Tarea 2 - Market Basket Analysis

Profesor: Marcelo Mendoza

Felipe Avaria 2923547-3
felipe.avaria@alumnos.usm.cl

Rolando Casanueva 201204505-3
rolando.casanueva.12@sansano.usm.cl

1. Introducción

En esta tarea se trabajara sobre el dataset "groceries.csv", con el software "Weka 3.8.1z python 2.7, se buscara obtener reglas de asociación y se probaran los algoritmos .ªprioriz "FG-growth". Toda la documentación, archivos y programas pueden ser encontrados en el repositorio de nuestro grupo: https://github.com/roloow/Patrones-en-Mineria-de-Datos

2. Desarrollo

2.1. Transformación dataset

Para el uso del dataste este tendrá que ser tratado, debido a que su estructura como lista de listas no es útil para su uso en Weka, por lo que se implemento el siguiente programa "**CSV2ARFF.py**"(**Ver Anexo 3.1** o mayor detalle en repositorio) que transforma los datos a una matriz de dimensiones cantidad de productos por cantidad de compras realizadas y también modifica su extención a "**.arff**", quedando con la estructura adecuada para su uso. Si bien cada función se encuentra propiamente comentada y explicada el algoritmo de conversión se basa en un recorrido total por el archivo para ver todos los distintos tipos de productos que se han comprado.

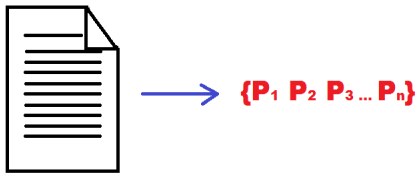

Imagen 1: representación de lectura para obtener conjunto productos

Y luego una segunda lectura del archivo para la creación de la matriz donde cada elemento que haya sido comprado en cada linea se representa por un 1 en la matriz mientras que los otros se representan por un signo de interrogación para que Weka no lo califique. Para mayor entendimiento se recomienda la lectura de la documentación. (http://www.cs.waikato.ac.nz/ml/weka/documentation.html)

$$M_{xy} = \begin{cases} 1 & \text{if } V_y \in SET_x \\ ? & \text{if } V_y \notin SET_x \end{cases}$$

Cabe mencionar que M es la matriz y V es el vector correspondiente de todos los productos mientras que SET es el conjunto de productos comprados en la fila correspondiente.

2.2. Utilización algoritmo Apriori

2.2.1. Parte 1: soporte mínimo 0.001

En esta sección se aplicara el algoritmo Apriori usando Weka, con un soporte mínimo fijo igual a 0.001, y parámetros de confianza 0.85, 0.86, 0.87, 0.88, 0.89, 0.9, 0.91, 0.92, 0.93, 0.94 y 0.95, donde se contabilizaran el numero de reglas encontradas, se mostraran los resultados a través de un gráfico confianza vs número de reglas y finalmente se tomaran las 5 reglas.

Resultados (soporte mínimo 0.001)

Los datos que obtuvimos fueron los que pueden ser apreciados en la **tabla 1** y con ellos se genera el **gráfico 1**.

Metrics	0.85	0.86	0.87	0.88	0.89	0.90	0.91	0.92	0.93	0.94	0.95
Rules	199	161	150	143	131	130	77	58	34	28	28

Tabla 1: Cantidad de reglas obtenidas en el algoritmo dependiendo de la confianza

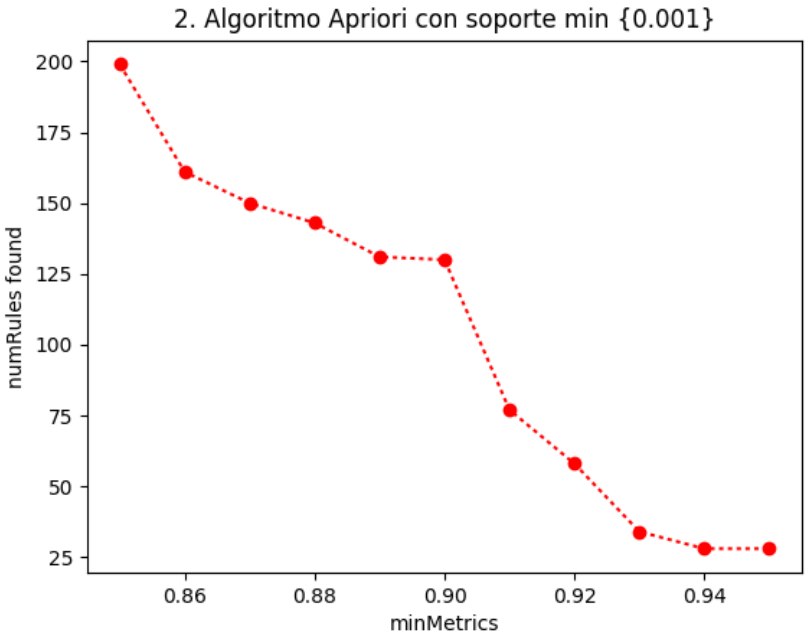

Gráfico 1: Confianza vs Número de reglas

Reglas inesperadas

1. `whipped/sour cream, root vegetables, flour ==> whole milk`

2. `tropical fruit, root vegetables, yogurt, other vegetables, oil ==> whole milk`

3. `whipped/sour cream, root vegetables, pip fruit, other vegetables ==> whole milk`

4. `root vegetables, cream cheese, butter ==> yogurt`

5. `fruit/vegetable juice, soda, citrus fruit, other vegetables ==> root vegetables`

La primera regla que seleccionamos, es la que presento la mayor cantidad de ocurrencias junto con una confianza igual a 1, donde en todos lo casos se compro whole milk si llevaba whipped/sour cream, root vegetables y flour.

La segunda regla es la regla con confianza 1 que tiene un menor soporte en este caso es 0.001, es decir tan solo 10 ocurrencias, también se selecciono por el largo de la regla y por que como en mucho de los casos el consecuente es whole milk.

La tercera regla es la que presento la mayor cantidad de ocurrencias con la confianza mas baja, y como en los otros casos, el consecuente volvio a ser whole milk.

La cuarta regla la seleccionamos por no tener como consecuente whole milk que se presenta en 127 consecuentes y tampoco presenta other vegetables que tiene 61 ocurrencias en el consecuente, si no que tiene yogurt y un valor de confianza relativamente alto pero un muy bajo soporte justo sobre el minimo con 10 ocurrencias.

La quinta regla es la que al igual que en el caso anterior no tiene como consecuente whole milk, ni other vegetables y tampoco yogurt, si no que tiene root vegetables que entre todas las reglas solo aparece 4 veces y esta es la que

tiene mayor confianza.

2.2.2. Parte 2: soporte mínimo 0.0013

En esta sección al igual que en la anterior, se aplicara el algoritmo Apriori usando Weka, con un soporte mínimo fijo, pero en este caso igual a 0.0013, y parámetros de confianza 0.85, 0.86, 0.87, 0.88, 0.89, 0.9, 0.91, 0.92, 0.93, 0.94 y 0.95, donde se contabilizaran el numero de reglas encontradas, se mostraran los resultados a través de un gráfico confianza vs número de reglas y finalmente se tomaran las 5 reglas.

Resultados (soporte mínimo 0.0013)

Los datos que obtuvimos fueron los que pueden ser apreciados en la **tabla 2** y con ellos se genera el **gráfico 2**.

Metrics	0.85	0.86	0.87	0.88	0.89	0.90	0.91	0.92	0.93	0.94	0.95
Rules	56	49	38	31	19	18	14	14	8	2	2

Tabla 2: Cantidad de reglas obtenidas en el algoritmo dependiendo de la confianza

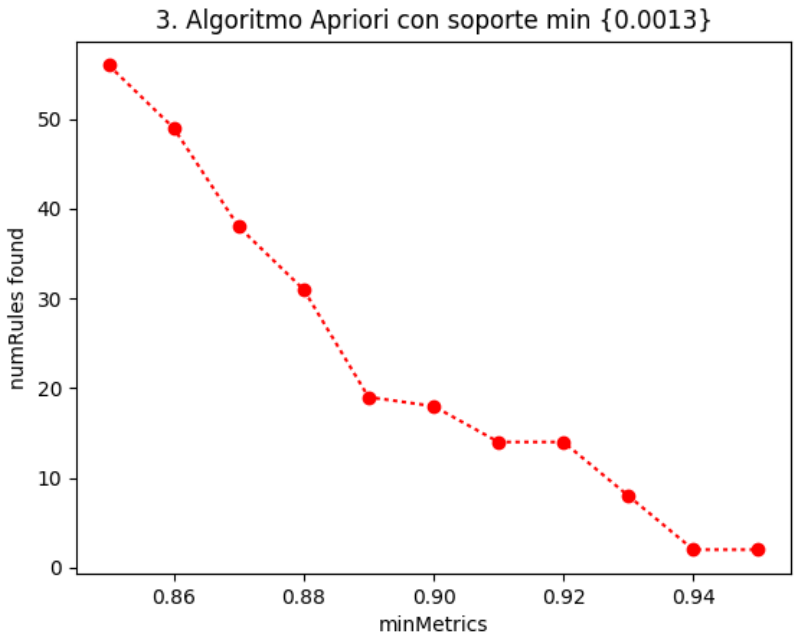

Gráfico 2: Confianza vs Número de reglas

Reglas inesperadas

```
1. whipped/sour cream, root vegetables, flour, ==> whole milk

2. whipped/sour cream, root vegetables, pip fruit, other vegetables ==> whole milk

3. tropical fruit, root vegetables, whole milk, citrus fruit ==> other vegetables

4.chicken, sausage, domestic eggs ==> whole milk

5. red/blush wine, liquor ==> bottled beer
```

La primera regla se repite al caso en que el soporte minimo era 0.001, por ser la mejor en cuanto a tener confianza 1 y alto soporte en este caso es aproximadamente 0.0017, es por eso que se repite en ambas preguntas, y como se comento anteriormente la seleccionamos para mostrar que la mayoria de las reglas tienen como consecuente la leche.

La segunda la seleccionamos por se la ultima regla que presento el algoritmo, por ser la con la confianza mas baja y entre las de baja confianza es la que tiene el menor soporte, que a pesar de todo no es tan bajo soporte, ya que presente 20 ocurrencias.

La tercera regla es seleccionada por ser la regla que presentaba mas ocurrencias y que no tenia como consecuente whole milk, si no que other vegetables, que seria el segundo consecuente mas repetido en las reglas presentadas, son 31 ocurrencias y tiene 0.86 de confianza.

La cuarta regla es una regla que nos llamo la atención por ser alimentos solo proteicos lo que podría representar a algún tipo de cliente que solo consume este tipo de alimentos, no tiene un soporte destacable esta justo sobre el mínimo permitido en esta sección.

La ultima regla la seleccionamos por tener como consecuente bottled beer y no tener other vegetables ni whole milk, que son los productos mas repetidos en los consecuentes por se muy usualmente comprados en prácticamente todas las ocasiones, y es la única regla que contiene bottled beer en el consecuente.

2.3. Utilización algoritmo FP-growth

Para el algoritmo FP-growth, se mantendrá el soporte mínimo en 0.001 y buscaremos reglas con confianza mínima en el rango 0.85, 0.86, 0.87, 0.88, 0.89, 0.9, 0.91, 0.92, 0.93, 0.94 y 0.95. Para cada valor de confianza, contando el número de reglas encontradas. Se repetirá el experimento con FP-growth subiendo el soporte mínimo en 0.0001 cada vez, hasta llegar a 0.0018. Para cada valor de confianza, contabilice el número de reglas encontradas. Luego, graficamos el número de reglas (eje y) en función de la confianza (eje x). Comentaremos sobre el comportamiento de las curvas. Finalmente, seleccionamos 5 reglas inesperadas.

Resultados

Dada la alta cantidad de datos obtenidos en esta sección, se decidió generar un gráfico conjunto de todos los resultados y mostrar la tabla de todos los resultados. A su vez, todos los gráficos particulares podrán ser encontrados en la sección de **Anexos 3.2**. Los datos y sus respectivas funciones para ser "ploteadas.están en el archivo **plot.py** dentro del repositorio por si se desea una mayor indagación.

Metrics	0.85	0.86	0.87	0.88	0.89	0.90	0.91	0.92	0.93	0.94	0.95
Rules (0.0018)	6	5	5	5	4	2	0	0	0	0	0
Rules (0.0017)	8	7	7	7	6	3	1	1	1	1	1
Rules (0.0016)	12	11	11	11	6	3	1	1	1	1	1
Rules (0.0015)	21	20	20	13	7	3	3	3	3	1	1
Rules (0.0014)	33	32	32	25	13	10	8	8	8	2	2
Rules (0.0013)	50	49	38	31	19	16	14	14	8	2	2
Rules (0.0012)	102	70	59	52	40	37	35	35	11	5	5
Rules (0.0011)	128	96	85	78	66	63	61	42	18	12	12
Rules (0.0010)	193	161	150	143	131	128	77	58	34	28	28

Tabla 3: Cantidad de reglas obtenidas en el algoritmo en base a un soporte estático y dependiendo de la confianza

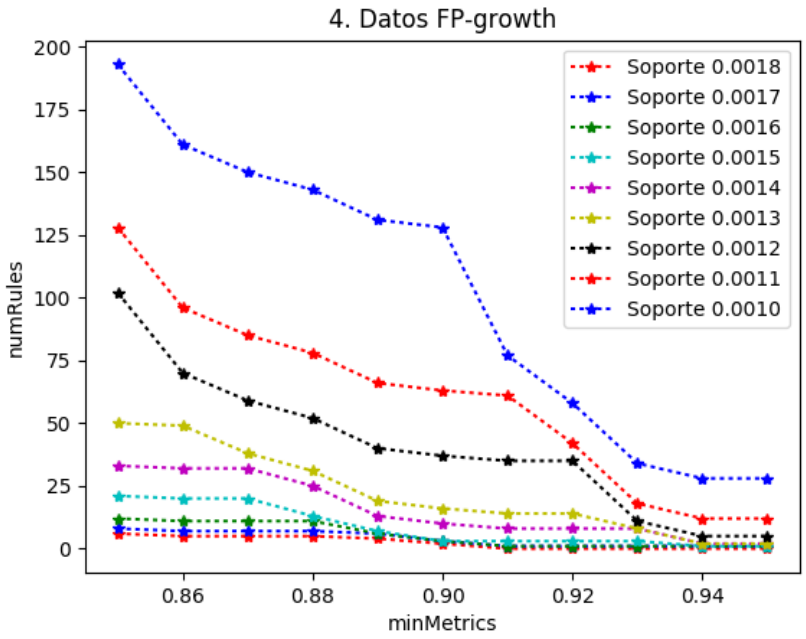

Gráfico 3: Confianzas vs Número de reglas

Reglas seleccionadas

```
1. root vegetables, whipped/sour cream, flour ==> whole milk

2. whole milk, root vegetables, tropical fruit, citrus fruit ==> other vegetables

3. tropical fruit, fruit/vegetable juice, whipped/sour cream ==> other vegetables

4. other vegetables, yogurt, root vegetables, tropical fruit, oil ==> whole milk

5. whole milk, tropical fruit, butter, curd ==> yogurt
```

La primera regla fue seleccionada por ser la regla con confianza 1 de mayor soporte,es decir en todos los casos en que se compro root vegetables, whipped/sour cream y flour se compro también whole milk y que esto ocurrió con un soporte del 0.0017 lo que vienen a ser 17 transacciones, esta es una regla importante y pude ser útil para desarrollar algún plan de marketing o para ordenar los productos de alguna forma que incentive a comprar ambos productos.

La segunda regla se selecciono por ser la de mayor soporte que seria cercano al 0.0031, es decir 31 ocurrencias y tiene una confianza relativamente baja, pero en ninguna caso es tan baja, esta regla puede ser provechosa para el estudio, ya que quizás es la relación que podría dejar mas ganancias dado el soporte.

La tercera regla se seleccionaron por ser las reglas que al fijar el soporte en el máximo estudiado en este caso 0.0018 fueron las dos reglas que presentaron mayor confianza en esta caso de 0.9 pero seleccionamos la mas corta y con un mayor lift, y en este casos hay 19 ocurrencias de la regla, esta quizá es una de las reglas que se le puede sacar mayor provecho porque uno se puede preguntar por que hay un % 10 de casos en que no se esta llevando este ultimo producto de la relación, quizás si estos productos son ubicados mas cerca en las vitrinas logren mayores ventas o si se agrega una promoción.

La cuarta regla es la que llama la atención por su alta confianza, pero que dado su soporte 0.001 es decir solo 10 ocurrencias no es realmente una regla muy importante aun que se puede decir que en todos los casos en que se compra other vegetables, yogurt, root vegetables, tropical fruit y oil se compra whole milk, pero como es un bajo numero de veces que ocurre esto no es algo que represente en gran media a los datos y quizás no es útil para establecer una oferta para incentivar cierto producto, en general para bajo soporte es mas fácil encontrar mas relaciones de alta confianza.

La ultima regla al igual que la anterior es de bajo soporte, pero en este caso también es de baja confianza, esta regla en particular tiene 0.0012 de soporte y 0.86 de confianza, es un regla que se podría haber generado por casualidad dado dado el numero de transacciones no es algo que se de de forma muy usual solo ocurrió 12 veces, cabe destacar que los soportes en general fueron bajos pero esta regla tiene la mitad de las ocurrencias que las reglas mas útiles.

3. Anexo

3.1. CSV2ARFF.py

Programa encargado de la transformación del dataset "groceries.csv.[a] "groceries.arff". Consta de 3 funciones bases las cuales están descritas a continuación con un formato de parametros de entrada, salida, código y explicación de la función. El programa en sí puede encontrarse en el repositorio con el nombre del enunciado de esta sección.

- Products

```
#-------------------------------------------------
#                products
#-------------------------------------------------
#    FUNCTION_IN_PARAMETERS_DEFINITION
#    FILE:   Opened Data Type File
#
#    FUNCTION_OUT_PARAMETERS_DEFINITION
#    out:      list of strings refering to the no
#              repeated names in the DataSet
#    FUNCTION_CODING
def products(FILE):
        my_set = set([])
        for line in FILE:
                line = line.strip().split(',')
                line_set = set(line)
                my_set |= line_set
        return list(my_set)
#    FUNCTION_EXPLANATION
#    As a brute force algorithm, it only reads
#        line by line adding the parameters to a set
#        with a union operation
#-------------------------------------------------
```

- Matrix

```
#-------------------------------------------------
#                 matrix
#-------------------------------------------------
#    FUNCTION_IN_PARAMETERS_DEFINITION
#    FILE:   Opened Data Type File
#
#    FUNCTION_OUT_PARAMETERS_DEFINITION
#    out:      tuple with a list of list and a list
#              of strings
#    FUNCTION_CODING
def matrix(FILE):
        matrix = []
        MSET = products(FILE)
        FILE.seek(0)
        for line in FILE:
                line = line.strip().split(',')
                vector = []
                for product in MSET:
                        if product in line:
                                vector.append('1')
                        else:
                                vector.append('?')
                matrix.append(vector)
        return (matrix, MSET)
#    FUNCTION_EXPLANATION
#        Firstly it takes the file and gets the
#        list of diferent products, then it creates
#        a matrix where 1 represents the product is
#        on current order, and 0 doesnt.
#-------------------------------------------------
```

- create_arff

```
#-------------------------------------------------------
#                   create_arff
#-------------------------------------------------------
#    FUNCTION_IN_PARAMETERS_DEFINITION
#    FILE:   Opened Data Type File
#    OUTNAME:   String value
#
#    FUNCTION_OUT_PARAMETERS_DEFINITION
#    out:     No return value
#    FUNCTION_CODING
def create_arff(FILE, OUTNAME):
        FILENAME = OUTNAME.lower() + '.arff'
        MATRIX, MSET = matrix(FILE)
        f = open(FILENAME, 'w')
        f.write('@RELATION ' + OUTNAME.lower() + '\n\n')
        for product in MSET:
                f.write("@ATTRIBUTE '" + product + "' {1}\n")
        f.write('\n@DATA\n')
        for vector in MATRIX:
                f.write( ','.join(vector) + '\n')
        f.close()
#    FUNCTION_EXPLANATION
#        Creates an ARFF file with the attributes
#        that were obtained in previews functionalities
#-------------------------------------------------------
```

3.2. Gráficos FP-Growth

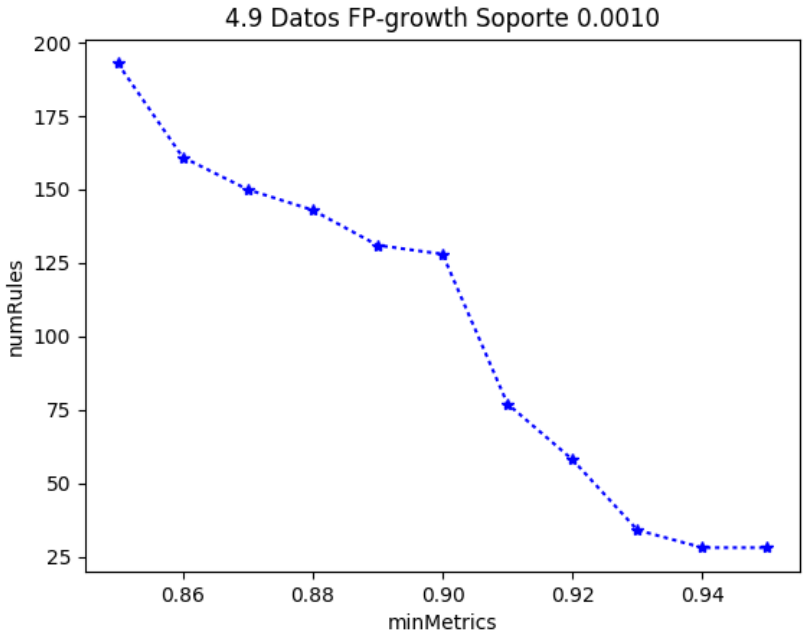

Gráfico 4: Confianza vs Número de reglas

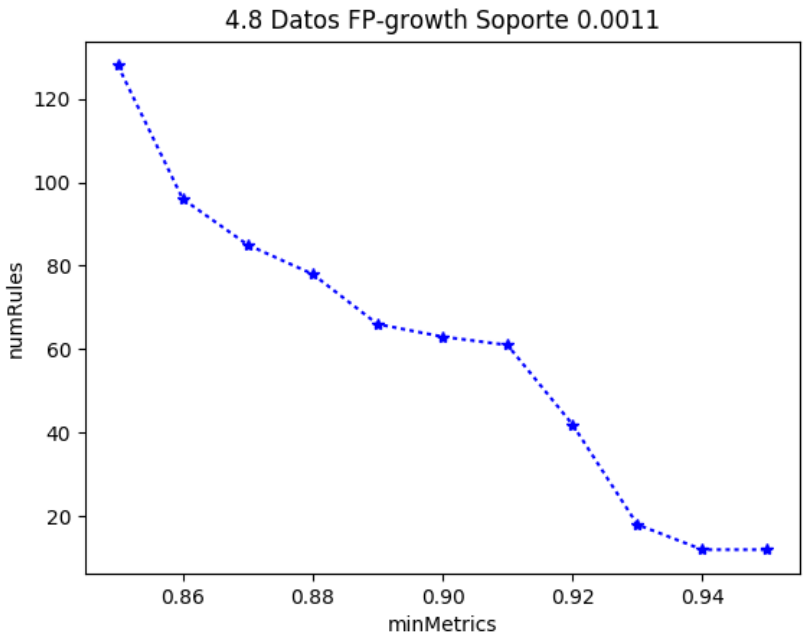

Gráfico 5: Confianza vs Número de reglas

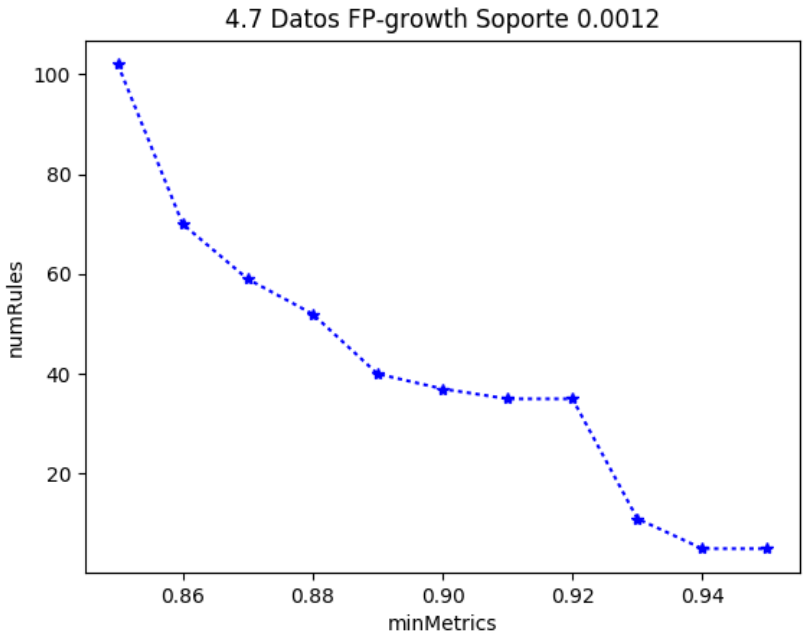

Gráfico 6: Confianza vs Número de reglas

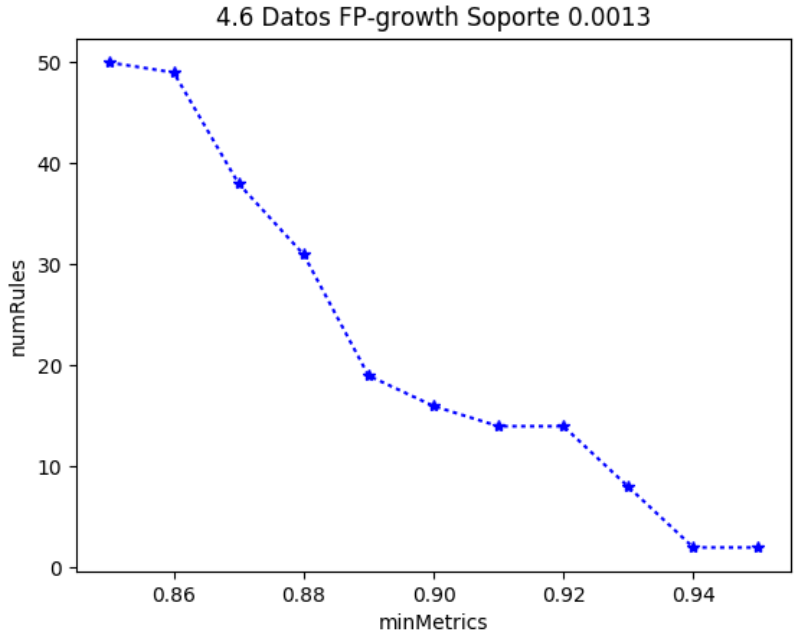

Gráfico 7: Confianza vs Número de reglas

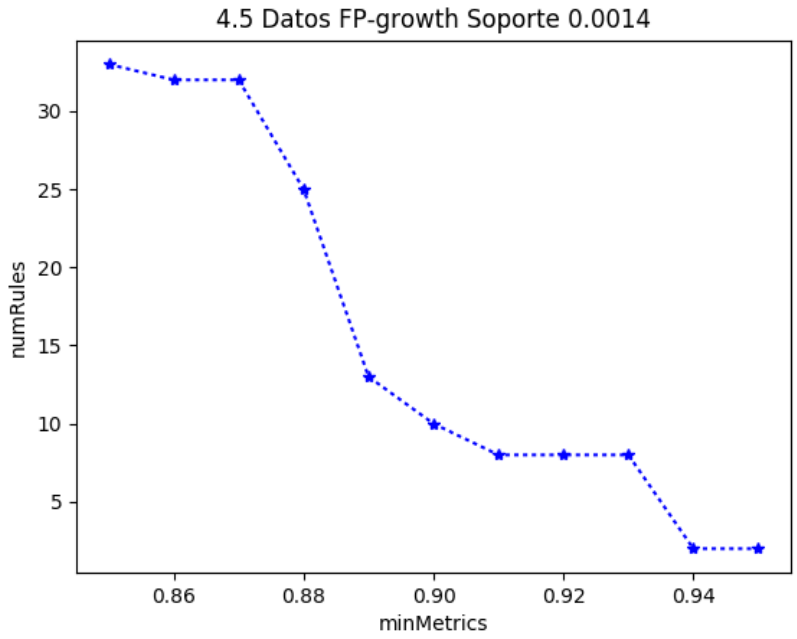

Gráfico 8: Confianza vs Número de reglas

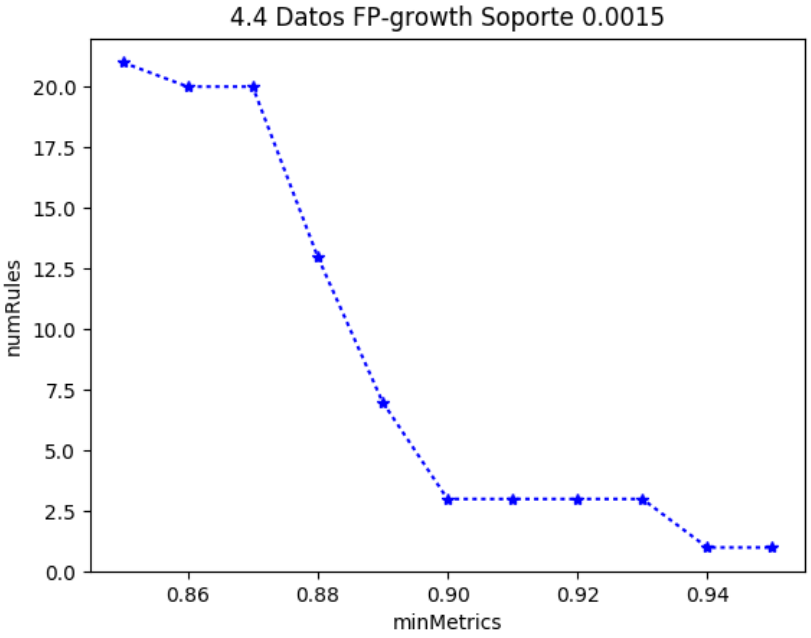

Gráfico 9: Confianza vs Número de reglas

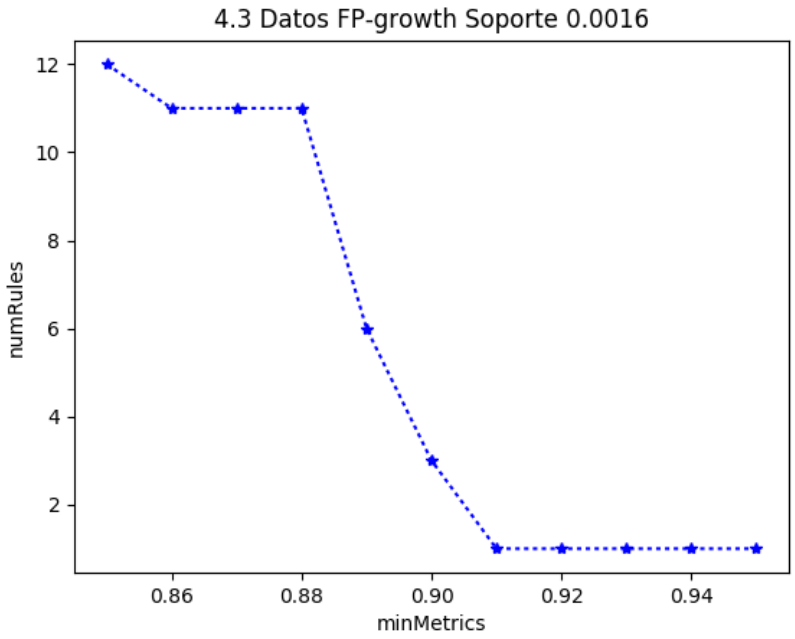

Gráfico 10: Confianza vs Número de reglas

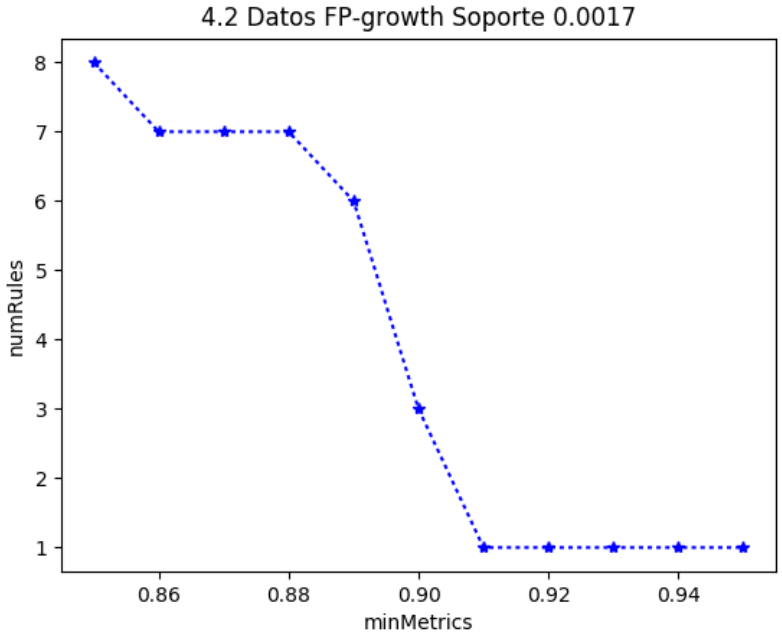

Gráfico 11: Confianza vs Número de reglas

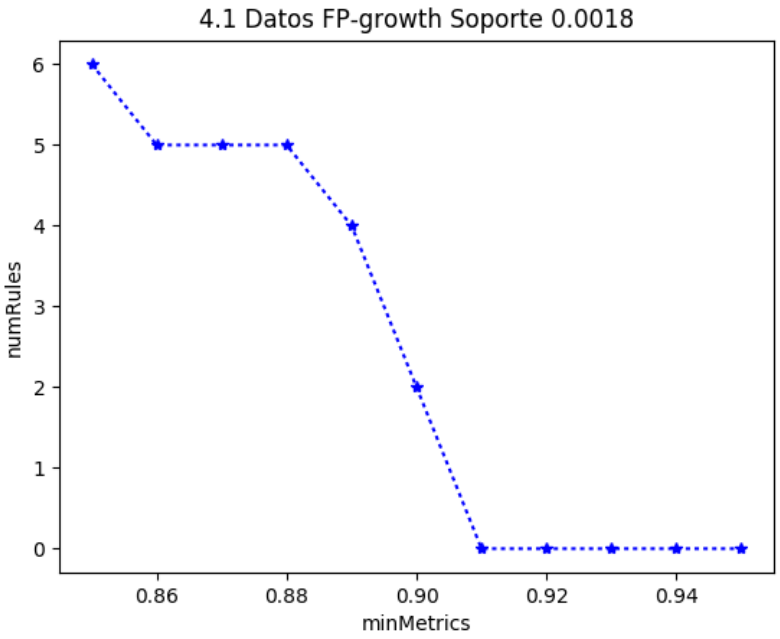

Gráfico 12: Confianza vs Número de reglas