RV College of Engineering®

Go, change the world

Autonomous institution affiliated to Visvesvaraya Technological University, Belagavi) | Approved by AICTE, New Delhi,

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Experiential Learning Report

Submitted by,

Student name	USN
Hardik Hiraman Pawar	1RV21CS046(17)
Tanmay S Lal	1RV21CS176(12)
S Mohammed Ashiq	1RV21CS132(28)
Mohammed Raza	1RV21CS093(29)

Under the guidance of

Ms. Veena Gadad

Assistant Professor

Department of Computer Science and Engineering

RV College of Engineering

In partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING
in

COMPUTER SCIENCE AND ENGINEERING
2021-2022

ABSTRACT

Privacy is the state of being free from public scrutiny or from having your secrets or personal information shared. Data protection is important, since it prevents the information of an organisation from fraudulent activities, hacking, phishing, and identity theft. It protects valuable information such as business transactions and financial statements.

The topic deals with anonymizing the given microdata table based on the anonymity model to preserve its privacy. The output of our project contains various anonymity techniques like k-anonymity , l- diversity , anatomy ,l-e diversity etc. The issue with the following papers was that it was either lacking the multiple sensitive attribute or loss of information. The flow of our work is as follows. Firstly we incorporated a method to mask the quasi identifier group and then check if there are more than one sensitive attribute occurring more than once in the given equivalence table(sub division of microdata table) as well as generalizing the quasi identifier group . The anotomy model is then incorporated due to the drawback of generalizing the quasi identifier groups. This model strengthened the privacy of data by dividing the main table into quasi identifier and sensitive attribute table where only the counts of frequencies of the sensitive value is presented instead of presenting the sensitive attribute as such. Later the sensitive attributes are classified as primary, secondary and tertiary sensitive attributes and then check for the diversification of the table and calculating the degree of diversity using appropriate formula.

The success of our work is that it outstands other researches is that for the maximum number of records the amount of information loss is very less . The respective graphs and results are attached to our report which describes the experimental analysis and success of our work.

ACRONYMS

- **QIT – QUASI IDENTIFIER TABLE**
- **ST – SENSITIVE TABLE**
- **MGID: MAXIMUM GROUP ID**
- **NMA: NON MASKED ATTRIBUTE**

TABLE OF CONTENTS

Page No

1.INTRODUCTION

Privacy is the state of being free from public attention. Privacy is of vital importance in day-to-day dealings with human beings as members of the society. This also covers confidentiality of activities like telephone calls, personal correspondence, financial information, medical histories and even emails.

In the context of privacy, there is great concern that the advancement in technology will have a damaging effect on it. Evidence shows that with the current advancement in technology, especially the internet brings speedy diminution in privacy. This can be seen in the ease in stealing of personal information, increase in cyber-crimes, all taking place over the internet.

Anonymization of published microdata has become a very important topic nowadays. The major difficulty is to publish data of individuals in a manner that the released table both provides enough information to the public and prevents disclosure of sensitive information.

Therefore, several authors proposed definitions of privacy to get anonymous microdata.
One definition is called k-Anonymity and states that every individual in one generalized block is indistinguishable from at least k - 1 other individuals. l -Diversity uses a stronger privacy definition and claims that every generalized block has to contain at least ` different sensitive values.

Anonymization plays an important role in privacy preserving data publishing. In this field, a sensitive data should not be revealed or linked to the identifier data. Privacy preserving data publishing covered this area since the appearance of k-anonymity model. K-anonymity allowed to publish microdata table by generalizing or suppressing quasi-identifier attributes. Many models in anonymizing microdata are developed such as l - diversity, p-sensitive, anatomy and their variants.

In this paper, we develop an efficient way for preserving data containing multiple sensitive attributes.

2. LITERATURE SURVEY

The previous researches in the process of publishing the privacy preserved data is mainly focused on single sensitive attribute and which cannot be implemented for the multiple sensitive attributes. The technique of anonymization was found to be more powerful to preserve the privacy of the data. In the process of anonymization **Latanya Sweeney** describes preserving of data based on the single sensitive attribute. The model of k-anonymity was based on partitioning the given microdata table into subdivisions which are called equivalence classes, in the manner such that each equivalence class of the microdata table should contain at least k number of records which cannot be distinguishable from the other set of records. This model was able to preserve the specific information pertaining to a person from the attacker. But the drawback of this model is that this model restricts the discussion to single sensitive attribute and the private table released in k-anonymity subjects to homogenous attack(for example :There are two rows of data pertaining to two individuals having same quasi-identifier(age, sex, etc.)then the probability of getting the data of an individual becomes easy for an attacker as the probability is 50%in each case) as well as background attack(attacking the sensitive information based on the other attributes such as age, sex ,code etc.).

Junjie Jia and Luting Chen describes the method of anonymizing the data based on multiple sensitive attributes. l-m-d anonymity deals with method of anonymizing the microdata table of multiple sensitive attributes by grouping the microdata table into equivalence classes and generalizing (for example the age of one person is 25 then the age is generalized as[20-30] so that the probability of knowing the age would be less)and masking(the process of protecting the quasi-identifiers with an asterisk(*)mark).For a given equivalence class ,the l-m-d anonymity looks for the sensitive attribute having different parent in the hierarchy tree. For example in one of the equivalence classes the individual might be prone to disease, pneumonia and other individual belonging to the same equivalence class might be prone to dyspepsia . Here in both the cases pneumonia belong to the parent respiratory disorder and dyspepsia belongs to the parent of stomach illness. l-m-d looks for the dissimilar parent because if the individuals belonging to the same parent present in the same equivalence class the probability of fetching the sensitive attribute of an individual becomes easy. If the attacker obtained the age and pin code(quasi-identifier) of an individual but he cannot get the sensitive attribute of a person as the sensitive groups are not similar. This method reduced the similarity attack on the records of the microdata. The drawback of l-m-d anonymity is that it takes into account the process of generalizing the quasi-identifier. Even though the generalization is an effective method of anonymization but the capability of holding large amount of data characteristics is low.

According to Widodo Murien Nugraheni and Irma Permata Sari the efficient way to preserve the privacy of the data and it is considered as one of the strongest anonymizing techniques. The concept of anatomy was effective over generalization because if the analyst is searching for a particular individual of age 30, but the age group is generalized, say [25-35]. It’s hard for an analyst to search for a particular individual. So, anatomy deals with the problem of generalization by dividing the microdata table into the two tables namely quasi-identifier table and the sensitive attribute table. In the quasi-identifier table, the quasi-identifier (age, sex, etc.) pertaining to an individual is displayed along with their group id. The sensitive attribute table consists of the group id, sensitive value or attribute (for example disease) and the count of the sensitive value (for example in the grouped sensitive attribute table, if the disease of same kind is repeating in same group id, then the count is increased to 1). The prominence of anatomy model is that it separates quasi-identifier attributes and sensitive attribute leading to the loss of less amount of considerable information. The process of anatomy was strong and effective but there was a need for diversifying data.

Widodo and A Wahyudin describes the process of converting the data into more diversified form, it is important to identify the primary and secondary sensitive attribute. There is a set of method to be followed to identify the same. The first step is to look for the sensitive attribute group for higher number of sensitive values (it is defined by the user). The sensitive attribute containing higher number of sensitive values (values with high sensitivity) is considered as primary sensitive attribute. If in case the sensitive values are same for both attribute then the one with more unique value is considered as primary sensitive attribute. For example, let us consider three sensitive attributes disease, occupation and education. Suppose the user defines that the number of sensitive values in disease attribute is 2, occupation attribute is 3 and education attribute is 4. The group with higher sensitivity, which means one with higher sensitive value is considered as primary sensitive attribute (Here, the education attribute contains higher value of sensitivity, therefore it is considered as primary sensitive attribute. Consider the second case where both disease and occupation has same sensitive values as 3 and in that case the sensitive attribute with a greater number of unique values is to be considered. (If the disease attribute has pneumonia, flu, dyspepsia as three values and occupation attribute have manager, doctor, doctor as three values, the unique values in disease attribute is 3 and in occupation attribute is 2. Here disease is considered as the primary sensitive attribute).

Michael Kern describes the method of calculating the diversity index for the anonymized table. The degree of diversity is an indication of how much the table is diversified. The degree of diversity is calculated using the formula (*No. of unique records/total no. of records*). For example, in the sensitive attribute table containing occupation as an attribute, if the count of people working as government employee is 1 and people working as the private employee is 2, the percentage of diversification comes out to be 66.66%. This is because the unique records here are only two government employee and private employee and the number of records is 3. If (*No. of unique records==total no. of records)* then the percentage of diversification will be 100% and the table is said to be diversified completely and considered to be strong method of preserving data.

This aim of this paper is to achieve most efficient way of preserving the privacy of the data. This paper includes the algorithms from the above-mentioned research works. This paper describes the algorithm to make the table of records more diversified in such a way that for the greater number of records the residue percentage should be less and the loss of information is comparatively small. This paper gives the graphical interpretation of various performance parameters such as *residue percentage vs no. of records, time(milliseconds) vs no. of records, residue percentage vs K and time(millisecond) vs K.* This protects the data from the attacker to steal the sensitive information pertaining to an individual.
The rest of this paper follows the methodology of obtaining the privacy preserved data with the set of algorithms followed by the analysis of the graph obtained through the experimental analysis, results and discussion and finally we present the conclusion drawn from this research.

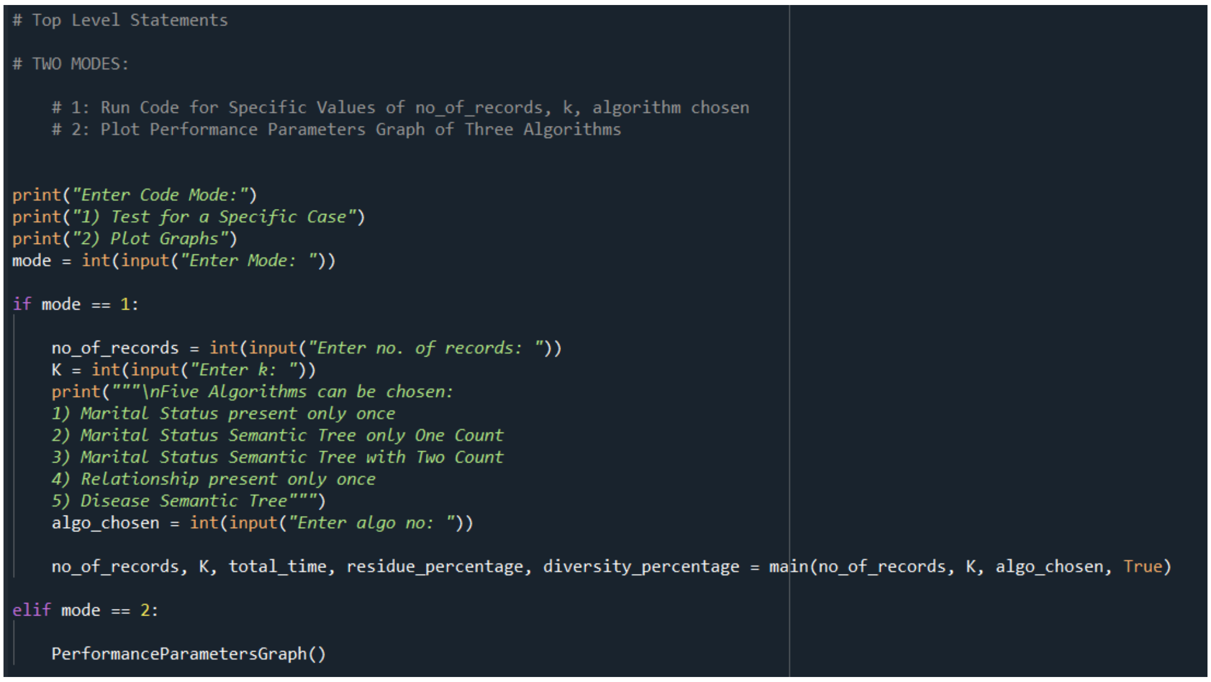
3. MOTIVATION

- Protecting privacy is key to ensuring human dignity.
- It gives safety and saves self – determination.
- It allows individuals freely develop their personality.
- Data with multiple sensitive attributes is safeguarded, which is important while publishing papers.

4. OBJECTIVE

While publishing data, utmost care should be taken that the data should be published in such a manner that it ensures privacy, but here a problem arises that if the data is published in highly hidden way, the researchers who need data for doing their research cannot get much information from this data as some of its contents would be hidden. So, our aim is to publish data by keeping in balance the privacy and its usefulness for researchers in their research by analysation of the data.

5.METHODOLOGY



```python
# Top Level Statements

# TWO MODES:

    # 1: Run Code for Specific Values of no_of_records, k, algorithm chosen
    # 2: Plot Performance Parameters Graph of Three Algorithms


print("Enter Code Mode:")
print("1) Test for a Specific Case")
print("2) Plot Graphs")
mode = int(input("Enter Mode: "))

if mode == 1:

    no_of_records = int(input("Enter no. of records: "))
    K = int(input("Enter k: "))
    print("""\nFive Algorithms can be chosen:
    1) Marital Status present only once
    2) Marital Status Semantic Tree only One Count
    3) Marital Status Semantic Tree with Two Count
    4) Relationship present only once
    5) Disease Semantic Tree""")
    algo_chosen = int(input("Enter algo no: "))

    no_of_records, K, total_time, residue_percentage, diversity_percentage = main(no_of_records, K, algo_chosen, True)

elif mode == 2:

    PerformanceParametersGraph()
```

At the start of the program execution, a choice is provided as to whether run the program for a particular set of: records, K value and algorithm or plot the graphs for the various performance parameters.

The former choice produces the resultant diversified output table based on the following steps:

I. Firstly, the microdata table is imported in a standardized manner and stored with the implementation of a nested dictionary. The keys of the outer dictionary contain the record number which is a unique number for each record and the values of the outer dictionary contain another dictionary with keys as attribute names and values as the actual data stored in the attributes for the particular record. The algorithm designed runs on the basis of native python datatypes viz., dictionaries, lists, tuples, and so on and so forth. The use of external libraries such as pandas and numpy is avoided as it increases the time complexity of the algorithm. The use of default python datatypes significantly boosts the time performance of the code thereby traversing through the dictionaries and iterating through the loops much faster. The usage of copy.deepcopy() method allows for making deep copies of the dictionaries in the program to prevent any accidental modification in the data.

The group id or more specifically the equivalence class number of a particular record is calculated with the formula given below.

$$Group\ ID = \lfloor \mathrm{n} - 1 \rfloor + 1, \qquad \mathrm{n} \in [1, \mathrm{N}]$$

$$where, \qquad \lfloor\ \rfloor = rounding\ down\ function$$

$$n \in \text{record numbers uniquely associated with the correspondig records}$$

$$N = Total\ no.of\ records$$

II. The next step is to perform the diversification of the records. Five different algorithms are compared in the program which lead to a variety of intricate results which will be covered in the Results section of the paper. The general methodology employed for diversification which is common to all five algorithms is the following:

1. The creation of a separate dictionary (called the temporary dictionary) is created to store the sensitive values that do not fit in the current nested dictionary of the microdata table. The separated dictionary will then later be used to add the separated values back into the dictionary according to where they fit based on certain conditions.

2. Iterate through each record in the temporary dictionary to pick and place records from the temporary dictionary into the new microdata dictionary based on a particular set of parameters along with the extra condition that the size of the existing equivalence class should be strictly less than k. This is because as previously mentioned, k anonymity states the present the presence of at most k records every equivalence class.

3. This completes the first stage of the diversification of the microdata table for the Primary Sensitive Attribute. A few residual records still remain in the temporary dictionary, but these are not completely unusable.

4. For the Secondary, Tertiary and Quaternary Sensitive Attributes, the diversification of the records is increased by utilising the residual records in the temporary dictionary. This is done by swapping the values of the Secondary, Tertiary and Quaternary Sensitive Attributes where are repeating in a particular equivalence class which makes a particular sensitive value appear less frequently in a given group of records for a particular equivalence class.

The five algorithms on the basis of which records are removed from the microdata table & added to the temporary dictionary and then again added back to the modified microdata table from the temporary dictionary are:

1. Based on the unique appearance primary sensitive attribute with few parents in the semantic hierarchical tree (Eg: Marital Status)

This algorithm ensures that in a particular equivalence class, a value of the chosen primary sensitive attribute appears only once in the group, i.e., it is unique for a specific equivalence class. Example: The value “Divorced” for the primary sensitive attribute “Marital Status” should appear only once in a chosen equivalence class.

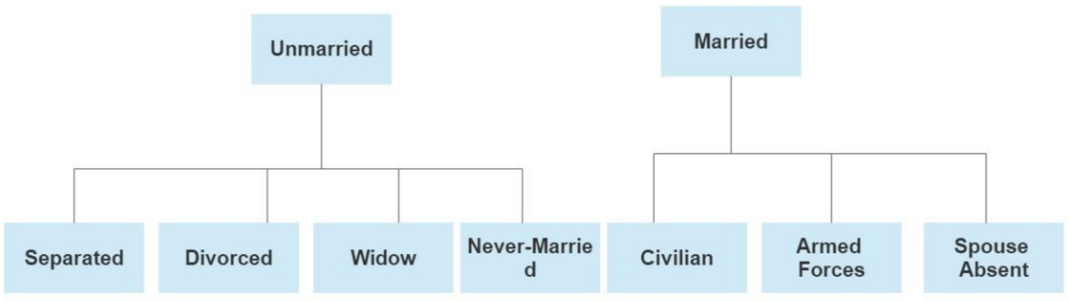
2. Based on the unique appearance of a parent in the semantic hierarchical tree consisting of a fewer number of parents for the primary sensitive attribute

It is similar to the former algorithm with the key difference being that instead of the value, the parent of the chosen primary sensitive attribute appears only once in the group, i.e., it is unique for a specific equivalence class. Example: If a record in an equivalence class has the value of “Widow” for the primary sensitive attribute “Marital Status”, the parent of “Widow” being “Unmarried”, no other record containing Marital Status value as Separated, Divorced or Never-Married is allowed to be added into the equivalence class. This algorithm poses the restriction that at most only two records can be added into an equivalence class leading to the yield of a high number of residue records (those records that cannot be added in any of the equivalence classes).

3. Based on the appearance of a parent in the semantic hierarchical tree consisting of a fewer number of parents for the primary sensitive attribute at most twice

To counter the undesirable effect produced the aforementioned algorithm, the parent of the chosen primary sensitive attribute is allowed to appear twice in a particular equivalence class. This leads to a great reduction in the number of residue records produced but it comes at the cost of loss in diversity.

According to the preferences, the trade-off between diversity and residue records percentage can be adjusted and the suitable algorithm can be chosen.



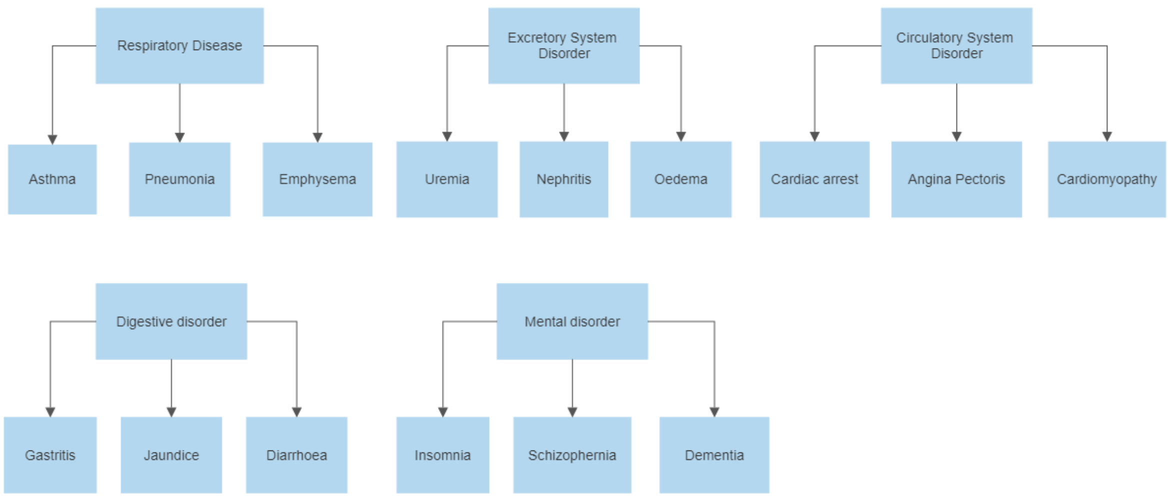
Semantic Hierarchical Tree for Marital Status

4. Based on the unique appearance primary sensitive attribute with more varying frequency of each value (Eg: Relationship)

It is an offshoot of the first algorithm where only the primary sensitive attribute is changed from one having fewer parents in the semantic hierarchical tree (Marital Status) to the attribute having more variations in its values (Relationship). This algorithm ensures that in a particular equivalence class, a value of the chosen primary sensitive attribute appears only once in the group, i.e., it is unique for a specific equivalence class. Example: The value "Son" for the primary sensitive attribute "Relationship" should appear only once in a chosen equivalence class.

5. Based on the unique appearance of a parent in the semantic hierarchical tree consisting of more number of parents for the primary sensitive attribute

It is related to the second algorithm where instead of choosing a primary sensitive attribute with lesser no. of parents in the semantic hierarchical tree (Marital Status), a primary sensitive attribute with a large no. of parents is chosen (Disease). For example, if in an equivalence class there is a presence of a disease, let's say Cardiomyopathy belong to the Circulatory System Disorder (Parent), then no other record with the disease belonging to the same parent (Circulatory System Disorder) will be allowed to be placed in that equivalence class. There is an increased no. of parents for the Disease attribute compared to the Marital Status attribute, leading to an improved performance in diversity as well as privacy preservation.



Semantic Hierarchical Tree for Disease

III. The third step is to segregate the modified microdata table into Quasi-Identifier Table (QIT) and Sensitive Table (ST). The ST Table displays the sensitive attribute that is removed from the modified table along with the Group ID and the no. of times it occurs in the equivalence class that it belongs to. The QIT table displays the non-sensitive attributes along with the record Group IDs.

IV. In the fourth step, masking and generalisation of the attributes is performed. The feature of the algorithm proposed in the paper is that the attributes which the user wants to mask can be selected and the appropriate masking for the chosen attribute can be performed with ease and efficiency. Given below are the formulas undertaken to perform the masking of certain sensitive attributes.

For generalising Age:

$$Lower\ Limit\ (L) = age - age\%10$$

$$Upper\ Limit\ (U) = L + 10 * ((group\_id - 1)\%3 + 1) - 1$$

$$Where, \quad \%\ represents\ the\ modulo\ operation$$

$$group\_id\ corresponds\ to\ the\ Group\ ID\ of\ a\ particular\ record\ in\ the\ equivalence\ class$$

$$The\ age\ is\ returned\ as\ the\ string\ \text{“}(L - U)\text{”}$$

This a type of cyclic masking where the Group IDs of 1, 2 & 3 correspond to the generalisation of the age in the ranges of 10, 20 & 30 respectively. The group IDs 4, 5, 6 follow the same manner of generalisation as that of Group IDs 1, 2, 3.

For generalising Gender: “M/F” is returned

For masking Zip Code: A simple algorithm is incorporated where the last three digits are hidden with the asterisk symbol (*) and the rest of the digits remain visible.

For masking Employment, Race or Salary: Asterisk symbol (*) is used.

V. The various performance parameters to evaluate and analyse the algorithm for varying values of the inputs (no. of records and k) is calculated with the following formulas:

$$Code\ Runtime\ =\ (end\_time - start\_time)\ x\ 1000\ rounded\ to\ 4\ places\ of\ decimal$$

$$Where, \quad start_{time} and\ end_{time} are\ initialised\ at\ the\ beginning\ and$$

$$end\ of\ the\ program\ respectively\ with\ the$$

$$function\ time.time()\ which\ returns\ the\ number\ of\ seconds\ passed\ since\ epoch.$$

Residue Percentage =

$$\frac{No.of\ records\ in\ Residue\ Dictionary}{Total\ no.of\ records\ in\ the\ original\ Microdata\ Table}$$

Diversity Percentage =

$$\frac{\sum_{EQ=1}^{MGID}\left(\frac{\sum Diversity\ of\ NMA}{No.of\ \ NMA}\right)}{MGID}$$

$$Where, \quad MGID\ =\ Maximum\ Group\ ID$$

$$NMA\ =\ Non\ Masked\ Attribute$$

The diversity for each NMA is calculated as follows:

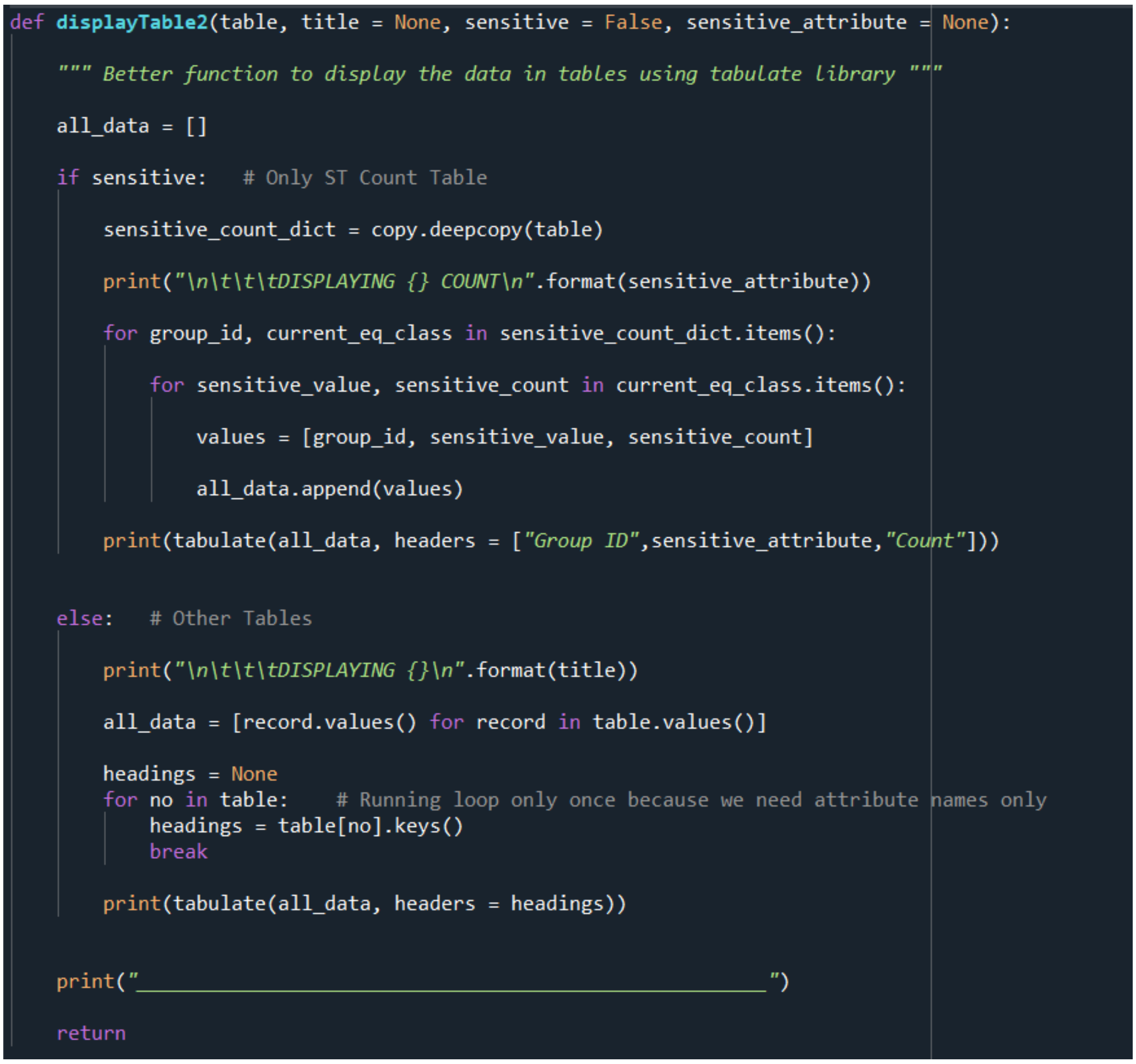
Diversity for each NMA =

$$\frac{No.of\ unique\ values\ of\ attribute}{Total\ no.of\ values\ of\ attribute}$$

A more advanced diversification formula can be formulated by taking into consideration. The weighted mean of the attributes. The attributes which are more sensitive can be given higher weights than those with less degree of sensitivity.

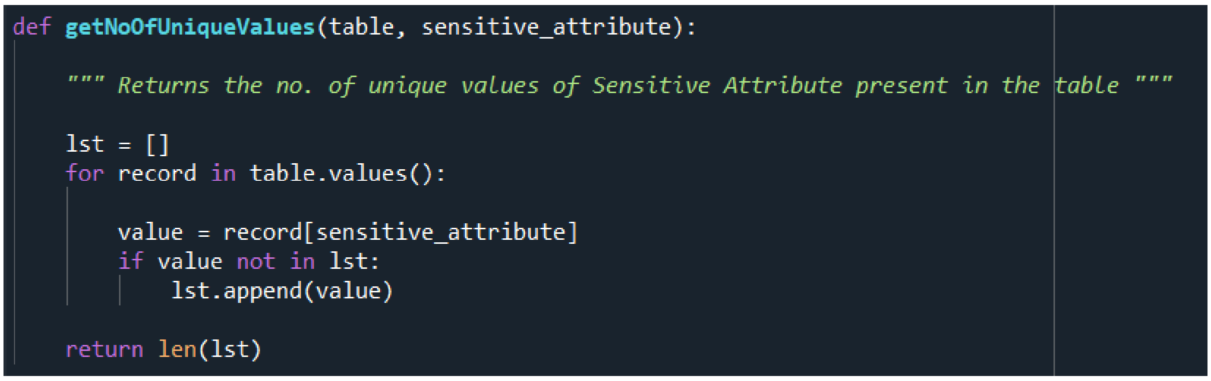
For the sensitive attribute of marital status, we have not taken the semantic hierarchical tree approach because each marital status value can have either one of two values: Unmarried and Married. Therefore, we have undertaken the diversification algorithm where at most only one specific value of marital status can be present in an equivalence class, which comes at the cost of some equivalence classes having the number of records as less than k.

Some Code Snippets:



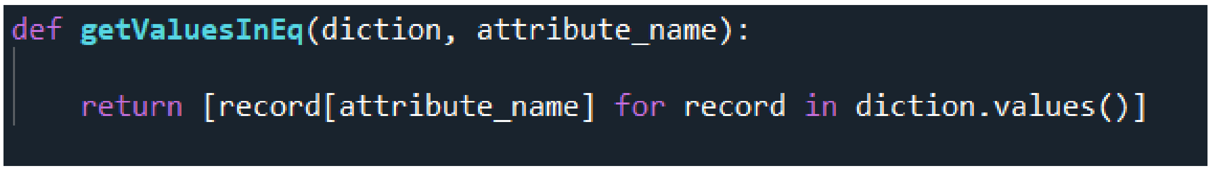
```
def displayTable2(table, title = None, sensitive = False, sensitive_attribute = None):

    """ Better function to display the data in tables using tabulate library """

    all_data = []

    if sensitive:   # Only ST Count Table

        sensitive_count_dict = copy.deepcopy(table)

        print("\n\t\t\tDISPLAYING {} COUNT\n".format(sensitive_attribute))

        for group_id, current_eq_class in sensitive_count_dict.items():

            for sensitive_value, sensitive_count in current_eq_class.items():

                values = [group_id, sensitive_value, sensitive_count]

                all_data.append(values)

        print(tabulate(all_data, headers = ["Group ID",sensitive_attribute,"Count"]))


    else:   # Other Tables

        print("\n\t\t\tDISPLAYING {}\n".format(title))

        all_data = [record.values() for record in table.values()]

        headings = None
        for no in table:    # Running loop only once because we need attribute names only
            headings = table[no].keys()
            break

        print(tabulate(all_data, headers = headings))


    print("______________________________________________________")

    return
```

The function displayTable2() is used for the displaying of various tables be it QIT, ST, Original Microdata, Modified Microdata and so on and so forth.



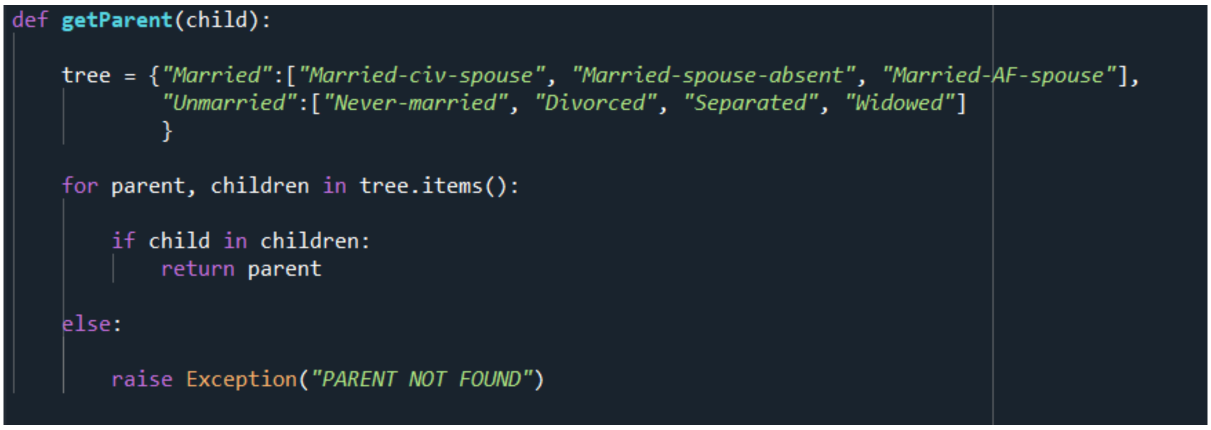
```python
def getNoOfUniqueValues(table, sensitive_attribute):

    """ Returns the no. of unique values of Sensitive Attribute present in the table """

    lst = []
    for record in table.values():

        value = record[sensitive_attribute]
        if value not in lst:
            lst.append(value)

    return len(lst)
```

The function getNoOfUniqueValues() calculates the total count of the no. of unique values present for the sensitive attribute. (sensitive attribute depends on the algorithm chosen)



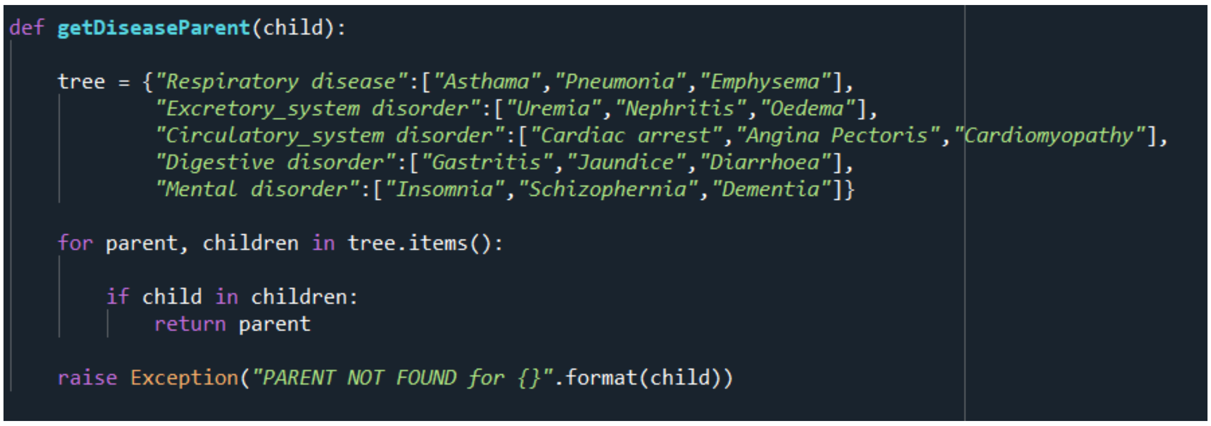
```python
def getValuesInEq(diction, attribute_name):

    return [record[attribute_name] for record in diction.values()]
```

The function getValuesInEq() is used to get the values of all the records in a particular equivalence class for a particular attribute. The use of list comprehension makes for a better and more efficient code.



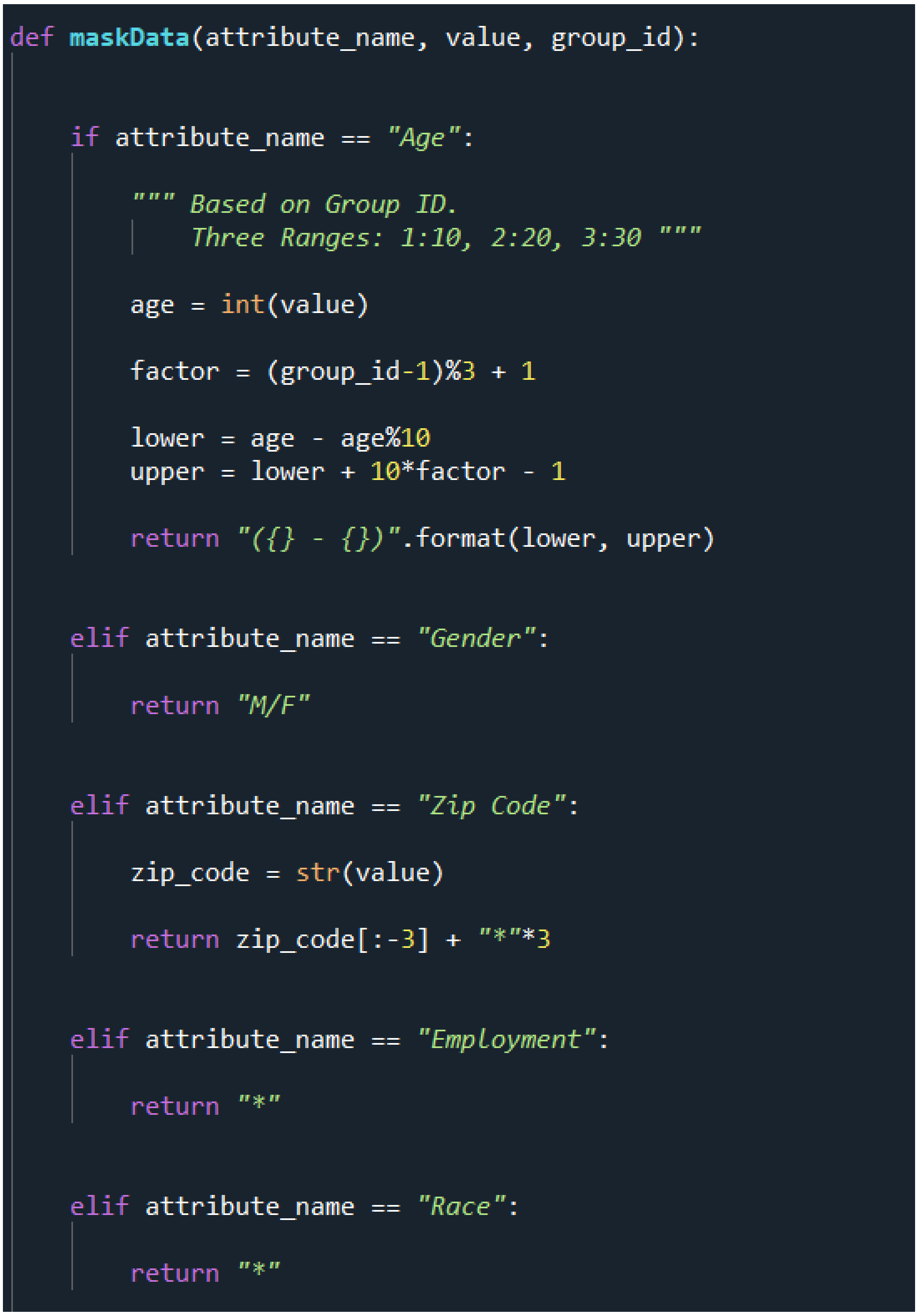
```python
def getParent(child):

    tree = {"Married":["Married-civ-spouse", "Married-spouse-absent", "Married-AF-spouse"],
            "Unmarried":["Never-married", "Divorced", "Separated", "Widowed"]
            }

    for parent, children in tree.items():

        if child in children:
            return parent

    else:

        raise Exception("PARENT NOT FOUND")
```

The function getParent() is used to get the Parent value of the Marital Status Attribute for a particular record.



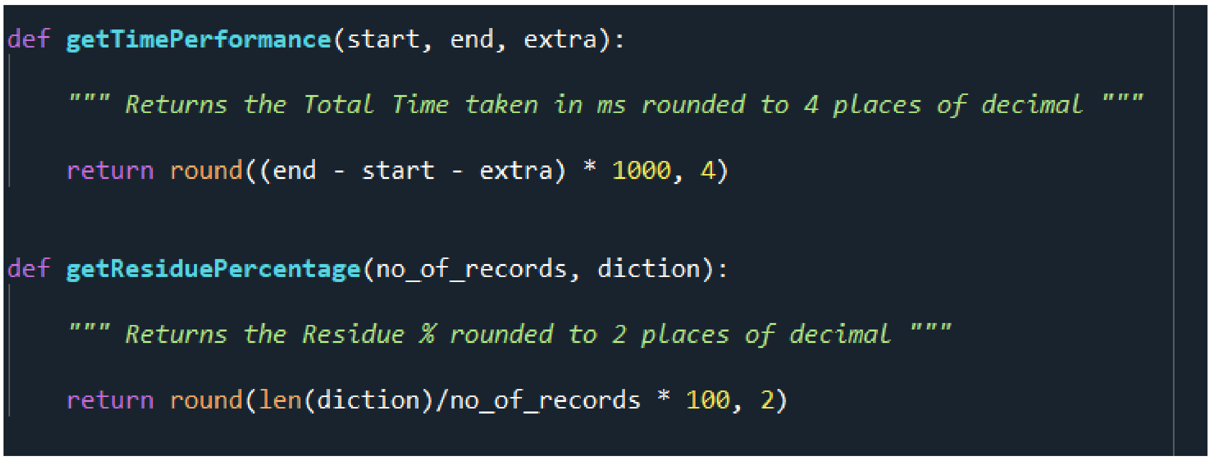
```python
def getDiseaseParent(child):

    tree = {"Respiratory disease":["Asthama","Pneumonia","Emphysema"],
            "Excretory_system disorder":["Uremia","Nephritis","Oedema"],
            "Circulatory_system disorder":["Cardiac arrest","Angina Pectoris","Cardiomyopathy"],
            "Digestive disorder":["Gastritis","Jaundice","Diarrhoea"],
            "Mental disorder":["Insomnia","Schizophernia","Dementia"]}

    for parent, children in tree.items():

        if child in children:
            return parent

    raise Exception("PARENT NOT FOUND for {}".format(child))
```

The function getDiseaseParent() is used to get the Parent value of the Disease Attribute for a particular record.



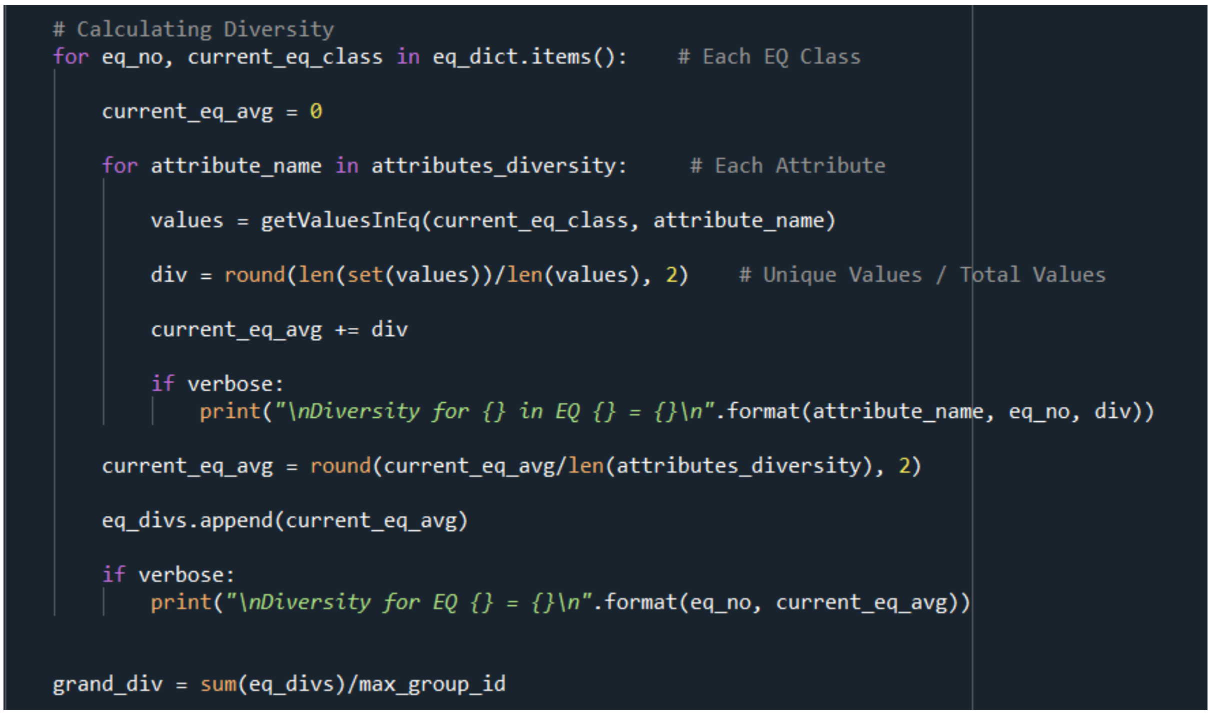
```python
def maskData(attribute_name, value, group_id):

    if attribute_name == "Age":
        """ Based on Group ID.
            Three Ranges: 1:10, 2:20, 3:30 """
        age = int(value)
        factor = (group_id-1)%3 + 1
        lower = age - age%10
        upper = lower + 10*factor - 1
        return "({} - {})".format(lower, upper)

    elif attribute_name == "Gender":
        return "M/F"

    elif attribute_name == "Zip Code":
        zip_code = str(value)
        return zip_code[:-3] + "*"*3

    elif attribute_name == "Employment":
        return "*"

    elif attribute_name == "Race":
        return "*"
```

The masking of data is done in the above mentioned code snippet. Age is masked in the range of 10, 20, 30 depending on the Equivalence Class No. (Group ID).



```python
def getTimePerformance(start, end, extra):

    """ Returns the Total Time taken in ms rounded to 4 places of decimal """

    return round((end - start - extra) * 1000, 4)

def getResiduePercentage(no_of_records, diction):

    """ Returns the Residue % rounded to 2 places of decimal """

    return round(len(diction)/no_of_records * 100, 2)
```

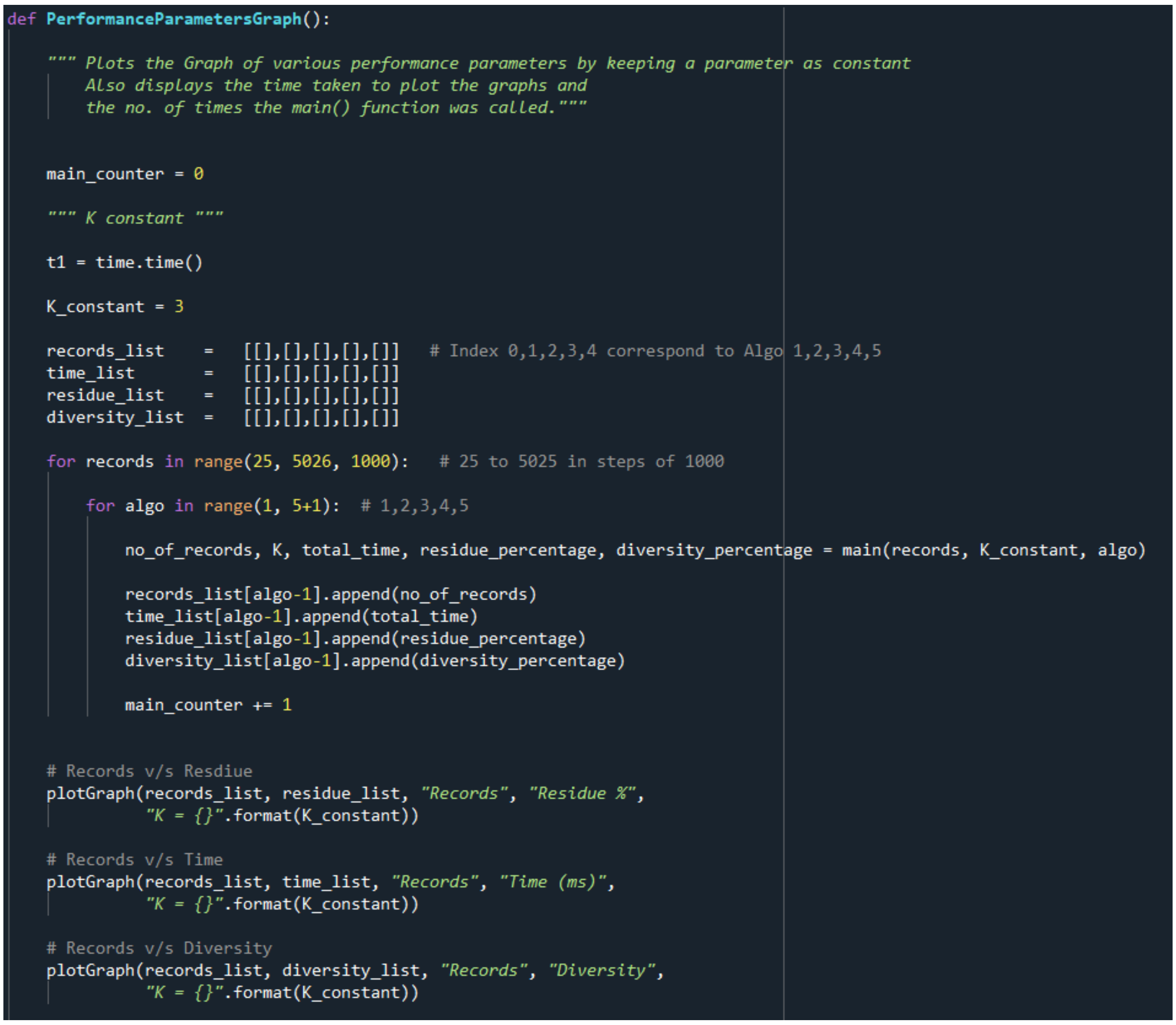
The following functions calculate the Time Performance and Residue Percentage of the Records for the algorithm chosen.



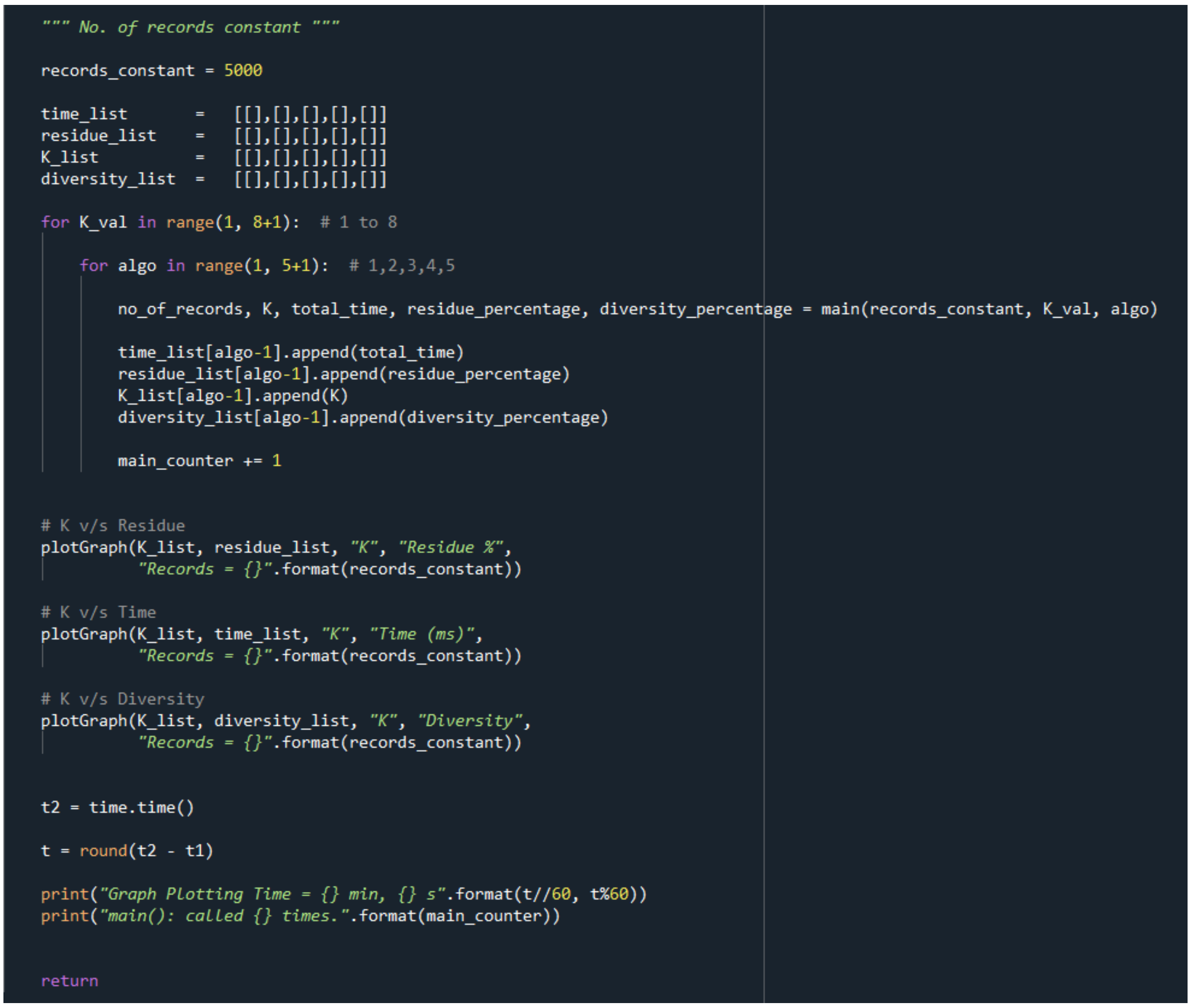
```python
    # Calculating Diversity
    for eq_no, current_eq_class in eq_dict.items():    # Each EQ Class

        current_eq_avg = 0

        for attribute_name in attributes_diversity:     # Each Attribute

            values = getValuesInEq(current_eq_class, attribute_name)

            div = round(len(set(values))/len(values), 2)     # Unique Values / Total Values

            current_eq_avg += div

            if verbose:
                print("\nDiversity for {} in EQ {} = {}\n".format(attribute_name, eq_no, div))

        current_eq_avg = round(current_eq_avg/len(attributes_diversity), 2)

        eq_divs.append(current_eq_avg)

        if verbose:
            print("\nDiversity for EQ {} = {}\n".format(eq_no, current_eq_avg))

    grand_div = sum(eq_divs)/max_group_id
```

For calculating diversity for each equivalence class and the entire modified microdata table, the above function is used, wherein we first iterate through each equivalence class and calculate the diversity continuously for each attribute. Finally, the grand average is taken and taken as the diversity of the entire modified microdata table.

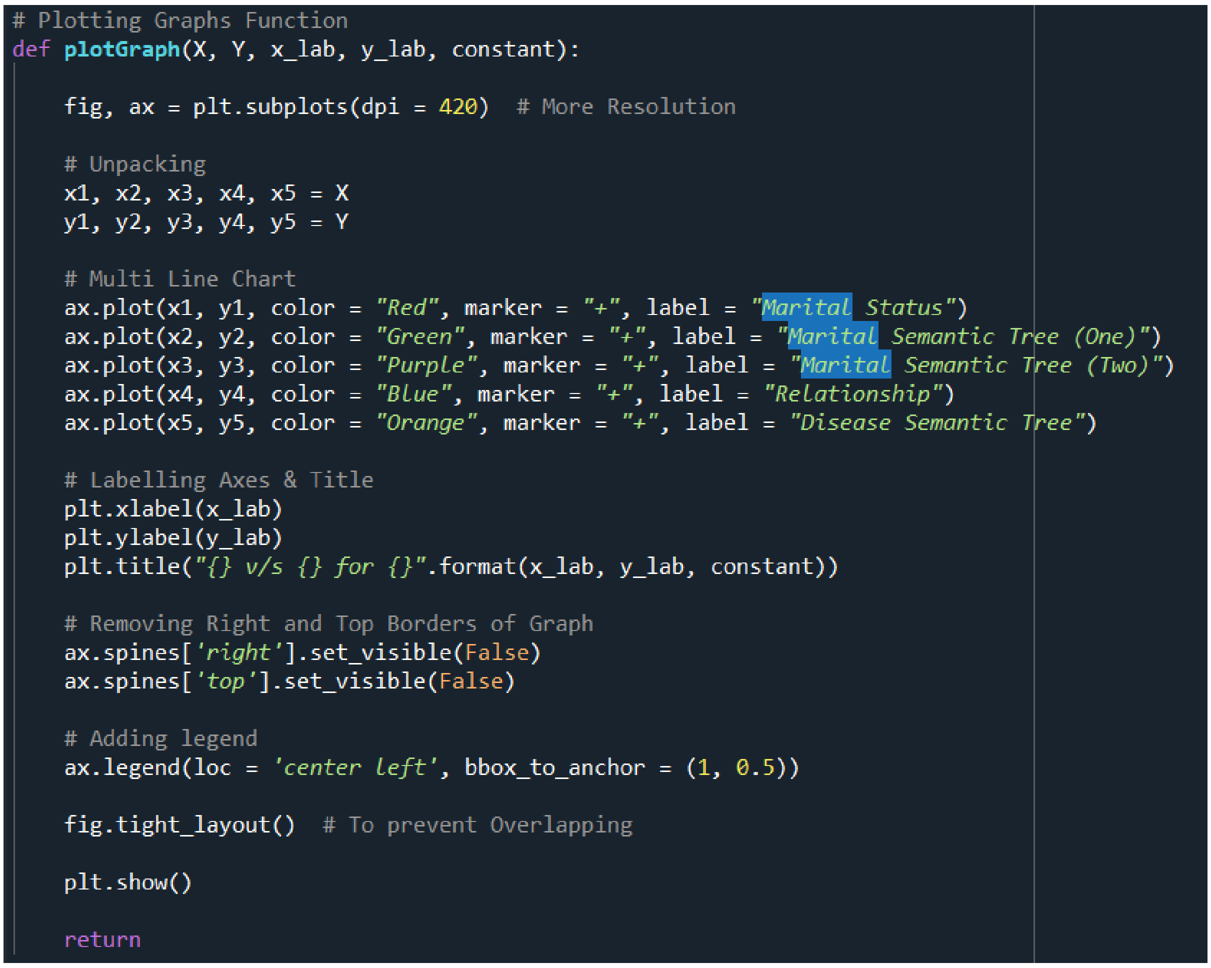
For the latter program mode chosen (i.e. Performance Parameters Graph), the entire program is run for various permutations and combinations of no. of records, k, algorithms, by keeping no. of records or k constant.



```python
def PerformanceParametersGraph():

    """ Plots the Graph of various performance parameters by keeping a parameter as constant
        Also displays the time taken to plot the graphs and
        the no. of times the main() function was called."""


    main_counter = 0

    """ K constant """

    t1 = time.time()

    K_constant = 3

    records_list    =   [[],[],[],[],[]]   # Index 0,1,2,3,4 correspond to Algo 1,2,3,4,5
    time_list       =   [[],[],[],[],[]]
    residue_list    =   [[],[],[],[],[]]
    diversity_list  =   [[],[],[],[],[]]

    for records in range(25, 5026, 1000):   # 25 to 5025 in steps of 1000

        for algo in range(1, 5+1):  # 1,2,3,4,5

            no_of_records, K, total_time, residue_percentage, diversity_percentage = main(records, K_constant, algo)

            records_list[algo-1].append(no_of_records)
            time_list[algo-1].append(total_time)
            residue_list[algo-1].append(residue_percentage)
            diversity_list[algo-1].append(diversity_percentage)

            main_counter += 1


    # Records v/s Resdiue
    plotGraph(records_list, residue_list, "Records", "Residue %",
              "K = {}".format(K_constant))

    # Records v/s Time
    plotGraph(records_list, time_list, "Records", "Time (ms)",
              "K = {}".format(K_constant))

    # Records v/s Diversity
    plotGraph(records_list, diversity_list, "Records", "Diversity",
              "K = {}".format(K_constant))
```



```
""" No. of records constant """

records_constant = 5000

time_list       =   [[],[],[],[],[]]
residue_list    =   [[],[],[],[],[]]
K_list          =   [[],[],[],[],[]]
diversity_list  =   [[],[],[],[],[]]

for K_val in range(1, 8+1):  # 1 to 8

    for algo in range(1, 5+1):  # 1,2,3,4,5

        no_of_records, K, total_time, residue_percentage, diversity_percentage = main(records_constant, K_val, algo)

        time_list[algo-1].append(total_time)
        residue_list[algo-1].append(residue_percentage)
        K_list[algo-1].append(K)
        diversity_list[algo-1].append(diversity_percentage)

        main_counter += 1


# K v/s Residue
plotGraph(K_list, residue_list, "K", "Residue %",
          "Records = {}".format(records_constant))

# K v/s Time
plotGraph(K_list, time_list, "K", "Time (ms)",
          "Records = {}".format(records_constant))

# K v/s Diversity
plotGraph(K_list, diversity_list, "K", "Diversity",
          "Records = {}".format(records_constant))


t2 = time.time()

t = round(t2 - t1)

print("Graph Plotting Time = {} min, {} s".format(t//60, t%60))
print("main(): called {} times.".format(main_counter))


return
```


```
# Plotting Graphs Function
def plotGraph(X, Y, x_lab, y_lab, constant):

    fig, ax = plt.subplots(dpi = 420)  # More Resolution

    # Unpacking
    x1, x2, x3, x4, x5 = X
    y1, y2, y3, y4, y5 = Y

    # Multi Line Chart
    ax.plot(x1, y1, color = "Red", marker = "+", label = "Marital Status")
    ax.plot(x2, y2, color = "Green", marker = "+", label = "Marital Semantic Tree (One)")
    ax.plot(x3, y3, color = "Purple", marker = "+", label = "Marital Semantic Tree (Two)")
    ax.plot(x4, y4, color = "Blue", marker = "+", label = "Relationship")
    ax.plot(x5, y5, color = "Orange", marker = "+", label = "Disease Semantic Tree")

    # Labelling Axes & Title
    plt.xlabel(x_lab)
    plt.ylabel(y_lab)
    plt.title("{} v/s {} for {}".format(x_lab, y_lab, constant))

    # Removing Right and Top Borders of Graph
    ax.spines['right'].set_visible(False)
    ax.spines['top'].set_visible(False)

    # Adding legend
    ax.legend(loc = 'center left', bbox_to_anchor = (1, 0.5))

    fig.tight_layout()  # To prevent Overlapping

    plt.show()

    return
```
The above code snippets show the underlying code that does the work of plotting multiline graphs with linear line between two consecutive points.

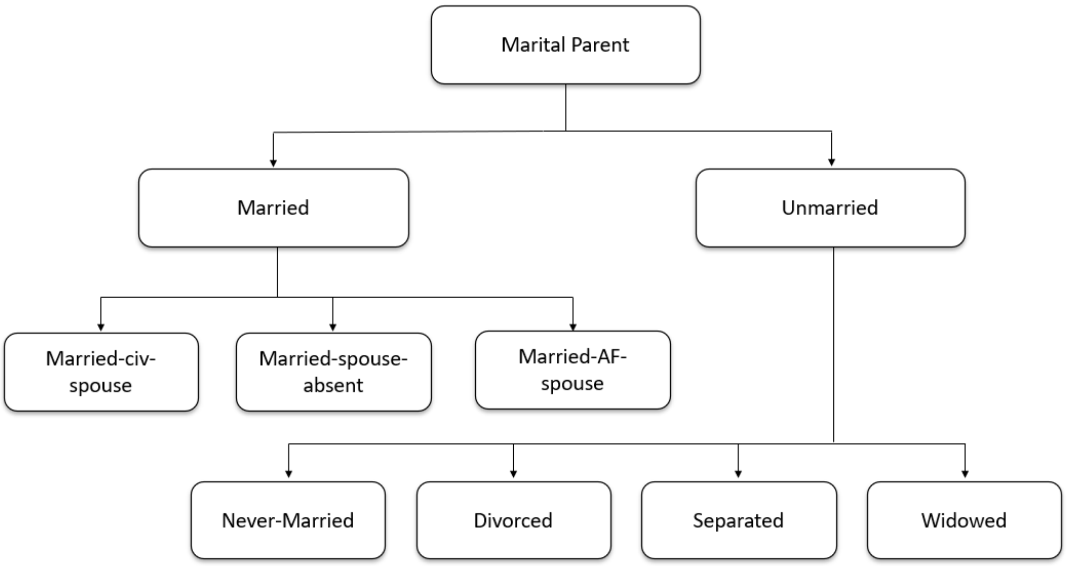
4. RESULTS AND CONCLUSIONS

The performance of the program is calculated by plotting out various graphs for various parameters keeping a check on the required constant conditions for each graphical estimation. A total of 6 graphs have been plotted to have the best estimation of the performance of the program. The following are the Parameters of Graph Plots: -

I. Records v/s Residue % for K=3
II. Records v/s Time for K=3
III. Records v/s Diversity for K=3
IV. K v/s Residue % for Records = 5000
V. K v/s Time for Records = 5000
VI. K v/s Diversity for Records = 5000

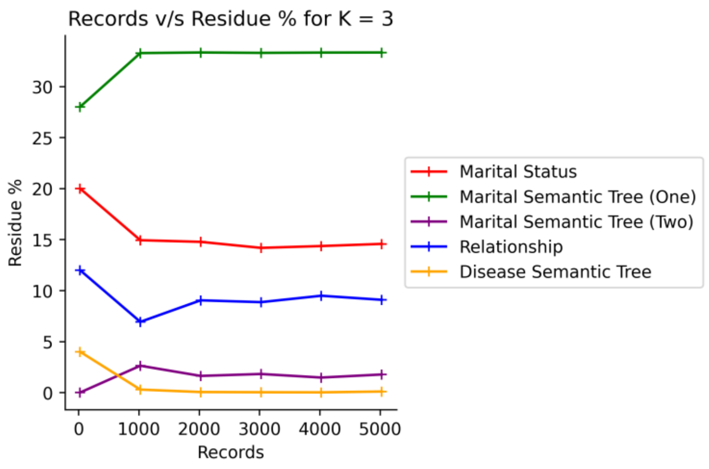
Each of these Graphs have been plotted for 5 different Algorithms: -

1) Marital Status – Here in the modified Microdata Table, Unique Values of Marital Status are present in each Equivalence Class.
2) Marital Semantic Tree (One) – In each Equivalence Class, Modified Microdata Table contains Unique Marital Parent, i.e., one Married and one Unmarried record.
3) Marital Semantic Tree (Two) – In each Equivalence Class, only one type of Marital Parent can be repeated at most 2 times.
4) Relationship – Each Equivalence Class consists of records containing Unique Relationships.
5) Disease Semantic Tree – Each Equivalence Class consists of records of Unique Parent Diseases.



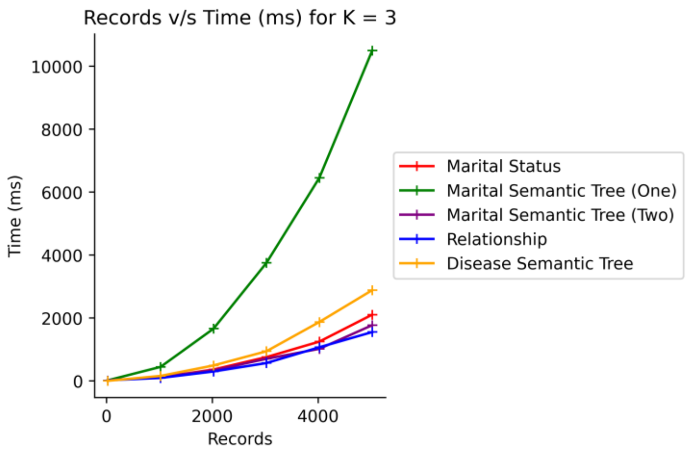
1. **No. of Records v/s Residue %**

This graph has been plotted for No. of Records against Residue % for a constant value of K=3. It's plotted for 5 different algorithms which are shown in the below graph. From the graph, we can notice that the best possible case for arrangement of records will be based on Unique Diseases in each equivalence group as the residue % for it is very less for larger records compared to the other algorithms. This is because number of Unique Diseases are more than the number of Unique Marital Parents. Hence a greater number of residues will be obtained for unique marital parents. For larger datasets, most of the algorithms have constant values which is shown by the curve being parallel to x-axis. Hence use of larger datasets makes it stable.



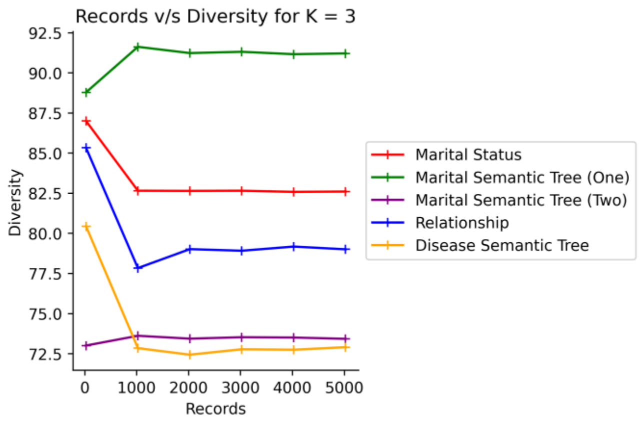
2. **No. of Records v/s Time (ms)**

The following graph has been plotted for No. of Records against Time taken for the execution of program in milliseconds for constant value of K=3. In the graph, we can see that the time taken for distribution of records based on Unique Marital Parent takes more time. Since the Uniqueness of Marital Parent is very low, large number of residues will be obtained in the residue list. Hence more time will be spent to iterate through the residue list multiple times for the replacement of records. Time taken for the execution of program is almost same for other algorithms.



3. No. of Records v/s Diversity

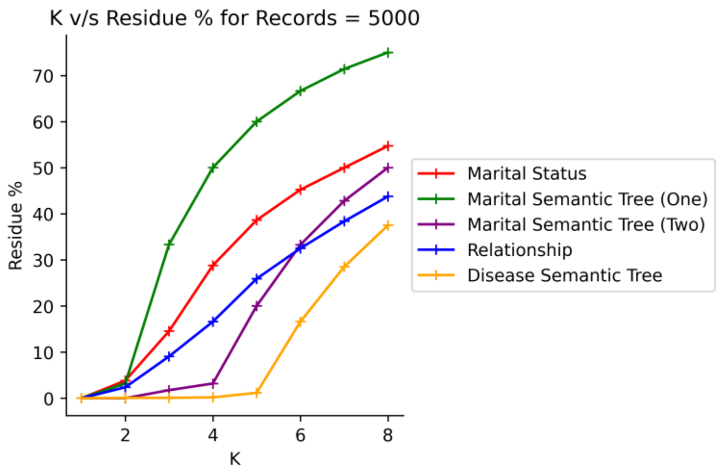
The below graph is plotted for No. of Records against amount of Diversity for the entire table at a constant value of K=3. For lesser datasets, the diversity is more for almost all algorithms. But as the datasets become larger, the diversity decreases and almost becomes constant after a point. Whereas for the Marital Semantic Tree for Unique Parents, diversity increases as the number of records increases because most of the records will be sent to residue list due to which the number of records in final Modified Microdata Table will be less. Hence less repetition and more diversity.



4. K v/s Residue %

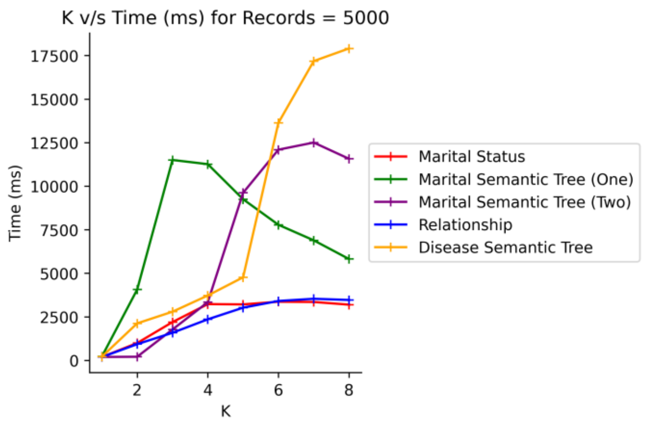
The following graph is plotted for K v/s Residue % for constant number of records=5000. There is an inclined increase in the slope of the curve for all algorithms. This is because large datasets of unique values are not present, so when the K value increases there will be more repetition which leads to higher residue %.

Since the no. of unique values in Marital Semantic Tree (One) is least, it has the highest number of residue records.



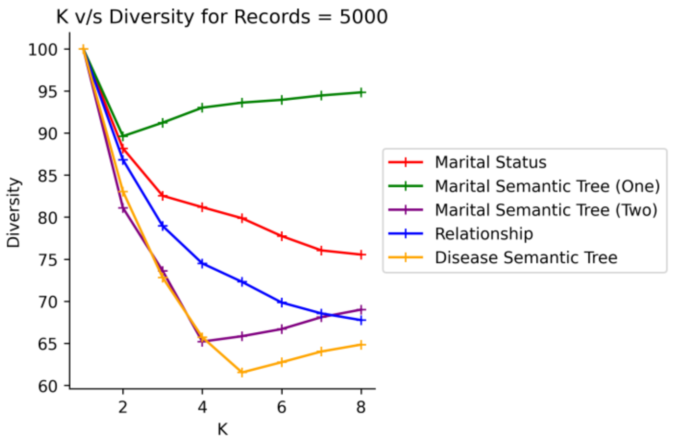
5. K v/s Time (ms)

The following graph is plotted for K against the Time taken for execution of entire program for constant number of records = 5000. From the graph, we can notice that for almost all algorithms Time taken for execution increases as K value increases because the program iterates through larger residue and then exchanges the records due to which time taken for execution increases. There is a special case for Marital Semantic Tree (One) curve, initially the time taken increases because of increase in number of exchanges but after K=4 the number of exchanges decreases because of having least number of Unique Values. So, time taken for execution also decreases.



6. K v/s Diversity

The following graph is plotted for K against the Diversity of Entire table for constant number of records=5000. Initially the diversity for all algorithms is very high for smaller values of K because the no. of unique values are more relative to K values, hence less repetition. But as K value increases, the number of unique values of records will become relatively low, hence more repetition and lower diversity. For Marital Semantic Tree (One), the Diversity remains high even for large values of K because it has least unique values, so most of the records remain in residue list and only few records will be there in final modified Microdata Table. Hence high diversity obtained.

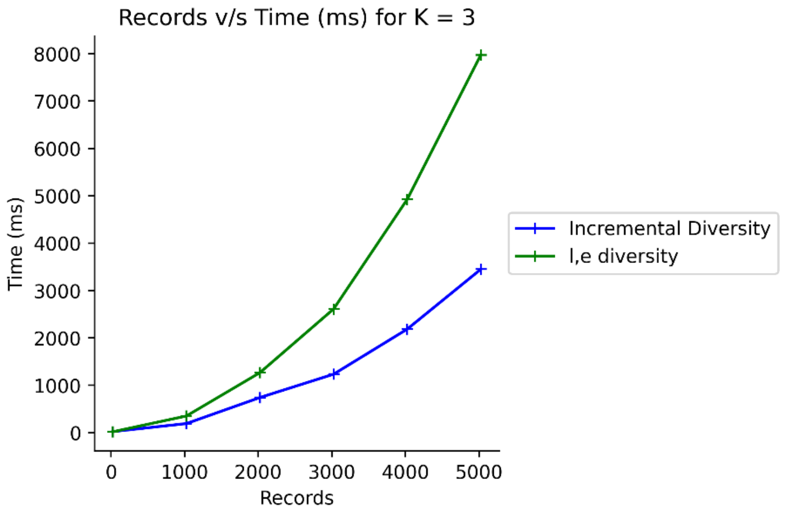


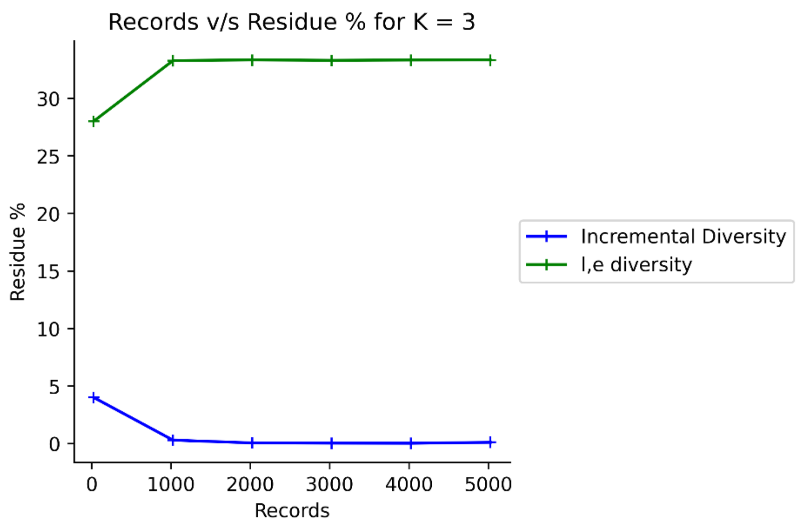
Comparison between l, e diversity algorithm and incremental diversity algorithm

l, e diversity algorithm employs the use of a primary sensitive attribute with lesser no. of parents in the semantic hierarchical tree (Eg: Marital Status). The records in the equivalence class are diversified such that only one common parent for the sensitive attribute should be present in a particular equivalence class.

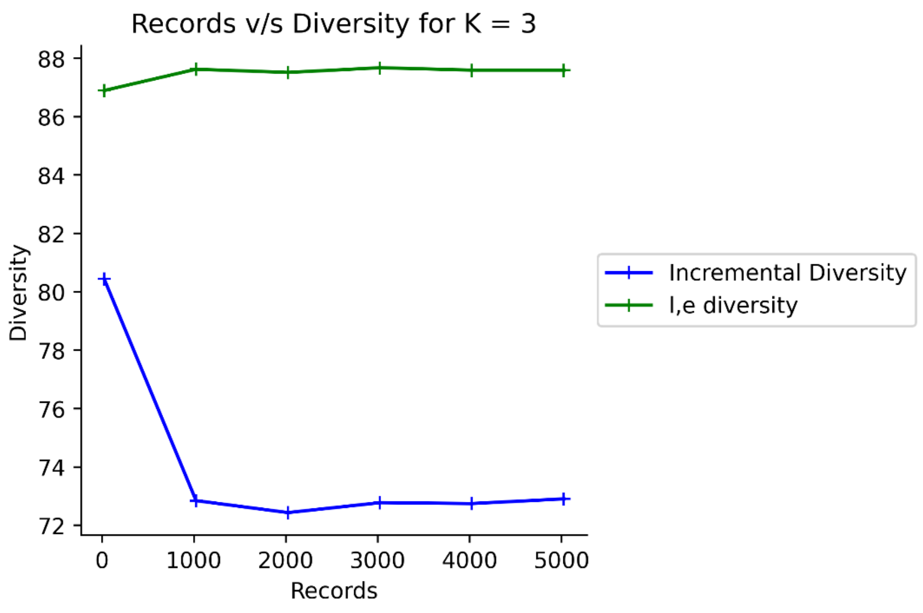
Incremental diversity algorithm makes use of a primary sensitive attribute with more no, of parents in the semantic hierarchical tree (Eg: Disease). It follows the same condition as l, e diversity where only non-repeating parents should be present in each equivalence class. In addition, it also performs the incremental diversification for secondary, tertiary and quaternary sensitive attributes thereby increasing the diversity characteristics for the equivalence classes and also the entire table.

Incremental diversity outperforms l, e diversity in terms of faster time performance (code runtime) and overall decrease in residue records percentage.



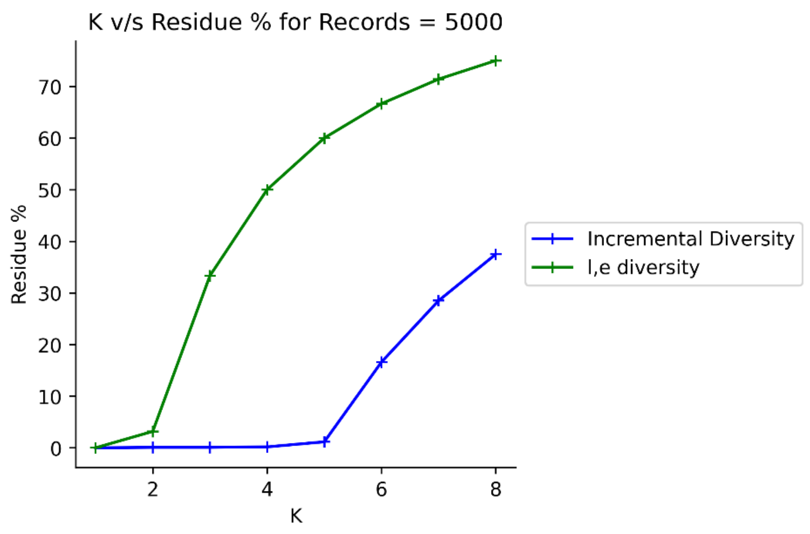


From the graphs we can draw the inference that incremental diversity produces lesser residue records and runs faster in spite of performing diversification for multiple sensitive attributes.
Producing lesser residue records is a double edged sword which leads to lesser diversity in the records present in the equivalence classes of the table. This is a drawback of the incremental diversity algorithm

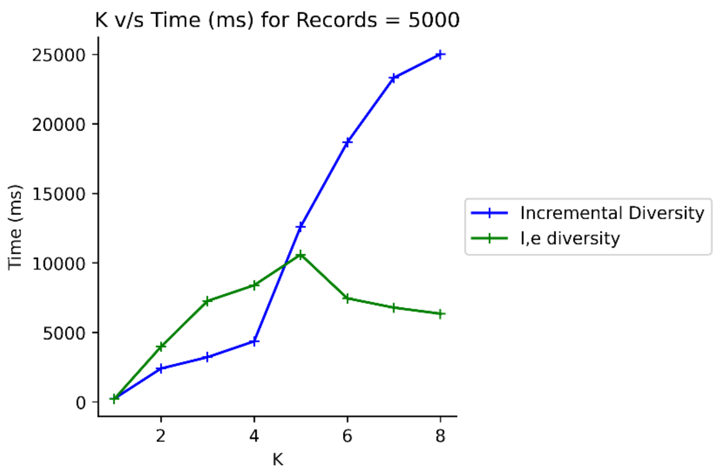


.

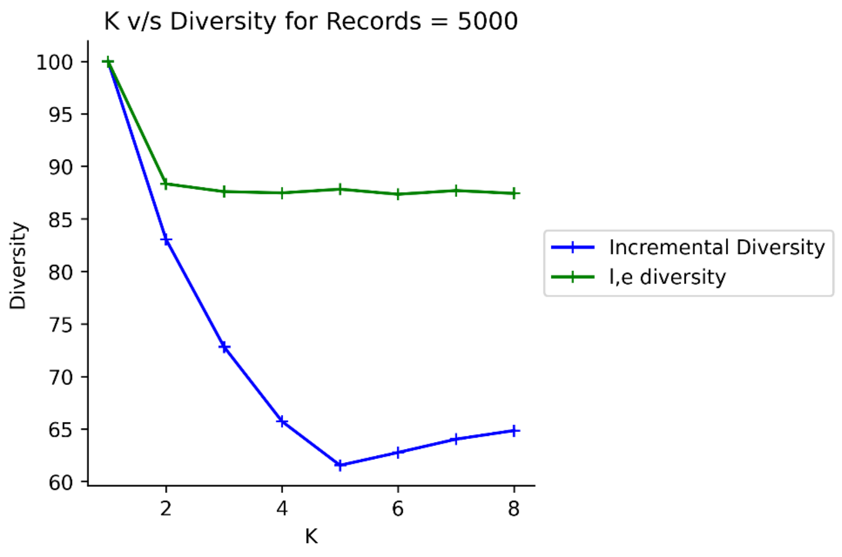
Lesser values of k produce lesser no. of residue records as the condition of non-repeating values is easily met by fewer records in each equivalence class (=k). Incremental Diversity produces lesser residue records as compared to l, e diversity for the same k value.



For values of k less than or equal to 4, incremental diversity takes lesser time to compute as compared l, e diversity. For values of k more than 4, more no. of exchanges and swapping is performed in the equivalence class due to more no. of records in each equivalence class, therefore time increases for incremental diversity.



Incremental diversity is found to be less diverse than l,e diversity as k value keeps increasing.



Choosing Primary Sensitive Attribute

The choice of the right primary sensitive attribute can make or break the diversification and privacy of a table. Generally, a sensitive attribute with more no. of parents in its semantic hierarchical tree (Eg: Disease) should be chosen as compared to a sensitive attribute with lesser no. of parents in its semantic tree (Eg: Marital Status). In the above result section with graphs, l, e diversity used Marital Status as the primary sensitive attribute and incremental diversity chooses Disease as the primary sensitive attribute. This leads to some interesting results:

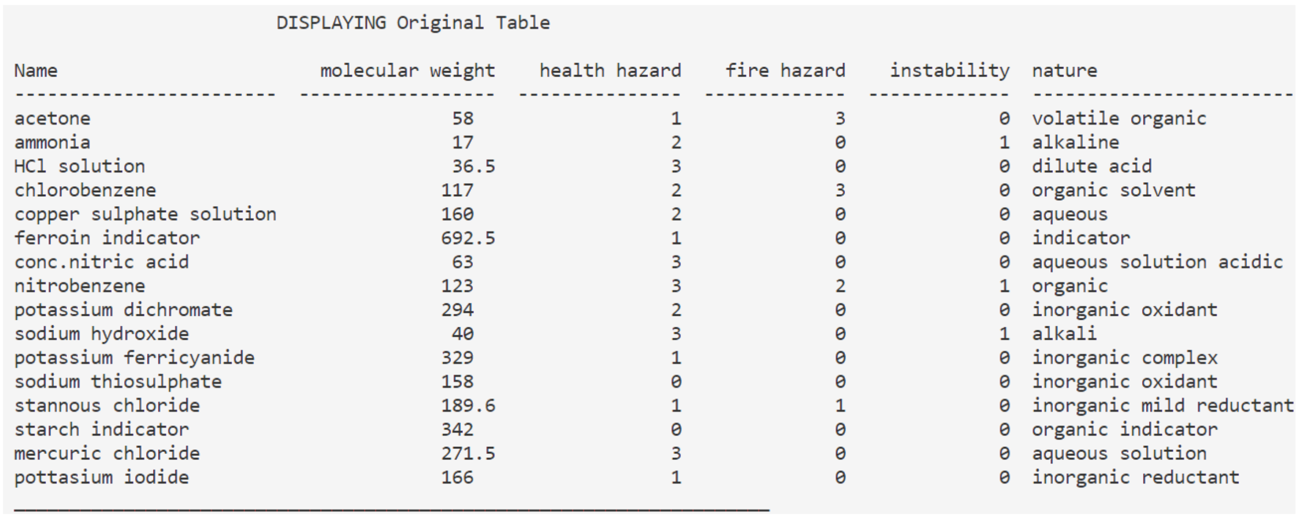
There will be a tradeoff between residue records produced and diversity. They are directly proportional to each other. Disease as the primary sensitive attribute produces lesser no. of residue records but at the same time diversity of the records is hampered. Marital Status as the primary sensitive attribute produces highly diverse table but it comes at the cost of more no. of residues produced and the loss of precious data.

In conclusion, it depends on the use case as to whether diversity needs to be sacrificed in order to produce lesser residue records and prevent huge data loss or, diversity is of utmost importance and it is okay to overlook the production of enormous residue records.

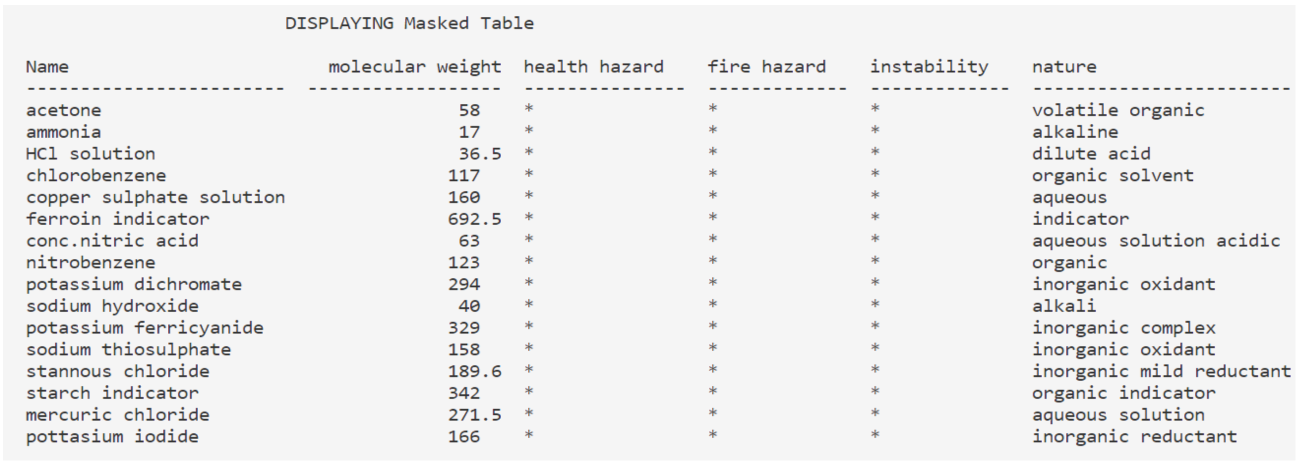
7.SOFTWARE TOOLS USED

1. Python programming language
2. Csv files
3. Math Module
4. Time Module
5. Matplotlib.pyplot module
6. Tabulate module
7. Copy module

8.APPLICATION IN THE FIELD OF CHEMISTRY



```
                        DISPLAYING Original Table

Name                        molecular weight   health hazard   fire hazard   instability   nature
------------------------  ------------------  --------------  ------------  ------------  ------------------------
acetone                                   58               1             3             0  volatile organic
ammonia                                   17               2             0             1  alkaline
HCl solution                              36.5             3             0             0  dilute acid
chlorobenzene                            117               2             3             0  organic solvent
copper sulphate solution                 160               2             0             0  aqueous
ferroin indicator                        692.5             1             0             0  indicator
conc.nitric acid                          63               3             0             0  aqueous solution acidic
nitrobenzene                             123               3             2             1  organic
potassium dichromate                     294               2             0             0  inorganic oxidant
sodium hydroxide                          40               3             0             1  alkali
potassium ferricyanide                   329               1             0             0  inorganic complex
sodium thiosulphate                      158               0             0             0  inorganic oxidant
stannous chloride                        189.6             1             1             0  inorganic mild reductant
starch indicator                         342               0             0             0  organic indicator
mercuric chloride                        271.5             3             0             0  aqueous solution
pottasium iodide                         166               1             0             0  inorganic reductant
_______________________________________________________________________
```



```
                        DISPLAYING Masked Table

Name                        molecular weight  health hazard   fire hazard   instability   nature
------------------------  ------------------  --------------  ------------  ------------  ------------------------
acetone                                   58  *               *             *             volatile organic
ammonia                                   17  *               *             *             alkaline
HCl solution                              36.5  *             *             *             dilute acid
chlorobenzene                            117  *               *             *             organic solvent
copper sulphate solution                 160  *               *             *             aqueous
ferroin indicator                        692.5  *             *             *             indicator
conc.nitric acid                          63  *               *             *             aqueous solution acidic
nitrobenzene                             123  *               *             *             organic
potassium dichromate                     294  *               *             *             inorganic oxidant
sodium hydroxide                          40  *               *             *             alkali
potassium ferricyanide                   329  *               *             *             inorganic complex
sodium thiosulphate                      158  *               *             *             inorganic oxidant
stannous chloride                        189.6  *             *             *             inorganic mild reductant
starch indicator                         342  *               *             *             organic indicator
mercuric chloride                        271.5  *             *             *             aqueous solution
pottasium iodide                         166  *               *             *             inorganic reductant
```

The important stages of code:

Stage 1: Creating a function to display any set of dictionaries in a tabular form.

Stage 2: Then creating function to read the set of dictionaries of chemicals.

Stage 3: Then with help of another two functions creating set of dictionaries of chemicals in which values of health hazard, fire hazard and instability are not present, instead the symbol * is filled in place of the values.

9.REFERENCES

[1] ***K-anonymity: A model for protecting the privacy:*** LATANYA SWEENEY, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.

[2] ***(1, m, d) - Anonymity: A Resisting Similarity Attack Model for Multiple Sensitive Attributes:*** Junjie Jia, Luting Chen School of Computer Science and Engineering, Northwest Normal University Lanzhou, China.

[3] ***Simple Distribution of Sensitive Values for Multiple Sensitive Attributes in Privacy Preserving Data Publishing to Achieve Anatomy:*** Widodo Informatics Education Universitas Negeri Jakarta, Indonesia, Murien Nugraheni Information Systems and Technology Universitas, Negeri Jakarta, Indonesia. Irma Permata.
Sari Information Systems and Technology Universitas Negeri Jakarta, Indonesia. 2021 2nd International Conference on Innovative and Creative Information Technology (Ilitch) September 23-25, 2021, UKSW Salaita, Indonesia.

[4] ***A preliminary phase on anatomizing multiple sensitive attributes by determining main sensitive attribute:*** W Widodo1 and A Wahyudin , Department of Informatics Education, Universitas Negeri Jakarta Jl. Rwamagana Mukai, Jakarta, Indonesia 2
Department of Computer Science, Universitas Pendidikan Indonesia, Bandung, Indonesia.

[5]***A Distributional Model of Sensitive Values on p-Sensitive in Multiple Sensitive Attributes:*** Widodo Department of Informatics Education Universitas Negeri Jakarta Jakarta,

Indonesia Wahyu Catur Wibowo Faculty of Computer Science University of Indonesia Jakarta, Indonesia.

[6] ***(l, e)-Diversity – A Privacy Preserving Model to Resist Semantic Similarity Attack :*** Hai yuan Wang, Jianmin Han, Jiyi Wang ,Department of Computer Science and Technology Zhejiang Normal University, Jinhua, Zhejiang, PRC Wang Xingzhi College Zhejiang Normal University, Jinhua , Zhejiang, PRC

[7] ***Anonymity: A Formalization of Privacy – ‘l’-Diversity:*** Michael Kern Betreuer: Ralph Holz Seminar Future Internet SS2013 Lehrstuhl Netzarchitekturen und Netzdienste Fakultät für Informatik, Technische Universität München.