South'
            elif value in [ 'Maharashtra', 'Gujarat', 'Rajasthan']:
                return 'East'
            elif value in [  'Kerala', 'Tamil_Nadu', 'Telangana',
'Andhra_Pradesh', 'Karnataka']:
                return 'Central and West'
            elif value in [  'Delhi', 'Chandigarh' ]:
                return 'Northern Union Territory'
            elif value in [ 'Puducherry']:
                return 'Southern Union Territory'


        elif level == 2:
            if value in [ 'Madhya_Pradesh', 'Odisha', 'West_Bengal',
'Tripura', 'Jharkhand', 'Mizoram', 'Assam', 'Chhattisgarh', 'Manipur',
'Haryana', 'Uttar_Pradesh', 'Himachal_Pradesh', 'Punjab', 'Uttarakhand',
'Sikkim', 'Bihar', 'Jammu and Kashmir', 'Maharashtra', 'Gujarat', 'Rajasthan',
'Kerala', 'Tamil_Nadu', 'Telangana', 'Andhra_Pradesh', 'Karnataka']:
                return 'States'
            elif value in [ 'Delhi', 'Chandigarh', 'Puducherry' ]:
                return 'Union Territories'


        elif level == 3 or level > 3:
            return '*'
```

Anonymization levels 1,1,2,1

```
normalization_levels = {'Age': 1, 'Current_Job_Years': 1, 'Profession': 2,
'State': 1}

def anonymize_df(df, normalization_levels):
    anonymized_df = df.copy()
    for col in df.columns:
        if col in normalization_levels:
            anonymized_df[col] = anonymized_df[col].apply(lambda x:
anonymizeBasedOnLevel(col, x, normalization_levels[col]))
    return anonymized_df

# Create the anonymized DataFrame
anonymized_df = anonymize_df(df.copy(), normalization_levels)

anonymized_df.head(20)
```

Output

	Age	Current_Job_Years	Profession	State
0	21 to 40	< 4	Engineering and Technology	North
1	21 to 40	> 9	Engineering and Technology	East
2	61 to 80	4 to 9	Engineering and Technology	Central and West
3	41 to 60	< 4	Engineering and Technology	North
4	41 to 60	< 4	Services	Central and West
5	61 to 80	< 4	Services	East
6	41 to 60	4 to 9	Miscellaneous	Central and West
7	21 to 40	< 4	Finance and Arts	East
8	21 to 40	> 9	Other Technologies	East
9	21 to 40	4 to 9	Engineering and Technology	Central and West
10	61 to 80	4 to 9	Other Technologies	South
11	21 to 40	4 to 9	Other Technologies	Central and West
12	21 to 40	> 9	Medical	Central and West
13	41 to 60	4 to 9	Finance and Arts	Central and West
14	41 to 60	4 to 9	Engineering and Technology	Central and West
15	21 to 40	4 to 9	Other Technologies	Central and West
16	61 to 80	4 to 9	Other Technologies	North
17	41 to 60	> 9	Services	East
18	21 to 40	4 to 9	Services	South
19	21 to 40	4 to 9	Other Technologies	Central and West

```
distortion_precision =
calculateDistortion_And_Precision(['Age','Current_Job_Years', 'Profession',
'State'], [1,1,2,1])
print('distortion and precision values', distortion_precision)
```

```
distortion and precision values {'distortion': 0.46, 'precision': 0.54}
```

Anonymization levels 1,1,3,2

```
normalization_levels = {'Age': 1, 'Current_Job_Years': 1, 'Profession': 3,
'State': 2}

def anonymize_df(df, normalization_levels):
    anonymized_df = df.copy()
    for col in df.columns:
        if col in normalization_levels:
            anonymized_df[col] = anonymized_df[col].apply(lambda x:
anonymizeBasedOnLevel(col, x, normalization_levels[col]))
    return anonymized_df

# Create the anonymized DataFrame
anonymized_df = anonymize_df(df.copy(), normalization_levels)

anonymized_df.head(20)
```

Output

	Age	Current_Job_Years	Profession	State
0	21 to 40	< 4	Applied Sciences and Technology	States
1	21 to 40	> 9	Applied Sciences and Technology	States
2	61 to 80	4 to 9	Applied Sciences and Technology	States
3	41 to 60	< 4	Applied Sciences and Technology	States
4	41 to 60	< 4	Public Services and Cultural Sector	States
5	61 to 80	< 4	Public Services and Cultural Sector	States
6	41 to 60	4 to 9	Public Services and Cultural Sector	States
7	21 to 40	< 4	Public Services and Cultural Sector	States
8	21 to 40	> 9	Applied Sciences and Technology	States
9	21 to 40	4 to 9	Applied Sciences and Technology	States
10	61 to 80	4 to 9	Applied Sciences and Technology	States
11	21 to 40	4 to 9	Applied Sciences and Technology	States
12	21 to 40	> 9	Applied Sciences and Technology	States
13	41 to 60	4 to 9	Public Services and Cultural Sector	States
14	41 to 60	4 to 9	Applied Sciences and Technology	States
15	21 to 40	4 to 9	Applied Sciences and Technology	States
16	61 to 80	4 to 9	Applied Sciences and Technology	States
17	41 to 60	> 9	Public Services and Cultural Sector	States
18	21 to 40	4 to 9	Public Services and Cultural Sector	States
19	21 to 40	4 to 9	Applied Sciences and Technology	States

```
distortion_precision =
calculateDistortion_And_Precision(['Age','Current_Job_Years', 'Profession',
'State'], [1,1,3,2])
print('distortion and precision values', distortion_precision)
```

```
distortion and precision values {'distortion': 0.6, 'precision': 0.4}
```

Anonymization levels 1,1,1,1

```
normalization_levels = {'Age': 1, 'Current_Job_Years': 1, 'Profession': 1,
'State': 1}

def anonymize_df(df, normalization_levels):
    anonymized_df = df.copy()
    for col in df.columns:
        if col in normalization_levels:
            anonymized_df[col] = anonymized_df[col].apply(lambda x:
anonymizeBasedOnLevel(col, x, normalization_levels[col]))
    return anonymized_df

# Create the anonymized DataFrame
anonymized_df = anonymize_df(df.copy(), normalization_levels)

anonymized_df.head(20)
```

Output

	Age	Current_Job_Years	Profession	State
0	21 to 40	< 4	Engineering	North
1	21 to 40	> 9	IT	East
2	61 to 80	4 to 9	Engineering	Central and West
3	41 to 60	< 4	IT	North
4	41 to 60	< 4	Administration	Central and West
5	61 to 80	< 4	Administration	East
6	41 to 60	4 to 9	Misc	Central and West
7	21 to 40	< 4	Finance	East
8	21 to 40	> 9	Air	East
9	21 to 40	4 to 9	Engineering	Central and West
10	61 to 80	4 to 9	Air	South
11	21 to 40	4 to 9	Air	Central and West
12	21 to 40	> 9	Medical Practice	Central and West
13	41 to 60	4 to 9	Finance	Central and West
14	41 to 60	4 to 9	Engineering	Central and West
15	21 to 40	4 to 9	Air	Central and West
16	61 to 80	4 to 9	Air	North
17	41 to 60	> 9	Administration	East
18	21 to 40	4 to 9	Law and Order	South
19	21 to 40	4 to 9	Air	Central and West

```
distortion_precision =
calculateDistortion_And_Precision(['Age','Current_Job_Years', 'Profession',
'State'], [1,1,1,1])
print('distortion and precision values', distortion_precision)
```

```
distortion and precision values {'distortion': 0.4, 'precision': 0.6}
```

```
Anonymization levels 2,2,4,3

MAX HIERARCHY
```

```
normalization_levels = {'Age': 2, 'Current_Job_Years': 2, 'Profession': 4,
'State': 3}

def anonymize_df(df, normalization_levels):
    anonymized_df = df.copy()
    for col in df.columns:
        if col in normalization_levels:
            anonymized_df[col] = anonymized_df[col].apply(lambda x:
anonymizeBasedOnLevel(col, x, normalization_levels[col]))
    return anonymized_df

# Create the anonymized DataFrame
anonymized_df = anonymize_df(df.copy(), normalization_levels)

anonymized_df.head(20)
```

Output

	Age	Current_Job_Years	Profession	State
0	*	*	*	*
1	*	*	*	*
2	*	*	*	*
3	*	*	*	*
4	*	*	*	*
5	*	*	*	*
6	*	*	*	*
7	*	*	*	*
8	*	*	*	*
9	*	*	*	*
10	*	*	*	*
11	*	*	*	*
12	*	*	*	*
13	*	*	*	*
14	*	*	*	*
15	*	*	*	*
16	*	*	*	*
17	*	*	*	*
18	*	*	*	*
19	*	*	*	*

```
distortion_precision =
calculateDistortion_And_Precision(['Age','Current_Job_Years', 'Profession',
'State'], [2,2,4,3])
print('distortion and precision values', distortion_precision)
```

```
distortion and precision values {'distortion': 1.0, 'precision': 0.0}
```

Algorithm for K Anonymity:

1. Import Libraries

- Import necessary libraries for data manipulation and analysis (e.g., `pandas`, `numpy`).

2. Load Dataset

- Specify the path to the dataset.

- Since our dataset is a JSON, we can convert it into CSV

- Load the dataset (CSV file) into a DataFrame using `pd.read_csv` function.

3. Select Required Columns

- Identify and specify the columns needed for the analysis. (here: Age, Current_Job_Years, Profession, State)

- Create a new DataFrame containing only these required columns.

4. Find Unique Values for Hierarchy

- For each required column, identify and store the unique values.

- Use these unique values to determine the hierarchy of the data.

5. Define Maximum Hierarchy Level

- Based on the unique values identified, define the maximum hierarchy level for each column.

- The maximum hierarchy level indicates the depth at which the data can be anonymized.

6. Define Functions

- Precision Calculation Function:

  - Define a function that calculates the precision of the anonymized data by comparing it with the original data.

- Distortion Calculation Function:

  - Define a function that calculates the distortion introduced by anonymization.

Distortion = (sum (current generalisation level / max generalisation level)) /
Precision = 1 – distortion

- Anonymization Function:

  - Define a function to perform the anonymization based on the level
    ex: if level = 1
    ------
    else if level = 2

    ------

7. Anonymize Data

  - Specify the desired level of anonymization. Call the anonymization function with the main DataFrame and the specified level.

  - The function should return a new DataFrame with anonymized data.

Algorithm for L diversity:

For l - diversity, we follow analogous steps to k - anonymity. After completing the initial steps, we develop a function to verify l -diversity. This function examines whether each quasi-identifier meets the l-diversity requirement. If any quasi-identifier fails to meet this criterion, we increase the level by 1 and repeat the anonymization process until achieving the desired l-diversity.

L - Diversity Overview:

l-Diversity is an extension of k-anonymity that aims to address its limitations by ensuring that sensitive attributes within each group of indistinguishable records (or equivalence class) have at least “l” "well-represented" values. This enhances privacy by protecting against attribute disclosure, where an attacker can infer sensitive information despite k-anonymity.

Concepts:

- Equivalence Class: A group of records indistinguishable from each other based on quasi-identifiers.
- Sensitive Attribute: Attributes that should be protected from inference attacks
- l-Diversity: A dataset is l-diverse if every equivalence class contains at least ‘l’ well-represented values for each sensitive attribute.

Types of l-Diversity:

- Distinct l-Diversity: Each equivalence class contains at least ‘l’ distinct values for the sensitive attribute.
- Entropy l-Diversity: Ensures diversity by measuring the entropy of sensitive attributes within each equivalence class.
- Recursive (c,l)-Diversity: Provides a more nuanced approach by considering the distribution of sensitive values, not just their presence.

Benefits:

- Enhanced Privacy: Protects against both re-identification and attribute disclosure by ensuring diversity of sensitive attributes within each equivalence class.
- Granularity: Provides a more refined privacy model compared to k-anonymity, reducing the risk of inferring sensitive information.

Applications:

- Similar as K - Anonymity

Challenges:

- Data Utility: Similar to k-anonymity, achieving l-diversity may involve significant generalization or suppression, potentially reducing data utility.
- Complexity: Implementing l-diversity can be computationally more intensive and complex, especially for high-dimensional data.

L – Diversity

Load the dataset

```
import pandas as pd
import numpy as np

project_data = "C:/Users/yadap/Desktop/DPS/Project/dataset.csv"
dfm = pd.read_csv(project_data)
dfm.head()
```

output

	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	State	Current_Job_Years	Current_House_Years	Risk_Flag
0	1	1303834	23	3	single	rented	no	Mechanical_engineer	Madhya_Pradesh	3.0	13	0
1	2	7574516	40	10	single	rented	no	Software_Developer	Maharashtra	9.0	13	0
2	3	3991815	66	4	married	rented	no	Technical_writer	Kerala	4.0	10	0
3	4	6256451	41	2	single	rented	yes	Software_Developer	Odisha	2.0	12	1
4	5	5768871	47	11	single	rented	no	Civil_servant	Tamil_Nadu	3.0	14	1

Taking only necessary columns

```
df = dfm[['Age', 'Current_Job_Years', 'Profession', 'State', 'Risk_Flag']]
df.head(10)
```

output

	Age	Current_Job_Years	Profession	State	Risk_Flag
0	23	3.0	Mechanical_engineer	Madhya_Pradesh	0
1	40	9.0	Software_Developer	Maharashtra	0
2	66	4.0	Technical_writer	Kerala	0
3	41	2.0	Software_Developer	Odisha	1
4	47	3.0	Civil_servant	Tamil_Nadu	1
5	64	0.0	Civil_servant	Maharashtra	0
6	58	8.0	Librarian	Tamil_Nadu	0
7	33	2.0	Economist	Gujarat	0
8	24	11.0	Flight_attendant	Rajasthan	0
9	23	5.0	Architect	Telangana	0

```
manageHierarchy = {'Age': 2, 'Current_Job_Years': 2, 'Profession': 4, 'State': 3}
```

```
def calculateDistortion_And_Precision(quasiIdentifiers, level):
    distortion = 0
    precision = 0
    for i in range(0,len(quasiIdentifiers)):
        distortion = distortion + level[i] / manageHierarchy[quasiIdentifiers[i]]
    distortion = round(distortion / len(quasiIdentifiers), 2)
    precision = round(1 - distortion,2)
    return {'distortion': distortion, 'precision': precision}
```

```
def anonymizeBasedOnLevel(key, value, level):
    # ANONYMIZING AGE
    if(key == 'Age'):
        if level == 0:
            return value
        elif level == 1:
            if 21 <= value <= 40:
                return '21 to 40'
            elif 41 <= value <= 60:
                return '41 to 60'
            elif 61 <= value <= 80:
                return '61 to 80'
        elif level == 2 or level > 2:
            return '*'

    # ANONYMIZING CURRENT JOB YEARS
    if(key == 'Current_Job_Years'):
        if level == 0:
            return value
        elif level == 1:
            if 0 <= value < 4:
                return '< 4'
            elif 4 <= value < 9:
                return '4 to 9'
            elif 9 <= value:
                return '> 9'
        elif level == 2 or level > 2:
            return '*'

    # ANONYMIZING PROFESSION
    if(key == 'Profession'):
        if level == 0:
            return value
        elif level == 1:
            if value in [
'Mechanical_engineer','Technical_writer','Architect','Design_Engineer','Techni
cian','Chemical_engineer','Engineer','Analyst','Civil_engineer','Industrial_En
gineer','Technology_specialist','Petroleum_Engineer']:
                return 'Engineering'
            elif value in [ 'Software_Developer',
'Computer_hardware_engineer', 'Computer_operator', 'Web_designer']:
                return 'IT'
            elif value in ['Physician', 'Surgeon', 'Psychologist', 'Dentist']:
                return 'Medical Practice'
            elif value in [ 'Biomedical_Engineer', 'Microbiologist']:
                return 'Medical Support'
```

```
            elif value in [
'Flight_attendant','Air_traffic_controller','Aviator']:
                return 'Air'
            elif value in ['Statistician', 'Scientist', 'Surveyor',
'Geologist']:
                return 'Research'
            elif value in [ 'Economist', 'Financial_Analyst',
'Chartered_Accountant']:
                return 'Finance'
            elif value in [ 'Comedian','Graphic_Designer', 'Artist',
'Drafter', 'Designer']:
                return 'Art'
            elif value in [ 'Police_officer', 'Magistrate', 'Lawyer',
'Consultant']:
                return 'Law and Order'
            elif value in [ 'Civil_servant', 'Politician', 'Secretary',
'Official', 'Army_officer', 'Firefighter']:
                return 'Administration'
            elif value in [  'Fashion_Designer', 'Chef', 'Librarian',
'Hotel_Manager']:
                return 'Misc'


        elif level == 2:
            if value in [
'Mechanical_engineer','Technical_writer','Architect','Design_Engineer','Techni
cian','Chemical_engineer','Engineer','Analyst','Civil_engineer','Industrial_En
gineer','Technology_specialist','Petroleum_Engineer', 'Software_Developer',
            'Computer_hardware_engineer', 'Computer_operator',
'Web_designer']:
                return 'Engineering and Technology'
            elif value in ['Physician', 'Surgeon', 'Psychologist', 'Dentist',
'Biomedical_Engineer', 'Microbiologist']:
                return 'Medical'
            elif value in [
'Flight_attendant','Air_traffic_controller','Aviator', 'Statistician',
'Scientist', 'Surveyor', 'Geologist']:
                return 'Other Technologies'
            elif value in [ 'Economist', 'Financial_Analyst',
'Chartered_Accountant', 'Comedian','Graphic_Designer', 'Artist', 'Drafter',
'Designer']:
                return 'Finance and Arts'
            elif value in [ 'Police_officer', 'Magistrate', 'Lawyer',
'Consultant', 'Civil_servant', 'Politician', 'Secretary', 'Official',
'Army_officer', 'Firefighter']:
                return 'Services'
            elif value in [  'Fashion_Designer', 'Chef', 'Librarian',
'Hotel_Manager']:
                return 'Miscellaneous'
```

```
        elif level == 3:
            if value in [
'Mechanical_engineer','Technical_writer','Architect','Design_Engineer','Techni
cian','Chemical_engineer','Engineer','Analyst','Civil_engineer','Industrial_En
gineer','Technology_specialist','Petroleum_Engineer', 'Software_Developer',
'Computer_hardware_engineer', 'Computer_operator', 'Web_designer',
'Physician', 'Surgeon', 'Psychologist', 'Dentist', 'Biomedical_Engineer',
'Microbiologist', 'Flight_attendant','Air_traffic_controller','Aviator',
'Statistician', 'Scientist', 'Surveyor', 'Geologist']:
                return 'Applied Sciences and Technology'

            elif value in [ 'Economist', 'Financial_Analyst',
'Chartered_Accountant', 'Comedian','Graphic_Designer', 'Artist', 'Drafter',
'Designer', 'Police_officer', 'Magistrate', 'Lawyer', 'Consultant',
'Civil_servant', 'Politician', 'Secretary', 'Official', 'Army_officer',
'Firefighter',  'Fashion_Designer', 'Chef', 'Librarian', 'Hotel_Manager']:
                return ' Public Services and Cultural Sector'


        elif level == 4 or level > 4:
            return '*'

    # ANONYMIZING STATE
    if(key == 'State'):
        if level == 0:
            return value


        elif level == 1:
            if value in [ 'Madhya_Pradesh', 'Odisha', 'West_Bengal',
'Tripura', 'Jharkhand', 'Mizoram', 'Assam', 'Chhattisgarh', 'Manipur']:
                return 'North'
            elif value in ['Haryana', 'Uttar_Pradesh', 'Himachal_Pradesh',
'Punjab', 'Uttarakhand', 'Sikkim', 'Bihar', 'Jammu and Kashmir']:
                return 'South'
            elif value in [ 'Maharashtra', 'Gujarat', 'Rajasthan']:
                return 'East'
            elif value in [  'Kerala', 'Tamil_Nadu', 'Telangana',
'Andhra_Pradesh', 'Karnataka']:
                return 'Central and West'
            elif value in [  'Delhi', 'Chandigarh' ]:
                return 'Northern Union Territory'
            elif value in [ 'Puducherry']:
                return 'Southern Union Territory'


        elif level == 2:
```

```
            if value in [ 'Madhya_Pradesh', 'Odisha', 'West_Bengal',
'Tripura', 'Jharkhand', 'Mizoram', 'Assam', 'Chhattisgarh', 'Manipur',
'Haryana', 'Uttar_Pradesh', 'Himachal_Pradesh', 'Punjab', 'Uttarakhand',
'Sikkim', 'Bihar', 'Jammu and Kashmir', 'Maharashtra', 'Gujarat', 'Rajasthan',
'Kerala', 'Tamil_Nadu', 'Telangana', 'Andhra_Pradesh', 'Karnataka']:
                return 'States'
            elif value in [ 'Delhi', 'Chandigarh', 'Puducherry' ]:
                return 'Union Territories'


        elif level == 3 or level > 3:
            return '*'
```

```
normalization_levels = {'Age': 1, 'Current_Job_Years': 1, 'Profession': 2,
'State': 1}

def anonymize_df(df, normalization_levels):
    anonymized_df = df.copy()
    for col in df.columns:
        if col in normalization_levels:
            anonymized_df[col] = anonymized_df[col].apply(lambda x:
anonymizeBasedOnLevel(col, x, normalization_levels[col]))
    return anonymized_df

# Create the anonymized DataFrame
anonymized_df = anonymize_df(df.copy(), normalization_levels)

anonymized_df.head(20)
```

output

	Age	Current_Job_Years	Profession	State	Risk_Flag
0	21 to 40	< 4	Engineering and Technology	North	0
1	21 to 40	> 9	Engineering and Technology	East	0
2	61 to 80	4 to 9	Engineering and Technology	Central and West	0
3	41 to 60	< 4	Engineering and Technology	North	1
4	41 to 60	< 4	Services	Central and West	1
5	61 to 80	< 4	Services	East	0
6	41 to 60	4 to 9	Miscellaneous	Central and West	0
7	21 to 40	< 4	Finance and Arts	East	0
8	21 to 40	> 9	Other Technologies	East	0
9	21 to 40	4 to 9	Engineering and Technology	Central and West	0
10	61 to 80	4 to 9	Other Technologies	South	0
11	21 to 40	4 to 9	Other Technologies	Central and West	0
12	21 to 40	> 9	Medical	Central and West	0
13	41 to 60	4 to 9	Finance and Arts	Central and West	0
14	41 to 60	4 to 9	Engineering and Technology	Central and West	1
15	21 to 40	4 to 9	Other Technologies	Central and West	0
16	61 to 80	4 to 9	Other Technologies	North	0
17	41 to 60	> 9	Services	East	1
18	21 to 40	4 to 9	Services	South	0
19	21 to 40	4 to 9	Other Technologies	Central and West	0

L diversity check function

```
# l-diversity implementation
def check_l_diversity(df, quasi_identifiers, sensitive_attribute, l):
    # Group by quasi-identifiers
    grouped = df.groupby(quasi_identifiers)

    # Check if each group satisfies l-diversity
    for name, group in grouped:
        if group[sensitive_attribute].nunique() < l:
            return False, group
    return True, None
```

Ensure l diversity if it is not satisfied

```
def ensure_l_diversity(df, quasi_identifiers, sensitive_attribute, l,
normalization_levels):
    while True:
        is_l_diverse, failing_group = check_l_diversity(df, quasi_identifiers,
sensitive_attribute, l)
        if is_l_diverse:
            return df
        # Increase the generalization level for the failing group
        for key in normalization_levels:
            normalization_levels[key] += 1
        df = anonymize_df(df, normalization_levels)

# Initial levels for generalization
normalization_levels = {'Age': 1, 'Current_Job_Years': 1, 'Profession': 2,
'State': 1}

# Apply k-anonymity and then ensure l-diversity
anonymized_df = anonymize_df(df.copy(), normalization_levels)
quasi_identifiers = ['Age', 'Current_Job_Years', 'Profession', 'State']
sensitive_attribute = 'Risk_Flag'
l = 1  # Desired level of l-diversity

anonymized_df = ensure_l_diversity(anonymized_df, quasi_identifiers,
sensitive_attribute, l, normalization_levels)

# Check l-diversity
is_l_diverse, failing_group = check_l_diversity(anonymized_df,
quasi_identifiers, sensitive_attribute, l)

if not is_l_diverse:
    print("The DataFrame does not satisfy l-diversity even after attempts to
generalize.")
else:
```

```
    print("The DataFrame satisfies l-diversity.")

print(anonymized_df.head(10))
```

output

```
The DataFrame satisfies l-diversity.
        Age Current_Job_Years                 Profession            State  \
0  21 to 40              < 4  Engineering and Technology            North
1  21 to 40              > 9  Engineering and Technology             East
2  61 to 80            4 to 9  Engineering and Technology  Central and West
3  41 to 60              < 4  Engineering and Technology            North
4  41 to 60              < 4                   Services  Central and West
5  61 to 80              < 4                   Services             East
6  41 to 60            4 to 9              Miscellaneous  Central and West
7  21 to 40              < 4           Finance and Arts             East
8  21 to 40              > 9         Other Technologies             East
9  21 to 40            4 to 9  Engineering and Technology  Central and West

   Risk_Flag
0          0
1          0
2          0
3          1
4          1
5          0
6          0
7          0
8          0
9          0
```

Distortion and precision

```
distortion_precision =
calculateDistortion_And_Precision(['Age','Current_Job_Years', 'Profession',
'State'], [1,1,2,1])
print('distortion and precision values', distortion_precision)
```

distortion= (sum (current generalisation level / max generalisation level)) /

number of attributes

Attribute	Current Gen Level	Max Gen Level
Age	2	1
Current_Job_Years	2	1
Profession	4	2
State	3	1

Distortion = (1/2 + 1/2 + 2/4 + 1/3) / 4 = 1.83/4 = 0.46

Precision = 1 – distortion = 1 – 0.46 = 0.54

Attribute	Current Gen Level	Max Gen Level
Age	2	1
Current_Job_Years	2	1
Profession	4	3
State	3	2

Distortion = (1/2 + 1/2 + 3/4 + 2/3) / 4 = 2.41/4 = 0.6

Precision = 1 – distortion = 1 – 0.6 = 0.4

Attribute	Current Gen Level	Max Gen Level
Age	2	1
Current_Job_Years	2	1
Profession	4	1
State	3	1

Distortion = (1/2 + 1/2 + 1/4 + 1/3) / 4 = 1.58/4 = 0.4

Precision = 1 – distortion = 1 – 0.4 = 0.6

Attribute	Current Gen Level	Max Gen Level
Age	2	2
Current_Job_Years	2	2
Profession	4	4
State	3	3

Distortion = (2/2 + 2/2 + 4/4 + 3/3) / 4 = 1.83/4 = 1

Precision = 1 – distortion = 1 – 1 = 0

Differential Privacy

Overview

Differential Privacy ensures that the output of a computation does not significantly change when any single individual's data is added or removed, protecting individual privacy within the dataset. The objective is to analyse datasets while preserving the privacy of individuals whose data is included.

Concepts:

- Epsilon (ε): Measures privacy loss. Smaller values provide stronger privacy but may reduce accuracy.
- Sensitivity: Measures how much a single individual's data can change the function's output. Lower sensitivity requires less noise.

Techniques and Algorithms:

- Laplace Mechanism: Adds noise based on the function's sensitivity and desired privacy level (ε). Suitable for real-valued functions.
- Exponential Mechanism: For non-numeric outputs, selecting outcomes based on their scores scaled by ε.
- Noisy Count and Histograms: Adds noise to count queries for private summary statistics.

Major Applications:

- Healthcare: Sharing patient data for research without compromising privacy.
- Census Data: Government agencies publish statistical data without revealing individual responses.
- Machine Learning: Training models on sensitive data while ensuring individual details are not exposed.
- Online Services: Protects user data in services like search engines and social networks.

Benefits:

- Provides mathematically provable privacy guarantees.
- Can be applied to various data types and analytical tasks.
- Allows organizations to transparently state their privacy guarantees.

Challenges:

- Balancing accuracy and privacy: Higher privacy (lower ε) means less accurate results.
- Complex implementation: Requires careful parameter tuning and understanding of data distributions.
- Scalability: Noise addition can be computationally intensive for large datasets.

Naïve Bayes Model

```
import pandas as pd

project_data = "C:/Users/yadap/Desktop/DPS/Project/dataset.csv"
df = pd.read_csv(project_data)
df.head()
```

Output

	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	State	Current_Job_Years	Current_House_Years	Risk_Flag
0	1	1303834	23	3	single	rented	no	Mechanical_engineer	Madhya_Pradesh	3.0	13	0
1	2	7574516	40	10	single	rented	no	Software_Developer	Maharashtra	9.0	13	0
2	3	3991815	66	4	married	rented	no	Technical_writer	Kerala	4.0	10	0
3	4	6256451	41	2	single	rented	yes	Software_Developer	Odisha	2.0	12	1
4	5	5768871	47	11	single	rented	no	Civil_servant	Tamil_Nadu	3.0	14	1

```
df.dtypes
```

Output

```
Id                   int64
Income               int64
Age                  int64
Experience           int64
Married/Single      object
House_Ownership     object
Car_Ownership       object
Profession          object
State               object
Current_Job_Years  float64
Current_House_Years  int64
Risk_Flag            int64
dtype: object
```

```
X = df[["Id", "Income",'Age','Experience','Married/Single',
'House_Ownership','Car_Ownership','Profession','State','Current_Job_Years','Cu
rrent_House_Years']]
y = df['Risk_Flag']
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

label_encoders = {}
for column in ['Married/Single', 'House_Ownership', 'Car_Ownership',
'Profession', 'State']:
    le = LabelEncoder()
    X[column] = le.fit_transform(X[column])
    label_encoders[column] = le

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```
# Create and train the Naive Bayes model
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)

# Make predictions
y_pred = nb_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
confusion_mat = confusion_matrix(y_test, y_pred)

print(f'Accuracy %: {round(100*accuracy,2)}')
print('Classification Report:')
print(classification_rep)
print('Confusion Matrix:')
print(confusion_mat)
```

Output

```
Accuracy %: 84.38
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.85      0.85      7243
           1       0.83      0.83      0.83      6199

    accuracy                           0.84     13442
   macro avg       0.84      0.84      0.84     13442
weighted avg       0.84      0.84      0.84     13442

Confusion Matrix:
[[6169 1074]
 [1026 5173]]
```

Differentially Private Naïve Bayes Model

```
import pandas as pd
from diffprivlib.models import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

# Load your dataset
df = pd.read_csv("C:/Users/yadap/Desktop/DPS/Project/dataset.csv")

# Define the features and target
X = df[["Id", "Income", 'Age', 'Experience', 'Married/Single',
'House_Ownership', 'Car_Ownership', 'Profession', 'State',
'Current_Job_Years', 'Current_House_Years']]
y = df['Risk_Flag']

# Encode categorical features
label_encoders = {}
for column in ['Married/Single', 'House_Ownership', 'Car_Ownership',
'Profession', 'State']:
    le = LabelEncoder()
    X[column] = le.fit_transform(X[column])
    label_encoders[column] = le

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create and train the differentially private Naive Bayes model
epsilon = 0.5
nb_model_dp = GaussianNB(epsilon=epsilon)
nb_model_dp.fit(X_train, y_train)

# Make predictions
y_pred_dp = nb_model_dp.predict(X_test)

# Evaluate the model
accuracy_dp = accuracy_score(y_test, y_pred_dp)
classification_rep_dp = classification_report(y_test, y_pred_dp)
confusion_mat_dp = confusion_matrix(y_test, y_pred_dp)

print(f'Accuracy with DP %: {round(100 * accuracy_dp, 2)}')
print('Classification Report with DP:')
print(classification_rep_dp)
print('Confusion Matrix with DP:')
print(confusion_mat_dp)
```

Output

```
Accuracy with DP %: 83.07
Classification Report with DP:
              precision    recall  f1-score   support

           0       0.85      0.83      0.84      7243
           1       0.81      0.83      0.82      6199

    accuracy                           0.83     13442
   macro avg       0.83      0.83      0.83     13442
weighted avg       0.83      0.83      0.83     13442

Confusion Matrix with DP:
[[6012 1231]
 [1045 5154]]
```

```python
import numpy as np

# Define epsilon values to test
epsilons = [0.5, 1, 2, 4]
accuracy_list = []
precision_list = []
recall_list = []

for epsilon in epsilons:
    nb_model_dp = GaussianNB(epsilon=epsilon)
    nb_model_dp.fit(X_train, y_train)
    y_pred_dp = nb_model_dp.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred_dp)
    report = classification_report(y_test, y_pred_dp, output_dict=True)

    accuracy_list.append(accuracy)
    precision_list.append(report['weighted avg']['precision'])
    recall_list.append(report['weighted avg']['recall'])
# Create a DataFrame to display the results
results_df = pd.DataFrame({
    'Epsilon': epsilons,
    'Accuracy': accuracy_list,
    'Precision': precision_list,
    'Recall': recall_list
})
print(results_df)
```

Output

	Epsilon	Accuracy	Precision	Recall
0	0.5	0.826588	0.826901	0.826588
1	1.0	0.840128	0.840322	0.840128
2	2.0	0.836185	0.836315	0.836185
3	4.0	0.841021	0.841156	0.841021

K – Means Model

```
import pandas as pd

project_data = "C:/Users/yadap/Desktop/DPS/Project/dataset.csv"
df = pd.read_csv(project_data)
df.head()
```

Output

	Id	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	State	Current_Job_Years	Current_House_Years	Risk_Flag
0	1	1303834	23	3	single	rented	no	Mechanical_engineer	Madhya_Pradesh	3.0	13	0
1	2	7574516	40	10	single	rented	no	Software_Developer	Maharashtra	9.0	13	0
2	3	3991815	66	4	married	rented	no	Technical_writer	Kerala	4.0	10	0
3	4	6256451	41	2	single	rented	yes	Software_Developer	Odisha	2.0	12	1
4	5	5768871	47	11	single	rented	no	Civil_servant	Tamil_Nadu	3.0	14	1

```
print(df.columns)
```

Output

```
Index(['Id', 'Income', 'Age', 'Experience', 'Married/Single',
       'House_Ownership', 'Car_Ownership', 'Profession', 'State',
       'Current_Job_Years', 'Current_House_Years', 'Risk_Flag'],
      dtype='object')
```

```
from sklearn.cluster import KMeans

# Define the features
X = df[["Id", "Income",'Age','Experience','Married/Single', 'House_Ownership',
'Car_Ownership', 'Profession', 'State', 'Current_Job_Years',
'Current_House_Years']]

from sklearn.preprocessing import LabelEncoder

label_encoders = {}
for column in ['Married/Single', 'House_Ownership', 'Car_Ownership',
'Profession', 'State']:
    le = LabelEncoder()
    X[column] = le.fit_transform(X[column])
    label_encoders[column] = le

# Perform KMeans clustering
k = 2  # Number of clusters
kmeans = KMeans(n_clusters=k, random_state=42)
kmeans.fit(X)

# Get cluster labels
cluster_labels = kmeans.labels_

# Add cluster labels to the dataframe
df['Cluster_Labels'] = cluster_labels

# Explore cluster centers
cluster_centers = kmeans.cluster_centers_
```

```
# Print cluster centers
print("Cluster Centers:")
print(cluster_centers)

# Explore cluster sizes
unique, counts = np.unique(cluster_labels, return_counts=True)
cluster_sizes = dict(zip(unique, counts))

# Print cluster sizes
print("Cluster Sizes:")
print(cluster_sizes)
```

Output:

```
Cluster Centers:
[[3.28502096e+04 7.50113209e+06 4.94986333e+01 9.80019609e+00
  9.08164478e-01 1.91181888e+00 2.95026442e-01 2.51961495e+01
  1.36495335e+01 6.25025254e+00 1.19879672e+01]
 [3.30276947e+04 2.47774026e+06 4.97672270e+01 9.91875298e+00
  9.01138531e-01 1.89011087e+00 3.06092036e-01 2.53154208e+01
  1.37052635e+01 6.25584168e+00 1.19881676e+01]]
Cluster Sizes:
{0: 33694, 1: 33516}
```

```
from sklearn.metrics import silhouette_score

# Compute silhouette score
silhouette_avg = silhouette_score(X, cluster_labels)
print("Silhouette Score:", silhouette_avg)
```

Output:

```
Silhouette Score: 0.6300726192670372
```

Differentially Private K – Means Model

```
import pandas as pd
from diffprivlib.models import KMeans
import matplotlib.pyplot as plt
import numpy as np

# Load the dataset
df = pd.read_csv("C:/Users/yadap/Desktop/DPS/Project/dataset.csv")

# Define the features
X = df[["Id", "Income",'Age','Experience','Married/Single', 'House_Ownership',
'Car_Ownership', 'Profession', 'State', 'Current_Job_Years',
'Current_House_Years']]

from sklearn.preprocessing import LabelEncoder

label_encoders = {}
for column in ['Married/Single', 'House_Ownership', 'Car_Ownership',
'Profession', 'State']:
    le = LabelEncoder()
    X[column] = le.fit_transform(X[column])
    label_encoders[column] = le

# Apply differentially private k-Means
k = 2 # Number of clusters
epsilon = 1.0  # Privacy budget
kmeans = KMeans(n_clusters=k, epsilon=epsilon, random_state=42)
kmeans.fit(X)

cluster_labels = kmeans.labels_
df['Cluster_Labels'] = cluster_labels

# Explore cluster centers
cluster_centers = kmeans.cluster_centers_

# Print cluster centers
print("Cluster Centers:")
print(cluster_centers)

unique, counts = np.unique(cluster_labels, return_counts=True)
cluster_sizes = dict(zip(unique, counts))

# Print cluster sizes
print("Cluster Sizes:")
print(cluster_sizes)
```

Output

```
Cluster Centers:
[[3.28113020e+04 7.58753510e+06 4.96026661e+01 9.59413532e+00
  9.08273385e-01 1.91263899e+00 2.96874089e-01 2.51362980e+01
  1.36394149e+01 6.24321339e+00 1.19931772e+01]
 [3.32270943e+04 2.58431747e+06 5.01991949e+01 9.92972186e+00
  9.04556154e-01 1.90558814e+00 3.04446769e-01 2.52563710e+01
  1.37728661e+01 6.30332550e+00 1.20230192e+01]]
Cluster Sizes:
{0: 33096, 1: 34114}
```

```
from sklearn.metrics import silhouette_score

silhouette_avg = silhouette_score(X, cluster_labels)
print("Silhouette Score:", silhouette_avg)
```

Output

```
Silhouette Score: 0.6297490249214073
```

```
epsilons = [0.5, 1, 2, 4]
silhouette_scores = []

for epsilon in epsilons:
    # Apply differentially private k-Means
    k = 2  # Number of clusters
    dp_kmeans = KMeans(n_clusters=k, epsilon=epsilon, random_state=42)
    dp_kmeans.fit(X)

    cluster_labels = dp_kmeans.labels_
    score = silhouette_score(X, cluster_labels)
    silhouette_scores.append(score)
    print(f"Silhouette Score for epsilon={epsilon}: {score}")

# Tabulate silhouette scores
results = pd.DataFrame({'Epsilon': epsilons, 'Silhouette Score':
silhouette_scores})
print("\nSilhouette Scores for different values of epsilon:")
print(results)
```

Output

```
Silhouette Scores for different values of epsilon:
   Epsilon  Silhouette Score
0      0.5          0.628983
1      1.0          0.629749
2      2.0          0.629959
3      4.0          0.629996
```

Future scope:

- Enhanced Models: Combining k-anonymity and l – diversity with other techniques like t-closeness to address its limitations.
- Developing new mechanisms for better privacy-utility trade-offs.
- Standardizing differential privacy implementations across industries.
- Enhanced integration with AI for privacy-preserving applications.