INF391 Reconocimiento de Patrones en Minería de Datos
Tarea 2

Universidad Técnica Federico Santa María, Campus San Joaquín
Departamento de Informática

8 de septiembre de 2015

Profesor Marcelo Mendoza

Juan Pablo Escalona
juan.escalonag@alumnos.usm.cl
201073515-k

Rafik Masad
rafik.masad@alumnos.usm.cl
201073519-2

Gianfranco Valentino
gianfranco.valentino@alumnos.usm.cl
2860574-9

1. Introducción

En el presente informe, se pretende analizar dos algoritmos que permiten encontrar reglas de asociación sobre un historial de transacciones. Estos algoritmos, Apriori y FP-Growth, se basan en parámetros como Confianza(Conf) y Soporte Mínimo(MinSup). Otros parámetros, permiten limitar el número máximo de reglas encontradas, para terminar la ejecución de forma temprana.

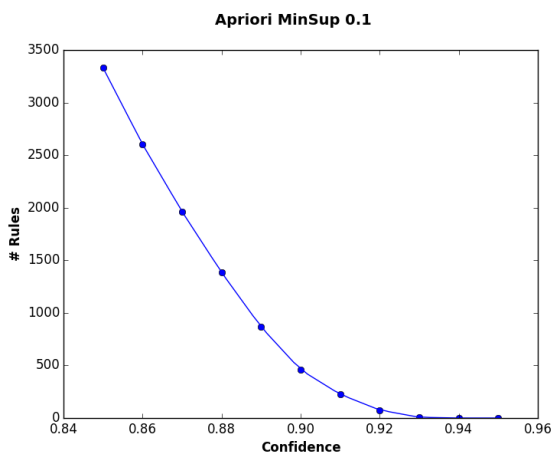
Se buscarán reglas interesantes, y ademas se pretende comparar tiempos de ejecución, para observar la eficiencia de ambos algoritmos, frente a distintas condiciones.

Para el desarrollo de la experiencia se escribió un script en python capaz de ejecutar los diferentes algoritmos con los parámetros a probar utilizando WEKA. Este script genera gráficos interesantes de salida, los cuales se analizan en el informe. Todos los scripts creados para esta experiencia se encuentran en 6.

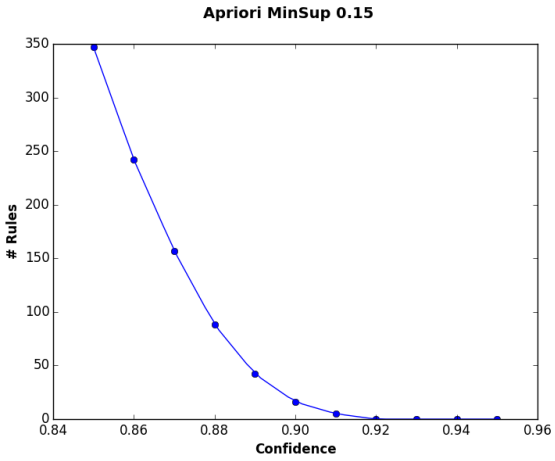
2. Algoritmo Apriori

En este experimento se gráfica la confianza entre 0.85 y 0.95 utilizando un soporte mínimo de 0.1 y 0.15. En el eje Y se presenta el número de reglas encontradas para cada una de las confianzas.

2.1. Gráficos de confianza y soporte mínimo



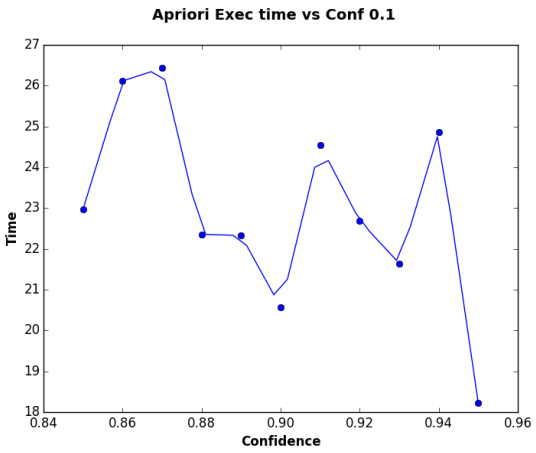
(a) Algoritmo Apriori: Confianza vs Número de reglas encontradas con soporte mínimo 0.1



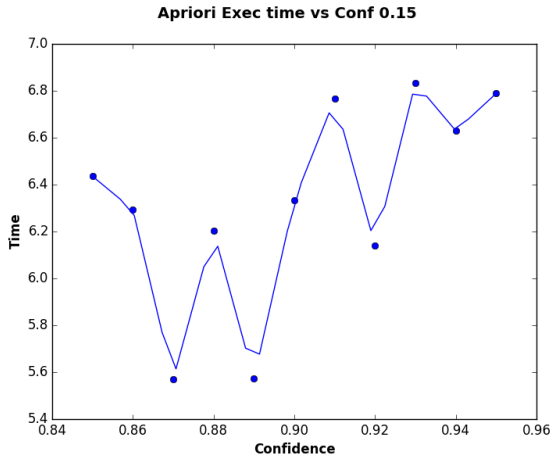
(b) Algoritmo Apriori: Confianza vs Número de reglas encontradas con soporte mínimo 0.15

Es importante destacar la diferencia entre los ejes y de ambos gráficos, hay una diferencia en magnitud de orden 10.

2.2. Gráficos de tiempo de ejecución



(a) Algoritmo Apriori: Tiempo de ejecución para confianzas entre 0.85 y 0.95 con soporte mínimo de 0.1

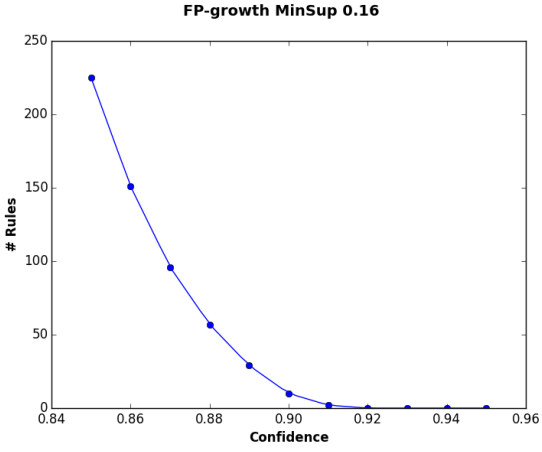


(b) Algoritmo Apriori: Tiempo de ejecución para confianzas entre 0.85 y 0.95 con soporte mínimo de 0.15

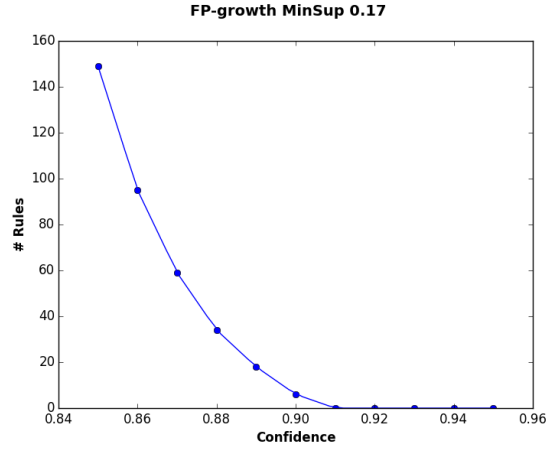
3. Algoritmo FP-Growth

Para este experimento se hace variar la confianza mínima entre 0.85 a 0.95 para cada uno de los soportes mínimos entre 0.16 y 0.25. Se generan entonces 10 gráficos con diferentes soportes mínimos, en el eje X se presenta la confianza entre 0.85 y 0.95. El eje Y representa el número de reglas encontradas.

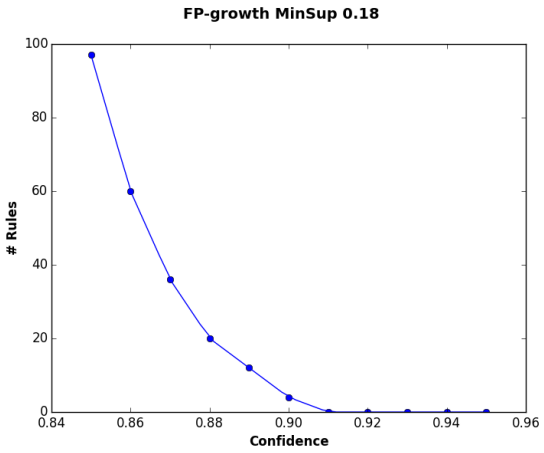
3.1. Gráficos de confianza y soporte mínimo



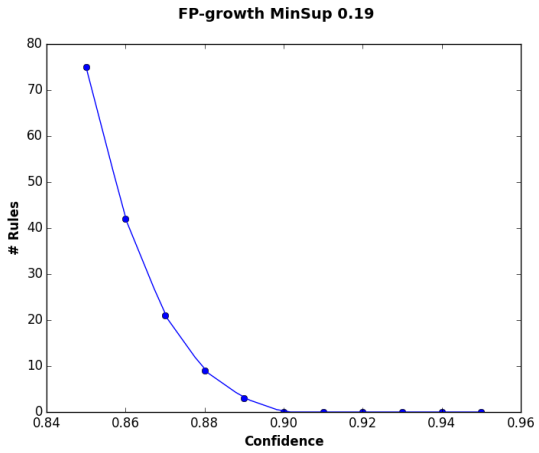
(a) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.16



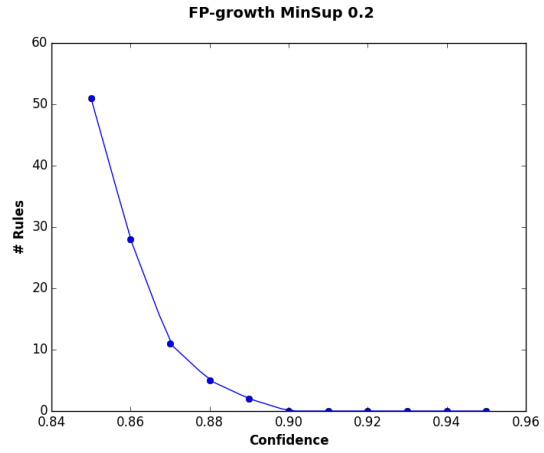
(b) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.17



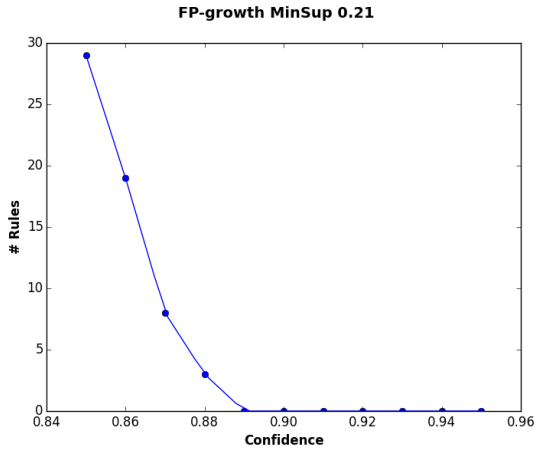
(a) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.18



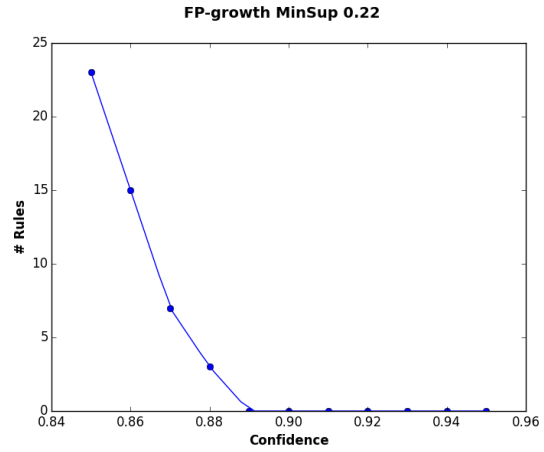
(b) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.19



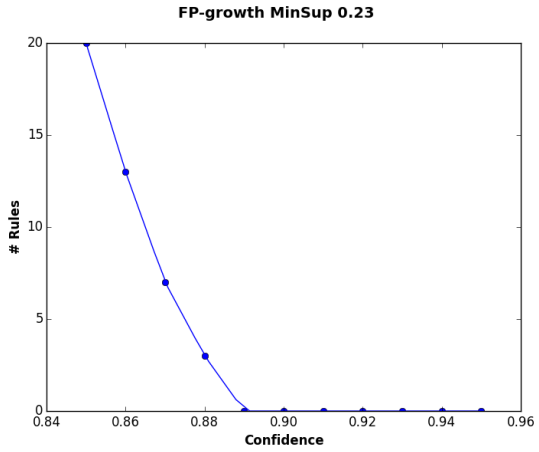
(a) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.20



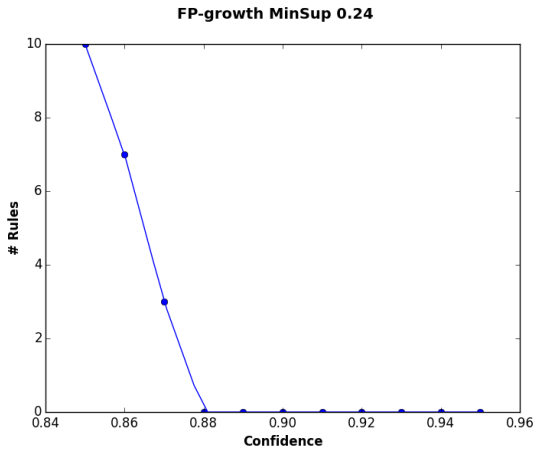
(b) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.21



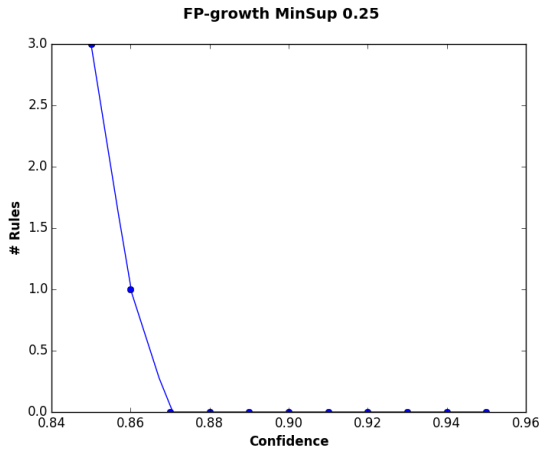
(a) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.22



(b) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.23

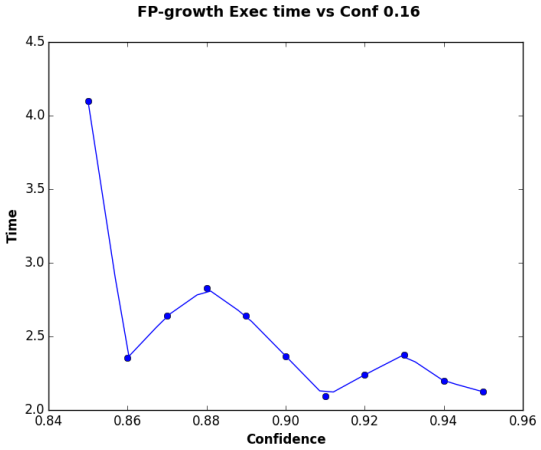


(a) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.24

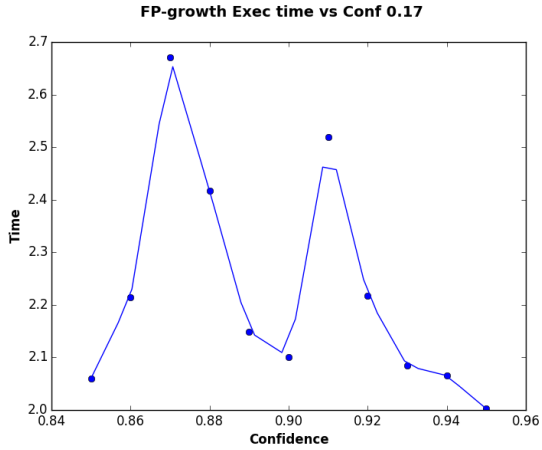


(b) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.25

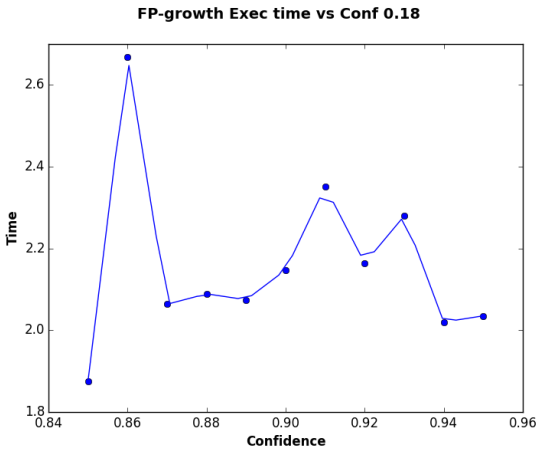
3.2. Gráficos de tiempo de ejecución



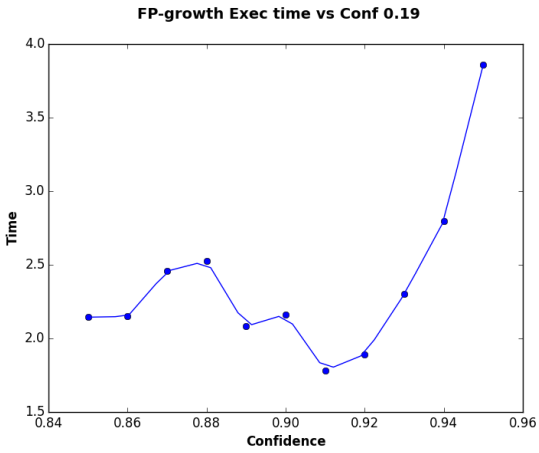
(a) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.16



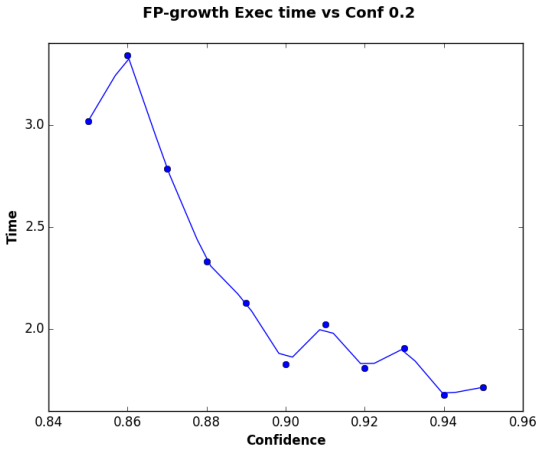
(b) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.17



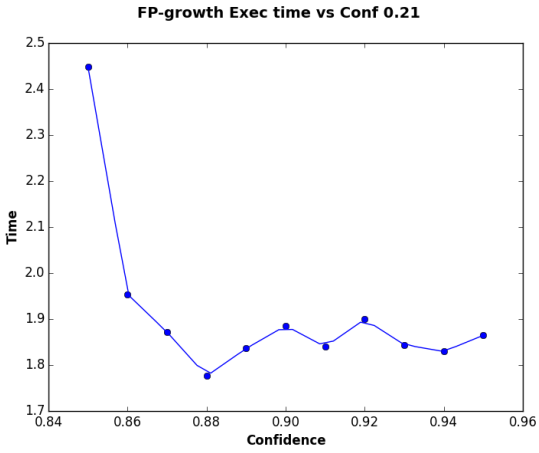
(a) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.18



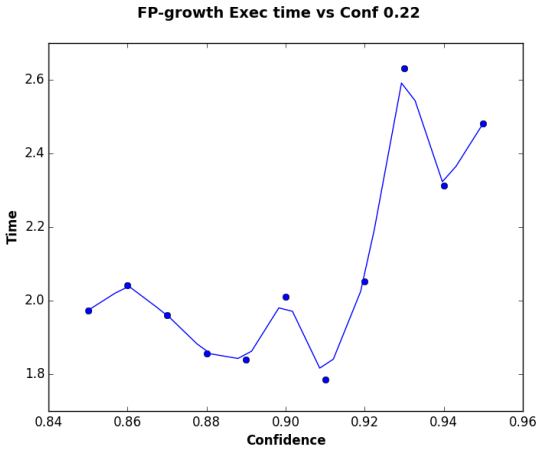
(b) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.19



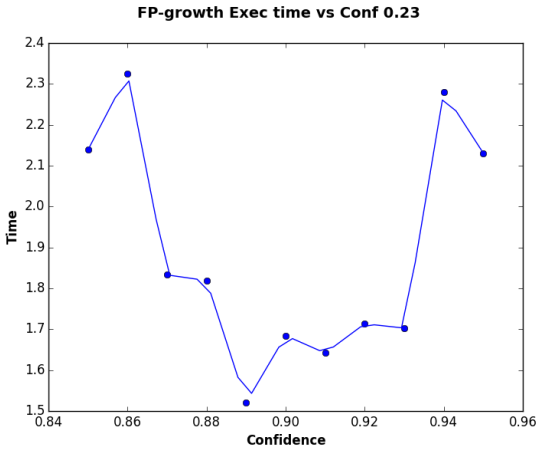
(a) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.20



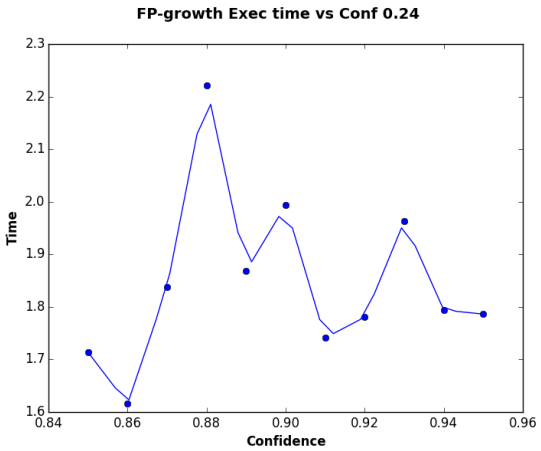
(b) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.21



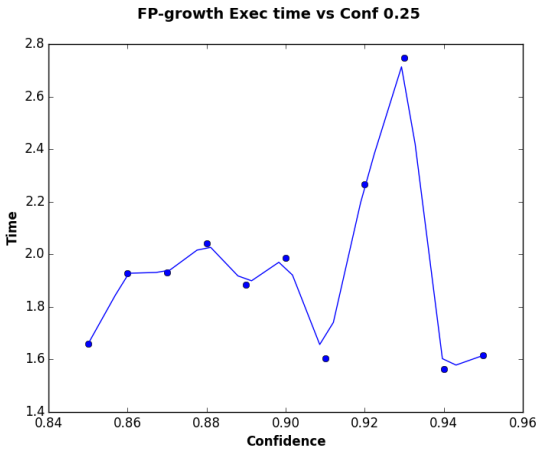
(a) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.22



(b) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.23



(a) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.24



(b) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.25

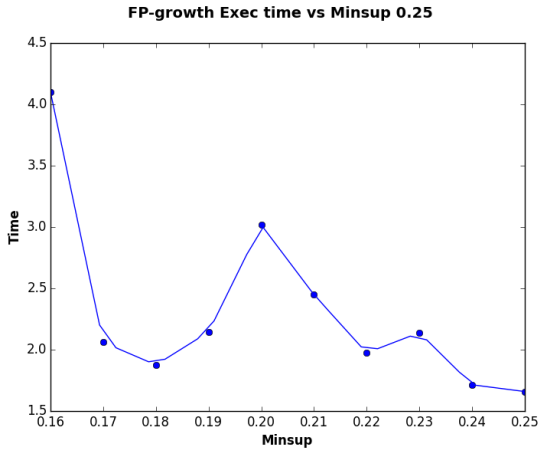


Figura 13: Algoritmo FP-growth: tiempo de ejecución, Confianza vs Soportes mínimos

4. Reglas interesantes

El monopolio del pan y pasteles

En un alto porcentaje de las reglas encontradas, sobre todo las de más alto porcentaje de confianza, encontramos que una combinación de compras implica la compra de pan o pasteles. Lo anterior nos da a inferir que estos productos son altamente consumidos y un porcentaje alto de los consumidores al comprar algún producto llevan además pan o pasteles ya que es algo que es necesario constantemente en el hogar.

¿Vegetales?

En una situación similar pero en menor medida a la de los panes y pasteles existen múltiples relaciones entre distintas combinaciones de compras y vegetales. En particular se repite bastante combinaciones que incluyen instrumentos de cocina. Al igual que los productos anteriores nos da a inferir que estos productos son altamente consumidos y un porcentaje alto de los consumidores al comprar algún producto llevan además vegetales.

Desayuno de campeones

Con una confianza del 87 % podemos determinar que la gente que compra comida para el desayuno, comida congelada y margarina o pañuelos de papel, compran galletas. Lo anterior lo podemos asociar a un comportamiento cultural propio del país de origen de los datos, Estados Unidos, donde la gente que va a comprar comida para el desayuno y congelados, compran galletas.

Comida de solteros

Con una confianza del 88 % los que compran galletas y comida preparada también compran comida congelada. Lo anterior lo podemos asociar a que los que compran comida preparada compran por la misma razón comida congelada.

Asado... y galletas

Con una confianza del 87 % los que compran galletas, carne y herramientas para cocinar o vegetales, compran, además, comida congelada. Lo anterior lo podemos asociar a que cuando compran carne y implementos para el asado generalmente lo acompañan con alguna comida congelada fácil de cocinar.

5. Conclusiones

Podemos observar que a medida que aumenta el MinSup la cantidad de reglas encontradas disminuye, lo que es esperable, dado que se le exige una mayor frecuencia a dicha regla. En cuanto, a la confianza, se comporta de forma análoga, pero limitando en cuanto a dado que ocurre la condición, ocurre la consecuencia con cierta frecuencia.

Si consideramos la calidad de las reglas encontradas, podemos darnos cuenta, que ambos algoritmos encontraron reglas muy similares, aunque FP-Growth, al remover vegetables y bread and cake, no fue capaz de encontrar una regla en ningún caso. Sin embargo, si se considera todo el dataset, se entiende que se buscan relaciones en el mismo conjunto de datos, y se esperaría coincidencias en cierta cantidad de reglas.

Sobre el tiempo de ejecución, podemos darnos cuenta que FP-Growth en el peor de los casos, alcanza un tiempo cercano al 66 % del tiempo de Apriori, por lo que consideramos que es mas eficiente, aunque encuentra un número de reglas bastante menor, alrededor del 66 % de Apriori. Nuevamente caemos en la discusión de eficacia o eficiencia, lo cual depende netamente de los recursos disponibles, las necesidades y quizás, la frecuencia en que se utiliza el algoritmo sobre el dataset.

6. Anexo

Script 1: Generador de gráficos para todos los parámetros

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-

import matplotlib.pyplot as plt
import subprocess
from scipy.interpolate import interp1d
import numpy as np
import re
import os.path
from time import time



def plotter(x, y, filename = 'foo.png', interp_points = 30, title = 'Alg wtih Min Sup = minsup', x_title = 'Confidence', y_title = '# Rules'):
    x_interp   = np.linspace(min(x), max(x), interp_points)
    y_interp = np.interp(x_interp, x, y)
```

```
17
18      fig, ax1 = plt.subplots()
19      ax1.set_xlabel(x_title, fontweight = 'bold')
20      ax1.set_ylabel(y_title, fontweight = 'bold')
21      fig.suptitle(title, fontsize = 14, fontweight = 'bold')
22      # ax1.set_ylim([0, 250])
23      ax1.plot(x, y, 'bo', x_interp, y_interp, 'b-')
24
25      plt.savefig(filename)
26
27
28  INPUT = "supermarket.arff"
29  RUN = "./run.sh"
30  FILTER = "./filter.sh"
31  MATCHER = re.compile(r'.*?([\d]+)\. .*')
32  filtered = "supermarket-filtered.arff"
33  filter_indexes = map(lambda x: str(x) ,
34                           [
35                               # 13, # bread and cake
36                               # 86  # vegetables
37                           ])
38  if filter_indexes:
39      cmd = [FILTER, INPUT, filtered,  ",".join(filter_indexes)]
40      subprocess.call(cmd)
41  else:
42      filtered = INPUT
43  c_float = map(lambda x: x/100.0, range(85, 96))
44  c = map(lambda x: str(x/100.0), range(85, 96))
45  alg_minsup = {
46      "Apriori": ['0.1', '0.15'],
47      "FP-growth": map(lambda x: str(x/100.0), range(16, 26))
48  }
49
50  alg_type = {"Apriori":"1", "FP-growth":"2"}
51  alg_time = {}
52
53  for alg, minsups in alg_minsup.items():
54      alg_time[alg] = {}
55      for minsup in minsups:
56          filename = None
57          ys = []
58          alg_time[alg][minsup] = {'x': [], 'y': []}
59          for conf in c:
60              filename = 'outs/'+'-'.join([alg,minsup,conf])+"-out.txt"
61
62              if not os.path.isfile(filename):
63                  cmd = [RUN, filtered, filename, conf, minsup, alg_type[alg]];
64                  t1 = time()
65                  subprocess.call(cmd)
66                  t2 = time()
67                  alg_time[alg][minsup]['x'].append(float(conf))
68                  alg_time[alg][minsup]['y'].append(t2 - t1)
69
70              weka_file = open(filename)
71              last = 0
72              for line in weka_file:
73                  mt = MATCHER.match(line)
74
75                  if mt:
```

```
                    last = mt.groups()[-1]
            weka_file.close()
            ys.append(last)
        graph_name =  alg+"-"+minsup
        plotter(c_float, ys, filename='img/'+graph_name+".png",  title=alg+" MinSup "+minsup)
        plotter(alg_time[alg][minsup]['x'], alg_time[alg][minsup]['y'], filename='img/'+graph_name+"-exec-conf.png", x_title="Confidence", y_title="Time", title=alg+" Exec time vs Conf "+minsup)

for alg, minsups in alg_minsup.items():
    xs = []
    ys = []
    for minsup in minsups:
        xs.append(float(minsup))
        ys.append(alg_time[alg][minsup]['y'][0])

    graph_name =  alg+"-"+minsup
    plotter(xs, ys, filename='img/'+graph_name+"-exec-minsup.png", x_title="Minsup", y_title="Time", title=alg+" Exec time vs Minsup "+minsup)
```

Script 2: Ejecuta Weka dado los parámetros indicados

```
#!/bin/bash

WEKA_PATH=.

CP="$CLASSPATH:$WEKA_PATH/weka.jar"
if [[ "$5" -eq 1  ]]; then
    java -cp $CP weka.associations.Apriori -N 5000 -T 0 -C $3 -D 0.05 -U 1.0 -M $4 -S -1.0 -c -1 -t $1 > $2
else
    java -cp $CP weka.associations.FPGrowth -P 2 -I -1 -N 5000 -T 0 -C $3 -D 0.05 -U 1.0 -M $4 -t $1 > $2
fi
```

Script 3: Filtra elementos específicos del dataset

```
#!/bin/bash

WEKA_PATH=.

CP="$CLASSPATH:$WEKA_PATH/weka.jar"

 java -cp $CP  weka.filters.unsupervised.attribute.Remove -R $3 -i $1 -o $2
```