Centro Univertário Senac
São José do Rio Preto

Carlos Eduardo Segundo
Eduardo de Freitas Gandorphi
Johny William de Oliveira Alves

Proposta para aplicação de Regras de Associação para varejos online

São José do Rio Preto
2020

Carlos Eduardo Segundo
Eduardo de Freitas Gandorphi
Johny William de Oliveira Alves

Proposta para aplicação de Regras de Associação para varejos online

Trabalho de Conclusão de Curso
apresentado ao Centro Universitário
SENAC São José do Rio Preto, como
exigência parcial para obtenção do
Grau de Especialista em Big Data

São José do Rio Preto

Carlos Eduardo Segundo
Eduardo de Freitas Gandorphi
Johny William de Oliveira Alves

Proposta para aplicação de Regras de Associação para varejos online

Trabalho de Conclusão de Curso
apresentado ao Centro Universitário
SENAC São José do Rio Preto, como
exigência parcial para obtenção do
Grau de Especialista em Big Data

Orientador: Prof.º. Dr. João Marcelo Rondina

A comissão avaliadora dos Trabalhos de Conclusão de Curso,

considerou os candidatos:

[ ] Aprovados [ ] Não Aprovados

Prof.º. Dr. João Marcelo Rondina

Resumo

O presente documento tem a intenção de demonstrar a utilização da ferramenta Apriori no que diz respeito a sugestão de cesta de compras. O algoritmo em questão tem o poder de identificar as relações entre as compras de forma a agrupar quais produtos têm a tendência de serem comprados juntos e os produtos que são comprados de forma independente. Antes de atingir esse objetivo, é descrito como o Apriori funciona nas suas bases e como será aplicado. Além disso a obtenção dos dados, limpeza e seleção desses registros também é um passo importante para que seja feito uma leitura assertiva da situação. Por fim, baseado numa massa de dados muito grande, teremos um cenário real onde será sugerido qual o produto com maior potencial de venda para um outro produto que já esteja no carrinho.

Abstract

The presented document will demonstrate how Apriori algorithm works and how it suggests products due to the market basket. Apriori is capable of identifying relations between products that usually are bought together and the ones that are bought individually. In order to do so, this study will show how Apriori was designed, its strength and how it was intended to be used in a way it can shine the most. Before putting hands on that algorithm, it is needed to have a massive data set to work with. The data set is essential to obtain reliable results that can lead to meaningful decisions. The bigger the data is, the better the outcome will be. Once you are loaded with all that information, it is needed to clean, organiz and select the data that will be relevant to that algorithm. By the end of this document, based on all that data that will be processed as a daily situation, the result will be the suggestion of products with higher probability of being bought by the customer based on his/her market basket.

1. Introdução

Pessoas são guiadas por hábitos e experiências, antigamente um dado demográfico poderia falar quase tudo de uma pessoa. Os tempos foram mudando e nessa nova era em que a informação está disponível de forma fácil e a qualquer momento de forma *online*, as pessoas estão criando rotinas que muitas vezes divergem de seus dados demográficos, acabam por se relacionarem pelos grupos de identidade ou hábitos.

1.1. Motivação

Reconhecer os hábitos de consumo das pessoas pode ser uma grande ferramenta para aumentar o ticket médio de uma venda ou ganhar novos mercados, como descrito por Duhigg (2012) no caso do analista de dados da Target, Andrew Pole, com o vasto histórico de compras dos clientes, em conjunto do uso de dados de cartões de crédito, cupons de descontos e o *tracking* realizado pelo site, identificava os clientes e seus hábitos de consumo, sendo capaz de aumentar o volume de vendas e otimizar seu processo de logística.

Nos Estados Unidos a gigante do varejo Amazon, conseguiu substituir seu time de editores que utilizava para indicar livros para seus clientes, por estudo de afinidade dos títulos, por um algoritmo que analisa o histórico de compras (PROVOST, 2016).

1.2. Aprendizado de máquina supervisionado

Com o histórico de compras dos usuários, juntamente com outros dados como hábitos de acesso, demográficos, formas de pagamento entre outros, podemos ensinar o computador como prever o comportamento futuro deste cliente. Também é possível prever os hábitos de vendas de determinados cliente e produtos; nas palavras de Provost (2016) “Se um alvo específico pode ser fornecido, o problema pode ser formulado como um problema supervisionado”, tendo a movimentação dos produtos contendo um conjunto de quais foram vendidos na mesma sacola, fornecendo essa associação para gerar um modelo de predição para quais produtos recomendar para os clientes no momento da compra.

1.3. Regras de Associação

Para entender melhor como funciona os estudos de correlações e associações;

> A existência de associações ou a correlação entre os atributos implica que eles frequentemente aparecem juntos em uma transação ou que uma variação na frequência de observação de um atributo num conjunto de transações ocorre com uma variação na frequência de um segundo atributo nesse mesmo conjunto de transações (SILVA, PERES e BOSCARIOLI, 2017)

Exemplificando a citação acima podemos utilizar o cenário seguinte, em uma grande rede de supermercados as pessoas que normalmente compram fraldas e chupetas, também costumam comprar pomadas e lenços umedecidos. Dessa maneira nas lojas físicas seria indicado deixar esses itens próximos uns dos outros para que o consumidor não se esqueça de colocar na sua cesta de compras. Já em uma loja virtual, poderia ser disponibilizados a vista do usuários esses outros itens para que possa ser anexado a compra da mesma forma.

De forma geral, as associações demonstram que certos produtos costumam ser comprados juntos. Levando essa teoria como base o algoritmo *Apriori* foi criado para otimizar essas correlações de uma forma um pouco mais sofisticada. Esse algoritmo usa os conceitos de *Support*, *Confidence*, *Lift* como apresentado por GARG (2018) para prever quais itens serão comprados juntos.

- ***Support*** - coeficiente matemático que representa a quantidade de casos em que o item ‘A’ e o item ‘B’ aparecem juntos em relação ao total de casos. Este coeficiente mostra o quão provável A e B aparecem juntos com relação ao todo;
- ***Confidence*** - coeficiente matemático que representa a quantidade de casos em que o item ‘A’ e o item ‘B’ aparecem juntos em relação aos casos que aparece somente o item ‘A’. Este coeficiente mostra o quão provável ‘A’ aparece junto com ‘B’ de forma relacionada;
- ***Lift*** - coeficiente matemático que relaciona a *Confidence* ao número de vezes que o item ‘B’ aparece sozinho. Esse coeficiente descreve de forma sucinta a chance de ‘A’ e ‘B’ aparecem juntos com relação a ‘A’ com relação a ‘B’.

De maneira geral, tais conceitos apresentam relações e cálculos simplistas entre dois itens. Entretanto esses mesmos cálculos serão aplicados de forma extensiva a todos os casos da massa de dados que será submetida. Uma vez que todo o processamento for feito, é possível tabular e priorizar as associações mais prováveis.

É importante ressaltar que muitas dessas associações serão consideradas óbvias ou repetidas. Nesse momento deve ser feito uma filtragem e limpeza deixando

apenas as associações pertinentes ao objetivo da exploração. Por fim é visto os casos de produtos que normalmente são comprados juntos mas que não são notados.

Uma última ressalva deve ser feita, estas associações estão diretamente ligadas a massa de dados submetida. Quanto maior a massa de dados e maior a variedade, mais assertiva ela tende a ser. De maneira que as descobertas feitas podem não ser verdade para um outro mercado, outra região ou outros itens.

2. Materiais e Métodos

Foi escolhido como ferramenta para o desenvolvimento desse estudo a linguagem de programação Python. Ferramenta essa escolhida em virtude de ser de acesso livre e com largo volume de bibliotecas especializada em manipulação de listas e métodos estatísticos. O Python é uma linguagem de programação muito utilizada em meios acadêmicos pela sua simplicidade, didática e alto valor agregado. Como citado a cima, ela traz consigo muitas bibliotecas prontas para uso imediato facilitando dessa forma a criação de novos estudos baseado em matérias anteriores.

A biblioteca para manipulação de dados tabulados escolhido foi a Pandas pois já que esta possui grande aceitação no mercado pela sua alta aderência devido a facilidade de uso e inúmeros fóruns de resolução de dúvidas.

2.1. Algoritmo de Associação

Após análise de mercado, o algoritmo que está mais consolidado e a mais tempo no mercado é o Algoritmo Apriori. Foi criado em 1994 por AGRAWAL SRIKANT.

O algoritmo de mineração de dados Apriori é utilizado na mineração de regras de associação em grandes massas de dados, encontrando todos os conjuntos de itens frequentes. A partir da mineração dos dados, o algoritmo Apriori identifica todas as regras de associação relevantes entre os itens.

Passo a passo do funcionamento do algoritmo:

- **Primeiro passo:** encontrar quais itens são semelhantes, mais frequentes e os contam;
- **Segundo passo:** Separar os itens mais relevantes, usando a contagem, descarta os menos relevantes, ou seja, retira os de menor valor, seguindo um parâmetro previamente definido. Ao concluir esse passo, é gerado um primeiro arquivo, o qual vamos chamar de Item Mestre**;**
- **Terceiro Passo:** A partir do Item Mestre, é feita uma leitura em toda a base de dados, gerando assim um novo arquivo que será o de itens potenciais a serem ofertados.

3. Desenvolvimento

Neste setor apresentamos o desenvolvimento do trabalho fazendo uso das ferramentas e conceitos apresentados na introdução, descrevendo a estruturação do ciclo de vida para apresentação dos dados e um estudo de caso para demonstrar o uso do framework desenvolvido.

Organizados com a estrutura de pastas para facilitar revisar o processo e dar resiliência ao projeto, pois alguns processos são demorados e pesados, podendo gerar problemas na execução, mas eles podem ser retomados a qualquer momento.

Tabela 1 - Estrutura de diretórios com detalhamento da finalidade.

Caminho	**Descrição**
/data	Arquivos de dados
/data/raw	Arquivos brutos vindos da extração
/data/splitted	Arquivos brutos divididos por linhas
/data/clustered	Agrupamento de produtos por clientes
/data/separated	Separados por arquivos contendo determinado produto
/utils	Programas auxiliares para processamento
/code	Programas para processamento da análise

Fonte: elaborado pelos autores

Como apresentado (tabela 1) usamos uma pasta para armazenar os dados separados para cada estado dos mesmos, mantendo os originais após cada transformação.

3.1. Coleta

A coleta de dados foi realizada em arquivos com a extensão CSV (Comma Separated Value) arquivos para dados estruturados e com esquema com texto tabulado por uso de algum caractere, no padrão americano vírgulas para o padrão nacional ponto e vírgula, mantendo a atenção pois no padrão brasileiro usamos vírgulas para separar decimais de números, ponto para milhares.

Dados migrados para armazenados na pasta “./data/raw” foram organizados (tabela 2) com a identificação de pessoas e produtos juntos com seus valores de venda

Tabela 2 - Dicionário de dados para as informações coletadas.

Nome	**Descrição**
CD_PESSOA	Identificação do cliente
QT_ITEM	Quantidade de produtos comprados
QT_PONTO	Quantidade pontos ganhos na compra do produto
CD_PRODUTO	Código do produto
DC_PRODUTO	Descrição detalhada
VL_VENDA	Valor de vendas do produto
CD_FAMILIA_PRODUTO	Código da identificação da família do produto
CD_LINHA_PRODUTO	Código da identificação da linha do produto

Fonte: elaborado pelos autores

Os arquivos foram separados em arquivos de 10 mil linhas (anexo 2) armazenados na pasta “/data/splitted” para facilitar o processamento

3.2. Transformação

Fizemos a escolha de realizar uma associação simples com a aplicação do Apriori apenas cruzando as relações de produtos com clientes. Para isso removemos os campos não utilizados e mantemos uma pequena lista de produtos selecionados aleatoriamente (tabela 3).

Tabela 3 - Relação de código de produtos selecionados.

2559	7248	28411	57996	58002	58006	58416
62529	69815	70825	70989	71663	72148	73440
74244	85400	86727	86944	86969	89073	92529
92536	92569	92598	92788	92798	92806	

Depois de escolher o escopo da análise, organizamos os arquivos com uma listagem de produtos relacionados por produtos contidos no mesmo arquivo, gerando inconsistência caso um cliente tenha sido separado no processo de divisão das linhas, e aplicação desse processo (anexo 3) foi armazenado para "/data/clustered". Posteriormente separamos essas presenças por produtos no caso o selecionado para análise o 2559 (anexo 4).

3.3. Apuração dos Resultados

Para realizar a análise realizamos a aplicação do algoritmo Apriori com as filtragens de no mínimo 0.3 de *support* e 0.6 de *confidence* limitando somente relações binárias outros produtos (anexo 5) como todas as linhas analisadas possuem o produto 2559, ele está em 100% dos casos buscamos, os produtos que aparecem em 30% ou mais das transações e os produtos que aparecem 60% ou mais das transações em relação a outros produtos.

Esses valores de parâmetros foi ajustado por tentativa e erro até apresentar uma quantidade satisfatória de resultados (tabela 4)

Tabela 4 - Relação de associação com o produto 2559.

Produto	***Support***	***Confidence***	***Lift***
72148	0,8	0,8	1,0
73440	0,8	0,8	1,0
86727	0,4	1,0	1,0
92536	0,4	1,0	1,0

Fonte: elaborado pelos autores

Neste caso para um cliente que comprou o item 2559 seria aconselhável indicar o produto 72148 ou 73440 pois ambos aparecem em 80% das vendas de 2559.

4. Conclusão

Aplicação do algoritmo é uma ótima maneira de compreender o hábito de compra para determinados usuários. Com esse algoritmo é notório quais produtos são vendidos de forma independente, quais são vendidos em conjunto com outros e quais são vendidos para perfis específicos e por isso têm baixa relevância. Isso não significa que se deve simplesmente ignorar os produtos com baixa saída ou com baixa influência, pois a análise feita foi simplesmente baseada em frequência e afinidade, não levando em consideração o peso financeiro, logístico ou alinhamento do marca. É muito importante salientar que o presente documento coloca em evidência a possibilidade de se usar um agrupamento e indicação de produtos para melhor o *ticket* médio da compra, e que em nenhum momento foi posto que tais produtos, que não estão nesse conjunto de importância, devem ser esquecidos ou retirados da loja. Uma outra análise pode ser feita em um segundo momento para pontuar a real viabilidade.

Por fim, é explícito que algoritmos de cesta de compras destaca o conjunto-venda e categoriza produtos, dessa forma é possível afirmar que quando se tem em mãos uma grande quantidade de informações de um determinado negócio deve sim ser aplicado tais mecanismos de aprendizado com o intuito de entender melhor os hábitos de compras daqueles clientes e no futuro afinar o planejamento de vendas.

Referências

DA SILVA, Leandro Augusto; PERES, Sarajane Marques; BOSCARIOLI, Clodis. **Introdução à Mineração de Dados: Com Aplicações em R**. Editora GEN LTC. Capítulo 5: Regras de Associação. 2017;

DUHIGG, Charles. **O poder do hábito: Por que fazemos o que fazemos na vida e nos negócios**. 1ª edição Rio de Janeiro: Objetiva, p. 195-225, 2012;

GARG, Anisha; **Complete guide to Association Rules**; Towards Data Science; Disponível em <https://towardsdatascience.com/association-rules-2-aa9a77241654>; Setembro de 2018;

PROVOST, Foster, FAWCETT, Tom. **Data Science para Negócios**. Rio de Janeiro, RJ: Alta Books, 2016;

Anexos

Anexo 1 - Extração dos dados

```
Foi gerado uma consulta SQL a uma base de dados de experimental liberada para consulta com dados ficticios.
```

Fonte: elaborado pelos autores

Anexo 2 - O arquivo */utils/split-files.py* na pasta *utils* para fragmentação dos arquivos por uma quantidade de linhas. Executado por "`python utils/split-files.py 10000 True data/raw/201915.csv data/splitted/201915`" para gerar arquivos com 10 mil linhas e cabeçalho do arquivo 201915.csv para vários 201915_numero, o encoding = "ISO-8859-1" adicionado por se tratar de um arquivo de caracteres latinos produzido com esse encoding.

```
import sys
import os

# Parâmetro para número de linhas para fragmentos
splitLen = int(sys.argv[1])

# Parâmetro determinar se deve considerar a primeira linha como cabeçalho
hasHeader = bool(sys.argv[2])

# Parâmetro do caminho do arquivo original
inputName = sys.argv[3]

# Parâmetro com o prefixo dos arquivos gerados
outputName = sys.argv[4]

input = open(os.getcwd() + '/' + inputName, 'r',
                encoding = "ISO-8859-1").read().split('\n')
filename, extension = os.path.splitext(inputName)

at = 1
headerData = ''
for lines in range(0, len(input), splitLen):

    if (hasHeader and at == 1):
        headerData = input[0]
        outputData = input[lines+1:lines+splitLen]
    else:
        outputData = input[lines:lines+splitLen]

    output = open(os.getcwd() + '/' + outputName + '_' +
```

```
                    str(at) + extension, 'w')
    if (hasHeader):
        output.write(headerData + '\n')
    output.write('\n'.join(outputData))
    output.close()

    at += 1
```

Fonte: elaborado pelos autores

Anexo 3 - O arquivo "*/code/clustered.py"* com a leitura dos arquivos na pasta de "*/data/splitted*" que não estão na pasta "*/data/clustered"*, filtrando pelos códigos selecionados e agrupando pelo código do cliente e armazenando em "/*data*/*clustered*".

```
import pandas as pd
from os import listdir

# Lista dos arquivos na pasta de separados
splitted_files = listdir('./data/splitted')
# Lista dos arquivos na pasta de agrupados
clustered_files = listdir('./data/clustered')
# Lista dos arquivos não agrupados ainda
cluster_files = list(set(splitted_files) - set(clustered_files))

produtos_analise = [2559, 7248, 28411, 57996, 58002, 58006, 58416,
                    62529, 69815, 70825, 70989, 71663, 72148, 73440,
                    74244, 85400, 86727, 86944, 86969, 89073, 92529,
                    92536, 92569, 92598, 92788, 92798, 92806]

for filename in cluster_files:
    df_dados = pd.read_csv('./data/splitted/' + filename)

    # Remover colunas não utilizadas
    df_dados = df_dados.drop(
        columns=[
            "QT_ITEM", "QT_PONTO", "DC_PRODUTO", "VL_VENDA",
            "ID_REFIL", "CD_FAMILIA_PRODUTO", "CD_LINHA_PRODUTO"]
    )

    # Separar produtos por cliente
    data = []
    for pessoa in df_dados.CD_PESSOA.unique():
        produtos = []
        for produto in df_dados[df_dados.CD_PESSOA == pessoa].CD_PRODUTO:
            if produto in produtos_analise:
                produtos.append(produto)
```

```
        if (len(produtos) > 0):
            data.append([pessoa, produtos])

    # Salvar agrupamento para a pasta de agrupamento
    if (len(data) > 0):
        df = pd.DataFrame (data, columns = ['Cliente', 'Produtos'])
        df.to_csv('./data/clustered/' + filename)
```

Fonte: elaborado pelos autores

Anexo 4 - O arquivo */code/*separated.py separando as listagens que o produto 2559 aparece para um arquivo na pasta "*/data/separated"*.

```
import pandas as pd
from os import listdir

# Lista de dados
clustered_files = listdir('./data/clustered')

# Produto escolhido para separação
codigo_produto = 2559

# Converter texto para lista
contem_produto = lambda produtos: codigo_produto in list(map(
                                int, produtos.strip(']['').split(', ')))

data = []
for filename in clustered_files:
    df = pd.read_csv('./data/clustered/' + filename)

    # Separar compras de clientes por produto
    for produtos in df[df.Produtos.apply(contem_produto)].Produtos:
        data.append(produtos)

# Salvar agrupamento para a pasta de separados
df = pd.DataFrame(data, columns = ['Produtos'])
df.to_csv('./data/separated/' + str(codigo_produto) + '.csv')
```

Fonte: elaborado pelos autores

Anexo 5 - O arquivo "*/code/analytics.py"* aplicação do modelo de apriori da biblioteca apyori com os parâmetros para filtrar os coeficientes e impressão dos resultados.

```
import pandas as pd
from apyori import apriori
```

```
# Converter texto para lista
converter_produto = lambda produtos: list(map(int,
                            produtos.strip(']['').split(', ')))
# Produto escolhido para análise
codigo_produto = 2559

# Ler arquivo com as transações com o produto
df = pd.read_csv('./data/separated/' + str(codigo_produto) + '.csv')

transactions = list(df.head(5).Produtos.apply(converter_produto))
analytics = apriori(transactions,
                    min_support=0.3,
                    min_confidence=0.6,
                    min_lift=1,
                    max_length=None)

for analytic in list(analytics):
    if ((codigo_produto in analytic.items) and (len(analytic.items) == 2)):
        print(analytic)
        print('\n')
```

Fonte: elaborado pelos autores