**Gautam Saini (U50159704)**

# The Project

This document is intended to provide a summary of the information security final project ‘Profiling internet users’ and covers technical aspects of project code execution. The project is designed in **Python 3.7** version language using PyCharm IDE.

# The Required Libraries

The project required a list of python libraries for the successful execution of the code. The list of necessary libraries and tasks performed by a library is given below.

1. Pandas: To perform data processing on given user data files
2. DateTime: To perform slicing/ window creation on giving epoch
3. pickle: To create intermediary data files produced after processing user data in user files.
4. Glob: To list out user data files
5. os: To perform general OS-related tasks such as creating folders etc.

# Project structure

**Scripts:**

The project mainly has four python scripts, and they are named as:

1. main.py:
2. computation.py
3. storage.py
4. utility.py.

To start the project code, execute or **run the main.py** script, which calls the primary method of the script and program start to process user data files. The complete program is broadly divided into two steps names as execution step 1 and execution step 2. The first step performs pre-processing on the data and creates intermediary files using the pickle library. The second step takes those intermediary pickle files and performs further calculation of spearman rank correlation, Z values, and final P values. An elaborate description of each level is covered in the section ‘execution step 1’ and section ‘execution step 2’.

**Folder structure:**

There is a total of 5 folders created by the program having data/computation files.

1. file\_processed: having pre-processed data of each user.
2. Spearman: having the Spearman correlation rank coefficient for each time window.
3. Week: Two CSV files week1 and week 2, each row has user data according to time widow.
4. Z Values: a CSV file in each time window having a matrix of z values for each user.
5. P values: final P table matrix for users according to the time window.

The overall python project structure is shown below in the images.

|  |  |
| --- | --- |
|  |  |

# Execution Steps 1 (Data pre-processing)

The program execution starts with the main.py python file by calling the main method, which triggers the step 1 execution. This step performs pre-processing on user data files stored in the ‘files’ folder of the project. These user files are taken from the location specified in the project documentation. This step reads files data in data frames using panda’s python library and processes them.

As files have a lot of data with some files having millions of records, it mandatory to apply several optimizations to reduce the execution time. There are three types of the window created using the user data according to epoch given in real first packet, and those are 10 seconds, 227 seconds, and 5 minutes. For calculation simplicity, the 5 min window time frame is referred to as **300 seconds window time frame** in the project.

**Optimizations:**

The following optimization has been applied to reading the data to speed up the process

1. Reading files from disk consumes a lot of time, and so as an optimization, each user data file is read-only once into the data frame, which is stored in RAM and then iterated three times for three-time windows.
2. While reading the data in data frame only required column from user files are fetched such as ‘Doctets,’ ‘Real First Packet,’ ‘Duration’ and other are not read into the data frame.
3. The third optimization is that data fetched from the files are the ones having a duration higher than 0 as we are not considering the doctets when the duration is 0.

**Date/time Overview:**

The pre-analysis of user files concludes that the user data was collected between 2/1/2013 to 2/28/2013. There is a total of 4 weeks this month, but for the project first two weeks considered for analysis. The first week which is recognized for analysis, is from 2/4/2013 to 8/2/2013 (Monday to Friday) and the second week from 2/11/2013 to 2/15/2013(Monday to Friday).

The real first packet is given in epoch milliseconds, and it is converted to date-time before proceeding it for slicing to obtain window time frame. The hours of the day considered for time windows are 8.00.00am to 5.00.00pm in a day and Monday, Tuesday, Wednesday, Thursday and Friday in a week. So basically, the time windows of 10, 227, and 300 seconds are created for each user for 5 days of the week from 8 am to 5 pm for day.

**Time windows:**

The total number of windows for given widow interval:

1. 10 seconds – there are 3240 windows in a day (8.00.00am to 5.00.00pm) or 16200 windows in a week (Mon, Tues, Wed, Thurs, Fri).
2. 227 seconds – there are 143 windows in a day (8.00.00am to 5.00.00pm) or 715 windows in a week (Mon, Tues, Wed, Thurs, Fri).
3. 5 Minutes (300 seconds) – In the analysis and for computational ease, 5\*60 = 300 seconds are taken. There are 108 windows in a day (8.00.00am to 5.00.00pm) or 540 windows in a week (Mon, Tues, Wed, Thurs, Fri).

**Averaging octets per duration when duration is larger than window:**

There is the slicing of data performed on doctets data only if the duration of respective doctets is larger than the current window size. For example, if doctets are 100 for duration 20 seconds and window size is 10, then to fit the data correctly for one window, the program divides the doctets and duration into two parts as (50,10) and (50,10). In this first doctets, the duration is considering for the current window, and another is added in the next window. The doctets per duration is calculated for each pair, and then an average of each doctets per duration is calculated for each window.

**Weeks Storage data structure:**

The final calculated window data is stored in the 3D python list for a week inside the Storage class object. The arrangements for each time window are given below.

10 seconds: [ [ [], [] ... 3240 windows], [ [], [] ... 3240 windows] …5 days]

227 seconds: [ [ [], [] ... 143 windows], [ [], [] ... 3240 windows] …5 days]

227 seconds: [ [ [], [] ... 108 windows], [ [], [] ... 108 windows] …5 days]

The point to note here is that each window holds only one value, which is the average doctets per duration of that time frame. The data of each user is calculated according to the time window and stored in the ‘Storage’ class object in variable ‘window\_week1’ and ‘window\_week2’. This object further stored as an intermediary file in the processed\_file folder of the project using the pickle library of python. Pickle library is very efficient and fast in storing and retrieving class objects and therefore considered to store user data in intermediary files.

# Execution Steps 2 (Calculations)

The execution step 2 performs calculations on pre-processed intermediary files stored in the ‘file\_processed’ folder. It executes three times for each time window frame of 10, 227, and 300 seconds. In each execution, it fetches a pre-processed data file of a user and performs the calculation on the data and finally stores final data files in respective folders.

**Essential Parameters for Step 2:**

The execution step 2 accepts four necessary parameters that are explained below:

1. Save\_weeks: If set to True, then it will store the pre-processed window for week 1 and week 2 in the ‘Weeks’ folder of the project. The folder further has three subfolders named 10, 227, and 300 having data of time window for both weeks. It will create CSV files in which each row represents a user data, and each cell represents the average of doctects per duration falling in the respective time window frame.
2. Save\_spearman: If set to True, then the program will store the Spearman correlation rank coefficient in the ‘spearman’ folder of the project. It will have subfolders named 10,227 and 300, and each folder will have data of R1a2a, R1a2b, R2a2b spearman correlation coefficient.
3. Save\_z: if set to True, then it will store the calculated Z value file in the ‘ZValue’ folder. The folder again has 3 subfolders for 3-time windows, and each is having a CSV file of Z value table.
4. Save\_p: It will store the final calculated P values in a folder PValues with three subfolders for three-time windows and each having the final calculated P-value table as a CSV file.

The first three params ‘save\_week’, ‘save\_spearman,’ ‘save\_z’ are optional and can be set to False to reduce the project execution time. The fourth param save\_p must be set to true as it produces the final calculated P values table to the analysis of user data.

**Coefficient/Value Calculations:**

There are three types of calculations performed on each data pre-processed file. The comparisons algorithm has n2 complexity as it compares each user data file with all data files using 2 for loops. Suppose that the program considers two use data pre proceed files named as UserA and UserB then the calculation performed in sequence by the program is given below.

1. **Spearman calculation:**

Spearman correlation rank coefficient calculation: The program uses the panda’s spearman rank correlation coefficient to calculate the coefficient between two weeks of data for two users.

R1a2a: UserA\_week1, UserA\_week2

R1a2b: UserA\_weeek1, UserB\_week2

R2a2b: UserA\_week2, UserB\_week2

**Weeks data:**

|  |  |  |
| --- | --- | --- |
| 10 sec |  |  |
| 227 sec |  |  |
| 5 min |  |  |

**Assumptions on Spearman Correlation Coefficient data:**

1. If UserA does not have data for a window, but UserB has data, then 0 is inserted in the UserA window frame.
2. If UserA has data in a time window, but UserB does not have then 0 in inserted in UserB corresponding time window frame.
3. If both UserA and UserB does not have data for a time window, then those time windows have been discarded.

**Assumptions on Spearman Correlation Coefficient Value:**

1. If the calculated coefficient value is NaN (Not a values), which is a case when one of the user’s week data is either empty or all values are the same. In this case, NaN is replaced by 0.0001
2. If the coefficient calculated value is 1, then it is replaced by 0.9999 to avoid divide by zero error in the calculation of z value.
3. If the coefficient calculated is -1, then it is replaced by -0.9999 to prevent failure in the computation of z value.
4. If the coefficient computed value is 0, then it is replaced by 0.0001, a minimal substitution instead of 0 to get meaningful z value in the calculation.

**Spearman calculated values:**

|  |  |  |  |
| --- | --- | --- | --- |
| 10 sec |  |  |  |
| 227 sec |  |  |  |
| 5 min |  |  |  |

1. **Z Value calculations:**

In the project documentation, a formula is given to calculate Z values. A function ZFormula is written in python to calculate z values the same as given in project documentation. It accepts N, R1a2a, R1a2b, R2a2b as parameters to perform the calculation of z value. The R1a2a, R1a2b, and R2a2b are calculated as per the process described in above section 1. The value of N is calculated as given below.

N = number of windows in a week

Total seconds in a week = 5 days \* 9 hours a day \* 60 minutes an hour \* 60 seconds a minute

= 16200

10 second window N = 16200/10 = 1620

227-second window N = 71.36

5 minutes window (300 seconds) N = 54

**Calculated Z values:**

|  |  |
| --- | --- |
| 10 sec |  |
| 227 sec |  |
| 5 min |  |

1. **P Value:**

A function ‘PFunction’ is writen in computation.py script to perform the computation for final P-value. It accepts z as parameters and returns the final calculated P-value.

**Calculated P values:**

|  |  |
| --- | --- |
| 10 sec |  |
| 227 sec |  |
| 5 min |  |

# Execution Time Analysis

The final project is optimized for the best possible approach to reduce the overall execution time.

The time analysis of the complete project is given below.

1. The complete project execution takes on average of **approximately 46 minutes** to complete.
2. Data pre-processing takes approximately **43.35 minutes to complete**.
3. Calculation on the pre-processed file takes approximately **3.04 minutes to complete**.

The final log file given below can be referred for a detailed analysis of the time taken by each step of the project.



# THANK YOU