Taller_Asociacion

May 24, 2025

1 Taller Asociación – Minería de Datos – 2025-I

Facultad de Ingeniería
Departamento de Ingeniería de Sistemas e Industrial
Universidad Nacional de Colombia – Sede Bogotá

Estudiantes:

- **Francisco José Salamanca Rivera**
  fsalamancar@unal.edu.co
- **Daniel Mauricio Bonilla Muñoz**
  dabonilla@unal.edu.co
- **David Camilo Gómez Medina**
  dcgomezne@unal.edu.co

Docente:

- **Elizabeth León Guzmán**
  eleonguz@unal.edu.co

1.0.1 Punto 1

Por cada un de las siguientes preguntas, proveer un ejemplo de una regla de asociación del dominio de "market basket" que satisface las siguientes condiciones. También, describir si las reglas son interesantes (subjetivamente).

Solución

Transacción	**Productos**
T1	pan, pañales, chocolate, gaseosa, leche
T2	pan, salchichas, leche, pañales, huevos
T3	leche, pañales, gaseosa
T4	pan, leche, huevos, chocolate, gaseosa, café
T5	leche, café, huevos, pan, gaseosa
T6	pan, leche, pañales, huevos, chocolate

Transacción	Productos
T7	pan, leche, pañales, chocolate

Ejemplos Reglas de asociación con soporte y confianza

- **(a) Una regla que tiene alto soporte y alta confianza:**
  {pan} → {leche} **soporte** = 6/7 = 0,85 **confianza** = 6/6 = 1
- **(c) Una regla que tiene bajo soporte y baja confianza:**
  {gaseosa, pan} → {pañales} **soporte** = 1/7 = 0,14 **confianza** = 1/3 = 0,33
- **(d) Una regla que tiene bajo soporte y alta confianza:**
  {café} → {huevos} **soporte** = 2/7 = 0,28 **confianza** = 2/2 = 1

La regla (a) {pan} → {leche} es muy interesante porque combina un alto soporte y una confianza del 100%, lo que indica que es una asociación fuerte y frecuente, útil para decisiones comerciales. La regla (c) {gaseosa, pan} → {pañales} es poco interesante, ya que tiene tanto bajo soporte como baja confianza, lo que la hace poco fiable y rara vez ocurre en los datos. En cambio, la regla (d) {café} → {huevos} es potencialmente interesante: aunque tiene bajo soporte, su confianza perfecta sugiere una relación consistente.

1.0.2 Punto 2

¿Por qué el proceso de descubrimiento de reglas de asociación es relativamente simple comparado con la generación de grandes conjuntos de ítems en bases de datos transaccionales?

Por que se hace un proceso previo el cual consiste en la identificacion de los items u objetos mas frecuentes en el dataset, que en comparacion con la generacióon del conjunto de datos, es la parte mas compleja de la tarea, ya que se necesita explicar un gran numero de combinaciones posibles entre los ítems.

1.0.3 Punto 3

Considere el siguiente conjunto de datos.

TID	Items
T01	milk, beer, diapers
T02	bread, butter, milk
T03	milk, diapers, cookies
T04	bread, butter, cookies
T05	beer, cookies, diapers
T06	milk, diapers, bread, butter
T07	bread, butter, diapers
T08	beer, diapers
T09	milk, diapers, bread, butter
T10	beer, cookies

1. **(a)** ¿Cuál es el número máximo de reglas de asociación que se pueden generar? (incluyendo reglas con soporte 0)

2. **(b)** ¿Cuál es el tamaño máximo de los conjuntos de ítems frecuentes que se pueden extraer (asumir umbral de minsoporte (>0))?

3. **(c)** Escribir una regla que contenga 3 ítems que se genere de este conjunto de datos.

4. **(d)** Encontrar un conjunto de ítems (de tamaño mayor a 2) con el valor de soporte máximo.

5. **(e)** Encontrar un par de ítems ((a) y (b)) tal que las reglas (a →b) y (b →a) tengan la misma confianza.

Respuesta

(a) Primero, se identifican los ítems únicos, los cuales son: *milk*, *beer*, *diapers*, *bread*, *butter* y *cookies*. Se obtiene un total de **6** ítems únicos.

El número de reglas posibles con (n) ítems es:

$$R = \sum_{k=1}^{n-1} \binom{n}{k} \cdot (2^k - 2)$$

Pero, se puede calcular de una manera más sencilla:

$$R = 3^n - 2^{n+1} + 1$$

Aplicando esta fórmula con (n = 6), obtenemos:

$$R = 3^6 - 2^7 + 1 = 602.$$

(b) El tamaño máximo de los conjuntos de ítems frecuentes hace referencia al conjunto de ítems más grande que aparece en al menos una transacción. De acuerdo con el conjunto de datos brindado, las transacciones T06 y T09 presentan el conjunto de datos más grande, con un valor de **4**. Este conjunto es:

$$\{\text{milk}, \text{diapers}, \text{bread}, \text{butter}\}.$$

Se calcula el soporte de la siguiente manera:

$$\text{Soporte}(\{\text{milk, diapers, bread, butter}\}) = \frac{2}{10} = 0.2.$$

De acuerdo con el soporte mínimo de 0, este conjunto es frecuente.

Conclusión:
El tamaño máximo de conjuntos frecuentes con soporte (>0) es **4**.

(c) Con base en el ítem anterior, una posible regla podría surgir de las transacciones T06 y T09. Por ejemplo, entre *milk*, *diapers* y *bread*. Una posible regla es:

$$\{\text{milk, diapers}\} \rightarrow \{\text{bread}\}.$$

Se verifica el soporte, que es el mismo calculado antes (aparece en T06 y T09):

$$\text{Soporte}(\{\text{milk, diapers, bread}\}) = \frac{2}{10} = 0.2.$$

Ahora, se calcula la confianza de la regla:

$$\text{Confianza}(\{\text{milk, diapers}\} \rightarrow \{\text{bread}\}) = \frac{\text{Soporte}(\{\text{milk, diapers, bread}\})}{\text{Soporte}(\{\text{milk, diapers}\})} = \frac{2/10}{4/10} = 0.5.$$

Por lo tanto, una **regla válida** es:

$$\{\text{milk, diapers}\} \rightarrow \{\text{bread}\}.$$

(d) En este apartado, aumenta la complejidad del problema, por lo que se aborda mediante Python. Se realizan los siguientes pasos:

- Se buscan conjuntos de tamaño 3 y 4 frecuentes junto con sus soportes.
- Se identifica el conjunto con el mayor soporte.

Como resultado se obtiene:

> El conjunto **{bread, butter, milk}** tiene tamaño **3** y soporte máximo **0.3**.

(e) De igual forma que el ítem anterior, se resuelve el problema por medio de Python. Se tienen en cuenta los pasos siguientes:

1. Buscar los pares (a) y (b) que cumplan la condición

   $$\frac{\text{soporte}(a,b)}{\text{soporte}(a)} = \frac{\text{soporte}(a,b)}{\text{soporte}(b)},$$

   es decir, ambos ítems tienen el mismo soporte.

2. Contar el soporte individual de cada ítem.
3. Identificar los pares con igual soporte.
4. Verificar la confianza de la regla.

Aplicando esto, se obtiene la siguiente solución:

El par **(milk, butter)** cumple con
$\text{conf}(milk \rightarrow butter) = \text{conf}(butter \rightarrow milk) = 0.6$.

[71]:
```python
from itertools import combinations, chain

# Se definen las transacciones
transactions = {
    "T01": {"milk", "beer", "diapers"},
    "T02": {"bread", "butter", "milk"},
    "T03": {"milk", "diapers", "cookies"},
    "T04": {"bread", "butter", "cookies"},
    "T05": {"beer", "cookies", "diapers"},
    "T06": {"milk", "diapers", "bread", "butter"},
    "T07": {"bread", "butter", "diapers"},
    "T08": {"beer", "diapers"},
    "T09": {"milk", "diapers", "bread", "butter"},
    "T10": {"beer", "cookies"}
}

all_items = set(chain.from_iterable(transactions.values()))

# Número total de ítems
n = len(all_items)

# (a) Número máximo de reglas
max_rules = 3**n - 2**(n+1) + 1
print(f"(a) Número máximo de reglas: {max_rules}")

# Función para calcular soporte (frecuencia relativa)
def support(itemset, transactions):
    count = 0
    for t in transactions.values():
        if itemset.issubset(t):
            count += 1
    return count / len(transactions)

# (b) Tamaño máximo de conjuntos frecuentes con soporte > 0
max_size = max(len(t) for t in transactions.values())
print(f"(b) Tamaño máximo de conjuntos frecuentes (soporte > 0): {max_size}")

# (c) Escribe una regla con 3 ítems (ejemplo)
# {milk, diapers} -> {bread}

antecedent = {"milk", "diapers"}
consequent = {"bread"}

# Soporte conjunto (antecedente U consecuente)
```

```
sup_conjunto = support(antecedent.union(consequent), transactions)

# Soporte antecedente
sup_antecedent = support(antecedent, transactions)

# Confianza de la regla
confianza = sup_conjunto / sup_antecedent if sup_antecedent > 0 else 0

print(f"(c) Regla: {antecedent} -> {consequent}")
print(f"    Soporte: {sup_conjunto:.2f}")
print(f"    Confianza: {confianza:.2f}")

# (d) Encontrar conjunto frecuente (tamaño > 2) con soporte máximo
# Primero se obtienen todos los conjuntos frecuentes de tamaño > 2

# Genera todos los conjuntos posibles de tamaño 3 y 4 (como máximo tamaño es 4)
max_possible_size = max_size
max_support = 0
max_support_set = None

for size in range(3, max_possible_size+1):
    # Generar combinaciones de todos los ítems
    for combo in combinations(all_items, size):
        combo_set = set(combo)
        sup = support(combo_set, transactions)
        if sup > max_support:
            max_support = sup
            max_support_set = combo_set

print(f"(d) Conjunto con soporte máximo (>2 ítems): {max_support_set}, soporte:
 ↪{max_support:.2f}")

# (e) Encontrar par de ítems (a, b) tal que las reglas a->b y b->a tengan misma
 ↪confianza
# Esto pasa si soporte(a) == soporte(b)

# Calcula soporte individual de cada ítem
item_supports = {item: support({item}, transactions) for item in all_items}

# Busca pares con igual soporte
pares_iguales = []
for a, b in combinations(all_items, 2):
    if abs(item_supports[a] - item_supports[b]) < 1e-6:  # igualdad con
 ↪tolerancia
        # Calcula soporte conjunto
        sup_conj = support({a, b}, transactions)
        if sup_conj > 0:
```

```
            conf_a_b = sup_conj / item_supports[a]
            conf_b_a = sup_conj / item_supports[b]
            if abs(conf_a_b - conf_b_a) < 1e-6:
                pares_iguales.append((a, b, conf_a_b))

# Muestra el primer par encontrado
if pares_iguales:
    a, b, conf = pares_iguales[0]
    print(f"(e) Par con reglas a->b y b->a misma confianza:")
    print(f"    a = {a}, b = {b}, confianza = {conf:.2f}")
else:
    print("(e) No se encontró ningún par con igual confianza.")
```

```
(a) Número máximo de reglas: 602
(b) Tamaño máximo de conjuntos frecuentes (soporte > 0): 4
(c) Regla: {'milk', 'diapers'} -> {'bread'}
    Soporte: 0.20
    Confianza: 0.50
(d) Conjunto con soporte máximo (>2 ítems): {'bread', 'butter', 'milk'},
soporte: 0.30
(e) Par con reglas a->b y b->a misma confianza:
    a = bread, b = butter, confianza = 1.00
```

1.0.4 Punto 4

Usando valores de umbral de soporte = 25 % y confianza = 60 %, encuentre:

a) Todos los conjuntos de ítems en la base de datos X

b) Reglas de asociación fuertes para la base de datos X

c) Analice las asociaciones engañosas para el conjunto de reglas obtenido en el numeral anterior.

[72]:
```
import pandas as pd
from efficient_apriori import apriori



#Hacer la tabla de datos
punto4 = {
    'TID': ['T01', 'T02', 'T03', 'T04', 'T05', 'T06', 'T07', 'T08'],
    'Items': [
        'A, B, C, D',
        'A, C, D, F',
          'C, D, E, G, A',
        'A, D, F, B',
        'B, C, G',
        'D, F, G',
        'A, B, G',
```

```
        'C, D, F, G'
    ]
}

punto4_df = pd.DataFrame(punto4)

# Convertir strings en listas (eliminar espacios)
punto4_df['Items'] = punto4_df['Items'].apply(lambda x: [item.strip() for item␣
 ↪in x.split(',')])

# Ver reglas usando apriori, usando umbral de soporte 0.25 y confianza 0.6
itemsets, rules = apriori(punto4_df['Items'], min_support=0.25,␣
 ↪min_confidence=0.6)


# Crear listas para cada atributo
lhs = []
rhs = []
support = []
confidence = []
lift = []

for rule in rules:
    lhs.append(tuple(rule.lhs))
    rhs.append(tuple(rule.rhs))
    support.append(rule.support)
    confidence.append(rule.confidence)
    lift.append(rule.lift)

# Crear DataFrame con los datos
df_rules = pd.DataFrame({
    'Antecedente': lhs,
    'Consecuente': rhs,
    'Soporte': support,
    'Confianza': confidence,
    'Lift': lift
})

print("a. Reglas con soporte mínimo de 0.25 y confianza mínima de 0.6:\n")
print(f"{df_rules}\n")

print("b. Reglas de asociación fuertes para la base de datos X \n")
print("La regla de asociacion mas fuerte es:\n")
print(df_rules.iloc[13])
print("\n")
print("Seguidas por las reglas, 10,11,17,20,21,24,25 con un lift de 1.3\n")
```

```
print("c. Reglas de asociación engañosas:")
print("Las reglas de asociacion que no aportan informacion y resultan engañosas␣
 ↪son:\n")
print("Las reglas, 2,3,8,9,12,14,23 con un lift menor que 1.\n Indicando que el␣
 ↪consecuente no es mas probable cuando el antecedente ocurre\n")
```

```
a. Reglas con soporte mínimo de 0.25 y confianza mínima de 0.6:

   Antecedente Consecuente  Soporte  Confianza      Lift
0          (B,)        (A,)    0.375   0.750000  1.200000
1          (A,)        (B,)    0.375   0.600000  1.200000
2          (C,)        (A,)    0.375   0.600000  0.960000
3          (A,)        (C,)    0.375   0.600000  0.960000
4          (D,)        (A,)    0.500   0.666667  1.066667
5          (A,)        (D,)    0.500   0.800000  1.066667
6          (D,)        (C,)    0.500   0.666667  1.066667
7          (C,)        (D,)    0.500   0.800000  1.066667
8          (G,)        (C,)    0.375   0.600000  0.960000
9          (C,)        (G,)    0.375   0.600000  0.960000
10         (F,)        (D,)    0.500   1.000000  1.333333
11         (D,)        (F,)    0.500   0.666667  1.333333
12         (G,)        (D,)    0.375   0.600000  0.800000
13      (B, D)        (A,)    0.250   1.000000  1.600000
14      (A, B)        (D,)    0.250   0.666667  0.888889
15      (C, D)        (A,)    0.375   0.750000  1.200000
16      (A, D)        (C,)    0.375   0.750000  1.200000
17      (A, C)        (D,)    0.375   1.000000  1.333333
18        (C,)      (A, D)    0.375   0.600000  1.200000
19        (A,)      (C, D)    0.375   0.600000  1.200000
20      (A, F)        (D,)    0.250   1.000000  1.333333
21      (C, F)        (D,)    0.250   1.000000  1.333333
22      (D, G)        (C,)    0.250   0.666667  1.066667
23      (C, G)        (D,)    0.250   0.666667  0.888889
24      (F, G)        (D,)    0.250   1.000000  1.333333
25      (D, G)        (F,)    0.250   0.666667  1.333333

b. Reglas de asociación fuertes para la base de datos X

La regla de asociacion mas fuerte es:

Antecedente    (B, D)
Consecuente      (A,)
Soporte          0.25
Confianza         1.0
Lift              1.6
Name: 13, dtype: object
```

```
Seguidas por las reglas, 10,11,17,20,21,24,25 con un lift de 1.3

c. Reglas de asociación engañosas:
Las reglas de asociacion que no aportan informacion y resultan engañosas son:

Las reglas, 2,3,8,9,12,14,23 con un lift menor que 1.
 Indicando que el consecuente no es mas probable cuando el antecedente ocurre
```

1.0.5 Punto 5

El algoritmo Apriori usa estrategías de generación y conteo para derivar conjuntos de items frecuentes. Conjuntos de item de tamaño k + 1 son creados de conjuntos de items de tamaño k . Un conjunto candidato es eliminado si uno de sus subconjuntos no es frecuente en la fase de poda. Supongamos que el algoritmo Apriori es aplicado a los datos de la siguiente tabla con un soporte mínimo de 30% (ejercicio realizado en clase).

id	items
1	a, b, d, e
2	b, c, d
3	a, b, d, e
4	a, c, d, e
5	b, c, d, e
6	b, d, e
7	c, d
8	a, b, c
9	a, d, e
10	b, d

(a) Dibujar el *lattice* de los conjuntos de ítems y etiquetar cada nodo con las siguientes letras:
- **N**: si el conjunto **no es considerado candidato**
- **F**: si el conjunto candidato es **frecuente**
- **I**: si el conjunto candidato **no es frecuente**

(b) ¿Cuál es el **porcentaje de conjuntos de ítems frecuentes**?

(c) ¿Cuál es el **radio de poda** en este conjunto de datos?
(El radio de poda se define como el **porcentaje de conjuntos de ítems no considerados candidatos**, ya sea porque no son generados durante la etapa de generación de candidatos o porque son podados en la etapa de poda).

(d) ¿Cuál es la **rata de falsa alarma**?
(Porcentaje de los conjuntos de ítems candidatos que son encontrados **NO frecuentes** después de calcular el soporte).

Solución punto (a)

- **(b) Porcentaje de ítems frecuentes:**
  $\frac{15}{30} = 0.5 = 50\%$
- **(c) Radio de poda:**
  $\frac{10}{30} = 0.33 = 33\%$
- **(d) Tasa de falsa alarma:**
  $\frac{5}{30} = 0.166 = 16\%$

1.0.6 Punto 6

Dada la siguiente base de datos transaccional Y:

TID	Ítems
1	A, B, C
2	A, C, D, E
3	A, B, D
4	A, C, F
5	A, B
6	A, E, F
7	A, B, D, E, F
8	A, F
9	B, D, E
10	B, D, E, F
11	B, C, D, E
12	C, D, E

Instrucciones

1. **Algoritmos**
   - Aplicar **Apriori**
   - Aplicar **FP-Growth**
2. **Soporte mínimo = 2**
   - Encontrar todos los conjuntos de ítems frecuentes.
   - Probar con varios valores de confianza para generar reglas de asociación.
3. **Análisis de resultados**
   - Ordenar los conjuntos frecuentes obtenidos.
   - Comparar resultados de Apriori vs FP-Growth (conjuntos y reglas).
   - Evaluar y comparar eficiencia (tiempo, número de pases, uso de memoria, etc.).
4. **Repetir con soporte mínimo = 3**
   - Volver a ejecutar ambos algoritmos.

- Comparar nuevos resultados con los del soporte = 2.

Respuesta:

Para dar respuesta a este problema, se hizo uso de la librería **`mlxtend`** de `Python`.

Acá, se utilizó el soporte relativo, es decir, el soporte requerido se dividió entre el total de las transacciones. Esto, para poder hacer uso de la facilidad de la librería **`mlxtend`**.

Análisis de los resultados

Para un soporte de 2, se evidenció que el mejor algoritmo es el de *FP-Growth* en comparación con *Apriori*. Fue más eficiente en cuanto a tiempo, por lo que si se tiene un conjunto de datos grande y se mantiene el soporte, *FP-Growth* puede ser una gran alternativa para realizar asociación.

Cuando se realiza la comparación con un soporte de 3, esto cambia. El algoritmo de *FP-Groth* y *Apriori* llegan a tener eficiencias muy semejantes en cuanto a tiempo; incluso, *Apriori* superando a *FP-Growth*. Esto sucede puede suceder por lo siguiente:

- Al ir aumentando el soporte, *FP-Growth* siempre debe realizar por lo menos dos pasadas completas sobre la base (una para contar frecuencias globales y ordernar ítems, otra para construir el *FP-tree*).
- Si el soporte es alto, sólo unos pocos ítems o pares resultan frecuentes, pero aún así *FP-Growth* incurre en la construcción del árbol y el gasto de punteros y enlaces internos.
- Con un soporte alto, *FP-Growth* "minaría" muy poco el árbol, debido a que se encontrarían patrones de tamaño no mayores a 1 o 2.

- *Apriori* puede resolverlo en una o dos pasadas sencillas, sin estructuras recursivas ni árboles.

Consideraciones Adicionales

Faltaría hacer el análisis cuando se aumenta el tamaño y la dimensionalida del dataset.

[73]:
```
#pip install mlxtend
```

[74]:
```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules, fpgrowth
from mlxtend.preprocessing import TransactionEncoder
import time

transactions6 = [
    ['A', 'B', 'C'],
    ['A', 'C', 'D', 'E'],
    ['A', 'B', 'D'],
    ['A', 'C', 'F'],
    ['A', 'B'],
    ['A', 'E', 'F'],
    ['A', 'B', 'D', 'E', 'F'],
    ['A', 'F'],
    ['B', 'D', 'E'],
```

```
    ['B', 'D', 'E', 'F'],
    ['B', 'C', 'D', 'E'],
    ['C', 'D', 'E']
]

#Convert to dataframe
te = TransactionEncoder()
te_ary = te.fit(transactions6).transform(transactions6)
df_t6 = pd.DataFrame(te_ary, columns=te.columns_)
df_t6
```

[74]:

	A	B	C	D	E	F
0	True	True	True	False	False	False
1	True	False	True	True	True	False
2	True	True	False	True	False	False
3	True	False	True	False	False	True
4	True	True	False	False	False	False
5	True	False	False	False	True	True
6	True	True	False	True	True	True
7	True	False	False	False	False	True
8	False	True	False	True	True	False
9	False	True	False	True	True	True
10	False	True	True	True	True	False
11	False	False	True	True	True	False

[75]:
```
#Total support
min_sup = 2

#Relative support
rel_sup = min_sup / len(transactions6)
rel_sup #0.166
```

[75]:
```
0.16666666666666666
```

[76]:
```
start = time.time()
r_apriori = apriori(df_t6, min_support= rel_sup, use_colnames = True)
end = time.time()
print(f"Tiempo de ejecución Apriori: {end - start} segundos")
```

```
Tiempo de ejecución Apriori: 0.002424001693725586 segundos
```

[77]:
```
start = time.time()
r_fpgrowth = fpgrowth(df_t6, min_support=rel_sup, use_colnames=True)
end = time.time()
print(f"Tiempo de ejecución FP-Growth: {end - start} segundos")
```

```
Tiempo de ejecución FP-Growth: 0.003003835678100586 segundos
```

[78]:
```
r_apriori
```

[78]:

	support	itemsets
0	0.666667	(A)
1	0.583333	(B)
2	0.416667	(C)
3	0.583333	(D)
4	0.583333	(E)
5	0.416667	(F)
6	0.333333	(B, A)
7	0.250000	(C, A)
8	0.250000	(D, A)
9	0.250000	(E, A)
10	0.333333	(F, A)
11	0.166667	(B, C)
12	0.416667	(B, D)
13	0.333333	(B, E)
14	0.166667	(B, F)
15	0.250000	(D, C)
16	0.250000	(C, E)
17	0.500000	(D, E)
18	0.166667	(D, F)
19	0.250000	(F, E)
20	0.166667	(B, D, A)
21	0.166667	(D, E, A)
22	0.166667	(F, E, A)
23	0.333333	(B, D, E)
24	0.166667	(B, F, D)
25	0.166667	(B, F, E)
26	0.250000	(D, C, E)
27	0.166667	(D, F, E)
28	0.166667	(B, F, D, E)

[79]:
```
r_fpgrowth['length'] = r_fpgrowth['itemsets'].apply(len)
r_fpgrowth = r_fpgrowth.sort_values('length', ascending=True)
r_fpgrowth
```

[79]:

	support	itemsets	length
0	0.666667	(A)	1
1	0.583333	(B)	1
2	0.416667	(C)	1
3	0.583333	(E)	1
4	0.583333	(D)	1
5	0.416667	(F)	1
23	0.166667	(D, F)	2
22	0.166667	(B, F)	2
21	0.250000	(F, E)	2

```
20  0.333333        (F, A)       2
18  0.250000        (D, A)       2
17  0.500000        (D, E)       2
16  0.250000        (E, A)       2
14  0.250000        (C, E)       2
12  0.166667        (B, C)       2
11  0.250000        (C, A)       2
8   0.333333        (B, E)       2
7   0.416667        (B, D)       2
6   0.333333        (B, A)       2
13  0.250000        (D, C)       2
27  0.166667     (D, F, E)       3
15  0.250000     (D, C, E)       3
10  0.333333     (B, D, E)       3
19  0.166667     (D, E, A)       3
9   0.166667     (B, D, A)       3
24  0.166667     (F, E, A)       3
25  0.166667     (B, F, E)       3
26  0.166667     (B, F, D)       3
28  0.166667  (B, F, D, E)       4
```

[80]:
```
#Total support
min_sup_3 = 3

#Relative support
rel_sup_3 = min_sup_3 / len(transactions6)
rel_sup_3 #0.25
```

[80]:
```
0.25
```

[81]:
```
start = time.time()
r_apriori_3 = apriori(df_t6, min_support= rel_sup_3, use_colnames = True)
end = time.time()
print(f"Tiempo de ejecución Apriori: {end - start} segundos")
```

```
Tiempo de ejecución Apriori: 0.0037508010864257812 segundos
```

[82]:
```
start = time.time()
r_fpgrowth_3 = fpgrowth(df_t6, min_support=rel_sup_3, use_colnames=True)
end = time.time()
print(f"Tiempo de ejecución FP-Growth: {end - start} segundos")
```

```
Tiempo de ejecución FP-Growth: 0.0030808448791503906 segundos
```

[83]:
```
r_apriori_3
```

[83]:
```
     support   itemsets
0   0.666667        (A)
```

```
1   0.583333        (B)
2   0.416667        (C)
3   0.583333        (D)
4   0.583333        (E)
5   0.416667        (F)
6   0.333333     (B, A)
7   0.250000     (C, A)
8   0.250000     (D, A)
9   0.250000     (E, A)
10  0.333333     (F, A)
11  0.416667     (B, D)
12  0.333333     (B, E)
13  0.250000     (D, C)
14  0.250000     (C, E)
15  0.500000     (D, E)
16  0.250000     (F, E)
17  0.333333  (B, D, E)
18  0.250000  (D, C, E)
```

[84]:
```
r_fpgrowth_3['length'] = r_fpgrowth_3['itemsets'].apply(len)
r_fpgrowth_3 = r_fpgrowth_3.sort_values('length', ascending=True)
r_fpgrowth_3
```

[84]:

	support	itemsets	length
0	0.666667	(A)	1
1	0.583333	(B)	1
2	0.416667	(C)	1
3	0.583333	(E)	1
4	0.583333	(D)	1
5	0.416667	(F)	1
16	0.250000	(D, A)	2
15	0.500000	(D, E)	2
14	0.250000	(E, A)	2
12	0.250000	(C, E)	2
11	0.250000	(D, C)	2
18	0.250000	(F, E)	2
17	0.333333	(F, A)	2
8	0.333333	(B, E)	2
7	0.416667	(B, D)	2
6	0.333333	(B, A)	2
10	0.250000	(C, A)	2
13	0.250000	(D, C, E)	3
9	0.333333	(B, D, E)	3

1.0.7 Punto 7

Usando el conjunto de datos del punto 5:

(a) Realizar la **tabla de contingencia** para las siguientes reglas:
- **b** $\rightarrow$ **c**
- **a** $\rightarrow$ **d**
- **b** $\rightarrow$ **d**
- **e** $\rightarrow$ **c**
- **c** $\rightarrow$ **a**

(b) Usar las tablas de contingencia del punto anterior para computar y **realizar un ranking** de las reglas usando:

- **i. Soporte**
- **ii. Confianza**
- **iii. Lift**

Regla: {b} $\rightarrow$ {c}

	c	c'	Total
b	2	4	6
b'	3	1	4
Total	5	5	10

Cálculos: - Soporte: $S = \frac{2}{10} = 0{,}2$ - Confianza: $C = \frac{2}{6} = 0{,}33$ - Lift: $\frac{0{,}33}{0{,}5} = 0{,}66$

Regla: {a} $\rightarrow$ {d}

	d	d'	Total
a	4	1	5
a'	5	0	5
Total	9	1	10

Cálculos: - Soporte: $S = \frac{4}{10} = 0{,}4$ - Confianza: $C = \frac{4}{5} = 0{,}8$ - Lift: $\frac{0{,}8}{0{,}9} = 0{,}88$

Regla: {b} $\rightarrow$ {d}

	d	d'	Total
b	6	1	7
b'	3	0	3
Total	9	1	10

Cálculos: - Soporte: $S = \frac{6}{10} = 0{,}6$ - Confianza: $C = \frac{6}{7} = 0{,}85$ - Lift: $\frac{0{,}85}{0{,}9} = 0{,}94$

Regla: {e} → {c}

	c	c'	Total
e	2	4	6
e'	3	1	4
Total	5	5	10

Cálculos: - Soporte: $S = \frac{2}{10} = 0{,}2$ - Confianza: $C = \frac{2}{6} = 0{,}33$ - Lift: $\frac{0{,}33}{0{,}5} = 0{,}66$

Regla: {c} → {a}

	a	a'	Total
c	2	3	5
c'	3	2	5
Total	5	5	10

Cálculos: - Soporte: $S = \frac{2}{10} = 0{,}2$ - Confianza: $C = \frac{2}{5} = 0{,}4$ - Lift: $\frac{0{,}4}{0{,}5} = 0{,}8$

B) Ranking

#	Regla	Soporte	Confianza	Lift
1	{b} → {d}	0.6	0.85	0.94
2	{a} → {d}	0.4	0.80	0.88
3	{c} → {a}	0.2	0.40	0.80
4	{b} → {c}	0.2	0.33	0.66
5	{e} → {c}	0.2	0.33	0.66

1.0.8 Punto 8

Importar el conjunto de datos marketBasquet.csv. Extraer reglas de asociación utilizando diferentes valores mínimos de confíanza. - Analice los resultados. - Reporte los itemsets frecuentes con sus respectivos valores de soporte. - Revise los valores de soporte reportados por el operador de itemsets frecuentes. - Verifíquelos manualmente. ¿Qué problema se presenta? - Extraiga reglas de asociación utilizando diferentes valores mínimos de confíanza. - Analice los resultados. - Reporte los itemsets frecuentes con sus respectivos valores de soporte. - Analice los resultados y discuta las diferencias con el esquema que no posee un filtro de atributos.

[85]:
```
# importar el conjunto marketBasket.csv
import pandas as pd
import numpy as np

marketBasket = pd.read_csv('/Users/fjosesala/Documents/GitHub/DataMining2025-1/
  ↪Workshops/Workshop3 - Association_rules/data/marketBasket.csv', sep=';')
```

```
marketBasket.head(10)
```

[85]:

	ID	Leche	Cerveza	Pañales	Pan	Mantequilla	Galletas
0	1	True	True	True	False	False	False
1	2	True	False	False	True	True	False
2	3	True	False	True	False	False	True
3	4	False	False	False	True	True	True
4	5	False	True	True	False	False	True
5	6	True	False	True	True	True	False
6	7	False	False	True	True	True	False
7	8	False	True	True	False	False	False
8	9	True	False	True	True	True	False
9	10	False	True	False	False	False	True

[86]:
```
# ver informacion del conjunto de datos
print(marketBasket.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   ID           10 non-null     int64
 1   Leche        10 non-null     bool
 2   Cerveza      10 non-null     bool
 3   Pañales      10 non-null     bool
 4   Pan          10 non-null     bool
 5   Mantequilla  10 non-null     bool
 6   Galletas     10 non-null     bool
dtypes: bool(6), int64(1)
memory usage: 272.0 bytes
None
```

[87]:
```
# Aplicar el algoritmo Apriori
from mlxtend.frequent_patterns import apriori, association_rules

# Eliminar columna id
basket = marketBasket.drop(columns=['ID'])

# Carcular items frecuentes, se udara un soporte minimo de 0.2
items_frecuentes = apriori(basket, min_support=0.2, use_colnames=True)

# ver resultados
print(items_frecuentes)
```

```
    support                itemsets
0       0.5                 (Leche)
```

```
1       0.4                          (Cerveza)
2       0.7                          (Pañales)
3       0.5                             (Pan)
4       0.5                      (Mantequilla)
5       0.4                         (Galletas)
6       0.4                    (Leche, Pañales)
7       0.3                       (Leche, Pan)
8       0.3                (Leche, Mantequilla)
9       0.3                  (Cerveza, Pañales)
10      0.2                 (Cerveza, Galletas)
11      0.3                      (Pañales, Pan)
12      0.3              (Pañales, Mantequilla)
13      0.2                 (Pañales, Galletas)
14      0.5                  (Pan, Mantequilla)
15      0.2               (Leche, Pañales, Pan)
16      0.2       (Leche, Pañales, Mantequilla)
17      0.3           (Leche, Pan, Mantequilla)
18      0.3         (Pañales, Pan, Mantequilla)
19      0.2  (Leche, Pañales, Pan, Mantequilla)
```

[88]:
```
# Verificar manualmente el soporte.
# En este caso leche junto a pañales

#Calcular el Numero de transacciondes en donde leche esta en conjunto con␣
 ↪pañales
leche_pañales = basket[(basket['Leche']) & (basket['Pañales'])].shape[0] /␣
 ↪basket.shape[0]
print(leche_pañales)
```

```
0.4
```

Algunos de los problemas que pueden ocurrir es tener inconsistencia en los datos booleanos, al igual que mlxtend espera un arreglo de datos booleanos, por lo que si se tiene un string o un int presentara error.

[89]:
```
# Generar reglas de asociacion con diferentes umbrales de confianza
from mlxtend.frequent_patterns import association_rules

# Generar reglas de asociacion conconfianza minima de 0.3
regla_conf_3 = association_rules(items_frecuentes, metric="confidence",␣
 ↪min_threshold=0.3)
#pasar el resultado a un dataframe
regla_conf_3 = pd.DataFrame(regla_conf_3)
regla_conf_3
```

[89]:
```
                   antecedents                  consequents  \
0                      (Leche)                    (Pañales)
1                    (Pañales)                      (Leche)
```

2	(Leche)	(Pan)
3	(Pan)	(Leche)
4	(Leche)	(Mantequilla)
5	(Mantequilla)	(Leche)
6	(Cerveza)	(Pañales)
7	(Pañales)	(Cerveza)
8	(Cerveza)	(Galletas)
9	(Galletas)	(Cerveza)
10	(Pañales)	(Pan)
11	(Pan)	(Pañales)
12	(Pañales)	(Mantequilla)
13	(Mantequilla)	(Pañales)
14	(Galletas)	(Pañales)
15	(Pan)	(Mantequilla)
16	(Mantequilla)	(Pan)
17	(Leche, Pañales)	(Pan)
18	(Leche, Pan)	(Pañales)
19	(Pañales, Pan)	(Leche)
20	(Leche)	(Pañales, Pan)
21	(Pan)	(Leche, Pañales)
22	(Leche, Pañales)	(Mantequilla)
23	(Leche, Mantequilla)	(Pañales)
24	(Pañales, Mantequilla)	(Leche)
25	(Leche)	(Pañales, Mantequilla)
26	(Mantequilla)	(Leche, Pañales)
27	(Leche, Pan)	(Mantequilla)
28	(Leche, Mantequilla)	(Pan)
29	(Pan, Mantequilla)	(Leche)
30	(Leche)	(Pan, Mantequilla)
31	(Pan)	(Leche, Mantequilla)
32	(Mantequilla)	(Leche, Pan)
33	(Pañales, Pan)	(Mantequilla)
34	(Pañales, Mantequilla)	(Pan)
35	(Pan, Mantequilla)	(Pañales)
36	(Pañales)	(Pan, Mantequilla)
37	(Pan)	(Pañales, Mantequilla)
38	(Mantequilla)	(Pañales, Pan)
39	(Leche, Pañales, Pan)	(Mantequilla)
40	(Leche, Pañales, Mantequilla)	(Pan)
41	(Leche, Pan, Mantequilla)	(Pañales)
42	(Pañales, Pan, Mantequilla)	(Leche)
43	(Leche, Pañales)	(Pan, Mantequilla)
44	(Leche, Pan)	(Pañales, Mantequilla)
45	(Leche, Mantequilla)	(Pañales, Pan)
46	(Pañales, Pan)	(Leche, Mantequilla)
47	(Pañales, Mantequilla)	(Leche, Pan)
48	(Pan, Mantequilla)	(Leche, Pañales)

```
49                      (Leche)    (Pañales, Pan, Mantequilla)
50                        (Pan)  (Leche, Pañales, Mantequilla)
51                 (Mantequilla)          (Leche, Pañales, Pan)

    antecedent support  consequent support  support  confidence      lift  \
0                  0.5                 0.7      0.4    0.800000  1.142857
1                  0.7                 0.5      0.4    0.571429  1.142857
2                  0.5                 0.5      0.3    0.600000  1.200000
3                  0.5                 0.5      0.3    0.600000  1.200000
4                  0.5                 0.5      0.3    0.600000  1.200000
5                  0.5                 0.5      0.3    0.600000  1.200000
6                  0.4                 0.7      0.3    0.750000  1.071429
7                  0.7                 0.4      0.3    0.428571  1.071429
8                  0.4                 0.4      0.2    0.500000  1.250000
9                  0.4                 0.4      0.2    0.500000  1.250000
10                 0.7                 0.5      0.3    0.428571  0.857143
11                 0.5                 0.7      0.3    0.600000  0.857143
12                 0.7                 0.5      0.3    0.428571  0.857143
13                 0.5                 0.7      0.3    0.600000  0.857143
14                 0.4                 0.7      0.2    0.500000  0.714286
15                 0.5                 0.5      0.5    1.000000  2.000000
16                 0.5                 0.5      0.5    1.000000  2.000000
17                 0.4                 0.5      0.2    0.500000  1.000000
18                 0.3                 0.7      0.2    0.666667  0.952381
19                 0.3                 0.5      0.2    0.666667  1.333333
20                 0.5                 0.3      0.2    0.400000  1.333333
21                 0.5                 0.4      0.2    0.400000  1.000000
22                 0.4                 0.5      0.2    0.500000  1.000000
23                 0.3                 0.7      0.2    0.666667  0.952381
24                 0.3                 0.5      0.2    0.666667  1.333333
25                 0.5                 0.3      0.2    0.400000  1.333333
26                 0.5                 0.4      0.2    0.400000  1.000000
27                 0.3                 0.5      0.3    1.000000  2.000000
28                 0.3                 0.5      0.3    1.000000  2.000000
29                 0.5                 0.5      0.3    0.600000  1.200000
30                 0.5                 0.5      0.3    0.600000  1.200000
31                 0.5                 0.3      0.3    0.600000  2.000000
32                 0.5                 0.3      0.3    0.600000  2.000000
33                 0.3                 0.5      0.3    1.000000  2.000000
34                 0.3                 0.5      0.3    1.000000  2.000000
35                 0.5                 0.7      0.3    0.600000  0.857143
36                 0.7                 0.5      0.3    0.428571  0.857143
37                 0.5                 0.3      0.3    0.600000  2.000000
38                 0.5                 0.3      0.3    0.600000  2.000000
39                 0.2                 0.5      0.2    1.000000  2.000000
40                 0.2                 0.5      0.2    1.000000  2.000000
41                 0.3                 0.7      0.2    0.666667  0.952381
```

42	0.3	0.5	0.2	0.666667	1.333333
43	0.4	0.5	0.2	0.500000	1.000000
44	0.3	0.3	0.2	0.666667	2.222222
45	0.3	0.3	0.2	0.666667	2.222222
46	0.3	0.3	0.2	0.666667	2.222222
47	0.3	0.3	0.2	0.666667	2.222222
48	0.5	0.4	0.2	0.400000	1.000000
49	0.5	0.3	0.2	0.400000	1.333333
50	0.5	0.2	0.2	0.400000	2.000000
51	0.5	0.2	0.2	0.400000	2.000000

	representativity	leverage	conviction	zhangs_metric	jaccard	\
0	1.0	0.05	1.500000	0.250000	0.500000	
1	1.0	0.05	1.166667	0.416667	0.500000	
2	1.0	0.05	1.250000	0.333333	0.428571	
3	1.0	0.05	1.250000	0.333333	0.428571	
4	1.0	0.05	1.250000	0.333333	0.428571	
5	1.0	0.05	1.250000	0.333333	0.428571	
6	1.0	0.02	1.200000	0.111111	0.375000	
7	1.0	0.02	1.050000	0.222222	0.375000	
8	1.0	0.04	1.200000	0.333333	0.333333	
9	1.0	0.04	1.200000	0.333333	0.333333	
10	1.0	-0.05	0.875000	-0.357143	0.333333	
11	1.0	-0.05	0.750000	-0.250000	0.333333	
12	1.0	-0.05	0.875000	-0.357143	0.333333	
13	1.0	-0.05	0.750000	-0.250000	0.333333	
14	1.0	-0.08	0.600000	-0.400000	0.222222	
15	1.0	0.25	inf	1.000000	1.000000	
16	1.0	0.25	inf	1.000000	1.000000	
17	1.0	0.00	1.000000	0.000000	0.285714	
18	1.0	-0.01	0.900000	-0.066667	0.250000	
19	1.0	0.05	1.500000	0.357143	0.333333	
20	1.0	0.05	1.166667	0.500000	0.333333	
21	1.0	0.00	1.000000	0.000000	0.285714	
22	1.0	0.00	1.000000	0.000000	0.285714	
23	1.0	-0.01	0.900000	-0.066667	0.250000	
24	1.0	0.05	1.500000	0.357143	0.333333	
25	1.0	0.05	1.166667	0.500000	0.333333	
26	1.0	0.00	1.000000	0.000000	0.285714	
27	1.0	0.15	inf	0.714286	0.600000	
28	1.0	0.15	inf	0.714286	0.600000	
29	1.0	0.05	1.250000	0.333333	0.428571	
30	1.0	0.05	1.250000	0.333333	0.428571	
31	1.0	0.15	1.750000	1.000000	0.600000	
32	1.0	0.15	1.750000	1.000000	0.600000	
33	1.0	0.15	inf	0.714286	0.600000	
34	1.0	0.15	inf	0.714286	0.600000	

35	1.0	-0.05	0.750000	-0.250000	0.333333
36	1.0	-0.05	0.875000	-0.357143	0.333333
37	1.0	0.15	1.750000	1.000000	0.600000
38	1.0	0.15	1.750000	1.000000	0.600000
39	1.0	0.10	inf	0.625000	0.400000
40	1.0	0.10	inf	0.625000	0.400000
41	1.0	-0.01	0.900000	-0.066667	0.250000
42	1.0	0.05	1.500000	0.357143	0.333333
43	1.0	0.00	1.000000	0.000000	0.285714
44	1.0	0.11	2.100000	0.785714	0.500000
45	1.0	0.11	2.100000	0.785714	0.500000
46	1.0	0.11	2.100000	0.785714	0.500000
47	1.0	0.11	2.100000	0.785714	0.500000
48	1.0	0.00	1.000000	0.000000	0.285714
49	1.0	0.05	1.166667	0.500000	0.333333
50	1.0	0.10	1.333333	1.000000	0.400000
51	1.0	0.10	1.333333	1.000000	0.400000

	certainty	kulczynski
0	0.333333	0.685714
1	0.142857	0.685714
2	0.200000	0.600000
3	0.200000	0.600000
4	0.200000	0.600000
5	0.200000	0.600000
6	0.166667	0.589286
7	0.047619	0.589286
8	0.166667	0.500000
9	0.166667	0.500000
10	-0.142857	0.514286
11	-0.333333	0.514286
12	-0.142857	0.514286
13	-0.333333	0.514286
14	-0.666667	0.392857
15	1.000000	1.000000
16	1.000000	1.000000
17	0.000000	0.450000
18	-0.111111	0.476190
19	0.333333	0.533333
20	0.142857	0.533333
21	0.000000	0.450000
22	0.000000	0.450000
23	-0.111111	0.476190
24	0.333333	0.533333
25	0.142857	0.533333
26	0.000000	0.450000
27	1.000000	0.800000

```
28   1.000000    0.800000
29   0.200000    0.600000
30   0.200000    0.600000
31   0.428571    0.800000
32   0.428571    0.800000
33   1.000000    0.800000
34   1.000000    0.800000
35  -0.333333    0.514286
36  -0.142857    0.514286
37   0.428571    0.800000
38   0.428571    0.800000
39   1.000000    0.700000
40   1.000000    0.700000
41  -0.111111    0.476190
42   0.333333    0.533333
43   0.000000    0.450000
44   0.523810    0.666667
45   0.523810    0.666667
46   0.523810    0.666667
47   0.523810    0.666667
48   0.000000    0.450000
49   0.142857    0.533333
50   0.250000    0.700000
51   0.250000    0.700000
```

[90]:
```
# Generar reglas de asociacion conconfianza minima de 0.5
regla_conf_5 = association_rules(items_frecuentes, metric="confidence",␣
 ↪min_threshold=0.5)
#pasar el resultado a un dataframe
regla_conf_5 = pd.DataFrame(regla_conf_5)
regla_conf_5
```

[90]:

	antecedents	consequents	antecedent support	\
0	(Leche)	(Pañales)	0.5	
1	(Pañales)	(Leche)	0.7	
2	(Leche)	(Pan)	0.5	
3	(Pan)	(Leche)	0.5	
4	(Leche)	(Mantequilla)	0.5	
5	(Mantequilla)	(Leche)	0.5	
6	(Cerveza)	(Pañales)	0.4	
7	(Cerveza)	(Galletas)	0.4	
8	(Galletas)	(Cerveza)	0.4	
9	(Pan)	(Pañales)	0.5	
10	(Mantequilla)	(Pañales)	0.5	
11	(Galletas)	(Pañales)	0.4	
12	(Pan)	(Mantequilla)	0.5	
13	(Mantequilla)	(Pan)	0.5	

```
14               (Leche, Pañales)                  (Pan)                0.4
15                   (Leche, Pan)              (Pañales)                0.3
16                 (Pañales, Pan)                (Leche)                0.3
17               (Leche, Pañales)           (Mantequilla)                0.4
18            (Leche, Mantequilla)              (Pañales)                0.3
19          (Pañales, Mantequilla)                (Leche)                0.3
20                   (Leche, Pan)           (Mantequilla)                0.3
21            (Leche, Mantequilla)                   (Pan)                0.3
22              (Pan, Mantequilla)                (Leche)                0.5
23                          (Leche)      (Pan, Mantequilla)                0.5
24                            (Pan)    (Leche, Mantequilla)                0.5
25                   (Mantequilla)             (Leche, Pan)                0.5
26                 (Pañales, Pan)           (Mantequilla)                0.3
27          (Pañales, Mantequilla)                   (Pan)                0.3
28              (Pan, Mantequilla)              (Pañales)                0.5
29                            (Pan)  (Pañales, Mantequilla)                0.5
30                   (Mantequilla)          (Pañales, Pan)                0.5
31           (Leche, Pañales, Pan)           (Mantequilla)                0.2
32  (Leche, Pañales, Mantequilla)                   (Pan)                0.2
33      (Leche, Pan, Mantequilla)              (Pañales)                0.3
34    (Pañales, Pan, Mantequilla)                (Leche)                0.3
35               (Leche, Pañales)      (Pan, Mantequilla)                0.4
36                   (Leche, Pan)  (Pañales, Mantequilla)                0.3
37            (Leche, Mantequilla)          (Pañales, Pan)                0.3
38                 (Pañales, Pan)    (Leche, Mantequilla)                0.3
39          (Pañales, Mantequilla)             (Leche, Pan)                0.3

    consequent support  support  confidence      lift  representativity  \
0                  0.7      0.4    0.800000  1.142857               1.0
1                  0.5      0.4    0.571429  1.142857               1.0
2                  0.5      0.3    0.600000  1.200000               1.0
3                  0.5      0.3    0.600000  1.200000               1.0
4                  0.5      0.3    0.600000  1.200000               1.0
5                  0.5      0.3    0.600000  1.200000               1.0
6                  0.7      0.3    0.750000  1.071429               1.0
7                  0.4      0.2    0.500000  1.250000               1.0
8                  0.4      0.2    0.500000  1.250000               1.0
9                  0.7      0.3    0.600000  0.857143               1.0
10                 0.7      0.3    0.600000  0.857143               1.0
11                 0.7      0.2    0.500000  0.714286               1.0
12                 0.5      0.5    1.000000  2.000000               1.0
13                 0.5      0.5    1.000000  2.000000               1.0
14                 0.5      0.2    0.500000  1.000000               1.0
15                 0.7      0.2    0.666667  0.952381               1.0
16                 0.5      0.2    0.666667  1.333333               1.0
17                 0.5      0.2    0.500000  1.000000               1.0
18                 0.7      0.2    0.666667  0.952381               1.0
```

19	0.5	0.2	0.666667	1.333333	1.0
20	0.5	0.3	1.000000	2.000000	1.0
21	0.5	0.3	1.000000	2.000000	1.0
22	0.5	0.3	0.600000	1.200000	1.0
23	0.5	0.3	0.600000	1.200000	1.0
24	0.3	0.3	0.600000	2.000000	1.0
25	0.3	0.3	0.600000	2.000000	1.0
26	0.5	0.3	1.000000	2.000000	1.0
27	0.5	0.3	1.000000	2.000000	1.0
28	0.7	0.3	0.600000	0.857143	1.0
29	0.3	0.3	0.600000	2.000000	1.0
30	0.3	0.3	0.600000	2.000000	1.0
31	0.5	0.2	1.000000	2.000000	1.0
32	0.5	0.2	1.000000	2.000000	1.0
33	0.7	0.2	0.666667	0.952381	1.0
34	0.5	0.2	0.666667	1.333333	1.0
35	0.5	0.2	0.500000	1.000000	1.0
36	0.3	0.2	0.666667	2.222222	1.0
37	0.3	0.2	0.666667	2.222222	1.0
38	0.3	0.2	0.666667	2.222222	1.0
39	0.3	0.2	0.666667	2.222222	1.0

	leverage	conviction	zhangs_metric	jaccard	certainty	kulczynski
0	0.05	1.500000	0.250000	0.500000	0.333333	0.685714
1	0.05	1.166667	0.416667	0.500000	0.142857	0.685714
2	0.05	1.250000	0.333333	0.428571	0.200000	0.600000
3	0.05	1.250000	0.333333	0.428571	0.200000	0.600000
4	0.05	1.250000	0.333333	0.428571	0.200000	0.600000
5	0.05	1.250000	0.333333	0.428571	0.200000	0.600000
6	0.02	1.200000	0.111111	0.375000	0.166667	0.589286
7	0.04	1.200000	0.333333	0.333333	0.166667	0.500000
8	0.04	1.200000	0.333333	0.333333	0.166667	0.500000
9	-0.05	0.750000	-0.250000	0.333333	-0.333333	0.514286
10	-0.05	0.750000	-0.250000	0.333333	-0.333333	0.514286
11	-0.08	0.600000	-0.400000	0.222222	-0.666667	0.392857
12	0.25	inf	1.000000	1.000000	1.000000	1.000000
13	0.25	inf	1.000000	1.000000	1.000000	1.000000
14	0.00	1.000000	0.000000	0.285714	0.000000	0.450000
15	-0.01	0.900000	-0.066667	0.250000	-0.111111	0.476190
16	0.05	1.500000	0.357143	0.333333	0.333333	0.533333
17	0.00	1.000000	0.000000	0.285714	0.000000	0.450000
18	-0.01	0.900000	-0.066667	0.250000	-0.111111	0.476190
19	0.05	1.500000	0.357143	0.333333	0.333333	0.533333
20	0.15	inf	0.714286	0.600000	1.000000	0.800000
21	0.15	inf	0.714286	0.600000	1.000000	0.800000
22	0.05	1.250000	0.333333	0.428571	0.200000	0.600000
23	0.05	1.250000	0.333333	0.428571	0.200000	0.600000

```
24      0.15    1.750000       1.000000  0.600000   0.428571    0.800000
25      0.15    1.750000       1.000000  0.600000   0.428571    0.800000
26      0.15         inf       0.714286  0.600000   1.000000    0.800000
27      0.15         inf       0.714286  0.600000   1.000000    0.800000
28     -0.05    0.750000      -0.250000  0.333333  -0.333333    0.514286
29      0.15    1.750000       1.000000  0.600000   0.428571    0.800000
30      0.15    1.750000       1.000000  0.600000   0.428571    0.800000
31      0.10         inf       0.625000  0.400000   1.000000    0.700000
32      0.10         inf       0.625000  0.400000   1.000000    0.700000
33     -0.01    0.900000      -0.066667  0.250000  -0.111111    0.476190
34      0.05    1.500000       0.357143  0.333333   0.333333    0.533333
35      0.00    1.000000       0.000000  0.285714   0.000000    0.450000
36      0.11    2.100000       0.785714  0.500000   0.523810    0.666667
37      0.11    2.100000       0.785714  0.500000   0.523810    0.666667
38      0.11    2.100000       0.785714  0.500000   0.523810    0.666667
39      0.11    2.100000       0.785714  0.500000   0.523810    0.666667
```

[91]:
```
# Generar reglas de asociacion conconfianza minima de 0.8
regla_conf_8 = association_rules(items_frecuentes, metric="confidence",␣
 ↪min_threshold=0.8)
#pasar el resultado a un dataframe
regla_conf_8 = pd.DataFrame(regla_conf_8)
regla_conf_8
```

[91]:
```
                    antecedents    consequents  antecedent support  \
0                        (Leche)      (Pañales)                 0.5
1                          (Pan)  (Mantequilla)                 0.5
2                  (Mantequilla)          (Pan)                 0.5
3                    (Leche, Pan)  (Mantequilla)                 0.3
4            (Leche, Mantequilla)          (Pan)                 0.3
5                  (Pañales, Pan)  (Mantequilla)                 0.3
6          (Pañales, Mantequilla)          (Pan)                 0.3
7           (Leche, Pañales, Pan)  (Mantequilla)                 0.2
8  (Leche, Pañales, Mantequilla)          (Pan)                 0.2

   consequent support  support  confidence      lift  representativity  \
0                 0.7      0.4         0.8  1.142857               1.0
1                 0.5      0.5         1.0  2.000000               1.0
2                 0.5      0.5         1.0  2.000000               1.0
3                 0.5      0.3         1.0  2.000000               1.0
4                 0.5      0.3         1.0  2.000000               1.0
5                 0.5      0.3         1.0  2.000000               1.0
6                 0.5      0.3         1.0  2.000000               1.0
7                 0.5      0.2         1.0  2.000000               1.0
8                 0.5      0.2         1.0  2.000000               1.0

   leverage  conviction  zhangs_metric  jaccard  certainty  kulczynski
```

0	0.05	1.5	0.250000	0.5	0.333333	0.685714
1	0.25	inf	1.000000	1.0	1.000000	1.000000
2	0.25	inf	1.000000	1.0	1.000000	1.000000
3	0.15	inf	0.714286	0.6	1.000000	0.800000
4	0.15	inf	0.714286	0.6	1.000000	0.800000
5	0.15	inf	0.714286	0.6	1.000000	0.800000
6	0.15	inf	0.714286	0.6	1.000000	0.800000
7	0.10	inf	0.625000	0.4	1.000000	0.700000
8	0.10	inf	0.625000	0.4	1.000000	0.700000

[92]:
```
#obtener nombres de las columnas
columnas = regla_conf_8.columns
print(columnas)
```

```
Index(['antecedents', 'consequents', 'antecedent support',
       'consequent support', 'support', 'confidence', 'lift',
       'representativity', 'leverage', 'conviction', 'zhangs_metric',
       'jaccard', 'certainty', 'kulczynski'],
      dtype='object')
```

Las reglas más consistentes y significativas entre diferentes umbrales de confianza son:

Pan → Mantequilla

Leche, Pan → Mantequilla

Leche, Mantequilla → Pan

Pañales, Pan → Mantequilla

Pañales, Mantequilla → Pan

Usar umbrales bajos de confianza genera muchas más reglas, pero muchas no aportan valor real (confianza baja o lift cercano a 1).

Por lo que se concluye, que umbrales bajos amplian el numero de reglas, con el riesgo de incluir asociaciones mas débiles, las reglas con umbrales altos son mas robustas y confiables.

Cuando se aplica el filtro para incluir solo productos con frecuencia mínima (Leche, Pan, Mantequilla, Pañales), el número de reglas extraídas fue reducido, pero más significativas, con reglas fuertes como Pan → Mantequilla con alta confianza y lift. Sin embargo, al analizar el esquema sin filtro, el número de reglas se incrementó exponencialmente, pero muchas tenían soporte y confianza bajos, reduciendo su utilidad.

1.0.9 Punto 9

Importe el conjunto de datos credit-german.csv (repositorio de machinelearning). Discretice los atributos numéricos en máximo 5 bins de igual tamaño. Aplique el algoritmo de reglas de asociación a este conjunto. Interprete las reglas producidas. Varíe los valores de soporte y de confianza ¿Qué sucede? Interprete las reglas producidas y escoja las que en su concepto son las más interesantes para el problema (justifíque).

Solución:

Se discretizaron los valores numérics por medio de la clase `KBinsDiscretizer` de `scikit-learn`, en donde se especificó el número de bins (5), con un identificador numérico y la estrategia *uniform*; para garantizar el mismo tamaño de los bins.

Se encontró que al tener un soporte alto (ej. 20%) y confianza moderada (60%), se obtienen reglas generales y poco específicas; además de una cantidad menor de reglas. Con un soporte bajo (ej. 5%) y confianza alta (80%), se obtienen reglas detalladas, pero con 'ruido'; además de una cantidad mayor de reglas.

Para solucionar esto, se propuso un soporte del 10% y confianza del 70%, para equilibrar generalidad y fiabilidad.

Discusión

Esto termina conviriténdose como en un problema de optimización, en donde el *trade-off* consiste en ajustar la confianza y luego el soporte. Dependiendo del enfoque enmarcado del estudio, se debe dar prioridad a una de las dos. Podría ser útil, hacer uso del valor de *lift*, estableciendo un límite para descartar reglas.

[93]:
```
#pip install ucimlrepo
```

[130]:
```
from ucimlrepo import fetch_ucirepo

# fetch dataset
statlog_german_credit_data = fetch_ucirepo(id=144)

# data (as pandas dataframes)
X = statlog_german_credit_data.data.features
y = statlog_german_credit_data.data.targets

# metadata
#print(statlog_german_credit_data.metadata)

# variable information
#print(statlog_german_credit_data.variables)
```

[131]:
```
variables = statlog_german_credit_data.variables
#variables
```

[96]:
```
X
```

[96]:

	Attribute1	Attribute2	Attribute3	Attribute4	Attribute5	Attribute6	\
0	A11	6	A34	A43	1169	A65	
1	A12	48	A32	A43	5951	A61	
2	A14	12	A34	A46	2096	A61	
3	A11	42	A32	A42	7882	A61	
4	A11	24	A33	A40	4870	A61	
..	…	…	…	…	…	…	
995	A14	12	A32	A42	1736	A61	
996	A11	30	A32	A41	3857	A61	

```
997        A14          12        A32        A43         804        A61
998        A11          45        A32        A43        1845        A61
999        A12          45        A34        A41        4576        A62

    Attribute7  Attribute8 Attribute9 Attribute10  Attribute11 Attribute12  \
0          A75           4        A93        A101            4        A121
1          A73           2        A92        A101            2        A121
2          A74           2        A93        A101            3        A121
3          A74           2        A93        A103            4        A122
4          A73           3        A93        A101            4        A124
..         …          …        …          …          …          …
995        A74           3        A92        A101            4        A121
996        A73           4        A91        A101            4        A122
997        A75           4        A93        A101            4        A123
998        A73           4        A93        A101            4        A124
999        A71           3        A93        A101            4        A123

     Attribute13 Attribute14 Attribute15  Attribute16 Attribute17  \
0             67        A143        A152            2        A173
1             22        A143        A152            1        A173
2             49        A143        A152            1        A172
3             45        A143        A153            1        A173
4             53        A143        A153            2        A173
..           …         …          …          …          …
995           31        A143        A152            1        A172
996           40        A143        A152            1        A174
997           38        A143        A152            1        A173
998           23        A143        A153            1        A173
999           27        A143        A152            1        A173

     Attribute18 Attribute19 Attribute20
0              1        A192        A201
1              1        A191        A201
2              2        A191        A201
3              2        A191        A201
4              2        A191        A201
..           …         …          …
995            1        A191        A201
996            1        A192        A201
997            1        A191        A201
998            1        A192        A201
999            1        A191        A201

[1000 rows x 20 columns]
```

[132]:
```
y
```

[132]:
```
     class
0        1
1        2
2        1
3        1
4        2
..     …
995      1
996      1
997      1
998      2
999      1

[1000 rows x 1 columns]
```

[133]:
```
X_new = X.copy()
```

[134]:
```
# Extrae descripciones del dataframe 'variables'
descriptions = variables["description"].tolist()  # Lista de descripciones en␣
 ↪orden Attribute1, Attribute2, ..., Attribute20

# Renombra las columnas de X
X_new.columns = descriptions[:X_new.shape[1]]
```

[100]:
```
#X_new
```

[135]:
```
import numpy as np
import pandas as pd


# Combina features (X) y target (y) en un unico DataFrame
data = pd.concat([X_new, y], axis=1)
```

[102]:
```
#data
```

[136]:
```
#ver variables numericas
numeric_cols = data.select_dtypes(include='number').columns.tolist()
numeric_cols
```

[136]:
```
['Duration',
 'Credit amount',
 'Installment rate in percentage of disposable income',
 'Present residence since',
 'Age',
 'Number of existing credits at this bank',
 'Number of people being liable to provide maintenance for',
 'class']
```

[137]:
```
#ver variables categoricas
categorical_cols = data.select_dtypes(include='object').columns.tolist()
categorical_cols
```

[137]:
```
['Status of existing checking account',
 'Credit history',
 'Purpose',
 'Savings account/bonds',
 'Present employment since',
 'Personal status and sex',
 'Other debtors / guarantors',
 'Property',
 'Other installment plans',
 'Housing',
 'Job',
 'Telephone',
 'foreign worker']
```

[ ]:
```
from sklearn.preprocessing import KBinsDiscretizer

# Columnas numericas (excluyendo el target "class")

numeric_cols = data.select_dtypes(include='number').columns.tolist()[:-1]

#numeric_cols = data.select_dtypes(include='number').columns.tolist()

# Aplica discretización
for col in numeric_cols:
    discretizer = KBinsDiscretizer(n_bins=5, encode='ordinal',␣
 ↪strategy='uniform')
    data[col] = discretizer.fit_transform(data[[col]]).astype(int).astype(str)


# Convierte todas las columnas a formato "feature=valor"
data_str = data.astype(str).apply(lambda x: x.name + "=" + x, axis=0)
transactions = data_str.values.tolist()
```

[139]:
```
numeric_cols
```

[139]:
```
['Duration',
 'Credit amount',
 'Installment rate in percentage of disposable income',
 'Present residence since',
 'Age',
 'Number of existing credits at this bank',
 'Number of people being liable to provide maintenance for']
```

[140]:
```
data
```

[140]:

	Status of existing checking account	Duration	Credit history	Purpose	\
0	A11	0	A34	A43	
1	A12	3	A32	A43	
2	A14	0	A34	A46	
3	A11	2	A32	A42	
4	A11	1	A33	A40	
..	…	…	…	…	
995	A14	0	A32	A42	
996	A11	1	A32	A41	
997	A14	0	A32	A43	
998	A11	3	A32	A43	
999	A12	3	A34	A41	

	Credit amount	Savings account/bonds	Present employment since	\
0	0	A65	A75	
1	1	A61	A73	
2	0	A61	A74	
3	2	A61	A74	
4	1	A61	A73	
..	…	…	…	
995	0	A61	A74	
996	0	A61	A73	
997	0	A61	A75	
998	0	A61	A73	
999	1	A62	A71	

	Installment rate in percentage of disposable income	\
0	4	
1	1	
2	1	
3	1	
4	3	
..	…	
995	3	
996	4	
997	4	
998	4	
999	3	

	Personal status and sex	Other debtors / guarantors	…	Property	Age	\
0	A93	A101	…	A121	4	
1	A92	A101	…	A121	0	
2	A93	A101	…	A121	2	
3	A93	A103	…	A122	2	
4	A93	A101	…	A124	3	

```
..                      …                        …  …      …  ..
995                     A92                       A101  …     A121   1
996                     A91                       A101  …     A122   1
997                     A93                       A101  …     A123   1
998                     A93                       A101  …     A124   0
999                     A93                       A101  …     A123   0

    Other installment plans Housing Number of existing credits at this bank  \
0                       A143    A152                                      1
1                       A143    A152                                      0
2                       A143    A152                                      0
3                       A143    A153                                      0
4                       A143    A153                                      1
..                       …     …                                    …
995                     A143    A152                                      0
996                     A143    A152                                      0
997                     A143    A152                                      0
998                     A143    A153                                      0
999                     A143    A152                                      0

      Job Number of people being liable to provide maintenance for Telephone  \
0    A173                                                    0            A192
1    A173                                                    0            A191
2    A172                                                    4            A191
3    A173                                                    4            A191
4    A173                                                    4            A191
..    …                                                 …             …
995  A172                                                    0            A191
996  A174                                                    0            A192
997  A173                                                    0            A191
998  A173                                                    0            A192
999  A173                                                    0            A191

    foreign worker class
0             A201     1
1             A201     2
2             A201     1
3             A201     1
4             A201     2
..             …   …
995           A201     1
996           A201     1
997           A201     1
998           A201     2
999           A201     1

[1000 rows x 21 columns]
```

[ ]:
```
from mlxtend.preprocessing import TransactionEncoder

te = TransactionEncoder()
te_data = te.fit(transactions).transform(transactions)
df_transactions = pd.DataFrame(te_data, columns=te.columns_)
```

[142]:
```
df_transactions = pd.get_dummies(data, prefix_sep="")
```

[149]:
```
df_transactions
```

[149]:

	class	Status of existing checking accountA11	\
0	1	True	
1	2	False	
2	1	False	
3	1	True	
4	2	True	
..	…	…	
995	1	False	
996	1	True	
997	1	False	
998	2	True	
999	1	False	

	Status of existing checking accountA12	\
0	False	
1	True	
2	False	
3	False	
4	False	
..	…	
995	False	
996	False	
997	False	
998	False	
999	True	

	Status of existing checking accountA13	\
0	False	
1	False	
2	False	
3	False	
4	False	
..	…	
995	False	
996	False	
997	False	
998	False	

```
999                                    False

     Status of existing checking accountA14  Duration0  Duration1  Duration2  \
0                                      False       True      False      False
1                                      False      False      False      False
2                                       True       True      False      False
3                                      False      False      False       True
4                                      False      False       True      False
..                                       …          …          …          …
995                                     True       True      False      False
996                                    False      False       True      False
997                                     True       True      False      False
998                                    False      False      False      False
999                                    False      False      False      False

     Duration3  Duration4  …  JobA171  JobA172  JobA173  JobA174  \
0        False      False  …    False    False     True    False
1         True      False  …    False    False     True    False
2        False      False  …    False     True    False    False
3        False      False  …    False    False     True    False
4        False      False  …    False    False     True    False
..         …          …  …      …        …         …
995      False      False  …    False     True    False    False
996      False      False  …    False    False    False     True
997      False      False  …    False    False     True    False
998       True      False  …    False    False     True    False
999       True      False  …    False    False     True    False

     Number of people being liable to provide maintenance for0  \
0                                                     True
1                                                     True
2                                                    False
3                                                    False
4                                                    False
..                                                     …
995                                                   True
996                                                   True
997                                                   True
998                                                   True
999                                                   True

     Number of people being liable to provide maintenance for4  TelephoneA191  \
0                                                    False                False
1                                                    False                 True
2                                                     True                 True
3                                                     True                 True
4                                                     True                 True
```

```
..                                                  …                   …
995                                                False                True
996                                                False               False
997                                                False                True
998                                                False               False
999                                                False                True

     TelephoneA192  foreign workerA201  foreign workerA202
0             True                True               False
1            False                True               False
2            False                True               False
3            False                True               False
4            False                True               False
..             …                   …                   …
995          False                True               False
996           True                True               False
997          False                True               False
998           True                True               False
999          False                True               False

[1000 rows x 84 columns]
```

[151]:
```
from mlxtend.frequent_patterns import apriori, association_rules

# quitar momentaneamente columna class
df_transactions = df_transactions.drop(columns=["class"])

# Itemsets frecuentes (soporte minimo = 10%)
frequent_itemsets = apriori(df_transactions, min_support=0.1, use_colnames=True)

# Reglas de asociacion (confianza minima = 70%)
rules = association_rules(frequent_itemsets, metric="confidence",␣
 ↪min_threshold=0.7)
```

[ ]:
```
# Filtra reglas donde el consecuente es el target "class=good" o "class=bad"
target_rules = rules[
    rules["consequents"].astype(str).str.contains("class=1|class=0")
]
```

[152]:
```
rules.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 84217 entries, 0 to 84216
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   antecedents         84217 non-null  object
 1   consequents         84217 non-null  object
```

```
 2   antecedent support  84217 non-null  float64
 3   consequent support  84217 non-null  float64
 4   support             84217 non-null  float64
 5   confidence          84217 non-null  float64
 6   lift                84217 non-null  float64
 7   representativity    84217 non-null  float64
 8   leverage            84217 non-null  float64
 9   conviction          84217 non-null  float64
 10  zhangs_metric       84217 non-null  float64
 11  jaccard             84217 non-null  float64
 12  certainty           84217 non-null  float64
 13  kulczynski          84217 non-null  float64
dtypes: float64(12), object(2)
memory usage: 9.0+ MB
```

[160]:
```
target_rules
```

[160]:
```
Empty DataFrame
Columns: [antecedents, consequents, antecedent support, consequent support,
support, confidence, lift, representativity, leverage, conviction,
zhangs_metric, jaccard, certainty, kulczynski]
Index: []
```

[154]:
```
frequent_itemsets
```

[154]:

	support	itemsets
0	0.274	(Status of existing checking accountA11)
1	0.269	(Status of existing checking accountA12)
2	0.394	(Status of existing checking accountA14)
3	0.433	(Duration0)
4	0.394	(Duration1)
…	…	…
16312	0.105	(Credit historyA32, Other installment plansA14…
16313	0.121	(JobA173, Credit historyA32, HousingA152, Othe…
16314	0.119	(Credit historyA32, HousingA152, Other install…
16315	0.120	(JobA173, Credit historyA32, Other installment…
16316	0.103	(JobA173, HousingA152, Other installment plans…

[16317 rows x 2 columns]

[155]:
```
# Ejemplo 1: Soporte bajo (5%) y confianza alta (80%)
frequent_itemsets_low_sup = apriori(df_transactions, min_support=0.05,␣
 ↪use_colnames=True)
rules_low_sup = association_rules(frequent_itemsets_low_sup,␣
 ↪metric="confidence", min_threshold=0.8)

# Ejemplo 2: Soporte alto (20%) y confianza moderada (60%)
```

```
frequent_itemsets_high_sup = apriori(df_transactions, min_support=0.2,␣
 ↪use_colnames=True)
rules_high_sup = association_rules(frequent_itemsets_high_sup,␣
 ↪metric="confidence", min_threshold=0.6)
```

[ ]:
```
rules_low_sup
```

[ ]:
```
                                              antecedents  \
0                                                (Age=0)
1                                                (Age=0)
2                                                (Age=0)
3                                                (Age=0)
4                                                (Age=1)
…                                                  …
568872  (Credit history=A32, class=1, Telephone=A191, …
568873  (Credit amount=0, Credit history=A32, class=1,…
568874  (Credit amount=0, Credit history=A32, class=1,…
568875  (Credit amount=0, Credit history=A32, class=1,…
568876  (Credit history=A32, class=1, Number of existi…

                                              consequents  antecedent support  \
0       (Number of people being liable to provide main…                0.411
1                       (Other debtors / guarantors=A101)                0.411
2                          (Other installment plans=A143)                0.411
3                                   (foreign worker=A201)                0.411
4                       (Other debtors / guarantors=A101)                0.332
…                                                  …                  …
568872  (Credit amount=0, Number of existing credits a…                0.075
568873  (Other installment plans=A143, Number of peopl…                0.075
568874  (Other installment plans=A143, Number of exist…                0.075
568875  (Number of existing credits at this bank=0, Nu…                0.076
568876  (Credit amount=0, Number of people being liabl…                0.077

        consequent support  support  confidence      lift  representativity  \
0                    0.845    0.391    0.951338  1.125844               1.0
1                    0.907    0.369    0.897810  0.989868               1.0
2                    0.814    0.344    0.836983  1.028235               1.0
3                    0.963    0.397    0.965937  1.003050               1.0
4                    0.907    0.305    0.918675  1.012872               1.0
…                    …      …          …       …                 …
568872               0.485    0.062    0.826667  1.704467               1.0
568873               0.677    0.062    0.826667  1.221073               1.0
568874               0.501    0.062    0.826667  1.650033               1.0
568875               0.531    0.062    0.815789  1.536327               1.0
568876               0.606    0.062    0.805195  1.328704               1.0

        leverage  conviction  zhangs_metric   jaccard  certainty  kulczynski
```

```
0       0.043705    3.185250       0.189775  0.452023   0.686053    0.707030
1      -0.003777    0.910071      -0.017081  0.388830  -0.098815    0.652323
2       0.009446    1.140985       0.046620  0.390465   0.123564    0.629794
3       0.001207    1.086214       0.005162  0.406346   0.079371    0.689095
4       0.003876    1.143556       0.019024  0.326552   0.125534    0.627474
…            …           …              …         …          …           …
568872  0.025625    2.971154       0.446818  0.124498   0.663430    0.477251
568873  0.011225    1.863462       0.195728  0.089855   0.463364    0.459124
568874  0.024425    2.878846       0.425894  0.120623   0.652639    0.475210
568875  0.021644    2.546000       0.377810  0.113761   0.607227    0.466275
568876  0.015338    2.022533       0.268025  0.099839   0.505571    0.453753

[568877 rows x 14 columns]
```

[156]:
```
rules_high_sup
```

[156]:
```
                                              antecedents  \
0                (Status of existing checking accountA11)
1                (Status of existing checking accountA11)
2                (Status of existing checking accountA11)
3                (Status of existing checking accountA11)
4                (Status of existing checking accountA11)
…                                                      …
13969  (Number of people being liable to provide main…
13970  (JobA173, Other installment plansA143, Telepho…
13971  (JobA173, Number of people being liable to pro…
13972  (JobA173, Other debtors / guarantorsA101, Tele…
13973            (JobA173, Credit amount0, TelephoneA191)

                                              consequents  antecedent support  \
0                                         (Credit amount0)                0.274
1                             (Savings account/bondsA61)                0.274
2                          (Other debtors / guarantorsA101)                0.274
3                             (Other installment plansA143)                0.274
4      (Number of people being liable to provide main…                0.274
…                                                      …                   …
13969  (JobA173, Other debtors / guarantorsA101, fore…                0.343
13970  (Number of people being liable to provide main…                0.335
13971  (Other debtors / guarantorsA101, Credit amount…                0.345
13972  (Number of people being liable to provide main…                0.347
13973  (Number of people being liable to provide main…                0.311

       consequent support  support  confidence      lift  representativity  \
0                   0.738    0.202    0.737226  0.998952               1.0
1                   0.603    0.219    0.799270  1.325489               1.0
2                   0.907    0.238    0.868613  0.957677               1.0
3                   0.814    0.220    0.802920  0.986388               1.0
```

```
4                   0.845    0.222    0.810219  0.958839                1.0
…                   …      …          …       …                     …
13969               0.558    0.210    0.612245  1.097213                1.0
13970               0.554    0.210    0.626866  1.131526                1.0
13971               0.531    0.210    0.608696  1.146319                1.0
13972               0.507    0.210    0.605187  1.193663                1.0
13973               0.623    0.210    0.675241  1.083854                1.0

       leverage  conviction  zhangs_metric   jaccard  certainty  kulczynski
0     -0.000212    0.997056      -0.001444  0.249383  -0.002953    0.505470
1      0.053778    1.977782       0.338239  0.332827   0.494383    0.581227
2     -0.010518    0.707833      -0.057379  0.252386  -0.412762    0.565508
3     -0.003036    0.943778      -0.018654  0.253456  -0.059571    0.536595
4     -0.009530    0.816731      -0.055828  0.247492  -0.224394    0.536470
…           …           …              …         …          …           …
13969  0.018606    1.139895       0.134855  0.303907   0.122726    0.494294
13970  0.024410    1.195280       0.174794  0.309278   0.163376    0.502964
13971  0.026805    1.198556       0.194875  0.315315   0.165662    0.502088
13972  0.034071    1.248693       0.248458  0.326087   0.199163    0.509694
13973  0.016247    1.160861       0.112288  0.290055   0.138571    0.506160

[13974 rows x 14 columns]
```

[157]:
```
# Recomendacion: Soporte (10%) y confianza (70%)
frequent_itemsets_recom_sup = apriori(df_transactions, min_support=0.1,␣
 ↪use_colnames=True)
rules_recom_sup = association_rules(frequent_itemsets_high_sup,␣
 ↪metric="confidence", min_threshold=0.7)
```

[158]:
```
rules_recom_sup
```

[158]:
```
                                          antecedents  \
0              (Status of existing checking accountA11)
1              (Status of existing checking accountA11)
2              (Status of existing checking accountA11)
3              (Status of existing checking accountA11)
4              (Status of existing checking accountA11)
…                                                    …
9292  (JobA173, Other debtors / guarantorsA101, Othe…
9293  (JobA173, Credit amount0, Other installment pl…
9294  (JobA173, Number of people being liable to pro…
9295  (JobA173, Other debtors / guarantorsA101, Cred…
9296  (JobA173, foreign workerA201, Credit amount0, …

                                      consequents  antecedent support  \
0                                  (Credit amount0)               0.274
1                       (Savings account/bondsA61)               0.274
```

```
2                      (Other debtors / guarantorsA101)                0.274
3                         (Other installment plansA143)                0.274
4     (Number of people being liable to provide main…                0.274
…                                                    …                …
9292  (Number of people being liable to provide main…                0.298
9293  (Number of people being liable to provide main…                0.267
9294  (Other debtors / guarantorsA101, Other install…                0.278
9295  (Number of people being liable to provide main…                0.276
9296  (Number of people being liable to provide main…                0.295

      consequent support  support  confidence      lift  representativity  \
0                   0.738    0.202    0.737226  0.998952               1.0
1                   0.603    0.219    0.799270  1.325489               1.0
2                   0.907    0.238    0.868613  0.957677               1.0
3                   0.814    0.220    0.802920  0.986388               1.0
4                   0.845    0.222    0.810219  0.958839               1.0
…                       …        …           …         …                 …
9292                0.606    0.210    0.704698  1.162868               1.0
9293                0.749    0.210    0.786517  1.050089               1.0
9294                0.718    0.210    0.755396  1.052083               1.0
9295                0.677    0.210    0.760870  1.123884               1.0
9296                0.638    0.210    0.711864  1.115775               1.0

      leverage  conviction  zhangs_metric   jaccard  certainty  kulczynski
0    -0.000212    0.997056      -0.001444  0.249383  -0.002953    0.505470
1     0.053778    1.977782       0.338239  0.332827   0.494383    0.581227
2    -0.010518    0.707833      -0.057379  0.252386  -0.412762    0.565508
3    -0.003036    0.943778      -0.018654  0.253456  -0.059571    0.536595
4    -0.009530    0.816731      -0.055828  0.247492  -0.224394    0.536470
…          …           …              …         …          …           …
9292  0.029412    1.334227       0.199512  0.302594   0.250503    0.525616
9293  0.010017    1.175737       0.065075  0.260546   0.149470    0.533445
9294  0.010396    1.152882       0.068566  0.267176   0.132609    0.523937
9295  0.023148    1.350727       0.152249  0.282638   0.259658    0.535531
9296  0.021790    1.256353       0.147180  0.290456   0.204045    0.520509

[9297 rows x 14 columns]
```