Mini-project in Privacy preserving Data mining in Vertically Partitioned Data

Rola Dabbah

About The Project :

The algorithm that we implemented in this project based on the article: Privacy Preserving Association Rule Mining in Vertically Partitioned Data .
Written by : Jaideep Vaidya , Chris Clifton.
References : the article submitted to The Eighth ACM SIGKDD International Conference of Knowledge Discovery and Data Mining (KDD -2002).

The Problem in general :

We present a privacy preserving algorithm to find frequent itemsets from vertically partitioned data, involving attributes other than the join key across distributed data sets with similar characterization, while limiting the sharing of data between sites.

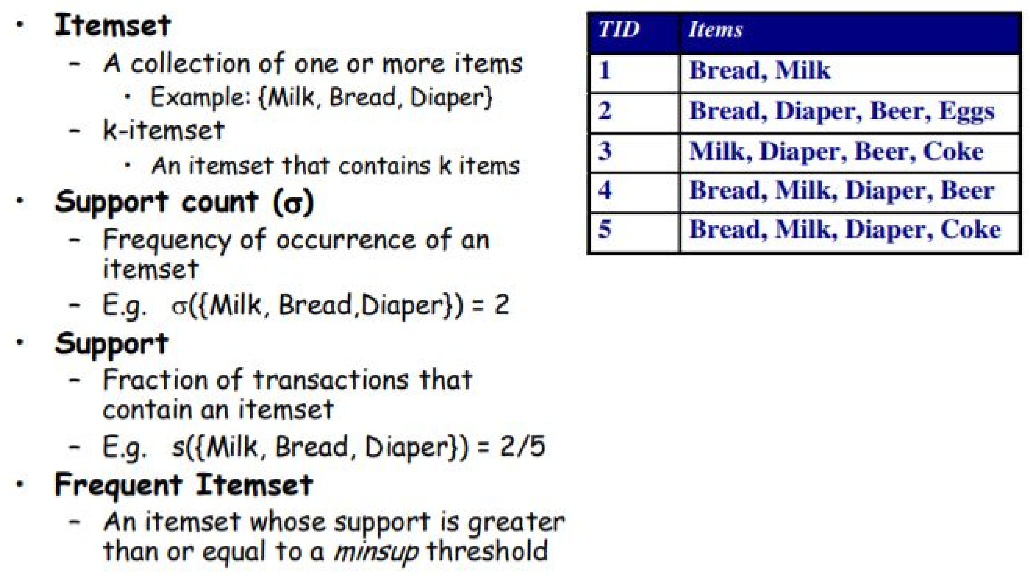
Description of the problem we are dealing with :

Informally , the **problem** that we implemented is to find frequent itemsets across two heterogeneous data sets (both are vertically partitioned) without revealing private data between the sets (using cryptographic protocol to achieve complete zero knowledge leakage for ID3 classification algorithm for two parties with vertically partitioned data).
The first database is designated the primary, and is the initiator of the protocol, the other database is the responder.
There is a join key present in both databases, the remaining attributes are present in one database or the other, but not both.

Description of the mining algorithm :

Our Data mining algorithm is a ID3 algorithm for finding frequent itemsets.
To define what does Frequent itemset means , we should define first:
DataSet - Set that contains data , each row is an itemset.
Itemset – A collection of one or more items
Support count (σ) – frequency of occurrence of an itemset (σ) in all the transactions in the dataSet .
Support – Fraction of transactions that contain an itemset
so **Frequent Itemset** – An itemset whose support is greater than or equal to a minsup threshold.

To make sure that we are fine with the definitions above , let's take a look on this example :



- **Itemset**
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- **Support count (σ)**
  - Frequency of occurrence of an itemset
  - E.g. σ({Milk, Bread,Diaper}) = 2
- **Support**
  - Fraction of transactions that contain an itemset
  - E.g. s({Milk, Bread, Diaper}) = 2/5
- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a *minsup* threshold

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

In other words , The algorithm we have implemented finds the Frequent itemset of a data that is distributed across two dataSets , without sharing information between the two dataSets .
Within this framework, we consider mining boolean frequent itemsets. The absence or presence of an attribute is represented as a 0 or 1. Transactions are integers of 0 and 1; the database can be represented as a matrix of {0,1}.
To find out if a particular itemset is frequent, we count the number of records where the values for all the attributes in the itemset are 1. This translates into a simple mathematical problem, given the following definitions:
Let the total number of attributes be l + m, where database A(the first dataset) has l attributes A1 through Al, and database B (the second dataset) has the remaining m attributes B1 through Bm. Transactions/records are a sequence of l + m 1s or 0s.

Let k be the support threshold required, and n be the total number of transaction/records.
Let $\overline{X}$ and $\overline{Y}$ represent columns in the database, i.e., xi = 1 iff row i has value 1 for attribute X. The scalar (or dot) product of two cardinality n vectors $\overline{X}$ and $\overline{Y}$ is defined as $\overline{X} \cdot \overline{Y} = \sum_{i=1}^{n} xi * yi$
Determining if the two-itemset (XY) is frequent thus reduces to testing if $\overline{X} \cdot \overline{Y} \geq$ k .

We will show how to generalize the above protocol from two itemsets to general frequent itemsets without sharing information other than through scalar product computation.
The generalization of this protocol to a w-itemset is straight-forward. Assume database A has p attributes a1, … ,ap and database B has q attributes b1, .. ,.bq, and we want to compute the frequency of the w = p + q-itemset (a1,… ,ap,b1,… ,bq). Each item in $\overline{X}$ (or $\overline{Y}$) is composed of the product of the corresponding individual elements, i.e., xi = $\prod_{j=1}^{p}$ aj and yi = $\prod_{j=1}^{q}$ bj ($\overline{X} = \{x_1, …, x_n\}$, $\overline{Y} = \{y_1, …, y_n\}$) , this computes $\overline{X}$ and $\overline{Y}$ without sharing information between database A and database B. The scalar product protocol then securely computes the frequency of the entire w-itemset ($\overline{X} \cdot \overline{Y}$) .
We will present an efficient way to compute scalar product $\overline{X} \cdot \overline{Y}$ without either side disclosing its vector. But first we will give an example :
suppose we want to compute if a particular 5-itemset is frequent, with database A having 2 of the attributes, and database B having the remaining 3 attributes. I.e., database A and database B want to know if the itemset l = (Aa,Ab,Ba,Bb,Bc) is frequent. A creates a new vector $\overline{X}$ of cardinality n where $\overline{X} = \overline{Aa} * \overline{Ab}$ (component multiplication) and database B creates a new vector $\overline{Y}$ of cardinality n where $\overline{Y} = \overline{Ba} * \overline{Bb} * \overline{Bc}$. Now the scalar product of $\overline{X}$ and $\overline{Y}$ provides the (in)frequency of the itemset.

To compute scalar product $\overline{X} \cdot \overline{Y}$ (without either side disclosing its vector) we used the following algorithm :

The complete algorithm to find frequent itemsets is:

```
1.  $L_1$ = {large 1-itemsets}
2.  for (k=2; $L_{k-1} \neq \phi$; k++) do begin
3.    $C_k$ = apriori-gen($L_{k-1}$);
4.    for all candidates $c \in C_k$ do begin
5.      if all the attributes in $c$ are entirely at A or B
6.        that party independently calculates $c.count$
7.      else
8.        let A have $l$ of the attributes and B have the remaining $m$ attributes
9.        construct $\vec{X}$ on A's side and $\vec{Y}$ on B's side where $\vec{X} = \prod_{i=1}^{l} \vec{A}_i$ and $\vec{Y} = \prod_{i=1}^{m} \vec{B}_i$
10.       compute $c.count = \vec{X} \cdot \vec{Y} = \sum_{i=1}^{n} x_i * y_i$
11.     endif
12.     $L_k = L_k \cup c | c.count \geq minsup$
13.   end
```

The algorithm explanation :

Step 1: initialization of the algorithm .

L1 contains all the itemsets (attributes) from both Databases (A and B) that passed the support - is greater than or equal to a minsup threshold (without duplications).

Step 3: the function apriori-gen takes the set of large itemsets Lk−1 found in the (k – 1)th pass as an argument and generates the set of candidate itemsets Ck. This is done by generating a superset of possible candidate itemsets and pruning this set.

Steps 4-7: for each c in Ck , check if all items (attributes) in c are in A (or B , but not both) and calculates the c.count from the Database A (or B) .

Steps 8-10: in this step we are dealing with c in Ck that have attributes from Database A and from Database B at the same time .

In this case as we explained above we have : |c| = l + m . while c have l attributes from Database A and the remaining m attributes are from Database B.

we should split the attributes into attributes from Database A and attributes from Database B.

•A = all the attributes from Database A .

•B = all the attributes from Database B .

now , database A should construct $\overline{X} = \prod_{i=1}^{l} \overline{A}i$

and database B should construct $\overline{Y} = \prod_{i=1}^{m} \overline{B}i$.

last step is to compute the c.count which equals to $\overline{X} \cdot \overline{Y} = \sum_{i=1}^{n} xi * yi$.

To compute $\overline{X} \cdot \overline{Y} = \sum_{i=1}^{n} xi * yi$ we used Protocol1 which is cryptographic protocol to achieve complete zero knowledge leakage for ID3 classification algorithm for two parties with vertically partitioned data .

(*) Before we move to Protocol 1 we should notice that only steps 1,3,10 and 12 require sharing information .
Steps 1,3 and 12 reveal no extra information to either party .
We now show how to compute step 10 without revealing information .

Protocol1 (Step 10) :

This protocol gets $\overline{X}$ and $\overline{Y}$ and returns $\overline{X} \cdot \overline{Y} = \sum_{i=1}^{n} xi * yi$ without revealing information .
When we get to protocol 1 we know that we have to compute the scalar product $\overline{X} \cdot \overline{Y}$ from the two databases while $\overline{X} = \{x_1 , \ldots, x_n\}$ and $\overline{Y} = \{y_1 , \ldots, y_n\}$.
This is how it works :

1. agent_A.generate_random_matrix()
2. agent_B.A=agent_A.A
3. X'=agent_A.create_x_tag()
4. S=agent_B.create_S(X')
5. Y'=agent_B.create_y_tag()
6. result=agent_a.calculate_result(S,Y')
7. agent_B.result=agent_A.result
8. return result.

Explanation of the steps above :

1. Database A generates random matrix A = { n x $\frac{n}{2}$ } (the total number of data).
2. Both Database A and DataBase B have access to the matrix A (the random matrix) .
3. - Database A generates random vector R={ $\frac{n}{2}$ * 1 }
   - Database A computes X' = $\overline{X}$ + A*R , which is a vector { n * 1}

   by this step , this is how X' will look like :

$$\begin{array}{l} \langle x_1 + a_{1,1} * R_1 + a_{1,2} * R_2 + \cdots + a_{1,n/2} * R_{n/2} \rangle \\ \langle x_2 + a_{2,1} * R_1 + a_{2,2} * R_2 + \cdots + a_{2,n/2} * R_{n/2} \rangle \\ \vdots \\ \langle x_n + a_{n,1} * R_1 + a_{n,2} * R_2 + \cdots + a_{n,n/2} * R_{n/2} \rangle \end{array}$$

   note : in our implementation we used the number 10 instead of $\frac{n}{2}$, because its not really affect the results and its more efficient in terms of running time .

- Database A sens all the n values (X') to Database B .

4. - Database B computes S = X' * $\overline{Y}$ (S is a scalar) by multiplying each value received with the corresponding y value and summing the result .

$$
\begin{aligned}
S = \\
&(x_1 * y_1 + x_2 * y_2 + \cdots + x_n * y_n) \\
&+(y_1 * a_{1,1} * R_1 + y_1 * a_{1,2} * R_2 + \\
&\quad \cdots + y_1 * a_{1,n/2} * R_{n/2}) \\
&+(y_2 * a_{2,1} * R_1 + y_2 * a_{2,2} * R_2 + \\
&\quad \cdots + y_2 * a_{2,n/2} * R_{n/2}) \\
&\vdots \\
&+(y_n * a_{n,1} * R_1 + y_n * a_{n,2} * R_2 + \\
&\quad \cdots + y_n * a_{n,n/2} * R_{n/2})
\end{aligned}
$$

5. - DataBase B computes the following n/2 (10 in our implementation) :
   This is the Y' - the result of $A^t * \overline{Y}$:

$$
\begin{aligned}
&\langle a_{1,1} * y_1 + a_{2,1} * y_2 + \cdots + a_{n,1} * y_n \rangle \\
&\langle a_{1,2} * y_1 + a_{2,2} * y_2 + \cdots + a_{n,2} * y_n \rangle \\
&\vdots \\
&\langle a_{1,n/2} * y_1 + a_{2,n/2} * y_2 + \cdots + a_{n,n/2} * y_n \rangle
\end{aligned}
$$

- Database B sends the n/2 (Y') and S to Database B.

6. in Database A :
   S can be written as follows :

$$
\begin{aligned}
S = \\
&y_1 * (x_1 + a_{1,1} * R_1 + a_{1,2} * R_2 + \cdots + a_{1,n/2} * R_{n/2}) \\
&+y_2 * (x_2 + a_{2,1} * R_1 + a_{2,2} * R_2 + \cdots + a_{2,n/2} * R_{n/2}) \\
&\vdots \\
&+y_n * (x_n + a_{n,1} * R_1 + a_{n,2} * R_2 + \cdots + a_{n,n/2} * R_{n/2})
\end{aligned}
$$

Simplifying the equation further , and grouping the $xi * yi$ terms , we get :

$$
\begin{aligned}
S = \\
&(x_1 * y_1 + x_2 * y_2 + \cdots + x_n * y_n) \\
&+(y_1 * a_{1,1} * R_1 + y_1 * a_{1,2} * R_2 + \\
&\quad \cdots + y_1 * a_{1,n/2} * R_{n/2}) \\
&+(y_2 * a_{2,1} * R_1 + y_2 * a_{2,2} * R_2 + \\
&\quad \cdots + y_2 * a_{2,n/2} * R_{n/2}) \\
&\vdots \\
&+(y_n * a_{n,1} * R_1 + y_n * a_{n,2} * R_2 + \\
&\quad \cdots + y_n * a_{n,n/2} * R_{n/2})
\end{aligned}
$$

The first line of the R.H.S. can be written as $\sum_{i=1}^{n} xi * yi$, the desired final result. In the remaining portion, we group the multiplicative components vertically and rearrange the equation to factor out all the Ri values, giving:

$$
\begin{aligned}
S = \\
&\sum_{i=1}^{n} x_i * y_i \\
&+R_1 * (a_{1,1} * y_1 + a_{2,1} * y_2 + \cdots + a_{n,1} * y_n) \\
&+R_2 * (a_{1,2} * y_1 + a_{2,2} * y_2 + \cdots + a_{n,2} * y_n) \\
&\vdots \\
&+R_{n/2} * (a_{1,n/2} * y_1 + a_{2,n/2} * y_2 + \cdots + a_{n,n/2} * y_n)
\end{aligned}
$$

Database A already knows the n/2 (10) Ri values. The n/2 (10) other values sent by Database B are the same as the coefficients on the R.H.S. of the above equation. Database A multiplies the n/2 (10) values received from database B with the corresponding Ri and subtracts the sum from S, giving the desired result.

7. Database A reports the result to Database B.
8. The protocol returns the result .

Note :

- **Sending information between the databases is encrypted , none of the sided can reveal information about the other .**
- **In each usage of the protocol 1 (in each time we face an itemset that have attributes from the both databases (A and B)) the algorithm generates new random matrix A , and new random vector R , this makes the algorithm safe , and the values cannot be revealed .**

Preprocessing

□ We start dealing with csv files .
The csv file contains 1085 items ,was divided into 2 files : file_A, file_B , each file have inserted into database table respectively :

- sex_offenders_A - with the fields :
  first_last_name , gender, age, month , maritial_status .
- sex_offenders_B - with the fields :
  first_last_name , state , education , criminal_record , race , work , economic_status .

□ We downloaded an csv file that contains a data about "Sex offenders in USA" , we looked up and we found a statistics for the most states in USA that have Sex Offenders issues , and according to this statistics we added the state field in the file_B, we did that so we can have a realistic results .

□ using the tables above ,we built new database tables - " boolean tables " : sex_offenders_A_extended , sex_offenders_B_extended .
Boolean tables means that the absence or the presence of an attribute is represented as 0 or 1 .Transactions are integers of 0 and 1 ; the database can be represented as matrix of {0,1}. To find out if particular itemset is frequent , we count the number of records where the values for all the attributes in the itemset are 1 .

- sex_offenders_A_extended - with the fields :
  name,
  MALE, FEMALE,
  age_11_20, age_21_30 , age_31_40, age_41_50, age_51_60, age_61_70, age_71_80, age_81_90, age_91_100,
  January, February, March, April, May, June, July, August, September, October, November, December,
  divorced, married, single.
- sex_offenfers_B_extended - with the fields :
  name ,
  Wisconsin , Delaware, Oregon,Arkansas, Michigan, South_Dakota, Wyoming, other,
  high_school, home_schooled, academic, primary_school, middle_school, none,
  Has_Criminal_record, doesnt_have_Criminal_record,
  WHITE, BLACK,
  works, doesnt_work,
  low, mid, high

While inserting into the first tables , we have merged the fields first name and last name to be an one field .

 While inserting into boolean tables , we have
 Table A :
while inserting the fields from sex_offenders_A table to sex_offenders_A_extended , we have make changes in the inserted values (age , month) we used range values over exact numerical values.

For example : let's say that we have file_A that contains the data :

first,last,gender,age,birthday,marital_status
name1,family1,female,30,1/1/1987,married
name2,family2,id2,male,56,4/11/1869,married

gender can have 2 valaues : {male , female }
age -9 values: {age_11_20,..age_91_100}
month - 12 values : {January,..,December}
marital status : {married , single , divorced}

 we inserted them to a sex_offenders_A table in this way :

sex_offenders_A :

first_last_name	gender	age	month	marital status
name1_family1	female	30	1	married
name2_family2	male	56	11	married

 then we inserted them into a sex_offenders_A_extended in this way :

first_last_name	male	female	age_11_20	...	age_31_40	...	age_91_100	January	...	December	married	divorced	single
name1_family1	0	1	0		1	0	0	1		0	1	0	0
name2_family2	1	0	0		0		0	0		0	1	0	0

and he same for file_B and the tables sex_offenders_B ,sex_offenders_B_extended .
states :{Wisconsin , Delaware, Oregon,Arkansas, Michigan, South_Dakota, Wyoming, other}
education :{high_school, home_schooled, academic, primary_school, middle_school, none}

criminal record : {Has_Criminal_record, doesnt_have_Criminal_record}
race : {WHITE, BLACK}
word : {works, doesnt_work}
economic status : {low, mid, high} .

=> The preprocessing performed on each part of the database is intended to serve many purposes:

1) to make the dealing with massive amounts of data easier .
2) to make sending information between the two sites (databaseA , databaseB) faster ,easy to deal with and for the encryption .
3) to translate the problem into a simple mathematical problem .
4) to get information from the database in a way that you can bind the databases effectively.

Software Description :

The language we used : Python (using sqlite3) .
The software that we implemented contains 5 classes :

1. **main.py:**
   This class is (the main class) responsible for running the whole algorithm .
   1. gets the csv files name , support , total number of data from the user input .
   2. creates the Database tables according to the csv files names.
   3. creates the boolean (extended) Database tables .
   4. creates an agentA , agentB objects .
   5. run the main_algorithm(..) from algorithm.py class with the given params above.
2. **databaseA.py:**
   This file contains a class named agent_A that is responsible for all operations that are done with data tables of dataBase A , and two more functions that creates the database tables .
   The class agent_A contains the following fields :
   1. A - random matrix {total x 10} - used by both Databases .
   2. R - random vector {10 x 1} - private to Database A .
   3. X - boolean vector - the multiplication of the boolean attributes .
   4. result - the count of an set that contains attributes that are belongs to both databases.

   The functions in this class :
   1. generate_random_matrix : creates a matrix {total x 10 } with random numbers .
   2. create_x_tag :
      as we mentioned above , X' = X + A*R , this functions computes the result of A*R (the result is a vector { total x 1}), returns the result of adding X to A*R .
   3. get_xvectors(c) :
      This function takes as parameter a set of attributes (c) that belongs to the Database A , returns a set of all the boolean vectors (columns) that presents those attributes from "sex_offenders_A_extended" table .
   4. calculate_result :
      as we wrote in the previous part that the result is calculated in A's side ,and result = S-T = X'*Y -T , when T = R*Y'.
      this function computes the S-T and returns it .

   The functions that creates the tables :
   1. create_sex_offenders_table_A:
      reads the data columns from the first csv file and inserts it into columns in database table named : "sex_offenders_A".

2. create_sex_offenders_A_boolean_table:
   reads the columns from the "sex_offenders_A" table and inserts it into boolean table named : "sex_offenders_A_extended" .

3. **databaseB.py:**
   This file contains a class named agent_B that is responsible for all operations that are done with data tables of dataBase B , and two more functions that creates the database tables .
   The class agent_B contains the following fields
   1. A - random matrix {total x 10}.
   2. Y - boolean vector - the multiplication of the boolean attributes .
   3. result - the count of an set that contains attributes that are belongs to both databases.

   The functions in this class :
   1. create_y_tag:
   2. as we mentioned above , Y' = A*Y , this functions computes the result of A*Y (the result is a vector { total x 1}) and returns it.
   3. create_s :
      as we wrote in the previous part that S is calculated in B's side ,and S = X'*Y .
      This function computes S and returns it .
   4. get_yvectors(c):
      This function takes as parameter a set of attributes (c) that belongs to the Database B , returns a set of all the boolean vectors (columns) that presents those attributes from the "sex_offenders_B_extended" table .

   The functions that creates the tables :
   1. create_sex_offenders_table_B :
      reads the data columns from the second csv file and inserts it into columns in database table named : "sex_offenders_B".
   2. create_sex_offenders_B_boolean_table:
      reads the columns from the "sex_offenders_B" table and inserts it into boolean table named : "sex_offenders_B_extended" .

4. **algorithm.py:**
   This class organizes the whole process in the required order off calling functions and passing data between the two data sets ,includes running the preserving security protocol.
   The functions in this class :
   1. protocol1 :
      This is the preserving security protocol that manages the functions that have access to the Database of each DataSet using agent_A, agent_B .
      This is how it works :

```
1.agent_A.generate_random_matrix()
2.agent_B.A=agent_A.A
3.X'=agent_A.create_x_tag()
4.S=agent_B.create_S(X')
5.Y'=agent_B.create_y_tag()
6.result=agent_a.calculate_result(S,Y')
7.agent_B.result=agent_A.result
8.return result
```

(explanation about the protocol is attached above in the description of the algorithm).

2. main_algorithm :
   This is the algorithm described above in the "description of the algorithm" section ,and it works exactly the same .
3. all_attributes_in_A(a_attr, c) :
   gets as parameter all the attributes of database A and a set c , returns true of all the the attributes in c existed in a_attr.
4. all_attributes_in_B(b_attr, c):
   gets as parameter all the attributes of database B and a set c , returns true of all the the attributes in c existed in b_attr.
5. split_attributes (a-attr ,c) :
   gets as parameter all the attributes of database A and a set c , returns 2 sets , the first contains all the attributes in c that belongs to a_attr , and the other is all the attributes in c that don't belong to a_attr (belongs to b_attr).
6. filter_with_supp(dict,sup,total) :
   get as a parameter dictionary with the values (c:c.count) , sup , total , and returns a filtered dictionary with the values from the dict that is > = sup * total .
7. L1(file_a,file_b) :
   This function is the initialization of the main_algorithm process . takes as parameters two csv files, iterates them and gets all the attributes from both without duplications.
   generates the 1-itemsets from all the attributes in database A and database B , and returns a dictionary contains the generated elements with their count that are >= sup*total (returns a filtered 1-itemsets dictionary)
8. generateCk(Lk-1):
   gets as parameter a set of k-1 itemsets that passed the filterelization , and generates a set of k-itemsets from the given Lk-1 sets (without filter).

5. **utils.py :**
   In this class we have all the common functions that we used in both databaseA and databaseB .
   This functions are general functions , and do not reveal data that related to the DataSet , because it does computational operations on the boolean vectors . The main functions in this class :
   1. mult_vecs(arr) :
      gets as parameter array of vectors and computes the multiplication of all the vectors in the arr => arr = {v0,v1,v2,..vn} this functions returns the $\prod_{i=0}^{n} vi$.
   2. scalar_mult(v1,v2):
      takes two vectors as arguments and returns scalar as a result of the multiplication $\prod_{i=1}^{n} xi * yi$ when v1={x1,x2,..xn} , v2={y1,y2,..,yn} .
   3. mult_two_vectors(v1,v2):
      takes two vectors and returns vector contains the result of vector multiplication , means returned_vec= v1[i]*v2[i] , 0<=i<n
   4. get_transpose(A) :
      takes matrix A as argument and returns A^t (transpose).
   5. get_count_from_vector (v) :
      returns the count of 1's records in vector v .

References :

https://dl.acm.org/citation.cfm?id=775142

About the data we choose :

We are dealing with data about "Sex offenders in USA " .
We have downloaded a data that contains an information about this subject , and devided it into 2 datasets so we can implement our project .

Communication Analysis

For a complete communication analysis of the process, we first analyze the communication cost of the component scalar product protocol, then analyze the communication cost of the entire association rule mining algorithm.
In Protocol I, A sends one message with n values. B replies with n/2+1 (10 + 1 in our implementation) values. A then sends the result. This gives three communication rounds. The bitwise communication cost is O(n) (n is the total number of data in dataset) with the constant being 3/2. The masking process for boolean data adds two rounds, with messages half the size of the first two rounds.

We present an example with total data number =10 :

PrintOuts :

input :

first csv file :

```
LAST,FIRST,GENDER,AGE,MONTH ,MARITAL STATUS
MCCULLOUGH,JOSHUA,MALE,BLACK,18,Y,,
RAMAGE-CANNON,DAVID,MALE,WHITE,18,Y,,
RODRIGUEZ,MIGUEL,MALE,WHITE,18,Y,,
ARROYO,CARLOS,MALE,WHITE,24,Y,,
PAGE,ANDREW,MALE,WHITE,24,N,,
VELAZQUEZ,PEDRO,MALE,WHITE,24,Y,,
CLARK,TYRICE,MALE,BLACK,25,Y,,
MCKEE,GREGORY,MALE,BLACK,26,Y,,
WILLIAMS,MARK,MALE,BLACK,26,N,,s
```

second csv file :

```
LAST,FIRST,STATE ,EDUCATION,CRIMIdoesnt_have_Criminal_recordAL RECORD,RACE,WORK,Economic situation
MCCULLOUGH,JOSHUA,Arkansas,home_schooled,Has_Criminal_record,BLACK,works,high
RAMAGE-CANNON,DAVID,Wisconsin,high_school,Has_Criminal_record,BLACK,works,high
RODRIGUEZ,MIGUEL,other,high_school,Has_Criminal_record,BLACK,works,high
ARROYO,CARLOS,Arkansas,high_school,doesnt_have_Criminal_record,WHITE,works,high
PAGE,ANDREW,Wisconsin,middle_school,Has_Criminal_record,BLACK,works,high
VELAZQUEZ,PEDRO,Wyoming,home_schooled,Has_Criminal_record,WHITE,works,high
CLARK,TYRICE,other,home_schooled,Has_Criminal_record,BLACK,doesnt_work,low
MCKEE,GREGORY,Delaware,middle_school,Has_Criminal_record,WHITE,works,high
WILLIAMS,MARK,Delaware,high_school,doesnt_have_Criminal_record,BLACK,works,high
```

Output:

```
insert the first Data csv file name  : a.csv
insert the second  Data csv file name  : b.csv
insert the total number of data: 10
insert a support value (percentage) : 0.22
Creating database sex_offendersA
Sex offenders A database created successfully
Creating database sex_offendersB
Sex offenders B database created successfully
Creating database sex_offenders_A_extended
Sex offenders extended database created successfully
Creating database sex_offenders_B_extended
Sex offenders extended database B created successfully
in main algorithm...
in L1...
C1 Items  : {'GENDER': 1, 'AGE': 1, 'MONTH ': 1, 'MARITAL STATUS': 1, 'MALE': 9, 'BLACK': 10, '18': 3, 'Y': 7, 'WHITE': 8, '24': 3, 'N': 2, '25': 1, '26': 2,
's': 1, 'STATE ': 1, 'EDUCATION': 1, 'CRIMIdoesnt_have_Criminal_recordAL RECORD': 1, 'RACE': 1, 'WORK': 1, 'Economic situation': 1, 'Arkansas': 2,
'home_schooled': 3, 'Has_Criminal_record': 7, 'works': 8, 'high': 8, 'Wisconsin': 2, 'high_school': 4, 'other': 2, 'doesnt_have_Criminal_record': 2,
'middle_school': 2, 'Wyoming': 1, 'doesnt_work': 1, 'low': 1, 'Delaware': 2}
C1 Length :  34

L1  : {'MALE': 9, 'BLACK': 10, '18': 3, 'Y': 7, 'WHITE': 8, '24': 3, 'home_schooled': 3, 'Has_Criminal_record': 7, 'works': 8, 'high': 8, 'high_school': 4}
L1 Length :  11

  in generateCK...
C2:  [('MALE', 'Y'), ('MALE', 'WHITE'), ('MALE', 'home_schooled'), ('MALE', 'works'), ('MALE', 'high'), ('MALE', 'high_school'), ('BLACK', 'MALE'),
('BLACK', 'Y'), ('BLACK', 'WHITE'), ('BLACK', 'home_schooled'), ('BLACK', 'Has_Criminal_record'), ('BLACK', 'works'), ('BLACK', 'high'), ('BLACK',
'high_school'), ('18', 'MALE'), ('18', 'BLACK'), ('18', 'Y'), ('18', 'WHITE'), ('18', '24'), ('18', 'home_schooled'), ('18', 'Has_Criminal_record'), ('18', 'works'),
('18', 'high'), ('18', 'high_school'), ('Y', 'home_schooled'), ('Y', 'works'), ('Y', 'high'), ('Y', 'high_school'), ('WHITE', 'Y'), ('WHITE', 'home_schooled'),
('WHITE', 'works'), ('WHITE', 'high'), ('WHITE', 'high_school'), ('24', 'MALE'), ('24', 'BLACK'), ('24', 'Y'), ('24', 'WHITE'), ('24', 'home_schooled'), ('24',
'Has_Criminal_record'), ('24', 'works'), ('24', 'high'), ('24', 'high_school'), ('home_schooled', 'works'), ('Has_Criminal_record', 'MALE'),
('Has_Criminal_record', 'Y'), ('Has_Criminal_record', 'WHITE'), ('Has_Criminal_record', 'home_schooled'), ('Has_Criminal_record', 'works'),
('Has_Criminal_record', 'high'), ('Has_Criminal_record', 'high_school'), ('high', 'home_schooled'), ('high', 'works'), ('high', 'high_school'), ('high_school',
'home_schooled'), ('high_school', 'works')]
length :  55

case all attributes in DB_A , c:  ('MALE', 'Y')
count :  {('MALE', 'Y'): 9}

the attributes are from DB_A and DB_B , c:  ('MALE', 'home_schooled')
DB_A :  ['MALE']
DB_B :  ['home_schooled']
vectors from DB_A:  [[(1,), (1,), (1,), (1,), (1,), (1,), (1,), (1,), (1,)]]
vectors from DB_B:  [[(1,), (0,), (0,), (0,), (0,), (1,), (1,), (0,), (0,)]]
count :  {('MALE', 'home_schooled'): 3}

case all attributes in DB_B , c:  ('home_schooled', 'works')
count :  {('home_schooled', 'works'): 2}

-----------------------------------
  in generateCK...
C3:  [('MALE', 'WHITE', 'Y'), ('MALE', 'Y', 'home_schooled'), ('MALE', 'Y', 'works'), ('MALE', 'Y', 'high'), ('MALE', 'Y', 'high_school'), ('MALE', 'WHITE',
'Y'), ('MALE', 'WHITE', 'home_schooled'), ('MALE', 'WHITE', 'works'), ('MALE', 'WHITE', 'high'), ('MALE', 'WHITE', 'high_school'), ('MALE', 'Y',
'home_schooled'), ('MALE', 'WHITE', 'home_schooled'), ('MALE', 'home_schooled', 'works'), ('MALE', 'high', 'home_schooled'), ('MALE', 'high_school',
'home_schooled'), ('MALE', 'Y', 'works'), ('MALE', 'WHITE', 'works'), ('MALE', 'home_schooled', 'works'), ('MALE', 'high', 'works'), ('MALE', 'high_school',
'works'), ('MALE', 'Y', 'high'), ('MALE', 'WHITE', 'high'), ('MALE', 'high', 'home_schooled'), ('MALE', 'high', 'works'), ('MALE', 'high', 'high_school'),
('MALE', 'Y', 'high_school'), ('MALE', 'WHITE', 'high_school'), ('MALE', 'high_school', 'home_schooled'), ('MALE', 'high_school', 'works'), ('MALE', 'high',
'high_school'), ('BLACK', 'MALE', 'Y'), ('BLACK', 'MALE', 'WHITE'), ('BLACK', 'MALE', 'Y'), ('BLACK', 'WHITE', 'Y'), ('BLACK', 'MALE', 'WHITE'),
('BLACK', 'WHITE', 'Y'), ('18', 'BLACK', 'MALE'), ('18', 'MALE', 'Y'), ('18', 'MALE', 'WHITE'), ('18', '24', 'MALE'), ('18', 'BLACK', 'MALE'), ('18', 'BLACK',
'Y'), ('18', 'BLACK', 'WHITE'), ('18', '24', 'BLACK'), ('18', 'MALE', 'Y'), ('18', 'BLACK', 'Y'), ('18', 'WHITE', 'Y'), ('18', '24', 'Y'), ('18', 'MALE', 'WHITE'),
('18', 'BLACK', 'WHITE'), ('18', 'WHITE', 'Y'), ('18', '24', 'WHITE'), ('18', '24', 'MALE'), ('18', '24', 'BLACK'), ('18', '24', 'Y'), ('18', '24', 'WHITE'), ('24',
'BLACK', 'MALE'), ('24', 'MALE', 'Y'), ('24', 'MALE', 'WHITE'), ('24', 'BLACK', 'MALE'), ('24', 'BLACK', 'Y'), ('24', 'BLACK', 'WHITE'), ('24', 'MALE', 'Y'),
('24', 'BLACK', 'Y'), ('24', 'WHITE', 'Y'), ('24', 'MALE', 'WHITE'), ('24', 'BLACK', 'WHITE'), ('24', 'WHITE', 'Y'), ('Has_Criminal_record', 'MALE',
'home_schooled'), ('Has_Criminal_record', 'MALE', 'works'), ('Has_Criminal_record', 'MALE', 'high'), ('Has_Criminal_record', 'MALE', 'high_school'),
('Has_Criminal_record', 'MALE', 'home_schooled'), ('Has_Criminal_record', 'home_schooled', 'works'), ('Has_Criminal_record', 'high', 'home_schooled'),
('Has_Criminal_record', 'high_school', 'home_schooled'), ('Has_Criminal_record', 'MALE', 'works'), ('Has_Criminal_record', 'home_schooled', 'works'),
('Has_Criminal_record', 'high', 'works'), ('Has_Criminal_record', 'high_school', 'works'), ('Has_Criminal_record', 'MALE', 'high'), ('Has_Criminal_record',
'high', 'home_schooled'), ('Has_Criminal_record', 'high', 'works'), ('Has_Criminal_record', 'high', 'high_school'), ('Has_Criminal_record', 'MALE',
'high_school'), ('Has_Criminal_record', 'high_school', 'home_schooled'), ('Has_Criminal_record', 'high_school', 'works'), ('Has_Criminal_record', 'high',
'high_school'), ('high', 'home_schooled', 'works'), ('high', 'high_school', 'home_schooled'), ('high', 'home_schooled', 'works'), ('high', 'high_school', 'works'),
('high', 'high_school', 'home_schooled'), ('high', 'high_school', 'works'), ('high_school', 'home_schooled', 'works'), ('high_school', 'home_schooled', 'works')]
Length :  96

-----------------------------------
  in generateCK...
C4:  [('MALE', 'WHITE', 'Y', 'home_schooled'), ('MALE', 'WHITE', 'Y', 'works'), ('MALE', 'WHITE', 'Y', 'high'), ('MALE', 'WHITE', 'Y', 'high_school'),
('MALE', 'Y', 'home_schooled', 'works'), ('MALE', 'Y', 'high', 'home_schooled'), ('MALE', 'Y', 'high_school', 'home_schooled'), ('MALE', 'Y', 'home_schooled',
'works'), ('MALE', 'Y', 'high', 'works'), ('MALE', 'Y', 'high_school', 'works'), ('MALE', 'Y', 'high', 'home_schooled'), ('MALE', 'Y', 'high', 'works'), ('MALE',
'Y', 'high', 'high_school'), ('MALE', 'Y', 'high_school', 'home_schooled'), ('MALE', 'Y', 'high_school', 'works'), ('MALE', 'Y', 'high', 'high_school'), ('MALE',
'WHITE', 'Y', 'home_schooled'), ('MALE', 'WHITE', 'home_schooled', 'works'), ('MALE', 'WHITE', 'high', 'home_schooled'), ('MALE', 'WHITE',
'high_school', 'home_schooled'), ('MALE', 'WHITE', 'Y', 'works'), ('MALE', 'WHITE', 'home_schooled', 'works'), ('MALE', 'WHITE', 'high', 'works'),
('MALE', 'WHITE', 'high_school', 'works'), ('MALE', 'WHITE', 'Y', 'high'), ('MALE', 'WHITE', 'high', 'home_schooled'), ('MALE', 'WHITE', 'high', 'works'),
('MALE', 'WHITE', 'high', 'high_school'), ('MALE', 'WHITE', 'Y', 'high_school'), ('MALE', 'WHITE', 'high_school', 'home_schooled'), ('MALE', 'WHITE',
'high_school', 'works'), ('MALE', 'WHITE', 'high', 'high_school'), ('MALE', 'high', 'high_school', 'works'), ('MALE', 'high', 'high_school', 'works'), ('BLACK',
'MALE', 'WHITE', 'Y'), ('BLACK', 'MALE', 'WHITE', 'Y'), ('18', 'BLACK', 'MALE', 'Y'), ('18', 'BLACK', 'MALE', 'WHITE'), ('18', 'MALE', 'WHITE', 'Y'), ('18',
'MALE', 'WHITE', 'Y'), ('18', '24', 'BLACK', 'MALE'), ('18', '24', 'MALE', 'Y'), ('18', '24', 'MALE', 'WHITE'), ('18', 'BLACK', 'MALE', 'Y'), ('18', 'BLACK',
'WHITE', 'Y'), ('18', 'BLACK', 'MALE', 'WHITE'), ('18', 'BLACK', 'WHITE', 'Y'), ('18', '24', 'BLACK', 'MALE'), ('18', '24', 'BLACK', 'Y'), ('18', '24', 'BLACK',
'WHITE'), ('18', '24', 'MALE', 'Y'), ('18', '24', 'BLACK', 'Y'), ('18', '24', 'WHITE', 'Y'), ('18', '24', 'MALE', 'WHITE'), ('18', '24', 'BLACK', 'WHITE'), ('18',
'24', 'WHITE', 'Y'), ('24', 'BLACK', 'MALE', 'Y'), ('24', 'BLACK', 'MALE', 'WHITE'), ('24', 'MALE', 'WHITE', 'Y'), ('24', 'MALE', 'WHITE', 'Y'), ('24',
'BLACK', 'MALE', 'Y'), ('24', 'BLACK', 'WHITE', 'Y'), ('24', 'BLACK', 'MALE', 'WHITE'), ('24', 'BLACK', 'WHITE', 'Y'), ('Has_Criminal_record', 'MALE',
'home_schooled', 'works'), ('Has_Criminal_record', 'MALE', 'high', 'home_schooled'), ('Has_Criminal_record', 'MALE', 'home_schooled', 'works'),
('Has_Criminal_record', 'MALE', 'high', 'works'), ('Has_Criminal_record', 'MALE', 'high', 'home_schooled'), ('Has_Criminal_record', 'MALE', 'high', 'works'),
('Has_Criminal_record', 'high', 'home_schooled', 'works'), ('Has_Criminal_record', 'high', 'high_school', 'home_schooled'), ('Has_Criminal_record',
'high_school', 'home_schooled', 'works'), ('Has_Criminal_record', 'high', 'home_schooled', 'works'), ('Has_Criminal_record', 'high', 'high_school', 'works'),
('Has_Criminal_record', 'high_school', 'home_schooled', 'works'), ('Has_Criminal_record', 'high', 'high_school', 'home_schooled'), ('Has_Criminal_record',
'high', 'high_school', 'works'), ('high', 'high_school', 'home_schooled', 'works'), ('high', 'high_school', 'home_schooled', 'works')]
Length :  80

-----------------------------------
  in generateCK...
C5:  [('MALE', 'WHITE', 'Y', 'home_schooled', 'works'), ('MALE', 'WHITE', 'Y', 'high', 'home_schooled'), ('MALE', 'WHITE', 'Y', 'high_school',
'home_schooled'), ('MALE', 'WHITE', 'Y', 'home_schooled', 'works'), ('MALE', 'WHITE', 'Y', 'high', 'works'), ('MALE', 'WHITE', 'Y', 'high_school', 'works'),
('MALE', 'WHITE', 'Y', 'high', 'home_schooled'), ('MALE', 'WHITE', 'Y', 'high', 'works'), ('MALE', 'WHITE', 'Y', 'high', 'high_school'), ('MALE', 'WHITE',
'Y', 'high_school', 'home_schooled'), ('MALE', 'WHITE', 'Y', 'high_school', 'works'), ('MALE', 'WHITE', 'Y', 'high', 'high_school'), ('MALE', 'Y', 'high',
'high_school', 'works'), ('MALE', 'Y', 'high', 'high_school', 'works'), ('MALE', 'WHITE', 'high', 'high_school', 'works'), ('MALE', 'WHITE', 'high',
'high_school', 'works'), ('18', 'BLACK', 'MALE', 'WHITE', 'Y'), ('18', 'BLACK', 'MALE', 'WHITE', 'Y'), ('18', '24', 'BLACK', 'MALE', 'Y'), ('18', '24', 'BLACK',
'MALE', 'WHITE'), ('18', '24', 'MALE', 'WHITE', 'Y'), ('18', '24', 'MALE', 'WHITE', 'Y'), ('18', '24', 'BLACK', 'MALE', 'Y'), ('18', '24', 'BLACK', 'WHITE',
'Y'), ('18', '24', 'BLACK', 'MALE', 'WHITE'), ('18', '24', 'BLACK', 'WHITE', 'Y'), ('24', 'BLACK', 'MALE', 'WHITE', 'Y'), ('24', 'BLACK', 'MALE', 'WHITE',
'Y'), ('Has_Criminal_record', 'high', 'high_school', 'home_schooled', 'works'), ('Has_Criminal_record', 'high', 'high_school', 'home_schooled', 'works')]
Length :  30

-----------------------------------
  in generateCK...
C6:  [('MALE', 'WHITE', 'Y', 'high', 'high_school', 'works'), ('MALE', 'WHITE', 'Y', 'high', 'high_school', 'works'), ('18', '24', 'BLACK', 'MALE', 'WHITE',
'Y'), ('18', '24', 'BLACK', 'MALE', 'WHITE', 'Y')]
Length :  4

-----------------------------------
  in generateCK...
C7:  []
Length :  0

-----------------------------------
Final Result  : {('MALE', 'Y'): 9, ('MALE', 'WHITE'): 9, ('MALE', 'home_schooled'): 3, ('MALE', 'works'): 8, ('MALE', 'high'): 8, ('MALE', 'high_school'): 4,
('BLACK', 'MALE'): 9, ('BLACK', 'Y'): None, ('BLACK', 'WHITE'): None, ('18', 'MALE'): 9, ('18', 'BLACK'): None, ('18', 'Y'): None, ('18', 'WHITE'): None, ('18',
'24'): None, ('WHITE', 'Y'): None, ('24', 'MALE'): 9, ('24', 'BLACK'): None, ('24', 'Y'): None, ('24', 'WHITE'): None, ('home_schooled', 'works'): 2,
('Has_Criminal_record', 'MALE'): 7, ('Has_Criminal_record', 'home_schooled'): 3, ('Has_Criminal_record', 'works'): 6, ('Has_Criminal_record', 'high'): 6,
('Has_Criminal_record', 'high_school'): 2, ('high', 'home_schooled'): 2, ('high', 'works'): 8, ('high', 'high_school'): 4, ('high_school', 'home_schooled'): 0,
('high_school', 'works'): 4, ('MALE', 'WHITE', 'Y'): 9, ('MALE', 'Y', 'home_schooled'): 3, ('MALE', 'Y', 'works'): 8, ('MALE', 'Y', 'high'): 8, ('MALE', 'Y',
'high_school'): 4, ('MALE', 'WHITE', 'home_schooled'): 3, ('MALE', 'WHITE', 'works'): 8, ('MALE', 'WHITE', 'high'): 8, ('MALE', 'WHITE', 'high_school'): 4,
('MALE', 'high', 'works'): 8, ('MALE', 'high_school', 'works'): 4, ('MALE', 'high', 'high_school'): 4, ('BLACK', 'MALE', 'Y'): 9, ('BLACK', 'MALE', 'WHITE'): 9,
('BLACK', 'WHITE', 'Y'): None, ('18', 'BLACK', 'MALE'): 9, ('18', 'MALE', 'Y'): 9, ('18', 'MALE', 'WHITE'): 9, ('18', '24', 'MALE'): 9, ('18', 'BLACK', 'Y'):
None, ('18', 'BLACK', 'WHITE'): None, ('18', '24', 'BLACK'): None, ('18', 'WHITE', 'Y'): None, ('18', '24', 'Y'): None, ('18', '24', 'WHITE'): None, ('24',
'BLACK', 'MALE'): 9, ('24', 'MALE', 'Y'): 9, ('24', 'MALE', 'WHITE'): 9, ('24', 'BLACK', 'Y'): None, ('24', 'BLACK', 'WHITE'): None, ('24', 'WHITE', 'Y'):
None, ('Has_Criminal_record', 'MALE', 'home_schooled'): 3, ('Has_Criminal_record', 'MALE', 'works'): 6, ('Has_Criminal_record', 'MALE', 'high'): 6,
('Has_Criminal_record', 'home_schooled', 'works'): 2, ('Has_Criminal_record', 'high', 'home_schooled'): 2, ('Has_Criminal_record', 'high_school',
'home_schooled'): 0, ('Has_Criminal_record', 'high', 'works'): 6, ('Has_Criminal_record', 'high_school', 'works'): 2, ('Has_Criminal_record', 'high',
'high_school'): 2, ('high', 'home_schooled', 'works'): 2, ('high', 'high_school', 'home_schooled'): 0, ('high', 'high_school', 'works'): 4, ('high_school',
'home_schooled', 'works'): 0, ('MALE', 'WHITE', 'Y', 'home_schooled'): 3, ('MALE', 'WHITE', 'Y', 'works'): 8, ('MALE', 'WHITE', 'Y', 'high'): 8, ('MALE',
'WHITE', 'Y', 'high_school'): 4, ('MALE', 'Y', 'high', 'works'): 8, ('MALE', 'Y', 'high_school', 'works'): 4, ('MALE', 'Y', 'high', 'high_school'): 4, ('MALE',
'WHITE', 'high', 'works'): 8, ('MALE', 'WHITE', 'high_school', 'works'): 4, ('MALE', 'WHITE', 'high', 'high_school'): 4, ('MALE', 'high', 'high_school',
'works'): 4, ('BLACK', 'MALE', 'WHITE', 'Y'): 9, ('18', 'BLACK', 'MALE', 'Y'): 9, ('18', 'BLACK', 'MALE', 'WHITE'): 9, ('18', 'MALE', 'WHITE', 'Y'): 9, ('18',
'24', 'BLACK', 'MALE'): 9, ('18', '24', 'MALE', 'Y'): 9, ('18', '24', 'MALE', 'WHITE'): 9, ('18', 'BLACK', 'WHITE', 'Y'): None, ('18', '24', 'BLACK', 'Y'): None,
('18', '24', 'BLACK', 'WHITE'): None, ('18', '24', 'WHITE', 'Y'): None, ('24', 'BLACK', 'MALE', 'Y'): 9, ('24', 'BLACK', 'MALE', 'WHITE'): 9, ('24', 'MALE',
'WHITE', 'Y'): 9, ('24', 'BLACK', 'WHITE', 'Y'): None, ('Has_Criminal_record', 'MALE', 'high', 'works'): 6, ('Has_Criminal_record', 'high', 'home_schooled',
'works'): 2, ('Has_Criminal_record', 'high', 'high_school', 'home_schooled'): 0, ('Has_Criminal_record', 'high_school', 'home_schooled', 'works'): 0,
('Has_Criminal_record', 'high', 'high_school', 'works'): 2, ('high', 'high_school', 'home_schooled', 'works'): 0, ('MALE', 'WHITE', 'Y', 'high', 'works'): 8,
('MALE', 'WHITE', 'Y', 'high_school', 'works'): 4, ('MALE', 'WHITE', 'Y', 'high', 'high_school'): 4, ('MALE', 'Y', 'high', 'high_school', 'works'): 4, ('MALE',
'WHITE', 'high', 'high_school', 'works'): 4, ('18', 'BLACK', 'MALE', 'WHITE', 'Y'): 9, ('18', '24', 'BLACK', 'MALE', 'Y'): 9, ('18', '24', 'BLACK', 'MALE',
'WHITE'): 9, ('18', '24', 'MALE', 'WHITE', 'Y'): 9, ('18', '24', 'BLACK', 'WHITE', 'Y'): None, ('24', 'BLACK', 'MALE', 'WHITE', 'Y'): 9,
('Has_Criminal_record', 'high', 'high_school', 'home_schooled', 'works'): 0, ('MALE', 'WHITE', 'Y', 'high', 'high_school', 'works'): 4, ('18', '24', 'BLACK',
'MALE', 'WHITE', 'Y'): 9}
My program took 0.03079456090927124 mins to run
```