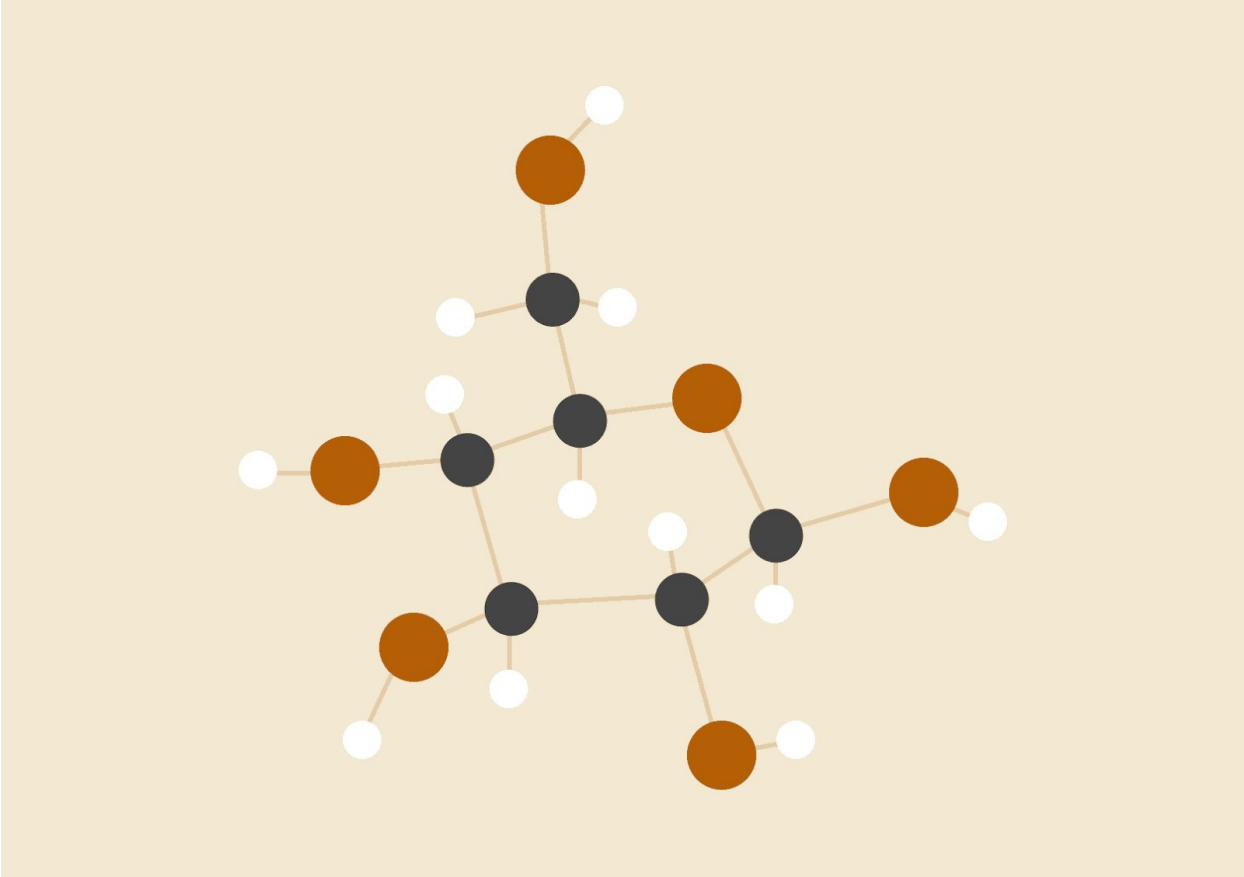
Info Security Project Report

Profiling Internet Users



Sachit Anand Bhootham

U5466-0433

24-04-2020

INTRODUCTION

Internet Profiling is collecting information about internet users and their online behavior to create profiles of their interests, tastes and purchasing habits etc.

In this project we are going to use the internet user's data of 54 users and compare the data of each user with every other user and find out if the data is distinguishable or indistinguishable and also how the window size used affects the results of being distinguishable or indistinguishable (10seconds, 227seconds and 300 seconds). I'm considering two weeks' data of each user and filtering it by considering the data between 8AM to 5PM and also ignoring the weekend data. (i.e Sat and Sun).

I have implemented this project by writing a python code. I have used python as it is easy to understand and more user friendly when compared to the other programming languages.

HYPOTHESIS

When we compare the internet user's data of distinct users, ideally it should be distinct but it also depends upon the window size because as the window size increases there might be a higher probability of a same subject being browsed by multiple users and hence the indistinguishability increases. And also in contrast if all the users have maximum indistinguishable data then decreasing the size of the window might increase the indistinguishability as maximum data is similar. To provide more evidence to this hypothesis and check how the window size plays a crucial role in profiling, this project has been implemented.

MATERIALS

1. Cisco NetFlow V5 - Source of data.
2. https://drive.google.com/drive/folders/1yoCnGBqO9mw9IMo3fTUKkpagqc7FB6ci - 54 users data in Excel files.
3. Spyder (Python 3.7) for coding.

PROCEDURE

As the code is implemented in python 3.7, I have imported all the necessary libraries such as xlrd, xlwt, xlsxwriter, math, scipy.stats, datetime, time etc. To understand the code easily I have implemented the code for two files initially and later using loops I have iterated it through all the possible combinations. The following steps have been followed.

Initial Data Processing : STEP : 1 - I have used xlrd library to read the excel file from its location and extract values from its cells. The column “Real first packet” was considered to extract the data frame. This column was initially converted to datetime from epoch time using the datetime library. Now “doctets” and “duration” columns are considered to calculate octets/duration ignoring all the rows having duration equal to 0. Now applying the following conditions on the real first packet datetime such as 1. Taking only two weeks data which is in between 8AM to 5PM, 2. Ignoring the weekend data i.e Saturday and Sunday etc. Having all these conditions applied I’m copying the following datetime values and octets/duration values in two different lists respectively. First value in list 1 corresponds to the first value to list 2 and so on.

```
    for i in range(1,sheet.nrows):
        X = int(sheet.cell_value(i,5)/1000)
        D = sheet.cell_value(i,9)
        O = sheet.cell_value(i,3)
        Y = datetime.datetime.fromtimestamp(X)
        if(datetime.date(Y.year, Y.month,1).weekday() == 0):
            day = 12
        elif(datetime.date(Y.year, Y.month,1).weekday() == 1):
            day = 11
        elif(datetime.date(Y.year, Y.month,1).weekday() == 2):
            day = 10
        elif(datetime.date(Y.year, Y.month,1).weekday() == 3):
            day = 9
        elif(datetime.date(Y.year, Y.month,1).weekday() == 4):
            day = 15
            dayStart = 4
        elif(datetime.date(Y.year, Y.month,1).weekday() == 5):
            day = 14
        else:
            day = 13
        if(Y.time() >= datetime.time(8,0,0) and Y.time() <= datetime.time(17,0,0)
            OperD = O/D
            OperDList.append(OperD)
            dateList.append(Y)
            durationList.append(D)
            docktetsList.append(O)

```

STEP : 2 - Now using 3 window sizes i.e 10secs, 227secs and 5mins I have iterated the entire list of values which were stored in STEP1. For example if we consider a 5mins window I have taken the entire datetime list and separated the list with 5mins slots and calculated the octets/duration average for every 5mins slot. In this case the total number of 5mins slots for a user with two weeks of data is 12*9*10 = 1080. Now I have separated these 1080 octets/duration averages into two weeks by storing them in two different lists. The same procedure is performed for 10secs and 227secs windows respectively. The same procedure is performed for other users respectively. By the end of STEP2 every user has two weeks data(octets/duration averages) separated into two lists for 3 windows.

```
121
122     for i in range(0,len(defList)):
123         T2 = defList[i]
124         T3 = T2.replace(second = 10, hour = 8, minute = 0)
125         for j in range(0,3240):
126             docktetsAve = 0
127             count = 0
128             for z in range(0,len(dateList)):
129                 if(dateList[z] >= T2 and dateList[z] < T3):
130                     docktetsAve = docktetsAve + OperDList[z]
131                     count = count + 1
132
133             if(count == 0):
134                 count = 1
135             if(T2 < datetime.datetime(year = 2013, month = 2, day = 8, hour = 18
136                 Sheet_1Week_1List.append(docktetsAve/count)
137             else:
138                 Sheet_1Week_2List.append(docktetsAve/count)
139
140             T2 = T2 + datetime.timedelta(seconds = 10)
141             T3 = T3 + datetime.timedelta(seconds = 10)
142
```

Spearman Correlation : Using these octets/duration averages for both week1 and week2 for each user we calculate the Spearman Correlation. To calculate the Spearman Correlation, I have used a library called “scipy.stats”. This library has a function called “spearmanr” which takes two array parameters of the same size. Using this I was able to find r1a2a, r1a2b, r2a2b by passing week1 and week2 octets/duration average lists of subject A as parameters in r1a2a, week1 and week2 octets/duration average lists of subject A and B respectively as parameters in r1a2b and week2 and week2 octets/duration average lists of subject A and B respectively as parameters in r2a2b. Note that we are only using week2 data of subject B for calculating the spearman correlation. The values are considered as 0 if nan and 0.99 if 1(to avoid divide by 0 error.)

```
147     r1a2a = scipy.stats.spearmanr(Sheet_1Week_1List,Sheet_1Week_2List,nan_policy=
148     if(r1a2a == 1):
149         r1a2a = 0.99
150     if(math.isnan(r1a2a)):
151         r1a2a = 0
```

MRR-Z Test : Based on the correlation values which are calculated in the previous step the main part of this project will be done. For the statistical framework of this project I’m using MRR-Z Test(Meng, Rosenthal and Rubins Z Test Statistic) to calculate the value of Z. This has been implemented in a function called ZTest. The Correlation coefficients which were calculated in the above step are passed as parameters and the corresponding Z

value is returned. “N” is also passed as a parameter, it is the sample size of the data set i.e number of windows in a week. For example if we consider the 5min window the value of N would be 504.

```
def ZTest(r1a2a,r1a2b,r2a2b):
    Z1a2a = 1/2*math.log((1+r1a2a)/(1-r1a2a))
    Z1a2b = 1/2*math.log((1+r1a2b)/(1-r1a2b))

    rm2 = ((r1a2a*r1a2a)+(r1a2b*r1a2b))/2
    f = (1-r2a2b)/(2*(1-rm2))
    h = (1-(f*rm2))/(1-rm2)

    a = (Z1a2a - Z1a2b)
    b = math.sqrt(16200 - 3)/(2*(1-r2a2b)*h)
    Z = a*b
    return Z
```

Calculating the P-value : Based on the Z value calculated in the above step the corresponding P values can be calculated. I have implemented the P-value calculation in a function called “PFunction”. This function takes only one parameter Z and returns the corresponding P value.

As every user is being compared with every other user total 54 P values are generated for a single user. I’m storing these 54 P-values of each user in separate lists and hence there are 54 such lists generated for a single window. These 54 lists are now stored in a single list where each list object is an element of the new list. This step is performed to copy all the corresponding P values in an Excel file in the order of rows. For every window an Excel file of dimension 54x54 is generated.

```
def PFunction(Z):
    p = 0.3275911
    a1 = 0.254829592
    a2 = -0.2884496736
    a3 = 1.421413741
    a4 = -1.453152027
    a5 = 1.061405429

    sign = 0
    if(Z < 0):
        sign = -1
    else:
        sign = 1

    x = abs(Z)/math.sqrt(2)
    t = 1/(1+(p*x))
    erf = 1-(((((a5 * t + a4)*t)+a3)*t + a2)*t + a1) * t * math.exp(-x*x);
    return 0.5 * (1 + sign * erf)
```

DATA

Windows/Characteristics	Distinguishable	Indistinguishable
5mins(300secs)	202	2714
227 seconds	148	2768
10 seconds	93	2823

RESULTS AND OBSERVATIONS

I have also implemented a small code which takes the Excel files containing the P-values and returns “Yes” if the value of P is less than or equal to 0.05 (P <= 0.05) that means the correlation coefficient calculated for internet usage patterns for an unknown subject B is significantly smaller than that for a known subject A and such a “subject B” will be

identified as a subject distinct from “subject A”. And returns “No” if the value of P is greater than 0.05(P>0.05) that means the correlation coefficient calculated for internet usage patterns for an unknown subject B is not significantly smaller than that for a known subject A and such a “subject B” will be identified as indistinguishable from “subject A”.

Note: All the data processing and calculations are implemented in the code only, there are no calculations or preprocessing of data performed outside the code. The end output of this code would be 6 Excel files created having the following file names.

- 300sec_window
- 300sec_Y_or_N
- 227sec_window
- 227sec_Y_or_N
- 10sec_window
- 10sec_Y_or_N

For each window I have taken a total of 2916 (i.e 54x54) combinations. One of my observations here is we compare the first user’s 2 weeks of data but only the 2nd week data of the second user. This may show the difference in the P-value when user A and user B are compared, with the P-value of user B and user A compared.

According to our hypothesis a user’s data must be distinct with any other user’s data but it also depends on the time window on which their usage comparison is based on. In this case we can see that in all the windows the number of indistinguishable cases are more when compared to distinguishable cases and also we observe that as the size of the window increases the size of the distinguishable cases also increase. We are seeing this behavior because the browsing data of maximum users is similar to each other. As maximum data is similar and when the window size is minimized the entire window may end up having the same data. This kind of data may be clustered as a data of the users working in the same organisation etc.

This may be a concern when the data is used for many important purposes like in some fields of Cyber Security and fields where the authenticity of the organisation is what matters most and given preference to.

But in contrast, comparison based on higher window size is also useful(where users have distinct internet usage) in some fields of security where Privacy or Anonymity is transparent or in cases like nabbing someone who has access to something to which they are not supposed to have, through the patterns in internet usage. And also several

machine learning algorithms are using these patterns obtained from profiling and using them for advertising, online shopping etc.

CONCLUSION

Therefore we can conclude by saying if there are users who have indistinguishable internet data usage, having a smaller window size may increase the indistinguishability and if there are users who have distinct internet data usage, having a greater window size may increase the indistinguishability.