**20 Thieves Solution by Syed**

**Initial Scenario**

I received email from Tim Brewer at 11:42 and after reading the email and problem readme file. I was up to solve the problem by 12:00 PM. Now 48 hours mean, I must submit it before 12:PM Friday. That’s great. I liked the problem and I liked the approach taken by freelancer.com.

**My Approach Log**

* Initially I just skimmed through the problem to get an idea of what the problem is all about. And after starting writing this solution file, I got back to readme file, but this time to read it with precision. It is important to understand the problem first rather than thinking about the solution. My intial analysis of the problem can be found under Initial Problem Analysis Heading below.
* After completing the initial problem analysis, I moved to the data analysis part. This doesn’t mean programmatically analyzing data, rather it is about reading and understanding the data. The details of initial data analysis can be found under Initial Data Analysis heading below.
* After completing the initial analysis of the data and writing email to Tim Brewer, I moved forward with exploratory data analysis of data to get insight of the data and to identify the need to data preprocessing. Details are available below under initial exploratory data analysis and pre-processing heading.
* After getting my important data ready for analysis. I started analyzing the data. The details of each analysis and my intensions are briefed under initial data analysis heading.
* My work continued in initial data analysis heading and I was able to fetch initial 20 suspects using association rule mining technique. I used FP Growth algorithm for this purpose. However, now onwards, I will also attempt to identify the suspects with my own custom build model. The details of the model and results are provided under heading My Crude Model.

**Initial Problem Analysis**

Few important observations.

* There is at-least one thief (Yes, it is obvious, but I preferred to mark it down)
* Scanner only scan the name and DOB for anyone who enters the club. (Not who left, be careful)
* Ranked list of up-to 20 suspects is needed. From most suspicious to least suspicious. There could be fewer. (Ranking problem)
* Post mid-night time will be considered as previous day
* We have visitor and theft logs
* Freelancer.com need details of how I approached and solved the problem and the problems I faced during the journey. (This is why I have created this document, I hope this will help)

**Initial Data Analysis**

Initial observations about the data:

* theft\_log.csv contains only the dates about the theft. (No time information is provided. This is lack of information, but we data scientist should work it out)
* visitor\_log.csv contains only the visit\_date, name, and dob. (Again we only have the day information. We only know that individual entered the place on the day, but we don’t know either he/she left on the same day or not)
* The joining attribute between visit and theft is date.
* Before moving forward, inquire from Tim Brewer that should be we asking question related to problem? Just to make sure we don’t commit any sins of 09 sins of data mining.

**Initial Exploratory Data Analysis and Pre-processing**

* Before, I start performing the exploratory data analysis, I preferred to get the data into a database. Well, it is not necessary. Many data scientist can process the same data using MS Excel or directly using any data mining tools. But I prefer to work using SQL. And it really help us while working with large datasets. Other tools often start crying when data size grows. I used SQL Server 2016, again only for my personal preference. I created a database nct, short for nightclubthieves. Created two tables each for theft\_log and visitor\_log. Then I bulk inserted data into tables. The queries for this step are provided as initialbulkinsert.sql with this solution.
* *Before moving forward, I must backup my work on GIT. I have version controlled all my work at:* [*https://bitbucket.org/saif137/mybi*](https://bitbucket.org/saif137/mybi) *in freelancertask folder.*
* Now I plan to perform all possible exploratory data analysis, transformations and preprocessing in database using SQL. Once done, with our exploratory data analysis then we will move forward with the problem.
* I started with theft\_log.csv data. It is available in theft\_log\_i table. I first preprocessed the date field to separate out the year, month and day information and stored it into separated table.
* I firsts tried to identify, if there is any patter for theft in terms of month? Either frequency of theft is more in few months or there is gradual increase or decrease in theft over time? For this purpose I wrote following query and found out the yes, July, November, and December has higher theft frequency.

--First check if theft occurence is same or varies across month

select

tyear [Year of theft]

, tmonth [Month of theft]

, count (\*) [Total theft]

from theft\_log

group by

tyear, tmonth

order by

tyear, tmonth

* I also plot the same data using MS Excel PivotChart is show in Figure 1. Visualization is very important for exploratory data analysis. It is also very important task for data scientists in addition to data preprocessing and modeling.

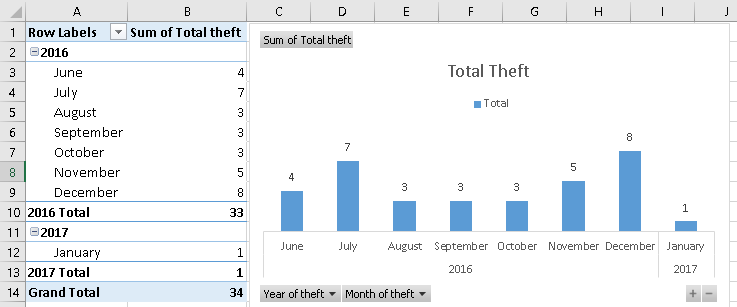


Figure 1: Month-wise Theft

* Another important exploration will be either, there are any specific days when more thefts have occurred? I identified day-wise theft occurrences using the query given below:

select

tdayname [Day]

, count(\*)

from

theft\_log

group by

tdayname

* Above query result only provided a hint that there has been less theft on Tuesday and Wednesday, but there is no day when there was no theft. Nor we can convincingly say that there are some days with more theft occurrences. It further tried to separate is out on the basis of month using query below:

--Theft occurences on basis of month

select

tmonthname [Month]

, tdayname [Day]

, count(\*)

from

theft\_log

group by

tmonthname, tdayname

order by

tmonthname, tdayname

* Above query result only provided a hint theft occurrence has been more on Sunday and Monday of December. But still I didn’t found any significant patter. Visualization of theft patter for days on montly basis is shown in Figure 2.

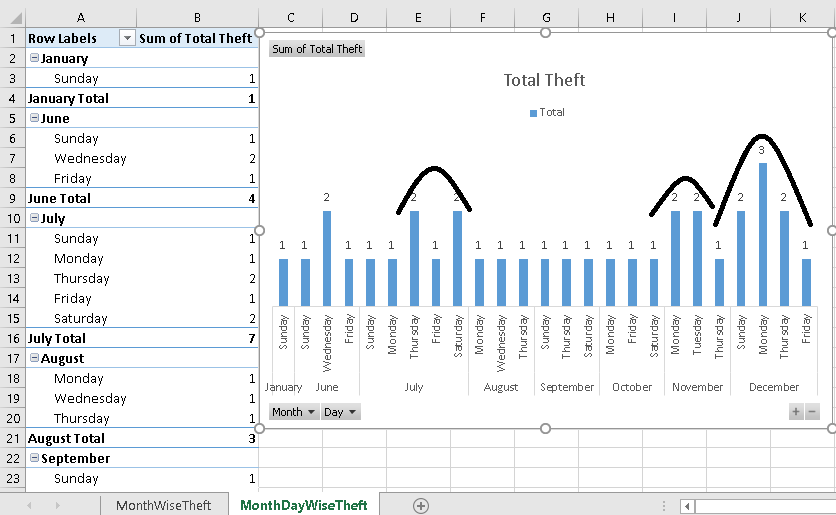


Figure 2: Theft patter for Days on Monthly Basis

* I immediately realized, that week-wise analysis should also be done to identify if there are some weekly patterns for theft. I added week number column to our table after recreating it with updated query. The updated queries are provided with solution as theft\_log\_pp.sql.

select

tweek [Week]

, count(\*) [Total Theft]

from

theft\_log

group by

tweek

* The weekly theft count also didn’t gave any specific pattern except a minor observation that last three weeks of the year has more theft occurrences together and similarly week 27 to 32 has continuous pattern of theft.
* *Before moving forward, I must backup my work on GIT. I have version controlled all my work at:* <https://bitbucket.org/saif137/mybi> *in freelancertask folder.*
* I will stop working on theft\_log data and now I will move to visitor\_log data that we have already loaded in visitor\_log\_i. I just tried to identify, how each individual can be identified. After checking with distinct and group by queries, it was clear that we should identify each individual using both name and dob.
* Before, I try to find out individuals, who may be guilty. I preferred to eliminate individuals are innocent. I filtered the records/visits for days when no theft was reported. This will reduce our search space. I created a separate table named visitor\_log\_t to store visit information for days when theft has occurred. While moving the data, we also separated the date parts of the visit using query shown below:

insert into visitor\_log

select

visit\_date

, [name]

, [dob]

, convert(integer, SUBSTRING(visit\_date, 1, 4)) as tyear --can also use datepart

, convert(integer, SUBSTRING(visit\_date, 6, 2)) as tmonth

, convert(integer, SUBSTRING(visit\_date, 9, 2)) as tday

, datename(dw, visit\_date) tdayname

, datename(mm, visit\_date) tmonthname

, datepart(wk, visit\_date) tweek

from visitor\_log\_i where visit\_date in (

select theft\_day from theft\_log\_i

)

* Above query reduced our search space from 42727 to 7288. Now we only need to consider individuals in this 7288 records to identify the probable thieves.
* *Before moving forward, I must backup my work on GIT. I have version controlled all my work at:* <https://bitbucket.org/saif137/mybi> *in freelancertask folder.*

**Initial Exploratory Data Analysis and Pre-processing**

* I started my analysis to answer few questions that I had. How many individuals visited on the day, when theft has occurred. I found out that between 195 to 245 visitors were present each day when theft was occurred.
* Next, I was curious to know, either there are visitors who were present on every theft day? Or I can ask the same question as, list the visitors according to their presence of number of theft days. There are 34 distinct theft days. The maximum theft days a visitor was found to be present is Karen Keeny and its 30 days, but beware, Karen could be the most loyal customer of the night club. We are still far away from making any prediction. The only thing that we can take away is that there are more than one thief.
* Next, I was curious to know, either there are visitors who were present together on every theft day? This can also be answered using Association Rule mining. Here I got stuck for two hours, because I tried to get it done using T-SQL and then limit issue arise when I attempted to generate a table with separate column for each visitor. I worked so hard but in the end stuck while creating a query for table with each visitor as a column. But I could do the same thing using MS Excel Pivot table in a minute. The snapshot is given below in Figure 3, where each id on column represent a unique visitor. However, this effort allowed me to replace name and dob with a unique identifier that I will be using for any further processing. This will anyhow reduce the data processed during analytics.

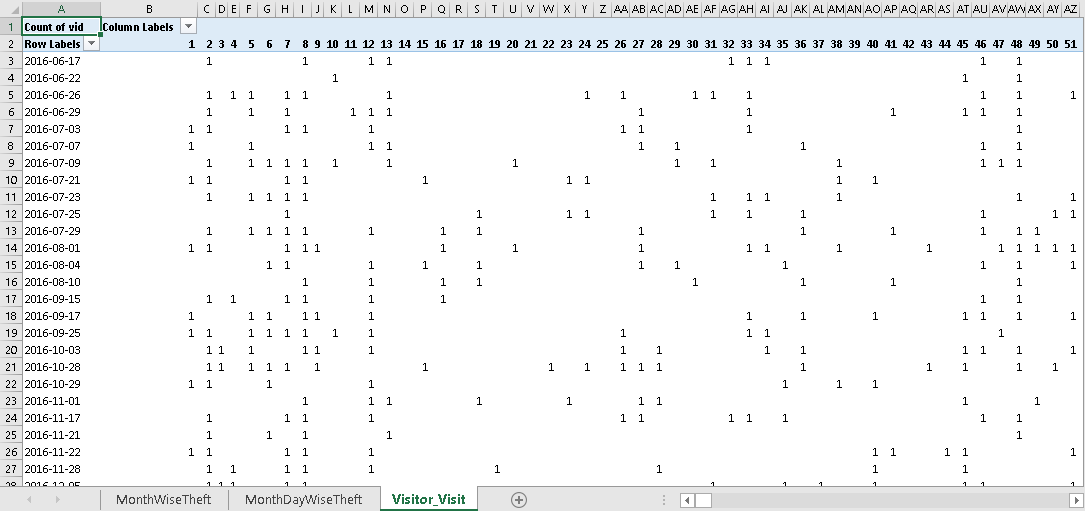


Figure 3: Visitors Visiting Together on Each Theft Day

* Using this tabular view of the data as shown in Figure 3, I was able to perform association rule mining using FP-Growth algorithm to identify, which visitors visited together during theft days. I was able to complete the initial model execution with default values using rapid miner, but then it became time to take break as library is closing. I will start again on this after2 hours from home.
* After updating the parameters, i.e., minimum confidence to 0.99 (to fetch only important and strong association, following visitors are of interest for us. But this doesn’t mean yet, that they are thieves, they could be our loyal customers as well.

Association Rules (Ignored results)

[726] --> [464] (confidence: 0.909)

[538] --> [464] (confidence: 0.909)

[289] --> [464] (confidence: 0.909)

[207] --> [464] (confidence: 0.909)

[773] --> [563] (confidence: 0.909) (Lift = 1.145)

[650] --> [302] (confidence: 0.909) (Lift = 1.145)

[207] --> [302] (confidence: 0.909) (Lift = 1.145)

[100] --> [632] (confidence: 0.909) (Lift = 1.236)

[679, 455] --> [464] (confidence: 0.909)

[563, 744] --> [464] (confidence: 0.909)

[464, 152] --> [563] (confidence: 0.909) (Lift = 1.145)

[563, 145] --> [464] (confidence: 0.909) (Lift = 1.145)

[464, 54] --> [455] (confidence: 0.909)

[744, 145] --> [464] (confidence: 0.909)

[563, 744] --> [152] (confidence: 0.909) (Lift = 1.189)

[744, 152] --> [563] (confidence: 0.909) (Lift = 1.145)

[784] --> [464] (confidence: 0.913)

[748] --> [464] (confidence: 0.913)

[142] --> [464] (confidence: 0.913)

[563, 455] --> [464] (confidence: 0.913)

[464, 302] --> [455] (confidence: 0.913) (Lift = 1.15)

[455, 302] --> [464] (confidence: 0.913)

[709] --> [464] (confidence: 0.917)

[145] --> [464] (confidence: 0.923)

[455] --> [464] (confidence: 0.926)

[701] --> [302] (confidence: 0.952) (Lift = 1.199)

[455, 54] --> [464] (confidence: 0.952)

[848] --> [464] (confidence: 0.955)

[455, 145] --> [464] (confidence: 0.955)

* I also attempted to get another perspective of the generated rules using the graph view. It gave me an interesting insight. As it can be observed in Figure 4, visitors 100 and 632 are present on theft days without any association with other visitors. It may be interesting, but it will need further investigation.

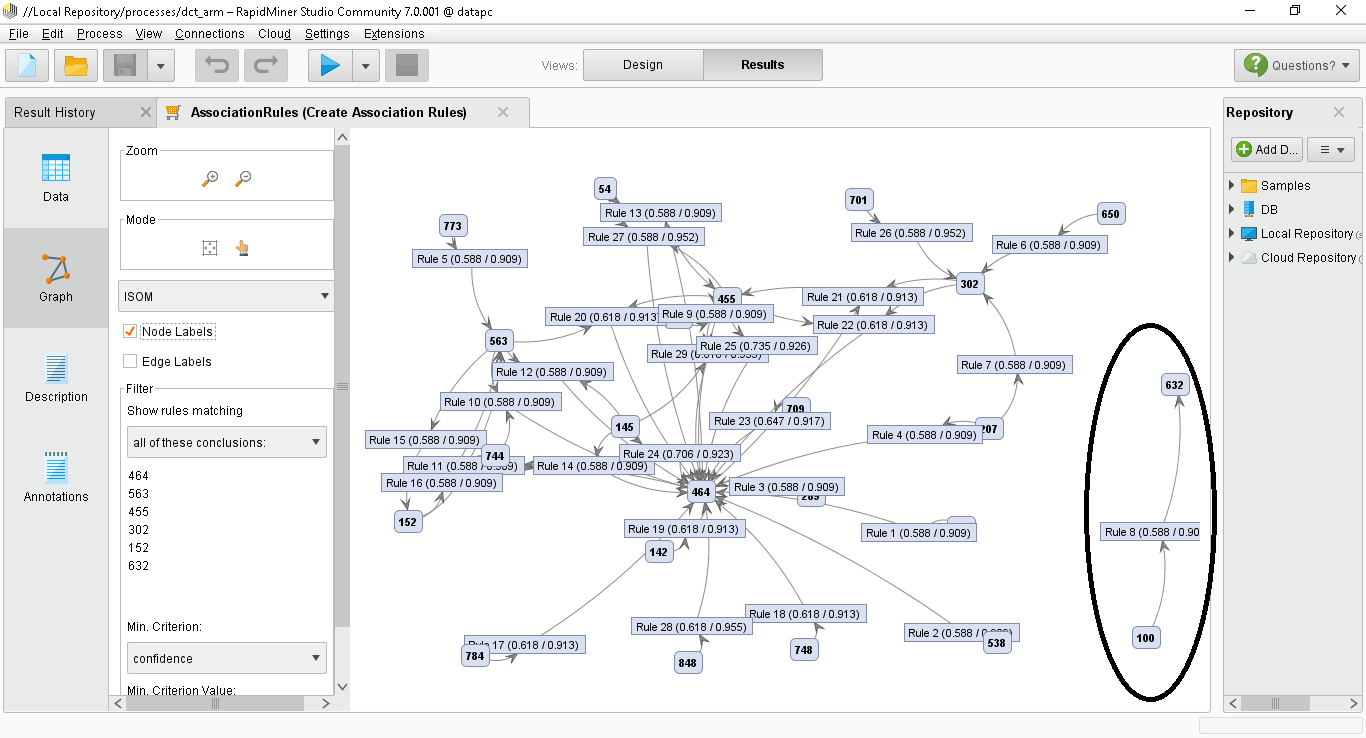


Figure 4: Graph View of Generated Association Rules (Ignored results)

* I will keep focus on following visitors during further analysis: 100, 632, 464, 145, 563, 455, 302, 709, 744, 207, 209, 152, 302 as these visitors has been present in existence of most other visitors.
* Now, its time to try another approach. I just wondered, why didn’t I took this problem as a classification problem? How about adding a new column with two possible classes, i.e., theft occurred or not. And then train the model to make prediction. This could also be a good approach to identify potential thieves. The problem is, all experiment of association rule mining will need to be re-run as I have to regenerate the visitor ids. The good part is, we can always identify the new ids for old ones using joins. So the problem can be solved. I just did as I mentioned above, I took all data and converted it into columnar format as we have done in our previous theft day only data set. Then each visit of theft day is marked as 1 and each visit on non-theft day is marked as 0. The script for preparing this data is available in sources with name reformulatingasPrediction.sql.
* After preparation of the data, we attempted to identify highly correlated attributes, which for us means visitors. With 0.6 cuttoff for correlation we got only four visitors of 330, 731, 745, 627 using R script given below:

#Loading initial data

train <- read.csv("C:/data/certi/mybiwork/freelancertask/visitor\_c.csv")

#Check that data is as it should be

#View(train)

set.seed(7)

library(mlbench)

library(caret)

train.visitors <- train[2:length(train) - 1]

myvisitors <- sapply(train.visitors, as.numeric)

corrmat <- cor(myvisitors)

hicorr <- findCorrelation(corrmat, cutoff=0.6)

print (hicorr)

* But beware, these are visitors that are correlated for both theft and non-theft visit. To ease up my problem, I will be re-generating my association rule mining results with new visitors id, so that I can myself keep track of visitors ids of interest. The good part is, this resulted in identifying better results as can be observed in Figure 5. Now I found out the visitors with above 70 % support and 100% confidence that they were present during the theft day. I have also listed the identified association rules with highest support visitor on the first row.

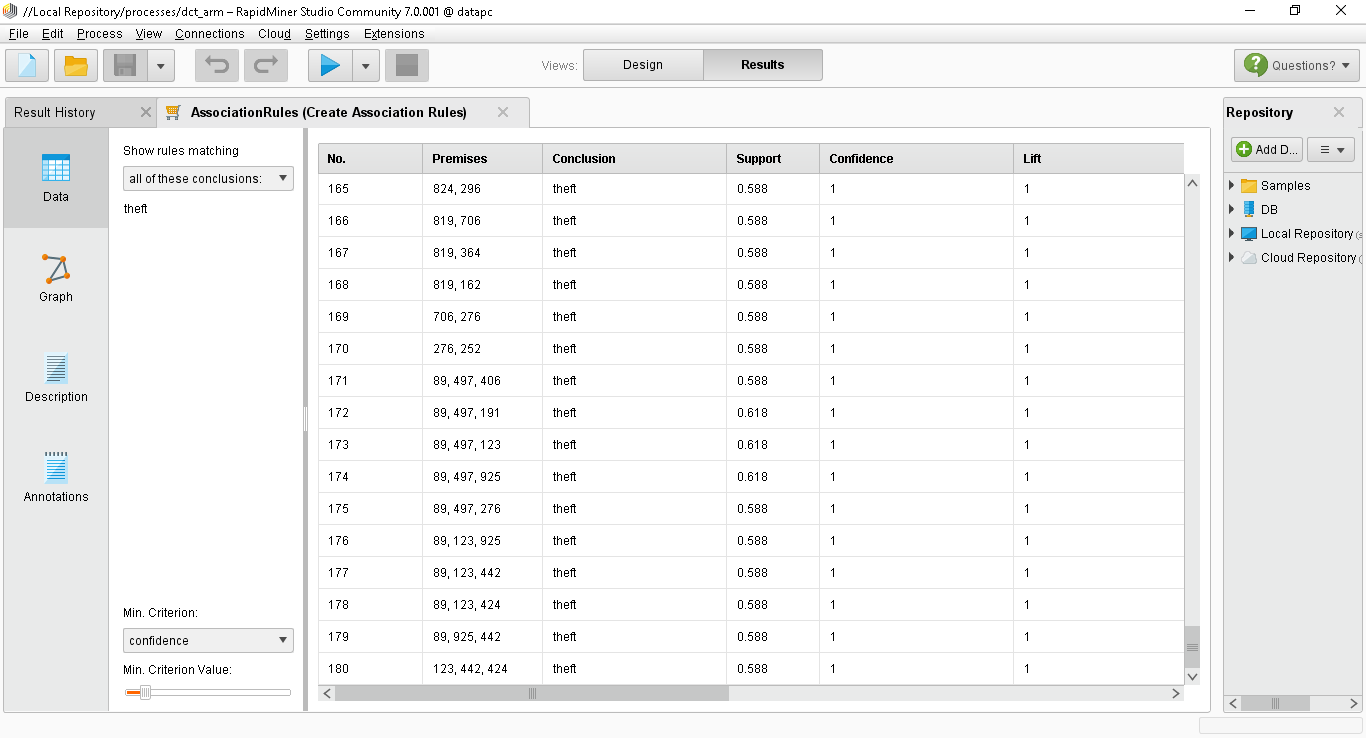


Figure 5: Group of Visitors During Theft Days

**AssociationRules (Theft during presence)**

Association Rules

[89] --> [theft] (confidence: 1.000)

[523] --> [theft] (confidence: 1.000) (Also present on non-theft days)

[497] --> [theft] (confidence: 1.000)

[406] --> [theft] (confidence: 1.000) (Also present on non-theft days)

[191] --> [theft] (confidence: 1.000) (Also present on non-theft days)

[123] --> [theft] (confidence: 1.000)

[925] --> [theft] (confidence: 1.000) (Also present on non-theft days)

[442] --> [theft] (confidence: 1.000) (Also present on non-theft days)

[424] --> [theft] (confidence: 1.000)

[234] --> [theft] (confidence: 1.000) (Also present on non-theft days)

[824] --> [theft] (confidence: 1.000)

[819] --> [theft] (confidence: 1.000)

[706] --> [theft] (confidence: 1.000) (Also present on non-theft days)

[364] --> [theft] (confidence: 1.000)

[276] --> [theft] (confidence: 1.000) (Also present on non-theft days)

[886] --> [theft] (confidence: 1.000)

[809] --> [theft] (confidence: 1.000) (Also present on non-theft days)

[612] --> [theft] (confidence: 1.000) (Also present on non-theft days)

[252] --> [theft] (confidence: 1.000)

[239] --> [theft] (confidence: 1.000)

[212] --> [theft] (confidence: 1.000)

[176] --> [theft] (confidence: 1.000)

[89, 497] --> [theft] (confidence: 1.000)

[89, 123] --> [theft] (confidence: 1.000)

[89, 925] --> [theft] (confidence: 1.000)

Following added after reducing support to 65%, to reach 20 suspects

[923] --> [theft] (confidence: 1.000)

[743] --> [theft] (confidence: 1.000)

[611] --> [theft] (confidence: 1.000) (Also present on non-theft days)

[438] --> [theft] (confidence: 1.000)

[17] --> [theft] (confidence: 1.000)

[162] --> [theft] (confidence: 1.000)

[89, 523] --> [theft] (confidence: 1.000)

[89, 406] --> [theft] (confidence: 1.000)

[89, 191] --> [theft] (confidence: 1.000)

[89, 442] --> [theft] (confidence: 1.000)

[89, 234] --> [theft] (confidence: 1.000)

[497, 191] --> [theft] (confidence: 1.000)

[497, 123] --> [theft] (confidence: 1.000)

[123, 424] --> [theft] (confidence: 1.000)

Following added after reducing support to 60%, to reach 20 suspects

[892] --> [theft] (confidence: 1.000) (Also present on non-theft days)

[868] --> [theft] (confidence: 1.000)

[82] --> [theft] (confidence: 1.000) (Also present on non-theft days)

[677] --> [theft] (confidence: 1.000)

[587] --> [theft] (confidence: 1.000) (Also present on non-theft days)

[296] --> [theft] (confidence: 1.000)

[262] --> [theft] (confidence: 1.000)

[157] --> [theft] (confidence: 1.000)

[133] --> [theft] (confidence: 1.000)

* I will also like to identify those individuals who were always absent on the day when no theft occurred as this will increase their suspicion if they were always present when theft was occured (This is my weak assumption: I Know). For this purpose, I also changed the positive value to false for FP Growth, which means when certain visitors were absent no theft occurred. I select 100% support and confidence for rules generation. We only want to omit the most loyal customers. The generated association rules are listed below. The good part is, none of them are in our suspicious visitor list.

# AssociationRules (No Theft During Absence)

Association Rules (None were found to be present on theft day with 100% conf)

[theft] --> [690] (confidence: 1.000)

[690] --> [theft] (confidence: 1.000)

[theft] --> [660] (confidence: 1.000)

[660] --> [theft] (confidence: 1.000)

[theft] --> [544] (confidence: 1.000)

[544] --> [theft] (confidence: 1.000)

[theft] --> [521] (confidence: 1.000)

[521] --> [theft] (confidence: 1.000)

[690] --> [660] (confidence: 1.000)

[660] --> [690] (confidence: 1.000)

[690] --> [544] (confidence: 1.000)

[544] --> [690] (confidence: 1.000)

[690] --> [521] (confidence: 1.000)

[521] --> [690] (confidence: 1.000)

[660] --> [544] (confidence: 1.000)

[544] --> [660] (confidence: 1.000)

[660] --> [521] (confidence: 1.000)

[521] --> [660] (confidence: 1.000)

[544] --> [521] (confidence: 1.000)

[521] --> [544] (confidence: 1.000)

[theft] --> [690, 660] (confidence: 1.000)

[690] --> [theft, 660] (confidence: 1.000)

[theft, 690] --> [660] (confidence: 1.000)

[660] --> [theft, 690] (confidence: 1.000)

[theft, 660] --> [690] (confidence: 1.000)

[690, 660] --> [theft] (confidence: 1.000)

[theft] --> [690, 544] (confidence: 1.000)

[690] --> [theft, 544] (confidence: 1.000)

[theft, 690] --> [544] (confidence: 1.000)

[544] --> [theft, 690] (confidence: 1.000)

[theft, 544] --> [690] (confidence: 1.000)

[690, 544] --> [theft] (confidence: 1.000)

[theft] --> [690, 521] (confidence: 1.000)

[690] --> [theft, 521] (confidence: 1.000)

[theft, 690] --> [521] (confidence: 1.000)

[521] --> [theft, 690] (confidence: 1.000)

[theft, 521] --> [690] (confidence: 1.000)

[690, 521] --> [theft] (confidence: 1.000)

[theft] --> [660, 544] (confidence: 1.000)

[660] --> [theft, 544] (confidence: 1.000)

[theft, 660] --> [544] (confidence: 1.000)

[544] --> [theft, 660] (confidence: 1.000)

[theft, 544] --> [660] (confidence: 1.000)

[660, 544] --> [theft] (confidence: 1.000)

[theft] --> [660, 521] (confidence: 1.000)

[660] --> [theft, 521] (confidence: 1.000)

[theft, 660] --> [521] (confidence: 1.000)

[521] --> [theft, 660] (confidence: 1.000)

[theft, 521] --> [660] (confidence: 1.000)

[660, 521] --> [theft] (confidence: 1.000)

[theft] --> [544, 521] (confidence: 1.000)

[544] --> [theft, 521] (confidence: 1.000)

[theft, 544] --> [521] (confidence: 1.000)

[521] --> [theft, 544] (confidence: 1.000)

[theft, 521] --> [544] (confidence: 1.000)

[544, 521] --> [theft] (confidence: 1.000)

[690] --> [660, 544] (confidence: 1.000)

[660] --> [690, 544] (confidence: 1.000)

[690, 660] --> [544] (confidence: 1.000)

[544] --> [690, 660] (confidence: 1.000)

[690, 544] --> [660] (confidence: 1.000)

[660, 544] --> [690] (confidence: 1.000)

[690] --> [660, 521] (confidence: 1.000)

[660] --> [690, 521] (confidence: 1.000)

[690, 660] --> [521] (confidence: 1.000)

[521] --> [690, 660] (confidence: 1.000)

[690, 521] --> [660] (confidence: 1.000)

[660, 521] --> [690] (confidence: 1.000)

[690] --> [544, 521] (confidence: 1.000)

[544] --> [690, 521] (confidence: 1.000)

[690, 544] --> [521] (confidence: 1.000)

[521] --> [690, 544] (confidence: 1.000)

[690, 521] --> [544] (confidence: 1.000)

[544, 521] --> [690] (confidence: 1.000)

[660] --> [544, 521] (confidence: 1.000)

[544] --> [660, 521] (confidence: 1.000)

[660, 544] --> [521] (confidence: 1.000)

[521] --> [660, 544] (confidence: 1.000)

[660, 521] --> [544] (confidence: 1.000)

[544, 521] --> [660] (confidence: 1.000)

* I just realized that I must complete the confusion box, so far we have identified visitors who were present and theft has occurred and who were not present and theft haven’t occurred. Now we should also identify those who were present and theft didn’t occurred and those who were absent and theft occurred.

# AssociationRules (No theft During Presence)

Association Rules (All found on theft days except those mentioned)

[191] --> [notheft] (confidence: 1.000)

[523] --> [notheft] (confidence: 1.000)

[925] --> [notheft] (confidence: 1.000)

[234] --> [notheft] (confidence: 1.000)

[406] --> [notheft] (confidence: 1.000)

[706] --> [notheft] (confidence: 1.000)

[82] --> [notheft] (confidence: 1.000)

[442] --> [notheft] (confidence: 1.000)

[276] --> [notheft] (confidence: 1.000)

[133] --> [notheft] (confidence: 1.000)

[612] --> [notheft] (confidence: 1.000)

[611] --> [notheft] (confidence: 1.000) (Was never present on theft days)

[809] --> [notheft] (confidence: 1.000)

[587] --> [notheft] (confidence: 1.000)

[892] --> [notheft] (confidence: 1.000)

[604] --> [notheft] (confidence: 1.000) (Was never present on theft days)

[191, 523] --> [notheft] (confidence: 1.000)

[191, 925] --> [notheft] (confidence: 1.000)

* I found many visitors who were absent during theft day with 100% support and confidence. The list of these visitors has be save as a separate file with name theft\_absence\_visitors.txt. Now we have 12 suspect after analyzing the all four possibilities of visitors present/absent during theft/non-theft days. To increase the list to 20, I will again run the first experiment for visitors present during theft day to meet the 20 requirements. Only matching the additional 8 to the other lists.
* *Here 24 hours of my task time is completed. Only 24 hours left.*
* Now we have our 20 initial suspects list. Time to explain our results before moving forward to get more insight. Belows is the list of suspects that we identified.
* 89 Karen Keeney 1993-12-25

89 Karen Keeney 1993-12-25

497 Judith Sanders 1993-08-26

123 Lynn Bernhart 1995-11-13

424 Daniel Laster 1995-12-31

824 Michael Mcbride 1993-04-26

819 Luz Connelly 1995-03-26

364 Kathleen Benzi 1995-09-15

886 Raymond Shannon 1997-08-06

252 Johnie Johnson 1995-12-28

239 John Davis 1996-02-05

212 Cheryl Robinson 1996-04-05

176 Florine Kim 1996-07-14

923 Sharon Barton 1997-09-08

743 Wes Carlson 1996-01-13

438 Charles Crandall 1997-05-12

17 Cynthia Allen 1996-08-29

162 Cynthia Dominquez 1996-05-16

868 Thomas Vanderwal 1997-05-06

677 Linda Gomez 1996-03-08

296 Charles Betts 1997-07-18

176 Florine Kim 1996-07-14

* What we claim about this list is as follows. With 70-60% support and 100% confidence from data we found that these visitors were present every time theft occurred. We also didn’t found these visitors in list of visitors who were absent during non-theft days with 100% support and from data. They were also not found in the list of visitors who were present on non-theft days with 100% support and confidence from data. They were also not found in the list of visitors who were absent on theft day with 100% support and confidence from data. The ranking of the customer is from first towards last considering their decrease support from data during rules generation.

**My Crude Model**

* I want to workout on the same problem using my custom-made point-scoring based greedy algorithm. It will be a very simple algorithm. For every visitor, it will score it positively if visitor was present on the day theft occurred and negatively if visitor was absent on the day or if visitor was present on the day when no theft occurred. We obtained the list of 20 visitors using this technique, from this list, 2 visitors were already on our existing list. The source code of the initial approach is provided as source with name dct\_freq\_tdv.py.
* I wanted to improve my crude approach. I just imagined, how humans will behave in similar scenario. I found that continuity of visitor presence during theft or absence of visitor presence during theft will effect how we rank each visitor. My implementation of this approach is available with name dct\_freq\_tdv\_con.py. However, the result are not what I wanted. Rather the results are matching more with visitors identified in Absent on Non Theft Day.
* I just got alarmed with my approach as it was not validating my initial list at all. Then I realized, that non-theft days don’t really add value to my calculation. So only focused on theft day for identifying vulnerable visitors. And then, yes, I felt back home. At-least this result was validating my results and my approach giving me confidence that I am moving in the right direction. The update source code is available with filename dct\_freq\_tdv\_con\_ana.py. The 20 most suspicious visitors using my crude model are as given below:

**""" Most vulnerable visitors under perseptions  
(191, 802) #Also present on non-theft days, skipped  
(89, 793) # This is also part of our initial list  
(497, 766) # This is also part of our initial list  
(276, 712) #Also present on non-theft days, skipped  
(523, 702) #Also present on non-theft days, skipped  
(706, 700) #Also present on non-theft days, skipped  
(442, 694) #Also present on non-theft days, skipped  
(123, 679) # This is also part of our initial list  
(234, 677) #Also present on non-theft days, skipped  
(886, 673) # This is also part of our initial list  
(925, 668) #Also present on non-theft days, skipped  
(406, 667) #Also present on non-theft days, skipped  
(824, 653) # This is also part of our initial list  
(967, 643)  
(819, 642) # This is also part of our initial list  
(364, 622) # This is also part of our initial list  
(424, 613) # This is also part of our initial list  
(176, 612) # This is also part of our initial list  
(743, 606) # This is also part of our initial list  
(612, 606) #Also present on non-theft days  
"""**

* Above results are quite optimistic in terms that I validate our existing results. I want to continue with custom-made modeling using the dynamic programming approach for identifying longest-common sub-sequence. The concept will be take theft days as a sequence and find the visitors with most similar days/sequence. This could also be a good approach for ranking the suspicious visitors.
* *Before moving forward, I must backup my work on GIT. I have version controlled all my work at:* <https://bitbucket.org/saif137/mybi> *in freelancertask folder.*
* The next approach will be