Fall 12-19-2016

Predicting User's Future Requests Using Frequent Patterns

Marc Nipuna Dominic Savio
San Jose State University

Predicting User's Future Requests Using Frequent Patterns

A Project
Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Marc Nipuna Dominic Savio

Fall 2016

The Designated Project Committee Approves the Project Titled

Predicting User's Future Requests Using Frequent Patterns

by

Marc Nipuna Dominic Savio

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE
SAN JOSE STATE UNIVERSITY

Dec 2016

ABSTRACT

Predicting User's Future Requests Using Frequent Patterns

by Marc Nipuna Dominic Savio

In this research, we predict User's Future Request using Data Mining Algorithm. Usage of the World Wide Web has resulted in a huge amount of data and handling of this data is getting hard day by day. All this data is stored as Web Logs and each web log is stored in a different format with different Field names like search string, URL with its corresponding timestamp, User ID's that helps for session identification, Status code, etc. Whenever a user requests for a URL there is a delay in getting the page requested and sometimes the request is denied. Our goal is to generate a Frequent Pattern Itemset on the Web Log we have chosen and after analyzing and processing the data we apply Apriori All algorithm with a minimum support to prune and improve the Frequent Pattern and thereby predict the User's Future Request which will help the user in successfully reaching out to the URL pages he has requested.

ACKNOWLEDGEMENTS

First, I would like to thank my Project Advisor, Dr. Tsau Young Lin for his constant support and guidance in this project and I thank him for believing in me. His contribution in this project is the stepping stones for the success of this project.

I would also take this opportunity to thank Dr. Robert Chun and Mr. Monish Prabu Chandran, for their advices and helpful thoughts during this project.

Lastly, I thank my parents and friends for always supporting me in this project and being the backbone of my success for the completion of my Master's program.

Table of Contents

1. Introduction……………………………………………………...…9

2. Background……………………………………………………..11

2.1 Data Mining……………………………………………………...11

2.2 Pattern discovery…………………………………………………11

2.3 Prediction……………………………………………………...11

3. Predicting User's Future Requests Using Frequent Patterns………….12

3.1 Algorithm……………………………………………………...14

3.2 Activity diagram…………………………………………………15

3.3 Use case diagram……………………………………………...16

3.4 Class diagram……………………………………………………17

3.5 Sequence diagram……………………………………………..18

3.6 Entity-relationship diagram…………………………………...19

3.7 Modules………………………………………………………..20

3.7.1 Dataset…………………………………………………………20

3.7.2 Data Preprocessing…………………………………………….22

3.7.3 Data Analysis………………………………………………….26

3.7.4 Pattern Discovery……………………………………………...28

3.7.5 Pattern Analysis………………………………………………30

4. System Requirements…………………………………………...33

4.1 Software…………………………………………………………33

4.2 Hardware………………………………………………………33

5. Software Description…………………………………………...34

5.1 Java………………………………………………………………34

5.2 Eclipse……………………………………………………………34

5.3 MS Excel…………………………………………………………34

5.4 Python…………………………………………………………....35

5.5 Canopy……………………………………………………………...35

6. Results……………………………………………..………….36

7. Conclusion……………………………………………..………….37

8. Future work……………………………………………….………38

References……………………………..………………………………..39

Appendix...…………………………………………………………….40

Additional Screenshots....……………………………………………….40

LIST OF TABLES

Table 1: Sample Candidate - 1 Itemset…………………………………………………………..28

Table 2: Sample Candidate - 2 Itemset…………………………………………………………..29

Table 3: Table 3: Sample Candidate-1 Itemset with Apriori…………………………………….30

Table 4: Table 3: Sample Candidate-1 Itemset with Apriori…………………………………….30

Table 5: Results with different min_support…………………………………………………….36

LIST OF FIGURES

Figure 1: Apriori Algorithm………………………………………………………………...14

Figure 2: Activity Diagram…………………………………………………………………15

Figure 3: Use Case Diagram…………………………………………………………...…...16

Figure 4: Class Diagram……………………………………………………………………17

Figure 5: Sequence Diagram………………………………………………………………..18

Figure 6: Entity - Relationship Diagram……………………………………………………19

Figure 7: Log File Sample………………………………………………………………….21

Figure 8: Preprocessed Log file…………………………………………………………….24

Figure 9: Status code diversity……………………………………………………………...26

Figure 10: Dataset Snippet in Text file……………………………………………………..40

Figure 11: Preprocessed Dataset in Excel…………………………………………………..41

Figure 12: URL as Numbers with sessions identified……………………………………...42

Figure 13: Minimum Support of 5%..43

Figure 14: Minimum Support of 10%..44

Figure 15: Minimum Support of 20%..45

Figure 16: Minimum Support of 25%..46

Figure 17: Minimum Support of 40%..47

Figure 18: Minimum Support of 50%..48

Figure 19: Minimum Support of 60%..49

Figure 20: Minimum Support of 70%..50

Figure 21: Minimum Support of 75%..51

Figure 22: Minimum Support of 80%..52

Chapter 1
Introduction

Web Log Usage Mining is growing day by day. We use a lot of data over the web by searching for answers for our questions. We do not get our answers always. In the Web Server, our search string is being searched and after the query is found the web displays us with the corresponding web pages. Sometimes, the web server fails to display the results and we change our search criteria to display the results.

But over time a user uses the same web pages like Gmail, Facebook and YouTube for instance. We can make the computer predict what pages will be required in future and can provide the users with their requested pages.

In this project, we will be predicting the User's Future request by studying the web log behavior and the key idea is to find a pattern that will help us to narrow down the results to a set of pages and display it to the user when it is needed.

Frequent Pattern mining uses data mining algorithms to find interesting, unexpected and useful patterns in the given database [1]. Here we will use Apriori algorithm which is undoubtedly the best algorithm to work on Frequent Pattern Itemset [1]. Apriori algorithm is applied on the transactions with a minimum support as an input to prune the results.

The Apriori algorithm helps mainly in finding incrementally frequent itemsets and associations and pattern mining is a subfield of Data Mining which has been active for a long time and is still very active [1].

We will use the database to generate the candidates for the Frequent Pattern Itemsets to find different patterns in the database which in end will be acted upon by the Apriori algorithm and help us predict the User's future request.

Chapter 2

Background

This chapter consists of the introduction to Data Mining, Pattern Discovery and Prediction

2.1 Data Mining

Data Mining is the process of discovering patterns in large data sets involving machine learning, artificial intelligence, DBMS and statistics. The main aim is to extract information from the available data and display the results in a structured format for human understanding [2].

2.2 Pattern Discovery

A pattern could be described as something that appears frequently in a database. Pattern discovery is a part of Data mining where we use different mining algorithms like path analysis, association rules, sequential patterns, clustering, etc as an effective process of Web Usage Mining where we can transform the web logs into knowledge to uncover the potential patterns [3].

2.3 Prediction

Prediction is a way to form patterns and allows us to predict the next event by studying the available input data. In this project, we will study the Web Usage Log for anomalies, apply frequent itemsets to find a pattern and prune the result by Apriori algorithm to predict the next/future event.

Chapter 3
Predicting User's Future Requests Using Frequent Patterns

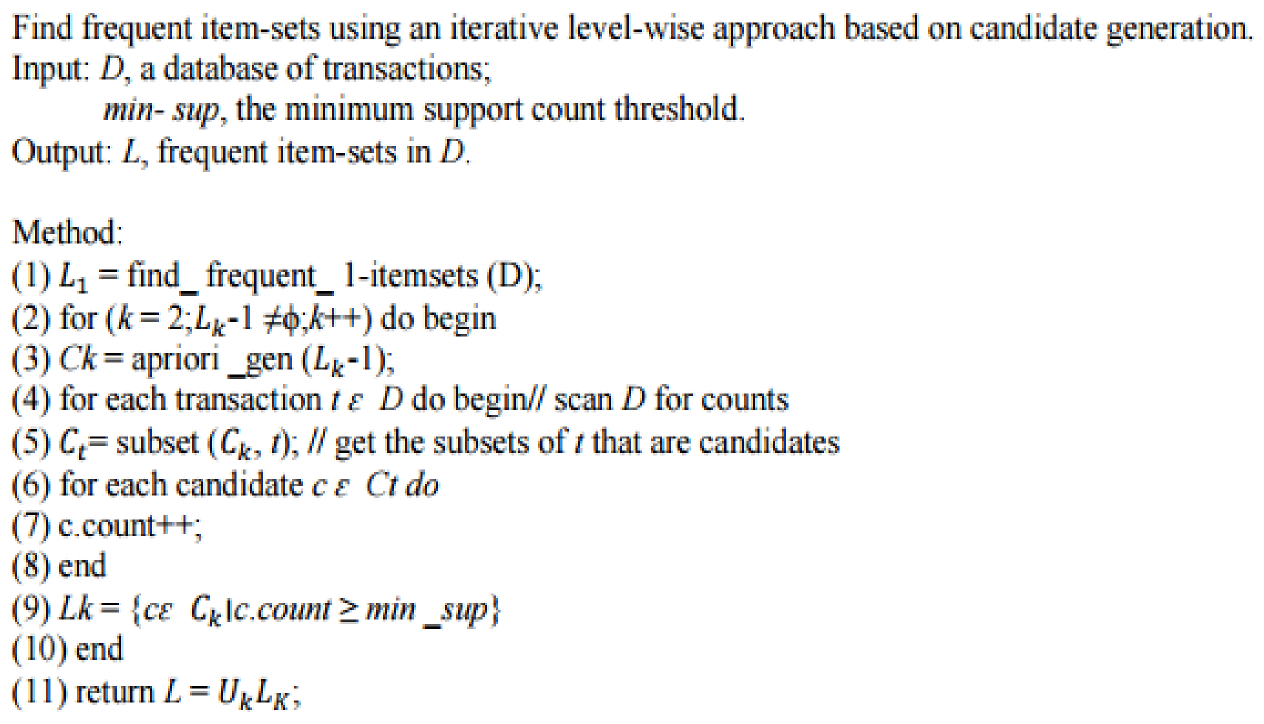
Prediction of User’s behavior starts with the first step of preprocessing the dataset as required for the data analysis. The original Web log is transformed from text format into a database format using Java program. The Data Analysis stage is responsible for cleaning the data so we can eliminate data that are not required in finding a pattern; There are many status codes in a web log. For Eg. “200” means the URL succeeded, “404” means error and “400” means Bad Request. So, we eliminate data with status codes 400, 404 and anything other than “200”.

We also identify the session in the web logs in the analysis phase. We find anomalies like, ‘there are many URL links succeeding in the weekends over the weekdays’ with the help of the status code. We find similar anomalies that are mentioned later in detail in the following chapters.

The URL links are changed into numbers with the respective to the order in which they appear by using the timestamp. This makes our step easier for Candidate Generation. The URL numbers are then divided into sessions and the Frequent Pattern Itemset comes into act. The candidate generation is done to find sequential patterns in the web log to find meaningful patterns of different lengths.

After generating the patterns, we use Apriori algorithm to prune the results of Frequent Pattern Itemsets. We use the minimum support to be 5% and this support value helps us to predict the more frequently used URL links. We start with 5% to produce many frequently used links with more lengths (say 10 consecutive links used frequently). By increasing the minimum support, we tend to prune the values and get a very accurate prediction of the User's Future Request. This project is carried out by using both Java and Python code. Java is used for Preprocessing the Web Log and Python for generating the candidates, discovering the frequent patterns and predicting the future request.

3.1 Algorithm



Find frequent item-sets using an iterative level-wise approach based on candidate generation.

Input: *D*, a database of transactions;

min- sup, the minimum support count threshold.

Output: *L*, frequent item-sets in *D*.

```
Method:
(1) $L_1$ = find_ frequent_ 1-itemsets (D);
(2) for ($k$ = 2;$L_k$-1 ≠ϕ;$k$++) do begin
(3) $Ck$ = apriori _gen ($L_k$-1);
(4) for each transaction $t$ ε  $D$ do begin// scan $D$ for counts
(5) $C_t$= subset ($C_k$, $t$); // get the subsets of $t$ that are candidates
(6) for each candidate $c$ ε  $Ct$ do
(7) c.count++;
(8) end
(9) $Lk$ = {$c$ε  $C_k$|$c.count \geq min$ _$sup$}
(10) end
(11) return $L = U_k L_K$;
```

Figure 1: Apriori Algorithm [4]

3.2 Activity Diagram

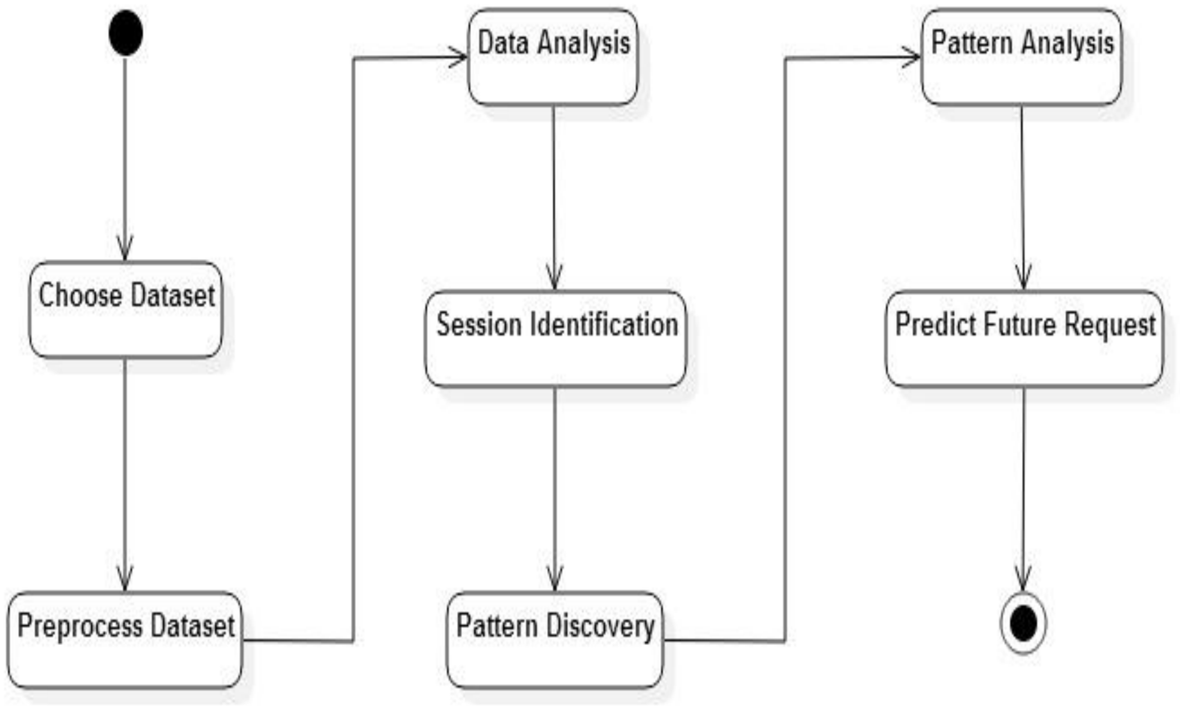


Figure 2: Activity Diagram

3.3 Use Case Diagram

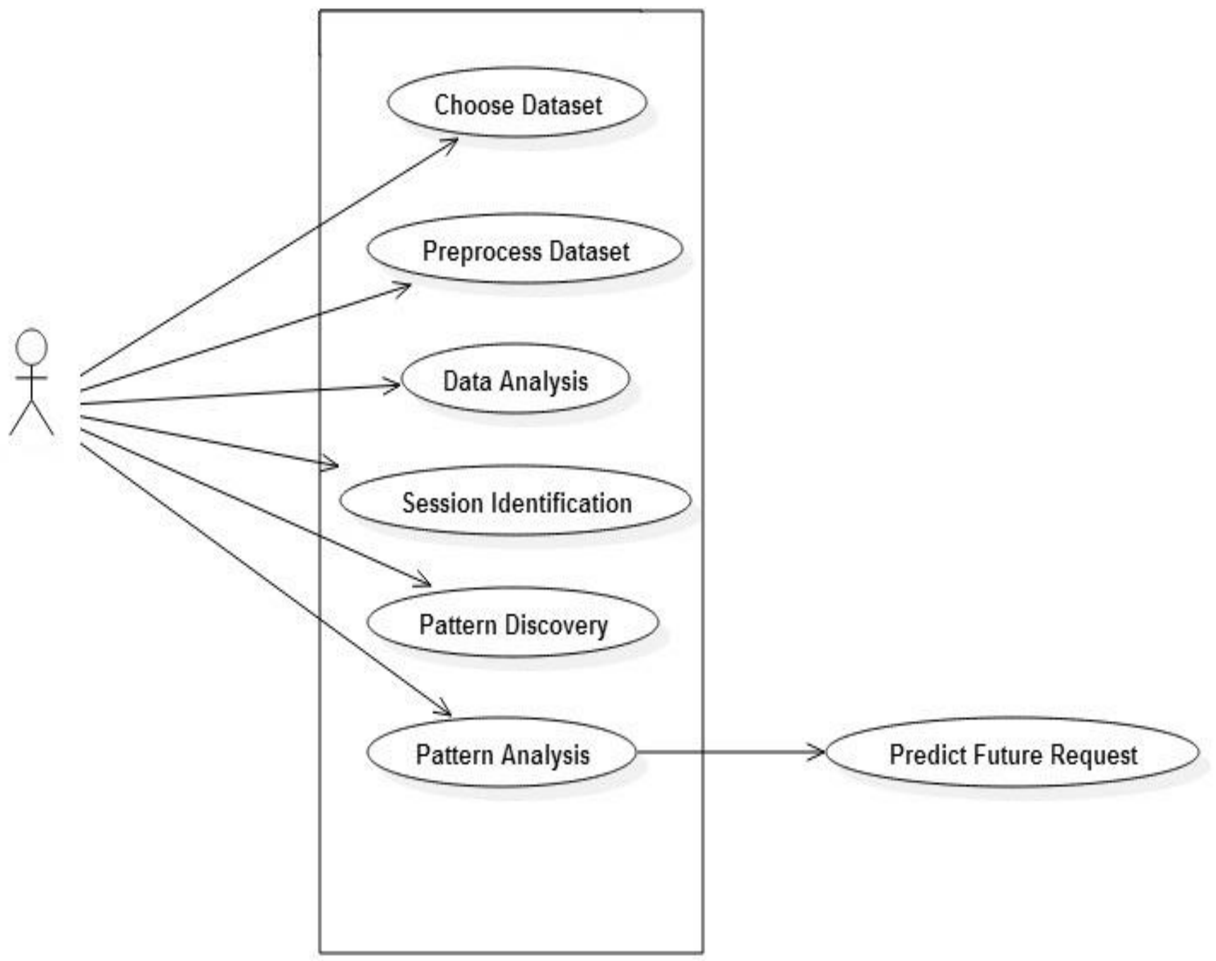


Figure 3: Use Case Diagram

3.4 Class Diagram

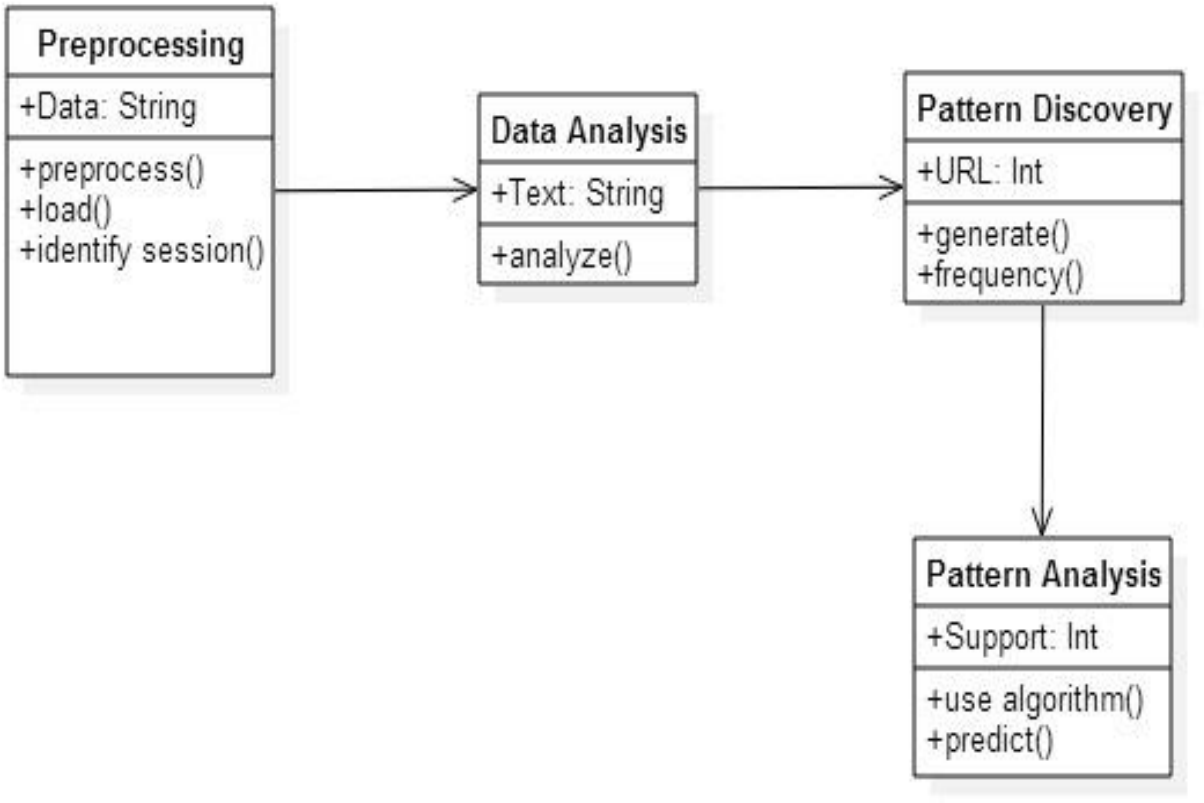


Figure 4: Class Diagram

3.5 Sequence Diagram

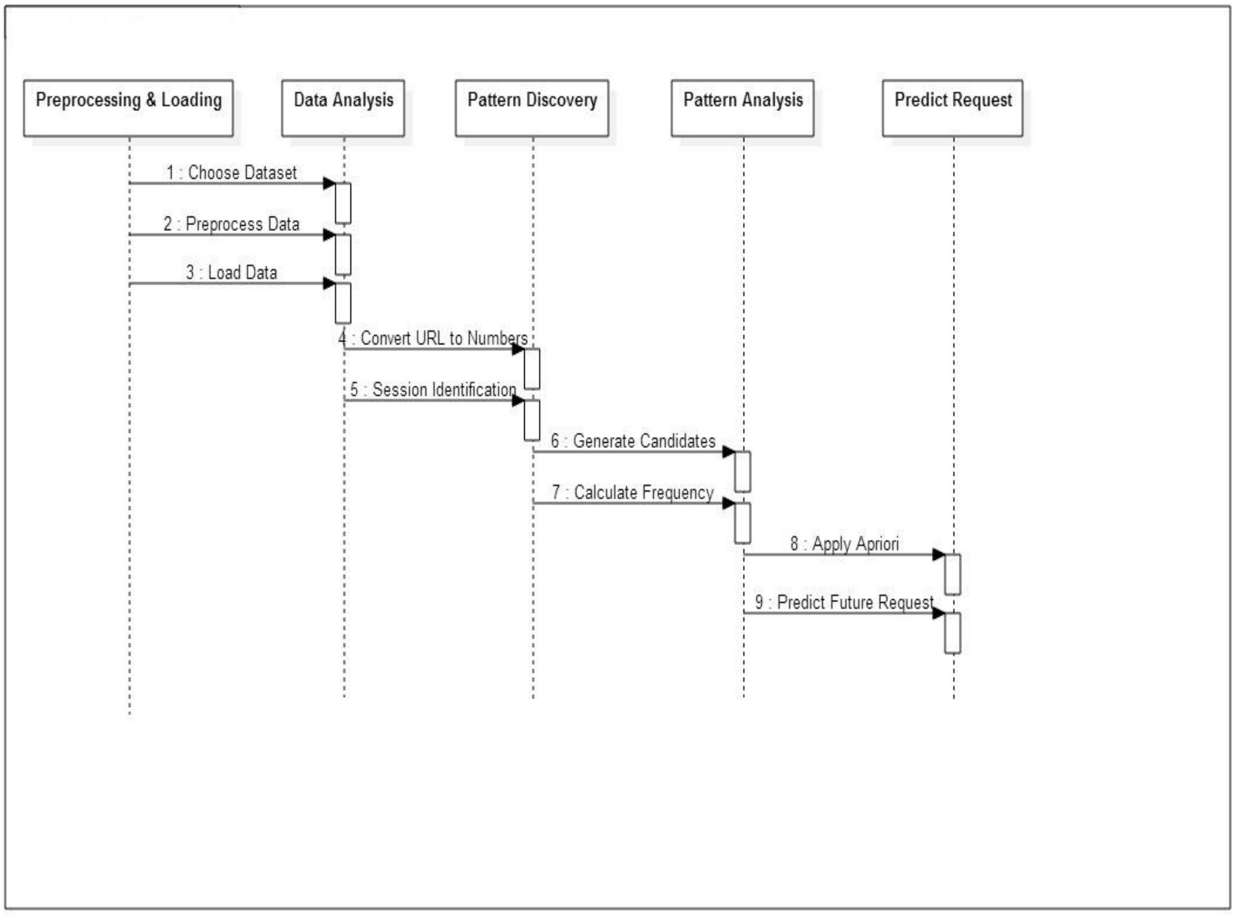


Figure 5: Sequence Diagram

3.6 Entity – Relationship Diagram

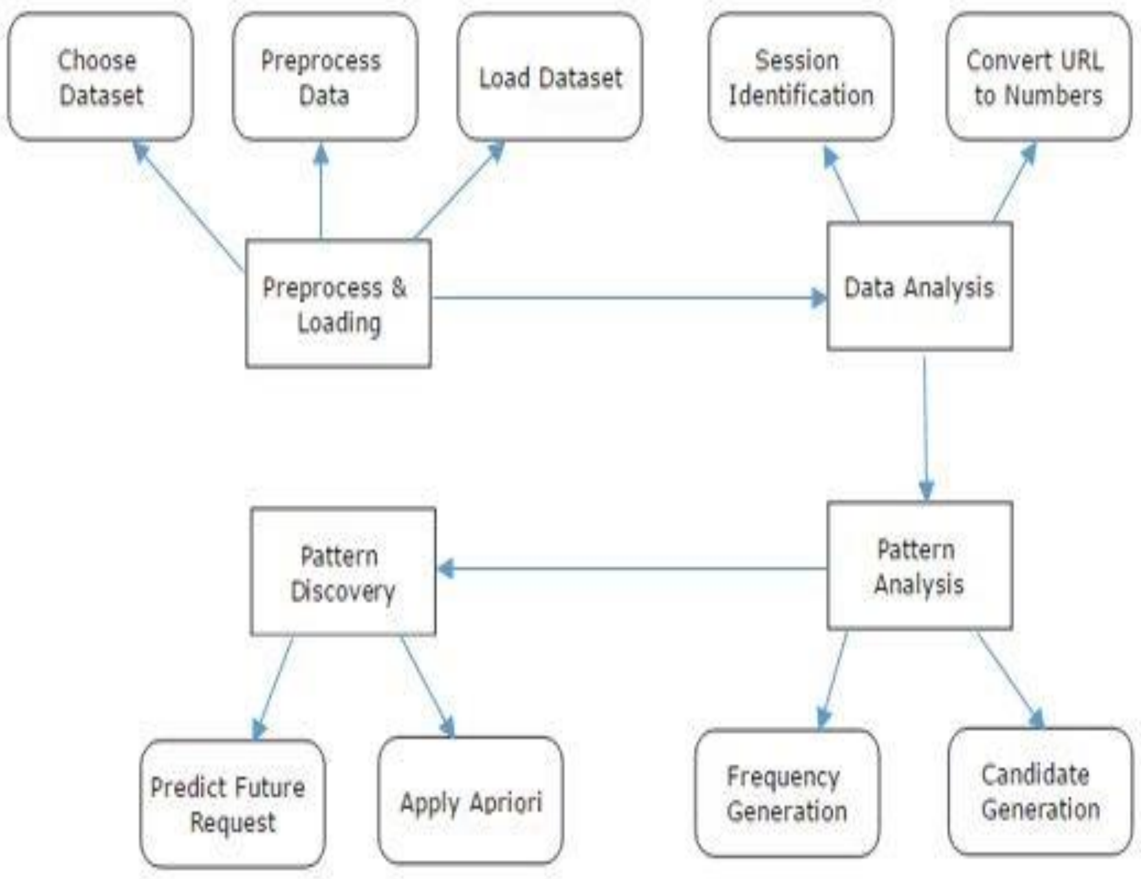


Figure 6: Entity - Relationship Diagram

3.7 Modules

3.7.1 Dataset

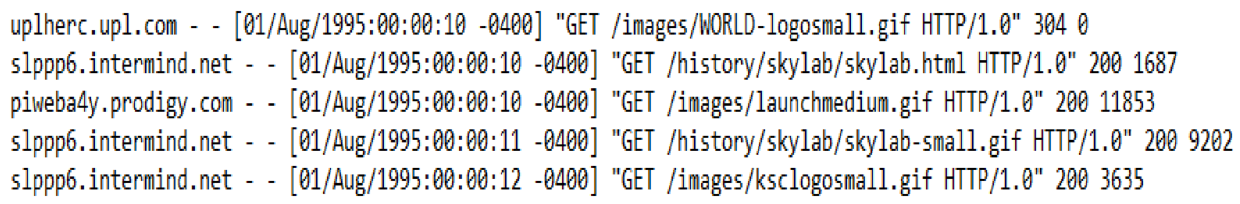
The dataset used in this project was obtained from HTTP requests to the NASA Kennedy Space Center WWW server in Florida: **http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html**

The format of the logs are as follows [5]:

1. **Host** making the request. A hostname when possible, otherwise the Internet address if the name could not be looked up [5]
2. **Timestamp** in the format "DAY MON DD HH:MM:SS YYYY", where DAY is the day of the week, MON is the name of the month, DD is the day of the month, HH:MM:SS is the time of day using a 24-hour clock, and YYYY is the year. The time zone is -0400 [5]
3. **Request** given in quotes [5]
4. **HTTP reply code** [5]
5. **Bytes** in the reply [5]

The dataset from NASA's HTTP request log is 21.8 MB compressed and 167.8 MB uncompressed [8]. There are almost 2 million records in the file and this volume of data is sufficient to perform data mining.

A snippet of the text file containing the web log is below:



```
uplherc.upl.com - - [01/Aug/1995:00:00:10 -0400] "GET /images/WORLD-logosmall.gif HTTP/1.0" 304 0
slppp6.intermind.net - - [01/Aug/1995:00:00:10 -0400] "GET /history/skylab/skylab.html HTTP/1.0" 200 1687
piweba4y.prodigy.com - - [01/Aug/1995:00:00:10 -0400] "GET /images/launchmedium.gif HTTP/1.0" 200 11853
slppp6.intermind.net - - [01/Aug/1995:00:00:11 -0400] "GET /history/skylab/skylab-small.gif HTTP/1.0" 200 9202
slppp6.intermind.net - - [01/Aug/1995:00:00:12 -0400] "GET /images/ksclogosmall.gif HTTP/1.0" 200 3635
```

Figure 7: Log File Sample

3.7.2 Data Preprocessing

The Log file in the snippet above should be loaded into a database so we can do the analysis and the Pattern Discovery steps.

- Here I have used a Java program to load the data from the text file and into an MS Excel file. As mentioned above, the data has 5 fields and the resulting data in the excel file is stored accordingly.

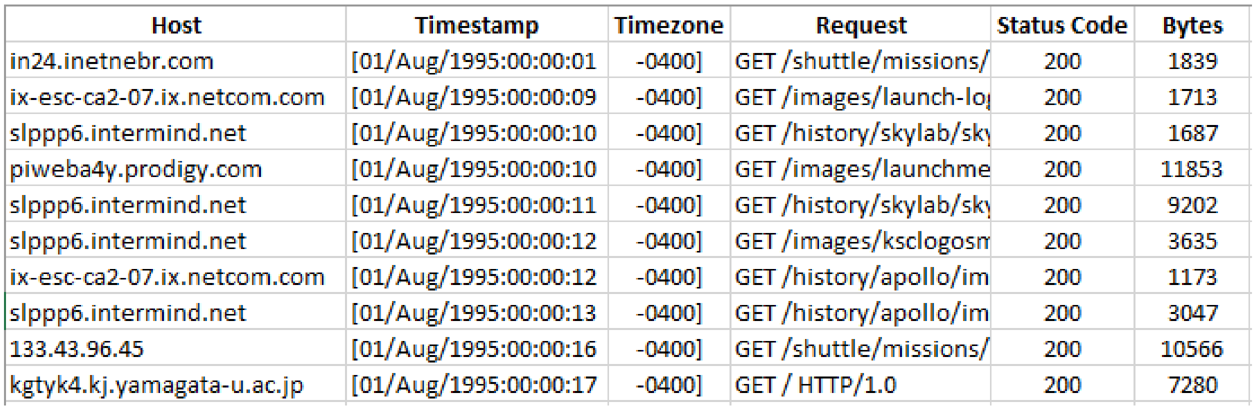
The Java Program is as follows:

```
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.LinkedHashSet;
import java.util.Set;

import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class PreprocessData {

    public static void main(String[] args) throws FileNotFoundException, IOException {
        if (args.length < 2 || args.length > 2) {
            System.out.println("Please enter testdata file and output file path");
            System.exit(1);
        }
```

```
try {
   // Open the file
   FileInputStream fstream = new FileInputStream(args[0]);
   DataInputStream in = new DataInputStream(fstream);
   BufferedReader br = new BufferedReader(new InputStreamReader(in));
   System.out.println(args[0]);

   Set<Object[]> dataSet = new LinkedHashSet<Object[]>();
   String strLine;
   //Read File Line By Line
   while ((strLine = br.readLine()) != null)   {
      //   split the line on your splitter(s)
      String[] splitted = strLine.split("\t", -1);
      dataSet.add(new Object[] {splitted[0], splitted[1], splitted[2], splitted[3],
       splitted[4]});
   }
   // Close the input stream
   in.close();

   // Create Excel workbook and set values in each cell
   XSSFWorkbook workbook = new XSSFWorkbook();
   XSSFSheet sheet = workbook.createSheet("User Data");
   int rownum = 0;
   for (Object[] data : dataSet) {
      Row row = sheet.createRow(rownum++);
      int cellnum = 0;
      for (Object value : data) {
         Cell cell = row.createCell(cellnum++);
         if (isDouble(value)) {
            value = Double.parseDouble((String) value);
         }
         if(value instanceof String) {
            cell.setCellValue((String)value);
         } else if(value instanceof Double) {
            cell.setCellValue((Double)value);
         }
      }
   }

   File outputXLSFile = new File(args[1]);
   outputXLSFile.getParentFile().mkdirs();
   outputXLSFile.createNewFile();
   FileOutputStream out =
         new FileOutputStream(outputXLSFile);
```

```
            workbook.write(out);
            out.close();
            workbook.close();
            System.out.println("Excel written successfully..");
        } catch (Exception e){ //Catch exception if any
            System.err.println("Error: " + e.getMessage());
        }
    }

    private static boolean isDouble(Object value) {
        try {
            Double.parseDouble((String) value);
        } catch(NumberFormatException e) {
            return false;
        } catch(NullPointerException e) {
            return false;
        }
        return true;
    }
}
```

After running the program, the data in the text file gets stored in the excel database as follows:



Host	Timestamp	Timezone	Request	Status Code	Bytes
in24.inetnebr.com	[01/Aug/1995:00:00:01	-0400]	GET /shuttle/missions/	200	1839
ix-esc-ca2-07.ix.netcom.com	[01/Aug/1995:00:00:09	-0400]	GET /images/launch-lo	200	1713
slppp6.intermind.net	[01/Aug/1995:00:00:10	-0400]	GET /history/skylab/sky	200	1687
piweba4y.prodigy.com	[01/Aug/1995:00:00:10	-0400]	GET /images/launchme	200	11853
slppp6.intermind.net	[01/Aug/1995:00:00:11	-0400]	GET /history/skylab/sky	200	9202
slppp6.intermind.net	[01/Aug/1995:00:00:12	-0400]	GET /images/ksclogosm	200	3635
ix-esc-ca2-07.ix.netcom.com	[01/Aug/1995:00:00:12	-0400]	GET /history/apollo/im	200	1173
slppp6.intermind.net	[01/Aug/1995:00:00:13	-0400]	GET /history/apollo/im	200	3047
133.43.96.45	[01/Aug/1995:00:00:16	-0400]	GET /shuttle/missions/	200	10566
kgtyk4.kj.yamagata-u.ac.jp	[01/Aug/1995:00:00:17	-0400]	GET / HTTP/1.0	200	7280

Figure 8: Preprocessed Log File

- From the picture above, we can come to decide to remove the ‘Timezone’, ‘Request’ and ‘Bytes’ columns from the database since they do not provide much information required for discovering frequent patterns
- Session Identification can be done with the timestamp in the data. Sessions are identified by taking the web log for every 30 minutes, so the web log is further classified into sessions considering the time in the timestamp and are divided
- After removal of the unwanted fields, we transfer the data to be analyzed

3.7.3 Data Analysis

The data was then uploaded into Splunk to find some anomalies. Some of the anomalies that were found are as follows:

1. The data can be categorized by their status code. We could find 6 different status codes in the web log.
   - Response code ‘200’ - OK
   - Response code ‘304’ - Not Modified
   - Response code ‘404’ - Not Found
   - Response code ‘403’ - Forbidden
   - Response code ‘501’ - Not Implemented
   - Response code ‘400’ - Bad Request

This was very much useful in removing data that did not have a status code of ‘200 – OK’ since the URLs which were not a success will not be used again by the user. So, we removed the logs without status code of ‘200’. The following graph will depict the diversity of the status code in the entire web log.

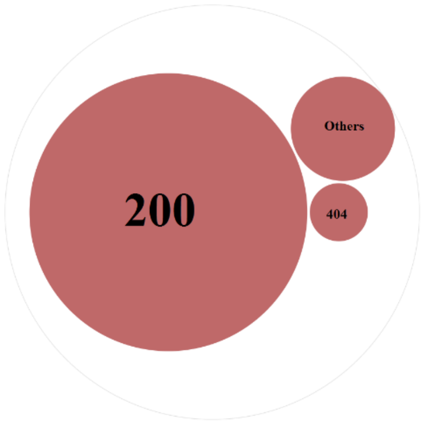


Figure 9: Status code diversity

2. By analyzing the timestamp of the data, we found that an average user searches more data during weekdays from Mondays through Thursdays.
3. Furthermore, the time in the timestamp shows us that more web logs were recorded between 10 AM and 4 PM. So mining logs at this particular time frame will be useful in the future.

3.7.4 Pattern Discovery

The data after analysis is filtered by the status code. So, data containing status code of ‘200’ is only considered here. In Pattern Discovery, we do the Candidate Generation for the Frequent Patterns.

A frequent pattern is a pattern that occurs frequently in a data set. The URLs that were strings in the text file and in the database, are each given a unique number which will make it easier to be handled for generating Frequent Pattern Itemset.

A small example of what Candidate Generation is as follows: Consider the following numbers as URLs that appear according to the timestamps under different sessions:

T1 → 1, 3, 6

T2 → 1, 4, 6

T3 → 1, 3, 6

The following table shows the candidates (itemsets) generated with their frequencies.

Table 1: Sample Candidate-1 Itemset

Itemset	Frequency
{1}	3
{3}	2
{4}	1
{6}	3

The following table illustrates the 2-candidate itemset generation with its frequencies. We can notice that itemset {3,4} has frequency of 0 and will be eliminated from the candidate generation.

Table 2: Sample Candidate-2 Itemset

Itemset	Frequency
{1,3}	2
{1,4}	1
{1,6}	3
{3,4}	0
{3,6}	2
{4,6}	1

This process will continue till n-candidate itemset is generated. This candidate generation is a pattern found in the web log.

3.7.5 Pattern Analysis

The candidate generation and its itemsets are evaluated in Pattern Analysis. We use Apriori algorithm to prune the generated candidate itemsets. We use a minimum support value for the frequency to eliminate values that don't have the required minimum support.

In the following table, we have a minimum support of 2, so we prune the itemsets and remove candidates that we generated with frequency less than 2.

Table 3: Sample Candidate-1 Itemset with Apriori

Itemset	Min_Sup =2
{1}	3
{3}	2
{6}	3

As seen above we prune itemsets that are less than the minimum support

Table 4: Sample Candidate-2 Itemset with Apriori

Itemset	Min_Sup =2
{1,3}	2
{1,6}	3
{3,6}	2

The steps mentioned above is done by our program in Python. The code used for predicting the future request is as follows:

```
from pymining import seqmining

mapping = {'a':'1', 'b':'2', 'c':'3', 'd':4, 'e':5, 'f':6, 'g':7, 'h':8, 'i':9, 'j':10, 'k':11, 'l':12,
'm':13, 'n':14, 'o':15, 'p':16, 'q':17, 'r':18, 's':19, 't':20, 'u':21, 'v':22, 'w':23, 'x':24,
'y':25, 'z':26}

seqs = list(line.strip() for line in open('WebData.txt'))

freq = seqmining.freq_seq_enum(seqs, 2557)

maxLen = 0
for item in freq:
        if len(item[0]) > maxLen:
maxLen = len(item[0])

f= open('Support.txt', 'w')
for item in freq:
        s = ''
        for char in item[0]:
                print char, type(item)
                s += str(mapping[str(char)]) + ","
                f.write("Pattern found : %s\t\tSupport = %s\n" % ([s.rstrip(',')],
                item[1]))
f.write('\n')
f.close()

f1= open('Length.txt', 'w')
for item in freq:
        s = ''
        for char in item[0]:
                print char, type(item)
                s += str(mapping[str(char)]) + ","
                f1.write("Pattern found : %s\t\tLength = %s\n" % ([s.rstrip(',')],
                len(item[0])))
f1.write('\n')
f1.close()
```

We import the pymining package and with the above code we can read the web logs which are in the form of numbers and then calculate the frequent pattern itemsets and in turn we discover the frequently used links by pruning them with Apriori Algorithm [6]. Initially, we keep the support at 50% where we get a lot of frequent patterns since the support is low. As we increase the support value to 20%, 25%, 40% and so on, we get more accurate prediction of the next URL. While keeping support at 10% we get more than 15,000 frequent patterns and with support at 80% which is very high, we get 15 frequent patterns. 80% is almost the most frequently used URL links.

CHAPTER 4

System Requirements

4.1 Software Requirements

• Operating System: Windows

• Programing Language: Java, Python

• IDE: Eclipse, Enthought Canopy

• Data Base: MS Excel

4.2 Hardware Requirements

• CPU: Minimum 2.4 GHz

• Hard Disk: Minimum 150 GB

• RAM: Minimum 4 GB

Chapter 5
Software Description

5.1 Java

Java is a programming language that widely works across all platforms. It was designed to have the 'look and feel' of C++, but very much simpler and enforces an object-oriented programming model. Java code is robust and since its object-oriented, an object can take advantage of being part of a class of objects and inherit code that is common to the class. A method can be thought of as one of the object's capabilities or behaviors [6].

5.2 Eclipse

Eclipse is an IDE mostly used for Java. Its workspace and plug-ins are very much suited to customize the development environment. Eclipse is written mostly in Java and is an open-source platform and is used to develop applications for various programming languages [7].

5.3 MS Excel

Microsoft Excel is a spreadsheet with all the basic features developed by Microsoft for all the operating systems. It can be used for graphing tools, visualization and creating macros. Microsoft Excel 2016 is the latest version of Excel and is compatible with Windows 10, etc [8].

5.4 Python

Python is a most widely used Programming language, that have very fewer lines of code when compared to the other major programming languages used like Java, C++, etc. It has a dynamic type system and automatic memory management with its own library making it one of the most preferred languages to be used by developers [9].

5.5 Canopy

Enthought Canopy is a Python IDE for scientific and analytic computing available free for students and under a commercial license. Since it is used for Python, adding new libraries into the IDE is very easy and an efficient tool to write code.

Chapter 6
Results

The project to predict User's Future Request is completed. The main objective is to analyze the web usage log, look for anomalies and generate the candidate itemsets which will help in finding a frequent pattern. We used 165 MB of the NASA dataset to find a pattern. The data was divided over time and its session were identified with the help of the timestamps. We then applied Apriori algorithm to the Frequent Pattern Itemsets to study the user's behavior and we predicted the User's future request. We initially had a minimum support of 5% and then gradually increased it to 80%. The more the minimum support value, the more accurate the prediction will be.

Table 5: Results with different min_support

Min_Support	Maximum Candidate Generation Length	Number of Itemsets Found
5%	Maximum Length = 12	40072
10%	Maximum Length = 11	15403
20%	Maximum Length = 9	4015
25%	Maximum Length = 8	2257
40%	Maximum Length = 6	512
50%	Maximum Length = 5	190
60%	Maximum Length = 5	70
70%	Maximum Length = 4	33
75%	Maximum Length = 3	24
80%	Maximum Length = 3	15

Chapter 7
Conclusion

The research paper allowed us to implement prediction on User's Future Request using Frequent Pattern Itemsets and Apriori algorithm to prune the patterns with a minimum support value that increases the accuracy of the prediction.

We implemented an approach where we used both Frequent Pattern Mining and Apriori Algorithm instead of using both methods separately. This approach is far better than implementing the prediction separately by Frequent Pattern Mining and by Apriori Algorithm.

We used Java for the preprocessing step and then with the use of MS Excel and Notepad++, we modified the data required to use them for the main prediction program. We used Python to generate the candidates and the frequent pattern itemsets and finally predict the future URL links.

Chapter 8
Future Work

- This project can be tested with a different web log dataset like AOL, MSN that has more Frequent patterns
- As mentioned in the data analysis module, the more number of URL links were recorded on a set of days of the week and during a particular period of the day. We can do the prediction based on these days and time
- We eliminated the URLs without status code of 200. We can include the other URLs and predict the user's behavior
- Apriori algorithm was used in this project, we can use other complex algorithms like SPADE, PrefixSpan, etc. and compare the performance with each other
- This approach can be implemented using Big Data Tools like Hadoop since we had a space limitation with MS Excel

REFERENCES

[1] An introduction to frequent pattern mining - The Data Mining Blog. [Online]. Available: http://data-mining.philippe-fournier-viger.com/introduction-frequent-pattern-mining/

[2] Wikipedia, 'Data Mining', 2016. [Online]. Available: https://en.wikipedia.org/wiki/Data_mining

[3] S4904131136.pdf. [Online]. Available: http://www.ijera.com/papers/Vol4_issue9/Version%204/S4904131136.pdf

[4] IJCAET2014030304.pdf. [Online]. Available: http://ijcaet.net/documents/vol3issue3/IJCAET2014030304.pdf

[5] NASA-HTTP - Two Months of HTTP Logs from the KSC-NASA WWW Server. [Online]. Available: http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html

[6] Pymining at master - bartdag/pymining – Github. [Online]. Available: https://github.com/bartdag/pymining/tree/master/pymining

[7] What is Java? - Definition from WhatIs.com. [Online]. Available: http://searchsoa.techtarget.com/definition/Java

[8] Wikipedia, 'Eclipse (software)', 2016. [Online]. Available: https://en.wikipedia.org/wiki/Eclipse_(software)

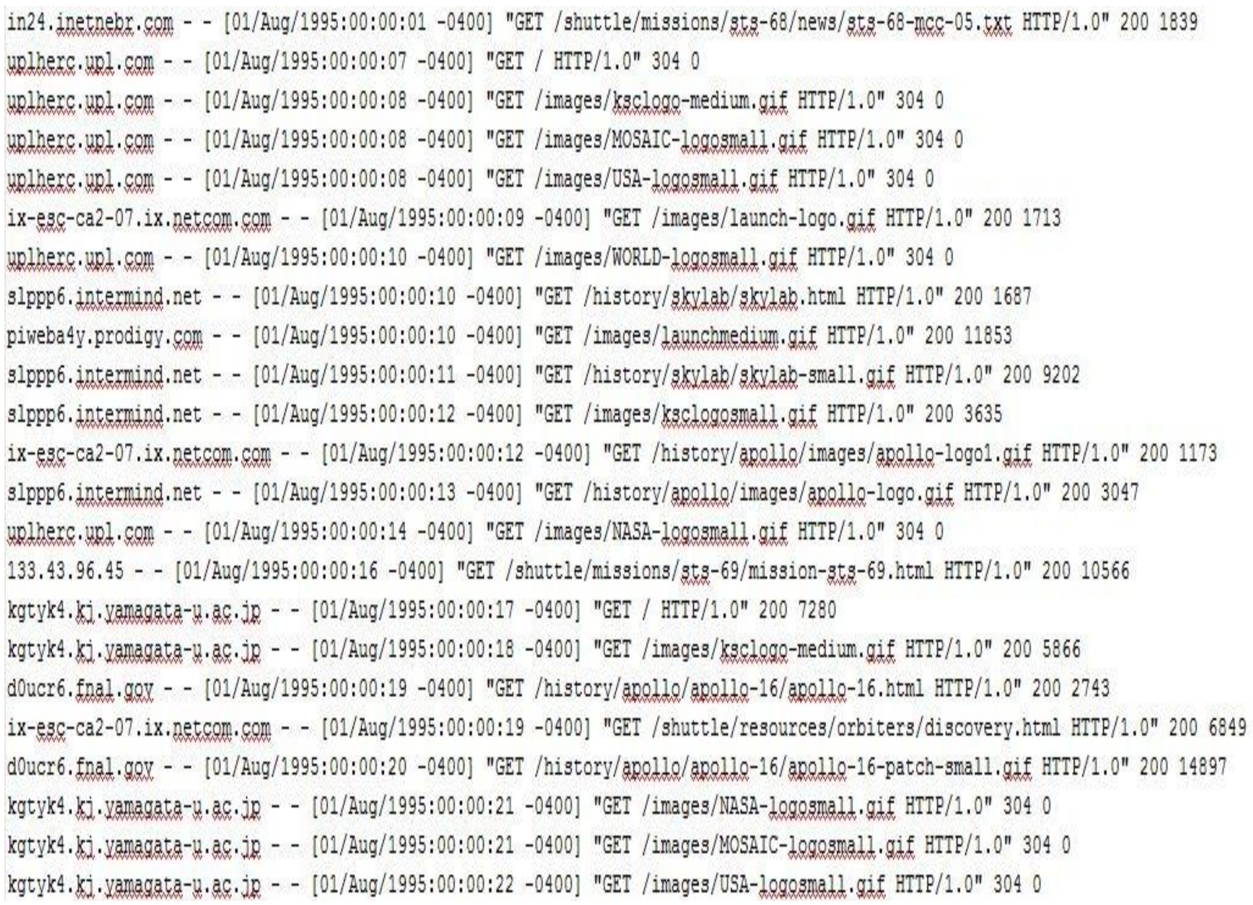
[9] Wikipedia, 'Microsoft Excel', 2016. [Online]. Available: https://en.wikipedia.org/wiki/Microsoft_Excel

[10] Wikipedia, 'Python (Programming language)', 2016. [Online]. Available: https://en.wikipedia.org/wiki/Python_(programming_language)

[11] Wikipedia, 'Enthought', 2016. [Online]. Available: https://en.wikipedia.org/wiki/Enthought

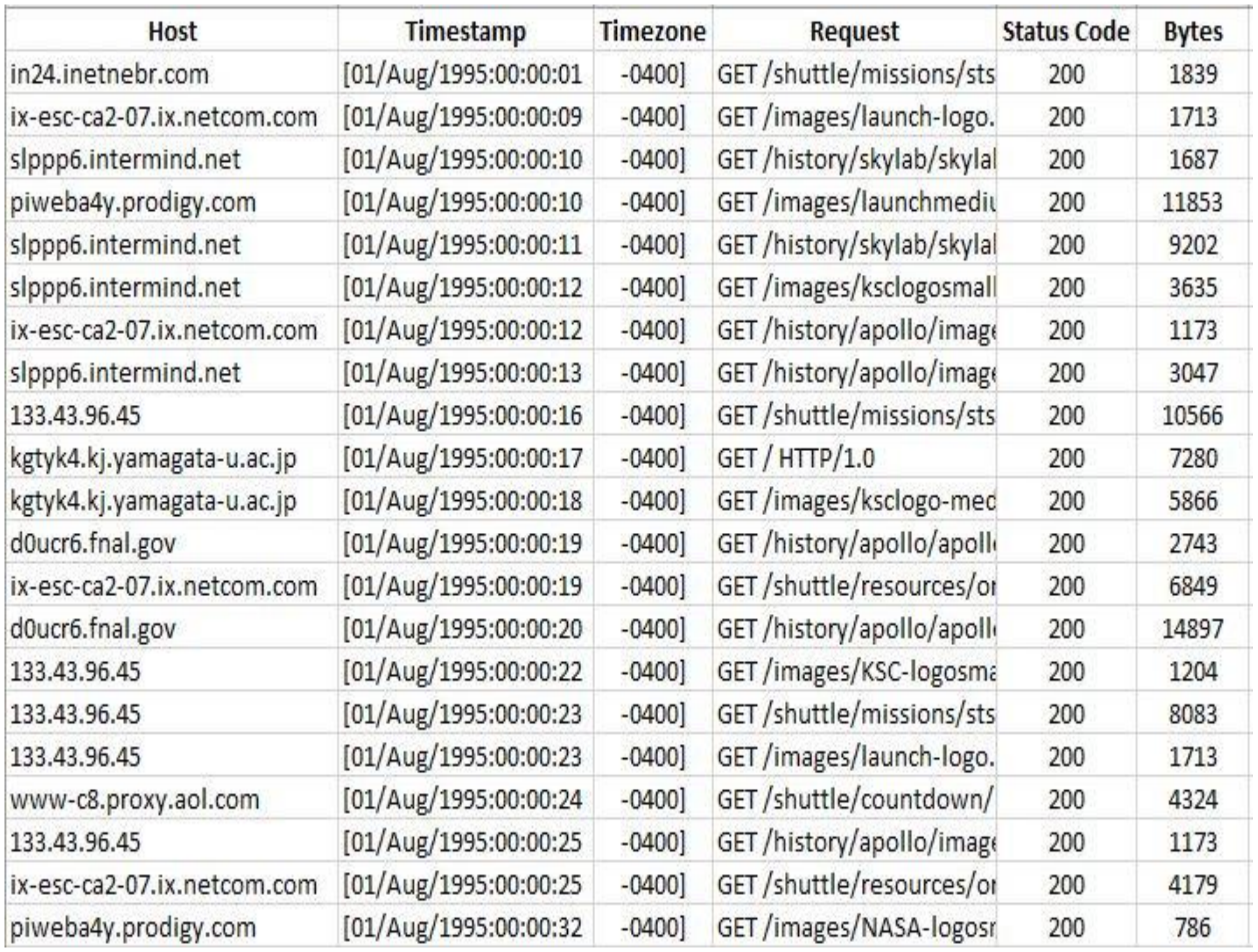
Appendix

Additional Screen Shots



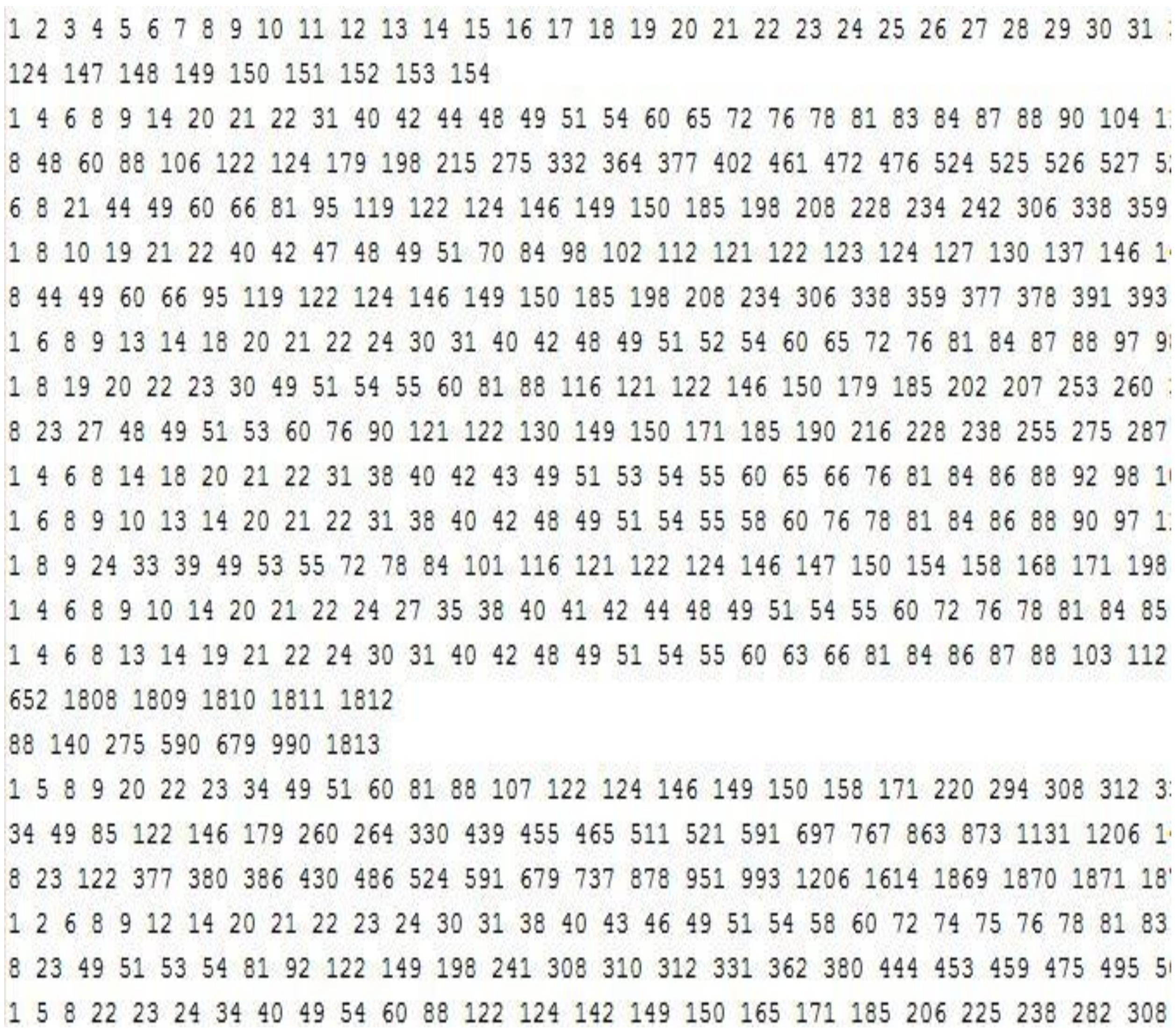
```
in24.inetnebr.com - - [01/Aug/1995:00:00:01 -0400] "GET /shuttle/missions/sts-68/news/sts-68-mcc-05.txt HTTP/1.0" 200 1839
uplherc.upl.com - - [01/Aug/1995:00:00:07 -0400] "GET / HTTP/1.0" 304 0
uplherc.upl.com - - [01/Aug/1995:00:00:08 -0400] "GET /images/ksclogo-medium.gif HTTP/1.0" 304 0
uplherc.upl.com - - [01/Aug/1995:00:00:08 -0400] "GET /images/MOSAIC-logosmall.gif HTTP/1.0" 304 0
uplherc.upl.com - - [01/Aug/1995:00:00:08 -0400] "GET /images/USA-logosmall.gif HTTP/1.0" 304 0
ix-esc-ca2-07.ix.netcom.com - - [01/Aug/1995:00:00:09 -0400] "GET /images/launch-logo.gif HTTP/1.0" 200 1713
uplherc.upl.com - - [01/Aug/1995:00:00:10 -0400] "GET /images/WORLD-logosmall.gif HTTP/1.0" 304 0
slppp6.intermind.net - - [01/Aug/1995:00:00:10 -0400] "GET /history/skylab/skylab.html HTTP/1.0" 200 1687
piweba4y.prodigy.com - - [01/Aug/1995:00:00:10 -0400] "GET /images/launchmedium.gif HTTP/1.0" 200 11853
slppp6.intermind.net - - [01/Aug/1995:00:00:11 -0400] "GET /history/skylab/skylab-small.gif HTTP/1.0" 200 9202
slppp6.intermind.net - - [01/Aug/1995:00:00:12 -0400] "GET /images/ksclogosmall.gif HTTP/1.0" 200 3635
ix-esc-ca2-07.ix.netcom.com - - [01/Aug/1995:00:00:12 -0400] "GET /history/apollo/images/apollo-logo1.gif HTTP/1.0" 200 1173
slppp6.intermind.net - - [01/Aug/1995:00:00:13 -0400] "GET /history/apollo/images/apollo-logo.gif HTTP/1.0" 200 3047
uplherc.upl.com - - [01/Aug/1995:00:00:14 -0400] "GET /images/NASA-logosmall.gif HTTP/1.0" 304 0
133.43.96.45 - - [01/Aug/1995:00:00:16 -0400] "GET /shuttle/missions/sts-69/mission-sts-69.html HTTP/1.0" 200 10566
kgtyk4.kj.yamagata-u.ac.jp - - [01/Aug/1995:00:00:17 -0400] "GET / HTTP/1.0" 200 7280
kgtyk4.kj.yamagata-u.ac.jp - - [01/Aug/1995:00:00:18 -0400] "GET /images/ksclogo-medium.gif HTTP/1.0" 200 5866
d0ucr6.fnal.gov - - [01/Aug/1995:00:00:19 -0400] "GET /history/apollo/apollo-16/apollo-16.html HTTP/1.0" 200 2743
ix-esc-ca2-07.ix.netcom.com - - [01/Aug/1995:00:00:19 -0400] "GET /shuttle/resources/orbiters/discovery.html HTTP/1.0" 200 6849
d0ucr6.fnal.gov - - [01/Aug/1995:00:00:20 -0400] "GET /history/apollo/apollo-16/apollo-16-patch-small.gif HTTP/1.0" 200 14897
kgtyk4.kj.yamagata-u.ac.jp - - [01/Aug/1995:00:00:21 -0400] "GET /images/NASA-logosmall.gif HTTP/1.0" 304 0
kgtyk4.kj.yamagata-u.ac.jp - - [01/Aug/1995:00:00:21 -0400] "GET /images/MOSAIC-logosmall.gif HTTP/1.0" 304 0
kgtyk4.kj.yamagata-u.ac.jp - - [01/Aug/1995:00:00:22 -0400] "GET /images/USA-logosmall.gif HTTP/1.0" 304 0
```

Figure 10: Dataset Snippet in Text File



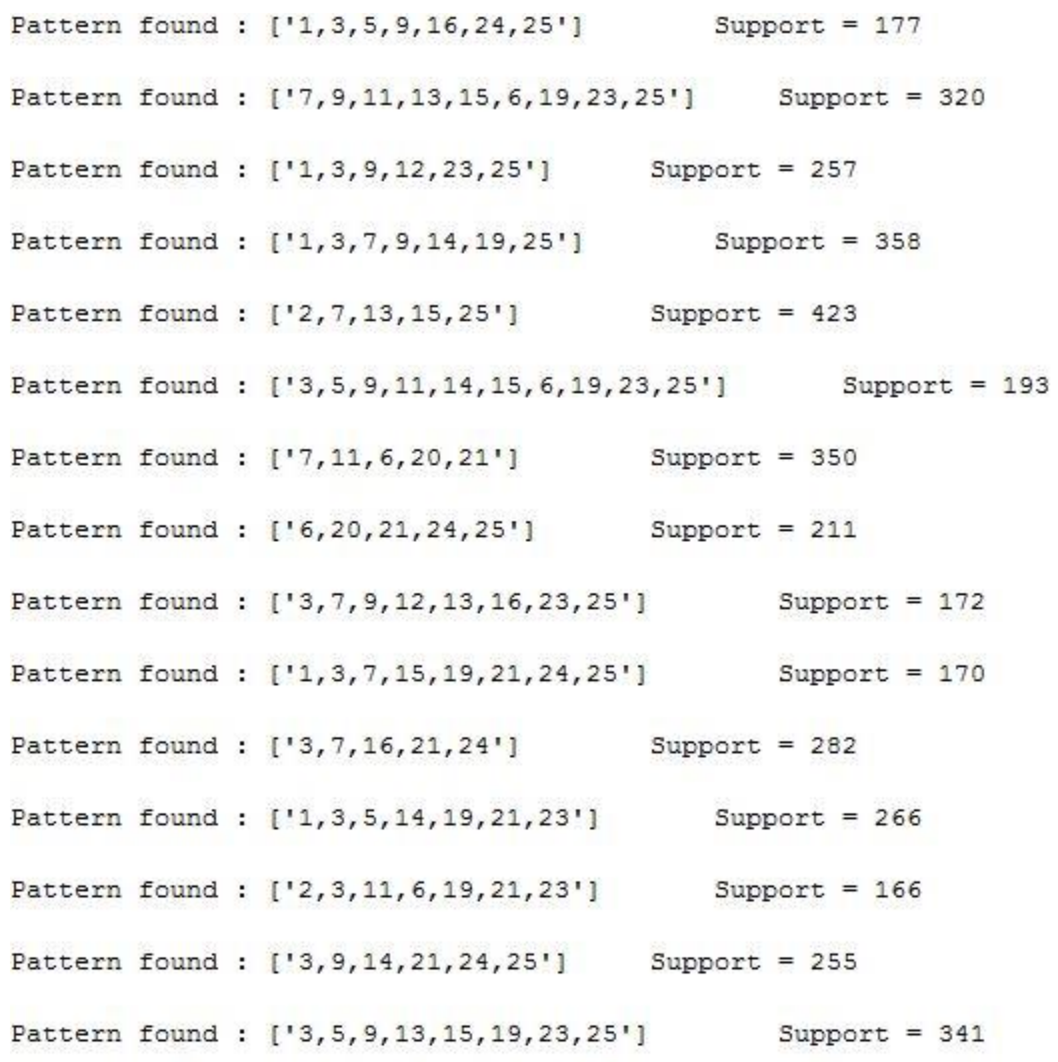
Host	Timestamp	Timezone	Request	Status Code	Bytes
in24.inetnebr.com	[01/Aug/1995:00:00:01	-0400]	GET /shuttle/missions/sts	200	1839
ix-esc-ca2-07.ix.netcom.com	[01/Aug/1995:00:00:09	-0400]	GET /images/launch-logo.	200	1713
slppp6.intermind.net	[01/Aug/1995:00:00:10	-0400]	GET /history/skylab/skylal	200	1687
piweba4y.prodigy.com	[01/Aug/1995:00:00:10	-0400]	GET /images/launchmediu	200	11853
slppp6.intermind.net	[01/Aug/1995:00:00:11	-0400]	GET /history/skylab/skylal	200	9202
slppp6.intermind.net	[01/Aug/1995:00:00:12	-0400]	GET /images/ksclogosmall	200	3635
ix-esc-ca2-07.ix.netcom.com	[01/Aug/1995:00:00:12	-0400]	GET /history/apollo/image	200	1173
slppp6.intermind.net	[01/Aug/1995:00:00:13	-0400]	GET /history/apollo/image	200	3047
133.43.96.45	[01/Aug/1995:00:00:16	-0400]	GET /shuttle/missions/sts	200	10566
kgtyk4.kj.yamagata-u.ac.jp	[01/Aug/1995:00:00:17	-0400]	GET / HTTP/1.0	200	7280
kgtyk4.kj.yamagata-u.ac.jp	[01/Aug/1995:00:00:18	-0400]	GET /images/ksclogo-mec	200	5866
d0ucr6.fnal.gov	[01/Aug/1995:00:00:19	-0400]	GET /history/apollo/apoll	200	2743
ix-esc-ca2-07.ix.netcom.com	[01/Aug/1995:00:00:19	-0400]	GET /shuttle/resources/or	200	6849
d0ucr6.fnal.gov	[01/Aug/1995:00:00:20	-0400]	GET /history/apollo/apoll	200	14897
133.43.96.45	[01/Aug/1995:00:00:22	-0400]	GET /images/KSC-logosma	200	1204
133.43.96.45	[01/Aug/1995:00:00:23	-0400]	GET /shuttle/missions/sts	200	8083
133.43.96.45	[01/Aug/1995:00:00:23	-0400]	GET /images/launch-logo.	200	1713
www-c8.proxy.aol.com	[01/Aug/1995:00:00:24	-0400]	GET /shuttle/countdown/	200	4324
133.43.96.45	[01/Aug/1995:00:00:25	-0400]	GET /history/apollo/image	200	1173
ix-esc-ca2-07.ix.netcom.com	[01/Aug/1995:00:00:25	-0400]	GET /shuttle/resources/or	200	4179
piweba4y.prodigy.com	[01/Aug/1995:00:00:32	-0400]	GET /images/NASA-logosr	200	786

Figure 11: Preprocessed Dataset in Excel



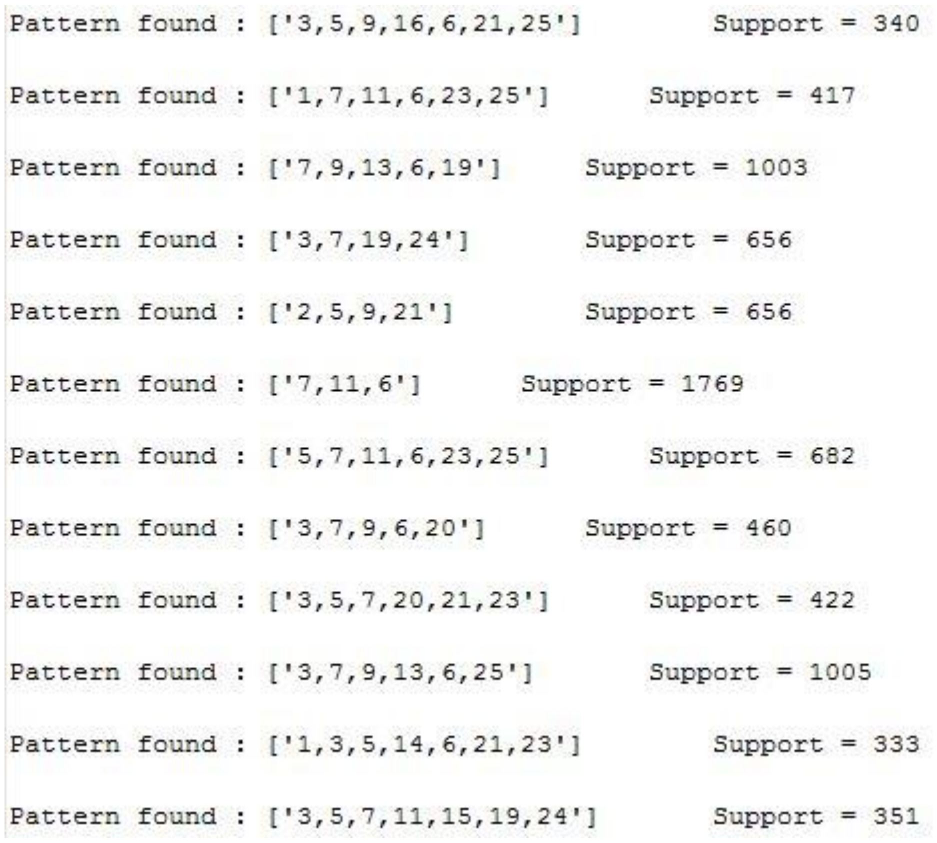
```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
124 147 148 149 150 151 152 153 154
1 4 6 8 9 14 20 21 22 31 40 42 44 48 49 51 54 60 65 72 76 78 81 83 84 87 88 90 104
8 48 60 88 106 122 124 179 198 215 275 332 364 377 402 461 472 476 524 525 526 527
6 8 21 44 49 60 66 81 95 119 122 124 146 149 150 185 198 208 228 234 242 306 338 359
1 8 10 19 21 22 40 42 47 48 49 51 70 84 98 102 112 121 122 123 124 127 130 137 146
8 44 49 60 66 95 119 122 124 146 149 150 185 198 208 234 306 338 359 377 378 391 393
1 6 8 9 13 14 18 20 21 22 24 30 31 40 42 48 49 51 52 54 60 65 72 76 81 84 87 88 97
1 8 19 20 22 23 30 49 51 54 55 60 81 88 116 121 122 146 150 179 185 202 207 253 260
8 23 27 48 49 51 53 60 76 90 121 122 130 149 150 171 185 190 216 228 238 255 275 287
1 4 6 8 14 18 20 21 22 31 38 40 42 43 49 51 53 54 55 60 65 66 76 81 84 86 88 92 98
1 6 8 9 10 13 14 20 21 22 31 38 40 42 48 49 51 54 55 58 60 76 78 81 84 86 88 90 97
1 8 9 24 33 39 49 53 55 72 78 84 101 116 121 122 124 146 147 150 154 158 168 171 198
1 4 6 8 9 10 14 20 21 22 24 27 35 38 40 41 42 44 48 49 51 54 55 60 72 76 78 81 84 85
1 4 6 8 13 14 19 21 22 24 30 31 40 42 48 49 51 54 55 60 63 66 81 84 86 87 88 103 112
652 1808 1809 1810 1811 1812
88 140 275 590 679 990 1813
1 5 8 9 20 22 23 34 49 51 60 81 88 107 122 124 146 149 150 158 171 220 294 308 312
34 49 85 122 146 179 260 264 330 439 455 465 511 521 591 697 767 863 873 1131 1206
8 23 122 377 380 386 430 486 524 591 679 737 878 951 993 1206 1614 1869 1870 1871
1 2 6 8 9 12 14 20 21 22 23 24 30 31 38 40 43 46 49 51 54 58 60 72 74 75 76 78 81 83
8 23 49 51 53 54 81 92 122 149 198 241 308 310 312 331 362 380 444 453 459 475 495
1 5 8 22 23 24 34 40 49 54 60 88 122 124 142 149 150 165 171 185 206 225 238 282 308
```

Figure 12: URL as Numbers with sessions identified



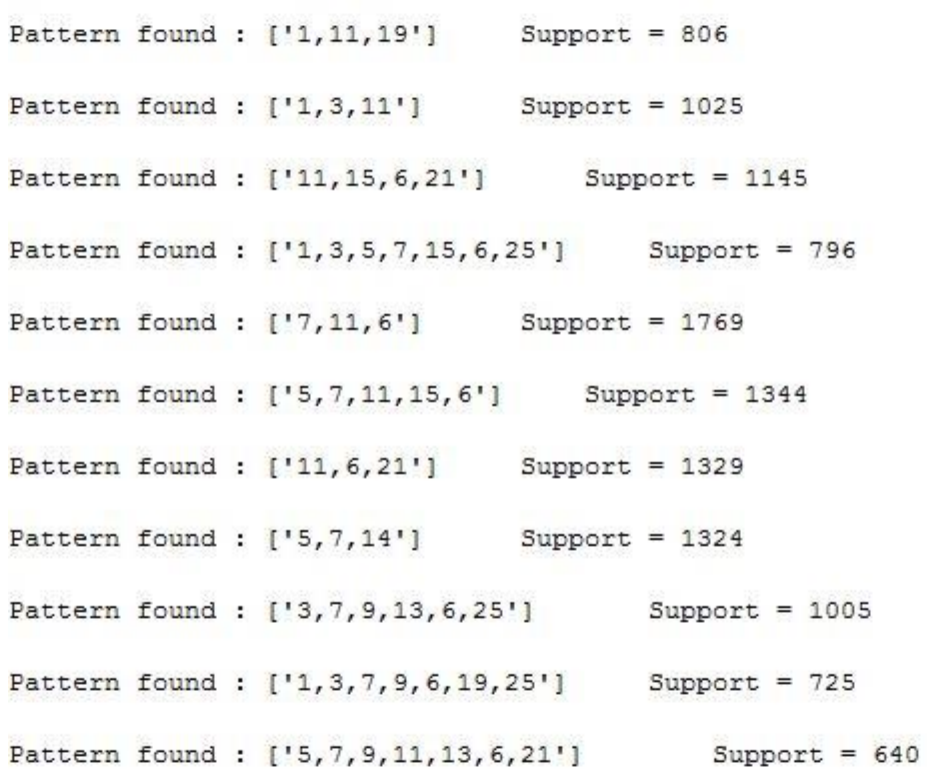
```
Pattern found : ['1,3,5,9,16,24,25']        Support = 177

Pattern found : ['7,9,11,13,15,6,19,23,25']     Support = 320

Pattern found : ['1,3,9,12,23,25']      Support = 257

Pattern found : ['1,3,7,9,14,19,25']        Support = 358

Pattern found : ['2,7,13,15,25']        Support = 423

Pattern found : ['3,5,9,11,14,15,6,19,23,25']       Support = 193

Pattern found : ['7,11,6,20,21']        Support = 350

Pattern found : ['6,20,21,24,25']       Support = 211

Pattern found : ['3,7,9,12,13,16,23,25']        Support = 172

Pattern found : ['1,3,7,15,19,21,24,25']        Support = 170

Pattern found : ['3,7,16,21,24']        Support = 282

Pattern found : ['1,3,5,14,19,21,23']       Support = 266

Pattern found : ['2,3,11,6,19,21,23']       Support = 166

Pattern found : ['3,9,14,21,24,25']     Support = 255

Pattern found : ['3,5,9,13,15,19,23,25']        Support = 341
```

Figure 13: Minimum Support of 5%



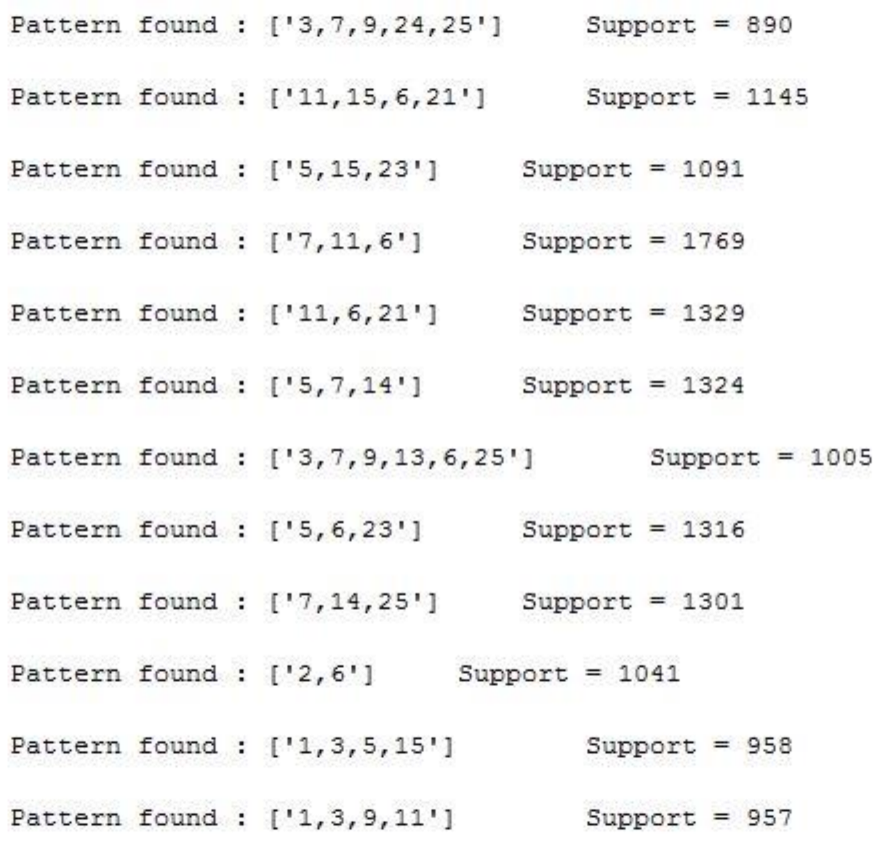
```
Pattern found : ['3,5,9,16,6,21,25']        Support = 340

Pattern found : ['1,7,11,6,23,25']      Support = 417

Pattern found : ['7,9,13,6,19']     Support = 1003

Pattern found : ['3,7,19,24']       Support = 656

Pattern found : ['2,5,9,21']        Support = 656

Pattern found : ['7,11,6']      Support = 1769

Pattern found : ['5,7,11,6,23,25']      Support = 682

Pattern found : ['3,7,9,6,20']      Support = 460

Pattern found : ['3,5,7,20,21,23']      Support = 422

Pattern found : ['3,7,9,13,6,25']       Support = 1005

Pattern found : ['1,3,5,14,6,21,23']        Support = 333

Pattern found : ['3,5,7,11,15,19,24']       Support = 351
```

Figure 14: Minimum Support of 10%



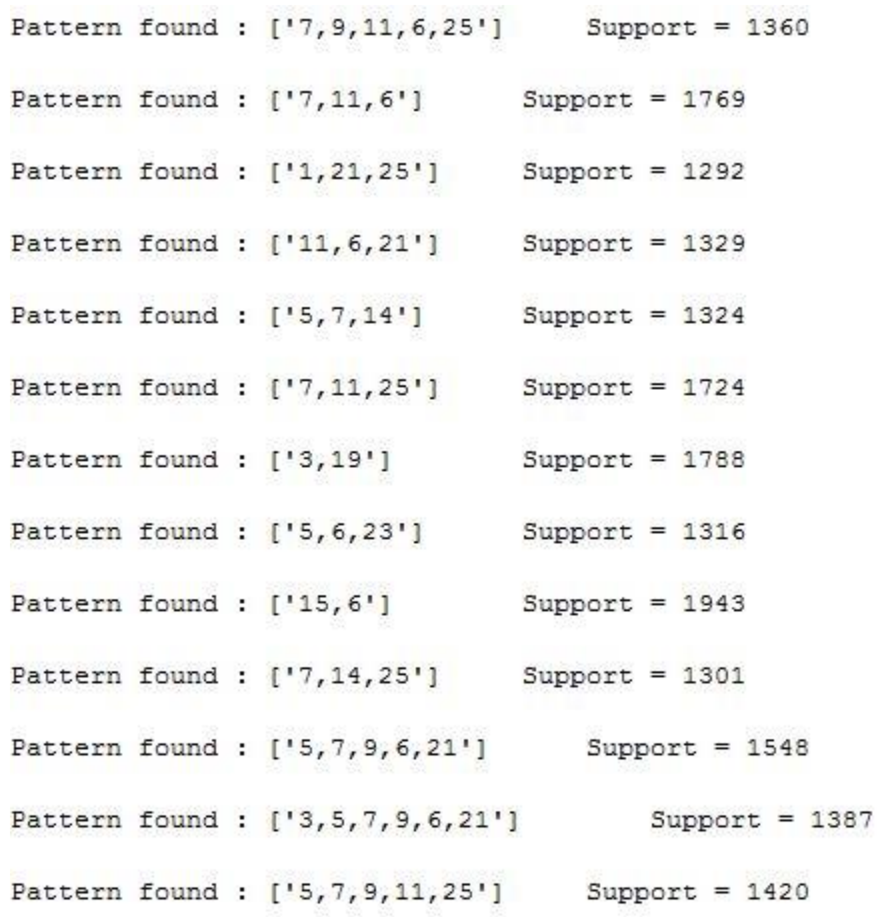
```
Pattern found : ['1,11,19']     Support = 806

Pattern found : ['1,3,11']      Support = 1025

Pattern found : ['11,15,6,21']      Support = 1145

Pattern found : ['1,3,5,7,15,6,25']     Support = 796

Pattern found : ['7,11,6']      Support = 1769

Pattern found : ['5,7,11,15,6']     Support = 1344

Pattern found : ['11,6,21']     Support = 1329

Pattern found : ['5,7,14']      Support = 1324

Pattern found : ['3,7,9,13,6,25']       Support = 1005

Pattern found : ['1,3,7,9,6,19,25']     Support = 725

Pattern found : ['5,7,9,11,13,6,21']        Support = 640
```

Figure 15: Minimum Support of 20%



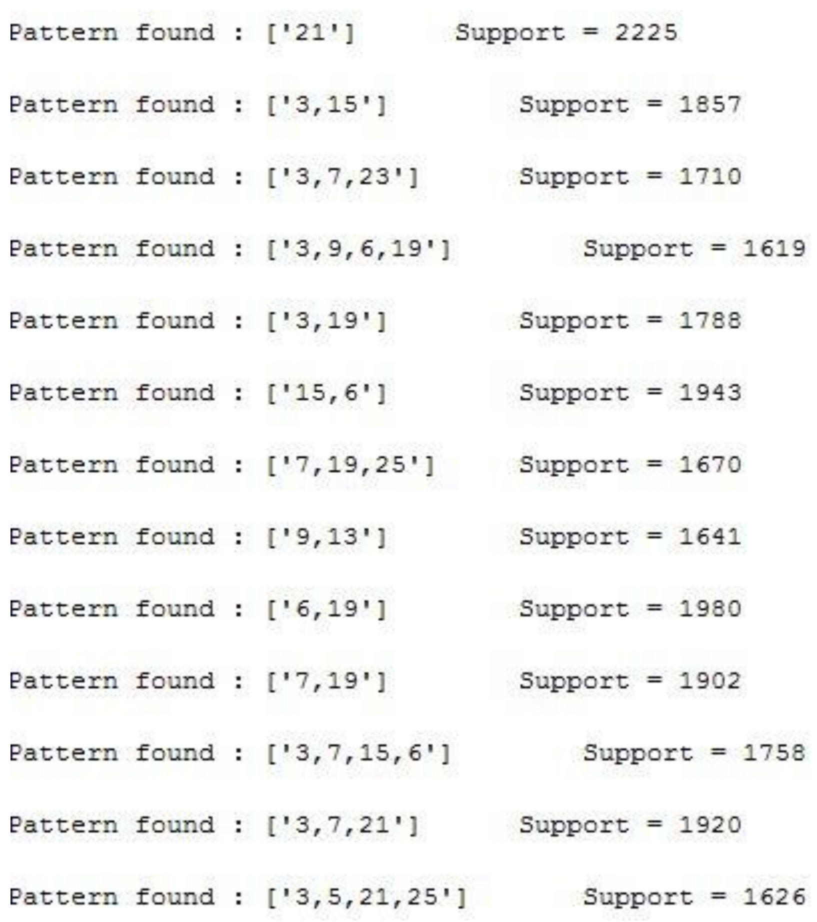
```
Pattern found : ['3,7,9,24,25']     Support = 890

Pattern found : ['11,15,6,21']      Support = 1145

Pattern found : ['5,15,23']     Support = 1091

Pattern found : ['7,11,6']      Support = 1769

Pattern found : ['11,6,21']     Support = 1329

Pattern found : ['5,7,14']      Support = 1324

Pattern found : ['3,7,9,13,6,25']       Support = 1005

Pattern found : ['5,6,23']      Support = 1316

Pattern found : ['7,14,25']     Support = 1301

Pattern found : ['2,6']     Support = 1041

Pattern found : ['1,3,5,15']        Support = 958

Pattern found : ['1,3,9,11']        Support = 957
```

Figure 16: Minimum Support of 25%



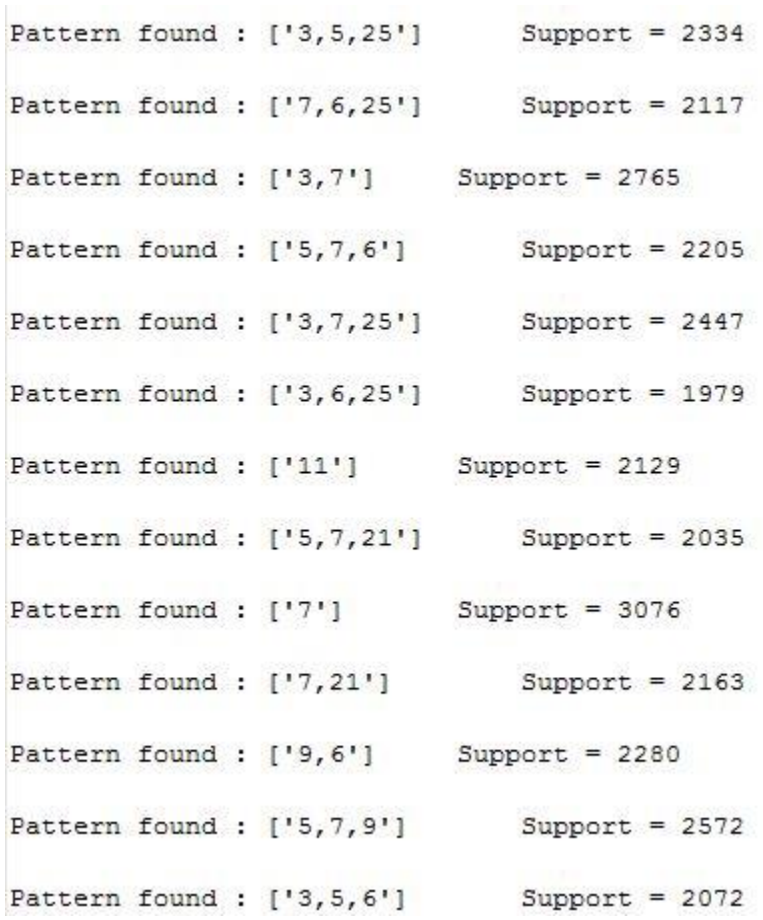
```
Pattern found : ['7,9,11,6,25']     Support = 1360

Pattern found : ['7,11,6']      Support = 1769

Pattern found : ['1,21,25']     Support = 1292

Pattern found : ['11,6,21']     Support = 1329

Pattern found : ['5,7,14']      Support = 1324

Pattern found : ['7,11,25']     Support = 1724

Pattern found : ['3,19']        Support = 1788

Pattern found : ['5,6,23']      Support = 1316

Pattern found : ['15,6']        Support = 1943

Pattern found : ['7,14,25']     Support = 1301

Pattern found : ['5,7,9,6,21']      Support = 1548

Pattern found : ['3,5,7,9,6,21']        Support = 1387

Pattern found : ['5,7,9,11,25']     Support = 1420
```

Figure 17: Minimum Support of 40%



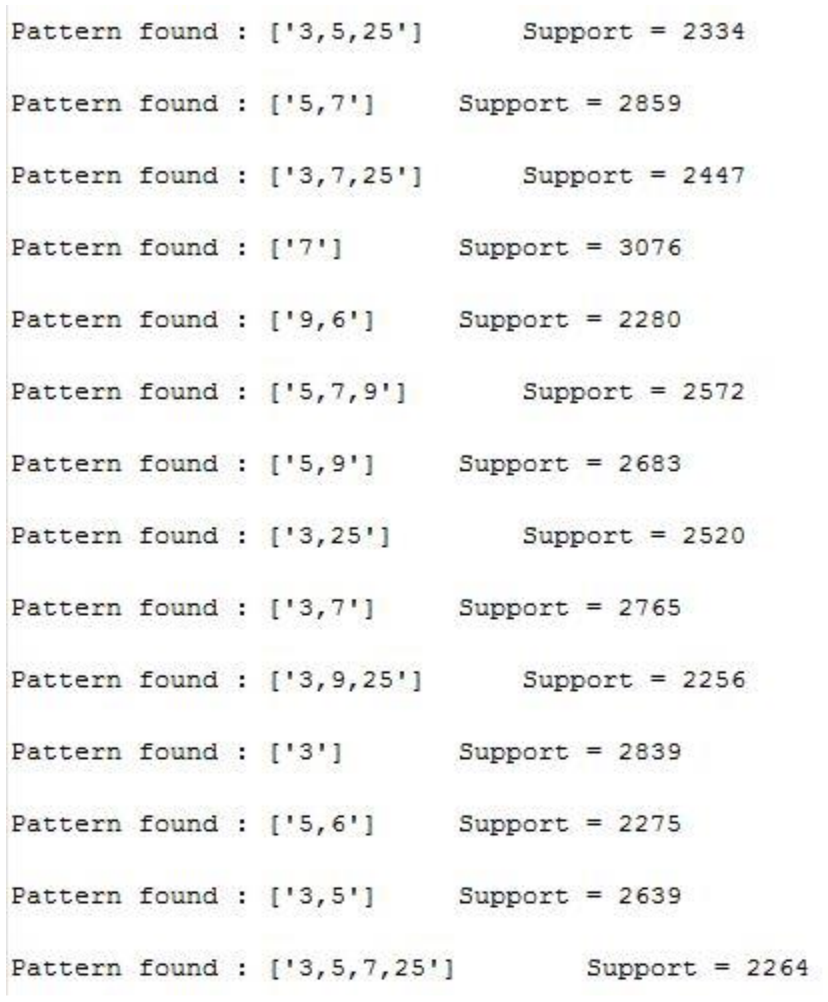
```
Pattern found : ['21']      Support = 2225

Pattern found : ['3,15']        Support = 1857

Pattern found : ['3,7,23']      Support = 1710

Pattern found : ['3,9,6,19']        Support = 1619

Pattern found : ['3,19']        Support = 1788

Pattern found : ['15,6']        Support = 1943

Pattern found : ['7,19,25']     Support = 1670

Pattern found : ['9,13']        Support = 1641

Pattern found : ['6,19']        Support = 1980

Pattern found : ['7,19']        Support = 1902

Pattern found : ['3,7,15,6']        Support = 1758

Pattern found : ['3,7,21']      Support = 1920

Pattern found : ['3,5,21,25']       Support = 1626
```

Figure 18: Minimum Support of 50%



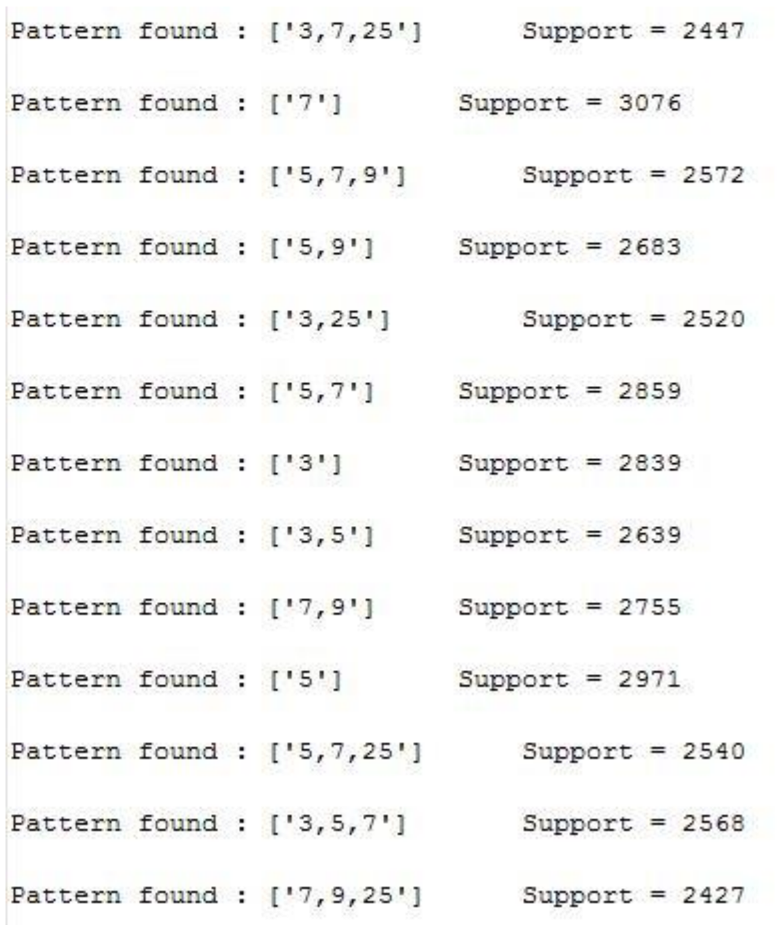
```
Pattern found : ['3,5,25']      Support = 2334

Pattern found : ['7,6,25']      Support = 2117

Pattern found : ['3,7']     Support = 2765

Pattern found : ['5,7,6']       Support = 2205

Pattern found : ['3,7,25']      Support = 2447

Pattern found : ['3,6,25']      Support = 1979

Pattern found : ['11']      Support = 2129

Pattern found : ['5,7,21']      Support = 2035

Pattern found : ['7']       Support = 3076

Pattern found : ['7,21']        Support = 2163

Pattern found : ['9,6']     Support = 2280

Pattern found : ['5,7,9']       Support = 2572

Pattern found : ['3,5,6']       Support = 2072
```

Figure 19: Minimum Support of 60%



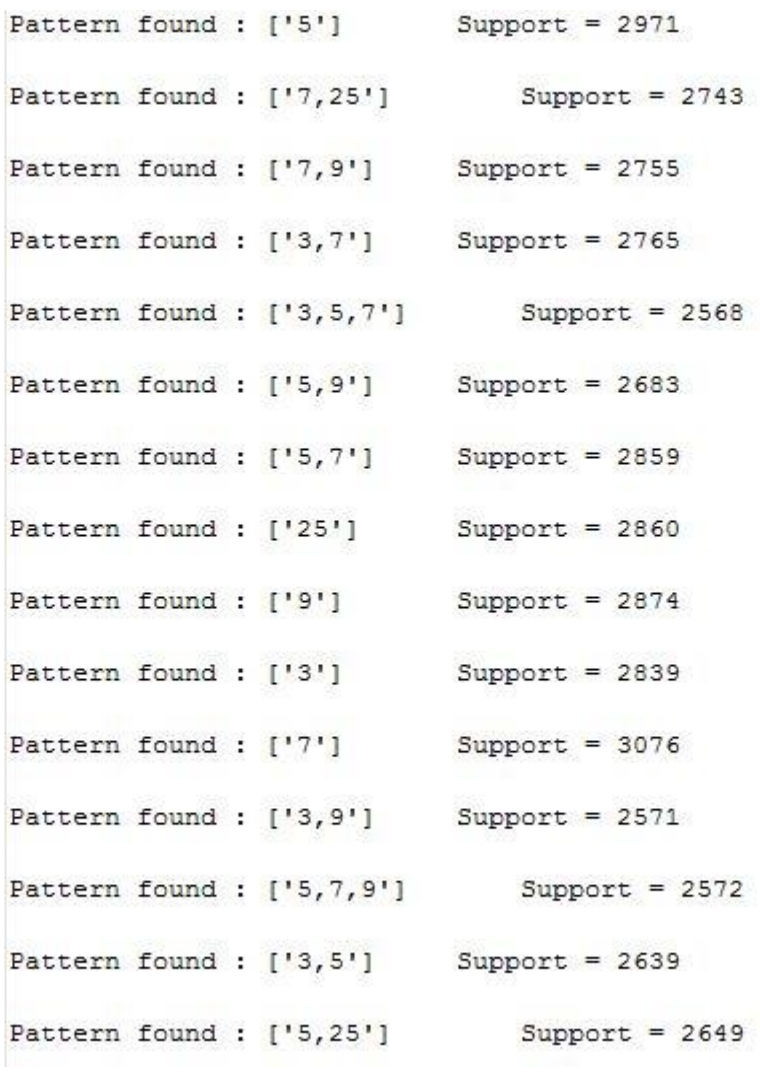
```
Pattern found : ['3,5,25']      Support = 2334

Pattern found : ['5,7']     Support = 2859

Pattern found : ['3,7,25']      Support = 2447

Pattern found : ['7']       Support = 3076

Pattern found : ['9,6']     Support = 2280

Pattern found : ['5,7,9']       Support = 2572

Pattern found : ['5,9']     Support = 2683

Pattern found : ['3,25']        Support = 2520

Pattern found : ['3,7']     Support = 2765

Pattern found : ['3,9,25']      Support = 2256

Pattern found : ['3']       Support = 2839

Pattern found : ['5,6']     Support = 2275

Pattern found : ['3,5']     Support = 2639

Pattern found : ['3,5,7,25']        Support = 2264
```

Figure 20: Minimum Support of 70%



```
Pattern found : ['3,7,25']      Support = 2447

Pattern found : ['7']       Support = 3076

Pattern found : ['5,7,9']       Support = 2572

Pattern found : ['5,9']     Support = 2683

Pattern found : ['3,25']        Support = 2520

Pattern found : ['5,7']     Support = 2859

Pattern found : ['3']       Support = 2839

Pattern found : ['3,5']     Support = 2639

Pattern found : ['7,9']     Support = 2755

Pattern found : ['5']       Support = 2971

Pattern found : ['5,7,25']      Support = 2540

Pattern found : ['3,5,7']       Support = 2568

Pattern found : ['7,9,25']      Support = 2427
```

Figure 21: Minimum Support of 75%



```
Pattern found : ['5']        Support = 2971

Pattern found : ['7,25']        Support = 2743

Pattern found : ['7,9']      Support = 2755

Pattern found : ['3,7']      Support = 2765

Pattern found : ['3,5,7']       Support = 2568

Pattern found : ['5,9']      Support = 2683

Pattern found : ['5,7']      Support = 2859

Pattern found : ['25']       Support = 2860

Pattern found : ['9']        Support = 2874

Pattern found : ['3']        Support = 2839

Pattern found : ['7']        Support = 3076

Pattern found : ['3,9']      Support = 2571

Pattern found : ['5,7,9']       Support = 2572

Pattern found : ['3,5']      Support = 2639

Pattern found : ['5,25']        Support = 2649
```

Figure 22: Minimum Support of 80%