

# Lista 5 IA

Daniel Salgado Magalhães - 821429

**1 - Utilizando-se o algoritmo Apriori, um suporte mínimo aceitável de 0.3 e confiança de 0.8, o número de ItemSets 1, 2, 3 e de regras a partir desta base de dados são**

Nº	Leite	Café	Cerveja	Pão	Manteiga	Arroz	Feijão
1	Não	Sim	Não	Sim	Sim	Não	Não
2	Sim	Não	Sim	Sim	Sim	Não	Não
3	Não	Sim	Não	Sim	Sim	Não	Não
4	Sim	Sim	Não	Sim	Sim	Não	Não
5	Não	Não	Sim	Não	Não	Não	Não
6	Não	Não	Não	Não	Sim	Não	Não
7	Não	Não	Não	Sim	Não	Não	Não
8	Não	Não	Não	Não	Não	Não	Sim
9	Não	Não	Não	Não	Não	Sim	Sim
10	Não	Não	Não	Não	Não	Sim	Não

**Cálculo do Suporte aceitável de pelo menos 0.3 →**

*QuantidadeSim/QuantidadeTotal*

Leite:  $2/10 = 0.2$

Café:  $3/10 = 0.3$

Cerveja:  $2/10 = 0.2$

Pão:  $5/10 = 0.5$

Manteiga:  $5/10 = 0.5$

Arroz:  $2/10 = 0.2$

Feijão:  $2/10 = 0.2$

**Valores para ItemSets baseado no cálculo de suporte aceitável, neste caso, café, pão e manteiga**

**ItemSet = 1**

Café:  $3/10 = 0.3$

Pão:  $5/10 = 0.5$

Manteiga:  $5/10 = 0.5$

**ItemSet = 2 (quando os dois são sim no mesmo número)**

Café e Pão:  $3/10 = 0.3$

Café e Manteiga:  $3/10 = 0.3$

Pão e Manteiga:  $4/10 = 0.4$

**ItemSet = 3 (quando os três são sim no mesmo número)**

Café, Pão e Manteiga:  $3/10 = 0.3$

**Cálculo da Confiança de pelo menos 0.8  $\rightarrow X/Y$**

X = quando todos os itens são sim no mesmo número

Y = quantidade suporte do primeiro item, no caso, o da esquerda

**ItemSet = 1**  $\rightarrow$  deve haver pelo menos dois itens

**ItemSet = 2**

Café  $\rightarrow$  Pão =  $3/3 = 1$

**Pão  $\rightarrow$  Café =  $3/5 = 0.6$**

Café  $\rightarrow$  Manteiga =  $3/3 = 1$

**Manteiga  $\rightarrow$  Café =  $3/5 = 0.6$**

Pão  $\rightarrow$  Manteiga =  $4/5 = 0.8$

Manteiga  $\rightarrow$  Pão =  $4/5 = 0.8$

**ItemSet = 3**

Café  $\rightarrow$  Pão e Manteiga =  $3/3 = 1$

**Pão  $\rightarrow$  Café e Manteiga =  $3/5 = 0.6$**

**Manteiga  $\rightarrow$  Café e Pão =  $3/5 = 0.6$**

Café e Pão  $\rightarrow$  Manteiga =  $3/3 = 1$

Café e Manteiga  $\rightarrow$  Pão =  $3/3 = 1$

**Pão e Manteiga  $\rightarrow$  Café =  $3/4 = 0.75$**

**Importante: No exercício pede confiança a partir de 0.8, sendo assim, os itens marcados em azul não serão levados em consideração no cálculo de Lift.**

### Cálculo de Lift

X = confiança dos itens

Y = quantidade suporte do segundo item, no caso, o da direita

Café → Pão

Café → Manteiga

Pão → Manteiga

Manteiga → Pão

Café → Pão e Manteiga

Café e Pão → Manteiga

Café e Manteiga → Pão

Regra	Confiança	Suporte(B)	Lift
Café → Pão	$3/3 = 1$	0.5	$1/0.5 = 2$
Café → Manteiga	$3/3 = 1$	0.5	$1/0.5 = 2$
Pão → Manteiga	$4/5 = 0.8$	0.5	$0.8/0.5 = 1.6$
Manteiga → Pão	$4/5 = 0.8$	0.5	$0.8/0.5 = 1.6$
Café → Pão e Manteiga	$3/3 = 1$	0.4	$1/0.4 = 2.5$
Café e Pão → Manteiga	$3/3 = 1$	0.5	$1/0.5 = 2$
Café e Manteiga → Pão	$3/3 = 1$	0.5	$1/0.5 = 2$

**2 - Considerando-se o código que está em Módulos/Apriori.ipynb, rode o código com a base acima e confira os resultados.**

Rodando o código com a base sem realizar alterações, o resultado é esse:

Podemos ordenar estas regras por uma métrica desejada

```
[43] RegrasFinais.sort_values(by="lift", ascending=False)
```

	Antecedente	Consequente	suporte	confiança	lift
6	[Cafe]	[Pao, Manteiga]	0.3	1.00	2.5
11	[Pao, Manteiga]	[Cafe]	0.3	0.75	2.5
0	[Cafe]	[Manteiga]	0.3	1.00	2.0
1	[Manteiga]	[Cafe]	0.3	0.60	2.0
2	[Cafe]	[Pao]	0.3	1.00	2.0
3	[Pao]	[Cafe]	0.3	0.60	2.0
7	[Manteiga]	[Pao, Cafe]	0.3	0.60	2.0
8	[Pao]	[Cafe, Manteiga]	0.3	0.60	2.0
9	[Cafe, Manteiga]	[Pao]	0.3	1.00	2.0
10	[Pao, Cafe]	[Manteiga]	0.3	1.00	2.0
4	[Manteiga]	[Pao]	0.4	0.80	1.6
5	[Pao]	[Manteiga]	0.4	0.80	1.6

É possível perceber que a quantidade de regras aumenta devido à uma confiança menos restrita. No código original, a confiança é definida como 0.6 ao invés de 0.8, que faz com que o número de regras seja 12, ao invés de 7.

### 3 - Considerando-se o código que está em Módulos/Apriori.ipynb, altere-o para que ele imprima os itemsets gerados, com os respectivos suportes

Alterando essa linha "regras = apriori(transacoes, min\_support = 0.3, min\_confidence = 0.6)", para essa "regras = apriori(transacoes, min\_support = 0.3, min\_confidence = 0.8)", faz com que a confiança mínima seja igual a pedida no exercício 1, e sendo assim, é possível ver os mesmos resultados.

Podemos ordenar estas regras por uma métrica desejada

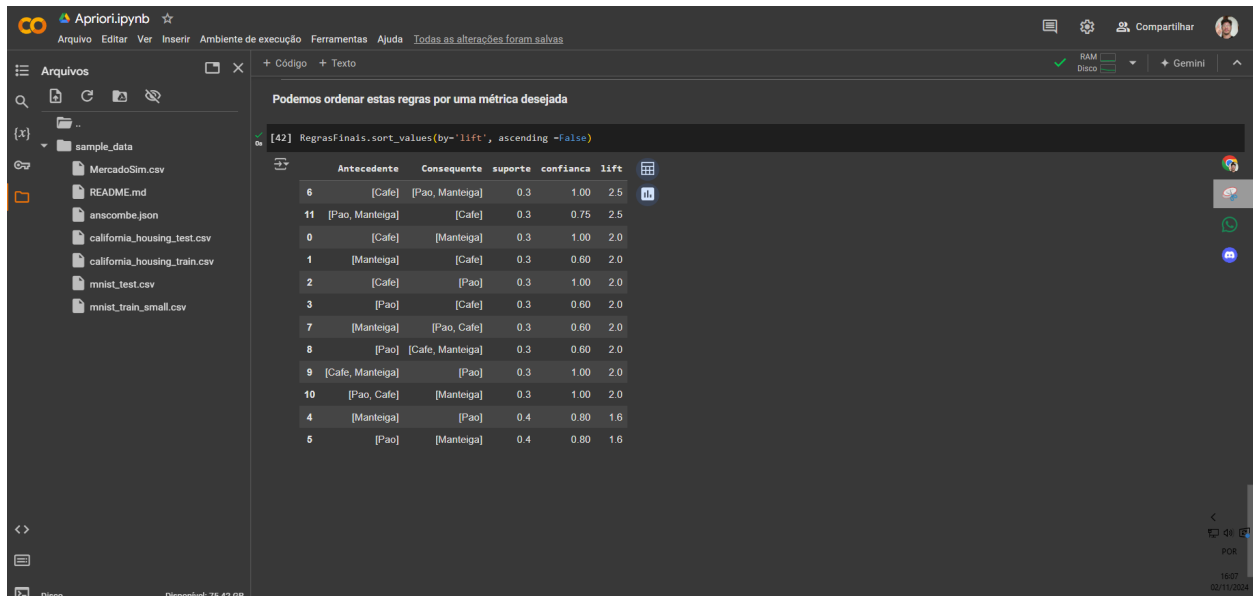
```
[29] RegrasFinais.sort_values(by="lift", ascending=False)
```

	Antecedente	Consequente	suporte	confiança	lift
4	[Cafe]	[Pao, Manteiga]	0.3	1.0	2.5
0	[Cafe]	[Manteiga]	0.3	1.0	2.0
1	[Cafe]	[Pao]	0.3	1.0	2.0
5	[Cafe, Manteiga]	[Pao]	0.3	1.0	2.0
6	[Pao, Cafe]	[Manteiga]	0.3	1.0	2.0
2	[Manteiga]	[Pao]	0.4	0.8	1.6
3	[Pao]	[Manteiga]	0.4	0.8	1.6

### 4 - Considerando-se o código que está em Módulos/Apriori.ipynb, altere-o para que ele gere regras de associação quando não há presença do produto. Ou seja, gostaria de ver regras da seguinte forma:

- Quem não leva álcool leva detergente;
- Quem não leva detergente leva arroz, etc

Não achei a base de dados de álcool e detergente, por isso realizei com a base de dados do mercado. Os resultados mostrados levaram a um aumento nas regras.



Podemos ordenar estas regras por uma métrica desejada

```
[42] RegrasFinais.sort_values(by='lift', ascending=False)
```

	Antecedente	Consequente	suporte	confiança	lift
6	[Cafe]	[Pao, Manteiga]	0.3	1.00	2.5
11	[Pao, Manteiga]	[Cafe]	0.3	0.75	2.5
0	[Cafe]	[Manteiga]	0.3	1.00	2.0
1	[Manteiga]	[Cafe]	0.3	0.60	2.0
2	[Cafe]	[Pao]	0.3	1.00	2.0
3	[Pao]	[Cafe]	0.3	0.60	2.0
7	[Manteiga]	[Pao, Cafe]	0.3	0.60	2.0
8	[Pao]	[Cafe, Manteiga]	0.3	0.60	2.0
9	[Cafe, Manteiga]	[Pao]	0.3	1.00	2.0
10	[Pao, Cafe]	[Manteiga]	0.3	1.00	2.0
4	[Manteiga]	[Pao]	0.4	0.80	1.6
5	[Pao]	[Manteiga]	0.4	0.80	1.6

## 5 - Investigue o funcionamento da biblioteca mlxtend para geração de regras de associação

<https://github.com/rasbt/mlxtend> Links to an external site.

[https://github.com/rasbt/mlxtend/blob/master/mlxtend/frequent\\_patterns/apriori.py](https://github.com/rasbt/mlxtend/blob/master/mlxtend/frequent_patterns/apriori.py)

A biblioteca mlxtend é utilizada para tarefas de aprendizado de máquina e mineração de dados, incluindo a geração de regras de associação com o método Apriori, conforme apresentado.

O módulo "frequent patterns" da mlxtend fornece classes que facilitam a extração de padrões frequentes e regras de associação em grandes conjuntos de dados. O código apriori.py, é uma classe desse módulo. O código possui uma estrutura geral formada por 3 funções principais e uma função de suporte, para que a função apriori consiga calcular os valores necessários.

Função generate\_new\_combinations: Gera novas combinações de itens para o próximo passo do algoritmo Apriori.

Função `generate_new_combinations_low_memory`: Similar à função anterior, mas otimizada para uso em grandes conjuntos de dados, usando menos memória.

Função `apriori`: A função principal que coordena a execução do algoritmo Apriori para identificar os conjuntos frequentes.

Função `_support`: É a função suporte que atua internamente para calcular o suporte dos conjuntos de itens.

Neste caso, a função principal é a mais importante, no caso a função `apriori`, que tem as seguintes estruturas:

- **Variáveis principais para realizar as operações:**

- `df`: DataFrame no formato one-hot encoding, onde cada coluna representa um item, e cada linha representa um valor binário
- `min_support`: suporte mínimo binário necessário para considerar um conjunto de itens frequente.
- `use_colnames`: se é verdade, usa os nomes das colunas do DataFrame nos resultados, ao invés dos índices.
- `max_len`: tamanho máximo dos conjuntos de itens a serem gerados.
- `verbose`: nível de detalhe das mensagens de progresso.
- `low_memory`: se é verdade, ativa o modo de baixa memória, que é mais lento, mas consome menos memória.

- **Processo que acontece na iteração do algoritmo:**

1. Verifica o suporte mínimo.
2. Verifica se o data frame está compactado, se não está, converte ele para a forma apropriada.
3. Calcula o suporte inicial dos itens individuais.
4. Aumenta o tamanho dos conjuntos de itens iterativamente, adicionando novos itens e verificando o suporte dos novos conjuntos.
5. Filtra os conjuntos de itens que não atingem o suporte mínimo e armazena aqueles que atingem.
6. Ao final, retorna um DataFrame com os conjuntos de itens e seus suportes.

Com essa função, juntando com as outras informadas, conseguimos realizar a verificação de itens, como por exemplo o exemplo do supermercado, usado para verificar itens que são comprados juntos.

## **6 - Faça uma resenha do artigo "A comprehensive review of visualization methods for association rule" que está no CANVAS**

O artigo "A comprehensive review of visualization methods for association rule mining" é focado na análise sobre as técnicas de visualização para mineração de regras de associação. A mineração de regras de associação (Association Rule Mining, em inglês, que é referida como ARM) é uma técnica de mineração de dados que identifica relações significativas entre itens em grandes conjuntos de dados transacionais, sendo aplicada em áreas como análise de mercado e diagnóstico médico. A visualização dos dados é realizada nesse processo para facilitar o entendimento dos usuários sobre os padrões encontrados.

O artigo divide as abordagens de visualização em métodos tradicionais, como gráficos de dispersão, "twokey plot", diagramas de matriz e gráficos em mosaico e "double decker plot", e as novas ideias nas regras de visualização das associações, incluindo diagramas de Ishikawa, representação molecular, mapas de metrô, diagramas de Sankey e "glyph-based plots". Cada método é descrito com às suas características, vantagens e limitações. Os métodos tradicionais são eficazes para conjuntos menores de regras, enquanto os métodos inovadores permitem uma exploração mais detalhada e interativa dos dados, podendo representar grandes volumes de informações de forma mais acessível e intuitiva.

Além de descrever esses métodos, o artigo também propõe uma taxonomia para classificar as técnicas de visualização de acordo com vários aspectos, focando em algumas perguntas, que são: "Como visualizar?", "Quais métodos de visualização usar?", "Quais características das regras de associação são essenciais para visualizar?", "O que visualizar?" e "Quais os atributos devem ser mostrados?". Após descrever cada uma das perguntas e como elas podem ser usadas no sistema, os autores demonstram os sistemas de visualização dos ARM e discutem os desafios e as lacunas atuais nesses métodos, como a necessidade de maior interatividade e escalabilidade nas ferramentas de visualização.