Aprendizagem Não Supervisionado

November 25, 2025

0.1 Aprendizado Não Supervisionado

Aprendizado Não Supervisionado é um ramo do aprendizado de máquina que se destaca por sua capacidade de lidar com dados não rotulados, ao contrário da aprendizagem supervisionada, que depende de dados previamente categorizados. Esse tipo de aprendizado se torna uma ferramenta poderosa para análise exploratória de dados, permitindo a descoberta de padrões e relações intrínsecas aos dados, sem a necessidade de conhecimento prévio sobre seu significado.

0.1.1 Funcionamento da Aprendizado Não Supervisionado:

O processo se inicia com a análise de dados não rotulados, buscando identificar padrões e relações. Como os dados não possuem categorias ou resultados predefinidos, o algoritmo precisa encontrar essas estruturas por conta própria. Essa tarefa, embora desafiadora, pode revelar ideias valiosas sobre os dados, que poderiam passar despercebidos em conjuntos de dados rotulados.

Imagine um shopping que coleta dados de seus clientes, como compras e informações demográficas. Através da Aprendizado Não Supervisionado, o shopping pode agrupar clientes com base em seus hábitos de compra, sem a necessidade de rotular previamente cada cliente.

0.1.2 Características dos Dados:

Os dados utilizados na Aprendizado Não Supervisionado geralmente são:

- **Não estruturados:** Podem conter informações ruidosas, valores ausentes ou dados desconhecidos.
- **Não rotulados:** Apresentam apenas valores para os parâmetros de entrada, sem um valor alvo (saída). Esse tipo de dado é mais fácil de coletar em comparação com os dados rotulados.

0.1.3 Algoritmos de Aprendizado Não Supervisionado:

Existem três principais categorias de algoritmos utilizados em conjuntos de dados não supervisionados:

1. **Agrupamento (Clustering):** Consiste em agrupar dados não rotulados em clusters com base em suas similaridades. O objetivo é identificar padrões e relações nos dados sem conhecimento prévio sobre seu significado. Um exemplo prático é a segmentação de clientes, onde o algoritmo agrupa clientes com comportamentos de compra semelhantes.
    - **Algoritmos comuns de agrupamento:**
        - Clusterização K-means
        - Clustering Hierárquico
        - Clustering baseado em densidade (DBSCAN)

  - Agrupamento de deslocamento médio
  - Agrupamento espectral
2. **Aprendizagem de Regras de Associação:** Busca descobrir relações entre parâmetros em grandes conjuntos de dados. Essa técnica é frequentemente utilizada na análise de cesta de compras, ajudando a entender a relação entre a venda de diferentes produtos. Um exemplo clássico é a relação entre a compra de leite e a probabilidade do cliente comprar pão ou ovos.
   - **Algoritmos comuns de aprendizagem de regras de associação:**
     - Algoritmo Apriori
     - Algoritmo FP-Growth
     - Algoritmo Eclat
     - Algoritmos Eficientes Baseados em Árvores
3. **Redução de Dimensionalidade:** Reduz o número de recursos em um conjunto de dados, preservando o máximo de informação possível. Essa técnica é útil para melhorar o desempenho de algoritmos de aprendizado de máquina e para visualização de dados.
   - **Exemplos de algoritmos de redução de dimensionalidade:**
     - Análise de Componentes Principais (PCA)
     - Análise Discriminante Linear (LDA)
     - Fatoração de matriz não negativa (NMF)
     - Incorporação Linear Local (LLE)
     - Isomap

0.1.4 Desafios da Aprendizado Não Supervisionado:

Apesar de seus benefícios, a Aprendizado Não Supervisionado apresenta alguns desafios:

- **Avaliação:** A ausência de rótulos ou categorias predefinidas dificulta a avaliação do desempenho dos algoritmos.
- **Interpretabilidade:** Compreender o processo de tomada de decisão dos modelos pode ser desafiador.
- **Overfitting:** Os algoritmos podem se ajustar excessivamente ao conjunto de dados de treinamento, limitando sua capacidade de generalização para novos dados.
- **Qualidade dos Dados:** Dados ruidosos ou incompletos podem levar a resultados imprecisos.
- **Complexidade Computacional:** Alguns algoritmos, especialmente aqueles que lidam com dados de alta dimensão ou grandes conjuntos de dados, podem ser computacionalmente caros.

0.1.5 Vantagens da Aprendizado Não Supervisionado:

- **Dispensa dados rotulados:** A coleta de dados rotulados pode ser cara e demorada, o que torna a Aprendizado Não Supervisionado uma alternativa atraente.
- **Revela padrões ocultos:** Os algoritmos podem identificar padrões e relações que não seriam óbvios para humanos.
- **Versatilidade:** Pode ser utilizada em uma variedade de tarefas, como agrupamento, redução de dimensionalidade e detecção de anomalias.
- **Exploração de novos dados:** Permite obter insights valiosos a partir de dados novos, o que pode não ser possível com outros métodos.

0.1.6 Desvantagens da Aprendizado Não Supervisionado:

- **Dificuldade de avaliação:** A ausência de rótulos ou categorias predefinidas torna a avaliação do desempenho dos algoritmos um desafio.
- **Dificuldade de interpretação:** A compreensão do processo de tomada de decisão dos modelos pode ser complexa.
- **Sensibilidade à qualidade dos dados:** Resultados imprecisos podem ocorrer com dados ruidosos ou incompletos.
- **Custo computacional:** Alguns algoritmos podem demandar alto poder de processamento.

0.1.7 Aplicações da Aprendizado Não Supervisionado:

As aplicações da Aprendizado Não Supervisionado são diversas e abrangem diferentes áreas:

- **Segmentação de clientes:** Agrupamento de clientes com base em seus dados demográficos, comportamento ou preferências.
- **Detecção de fraude:** Identificação de transações financeiras que se desviam dos padrões esperados.
- **Sistemas de recomendação:** Recomendação de itens a usuários com base em seu comportamento ou preferências anteriores.
- **Processamento de linguagem natural (PLN):** Modelagem de tópicos, agrupamento de documentos e marcação de classes gramaticais.
- **Análise de imagem:** Segmentação de imagem, detecção de objetos e reconhecimento de padrões de imagem.

A Aprendizado Não Supervisionado se apresenta como uma ferramenta poderosa para análise de dados, com a capacidade de revelar padrões ocultos e gerar insights valiosos. A compreensão de seus algoritmos, vantagens e desvantagens é fundamental para sua aplicação eficaz em diferentes áreas.

0.2 K-means

O algoritmo K-means é uma técnica de aprendizado de máquina não supervisionado utilizada para **agrupar dados não rotulados em clusters**, com base em suas similaridades. Em outras palavras, o K-means busca dividir os dados em grupos, onde os elementos dentro de cada grupo são mais parecidos entre si do que com elementos de outros grupos.

Imagine um conjunto de pontos dispersos em um gráfico. O algoritmo K-means tentará encontrar **os K melhores centros de clusters**, chamados de **centroides**, para que os pontos ao redor de cada centro sejam o mais próximos possível uns dos outros.

0.2.1 Como o K-means Funciona?

O K-means opera de forma iterativa, buscando **minimizar a distância entre os pontos de dados e os centroides dos clusters**. O processo se divide em etapas:

1. **Inicialização:** O algoritmo seleciona aleatoriamente **K pontos** no espaço dos dados. Esses pontos representam os **centroides iniciais dos clusters**.
2. **Atribuição:** Cada ponto de dado é **atribuído ao cluster cujo centro (centroide) está mais próximo**, utilizando a **distância euclidiana** como medida de proximidade.

3. **Atualização:** Os centroides dos clusters são **recalculados**, sendo a **média de todos os pontos de dados atribuídos a cada cluster**.
4. **Repetição:** As etapas 2 e 3 são repetidas até que os centroides dos clusters se estabilizem, ou seja, **não apresentem mudanças significativas**.

Exemplo: Imagine um conjunto de dados com informações sobre clientes de um shopping, como idade, renda e frequência de visitas. O K-means poderia ser aplicado para agrupá-los em diferentes categorias, como "clientes VIP", "clientes regulares" e "clientes ocasionais", com base em seus padrões de comportamento.

0.2.2 Encontrando o Número Ideal de Clusters (K)

A escolha do número de clusters (K) é uma etapa fundamental na aplicação do K-means e pode influenciar significativamente os resultados. Um método comumente utilizado para determinar o K ideal é o **método do cotovelo ("Elbow Method")**.

O método do cotovelo consiste em executar o K-means para diferentes valores de K e calcular a **soma dos quadrados das distâncias entre os pontos de dados e seus respectivos centroides (SSE)**. Ao plotar o SSE em função de K, busca-se um ponto de inflexão no gráfico, semelhante a um cotovelo, que indica um valor de K onde o ganho em termos de redução do SSE se torna marginal.

As fontes fornecem um exemplo prático de como utilizar o método do cotovelo em Python para encontrar o número ideal de clusters.

0.2.3 Vantagens e Desvantagens do K-means

O K-means é um algoritmo relativamente simples e eficiente, o que o torna uma escolha popular para diversas aplicações.

Vantagens:

- **Simplicidade:** Fácil de entender e implementar.
- **Eficiência:** Geralmente mais rápido que outros algoritmos de agrupamento, especialmente para grandes conjuntos de dados.
- **Forte acoplamento:** Cria uma forte relação entre os pontos de dados dentro de um cluster.

Desvantagens:

- **Definição prévia de K:** A necessidade de definir o número de clusters (K) antes da execução pode ser uma limitação.
- **Sensibilidade à inicialização:** Resultados podem variar dependendo da escolha inicial dos centroides.
- **Dificuldade com clusters de diferentes tamanhos e formas:** Pode ter problemas para identificar clusters com características distintas.
- **Sensível a ruídos:** Outliers (ruídos) nos dados podem afetar negativamente os resultados.
- **Não fornece informações sobre a qualidade dos clusters**.

0.2.4 Aplicações do K-means

O K-means encontra aplicações em diversas áreas, incluindo:

- **Segmentação de clientes:** Agrupar clientes com base em seus dados demográficos, comportamento de compra ou preferências.
- **Detecção de fraudes:** Identificar transações financeiras suspeitas que se desviam dos padrões esperados.
- **Sistemas de recomendação:** Sugerir produtos ou serviços aos usuários com base em seus interesses e histórico de interações.
- **Processamento de linguagem natural (PNL):** Agrupar documentos por tópicos, realizar agrupamento de documentos e identificar classes gramaticais.
- **Análise de imagens:** Segmentar imagens, detectar objetos e reconhecer padrões visuais.

0.3 Entendendo o Agrupamento Hierárquico: Uma Abordagem Detalhad

O agrupamento hierárquico, também conhecido como clusterização hierárquica, é um método de **aprendizado de máquina não supervisionado** que busca construir uma hierarquia de clusters, ou seja, uma estrutura em forma de árvore que representa as relações entre os grupos de dados. A ideia por trás dessa técnica é que **pontos de dados próximos uns dos outros são mais similares ou relacionados do que pontos de dados mais distantes**.

0.3.1 Tipos de Agrupamento Hierárquico

Existem duas abordagens principais para o agrupamento hierárquico:

- **Agrupamento Aglomerativo:** Também conhecido como abordagem "bottom-up", o agrupamento aglomerativo inicia o processo tratando **cada ponto de dado como um cluster individual**. Em seguida, os clusters mais similares são **combinados iterativamente**, formando clusters maiores, até que todos os pontos de dados estejam em um único cluster.
- **Agrupamento Divisivo:** Conhecido como abordagem "top-down", o agrupamento divisivo começa com **todos os pontos de dados em um único cluster**. Em seguida, esse cluster é **dividido recursivamente** em subclusters, até que cada ponto de dado esteja em um cluster individual.

0.3.2 Dendrograma: Visualizando a Hierarquia de Clusters

A representação visual da hierarquia de clusters gerada pelo agrupamento hierárquico é chamada de **dendrograma**. O dendrograma é uma estrutura em forma de árvore que mostra as relações entre os clusters em diferentes níveis de granularidade.

- **Base do Dendrograma:** Representa os pontos de dados individuais.
- **Topo do Dendrograma:** Representa o cluster único que contém todos os pontos de dados.
- **Ramos do Dendrograma:** Mostram a fusão (agrupamento aglomerativo) ou divisão (agrupamento divisivo) dos clusters.

A altura dos ramos no dendrograma indica a distância ou dissimilaridade entre os clusters. Ramos mais altos indicam maior dissimilaridade, enquanto ramos mais baixos indicam maior similaridade.

0.3.3 Determinando o Número Ideal de Clusters

O dendrograma fornece uma visão geral da estrutura hierárquica dos clusters, permitindo a **identificação do número ideal de clusters**. Para determinar esse número, podemos analisar a altura

em que os ramos do dendrograma formam clusters distintos. Cortar o dendrograma nessa altura resultará no número desejado de clusters.

0.3.4 Calculando a Distância Entre Clusters

A fusão ou divisão dos clusters no agrupamento hierárquico é baseada em uma medida de **distância ou similaridade entre os clusters**. Existem diferentes métodos para calcular essa distância, sendo alguns dos mais comuns:

- **Distância Mínima:** Calcula a menor distância entre quaisquer dois pontos dos clusters.
- **Distância Máxima:** Calcula a maior distância entre quaisquer dois pontos dos clusters.
- **Distância Média:** Calcula a distância média entre todos os pares de pontos dos clusters.
- **Método de Ward:** A similaridade entre dois clusters é baseada no aumento do erro quadrático quando os clusters são mesclados.

0.3.5 Comparando Agrupamento Aglomerativo e Divisivo

- **Complexidade:** O agrupamento divisivo é mais complexo de implementar, pois requer um método de agrupamento plano, como o K-means, para dividir os clusters recursivamente.
- **Eficiência:** O agrupamento divisivo pode ser mais eficiente se não precisarmos gerar uma hierarquia completa até os pontos de dados individuais. A complexidade de tempo do agrupamento aglomerativo pode ser otimizada com o uso de estruturas de dados adequadas.
- **Precisão:** O agrupamento divisivo pode ser mais preciso, pois leva em consideração a distribuição global dos dados ao tomar decisões de particionamento de alto nível.

0.3.6 Implementação do Agrupamento Hierárquico

As fontes fornecem um exemplo de código em Python utilizando a biblioteca scikit-learn para implementar o agrupamento hierárquico. O código demonstra como:

- Gerar um conjunto de dados de exemplo.
- Realizar o agrupamento hierárquico utilizando o método de Ward para calcular a distância entre os clusters.
- Plotar o dendrograma.

0.3.7 Aplicações do Agrupamento Hierárquico

O agrupamento hierárquico encontra aplicações em diversas áreas, como:

- **Biologia:** Classificação de espécies e análise de dados genéticos.
- **Marketing:** Segmentação de clientes e análise de mercado.
- **Processamento de imagens:** Segmentação de imagens e reconhecimento de padrões.
- **Análise de texto:** Agrupamento de documentos e modelagem de tópicos.

0.4 DBSCAN: Agrupando Dados Baseado em Densidade

O DBSCAN (Density-Based Spatial Clustering of Applications with Noise), ou **Agrupamento Espacial Baseado em Densidade de Aplicações com Ruído**, é um algoritmo de aprendizado de máquina não supervisionado que **agrupa pontos de dados com base em sua densidade**.

A ideia principal é que os pontos de dados em um cluster estejam próximos uns dos outros (alta densidade) e sejam separados de outros clusters por regiões de baixa densidade.

0.4.1 Por que Usar o DBSCAN?

Algoritmos de agrupamento como o K-means e o agrupamento hierárquico funcionam bem para encontrar clusters esféricos ou convexos, ou seja, clusters compactos e bem separados. No entanto, esses algoritmos têm dificuldades em lidar com:

- **Clusters de formas arbitrárias:** Dados reais podem conter clusters com formas complexas e não convexas.
- **Dados com ruído:** A presença de outliers (ruídos) pode afetar negativamente o desempenho desses algoritmos.

O DBSCAN se destaca nesses cenários, pois consegue identificar clusters de **formas arbitrárias e é robusto à presença de ruído**.

0.4.2 Funcionamento do DBSCAN: Parâmetros e Tipos de Pontos

O algoritmo DBSCAN utiliza dois parâmetros principais:

- **eps ( ):** Define o raio de vizinhança em torno de um ponto de dado. Pontos dentro desse raio são considerados vizinhos.
  - Um valor muito pequeno de `eps` pode levar à identificação de muitos outliers.
  - Um valor muito grande de `eps` pode resultar na fusão de clusters distintos.
- **MinPts:** Define o número mínimo de pontos de dados necessários dentro do raio `eps` para que um ponto seja considerado um ponto central (core point).
  - Geralmente, o valor de `MinPts` deve aumentar com o tamanho do conjunto de dados e a dimensionalidade dos dados.

Com base nesses parâmetros, o DBSCAN classifica os pontos de dados em três tipos:

- **Ponto Central (Core Point):** Um ponto é considerado central se tiver pelo menos `MinPts` vizinhos dentro do raio `eps`.
- **Ponto de Borda (Border Point):** Um ponto de borda tem menos de `MinPts` vizinhos dentro do raio `eps`, mas está dentro do raio `eps` de um ponto central.
- **Ruído (Noise) ou Outlier:** Um ponto que não é um ponto central nem um ponto de borda é considerado ruído.

0.4.3 Etapas do Algoritmo DBSCAN

O DBSCAN segue as seguintes etapas para agrupar os dados:

1. **Encontrar Pontos Centrais:** O algoritmo identifica todos os pontos centrais, marcando-os como visitados.
2. **Criar Clusters:** Para cada ponto central não atribuído a um cluster, um novo cluster é criado.
3. **Expandir Clusters:** O algoritmo expande o cluster recursivamente, adicionando todos os pontos de dados conectados por densidade ao ponto central inicial.
   - **Conexão por Densidade:** Dois pontos são considerados conectados por densidade se houver um caminho de pontos centrais entre eles, onde cada ponto está dentro do raio `eps` do ponto anterior.

4. **Classificar Ruído:** Os pontos de dados que não pertencem a nenhum cluster são classificados como ruído.

0.4.4 Implementando o DBSCAN em Python

As fontes fornecem um exemplo de implementação do DBSCAN em Python utilizando a biblioteca `scikit-learn`. O código demonstra como:

- Criar um conjunto de dados de exemplo.
- Ajustar o modelo DBSCAN aos dados.
- Visualizar os clusters resultantes.

0.4.5 Métricas de Avaliação para DBSCAN

Existem diferentes métricas para avaliar o desempenho do DBSCAN, como:

- **Coeficiente de Silhueta:** Mede a similaridade entre os pontos dentro de um cluster em comparação com outros clusters. Um valor próximo a 1 indica clusters bem definidos e separados.
- **Índice Rand Ajustado:** Mede a similaridade entre a estrutura de clusters encontrada pelo algoritmo e a estrutura real dos dados (se conhecida). Um valor próximo a 1 indica alta similaridade.

[ ]:

0.5 Algoritmo Apriori: Desvendando Padrões Frequentes em Conjuntos de Dados

O algoritmo Apriori, nomeado por sua utilização de conhecimento prévio ("a priori") sobre propriedades de conjuntos de itens frequentes, é uma técnica fundamental em mineração de dados, especificamente na descoberta de **regras de associação**. Desenvolvido por R. Agrawal e R. Srikant em 1994, ele tem como objetivo identificar **conjuntos de itens que aparecem juntos com frequência em um conjunto de dados**, como produtos comprados juntos em um supermercado ou páginas da web visitadas em sequência.

0.5.1 Princípios Fundamentais:

O Apriori se baseia na **propriedade Apriori**, que afirma que:

- **Se um conjunto de itens é frequente, todos os seus subconjuntos não vazios também são frequentes.**
- **Consequentemente, se um conjunto de itens é infrequente, todos os seus superconjuntos também serão infrequentes.**

Essa propriedade permite ao algoritmo **reduzir o espaço de busca**, evitando a análise de conjuntos de itens que já se sabe serem infrequentes.

0.5.2 Como o Algoritmo Apriori Funciona:

O Apriori opera em um processo iterativo, **aumentando gradualmente o tamanho dos conjuntos de itens** analisados:

1. **Fase de Varredura:** O algoritmo percorre o conjunto de dados para **contar a frequência de cada item individual**, criando um conjunto de **candidatos a itens frequentes (C1)**. Itens que não atingem o **suporte mínimo** definido são descartados. O suporte mínimo é um limite predefinido que representa a **frequência mínima** com que um conjunto de itens deve aparecer no conjunto de dados para ser considerado frequente.
2. **Geração de Conjuntos Candidatos:** A partir dos itens frequentes encontrados na etapa anterior, o algoritmo **gera novos conjuntos candidatos** com um item adicional (C2, C3, etc.). A geração de candidatos se baseia na **junção de conjuntos de itens frequentes**, garantindo que **todos os subconjuntos** dos novos candidatos também sejam frequentes (propriedade Apriori).
3. **Poda de Conjuntos Infrequentes:** O algoritmo **elimina conjuntos candidatos que não atendem ao suporte mínimo**, realizando uma nova varredura no conjunto de dados para contar sua frequência.
4. **Iteração:** As etapas 2 e 3 são repetidas, **aumentando o tamanho dos conjuntos de itens** analisados a cada iteração, até que **nenhum novo conjunto candidato frequente seja encontrado**.

0.5.3 Exemplo Prático:

Imagine um conjunto de dados de transações de um supermercado. O objetivo é **descobrir quais produtos são frequentemente comprados juntos**. Definindo um suporte mínimo de 3 (ou seja, um conjunto de itens precisa aparecer em pelo menos 3 transações para ser considerado frequente), o Apriori poderia encontrar conjuntos de itens frequentes como:

- `{Cerveja}`
- `{Fraldas}`
- `{Cerveja, Fraldas}`

0.5.4 Geração de Regras de Associação:

Após a identificação dos conjuntos de itens frequentes, o algoritmo **gera regras de associação** a partir deles. Uma regra de associação é uma expressão do tipo **"Se {A}, então {B}"**, onde {A} e {B} são conjuntos de itens. A **confiança** de uma regra de associação é a probabilidade de que {B} esteja presente em uma transação que contém {A}.

No exemplo do supermercado, a partir do conjunto frequente `{Cerveja, Fraldas}`, poderíamos gerar regras como:

- `Se {Cerveja}, então {Fraldas}`
- `Se {Fraldas}, então {Cerveja}`

A confiança de cada regra seria calculada com base na frequência do conjunto `{Cerveja, Fraldas}` em relação à frequência de {Cerveja} e {Fraldas} individualmente.

0.5.5 Limitações do Algoritmo Apriori:

- **Custo Computacional:** O Apriori pode ser **computacionalmente caro**, especialmente para conjuntos de dados grandes com muitos itens frequentes, pois **gera um grande número de conjuntos candidatos** a cada iteração.

- **Múltiplas Varreduras do Conjunto de Dados:** O algoritmo precisa **percorrer o conjunto de dados várias vezes** para contar a frequência dos conjuntos candidatos, o que pode ser **demorado para grandes conjuntos de dados**.
- **Sensibilidade ao Suporte Mínimo:** A escolha do **suporte mínimo** pode impactar significativamente o número de conjuntos de itens frequentes encontrados. Um valor **muito baixo** pode levar à geração de um grande número de regras irrelevantes, enquanto um valor **muito alto** pode deixar de identificar padrões importantes.

0.5.6 Aplicações do Algoritmo Apriori:

Apesar de suas limitações, o Apriori é uma técnica amplamente utilizada em diversas áreas:

- **Análise de Mercado (Market Basket Analysis):** Identificar produtos frequentemente comprados juntos, permitindo otimizar a disposição de produtos em lojas, criar promoções combinadas e direcionar campanhas de marketing.
- **Sistemas de Recomendação:** Recomendar produtos ou serviços a clientes com base em seus históricos de compras ou navegação.
- **Análise de Navegação Web:** Descobrir padrões de navegação de usuários em sites, permitindo otimizar a organização do conteúdo e a usabilidade do site.
- **Bioinformática:** Identificar padrões em sequências genéticas, como genes que são frequentemente expressos juntos.

O algoritmo Apriori, embora com suas limitações, fornece uma base sólida para a descoberta de regras de associação e continua sendo uma ferramenta valiosa em mineração de dados. Sua aplicação em diversos domínios demonstra sua capacidade de extrair insights relevantes a partir de dados, permitindo a tomada de decisões estratégicas.

0.5.7 Intuição do Algoritmo Apriori

Para entender de forma intuitiva o que o algoritmo Apriori faz, vamos trabalhar com uma amostra de um conjunto de dados de transações em uma loja de conveniência fictícia. Cada linha da amostra contém uma transação identificada por um ID. Se a coluna referente ao produto (Cerveja, Fralda, Chiclete, Refrigerante e Salgadinho) estiver marcada com 1, significa que esse produto foi comprado nessa transação; se estiver marcada com 0, significa que não foi comprado.

Amostra do Conjunto de Dados Para ilustrar, veja a transação com ID igual a 4. Nessa transação, foram comprados os itens cerveja, fralda, refrigerante e salgadinho.

ID	Cerveja	Fralda	Goma	Refrigerante	Lanche
1	1	1	1	1	0
2	1	1	0	0	0
3	1	1	1	0	1
4	1	1	0	1	1
5	0	1	0	1	0
6	0	1	0	0	0
7	0	1	0	0	0
8	0	0	0	1	1
9	0	0	0	1	1

Regras de Associação Com isso entendido, podemos definir o que é uma Regra de Associação. Trata-se de um método para explorar relações entre itens em conjuntos de dados. Vamos definir alguns termos importantes para as Regras de Associação:

- **I (Itens)**: Conjunto dos seus n atributos {i_1, i_2, ..., i_n}.
- **D (Database)**: Conjunto das m transações {t_1, t_2, ..., t_m}.
- Toda transação t_i é única em D e consiste em um subconjunto dos Itens I.

Vamos definir uma Regra de Associação como a relação (X => Y), onde X e Y são subconjuntos de I. Eles não podem ter nenhum elemento em comum. X é chamado de antecedente e Y de consequência da Regra.

Com isso definido, considere a transação de ID igual a 2. Podemos ter a seguinte Regra de Associação: {Cerveja} => {Fralda}. Essa é a Regra de Associação que define que, nessa transação, quem comprou cerveja também comprou fralda.

Nesse ponto, se você começar a montar as Regras de Associação na sua cabeça, perceberá que mesmo para um conjunto de dados pequeno, existem muitas regras. O desafio agora é encontrar um modo de selecionar as regras relevantes. O método utilizado para encontrar tais regras envolve o cálculo de medidas como Suporte, Confiança, Lift e Conviction.

Embora o Apriori use apenas o Suporte, falarei um pouco dessas quatro medidas para maior entendimento do problema que estamos tentando resolver.

1. **Support (Suporte)**

A medida que indica a proporção de X em D.

$$\text{Supp}(X) = \frac{\#X \text{ em } D}{\#D}$$

Ou seja, Supp({Cerveja}) = 4/9.

2. **Confidence (Confiança)**

A medida de Confiança é calculada em cima de uma Regra (X => Y). Ela expressa a proporção de "Se X for comprado, qual a chance de Y ser comprado?"

$$\text{Conf}(X \Rightarrow Y) = \frac{\text{Supp}(X \cup Y)}{\text{Supp}(X)}$$

Ou seja, Conf({Cerveja} => {Chiclete}) = (2/9) / (4/9) = 1/2.

Isso nos diz que 50% das vezes que cerveja é comprada, chiclete também é comprado.

0.5.8 Lift (Alavancagem)

A medida de Lift indica a probabilidade de Y ser comprado se X for comprado, considerando a popularidade geral de Y. Em outras palavras, ela nos ajuda a entender se a compra de X realmente aumenta a chance de Y ser comprado, além do que seria esperado pela popularidade de Y sozinho.

$$\text{Lift}(X \Rightarrow Y) = \frac{\text{Supp}(X \cup Y)}{\text{Supp}(X) \times \text{Supp}(Y)}$$

No caso da regra ({Cerveja} => {Chiclete}), temos:

$$\text{Lift}(Cerveja \Rightarrow Chiclete) = \frac{2/9}{(4/9) \times (2/9)} = \frac{2/9}{8/81} = \frac{2 \times 81}{9 \times 8} = \frac{162}{72} = 2/3$$

Com base nesse valor, podemos concluir:

- Se Lift(X => Y) > 1, então é provável que Y seja comprado quando X for comprado.
- Se Lift(X => Y) 1, então NÃO é provável que Y seja comprado quando X for comprado.

0.5.9 Conviction (Convicção)

A medida de Conviction calcula a frequência com que X ocorre sem que Y ocorra, ou seja, está interessada em quando a regra falha.

$$\text{Conv}(X \Rightarrow Y) = \frac{1 - \text{Supp}(Y)}{1 - \text{Conf}(X \Rightarrow Y)}$$

Essa medida varia entre [0, ∞]. Se Conf(X => Y) for igual a 1, o denominador da fórmula é zero e o resultado da Conviction é infinito. Se Supp(Y) for igual a 1, ou seja, Y está presente em todas as transações, então Conv é igual a 0 — você nunca erra.

Vamos para um exemplo:

Considere X = {Refrigerante, Salgadinho} e Y = {Cerveja}.

Primeiro, calculamos Supp(Y) = 4/9. Depois, calculamos Conf(X => Y) = 3/9.

$$\text{Conv}(X \Rightarrow Y) = \frac{(9/9) - (4/9)}{(9/9) - (3/9)} = \frac{5/9}{6/9} = \frac{5}{6}$$

Certifique-se de que você compreendeu bem as medidas antes de ir para a próxima seção!

0.5.10 Algoritmo Apriori

Com o entendimento das Regras de Associação e de como interpretá-las, chegou a hora de entender o Apriori e como ele nos ajuda nesse processo de mineração de dados.

O Apriori trabalha com o conceito de itens frequentes, que são os itens do seu conjunto I que têm a pontuação de suporte maior que um limiar (threshold). Ou seja, precisamos calcular o suporte de todas as combinações de itens e extrair um subconjunto de itens frequentes. Sabendo disso, os passos do Apriori são:

1. Dentro dos itens I, extraia um subconjunto (I_freq) dos itens que têm o seu suporte maior que o threshold.
2. Dentro de I_freq, itere para formar as combinações dos I_freq com I, aplique o threshold e acumule em I_freq.
3. Pare quando, ao aplicar o threshold, nenhum item sobrar.

Por exemplo, considere o nosso conjunto de dados de amostra e threshold = 0.4:

1. Vamos calcular o suporte para grupos de 1 item:

	Suporte	Conjunto de itens	Comprimento
0	0,4444	(cerveja)	1
1	0,7777	(fralda)	1
2	0,2222	(goma)	1
3	0,5555	(refrigerante)	1
4	0,4444	(lanche)	1

2. Agora removemos os Items com Support menor que o threshold, ou seja, sai o Chiclete (note que esse é o primeiro I_freq):

	Suporte	Conjunto de itens	Comprimento
0	0,4444	(cerveja)	1
1	0,7777	(fralda)	1
2	0,5555	(refrigerante)	1
3	0,4444	(lanche)	1

3. Depois disso, realizamos o mesmo cálculo pra os pares de Itens, como por exemplo (fralda, cerveja)

	Suporte	Conjunto de itens	Comprimento
0	0,4444	(fralda,cerveja)	2
1	0,2222	(goma,cerveja)	2
2	0,2222	(refrigerante,cerveja)	2
3	0,2222	(lanche,cerveja)	2
4	0,2222	(goma,fralda)	2
5	0,3333	(refrigerante,fralda)	2
6	0,2222	(lanche,fralda)	2
7	0,1111	(goma,refrigerante)	2
8	0,1111	(goma,lanche)	2
9	0,3333	(refrigerante,lnche)	2

4. Aplicando o threshold, sobra apenas (Diaper, Beer), que devemos acumular em I_freq:

	Suporte	Conjunto de itens	Comprimento
0	0,4444	(fralda,cerveja)	2

Veja o I_freq do momento:

	Suporte	Conjunto de itens	Comprimento
0	0,4444	(cerveja)	1
1	0,7777	(fralda)	1
2	0,2222	(goma)	1
3	0,5555	(refrigerante)	1

	Suporte	Conjunto de itens	Comprimento
4	0,4444	(lanche)	1
0	0,4444	(fralda,cerveja)	2

5. Calculamos agora para 3 items:

	Suporte	Conjunto de itens	Comprimento
0	0,2222	(goma,fralda,cerveja)	2
1	0,2222	(refrigerante,fralda,cerveja)	2
2	0,2222	(lanche,fralda,cerveja)	2
3	0,1111	(goma,refrigerante,cerveja)	2
4	0,1111	(goma,lanche,cerveja)	2
5	0,1111	(refrigerante,lanche,cerveja)	2
6	0,1111	(goma,fralda,refrigerante)	2
7	0,1111	(goma,fralda,lanche)	2
8	0,1111	(refrigerante,fralda,lanche)	2
9	0,0000	(goma,lanche,refrigerante)	2

6. Após aplicar o threshold, percebemos que nenhum Item sobrou, logo o algoritmo para, visto que esse é o critério de parada.

	Suporte	Conjunto de itens	Comprimento
0	0,4444	(cerveja)	1
1	0,7777	(fralda)	1
2	0,2222	(goma)	1
3	0,5555	(refrigerante)	1
4	0,4444	(lanche)	1
0	0,4444	(fralda,cerveja)	2

Pronto, encontramos as Regras de Associação mais frequentes.

0.6 Algoritmo FP-Growth

Contexto da Mineração de Padrões Frequentes:

Imagine um supermercado com milhares de transações de compra. Cada transação é uma lista de itens que um cliente comprou. A mineração de padrões frequentes visa descobrir **relações interessantes entre os itens**, como "clientes que compram cerveja também tendem a comprar fraldas". Essas informações podem ser valiosas para **tomada de decisões de marketing, gerenciamento de estoque e personalização da experiência do cliente**.

O Problema do Algoritmo Apriori:

Um dos primeiros algoritmos para mineração de padrões frequentes foi o **Apriori**. Ele funciona gerando conjuntos de itens candidatos e verificando sua frequência no banco de dados. No entanto, esse processo pode ser **computacionalmente caro**, especialmente para conjuntos de dados

grandes, pois envolve **múltiplas varreduras do banco de dados** e a **geração de muitos conjuntos candidatos desnecessários**.

A Solução do FP-Growth:

O algoritmo FP-Growth surge como uma **alternativa mais eficiente ao Apriori**, contornando suas deficiências. Ele utiliza uma estrutura de dados chamada **FP-Tree (Frequent Pattern Tree)** para armazenar os padrões frequentes de forma compacta e eficiente.

Construindo a Árvore FP-Tree:

1. **Contagem de Frequência:** O primeiro passo é contar a frequência de cada item individual no conjunto de dados.
2. **Ordenação e Filtragem:** Os itens são ordenados em ordem decrescente de frequência. Itens com frequência abaixo de um limite predefinido (**suporte mínimo**) são descartados.
3. **Construção da Árvore:** A FP-Tree é construída inserindo as transações no banco de dados. Os itens em cada transação são percorridos em ordem decrescente de frequência.
   - Se um item já existe na árvore, seu contador de frequência é incrementado.
   - Caso contrário, um novo nó é criado para o item, com um contador de frequência inicializado em 1.
4. **Conexões entre Nós:** Os nós da árvore são conectados de forma a **representar os padrões de coocorrência entre os itens**. Cada caminho da raiz até uma folha representa um padrão frequente.

As duas principais desvantagens do Algoritmo Apriori são:

1. Em cada etapa, conjuntos de candidatos precisam ser construídos.
2. Para construir os conjuntos candidatos, o algoritmo precisa escanear repetidamente o banco de dados.

Essas duas propriedades inevitavelmente tornam o algoritmo mais lento. Para superar essas etapas redundantes, um novo algoritmo de mineração de regras de associação foi desenvolvido, chamado Frequent Pattern Growth Algorithm. Ele supera as desvantagens do algoritmo Apriori ao armazenar todas as transações em uma Estrutura de Dados Trie. Considere os seguintes dados:

ID da transação	Items
T1	{E,K,M,N,O,Y}
T2	{D,E,K,N,O,Y}
T3	{A,E,K,M}
T4	{C,K,M,U,Y}
T5	{C,E,I,K,O,O}

Os dados acima são um conjunto de dados hipotético de transações com cada letra representando um item. A frequência de cada item individual é computada:

Item	Freqüência
A	1
C	2
D	1

Item	Freqüência
E	4
I	1
K	5
M	3
N	2
O	4
U	1
Y	3

Seja o suporte mínimo 3. Um conjunto de Padrão Frequente é construído, o qual conterá todos os elementos cuja frequência é maior ou igual ao suporte mínimo. Esses elementos são armazenados em ordem decrescente de suas respectivas frequências. Após a inserção dos itens relevantes, o conjunto L se parece com isto:

L = {K : 5, E : 4, M : 3, O : 4, Y : 3}

Agora, para cada transação, o respectivo conjunto item ordenADO é construído. Isso é feito iterando o conjunto Frequent Pattern e verificando se o item atual está contido na transação em questão. Se o item atual estiver contido, o item é inserido no conjunto Ordered-Item para a transação atual. A tabela a seguir é construída para todas as transações:

ID da transação	Itens	COnjunto de itens ordenado
T1	{E,K,M,N,O,Y}	{K,E,M,O,Y}
T2	{D,E,K,N,O,Y}	{K,E,O,Y}
T3	{A,E,K,M}	{K,E,M}
T4	{C,K,M,U,Y}	{K,M,Y}
T5	{C,E,I,K,O,O}	{K,E,O}

Agora, todos os conjuntos de itens ordenados são inseridos em uma estrutura de dados Trie.

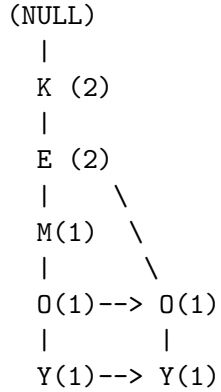
0.6.1 a) Inserindo o conjunto {K, E, M, O, Y}:

Aqui, todos os itens são simplesmente vinculados um após o outro na ordem de ocorrência no conjunto e inicializam a contagem de suporte para cada item como 1.

```
(NULL)
  |
  K (1)
  |
  E (1)
  |
  M (1)
  |
  O (1)
  |
  Y (1)
```

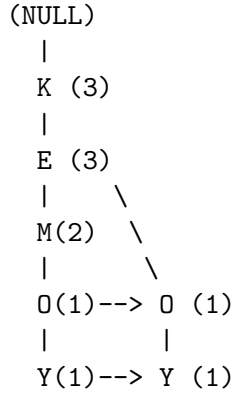
0.6.2 b) Inserindo o conjunto {K, E, O, Y}:

Até a inserção dos elementos K e E, simplesmente a contagem de suporte é aumentada em 1. Ao inserir O, podemos ver que não há ligação direta entre E e O, portanto, um novo nó para o item O é inicializado com a contagem de suporte como 1 e o item E é vinculado a este novo nó. Ao inserir Y, primeiro inicializamos um novo nó para o item Y com a contagem de suporte como 1 e vinculamos o novo nó de O com o novo nó de Y.



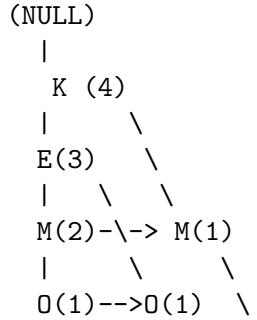
0.6.3 c) Inserindo o conjunto {K, E, M}:

Aqui simplesmente a contagem de suporte de cada elemento é aumentada em 1.



0.6.4 d) Inserindo o conjunto {K, M, Y}:

Semelhante à etapa b), primeiro a contagem de suporte de K é aumentada, então novos nós para M e Y são inicializados e vinculados adequadamente



```
|      |      \
Y(1)-->Y(1)-->Y(1)
```

0.6.5 e) Inserindo o conjunto {K, E, O}:

Aqui simplesmente as contagens de suporte dos respectivos elementos são aumentadas. Note que a contagem de suporte do novo nó do item O é aumentada.

```
(NULL)
  |
   K (5)
  |     \
  E(4)   \
  |   \   \
  M(2)-\-> M(1)
  |     \     \
  O(1)-->O(2)  \
  |      |      \
  Y(1)-->Y(1)-->Y(1)
```

Agora, para cada item, a Base de Padrão Condicional é computada, que são rótulos de caminho de todos os caminhos que levam a qualquer nó do item dado na árvore de padrões frequentes. Observe que os itens na tabela abaixo são organizados na ordem crescente de suas frequências.

Item	Base de Padrão Condicional
Y	{{K,E,M,O:1},{K,E,O:1},{K,M:1}}
O	{{K,E,M:1},{K,E:2}}
M	{{K,E:2},{K:1}}
E	{K:4}
K	

Agora, para cada item, a Àrvore de Padrões Condicionais Frequentes é construída. Isso é feito pegando o conjunto de elementos que é comum em todos os caminhos na Base de Padrão Condicional daquele item e calculando sua contagem de suporte somando as contagens de suporte de todos os caminhos na Conditional Pattern Base.

Item	Base de Padrão Condicional	Arvore de Padrões Condicionais Frequentes
Y	{{K,E,M,O:1},{K,E,O:1},{K,M:1}}	{K:3}
O	{{K,E,M:1},{K,E:2}}	{K,E:3}
M	{{K,E:2},{K:1}}	{K:3}
E	{K:4}	{K:4}
K		

A partir da Àrvore de Padrões Condicionais Frequentes, as regras de Padrões Frequentes são geradas pareando os itens do conjunto de Árvore de Padrões Frequentes Condicionais com o item correspondente, conforme fornecido na tabela abaixo. | Item | Regras de Padrões Frequentes |

|:—-:|:———————-:| | Y | {<K,Y:3>} | | O | {<K,O:3>,<E,O:3>,<E,K,O:3>} | | M | {<K,M:3>} | | E | {<E,K:3>} | | K | |

Para cada linha, dois tipos de regras de associação podem ser inferidos, por exemplo, para a primeira linha que contém o elemento, as regras **K -> Y** e **Y -> K** podem ser inferidas. Para determinar a regra válida, a confiança de ambas as regras é calculada e aquela com confiança maior ou igual ao valor mínimo de confiança é retida.

Extraindo Padrões Frequentes da FP-Tree:

A partir da FP-Tree, a extração de padrões frequentes se torna uma tarefa simples. Percorrendo a árvore, **cada caminho da raiz até um nó (incluindo o nó)** representa um padrão frequente. A frequência do padrão é dada pelo contador de frequência do último nó no caminho.

Vantagens do FP-Growth:

- **Eficiência:** A estrutura da FP-Tree permite a **identificação rápida de padrões frequentes sem gerar conjuntos candidatos**, tornando o algoritmo mais eficiente que o Apriori, principalmente para conjuntos de dados grandes.
- **Compactação:** A FP-Tree armazena os padrões de forma compacta, **otimizando o uso de memória**.
- **Escalabilidade:** O FP-Growth escala bem para conjuntos de dados com um grande número de transações e itens.

Aplicações Práticas:

O FP-Growth tem diversas aplicações práticas em áreas como:

- **Análise de Mercado:** Descobrir padrões de compra de clientes para **segmentar ofertas, otimizar o layout de lojas e personalizar a experiência de compra**.
- **Recomendação de Produtos:** Identificar produtos frequentemente comprados em conjunto para **sistemas de recomendação**.
- **Detecção de Fraude:** Encontrar padrões suspeitos em transações financeiras para **identificar possíveis fraudes**.
- **Bioinformática:** Descobrir padrões em sequências genéticas ou dados de expressão gênica.

O FP-Growth é uma ferramenta poderosa para a descoberta de padrões frequentes em grandes conjuntos de dados. Sua eficiência, compactação e escalabilidade o tornam uma escolha popular em diversas aplicações, impulsionando a tomada de decisões estratégicas e a descoberta de insights valiosos.

[ ]: