UNIVERSIDADE FEDERAL DE JUIZ DE FORA
CURSO DE GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

PEDRO REIS CABREIRA

ANÁLISE COMPARATIVA E COMPLEMENTAR DO EMPREGO DE ALGORITMOS DE CLUSTERIZAÇÃO E ASSOCIAÇÃO EM BASE DE DADOS *E-COMMERCE*

JUIZ DE FORA
2023
PEDRO REIS CABREIRA

ANÁLISE COMPARATIVA E COMPLEMENTAR DO EMPREGO DE ALGORITMOS DE CLUSTERIZAÇÃO E ASSOCIAÇÃO EM BASE DE DADOS E-COMMERCE

Trabalho de Conclusão de Curso apresentado a Faculdade de Engenharia da Universidade Federal de Juiz de Fora, como requisito parcial para a obtenção do título de Engenheiro de Produção.

Orientadora: DSc. Mariana Paes da Fonseca

JUIZ DE FORA

2023

Ficha catalográfica elaborada através do programa de geração
automática da Biblioteca Universitária da UFJF,
com os dados fornecidos pelo(a) autor(a)

Cabreira, Pedro Reis .
Análise comparativa e complementar do emprego de algoritmos de clusterização e associação em base de dados e-commerce / Pedro Reis Cabreira. -- 2023.
82 p. : il.

Orientadora: Mariana Paes da Fonseca
Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Juiz de Fora, Faculdade de Engenharia, 2023.

1. Associação. 2. Clusterização. 3. Base de dados. 4. Mineração de dados. I. Paes da Fonseca, Mariana , orient. II. Título.

PEDRO REIS CABREIRA

ANÁLISE COMPARATIVA E COMPLEMENTAR DO EMPREGO DE ALGORITMOS DE CLUSTERIZAÇÃO E ASSOCIAÇÃO EM BASE DE DADOS E-COMMERCE

Trabalho de Conclusão de Curso apresentado a Faculdade de Engenharia da Universidade Federal de Juiz de Fora, como requisito parcial para a obtenção do título de Engenheiro de Produção.

Aprovada em 1 de dezembro de 2023

BANCA EXAMINADORA

Dsc, Mariana Paes da Fonseca (Orientadora)

UFJF

Dsc, Roberto Malheiros Moreira Filho

UFJF

Dsc, Romir Soares de Souza Filho

UFJF

AGRADECIMENTOS

Agradeço a Deus, aos familiares que me apoiaram, à Universidade Federal de Juiz de Fora, a todos os professores que agregaram na minha formação e a orientadora e professora Mariana.

RESUMO

O aumento exponencial na disponibilidade de dados na era contemporânea abre portas para oportunidades significativas, proporcionando vantagens competitivas às empresas que exploram eficazmente esses recursos. Neste contexto, o emprego de técnicas de Ciência de Dados e Mineração de Dados torna-se crucial para extrair insights valiosos e promover decisões informadas. Entre diversas técnicas disponíveis, foi escolhido o uso das técnicas de associação e clusterização através do uso dos algoritmos *Apriori*, *FP-Growth* e *K-means*, que em conjunto, podem oferecer um arsenal poderoso para analisar, entender e explorar dados em diversas aplicações. Ao aplicá-los de maneira integrada e adaptada ao contexto específico, é possível obter insights valiosos que podem ser traduzidos em estratégias informadas e decisões mais eficientes.

As análises realizadas nesse trabalho foram feitas utilizando uma base de dados de transações de um site de *e-commerce*. Foi realizada uma análise exploratória prévia dos dados que ajudou a revelar padrões de comportamento dos consumidores. Em seguida para aprofundar o entendimento, foi optado por utilizar *Apriori* e *FP-Growth* nessa base para identificar relações entre as categorias de produtos. Outra análise realizada no trabalho foi a comparação de tempos de execução entre *Apriori* e *FP-Growth*, em duas bases de dados com tamanhos diferentes, o que ofereceu insights sobre a eficiência relativa desses métodos em um contexto em que podem existir bases de dados de diferentes tamanhos.

Por último foi realizada a implementação prévia do algoritmo *K-means* sobre valores de pagamento, o que permitiu a formação de diferentes clusters de clientes em função da receita que é gerada por ele. Em seguida foi aplicado o algoritmo *Apriori* nos clusters gerados pelo *K-means*, o que proporcionou resultados distintos em comparação com a análise anterior. Essa abordagem revelou padrões de associação específicos em diferentes segmentos de clientes, contribuindo para uma compreensão mais refinada do comportamento de compra.

Este trabalho contribui para a compreensão de como algoritmos de clusterização e associação podem ser integrados de maneira sinérgica**.** A análise comparativa e complementar destas técnicas, aplicadas a uma base de dados e-commerce, destaca a importância de adaptar as metodologias de acordo com as características específicas do conjunto de dados em questão.

Palavras-chave: Associação, clusterização, base de dados

ABSTRACT

The exponential increase in data availability in the contemporary era opens doors to significant opportunities, providing competitive advantages to companies that effectively exploit these resources. In this context, the use of Data Science and Data Mining techniques becomes crucial to extract valuable insights and promote informed decisions. Among several available techniques, the use of association and clustering techniques was chosen through the use of Apriori, FP-Growth and K-means algorithms, which together can offer a powerful arsenal to analyze, understand and explore data in various applications. By applying them in an integrated way and adapted to the specific context, it is possible to obtain valuable insights that can be translated into informed strategies and more efficient decisions.

The analyzes carried out in this work were carried out using a transaction database from an e-commerce website. A preliminary exploratory analysis of the data was carried out, which helped to reveal patterns of consumer behavior. Then, to deepen understanding, it was decided to use Apriori and FP-Growth on this basis to identify relationships between product categories. Another analysis carried out in the work was the comparison of execution times between Apriori and FP-Growth, in two databases with different sizes, which offered insights into the relative efficiency of these methods in a context in which there may be databases of different sizes.

Finally, the K-means algorithm was previously implemented on payment values, which allowed the formation of different clusters of customers depending on the revenue generated by them. The Apriori algorithm was then applied to the clusters generated by K-means, which provided different results compared to the previous analysis. This approach revealed specific association patterns across different customer segments, contributing to a more refined understanding of purchasing behavior.

This work contributes to the understanding of how clustering and association algorithms can be integrated in a synergistic way. The comparative and complementary analysis of these techniques, applied to an e-commerce database, highlights the importance of adapting the methodologies according to the specific characteristics of the data set in question.

Keywords: Association, clustering, database

LISTA DE FIGURAS

Figura 1: Exemplo de processamento da *FP-Tree* 34

Figura 2: Exemplo de sistema de recomendação de *Checkout* e *Instore* 41

Figura 3: Penetração do sistema de recomendação pelo tempo 42

Figura 4: A receita extra mensal direta gerada pelos sistemas de recomendação, apresentada como uma porcentagem do volume de negócios total mensal. 43

Figura 5: A receita extra direta, indireta e total gerada pelos sistemas de recomendação. 44

Figura 6: Bibliotecas Python utilizadas no trabalho 46

Figura 7: Importação das bases de dados para as suas respectivas variáveis 47

Figura 8: Atributos pertencentes a base de dados *clientes* 47

Figura 9: Atributos pertencentes a base de dados *itens_pedidos* 47

Figura 10: Atributos pertencentes a base de dados *pagamentos* 47

Figura 11: Atributos pertencentes a base de dados *pedidos* 48

Figura 12: Atributos pertencentes a base de dados *produtos* 48

Figura 13: Principais estados realizadores de pedidos 48

Figura 14: Principais cidades realizadoras de pedidos 49

Figura 15: Gráficos dos principais estados e cidades realizadores de pedidos 49

Figura 16: Estatísticas descritivas em relação aos valores de pagamento 50

Figura 17: Código para criação do histograma de valores de pagamento 50

Figura 18: Histograma dos valores de pagamento 51

Figura 19: Código para criação do histograma de valores de pagamento eliminando os outliers 52

Figura 20: Histograma dos valores de pagamento sem os *outliers* 52

Figura 21: Código para criação do gráfico de pizza relativo aos métodos de pagamento e gráfico 54

Figura 22: Código para criação do histograma relativo às parcelas de pagamento e histograma. 55

Figura 23: Código para criação do gráfico de linha relativo a quantidade de pedidos por data do pedido e gráfico 56

Figura 24: Código para adicionar a coluna de categoria de produtos na base *itens_pedidos* 57
Figura 25: Código para eliminar linhas com valores nulos e para informar a quantidade de produtos e categorias diferentes 57
Figura 26: Código para criação da base transações 57
Figura 27: Código para criação do gráfico de barras relativo a categorias de produtos e quantidade vendida e gráfico. 58
Figura 28: Código para converter a base *transacoes* em lista 59
Figura 29: Código para criação da matriz *one-hot* 60
Figura 30: Código para a aplicação do algoritmo *Apriori* 61
Figura 31: Código para a aplicação das regras de associação. 62
Figura 32: Código para aplicar o filtro em relação a influência. 62
Figura 33: Código para a aplicação do algoritmo *FP-Growth* 63
Figura 34: Código para calcular o tempo de execução dos algoritmos *Apriori* e *FP-Growth* 63
Figura 35: Código para criação do gráfico de comparação do tempo de execução entre os algoritmos 64
Figura 36: Código para filtragem da base *itens_pedidos_frequentes* 65
Figura 37: Código para a criação da base *transacoes_produtos* 65
Figura 38: Código para criação da nova matriz *one-hot* 66
Figura 39: Código para calcular os novos tempos de execução dos algoritmos *Apriori* e *FP-Growth* 66
Figura 40: Código para criação do gráfico de comparação dos novos tempos de execução entre os algoritmos 67
Figura 41: Código para criação da base *receita_cliente* 68
Figura 42: Código para criação do gráfico da curva da soma dos quadrados dos erros em função do número de clusters 69
Figura 43: Código para a aplicação do algoritmo *K-means* 70
Figura 44: Código para criação da base *receita_cliente3* 70
Figura 45: Código para obter a soma da receita total e a receita dos clusters 3,4 e 5 71
Figura 46: Código para adicionar os clusters na tabela *itens_pedidos* 71
Figura 47: Código para criação das bases de transações referente a cada cluster 72

Figura 48: Código para criação das matrizes *one-hot* 73
Figura 49: Código para a aplicação do algoritmo *Apriori* 74
Figura 50: Código para aplicação da regra de associação na base referente ao cluster 0 74
Figura 51: Código para aplicação da regra de associação na base referente ao cluster 1 75
Figura 52: Código para aplicação da regra de associação na base referente ao cluster 2 75
Figura 53: Código para aplicação da regra de associação na base referente ao cluster 3 76
Figura 54: Código para aplicação da regra de associação na base referente ao cluster 4 76

LISTA DE QUADROS

Quadro 1: Algoritmos associados às tarefas de Data Mining e tipos de Machine Learning 25

Quadro 2: Base de dados das transações 29

Quadro 3: Conjunto de itens candidatos k = 1, e seu respectivo suporte. 30

Quadro 4: Conjunto de itens candidatos k = 2, e seu respectivo suporte. 31

Quadro 5: Principais resultados de associação obtidos com a aplicação do Apriori. 32

Quadro 6: Principais resultados de associação obtidos com a aplicação do Apriori. 39

SUMÁRIO

1. INTRODUÇÃO 12

1.1 JUSTIFICATIVA 14

1.2 ESCOPO DO TRABALHO 15

1.3 ELABORAÇÃO DOS OBJETIVOS 15

1.4 DEFINIÇÃO DA METODOLOGIA 16

1.5 ESTRUTURA DO TRABALHO 17

2. REVISÃO TEÓRICA

2.1 CIÊNCIA DE DADOS, DATA MINING E MACHINE LEARNING 18

2.2 FERRAMENTAL COMPUTACIONAL, ESTATÍSTICO E MATEMÁTICO EMPREGADO EM ALGORITMOS DE ASSOCIAÇÃO E CLUSTERIZAÇÃO. 26

2.3 APLICAÇÕES DE REGRAS DE ASSOCIAÇÃO NO E-COMMERCE 36

3. DESENVOLVIMENTO 45

3.1 PROCESSO DE EXTRAÇÃO DOS DADOS 45

3.2 ANÁLISE EXPLORATÓRIA DOS DADOS 46

3.3 APLICAÇÃO DOS ALGORITMOS DE ASSOCIAÇÃO 58

3.4 COMPARAÇÃO DOS TEMPOS DE EXECUÇÃO ENTRE ALGORITMOS DE ASSOCIAÇÃO 63

3.5 APLICAÇÃO DO ALGORITMO DE CLUSTERIZAÇÃO 67

3.6 APLICAÇÃO DO ALGORITMO DE ASSOCIAÇÃO EM DIFERENTES CLUSTERS 71

4. CONCLUSÃO 77

REFERÊNCIAS 78

1. INTRODUÇÃO

Marson (2014) afirma que, historicamente, a indústria foi o fator mais poderoso de aceleração do crescimento econômico. O setor industrial exerceu grande impacto em diversos setores da economia e sobre todo o ambiente institucional e social. Nas últimas décadas, o crescimento industrial foi bastante expressivo, pois o desenvolvimento tecnológico causou grandes mudanças e rupturas, transformando a maneira como as empresas operam e como os consumidores interagem com os produtos e serviços.

Uma das mudanças percebidas pela indústria e pelo mercado é a grande quantidade de dados disponíveis. De acordo com a empresa de tecnologia Cisco (2020), em seu relatório *Visual Networking Index* (VNI), a quantidade de dados gerados na internet diariamente em 2020 foi de cerca de 4,7 zettabytes (ou 4,7 trilhões de gigabytes). Dessa forma, a capacidade das empresas conseguirem transformá-los em informações relevantes é cada vez mais essencial para adquirir vantagem competitiva.

Uma maneira que está se tornando comum para se fazer essa manipulação é através de ciência de dados. A ciência de dados é uma área que integra as áreas de ciência da computação, estatística e matemática e que concentra seus estudos no desenvolvimento de ferramentas, metodologias e técnicas para a solução de diferentes problemas em ambientes individuais, organizacionais e sociedade (AALST, 2016).

De acordo com Amaral (2016), a ciência de dados é resumida de forma equivocada apenas aos processos de análises dos dados, onde com o uso de estatística e aprendizado de máquina, ou com a simples aplicação de filtros, pode se produzir informação e conhecimento. Nessa visão “míope”, a ciência de dados passa a ser vista apenas como um nome mais elegante para a própria estatística. O autor entende que a ciência de dados é bem mais complexa do que parece. Segundo Cardoso (2019), a ciência de dados é um novo campo transdisciplinar que constrói e sintetiza várias disciplinas e corpos de conhecimento relevantes, incluindo estatística, informática, computação, comunicação, gerenciamento e sociologia.

Dessa forma, a adoção da Ciência de Dados pelo mercado tem sido cada vez mais ampla. Empresas de diferentes setores e tamanhos estão reconhecendo o valor dos dados e os benefícios de os utilizarem para obter melhores resultados. Em um estudo, os pesquisadores

Müller et al. (2018) analisaram um conjunto de dados de 814 empresas durante os anos 2008-2014 e descobriram que os investimentos em *Big Data*[1] e *Data Science*[2] estão associados a uma melhoria de 3 a 7% na produtividade dos negócios. Além disso, analisando organizações esportivas profissionais, Troilo et al. (2016) compararam o crescimento anual esperado para a indústria (3%) com o crescimento alcançado por organizações que adotaram soluções de *analytics*[3] (7,2%) no ano seguinte à adoção.

Com o aumento da popularidade dessas ferramentas, o engenheiro de produção tem a possibilidade de se beneficiar bastante, já que elas podem ser muito úteis para ajudar na melhoria da produtividade. No site da *University of Californnia. Riverside* (2023) é destacado que a engenharia é uma indústria que tem sido particularmente influenciada pela crescente necessidade de coleta e análise de dados. Como o *Big Data* começou a desempenhar um papel maior nas indústrias em todo o mundo, os engenheiros foram chamados para desempenhar um papel influente na maneira como essas informações são coletadas, armazenadas e aproveitadas. Profissionais com formação em engenharia geralmente provam ser particularmente hábeis no desenvolvimento de técnicas para analisar grupos de dados para extrair informações valiosas. Dessa forma a ciência de dados pode proporcionar ao engenheiro *insights* valiosos para entender e aprimorar os processos produtivos, além de ajudar a enfrentar os desafios no campo.

Dessa forma este trabalho buscou utilizar técnicas de ciência de dados, explorando o *Data Mining* através do uso de regras de associação e clusterização em uma base de dados de

---

[1] De acordo com Sagiroglu (2013), Big data é um termo para conjuntos de dados massivos com uma estrutura grande, variada e complexa, com dificuldades para ser armazenada, analisada e visualizada para outros processos ou resultados.

[2] De acordo com Cao (2017) ciência de dados é um campo interdisciplinar que sintetiza e se baseia em estatística, informática, computação, comunicação, gestão e sociologia para estudar dados e seus ambientes, a fim de transformar dados em percepções e decisões seguindo um pensamento e metodologia.

[3] De acordo com Sagiroglu (2013), *analytics* é o processo de pesquisa em relação a dados, para extrair informações, como padrões ocultos e correlações secretas, que possam ser úteis e usadas para tomar decisões..

uma grande empresa do setor de vendas *e-commerce*. O objetivo do uso dessas ferramentas no trabalho foi de demonstrar o potencial do uso de *Data Mining* para tomadas de decisões através do resultado obtido, e demonstrar a utilidade de regras de associação e da clusterização em bases de dados de empresas do setor de vendas *e-commerce*.

1.1 JUSTIFICATIVA

A área de *e-commerce* tem experimentado um crescimento significativo nos últimos anos, impulsionado pelo avanço da tecnologia e mudanças nos hábitos de consumo das pessoas. De acordo com De Melo (2021), o faturamento do *e-commerce* vem em ascendência no período 2011-2020, crescendo 351,3%, com destaque para o ano de 2020, quando cresceu 41% em relação ao ano anterior. O relatório nº43 da *Webshoppers* (2021) salienta que a pandemia da Covid-19 contribuiu muito para a elevação dos números do *e-commerce* no Brasil, pois muitas pessoas passaram a buscar novas formas de adquirir mercadorias sem ter que frequentar os tradicionais pontos de compras.

Dessa forma é inevitável que, com o crescimento do mercado, também aconteça o acirramento da competição entre as empresas. Com o aumento da concorrência nesse setor bastante dinâmico, é fundamental para as empresas de *e-commerce* adotarem estratégias eficazes para atrair e reter clientes, aumentar as vendas e melhorar a experiência do usuário. Nesse contexto, a aplicação de técnicas de análise de dados, como regras de associação, clusterização, *Data Mining* e ciência de dados, desempenha um papel fundamental na obtenção de *insights* valiosos para a tomada de decisões estratégicas.

A aplicação de regras de associação em uma base de dados de *e-commerce* permite a descoberta de padrões ocultos e associações entre os itens comprados pelos clientes. Isso é extremamente relevante, pois proporciona uma compreensão mais profunda do comportamento do consumidor, das preferências, dos padrões de compra e das interações entre os produtos, podendo ser usado até como base para um sistema de recomendações de produtos. De acordo com Vaidya (2017), uma grande variedade de informações está disponível na web. Algumas pessoas que não têm experiência com internet, podem tomar decisões ruins se recomendações não estão disponíveis. Por causa dos sistemas de recomendação os visitantes que navegam nos sites acabam comprando o produto e ficam

satisfeitos. Quanto mais os clientes estão satisfeitos com os sistemas de recomendação, eles os usam e mais popular torna-se o site *e-commerce*. Com o uso de regras de associação em bases de dados *e-commerce*, podem ser identificadas relações entre itens frequentemente comprados juntos, possibilitando criar estratégias de vendas cruzadas, recomendações personalizadas e promoções direcionadas, o que pode ajudar no aumento de vendas, fidelização de clientes e melhorias na experiência de compra. Além disso, existem algumas questões que também são relevantes quando se trata da aplicação desses algoritmos, como qual deles utilizar de acordo com o tamanho da base disponível, ou se eles podem ser utilizados em conjunto com outros algoritmos para obter melhores resultados, por exemplo a aplicação prévia da clusterização, que pode agregar bastante no uso de algoritmos de associação, pois além de encontrar associações globais em toda a base de dados, também é possível descobrir padrões específicos dentro de cada cluster, fornecendo *insights* mais detalhados e contextuais.

1.2 ESCOPO DO TRABALHO

O presente trabalho utiliza a análise exploratória e ferramentas estatísticas e computacionais de *Data Mining* em um banco de dados e-commerce da empresa Olist disponibilizado no site Kaggle.com, com informações de mais de 100.000 compras entre os anos de 2016 e 2018, para responder às seguintes questões: Quais as principais associações encontradas no banco de dados e-commerce utilizado no trabalho ? Existe diferenças no desempenho entre os algoritmos *Apriori* e *FP-Growth* ? O desempenho de algoritmos de associação pode ser aprimorado com o uso prévio de algoritmos de agrupamento? Quais conclusões podem ser tiradas a partir do uso dos algoritmos de *Data Mining* nesse banco de dados *e-commerce* para tomadas de decisões estratégicas?

1.3 ELABORAÇÃO DOS OBJETIVOS

O objetivo geral do presente trabalho foi avaliar o emprego de algoritmos de associação em uma base de dados *e-commerce*, em relação aos resultados que eles podem gerar e como eles podem ser importantes para entender melhor o comportamento e padrões de compra dos clientes. Para isso foi necessário realizar alguns objetivos específicos como

aplicar os algoritmos *Apriori* e *FP-Growth* na base de dados *e-commerce* para obter os resultados de padrões de compra, e em seguida comparar os tempos de execução entre eles em bases de dados com tamanhos diferentes. Posteriormente foram analisados os resultados da aplicação do algoritmo *K-means* de forma prévia e complementar com os algoritmos de associação, que geraram resultados diferentes de padrões de compra. O trabalho demonstra como os algoritmos de associação podem ser explorados para gerar associações que podem ser úteis para realizar análise de padrões de compra e como a utilização da clusterização através do algoritmo *K-means* pode potencializar os resultados obtidos desses algoritmos. A aplicação da regra de associação e do algoritmo de clusterização é feita com uso de ferramentas de *Data Mining*, para responder às questões evidenciadas no escopo.

1.4 DEFINIÇÃO DA METODOLOGIA

A pesquisa possui uma abordagem descritiva, fornecendo uma descrição detalhada dos resultados obtidos através da demonstração do uso de algoritmos de associação e de clusterização para um caso prático. Dessa forma a pesquisa também possui natureza aplicada e adota o procedimento experimental.

Para que o objetivo da pesquisa seja alcançado, é necessário ter bem definido as ferramentas utilizadas, assim como as etapas necessárias. Na primeira etapa foi necessário identificar uma base de dados pública de algum site de *e-commerce* que tivesse registros suficientes para realizar a análise desejada. A base de dados escolhida foi da empresa nacional Olist, que é uma plataforma de comércio eletrônico que conecta pequenas e médias empresas a grandes *marketplaces*, como Mercado Livre e Amazon

Posteriormente foi necessária a utilização da linguagem Python de programação para realizar o tratamento dos dados para corrigir ou eliminar linhas e colunas que possuíam características que causam problemas na execução do algoritmo de associação. Em seguida foi realizada uma análise exploratória dos dados com empregos de gráficos para entender melhor a disposição dos dados e obter percepções sobre suas relações que puderam ser úteis durante o restante da pesquisa.

Em seguida, foram aplicados os algoritmos *Apriori* e *FP-Growth*, ferramentas de *Data Mining* disponíveis em Python, para gerar regras de associação. Essas regras foram então

analisadas e interpretadas, permitindo descrever os principais padrões de compra identificados na base de dados. Em seguida o tempo de execução dos algoritmos foi comparado em duas bases de dados de tamanhos diferentes, para destacar a diferença do desempenho entre eles.

Após, foi aplicado o algoritmo de clusterização *K-means*, para agrupar clientes que possuem diferentes características em relação aos valores gastos por eles. Com a formação dos diferentes *clusters*, foi possível aplicar novamente o algoritmo *Apriori*, obtendo novos resultados de associações. Dessa forma, buscou-se fornecer percepções sobre a efetividade dos algoritmos na análise de padrões de compra, evidenciar as diferenças de desempenho entre eles, assim como o potencial aprimoramento de seu uso utilizando o algoritmo de clusterização de forma prévia.

1.5 ESTRUTURA DO TRABALHO

O presente trabalho está dividido em quatro capítulos, sendo o primeiro relacionado à introdução que demonstra a importância da ciência de dados no cenário atual em que cada vez mais existem dados disponíveis, que podem gerar informações úteis para tomadas de decisões estratégicas. Além disso o capítulo também evidencia os objetivos da pesquisa e as metodologias utilizadas.

O capítulo dois está relacionado com a revisão bibliográfica, onde são apresentados conceitos em relação a ciência de dados, *Data Mining* e *Machine Learning*. Juntamente desses conceitos são evidenciados os relacionados aos principais algoritmos de associação e ao algoritmo *K-means* de clusterização, assim como alguns exemplos que demonstram a utilidade do uso desses algoritmos.

O terceiro capítulo é referente à execução e desenvolvimento do trabalho utilizando as ferramentas de *Data Mining* na base de dados, e demonstrando as etapas de forma prática até ao alcance do resultado, evidenciando as justificativas de se utilizar tais ferramentas estatísticas e computacionais, assim como os resultados obtidos da análise exploratória dos dados, do emprego dos algoritmos de associação e de clusterização.

O último capítulo se refere a parte da conclusão, demonstrando as respostas pras perguntas presentes no escopo do trabalho e as percepções do autor em relação aos resultados obtidos.

2. REVISÃO TEÓRICA

2.1 CIÊNCIA DE DADOS, DATA MINING E MACHINE LEARNING

De acordo com Cao (2017) a Ciência de Dados pode ter diferentes definições de acordo com diferentes perspectivas, seja ela superficial, disciplinar ou de produtos de dados.

a. Definição superficial: Ciência de dados é o estudo dos dados.

b.Definição disciplinar: Ciência de Dados é um novo campo interdisciplinar que sintetiza e se baseia em estatística, informática, computação, comunicação, gestão e sociologia para estudar dados e seus ambientes (incluindo domínios e outros aspectos contextuais, como organização e aspectos sociais) a fim de transformar dados em percepções e decisões seguindo um pensamento e metodologia.

c. Definição produto de dados: Um produto de dados é um tipo de entrega a partir de dados, e pode ser uma descoberta, previsão, serviço, recomendação, *insight* de tomada de decisão, pensamento, modelo, modo, paradigma, ferramenta ou sistema. Os valores dos produtos de dados que são alcançados no final são conhecimento, inteligência, sabedoria e decisão. Dessa forma, a ciência de dados possui uma ampla gama de possibilidades de aplicação em diversos setores e áreas de conhecimento.

Um termo que é muito utilizado quando o assunto é ciência de dados é o *Data Mining*. De acordo com Osman (2019), o *Data Mining* é definido como a prática de examinar um grande banco de dados pré-existente para gerar novas informações. Suas técnicas são muito úteis, pois podem ajudar as organizações a analisar B*ig Data* de maneira eficiente. Existem várias delas disponíveis que podem ser usadas para alimentar diferentes problemas e fornecer *insights* sobre esses problemas de negócios.

Segundo Jackson (2002), vários tipos de problemas ou tarefas de análise são normalmente encontrados durante um projeto de *Data Mining*. Dependendo do resultado desejado, diferentes técnicas de análise de dados com objetivos diferentes podem ser aplicadas sucessivamente. Por exemplo, para determinar quais clientes provavelmente comprarão um novo produto, um analista de negócios pode precisar primeiro usar a análise de *cluster* para segmentar o banco de dados do cliente e, em seguida, aplicar a análise de

regressão para prever o comportamento de compra para cada cluster. As tarefas de análise de *Data Mining* normalmente se enquadram nas categorias listadas abaixo (JACKSON, 2002).

a. *Data Summarization*: Dá ao usuário uma visão geral da estrutura dos dados e é geralmente realizada nas fases iniciais de um projeto. Este tipo de análise exploratória inicial de dados pode ajudar a entender a natureza dos dados e a encontrar hipóteses potenciais para informação. Técnicas simples de estatística descritiva e de visualização geralmente se aplicam.

b. *Segmentation*: Separa os dados em subgrupos ou classes interessantes e significativas. Nesse caso, o analista pode hipotetizar determinados subgrupos como relevantes para a questão do negócio com base no conhecimento prévio ou com base no resultado da descrição e resumo dos dados. As técnicas automáticas de *clustering* podem detectar estruturas anteriormente insuspeitas e ocultas nos dados que permitem a segmentação. Técnicas de clusterização, visualização e redes neurais geralmente se aplicam.

c. *Classification*: Assume que um conjunto de objetos, caracterizados por alguns atributos ou recursos, pertencem a diferentes classes. O rótulo de classe é um identificador qualitativo discreto, por exemplo, grande, médio ou pequeno. O objetivo é construir modelos de classificação que atribuam a classe correta para objetos não vistos e não rotulados anteriormente. Os modelos de classificação são mais usados para modelagem preditiva, análise discriminante, árvore de decisão, métodos de indução de regras e algoritmos genéticos.

d. *Prediction*: É muito semelhante à classificação. A diferença é que na predição, a classe não é um atributo qualitativo discreto, mas contínuo. O objetivo da predição é encontrar o valor numérico do atributo alvo para objetos não vistos, este tipo de problema também é conhecido como regressão, e se a previsão lidar com dados de séries temporais, ela é frequentemente chamada de previsão. A análise de regressão, árvores de decisão e redes neurais geralmente se aplicam.

e. *Dependency Analysis*: Trata de encontrar um modelo que descreva dependências (ou associações) entre itens de dados ou eventos. As dependências podem ser usadas para prever o valor de um item com base em informações sobre outros itens de dados. A análise de dependência tem conexões estreitas com classificação e previsão porque as dependências são usadas implicitamente para a formulação de modelos preditivos. Análise de correlação,

análise de regressão, regras de associação, raciocínio baseado em casos e técnicas de visualização geralmente se aplicam.

De acordo com Ming-Syan (1996), para realizar uma mineração de dados eficaz, é preciso primeiro examinar que tipo de recursos um sistema de descoberta de conhecimento aplicado deve ter e que tipo de desafios podem ser enfrentados no desenvolvimento de suas técnicas. Ele também destaca os requisitos e desafios presentes no *Data Mining* (MING-SYAN, 1996):

a. Manipulação de diferentes tipos de dados: Existem muitos tipos de dados e bancos de dados usados em diferentes aplicações, pode-se esperar que um sistema de descoberta de conhecimento deve ser capaz de realizar a mineração de dados eficaz nos diferentes tipos. Como muitos dos bancos de dados disponíveis ainda são relacionais[4], é crucial que um sistema de mineração de dados seja eficiente e tenha descoberta de conhecimento eficaz em dados relacionais. Além disso, muitos bancos de dados aplicáveis contêm tipos de dados, como dados estruturados e dados complexos, objetos, hipertexto, dados multimídia, dados espaciais, dados temporais, dados de transação, dados legados, etc. Um sistema poderoso deve ser capaz de realizar mineração de dados em tais tipos complexos de dados também. Complementando a visão de Ming-Syan (1996), o ranking de sistemas de gerenciamento de banco de dados, criado pelo site db-engines.com/en/ranking (2023), onde são utilizados parâmetros como número de menções do sistema em sites, frequência de discussões técnicas sobre o sistema, número de ofertas de emprego, em que o sistema é mencionado, quantidade de perfis em redes profissionais, nas quais o sistema é citado e relevância nas redes sociais para identificar os sistemas mais populares. Chegando ao resultado de que dos 10 sistemas mais populares, 7 ainda são relacionais.

b. Eficiência e escalabilidade de algoritmos de *Data Mining*: Para extrair efetivamente informações de uma enorme quantidade de dados em um banco de dados, os algoritmos de descoberta de conhecimento devem ser eficientes e escaláveis para grandes volumes. Ou seja,

[4] De acordo com a empresa Oracle (2023), um banco de dados relacional é um tipo de banco de dados que armazena e fornece acesso a pontos de dados relacionados entre si. Bancos de dados relacionais são baseados no modelo relacional, uma maneira intuitiva e direta de representar dados em tabelas. Em um banco de dados relacional, cada linha na tabela é um registro com uma ID exclusiva chamada chave. As colunas da tabela contêm atributos dos dados e cada registro geralmente tem um valor para cada atributo, facilitando o estabelecimento das relações entre os pontos de dados.

o tempo de execução de um algoritmo de *Data Mining* deve ser previsível e aceitável em grandes bancos de dados.

c. Utilidade, certeza e expressividade dos resultados do *Data Mining*: O conhecimento descoberto deve retratar com precisão o conteúdo do banco de dados e ser útil para determinados formulários. A imperfeição deve ser expressa por medidas de incerteza, na forma de regras aproximadas ou regras quantitativas. Ruído e dados excepcionais devem ser tratados com elegância em sistemas de mineração de dados. Isso também motiva um estudo sistemático de medir a qualidade do conhecimento descoberto, incluindo interesse e confiabilidade, pela construção de modelos estatísticos, analíticos, simulados e ferramentas.

d. Expressão de vários tipos de solicitações e resultados de *Data Mining*: Diferentes tipos de informação podem ser descobertas a partir de uma grande quantidade de dados. Além disso, é possível examinar informação descoberta a partir de diferentes pontos de vista e apresentá-los em diferentes formas. Isso exige que se expresse tanto as solicitações de *Data Mining*, quanto a informação em linguagens de alto nível, gráficos e interfaces de usuário para que a tarefa de mineração de dados possa ser especificada por não especialistas e a descoberta possa ser compreensível e diretamente utilizável pelos usuários.

e. Minerar informações de diferentes fontes de dados: A ampla área de rede computacional, incluindo Internet, conecta muitas fontes de dados e forma enormes bancos de dados distribuídos e heterogêneos. Extraindo conhecimento de diferentes fontes de dados formatados ou não formatados com diversas semânticas de dados representam novos desafios para o *Data Mining*. Por outro lado, o *Data Mining* pode ajudar a revelar as regularidades de dados de alto nível em bancos de dados heterogêneos que dificilmente podem ser descobertas por sistemas de consulta simples.

f. Proteção da privacidade e segurança dos dados: Quando os dados podem ser visualizados de vários ângulos diferentes e em diferentes níveis de abstração, ameaça o objetivo de proteger a segurança dos dados contra a invasão de privacidade. É importante estudar quando a descoberta de conhecimento pode levar a uma invasão de privacidade e quais medidas de segurança podem ser desenvolvidas para impedir a divulgação de informações confidenciais.

Além do *Data Mining*, outro termo que está bastante associado à Ciência de Dados é o *Machine Learning*. De acordo com Teng (2018), aprender é o aprimoramento ou melhoria da

capacidade do sistema em seu trabalho repetido. Quando o sistema executa o mesmo ou tarefas semelhantes na próxima vez, será melhor ou mais eficiente. O *Machine Learning* é uma maneira importante para que os computadores adquiram conhecimento e um importante indicador de inteligência artificial. É uma disciplina que estuda como usar computadores para simular ou realizar atividades de aprendizagem . É para estudar como fazer com que as máquinas obtenham novos conhecimentos e habilidades, identificando e usando conhecimento. Geralmente, acredita-se que o aprendizado de máquina é um processo de aquisição de conhecimento com um propósito específico. Seu desempenho interno é um processo de crescimento do conhecimento do desconhecido para o conhecido. Seu desempenho externo é a melhoria de algum desempenho e adaptabilidade do sistema, podendo então terminar a tarefa que não pôde ser concluída ou concluída da melhor forma.

De acordo com El Naqa (2015), historicamente o início do *Machine Learning* pode ser rastreado até o século XVII, com o desenvolvimento de máquinas que podiam emular a capacidade humana de adicionar e subtrair, criadas por Pascal e Leibniz. Na história moderna, Arthur Samuel da IBM cunhou o termo "*Machine Learning*" e demonstrou que os computadores podem ser programados para aprender a jogar damas. Posteriormente seguiu-se com o desenvolvimento do *perceptron*[5], criado por Rosenblatt como uma das primeiras arquiteturas de rede neural em 1958. No entanto, o entusiasmo inicial sobre o *perceptron* foi atenuado por uma observação feita por Minsky de que a capacidade de classificação do *perceptron* é limitada. Um avanço foi alcançado em 1975 pelo desenvolvimento do *perceptron* multicamada (MLP) da Werbos. Isso foi seguido pelo desenvolvimento de árvores de decisão por Quinlan em 1986 e máquinas de vetores de suporte por Cortés e Vapnik. Conjuntos de algoritmos de *Machine Learning*, que combinam múltiplos aprendizes, foram subsequentemente propostos, incluindo *Adaboost* e florestas aleatórias. Mais recentemente, diversos algoritmos têm surgido sob a noção de *Deep Learning*[6]. Esses algoritmos são

[5] É um dos primeiros modelos de neurônios artificiais e é considerado uma base fundamental para a área de aprendizado de máquina. De acordo com Rosenblatt (1958) o *perceptron* é projetado para ilustrar algumas das propriedades fundamentais de sistemas inteligentes em geral.

[6] De acordo com Deng (2014), *Deep Learning* é uma classe de algoritmos de *machine learning* que usa uma cascata de várias camadas de unidades de processamento não lineares para recursos de extração e transformação. Cada camada sucessiva usa a saída da camada anterior como entrada, e aprende níveis múltiplos de representações que correspondem a diferentes níveis de abstração.

capazes de aprender boas representações dos dados que facilitam a extração de informações úteis ao construir classificadores ou outros preditores.

Alzubi (2018) destaca que os problemas do mundo real possuem alta complexidade, o que os torna excelentes candidatos para aplicação de *Machine Learning*. Ele pode ser aplicado a várias áreas da computação para projetar e programar algoritmos com saída de alto desempenho, por exemplo, filtragem de spam de e-mail, detecção de fraude em redes sociais, negociação de ações on-line, detecção de rostos e formas, diagnóstico médico, previsão de tráfego, recomendação de produtos, entre outros. Outros exemplos que podem ser percebidos na vida das pessoas são os carros autônomos do Google, as sugestões da Netflix para filmes e shows que uma pessoa pode gostar, mecanismos de recomendação on-line, como sugestões de amigos no Facebook, “mais itens a considerar” e “pegue algo para você” na Amazon, e detecção de fraude de cartão de crédito.

Existem diferentes abordagens por parte de diferentes autores em relação às categorias de paradigmas do *Machine Learning*. Na visão de McCrea (2015) e de Berry (2020) ele pode ser dividido em aprendizado supervisionado e não supervisionado, já na visão de El Naqa (2015) além das duas categorias citadas, ela adiciona o aprendizado semi-supervisionado. Porém, de acordo com Berry (2020) a maioria dos artigos reconhece somente o *Machine Learning* supervisionado e não supervisionado.

De acordo com Kotsiantis (2007) no *Machine Learning* supervisionado a aprendizagem envolve atributos de saída predeterminados além do uso de atributos de entrada. Os algoritmos tentam prever e classificar o atributo predeterminado, e suas precisões e erros de classificação ao lado de outras medidas de desempenho são dependentes das contagens do atributo predeterminado corretamente previsto ou classificado. Também é importante notar que o processo de aprendizagem para, quando o algoritmo atinge um nível aceitável de desempenho, sendo que esses algoritmos podem ser classificados como de regressão ou classificação. Mahesh (2018) complementa com a explicação dizendo que o conjunto de dados de entrada é dividido em conjunto de dados de treinamento e teste sendo que o aprendizado supervisionado infere uma função nos dados rotulados de treinamento, o qual possui a variável de saída que precisa ser prevista ou classificada.

Um exemplo comum de aplicação de algoritmos de regressão é a previsão de preços de imóveis. Usando um conjunto de dados históricos de imóveis vendidos, características

como área, número de quartos e localização são usadas para treinar um modelo de regressão. Esse modelo é então utilizado para estimar o preço justo de mercado de novos imóveis com base nessas características. Isso é útil para corretores de imóveis e empresas imobiliárias na determinação de preços, financiamento e investimentos. Outro exemplo, mas em relação aos algoritmos de classificação, é a detecção de spam em e-mails. Utilizando um conjunto de dados de treinamento com exemplos de e-mails rotulados como spam ou legítimos, um modelo de classificação é treinado para identificar automaticamente se um e-mail é spam ou não com base em suas características, como palavras-chave, presença de anexos e remetente. Esse modelo pode ser usado por provedores de e-mail e usuários individuais para filtrar e separar automaticamente e-mails indesejados, economizando tempo e protegendo contra ameaças e golpes.

Ainda na visão de Kotsiantis (2007), no caso do aprendizado de dados não supervisionado, envolve o reconhecimento de padrões sem o envolvimento de um atributo alvo. Ou seja, todas as variáveis utilizadas na análise são usadas como entradas e por causa da abordagem, as técnicas são adequadas para clusterização e mineração de associação. De acordo com Hofmann (2001), os algoritmos de aprendizado não supervisionado são adequados para criar os rótulos nos dados que são posteriormente usados para implementar tarefas de aprendizado supervisionado. Ou seja, algoritmos não supervisionados de clusterização identificam agrupamentos inerentes dentro dos dados não rotulados e subsequentemente atribua um rótulo a cada valor de dados. Por outro lado, os algoritmos não supervisionados de associação tendem a identificar regras que representam relacionamentos com precisão entre atributos.

Um exemplo de aplicação de algoritmos de clusterização é a segmentação de clientes em uma empresa. Utilizando dados dos clientes, como idade, gênero, renda e histórico de compras, algoritmos de clusterização são usados para agrupar os clientes em segmentos com características semelhantes. Esses segmentos representam grupos de clientes com comportamentos de compra e características demográficas similares. Com essa segmentação, a empresa pode personalizar suas estratégias de marketing, direcionando campanhas específicas, ofertas e mensagens para cada segmento, visando aumentar a eficiência das campanhas, melhorar o atendimento ao cliente e impulsionar as vendas. Outro exemplo, mas em relação aos algoritmos de associação, é a análise de cestas de compras em um

supermercado. Usando algoritmos de associação é possível identificar padrões e associações entre os produtos frequentemente comprados juntos pelos clientes. Isso permite que o supermercado otimize o *layout* da loja, a organização dos produtos e as estratégias de *merchandising*. A análise de associação ajuda a identificar produtos complementares, sugerir promoções cruzadas e melhorar a experiência de compra, impulsionando as vendas e facilitando a tomada de decisões estratégicas.

A partir das definições de *Data Mining* e *Machine Learning* citadas anteriormente, é interessante destacar a relação entre os dois temas, já que são campos inter-relacionados e que compartilham muitas técnicas e abordagens semelhantes. . A relação entre os tipos de *Machine Learning*, tarefas de *Data Mining* e os principais algoritmos utilizados podem ser visualizadas no Quadro 1.

Quadro 1: Algoritmos associados às tarefas de *Data Mining* e tipos de *Machine Learning*

Machine Learning	**Tarefas de *Data Mining***	**Principais algoritmos**
Supervisionado	Classificação	Árvore de decisão
		Naive Bayes
		Máquina de vetores de suporte (SVM)
		K-vizinhos mais próximos (KNN)
	Predição	Regressão linear
		Regressão logística
		Regressão de Ridge
Não supervisionado	Segmentação	K-means
		DBSCAN
		Clusterização aglomerativa hierárquica
	Análise de dependência	Apriori
		FP-growth
		Eclat

Fonte: Adaptado de Mahesh (2018)

Na prática, *Machine Learning* e *Data Mining* são frequentemente usados juntos em uma abordagem integrada para resolver problemas complexos de análise de dados. Ao

identificar qual tipo de tarefa de *Data Mining* o problema está envolvido, é necessário utilizar os algoritmos de *Machine Learning* adequados para solucionar esse problema.

2.2 FERRAMENTAL COMPUTACIONAL, ESTATÍSTICO E MATEMÁTICO EMPREGADO EM ALGORITMOS DE ASSOCIAÇÃO E CLUSTERIZAÇÃO.

Em relação ao ferramental computacional, Stancin (2019) destaca que o uso de Python na área de ciência de dados tem atingido níveis sem precedentes, especialmente na área de ferramentas e bibliotecas disponíveis gratuitamente. Em enquete publicada pelo portal oficial KDNuggets (2018), sobre a categoria "Top Analytics, Data Science, *Machine Learning* Tools", constatou-se que o Python é utilizado por 65,2% de cerca de 2.000 participantes, em comparação com 52,7% para RapidMiner e 48,5% para R, seus dois principais concorrentes. Em perspectiva prática, nos últimos três anos, Python tornou-se a linguagem de programação de escolha para a comunidade de Data Science, com R sendo a segunda escolha.

Demšar (2013) contribui com a explicação que Python é uma linguagem de script com sintaxe clara e simples, o que também o tornou popular na educação. Sua execução relativamente lenta pode ser contornada usando bibliotecas que implementam as tarefas computacionalmente intensivas em linguagens de baixo nível. Saabith (2021) complementa ao dizer que Python é uma linguagem fácil de aprender, fácil de depurar, amplamente usada, orientada a objetos, de código aberto e de alto desempenho, e há muitos outros benefícios na linguagem. Ele foi construído com bibliotecas que são usadas por programadores todos os dias na resolução de problemas.

Porcu (2018) traz em seu livro "*Python for Data Mining Quick Syntax Reference*" as principais bibliotecas disponíveis e utilizadas para *Data Mining* e *Machine Learning*, sendo elas:

a. Pandas É uma biblioteca Python de código aberto que contém várias ferramentas para importar, gerenciar e manipular dados. Ele tem uma série de recursos de alto nível para manipular, reorganizar e escanear dados estruturados, incluindo divisão, gerenciamento de valores ausentes, reestruturação de dados, extração de partes do conjunto de dados, importação e análise de dados da Web. Pandas é uma das mais importantes bibliotecas de

Data Mining. É possível ler, importar, organizar, manipular e realizar cálculos estatísticos básicos em dados estruturados.

b. NumPy: No início do desenvolvimento do Python, não existiam muitas ferramentas que conseguiam atender as necessidades dos programadores, logo eles mesmos desenvolveram novas ferramentas para computação científica. A primeira tentativa resultou no pacote *Numeric*, que foi desenvolvido em 1995, seguido por um pacote alternativo chamado Numarray. A fusão das funções desses dois pacotes ganhou vida em 2006 com o NumPy. NumPy significa *Numeric Python*, é uma biblioteca de código aberto dedicada à computação científica, especialmente no que diz respeito à gerenciamento de matriz. A principal estrutura de dados fornecida pelo NumPy é o ndarray (*n-dimensional array*), que permite armazenar e manipular dados de forma eficiente em arrays de dimensões múltiplas. Esses arrays podem conter elementos de um único tipo de dados, o que permite um desempenho otimizado e operações vetorizadas.

c. SciPy: É um dos pacotes mais importantes para análise matemática e estatística em Python e está intimamente ligada ao NumPy. SciPy contém mais de 60 funções estatísticas organizadas em famílias de módulos.

d. Matplotlib: A criação de gráficos é uma etapa importante na análise exploratória e um dos primeiros estágios na análise de dados. É possível usar o Matplotlib para construir uma variedade de gráficos analíticos que exibem os dados de maneiras diferentes.

e. Scikit-Learn: O Scikit-learn é um dos pacotes mais importantes e mais usados para *Machine Learning* com Python. Pedregosa (2011) complementa ao dizer que o Scikit-learn expõe uma ampla variedade de algoritmos de *Machine Learning*, supervisionados e não supervisionados, usando uma interface consistente e orientada a tarefas, permitindo assim uma comparação fácil de métodos para um dada aplicação. Uma vez que depende do ecossistema científico Python, pode ser facilmente integrado em aplicações fora da faixa tradicional de análise de dados estatísticos.

Nesse projeto tem-se como interesse o uso de *Data Mining* para a aplicação de algoritmos de associação como o *Apriori* e o *FP-growth*, e clusterização como o *K-means*, portanto o ferramental matemático e estatístico em relação a essas técnicas será discutido a seguir. Agrawal (1993) contribui com os conceitos acerca da regra de associação. Seja Z = I1, I2,..., Im um conjunto de atributos binários, chamados itens. Seja T um banco de dados de

transações. Cada transação t é representada como um vetor binário, com t[k] = 1 se t comprou o item Ik, e t[k] = O caso contrário. Há uma tupla[7] no banco de dados para cada transação. Seja X um conjunto de alguns itens em Z. Dizemos que uma transação t satisfaz X se para todos os itens Ik em X, t[k] = 1.

Por uma regra de associação, queremos dizer uma implicação da forma X → Ij, onde X é um conjunto de alguns itens em Z, e Ij é um único item em Z que não está presente em X. A regra X → Ij é satisfeita no conjunto de transações T com o fator de confiança O < c < 1, se pelo menos c% das transações em T que satisfazem X também satisfazem Ij. É usada a notação X → Ij | c para especificar que a regra X → Ij tem um fator de confiança de c. Dado o conjunto de transações T, o interesse é na geração de todas as regras que satisfaçam certas restrições de duas formas diferentes:

a. Restrições sintáticas: Essas restrições envolvem itens que podem aparecer em uma regra. Por exemplo, podemos estar interessados apenas em regras que tenham um item específico aparecendo no consequente, ou regras que tenham um item específico aparecendo no antecedente. Combinações das restrições acima também são possíveis, podemos solicitar todas as regras que tenham itens de algum conjunto de itens predefinido X aparecendo em consequente e itens de algum outro conjunto de itens Y aparecendo no antecedente.

b. Restrições de suporte: essas restrições dizem respeito ao número de transações em T que suportam uma regra. O suporte para uma regra é definido como a fração de transações em T que satisfazem a união de itens em consequente e antecedente da regra, sendo o seu cálculo feito dividindo o número de transações que contêm o conjunto de itens ou item pelo número total de transações no conjunto de dados. Suporte não deve ser confundido com confiança, embora a confiança seja uma medida da força da regra, suporte corresponde à significância estatística. Além da significância estatística, outra motivação para restrições de suporte vem do fato de que geralmente há interesse apenas em regras com suporte acima de um limite mínimo por motivos comerciais. Se o suporte não estiver de acordo com um valor mínimo definido, significa que a regra de associação não vale a pena consideração ou que é simplesmente menos preferido.

---

[7] De acordo com o site do IME (2020), tupla é uma é uma sequência imutável de valores de qualquer tipo. Para criar uma tupla, lista-se uma sequência de valores separados por vírgulas e, opcionalmente, entre parênteses.

Um dos algoritmos mais comuns utilizado para regras de associação é chamado de *Apriori*, e Panjaitan (2019) contribui com a explicação do algoritmo, demonstrando como funcionam suas etapas.

Etapa a: É formado um conjunto de itens candidatos k a partir de uma combinação de itens obtidos de uma iteração feita no conjunto de itens k-1 anteriormente. Essa iteração percorre todas as transações do conjunto de dados e conta a ocorrência de cada item individual, se caso k >1, ele conta a ocorrência do conjunto de itens do tamanho correspondente ao tamanho de k.

Etapa b: É realizado o cálculo do suporte de cada candidato do conjunto de itens k. O suporte de cada um dos candidatos é obtido pela varredura do banco de dados para calcular o número de transações que contêm os itens candidatos.

Etapa c: É definido um padrão de alta frequência. Um padrão de alta frequência que contém k itens é determinado a partir do conjunto de itens candidatos cujo suporte é maior que o suporte mínimo.

Etapa d: Se nenhum novo padrão de alta frequência for obtido, todo o processo é interrompido. Se não, então k é adicionado o valor de 1 e o processo volta para a etapa a.

Um exemplo de aplicação do algoritmo é demonstrado por Panjaitan (2019) e ajuda a ilustrar o funcionamento do *Apriori* de forma prática. Neste exemplo foi aplicado o algoritmo *Apriori* em um conjunto de dados de transações feitas em um restaurante chamado Cafe Bojack *Coffee Shop* na cidade de Medan, Indonésia. No Quadro 2 é possível visualizar a base de dados de transações do restaurante.

Quadro 2: Base de dados das transações

Transação	Itens comprados
1	Chocolate escuro, chá doce, arroz frito, arroz, ovo frito, nugget de frango.
2	Sprite, chocolate escuro, cappuccino, teh tarik, sopa de noodle, noodle frito, nutri sari, chá doce, banana assada
3	Arroz frito, chocolate, berinjela, chá doce, chocolate escuro, noodle frito
4	Milo, teh tarik, banana assada, banana frita
5	Noodle com molho, arroz frito, chá doce, leite
6	Teh tarik, kuku bima, chá doce, batata frita, noodle frito, arroz frito
7	Leite, arroz frito, chá de limão, cappuccino, água

8	Arroz frito, chá de limão, milo, kuku bima, noodle frito
9	Capputino, milo, martabak mie, leite, sopa de noodle
10	Batata frita, chá com leite, arroz frito, nutri sari, cappuccino, chocolate escuro
11	Chá doce, cappuccino, milo, batata frita, arroz frito, banana grelhada
12	Noodles frito, noodle martabak, noodles, arroz, chá tarik, chá doce, chá com leite
13	Sopa de noodle, laranja, arroz, cappuccino
14	Nutri sari, cappuccino, milo, berinjela, arroz frito, café, abacate
15	Chá doce, nasi goreng, coca-cola, banana assada
16	Nutri sari, teh tarik, cappuccino, abacate, noodles frito, arroz frito, sopa de noodle
17	Chá de limão, teh tarik, leite, arroz frito, sopa de noodle
18	Cappuccino, chá doce, milo, noodles, arroz, chocolate escuro
19	Chocolate escuro, capputino, arroz frito, sopa de noodle
20	Arroz frito, teh tarik, cappuccino, noodle frito, chocolate escuro

Fonte: Adaptado de Panjaitan (2019)

Em seguida foi criado o conjunto de itens candidatos com k=1 a partir da frequência de aparição de cada item na base de transações. Pode ser observado no Quadro 3.

Quadro 3: Conjunto de itens candidatos k = 1, e seu respectivo suporte.

Item	Frequência do item candidato
Água	1/20*100%=5%
Abacate	2/20*100%=10%
Chocolate	1/20*100%=5%
Coca-cola	1/20*100%=5%
Nugget de frango	1/20*100%=5%
Cappuccino	11/20*100%=55%
Chocolate escuro	7/20*100%=35%
Batata frita	3/20*100%=15%
Kuku bima	2/20*100%=10%
Café	1/20*100%=5%
Chá de limão	3/20*100%=15%

Noodles frito	7/20*100%=35%
Sopa de noodles	9/20*100%=45%
Milo	6/20*100%=30%
Noodles martabak	2/20*100%=10%
Ovo frito	1/20*100%=5%
Arroz frito	14/20*100%=70%
Arroz	4/20*100%=20%
Nutri sari	4/20*100%=20%
Laranja	1/20*100%=5%
Banana grelhada	4/20*100%=20%
Banana frita	1/20*100%=5%
Sprite	2/20*100%=10%
Leite	3/20*100%=15%
Berinjela	2/20*100%=10%
Chá doce	9/20*100%=45%
Chá com leite	2/20*100%=10%
Teh tarik	7/20*100%=35%

Fonte: Adaptado de Panjaitan (2019)

Posteriormente foi definido que o nível de suporte mínimo seria de 15% e o nível de confiança de 50%. Dessa maneira o conjunto de itens candidatos com k=2 foi criado, e pode ser observado no Quadro 4.

Quadro 4: Conjunto de itens candidatos k = 2, e seu respectivo suporte.

Item 1	Item 2	Frequência dos itens candidatos
Arroz frito	Cappuccino	7/20*100%=35%
Arroz frito	Sopa de noodle	4/20*100%=20%
Arroz frito	Chá doce	6/20*100%=30%
Arroz frito	Chocolate escuro	5/20*100%=25%
Arroz frito	Teh tarik	4/20*100%=20%
Arroz frito	Noodles frito	5/20*100%=25%
Arroz frito	Milo	3/20*100%=15%
Arroz frito	Nutri sari	3/20*100%=15%

Arroz frito	Leite	3/20*100%=15%
Arroz frito	Batata frita	3/20*100%=15%
Cappuccino	Sopa de noodle	6/20*100%=30%
Cappuccino	Chá doce	3/20*100%=15%
Cappuccino	Chocolate escuro	5/20*100%=25%
Cappuccino	Teh tarik	3/20*100%=15%
Cappuccino	Noodles frito	3/20*100%=15%
Cappuccino	Milo	4/20*100%=20%
Cappuccino	Nutri sari	4/20*100%=20%
Sopa de noodle	Chá doce	4/20*100%=20%
Sopa de noodle	Chocolate escuro	3/20*100%=15%
Sopa de noodle	Teh tarik	4/20*100%=20%
Sopa de noodle	Noodle frito	3/20*100%=15%
Sopa de noodle	Arroz	3/20*100%=15%
Sopa de noodle	Leite	3/20*100%=15%
Chá doce	Chocolate escuro	4/20*100%=20%
Chá doce	Teh tarik	3/20*100%=15%
Chá doce	Noodle frito	4/20*100%=20%
Chá doce	Arroz	3/20*100%=15%
Chá doce	Banana grelhada	3/20*100%=15%
Chocolate escuro	Noodle frito	3/20*100%=15%

Fonte: Adaptado de Panjaitan (2019)

Depois que todos os padrões de alta frequência foram encontrados, as regras de associação que atendem aos mínimos requisitos são selecionadas, e posteriormente as mais importantes são utilizadas para tomadas de decisão. No Quadro 5 é possível observar o resultado da aplicação do *Apriori*.

Quadro 5: Principais resultados de associação obtidos com a aplicação do Apriori.

Conjunto de itens	Suporte	Confiança
Se você compra arroz frito, você irá comprar cappuccino	35%	50%
Se você compra cappuccino,	30%	54,55%

você irá comprar sopa de noodle		
Se você compra teh tarik, você irá comprar noodle frito	25%	71,43%

Fonte: Adaptado de Panjaitan (2019)

Com esse exemplo é possível perceber como funciona o *Apriori* e como ele pode ser usado para analisar dados de transações de vendas e identificar padrões de compra, permitindo descobrir quais itens são frequentemente comprados juntos, ajudando a empresa a otimizar a organização da loja, definir estratégias de promoção e criar pacotes de produtos atraentes para melhorar seu negócio.

Porém uma questão importante é destacada por Aggarwal (2001). É difícil para um usuário adivinhar quantas regras podem satisfazer um dado nível de suporte e confiança. Normalmente, pode-se estar interessado em apenas algumas regras. Isso torna o problema ainda mais difícil, pois um usuário pode precisar executar a consulta várias vezes para encontrar níveis apropriados de mínimo suporte e mínima confiança para minar as regras. Na verdade, na maioria dos casos, o número de regras redundantes é significativamente maior do que o número de regras essenciais e ter muitas regras redundantes anulam o objetivo principal do *Data Mining* em primeiro lugar. Han (2004) complementa ao dizer que o algoritmo *Apriori* alcança um bom desempenho obtido pela redução (possivelmente significativa) do tamanho dos conjuntos de candidatos. No entanto, em situações com grande número de padrões, padrões longos ou limites mínimos de suporte bastante baixos, um algoritmo do tipo *Apriori* pode encontrar dificuldades.

Dessa forma, outros algoritmos mais eficientes para regras de associação foram criados. Um exemplo de algoritmo é o *FP-Growth* (F*requent Pattern Growth)*, que é discutido por Borgelt (2005), ele diz que o algoritmo se baseia em uma estrutura de dados chamada *FP-Tree* (*Frequent Pattern Tree*) para representar os padrões frequentes de maneira compacta, o que melhora a eficiência de processamento em comparação com o *Apriori*. Esse processamento é demonstrado na figura 1, que mostra um exemplo de banco de dados de transações na coluna denominada “Itens comprados”. É definido um nível de suporte igual a 3, e posteriormente é criada uma tabela com a frequência dos itens em ordem decrescente, já desconsiderando os itens com nível de suporte inferior ao determinado. Depois de fazer isso e

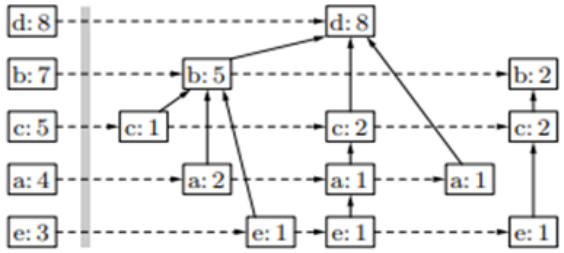
classificar os itens em cada transação de forma descendente referenciando suas frequências obtêm-se um banco de dados reduzido mostrado na coluna “Itens comprados considerando suporte mínimo”.

Figura 1: Exemplo de processamento da FP-Tree.

		Suporte mín = 3
Id transação	**Itens comprados**	**itens comprados considerando suporte mínimo**
1	(f,a,d)	(a,d)
2	(a,d,c,e)	(a,d,c,e)
3	(b,d)	(b,d)
4	(b,c,d)	(b,c,d)
5	(b,c)	(b,c)
6	(b,a,d)	(b,a,d)
7	(b,e,d)	(b,e,d)
8	(f,c,d)	(c,d)
9	(b,e,c,g)	(b,e,c)
10	(b,a,d)	(b,a,d)

Item	**Frequência**
d	8
b	7
c	5
a	4
e	3
f	2
g	1

Árvore FP



Fonte: Adaptado de Borgelt (2005)

Depois que todos os itens infrequentes abaixo do nível de suporte tiverem sido excluídos do banco de dados de transações, ele é transformado em uma árvore FP. Uma árvore FP é construída de forma a capturar as relações de frequência entre os itens. Ou seja, cada caminho representa um conjunto de transações que compartilham o mesmo item, e cada nó corresponde a um item. Além disso, todos os nós referentes ao mesmo item são vinculados em uma lista, de modo que todas as transações que contenham um determinado item podem ser facilmente encontradas e contadas percorrendo esta lista. A lista pode ser acessada através de um elemento *head*, que também indica o número total de ocorrências do item na base de dados. Como exemplo, a figura 1 mostra a árvore FP para o conjunto de itens comprados considerando o suporte mínimo.

Para cada seção um nó da árvore FP é criado e rotulado com o item correspondente à seção. Cada seção é então processada recursivamente, dividida em subseções, e então uma nova camada de nós (uma por subseção) é criada. Tal abordagem ainda tem a vantagem de

que o banco de dados de transações não precisa ser carregado de uma forma simples (por exemplo, como uma lista de inteiros *arrays*) na memória principal. Como apenas uma transação é processada por vez, apenas a representação da árvore FP e uma nova transação estão na memória principal. Isso geralmente economiza espaço, porque uma árvore FP é frequentemente uma representação muito mais compacta de um banco de dados de transações. Dessa forma é possível perceber que ao utilizar a árvore FP, o algoritmo *FP-Growth* é capaz de encontrar rapidamente os conjuntos de itens frequentes, facilitando a descoberta de regras de associação relevantes. A compactação da árvore e as operações otimizadas de busca tornam o *FP-Growth* uma abordagem escalável e eficiente para a mineração de regras de associação em grandes conjuntos de dados.

Em relação aos algoritmos de clusterização, Nunes (2016) contribui inicialmente com alguns conceitos básicos sobre o clustering e a sua notação. Um clustering C = {C1, C2, . . . , Ck} consiste numa partição de um conjunto de dados em subconjuntos Ci (chamados clusters), onde os elementos de cada cluster apresentam alguma semelhança entre si e diferem dos elementos que se encontram nos outros clusters. Considere-se um conjunto de dados $D = \{x\}_{i=i}^{n}$ com n pontos num espaço de dimensão d. Seja C = {C1, C2, . . . , Ck} um clustering do conjunto de dados. Para cada cluster Ci existe um ponto zi que o representa e que é designado por centroide. Existem diferentes formas de obtê-lo, mas neste trabalho, será considerado o centroide obtido através da média de todos os pontos pertencentes ao cluster, isto é:

$$z_i = \frac{1}{n_i} \sum_{x_j \in C_i} x_j$$

Onde ni é o número de elementos de cada cluster.

Na aplicação dos algoritmos de clusters é necessário avaliar iterativamente a qualidade do clustering e, para tal, recorreu-se a uma função de semelhança. Esta denomina-se de soma dos erros quadrados (*Sum of Squared Errors*) SSE e mede a semelhança de cada elemento através da sua distância aos centroides:

$$SSE(C) = \sum_{i=1}^{k} \sum_{x_j \in C_i} D(x_j, z_i)$$

Esta função também pode ser vista como uma avaliação de dispersão dos elementos dentro de um cluster. Com a explicação dos conceitos sobre clusterização, se torna possível

realizar a descrição do algoritmo K-means que será utilizado neste trabalho. Para Nunes (2016) o K-means aplica uma abordagem gananciosa para encontrar o clustering que minimiza o SSE, convergindo para uma solução local em vez de um clustering ótimo global. Este algoritmo começa por distribuir aleatoriamente os pontos do conjunto D em k clusters e calcula os centroides através da média dos pontos do cluster Ci. Em seguida aplicam-se duas fases iterativamente: a atribuição de clusters e a atualização dos centroides.

Na atribuição de clusters, cada ponto xj ∈ D é associado ao cluster que tem o centroide zi mais próximo do ponto, isto é, xj é atribuído ao cluster Cj∗ quando:

$$j* = argmin \sum_{i=1}^{k} \sum_{j=1}^{n} \|x_j - z_i\|^2$$

Essa equação, procura o índice j∗ que minimiza a variabilidade intra-cluster, ou seja, a soma total dos quadrados das distâncias entre os pontos de dados e os centróides nos clusters. Em seguida atualizam-se os centroides através da média de todos os pontos que se encontram no cluster Ci. Estas duas fases são realizadas iterativamente até alcançarem um mínimo local, isto é, o K-means converge se os centroides não mudarem após uma iteração.

2.3 APLICAÇÕES DE REGRAS DE ASSOCIAÇÃO NO E-COMMERCE

De acordo com Urne (2020), atualmente no mercado globalizado, a tecnologia desempenha um papel vital em todas as atividades presentes em uma cadeia produtiva. Todos os tipos de tecnologias são introduzidos nas organizações e também no mercado. Essas tecnologias ajudam organizações e clientes tanto para negócios bidirecionais quanto comunicação. Uma das melhores tecnologias utilizadas é o e-commerce. Nas palavras de Laudon (2008), e-commerce é o uso da internet e da web para realizar negócios.

O *e-commerce* tem experimentado um grande avanço nas últimas décadas, impulsionado principalmente pelo aumento da conectividade à Internet, da popularidade dos dispositivos móveis e da crescente confiança dos consumidores em fazer compras online. Esse setor tem se beneficiado bastante do uso de ciência de dados para obter melhores resultados.

De acordo com Pradeep (2020), a ciência de dados é usada no setor e-commerce de várias maneiras. Alguns dos principais aspectos são destacados a seguir.

a. Análise da cesta de mercado: Esta é uma das abordagens mais antigas e também é conhecida como análise de associação em termos de mineração de dados. O conceito principal é encontrar o item/itens relacionados aos itens apreciados/comprados pelo cliente. Isso é realizado ao gerar regras de associação frequentes. Muitos parâmetros como *support, confidence, lift* e *leverage* são utilizados. Onde o *support* é descobrir todas as transações e o número de transações em que o item está presente. *Confidence* é sobre a probabilidade de comprar/curtir um item junto com outro item conhecido que o cliente já comprou/gostou anteriormente. *Lift* é a métrica usada para medir a força da associação entre dois itens. É calculado comparando a frequência com que dois itens são comprados juntos em relação à frequência esperada de compra, assumindo que a compra de um item é independente da compra do outro.

b. Otimização de preço: É outra área em que as abordagens de ciência de dados e aprendizado de máquina são muito usadas. No início da era do e-commerce, o preço de um item era fixado com base em parâmetros como margem de lucro, preço do concorrente e preço sugerido decidido pelo vendedor. Usando modelos de aprendizado de máquina como regressão linear univariada, a relação entre as vendas e o custo dos itens pode ser derivada. Para ter uma compreensão aprofundada, diversas regressões podem ser usadas, como a linear ou a regressão polinomial, que podem ser usadas para decidir a relação entre o custo de um item com base em localização do cliente, frequência de compra, sazonalidade.

c. Mecanismos de recomendação: Essa é uma das áreas em que o aprendizado de máquina é mais usado. São feitas recomendações de itens relacionados aos itens curtidos pelos usuários. Existem muitas variações de sistemas de recomendação na literatura. Os três principais tipos de mecanismos de recomendação são baseados em popularidade, conteúdo e colaboração. Em recomendações baseadas em popularidade, o interesse do usuário não é levado em consideração. Os itens mais populares são classificados de acordo com o local e hora, e os principais N itens são recomendados ao usuário. Muitas vezes, esse sistema de recomendação simples funciona bem porque um item popular é apreciado por todos. A popularidade geralmente é calculada usando parâmetros como o número de itens comprados/consumidos em um período de tempo. O segundo tipo de recomendador é a filtragem baseada em conteúdo. Se um usuário gostou/comprou um item, outros itens semelhantes ao item interessado é recomendado ao usuário. A similaridade entre os itens é

calculada usando as informações de metadados. Por exemplo, se um usuário compra/gosta de livros de um autor, outros livros do mesmo autor lhe serão recomendados. Há muitas maneiras de saber se um usuário gostou de um item. 1) Avaliação explícita dada por um usuário a um item. 2) Analisar as revisões usando técnicas de processamento de linguagem natural. 3) Recomendar itens que o usuário visualizou. 4) Analisar *logs*[8] de cliques, como a categoria de informação preferida do usuário. 5) Parâmetros como o preço também podem ser levados em consideração. Se um usuário tende a comprar itens com desconto, então os itens com grandes descontos podem ser recomendados. A filtragem colaborativa é outro tipo de sistema de recomendação. Ao contrário dos recomendadores baseados em conteúdo, a similaridade entre os usuários é levada em consideração. Nesta abordagem, um conjunto semelhante de usuários é encontrado. Em seguida, os itens adquiridos ou apreciados por um usuário são recomendados para usuários semelhantes. As medidas de similaridade usadas pela abordagem de conteúdo e usadas pela abordagem de filtragem colaborativa são quase as mesmas, porém com modificações pequenas.

d. Análise de imagem: Este é um conceito novo e pode ser usado para entender o usuário. No caso de sites de comércio eletrônico de moda, por exemplo, as seguintes informações sobre o interesse do usuário podem ser conhecidas. 1) Cor do vestido 2) Desenho do vestido 3) Fundo da imagem 4) Efeito de iluminação. 5) Apresentação do vestido pela modelo. A análise de imagem também pode ser usada para encontrar itens semelhantes. Na abordagem tradicional, itens semelhantes são pesquisados com base nos metadados do item e nas informações do usuário. Para estender ainda mais o conceito, imagens semelhantes podem ser pesquisadas usando *deep learning*. A rede neural convolucional é uma abordagem de *deep learning* especialmente projetada para lidar com informação em imagens.

e. Proteção contra fraudes: Com uma enorme quantidade de dados, é impossível rastrear compras fraudulentas manualmente. Por exemplo, um cliente pode encomendar o produto apenas para fazer duplicatas dele e devolvê-lo. Da mesma maneira, alguns produtos vendidos em *e-commerce* podem não ser para revenda, mas as pessoas podem comprar itens

[8] De acordo com a Amazon Web Service (2023), Arquivos de log são arquivos gerados por software que contêm informações sobre as operações, as atividades e os padrões de uso de uma aplicação, servidor ou sistema de TI. Eles incluem um registro histórico de todos os processos, eventos e mensagens junto com dados descritivos adicionais, como carimbos de data/hora, para contextualizar essas informações..

com desconto com vários endereços e revendê-los para lucrar em cima do custo original. Sites de comércio eletrônico também podem identificar clientes que optam pelo pagamento à vista e devolvem repetidamente o produto, gerando prejuízo para as empresas. Esses tipos das transações podem ser identificados usando análise de dados para melhorar a qualidade dos negócios.

f. Pesquisa de itens: A simples pesquisa por palavras-chave é um método tradicional de pesquisa que não é adequado para sites de comércio eletrônico. Parâmetros adicionais, como contexto do usuário, localização e estação do ano, são úteis para descobrir bons padrões.

Um exemplo de aplicação de análise de cesta de mercado pode ser percebido através de um estudo realizado por Panjaitan (2019). Nesse estudo foi aplicado o algoritmo Apriori em uma base de dados de transações de um restaurante, buscando criar regras de associação e identificar padrões entre os itens vendidos para entender melhor o comportamento de compra dos clientes e poder tomar decisões estratégicas. A base de dados e as etapas de execução do algoritmo estão destacadas no capítulo 2.2, e os resultados da análise da cesta de mercado serão demonstrados a seguir. Após a aplicação do algoritmo as principais regras de associação foram encontradas de acordo com os níveis mínimos de suporte e confiança, e estão destacadas no Quadro 6.

Quadro 6: Principais resultados de associação obtidos com a aplicação do Apriori.

Conjunto de itens	Suporte	Confiança
Se você compra arroz frito, você irá comprar cappuccino	35%	50%
Se você compra cappuccino, você irá comprar sopa de noodle	30%	54,55%
Se você compra teh tarik, você irá comprar noodle frito	25%	71,43%

Fonte: Adaptado de Panjaitan (2019)

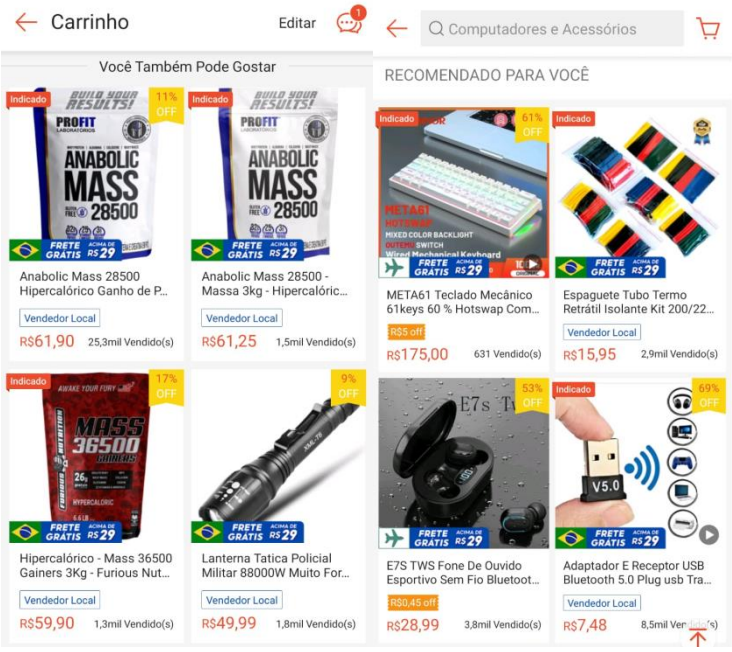
A partir desses resultados algumas decisões estratégicas foram criadas. Foi possível realizar um melhor planejamento dos suprimentos, já que esses produtos estão fortemente associados uns aos outros, foi possível ajustar os níveis de estoque desses itens para garantir disponibilidade adequada e evitar estoques excessivos ou insuficientes. Também foi possível

realizar estratégias de vendas cruzadas para ajudar a aumentar as vendas e o valor médio das transações, pois ao oferecer sugestões de produtos relacionados, existe um incentivo para que os clientes adquiram ambos os produtos.

Outro exemplo que ajuda a demonstrar os resultados de uma dessas aplicações pode ser observado a partir de um estudo de caso realizado por Dias (2008). Nessa pesquisa, foi feita uma análise dos resultados obtidos a partir da implantação e do uso de um sistema de recomendações, que pode ser formado utilizando os resultados de algoritmos de associação como o *Apriori* e o *FP-Growth* citados anteriormente como base para a criação, na empresa LeShop (www.LeShop.ch), que mudou seu nome para Migros (www.Migros.ch) em novembro de 2020, e é considerada líder no mercado Suíço de supermercados online. A avaliação do desempenho do sistema de recomendação utilizado ocorreu durante um período de 21 meses, de maio de 2006 a janeiro de 2008, e teve o foco em analisar o valor agregado ao negócio. A análise abrange a penetração do comprador, assim como a receita direta e indireta gerada pelo sistema de recomendação.

As ferramentas utilizadas foram de um sistema de recomendação *In-store*, lançado em 2 de Outubro de 2007, e um de *Checkout*, lançado em 5 de maio de 2006, que foram integrados na sessão de compras típica, como é ilustrado no fluxo apresentado na Figura 8. O sistema de recomendação *In-store* fica disponível para os compradores na parte inferior da página inicial de cada categoria, e os produtos recomendados são relacionados à categoria escolhida. O sistema *Checkout* fica disponível para o comprador na seção do carrinho de compras, em que são recomendados produtos relacionados aos produtos que estão nessa seção.

Figura 2: Exemplo de sistema de recomendação de *Checkout* e *Instore*

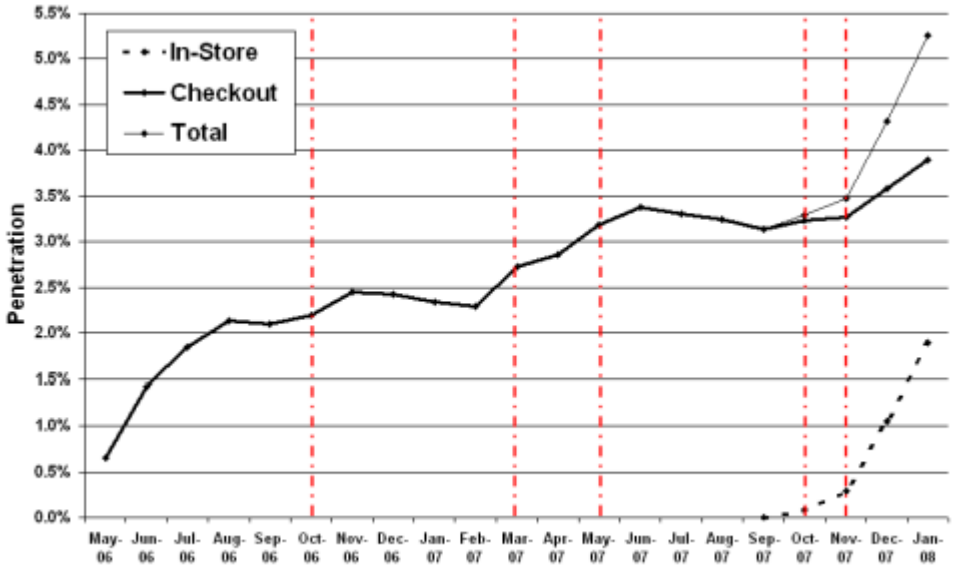


Fonte: Autor

O sistema teve seus dados atualizados diversas vezes durante o período de 21 meses, e a data dessas atualizações estão representadas como linhas vermelhas nos gráficos de análise dos resultados que são apresentados na Figuras 3, 4 e 5 a seguir. Os gráficos abaixo ajudam a demonstrar os resultados através de indicadores que são explicados a seguir.

a. Penetração do comprador: Foi feita uma análise de como o uso de cada sistema de recomendação se disseminou ao longo do tempo. Esta é a medida mais fundamental do valor de um sistema de recomendação, porque não importa quão boas sejam as recomendações, se os usuários (no caso os compradores) não utilizam o sistema, ele não poderá gerar nenhum valor para o negócio. A Figura 3 demonstra a penetração de cada um dos sistemas recomendadores ao longo do tempo. A medida de penetração (ou seja, a unidades no eixo y) usada, é a proporção de compradores que aceitou pelo menos uma das recomendações, representada como uma porcentagem de todos os compradores que compraram pelo menos um item da LeShop.

Figura 3: Penetração do sistema de recomendação pelo tempo

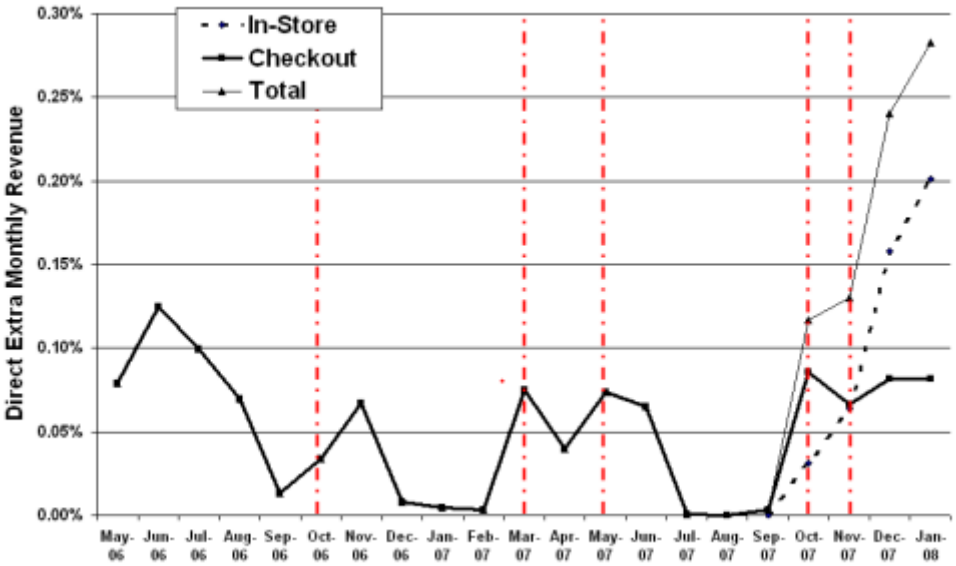


Fonte: Dias (2008)

Observa-se que atualizar os arquivos do recomendador de *checkout*, resulta em um aumento consistente do número de novos compradores utilizando o sistema de recomendação. Além disso, o uso do sistema de recomendação continua a crescer com o passar do tempo, e isso pode ocorrer porque os compradores estão cada vez mais confortáveis com a nova tecnologia. Ao contrário do recomendador de checkout, a penetração do recomendador in-store aumentou rapidamente desde o seu lançamento. Os pesquisadores acreditam que isso ocorre porque o a exposição do recomendador in-store é, em média, nove vezes o do recomendador de checkout., já que ele está disponível em todas páginas iniciais das categorias de produto, e o de checkout somente uma vez quando o comprador decide finalizar suas compras.

b. Aumento direto da receita: Em seguida, foi analisada a receita extra direta gerada pelos sistemas de recomendação. A receita extra direta é simplesmente a quantidade total de dinheiro que os compradores gastaram na compra de itens recomendados pelos sistemas. Uma representação gráfica da receita extra direta computada mensalmente e apresentada como uma porcentagem do faturamento mensal total da LeShop é dada na Figura 4.

Figura 4: A receita extra mensal direta gerada pelos sistemas de recomendação, apresentada como uma porcentagem do volume de negócios total mensal.



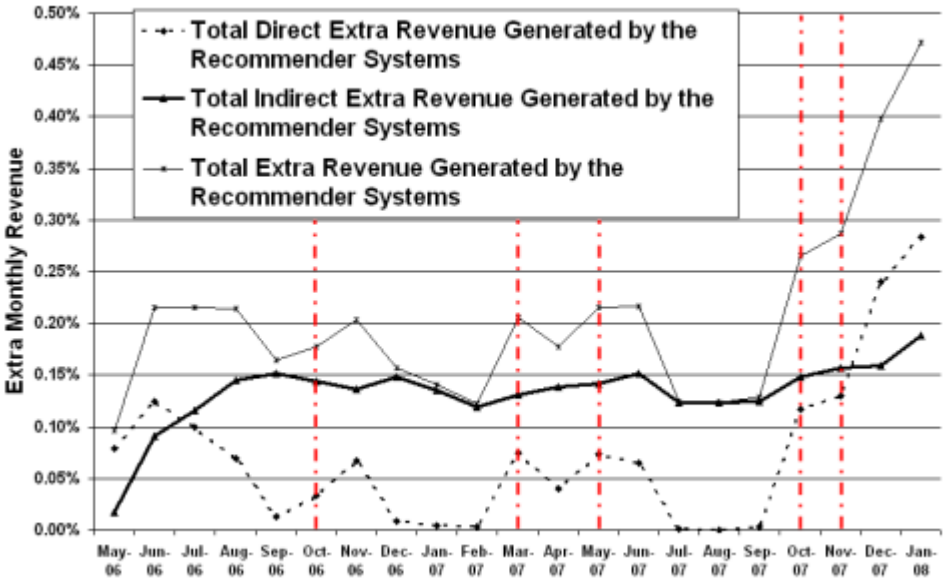
Fonte: Dias (2008)

Uma lição importante a retirar desta análise é que é imprescindível manter os arquivos do modelo atualizados, para manter um fluxo constante de receita extra direta. É possível ver o desempenho dos sistemas cair muito rapidamente até que sejam atualizados novamente usando os dados mais recentes. Vê-se também que, dada a sua maior exposição (em média 9 vezes maior), o recomendador in-store é capaz de gerar significativamente mais receita extra direta do que o recomendador de *checkout*. Também é importante observar que, embora o maior valor percentual alcançado seja de 0,30%, o que parece uma pequena quantidade, quando se considera a magnitude do faturamento mensal de um negócio de comércio eletrônico bem-sucedido, como é o caso, o valor real gerado é substancial.

c. Aumento indireto de receita: A receita extra indireta gerada pelos sistemas de recomendação compreende dois componentes. O primeiro componente é a quantia total de dinheiro que os compradores gastaram em compras repetidas de itens apresentados a eles pela primeira vez pelos sistemas de recomendação. O segundo componente é a quantia total de dinheiro que os compradores gastaram em compras de itens não recomendados, de categorias apresentadas pela primeira vez a eles (através de outros itens recomendados da mesma categoria) pelos sistemas de recomendação. Em ambos os casos, uma determinada compra contribui para a receita extra indireta apenas se o item comprado não foi recomendado ao comprador durante a sessão de compras vigente. Os pesquisadores se referem a

isso como a receita extra “indireta”, pois essa receita extra não foi gerada diretamente através da compra de itens recomendados, mas sim por causa da influência dos recomendadores. Conforme mostra a Figura 5, a receita extra indireta gerada pelos recomendadores aumentou substancialmente o valor extra total gerado.

Figura 5: A receita extra direta, indireta e total gerada pelos sistemas de recomendação.



Fonte: Dias (2008)

As linhas pontilhadas vermelhas verticais indicam o ponto em que cada sistema foi atualizado com novos dados.

Uma lição importante que aprendemos durante este estudo de caso é que o efeito de um sistema de recomendação se estende muito além da receita extra direta gerada pela compra de itens recomendados. A análise mostra como os recomendadores geraram uma quantidade substancial de receita adicional para o negócio, apresentando aos compradores novas categorias das quais eles continuaram a comprar.

Um outro exemplo de aplicação de *Data Mining* que é interessante citar, e que serviu como inspiração para a análise realizada neste trabalho, é sobre o estudo realizado por Rodrigues (2023), em que ele utiliza o algoritmo *K-means* para clusterizar as instituições de ensino superior públicas do país, com a observação de dados sobre despesas, quantidades de docentes e técnicos, localização e categoria administrativa das instituições de ensino superior.

Em seguida ele obteve quatro clusters diferentes e realizou uma análise que possibilitou a identificação de similaridades e dissimilaridades entre as instituições. Os clusters encontrados, assim como dados de concluintes de cursos de graduação nas instituições de ensino superior (como faixa etária, tempo de graduação e forma de ingresso), foram considerados em um segundo cenário do estudo, em que foi usado o algoritmo *Apriori* para geração de regras de associação que ajudaram basear a caracterização dos perfis socioeconômicos dos estudantes. Dessa forma as regras de associação geradas permitiram uma melhor percepção sobre os perfis socioeconômicos de discentes de instituições de ensino superior públicas brasileiras, pois foi possível observar os grupos aos quais as instituições estavam vinculadas, resultantes do agrupamento realizado no primeiro cenário. Essa análise em que foi feita a aplicação do algoritmo *Apriori* em uma base que já havia sido clusterizada previamente, permitiu descobrir associações que são mais relevantes e específicas para o contexto de cada grupo, o que também ajuda a trazer uma compreensão mais profunda dos padrões sobre os perfis socioeconômicos dos discentes.

Com os exemplos citados neste capítulo, foi possível perceber diferentes aplicações de ciência de dados relacionadas à *Data Mining* e regras de associações, evidenciando os resultados positivos obtidos e como eles agregam valor para as empresas.

3. DESENVOLVIMENTO

3.1 PROCESSO DE EXTRAÇÃO DOS DADOS

A base de dados utilizada neste trabalho foi cedida pela Olist, a maior loja de departamentos dos mercados brasileiros, e está disponível no site Kaggle.com, contendo informações de 100 mil pedidos de 2016 a 2018 feitos em vários *marketplaces* no Brasil. A Olist conecta pequenas empresas de todo o Brasil a canais sem complicações e com um único contrato. Esses comerciantes podem vender seus produtos através da Olist Store e enviá-los diretamente aos clientes usando os parceiros logísticos da Olist.

Este conjunto de dados está disponível para *download* no formato *csv* (*https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce*), e possui diversas tabelas e informações que serão especificadas nos próximos tópicos do trabalho, e que serão úteis para

realizar a análise exploratória dos dados e posteriormente a aplicação dos algoritmos de associação.

3.2 ANÁLISE EXPLORATÓRIA DOS DADOS

Inicialmente foi realizada uma análise exploratória dos dados presentes no banco de dados com informações de transações da empresa Olist. Essa análise permite uma maior compreensão dos dados, ajudando a entender a estrutura e como eles estão organizados, além de outras utilidades como corrigir irregularidades de dados ausentes e *outliers*, identificar padrões e relações entre variáveis, ajudando a entender o comportamento do cliente e sazonalidade de compras por exemplo, e realizar o pré-processamento dos dados ajudando a selecionar os atributos necessários para serem utilizados durante a aplicação dos algoritmos de associação posteriormente.

O presente trabalho foi realizado utilizando a plataforma *Jupyter Notebook*, uma popular aplicação de código aberto que permite criar e compartilhar documentos interativos que contenham código, texto formatado, visualizações de dados e outros elementos. Foi originalmente desenvolvida para suportar a linguagem de programação Python, mas agora oferece suporte a muitas outras linguagens. A linguagem de programação utilizada neste trabalho foi o Python.

Inicialmente, foram importadas as bibliotecas de Python necessárias para o trabalho, as mesmas estão apresentadas na figura 6.

Figura 6: Bibliotecas Python utilizadas no trabalho

```
# importando as bibliotecas e ferramentas necessárias

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
%matplotlib inline
import seaborn as sns
import datetime
import scipy.stats as stats
```

Fonte: Autor

Posteriormente, foi usada a função *read_csv()* da biblioteca *Pandas* para importar as bases de dados da empresa Olist, em um total de 5 bases de dados, com os nomes de *clientes*,

itens_pedidos, *pagamentos*, *pedidos* e *produtos*, que foram utilizadas para realizar a análise exploratória e também para aplicação dos algoritmos de associação no próximo tópico do trabalho.

Figura 7: Importação das bases de dados para as suas respectivas variáveis

```
# importando a base de dados da Olist

clientes = pd.read_csv("olist_customers_dataset.csv")
itens_pedidos = pd.read_csv("olist_order_items_dataset.csv")
pagamentos = pd.read_csv("olist_order_payments_dataset.csv")
pedidos = pd.read_csv("olist_orders_dataset.csv")
produtos = pd.read_csv("olist_products_dataset.csv")
```

Fonte: Autor

Os atributos de cada uma das bases podem ser observados nas imagens a seguir.

Figura 8: Atributos pertencentes a base de dados *clientes*

A base de dados clientes possui 99441 linhas e 5 colunas.

	customer_id	customer_unique_id	customer_zip_code_prefix	customer_city	customer_state
0	06b8999e2fba1a1fbc88172c00ba8bc7	861eff4711a542e4b93843c6dd7febb0	14409	franca	SP
1	18955e83d337fd6b2def6b18a428ac77	290c77bc529b7ac935b93aa66c333dc3	9790	sao bernardo do campo	SP
2	4e7b3e00288586ebd08712fdd0374a03	060e732b5b29e8181a18229c7b0b2b5e	1151	sao paulo	SP

Fonte: Autor

Figura 9: Atributos pertencentes a base de dados *Itens_pedidos*

A base de dados Itens_pedidos possui 111047 linhas e 7 colunas.

	order_id	order_item_id	product_id	seller_id	shipping_limit_date	price	freight_value
0	00010242fe8c5a6d1ba2dd792cb16214	1	4244733e06e7ecb4970a6e2683c13e61	48436dade18ac8b2bce089ec2a041202	2017-09-19 09:45:35	58.90	13.29
1	00018f77f2f0320c557190d7a144bdd3	1	e5f2d52b802189ee658865ca93d83a8f	dd7ddc04e1b6c2c614352b383efe2d36	2017-05-03 11:05:13	239.90	19.93
2	000229ec398224ef6ca0657da4fc703e	1	c777355d18b72b67abbeef9df44fd0fd	5b51032eddd242adc84c38acab88f23d	2018-01-18 14:48:30	199.00	17.87

Fonte: Autor

Figura 10: Atributos pertencentes a base de dados *pagamentos*

A base de dados Pagamentos possui 103886 linhas e 5 colunas.

	order_id	payment_sequential	payment_type	payment_installments	payment_value
0	b81ef226f3fe1789b1e8b2acac839d17	1	credit_card	8	99.33
1	a9810da82917af2d9aefd1278f1dcfa0	1	credit_card	1	24.39
2	25e8ea4e93396b6fa0d3dd708e76c1bd	1	credit_card	1	65.71

Fonte: Autor

Figura 11: Atributos pertencentes a base de dados *pedidos*

A base de dados Pedidos possui 99441 linhas e 8 colunas.

	order_id	customer_id	order_status	order_purchase_timestamp
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-10-02 10:56:33
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-07-24 20:41:37
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	2018-08-08 08:38:49

	order_approved_at	order_delivered_carrier_date	order_delivered_customer_date	order_estimated_delivery_date
0	2017-10-02 11:07:15	2017-10-04 19:55:00	2017-10-10 21:25:13	2017-10-18 00:00:00
1	2018-07-26 03:24:27	2018-07-26 14:31:00	2018-08-07 15:27:45	2018-08-13 00:00:00
2	2018-08-08 08:55:23	2018-08-08 13:50:00	2018-08-17 18:06:29	2018-09-04 00:00:00

Fonte: Autor

Figura 12: Atributos pertencentes a base de dados *produtos*

A base de dados Produtos possui 32951 linhas e 9 colunas.

	product_id	product_category_name	product_name_lenght	product_description_lenght	product_photos_qty
0	1e9e8ef04dbcff4541ed26657ea517e5	perfumaria	40.0	287.0	1.0
1	3aa071139cb16b67ca9e5dea641aaa2f	artes	44.0	276.0	1.0
2	96bd76ec8810374ed1b65e291975717f	esporte_lazer	46.0	250.0	1.0

	product_weight_g	product_length_cm	product_height_cm	product_width_cm
0	225.0	16.0	10.0	14.0
1	1000.0	30.0	18.0	20.0
2	154.0	18.0	9.0	15.0

Fonte: Autor

Foram realizadas análises para cada base de dados, começando pela base *clientes*. Foi feito um levantamento sobre quais estados e cidades do Brasil realizam pedidos com maior frequência, e também qual a porcentagem que as top 10 cidades com mais pedidos representa em relação ao total de pedidos, chegando nos seguintes resultados.

Figura 13: Principais estados realizadores de pedidos

```
Estão presentes 27 estados diferentes nessa base dos clientes. Os TOP 10 estados são:
customer_state
SP    41746
RJ    12852
MG    11635
RS     5466
PR     5045
SC     3637
BA     3380
DF     2140
ES     2033
GO     2020
```

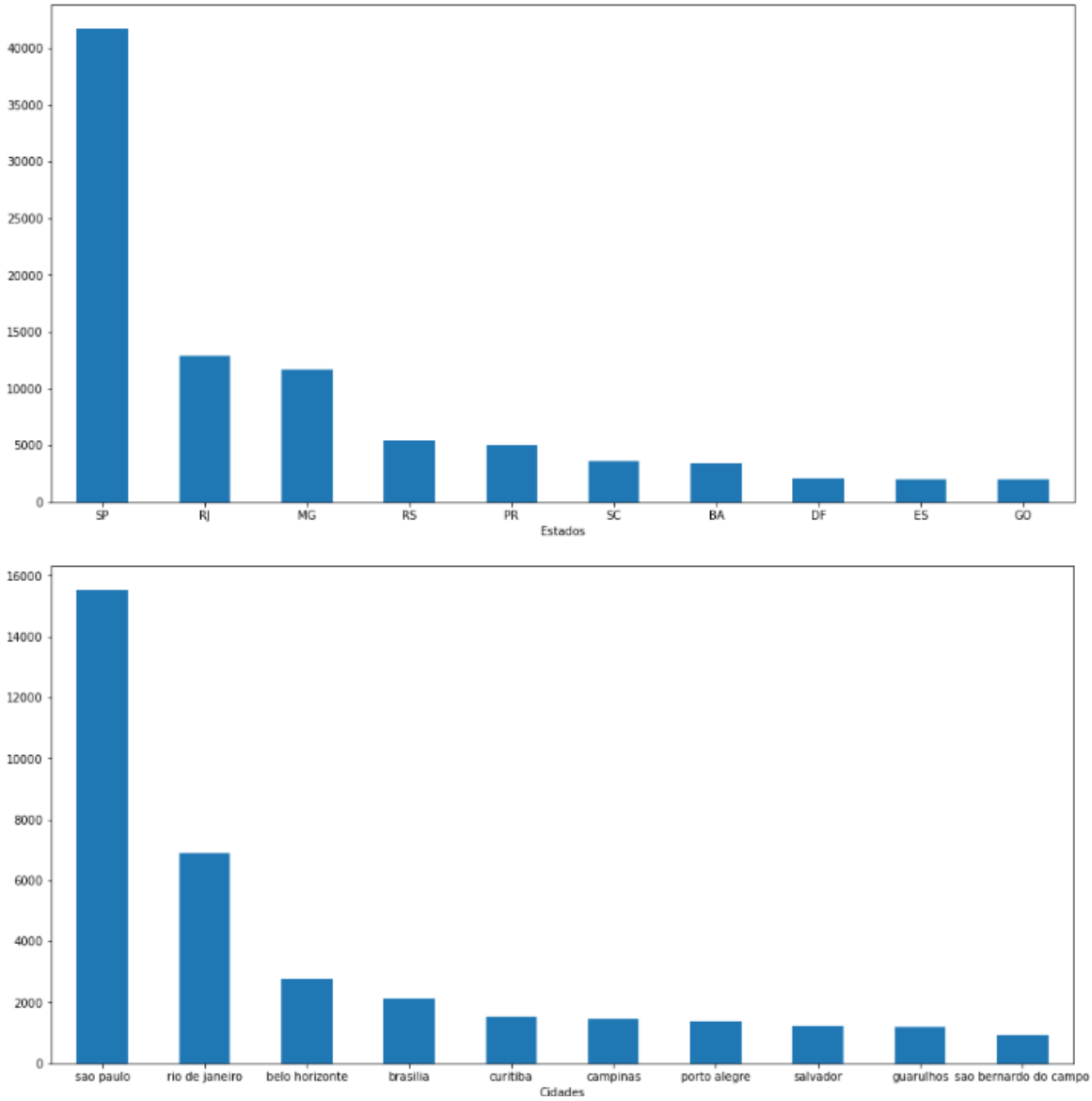
Fonte: Autor

Figura 14: Principais cidades realizadoras de pedidos

```
Estão presentes 4119 cidades diferentes nessa base dos clientes. As TOP 10 cidades são:
customer_city
sao paulo                15540
rio de janeiro            6882
belo horizonte            2773
brasilia                  2131
curitiba                  1521
campinas                  1444
porto alegre              1379
salvador                  1245
guarulhos                 1189
sao bernardo do campo      938
Name: customer_id, dtype: int64

As TOP 10 cidades cobrem 35.2 por cento de todos os pedidos.
```

Fonte: Autor

Figura 15: Graficos dos principais estados e cidades realizadores de pedidos



Fonte: Autor

Com os resultados foi possível perceber que uma grande parte dos pedidos é feita a partir da região Sudeste do país, sendo que 35,2% de todos os pedidos são realizados por apenas 10 cidades de um total de 4119 cidades. Posteriormente a análise foi feita sobre a base de dados pagamentos, trazendo informações em relação aos valores de pagamentos, e um histograma desses valores. Essas informações sobre os valores de pagamento são obtidas através da função *describe()* disponível na biblioteca *Pandas*, que permite gerar estatísticas descritivas de uma série de dados, oferecendo informações sobre suas características, como a média, o desvio padrão e a quantidade dos valores presentes na coluna analisada.

Figura 16: Estatísticas descritivas em relação aos valores de pagamento.

```
# realizando análise da base de dados pagamentos

pagamentos["payment_value"].describe()
```

```
count    103886.000000
mean        154.100380
std         217.494064
min           0.000000
25%          56.790000
50%         100.000000
75%         171.837500
max       13664.080000
```

Fonte: Autor

Figura 17: Código para criação do histograma de valores de pagamento

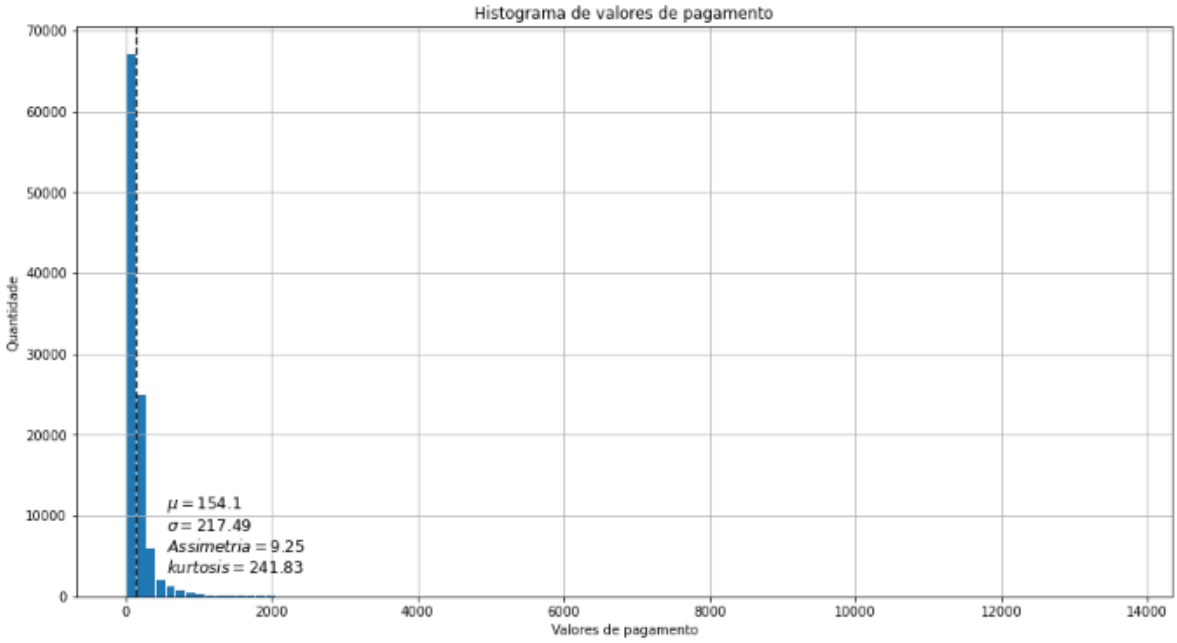
```
#criando um histograma dos valores de pagamento

media = pagamentos["payment_value"].mean()
std = pagamentos["payment_value"].std()
skew = pagamentos["payment_value"].skew()
kurt = pagamentos["payment_value"].kurtosis()

text1 = '$\mu=$' + str(round(media,2))
text2 = '$\sigma=$' +str(round(std,2))
text3 = '$Assimetria=$' + str(round(skew,2))
text4 = '$kurtosis=$' + str(round(kurt,2))
text = text1 + "\n" + text2 + "\n" + text3 + "\n" + text4

pagamentos.hist(column = "payment_value", bins = 100, figsize=(15,8), rwidth=0.9)
plt.axvline(media, color='k', linestyle='--')
plt.text(media+400, 2900, text, fontsize=12)
plt.xlabel("Valores de pagamento")
plt.ylabel("Quantidade")
plt.title("Histograma de valores de pagamento")
```

Fonte: Autor

Para a criação do histograma, foram utilizadas diferentes funções disponíveis nas bibliotecas Python instaladas. Inicialmente foram utilizadas as funções *mean(), std(), skew()* e *kurtosis(),* sendo todas pertencentes a biblioteca *Pandas*, para obter os valores da média,

desvio padrão, coeficiente de assimetria e curtose, que serão plotados no gráfico do histograma para facilitar a interpretação dos dados. Posteriormente são usadas variáveis para criar strings formatadas que exibem os valores da média, desvio padrão, assimetria e curtose, arredondados para duas casas decimais, e então são concatenadas em uma única *string* chamada *text* onde os valores são exibidos em uma nova linha. Para a criação do histograma foi utilizada a função *hist()* da biblioteca *matplotlib* , e foram passados os parâmetros como a coluna dos valores de pagamento que será utilizada, a quantidade de barras na distribuição, o tamanho do gráfico e a distância relativa entre as barras. Além disso outras funções foram utilizadas para alterar o nome dos eixos x, y e o título, e também incluir a *string text* criada anteriormente no gráfico.

Figura 18: Histograma dos valores de pagamento



Fonte: Autor

É possível perceber no histograma que existe um alto valor de desvio padrão, demonstrando que os valores de pagamento possuem uma grande dispersão em relação a media, além disso o valor do índice de assimetria demonstra que a distribuição é inclinada para a direita, indicando que pode haver a presença de valores extremamente altos em comparação com a media e por último o valor excessivo de curtose indica que a distribuição possui uma calda bem mais pesada em relação a distribuição normal, o que também indica presença de valores altos na base que estão afetando a análise da distribuição. Portanto foi

realizada a remoção desses valores mais altos da base de dados, para posteriormente criar um novo histograma.

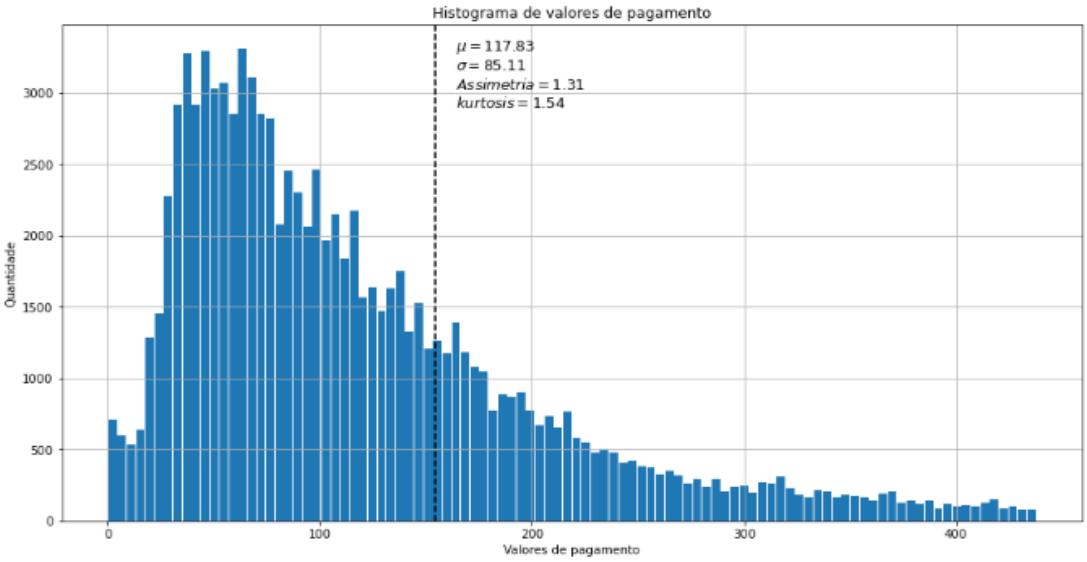
Figura 19: Código para criação do histograma de valores de pagamento eliminando os valores mais altos

```
# eliminando os 5% dos dados com valores mais altos para eliminar os outliers
q95 = pagamentos["payment_value"].quantile(.95)
pagamentos_q95 = pagamentos[pagamentos["payment_value"]<q95]

media2 = pagamentos_q95["payment_value"].mean()
std2 = pagamentos_q95["payment_value"].std()
skew2 = pagamentos_q95["payment_value"].skew()
kurt2 = pagamentos_q95["payment_value"].kurtosis()

text5 = '$\mu=$' + str(round(media2,2))
text6 = '$\sigma=$' +str(round(std2,2))
text7 = '$Assimetria=$' + str(round(skew2,2))
text8 = '$kurtosis=$' + str(round(kurt2,2))
text9 = text5 + "\n" + text6 + "\n" + text7 + "\n" + text8

pagamentos_q95.hist(column = "payment_value", bins = 100, figsize=(15,8), rwidth=0.9)
plt.axvline(media, color='k', linestyle='--')
plt.text(media+10, 2900, text9, fontsize=12)
plt.xlabel("Valores de pagamento")
plt.ylabel("Quantidade")
plt.title("Histograma de valores de pagamento")
```

Fonte: Autor

No código acima, foi utilizada a função *quantile()* da biblioteca *Pandas* para separar a parte da base de dados que continham somente os valores abaixo do percentil 95, eliminando assim os 5% dos valores mais altos presentes na base onde estão concentrados os poucos pagamentos de valor elevado, e que estão fora do padrão de compra, dessa forma foi possível realizar uma análise dos pagamentos que representam a maioria dos clientes. Posteriormente, foi utilizada a mesma lógica utilizada no gráfico anterior para criar o histograma corrigido.

Figura 20: Histograma dos valores de pagamento sem os valores mais altos



Fonte: Autor

Após eliminar os valores mais altos foi possível perceber que média diminuiu para um valor de 117, representando melhor o valor médio típico dos pagamentos. O desvio padrão também obteve uma diminuição substancial, indicando que agora os dados estão menos dispersos em relação à média. A assimetria ainda é positiva, porém passou de uma valor de 9,25 para 1,31, o que indica que a distribuição está mais próxima de ser uma distribuição simétrica e, por último, a grande redução na curtose para o valor de 1,54 demonstra que a distribuição está próxima de uma distribuição normal em termos de forma.

Com esses dados é possível verificar que mesmo eliminando os valores mais altos, os valores dos pedidos possuem uma grande variabilidade, já que o valor do desvio padrão ainda é considerado alto. Além disso, a cauda da direita da distribuição é mais longa do que a cauda da esquerda, o que é destacado pelo coeficiente de assimetria. É possível perceber que 75% dos pedidos possuem valor de até R$ 171,83, destacando que grande parte dos clientes da empresa realizam compras de baixo valor, sendo que o pedido de valor máximo foi de R$ 13664,00.

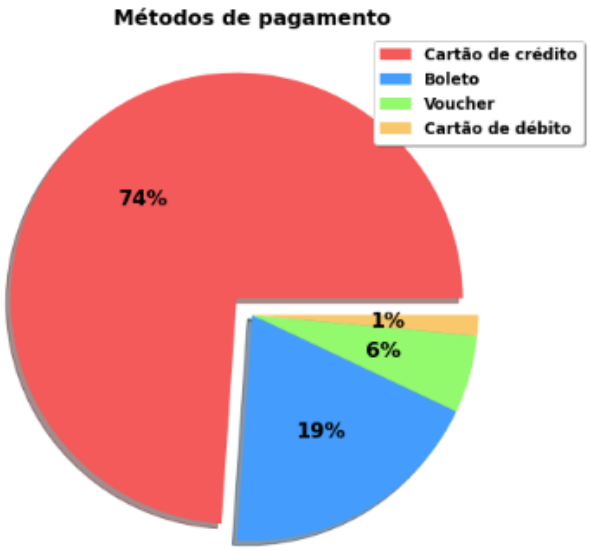
Outra análise feita em relação à base de dados *pagamento*, foi sobre os meios de pagamento utilizados pelos clientes e sobre o parcelamento. No código seguinte foi utilizado as funções *plot()* e *subplot()* pertencentes a biblioteca *matplotlib* para criar um gráfico de pizza que demonstra qual a porcentagem de utilização de cada método de compra. Para isso foi criada uma variável p que recebeu a contagem da quantidade de ocorrências de cada método de pagamento que fossem diferentes de *not_defined*, presentes na base de dados pagamentos na coluna *payment_type*. Posteriormente a função *plot()* foi utilizada em relação a p, para criar o gráfico de pizza, sendo que diversos parâmetros foram utilizados na função, como o tipo do gráfico, as cores, a legenda, título e o afastamento de uma das fatias do gráfico em relação às outras por exemplo.

Para a análise relativa aos parcelamentos, foi criada uma variável chamada *credito*, que recebe da base de dados pagamentos e coluna *payment_type* todas as operações em que foram utilizado o método de pagamento cartão de crédito, e posteriormente foi utilizada a função *hist()* sobre a variável *credito*, passando como parâmetro de avaliação a coluna de quantidade de parcelamentos, para criar um histograma sobre a quantidade de parcelamentos nos pedidos. Os resultados estão demonstrados nas imagens.

Figura 21: Código para criação do gráfico de pizza relativo aos métodos de pagamento e gráfico

```
#Criando um gráfico de pizza com os métodos de pagamento

fig, ax = plt.subplots(figsize=(9, 8), subplot_kw=dict(aspect="equal"))
explode = (0.1, 0, 0, 0)
colors = ['#f45a5a', '#449dfc', '#93f96d', '#f9c86d']
legend = ["Cartão de crédito", "Boleto", "Voucher", "Cartão de débito"]

p = pagamentos["payment_type"][pagamentos["payment_type"] != "not_defined"].value_counts()
p.plot(kind="pie", legend=False, labels=None, startangle=0, explode=explode, autopct='%1.0f%%', pctdistance=0.6,
       shadow=True, textprops={'weight':'bold', 'fontsize':16},
       colors=colors, ax=ax)
ax.legend(legend, loc='best', shadow=True, prop={'weight':'bold', 'size':12}, bbox_to_anchor=(0.6, 0, 0.5,1))
plt.title("Métodos de pagamento", fontweight='bold', size=16)
plt.ylabel("")
```

```
Text(0, 0.5, '')
```

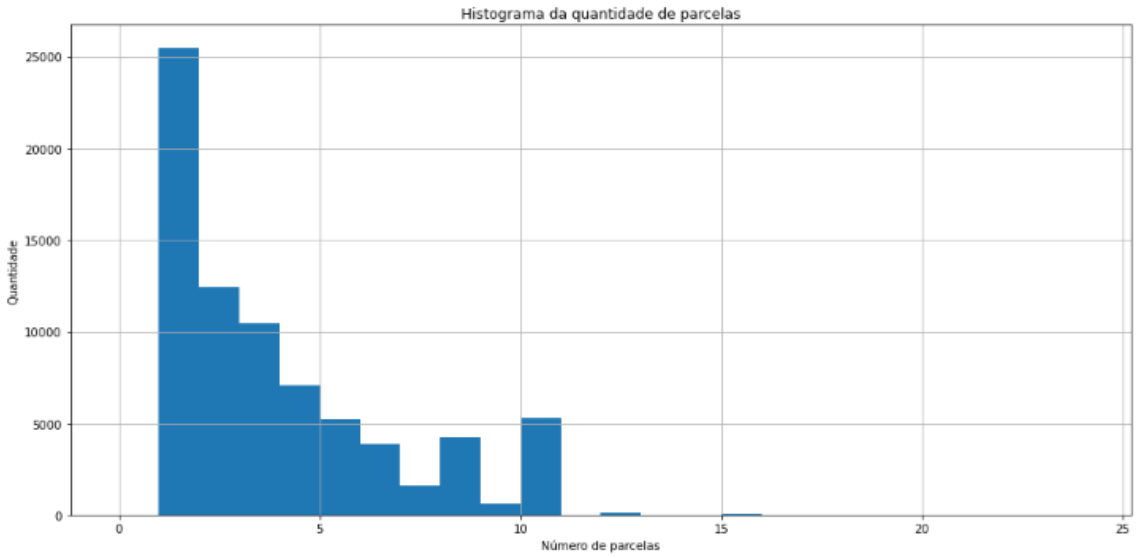


Fonte: Autor

Figura 22: Código para criação do histograma relativo às parcelas de pagamento e histograma.

```
#histograma da quantidade de parcelas

credito = pagamentos[pagamentos["payment_type"]=="credit_card"]
fig, ax = plt.subplots(figsize=(16,8))
credito.hist(column = "payment_installments", bins = credito["payment_installments"].max(), ax=ax)
plt.xlabel("Número de parcelas")
plt.ylabel("Quantidade")
plt.title("Histograma da quantidade de parcelas")
```

```
Text(0.5, 1.0, 'Histograma da quantidade de parcelas')
```



Fonte: Autor

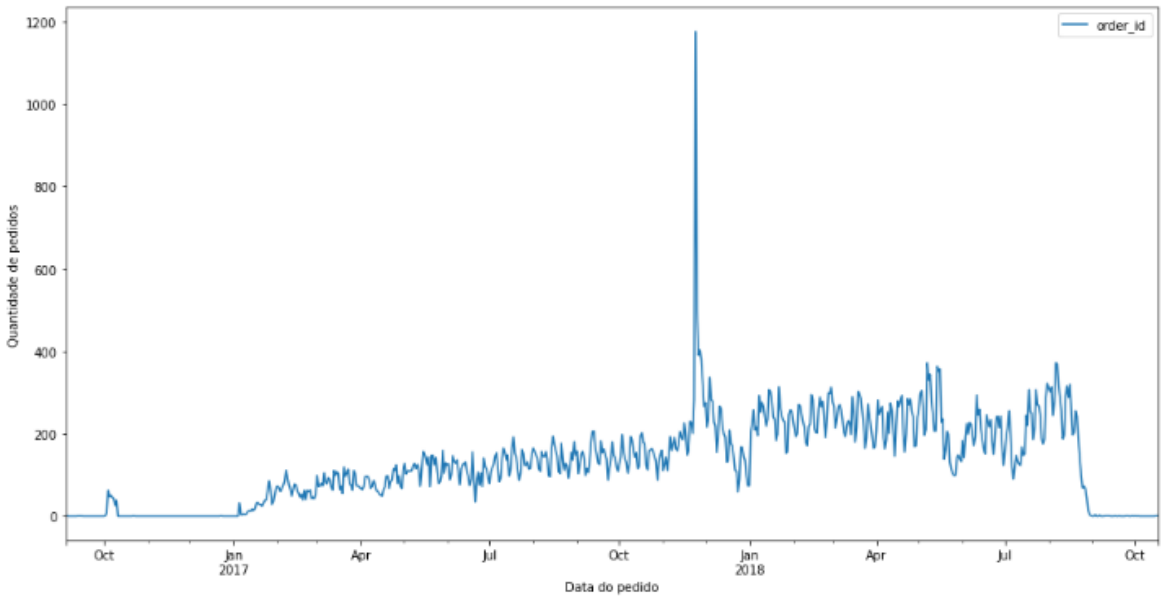
Com esses dados pode-se verificar que apesar de 75% das compras serem de baixo valor, 74% dos pagamentos são realizados por cartão de crédito, sendo que 50% dessas compras são feitas em apenas uma parcela, destacando a preferência dos clientes por esse meio de pagamento. Além disso, 75% das vendas por cartão de crédito são parceladas em até 4 vezes, o que é positivo para a empresa, já que receber em um menor prazo ajuda a melhorar o ciclo de caixa.

Posteriormente, foi realizada uma análise na base de dados *pedidos*, buscando evidenciar uma relação entre quantidade de pedidos e a data. Para isso foi utilizada a função *to_datetime* da biblioteca *Pandas* para modificar os horários da compra do tipo *string* para *datetime*, o que irá permitir realizar operações temporais. Posteriormente foi utilizada a função *groupby* da biblioteca *Pandas*, para agrupar os dados de venda em relação ao dia, para que pudesse ser analisado no gráfico que foi criado, quais épocas do ano possui maior quantidade de pedidos.

Figura 23: Código para criação do gráfico de linha relativo a quantidade de pedidos por data do pedido e gráfico

```
# alteração do formato da data e criação de gráfico de quantidade de pedidos pela data de compra

pedidos_2 = pedidos.copy()
pedidos_2["order_purchase_timestamp"] = pd.to_datetime(pedidos["order_purchase_timestamp"], format='%Y-%m-%d %H:%M:%S')

quantidade = pedidos_2.set_index("order_purchase_timestamp").groupby(pd.Grouper(freq='D')).count()
fig = plt.figure(figsize=(16,8))
ax = fig.gca()
quantidade.plot(y = "order_id", use_index=True, ax=ax)
plt.xlabel('Data do pedido')
plt.ylabel('Quantidade de pedidos')
```

```
Text(0, 0.5, 'Quantidade de pedidos')
```



Fonte: Autor

Com o resultado, é possível perceber um grande pico de vendas durante o período do Natal, seguido de queda das vendas e posteriormente a presença de pequenos picos no mês de Maio, associado ao Dias das Mães, e no mês de Julho.

Para a próxima análise, a coluna com o nome da categoria dos produtos foi adicionada a base de dados *itens_pedidos*, relacionando cada produto vendido com sua devida categoria, isso foi necessário para poder criar um gráfico relativo à quantidade de pedidos por categoria de produtos, e também será útil posteriormente quando for utilizado os algoritmos de associação no próximo tópico do trabalho, pois será necessário relacionar os pedidos com as categorias de produtos, já que na base *itens_pedidos* originalmente não possui o nome dos produtos para fazer essa relação. No código abaixo foi utilizada a função *merge()* pertencente a biblioteca *Pandas* para realizar a junção do atributo categorias de produto pertencente a base de dados *produtos* na base de dados *itens_pedidos*, utilizando o id do produto como atributo comum para realizar essa junção. Posteriormente foi utilizado o método *dropna()* pertencente

a biblioteca *Pandas* para poder remover as linhas da base de dados *itens_pedidos* em que estivessem presentes valores nulos na coluna de categoria dos produtos. Após essa etapa foi utilizada a função *nunique()* da biblioteca *Pandas* para trazer a informação sobre a quantidade de produtos e categorias de produtos diferentes na base de dados. Em seguida foi criada uma nova base de dados chamada transações que recebe da base dados *itens_pedidos* à coluna do id dos pedidos juntamente com a coluna contendo as categorias de produtos únicas em cada pedido, dessa forma será possível saber quantos pedidos possuem uma, duas ou mais categorias de produtos, servindo como base de dados para a aplicação dos algoritmos de associação posteriormente no trabalho. Por último foi criado um gráfico de barras utilizando a função *plot()* nesta nova base transações, para trazer a relação entre categorias de produto e quantidade vendida de cada uma delas.

Figura 24: Código para adicionar a coluna de categoria de produtos na base *itens_pedidos*

```
# Adicionando a coluna com o nome das categorias dos produtos da base produtos na base itens_pedidos

itens_pedidos = itens_pedidos.merge(produtos[['product_id','product_category_name']], on='product_id', how='left')
```

Fonte: Autor

Figura 25: Código para eliminar linhas com valores nulos e para informar a quantidade de produtos e categorias diferentes.

```
# ecluindo os dados de produtos que não possuem categoria

itens_pedidos.dropna(inplace=True, subset=['product_category_name'])

#verificando a quantidade de produtos e categorias diferentes

print("Número de produtos diferentes : ", itens_pedidos['product_id'].nunique())
print("Número de categorias de produtos diferentes : ", itens_pedidos['product_category_name'].nunique())
```

```
Número de produtos diferentes :  32341
Número de categorias de produtos diferentes :  73
```

Fonte: Autor

Figura 26: Código para criação da base transações

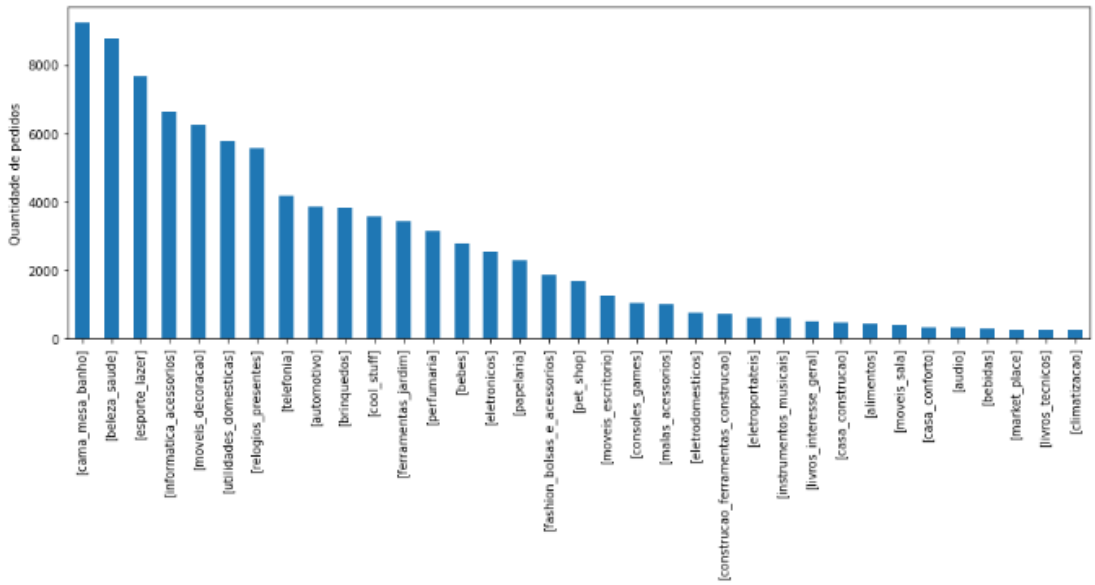
```
# criando uma nova base somente com o id dos pedidos e com a categoria dos produtos

transacoes = itens_pedidos.groupby("order_id").product_category_name.unique()
transacoes.head()
```

Fonte: Autor

Figura 27: Código para criação do gráfico de barras relativo a categorias de produtos e quantidade vendida e gráfico.

```
# criando um gráfico relacionando a quantidade de pedidos e as categorias de produtos

transacoes.value_counts()[:35].plot(kind='bar', figsize=(15,5))
plt.ylabel('Quantidade de pedidos')
```

```
Text(0, 0.5, 'Quantidade de pedidos')
```



Fonte: Autor

A partir do resultado, também foi possível verificar que 32341 produtos diferentes já foram vendidos, sendo que eles pertencem a 73 categorias de produtos diferentes. Também que grande parte dos pedidos estão relacionados às seguintes categorias de produtos: cama, mesa e banho; beleza e saúde; esporte e lazer; informática e acessórios; móveis de decoração; utilidades domésticas; relógios e presentes; telefonia; automotivos e brinquedos.

3.3 APLICAÇÃO DOS ALGORITMOS DE ASSOCIAÇÃO

Nesta etapa foram utilizados os algoritmos *Apriori* e *Fp-Growth* na base de dados *transacoes* criada anteriormente, para criar uma matriz com as categorias de produto e seus respectivos suportes. Em seguida foi utilizada a regra de associação para obter associações entre as categorias de produto, levando em consideração parâmetros como suporte, confiança e alavancagem. Os algoritmos *Apriori* e *Fp-Growth* foram aplicados novamente, porém desta vez em uma base de dados contendo o Id das transações e o Id de cada produto, que possui um volume de dados maior do que a base anterior formada pelo Id das transações e categorias

de produto, o que permitiu comparar o desempenho dos algoritmos em bases com tamanhos diferentes.

Para dar início a esta etapa, a base de dados criada anteriormente *transacoes* precisou ser transformada para o formato de lista, utilizando o método *tolist()* da biblioteca *NumPy*, para que os algoritmos de associação conseguissem calcular a frequência de coocorrência da categoria de produtos em diferentes transações. Também foi verificada a quantidade de transações presente na base.

Figura 28: Código para converter a base *transacoes* em lista

```
# Convertendo a base de dados transacoes para uma lista

transacoes = transacoes.tolist()

# quantidade de transações
len(transacoes)
```

```
97277
```

Fonte: Autor

Posteriormente uma etapa importante é a criação da matriz *one-hot*, que é uma técnica usada para representar dados categóricos, como as categorias de produtos, em uma forma numérica que é adequada para algoritmos de aprendizado de máquina. Nesta matriz, cada coluna possui uma categoria de produto e cada linha representa uma transação, caso a categoria esteja presente na transação é marcado como *True* e caso não esteja presente é marcado como *False*. A representação *one-hot* torna mais fácil identificar padrões de associação entre categorias de itens em transações.

Figura 29: Código para criação da matriz *one-hot*

```
from mlxtend.preprocessing import TransactionEncoder

encoder = TransactionEncoder()
encoder.fit(transacoes)
onehot = encoder.transform(transacoes)
onehot = pd.DataFrame(onehot, columns = encoder.columns_)
onehot.head()
```

	agro_industria_e_comercio	alimentos	alimentos_bebidas	artes	artes_e_artesanato	artigos_de_festas	artigos_de_natal	audio	automotivo	bebes	...
0	False	False	False	False	False	False	False	False	False	False	...
1	False	False	False	False	False	False	False	False	False	False	...
2	False	False	False	False	False	False	False	False	False	False	...
3	False	False	False	False	False	False	False	False	False	False	...
4	False	False	False	False	False	False	False	False	False	False	...

5 rows × 73 columns

Fonte: Autor

Para isso foi utilizada a classe *TransactionEncoder* da biblioteca *mlxtend.preprocessing*. Foi criado um objeto chamado *encoder* para receber a classe *TransactionEncoder* e realizar a codificação *one-hot* das transações. Posteriormente foi utilizado o método *fit()* da biblioteca *mlxtend* na base de dados, para analisar as transações e aprender quais categorias de produtos estão presentes, e criar um mapeamento entre categorias de produtos e colunas na matriz *one-hot*. Para criar a matriz *one-hot* foi utilizado o método *transform()* da biblioteca *mlxtend*, e em seguida ela foi convertida em um *DataFrame* do *Pandas*.

Com a matriz *one-hot* criada, foi possível realizar a aplicação dos algoritmos *Apriori* e *Fp-Growth*. Para a aplicação do algoritmo, foi utilizado a função *apriori* do módulo *frequent_patterns* da biblioteca *mlxtend*, junto de alguns parâmetros como a matriz *one-hot* e o nível mínimo de suporte, que nesse caso foi escolhido um valor baixo para que a matriz resultante também fornecesse as transações que possuem duas categorias de produto. O resultado da aplicação do algoritmo foi armazenado em uma nova base chamada *frequent_item_sets*.

Figura 30: Código para a aplicação do algoritmo *Apriori*

```
from mlxtend.frequent_patterns import apriori

frequent_item_sets = apriori(onehot, use_colnames = True, min_support = 0.00005, max_len = 2)
frequent_item_sets
```

	support	itemsets
0	0.001871	(agro_industria_e_comercio)
1	0.004626	(alimentos)
2	0.002334	(alimentos_bebidas)
3	0.002077	(artes)
4	0.000236	(artes_e_artesanato)
...	...	...
103	0.000062	(informatica_acessorios, telefonia)
104	0.000051	(papelaria, malas_acessorios)
105	0.000072	(moveis_sala, moveis_decoracao)
106	0.000072	(moveis_decoracao, relogios_presentes)
107	0.000247	(utilidades_domesticas, moveis_decoracao)

108 rows × 2 columns

Fonte: Autor

Com a matriz de conjuntos de categorias de produto e seus respectivos suportes pronta, é possível aplicar as regras de associação para obter como resultado possíveis padrões de compra, que podem ser pautados em parâmetros como o índice de suporte, confiança e alavancagem. Para isso foi utilizada a função *association_rules* também do módulo *frequent_patterns* da biblioteca *mlxtend*, junto de alguns parâmetros como a métrica utilizada para fazer as associações e o seu valor mínimo aceitável. Nesse caso foi escolhida a métrica confiança, pois ela mede a probabilidade de que a compra de um item B ocorra quando o item A é comprado, o que ajuda a indicar o quão confiável é a afirmação de que, se um cliente comprar o item A, ele provavelmente comprará o item B. Os resultados são demonstrados a seguir.

Figura 31: Código para a aplicação das regras de associação.

```
from mlxtend.frequent_patterns import association_rules

associacoes = association_rules(frequent_item_sets, metric = 'confidence', min_threshold = 0.01)
associacoes
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(artes)	(moveis_decoracao)	0.002077	0.066295	0.000051	0.024752	0.373367	-0.000086
1	(audio)	(relogios_presentes)	0.003598	0.057814	0.000062	0.017143	0.296516	-0.000146
2	(casa_conforto)	(cama_mesa_banho)	0.004081	0.096806	0.000442	0.108312	1.118859	0.000047
3	(moveis_decoracao)	(cama_mesa_banho)	0.066295	0.096806	0.000720	0.010854	0.112125	-0.005698
4	(casa_conforto)	(moveis_decoracao)	0.004081	0.066295	0.000062	0.015113	0.227970	-0.000209
5	(casa_construcao)	(ferramentas_jardim)	0.005037	0.036165	0.000072	0.014286	0.395017	-0.000110
6	(casa_construcao)	(moveis_decoracao)	0.005037	0.066295	0.000134	0.026531	0.400189	-0.000200
7	(construcao_ferramentas_iluminacao)	(moveis_decoracao)	0.002508	0.066295	0.000113	0.045082	0.680018	-0.000053
8	(moveis_sala)	(moveis_decoracao)	0.004338	0.066295	0.000072	0.016588	0.250209	-0.000216

Fonte: Autor

Após a obtenção desses resultados foi adicionado um filtro para que sejam apresentadas somente aquelas associações em que o valor da influência seja positivo, pois essa métrica compara a frequência real de compra conjunta dos itens com a frequência esperada se eles fossem independentes, portanto um valor positivo indica que a compra conjunta dos itens é maior do que o esperado ao acaso, sendo importante para identificar associações que são mais do que apenas uma coincidência aleatória. Após isso, apenas um resultado foi apresentado.

Figura 32: Código para aplicar o filtro em relação a influência.

```
associacoes = associacoes[associacoes['leverage'] > 0.00]
associacoes
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
2	(casa_conforto)	(cama_mesa_banho)	0.004081	0.096806	0.000442	0.108312	1.118859	0.000047

Fonte: Autor

Com esse resultado é possível afirmar que existe uma probabilidade de cerca de 10,8% de um cliente comprar um produto de cama mesa e banho caso ele tenha comprado um produto da categoria casa e conforto. Após essa etapa todos os passos utilizados anteriormente para a aplicação do algoritmo *Apriori* também foram realizados para a aplicação do algoritmo

FP-Growth, através da função *fpgrowth* também do módulo *frequent_patterns* da biblioteca *mlxtend*.

Figura 33: Código para a aplicação do algoritmo *FP-Growth*

```
from mlxtend.frequent_patterns import fpgrowth

frequent_item_sets2 = fpgrowth(onehot, use_colnames = True, min_support = 0.00005)

associacoes2 = association_rules(frequent_item_sets2, metric = 'confidence', min_threshold = 0.01)

associacoes2 = associacoes2[associacoes2['leverage'] > 0.00]
associacoes2
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
2	(casa_conforto)	(cama_mesa_banho)	0.004081	0.096806	0.000442	0.108312	1.118859	0.000047

Fonte: Autor

3.4 COMPARAÇÃO DOS TEMPOS DE EXECUÇÃO ENTRE ALGORITMOS DE ASSOCIAÇÃO

Os resultados obtidos através da aplicação dos dois algoritmos são os mesmos, porém um ponto percebido durante a execução do código foi que o algoritmo *Apriori* demorou mais tempo para fornecer o resultado, portanto foi feita uma análise em relação ao tempo de execução desses dois algoritmos, através do módulo *time* que é padrão da linguagem *Python*.

Figura 34: Código para calcular o tempo de execução dos algoritmos *Apriori* e *FP-Growth*

```
import time

start_time = 0
start_time = time.time()

frequent_item_sets2 = fpgrowth(onehot, use_colnames = True, min_support = 0.00005)
frequent_item_sets2

tempo_execucao = time.time() - start_time
print("Tempo de execução:",tempo_execucao)
```

```
Tempo de execução: 0.9010517597198486
```

```
import time

start_time2 = 0
start_time2 = time.time()

frequent_item_sets = apriori(onehot, use_colnames = True, min_support = 0.00005, max_len = 2)
frequent_item_sets

tempo_execucao2 = time.time() - start_time2
print("Tempo de execução:",tempo_execucao2)
```

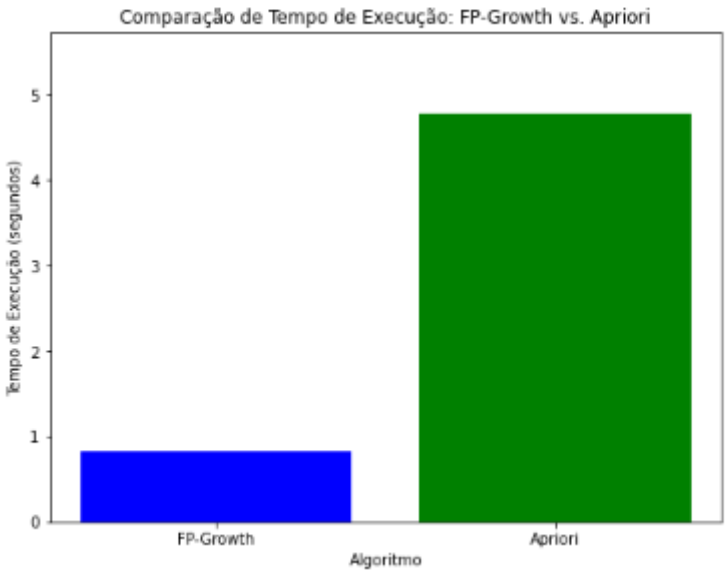
```
Tempo de execução: 4.886279344558716
```

Fonte: Autor

Para melhor visualização os resultados foram expressos em um gráfico.

Figura 35: Código para criação do gráfico de comparação do tempo de execução entre os algoritmos

```
algoritmos = ['FP-Growth', 'Apriori']

# Tempos de execução correspondentes
tempos = [tempo_execucao, tempo_execucao2]

# Criar o gráfico de barras
plt.figure(figsize=(8, 6))
plt.bar(algoritmos, tempos, color=['blue', 'green'])
plt.xlabel('Algoritmo')
plt.ylabel('Tempo de Execução (segundos)')
plt.title('Comparação de Tempo de Execução: FP-Growth vs. Apriori')
plt.ylim(0, max(tempos) * 1.2)  # Define o limite superior do eixo y
plt.show()
```



Fonte: Autor

Dessa forma é possível perceber que o algoritmo *Apriori* possui um tempo de execução quase cinco vezes maior que o *FP-Growth*. Nesse caso específico a diferença foi sobre poucos segundos, já que a base de dados *one-hot* utilizada pelos algoritmos é relativamente pequena contendo apenas 97277 linhas e 71 colunas. Isso levou a um questionamento se essa proporção entre os tempos de execução continuaria a mesma caso a base de dados fosse maior, portanto foi realizada uma nova análise dos tempos de execução, porém desta vez utilizando uma base de dados de transações, contendo os Id´s das transações e o Id dos produtos, não somente a categoria dos produtos.

Para criar esta base, foi necessário inicialmente filtrar os produtos para que a base contenha somente aqueles que tiveram uma frequência de venda igual ou superior a 10, já que produtos com frequência menor que essa não possuem influência relevante para obtenção dos resultados. O código calcula a contagem de ocorrências de produtos na base de dados *itens_pedidos* através da função *value_counts()* pertencente à biblioteca *Pandas*. Em seguida, ele filtra os registros para reter apenas os produtos que são considerados frequentes com base em um limite de 10 ou mais ocorrências.

Figura 36: Código para filtragem da base *itens_pedidos_frequentes*

```
frequencia_produtos = itens_pedidos['product_id'].value_counts()


itens_pedidos_frequentes = itens_pedidos[itens_pedidos.isin(frequencia_produtos.index[frequencia_produtos>=10]).values]

itens_pedidos_frequentes['product_id'].value_counts()
```

```
aca2eb7d00ea1a7b8ebd4e68314663af    527
99a4788cb24856965c36a24e339b6058    488
422879e10f46682990de24d770e7f83d    484
389d119b48cf3043d311335e499d9c6b    392
368c6c730842d78016ad823897a372db    388
                                   ...
0e1fa2aadc04afbf8fb30200aeba06a2     10
252641aa4855aef622089db60c4ad90a     10
317a06884480d645d7f0bc82455ec277     10
2b849e71ada03a262ef4338a7abd8b79     10
1d2d3b19f5c3d093d4dbccd3cd963a8f     10
```

Fonte: Autor

Após essa filtragem, foi criada a nova base formada pelo Id das transações e os respectivos Ids de produto para cada transação. Para isso foi utilizado o método *groupby()* e a função *unique()* da biblioteca *Pandas* para agrupar os registros com base na coluna "*order_id*", o que significa que os registros são agrupados por transação, de modo que todos os produtos únicos de uma mesma transação fiquem juntos.

Figura 37: Código para a criação da base *transacoes_produtos*

```
transacoes_produtos = itens_pedidos_frequentes.groupby("order_id").product_id.unique()
transacoes_produtos
```

```
order_id
00042b26cf59d7ce69dfabb4e55b4fd9    [ac6c3623068f30de03045865e4e10089]
00061f2a7bc09da83e415a52dc8a4af1    [d63c1011f49d98b976c352955b1c4bea]
0006ec9db01a64e59a68b2c340bf65a7    [99a4788cb24856965c36a24e339b6058]
0008288aa423d2a3f00fcb17cd7d8719    [368c6c730842d78016ad823897a372db]
000aed2e25dbad2f9ddb70584c5a2ded    [4fa33915031a8cde03dd0d3e8fb27f01]
                                                      ...
fff90cdcb3b2e6cfb397d05d562fd3fe    [764292b2b0f73f77a0272be03fdd45f3]
fffb9224b6fc7c43ebb0904318b10b5f    [43423cdffde7fda63d0414ed38c11a73]
fffc94f6ce00a00581880bf54a75a037    [4aa6014eceb682077f9dc4bffebc05b0]
fffce4705a9662cd70adb13d4a31832d    [72a30483855e2eafc67aee5dc2560482]
fffe18544ffabc95dfada21779c9644f    [9c422a519119dcad7575db5af1ba540e]
Name: product_id, Length: 43778, dtype: object
```

Fonte: Autor

Com a matriz *transacoes_produtos* disponível, foi possível converte-la para uma lista como foi feito na base anterior com as categorias de produtos, para poder criar uma nova matriz *one-hot*, que será utilizada para a nova aplicação do *Apriori* e do *FP-Growth*.

Figura 38: Código para criação da nova matriz *one-hot*

```
transacoes_produtos = transacoes_produtos.tolist()

encoder = TransactionEncoder()

encoder.fit(transacoes_produtos)

onehot1 = encoder.transform(transacoes_produtos)

onehot1 = pd.DataFrame(onehot1, columns = encoder.columns_)

onehot1.head()
```

	001b72dfd63e9833e8c02742adf472e3	00250175f79f584c14ab5cecd80553cd	005030ef108f58b46b78116f754d8d38
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False

Fonte: Autor

Desta vez a matriz *one-hot* resultante possui 97277 linhas e 1899 colunas, sendo bem maior em comparação a matriz *one-hot* encontrada anteriormente. Dessa forma os algoritmos *Apriori* e *FP-growth* foram aplicados novamente, e o tempo de execução foi comparado em um novo gráfico.

Figura 39: Código para calcular os novos tempos de execução dos algoritmos *Apriori* e *FP-Growth*

```
start_time3 = 0
start_time3 = time.time()

frequent_item_sets3 = apriori(onehot1, use_colnames = True, min_support = 0.001, max_len = 2)
frequent_item_sets3

tempo_execucao3 = time.time() - start_time3
print("Tempo de execução:",tempo_execucao3)
```

```
Tempo de execução: 198.6973648071289
```

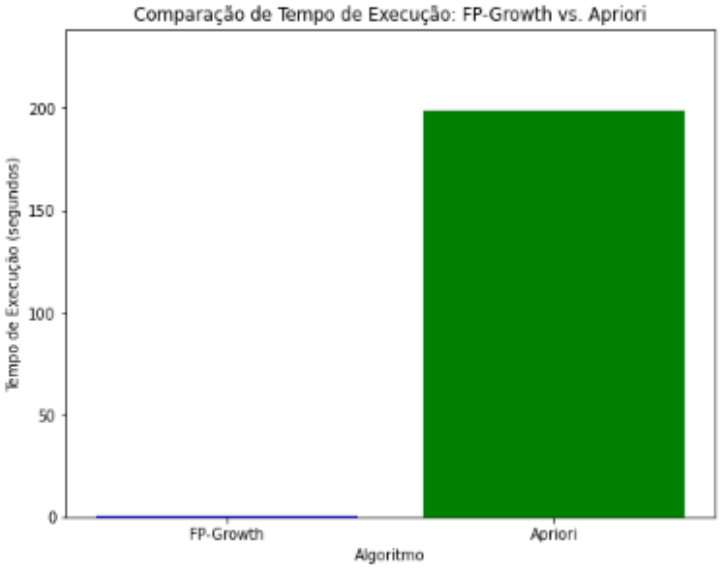
```
start_time4 = 0
start_time4 = time.time()

frequent_item_sets4 = fpgrowth(onehot1, use_colnames = True, min_support = 0.001)
frequent_item_sets4

tempo_execucao4 = time.time() - start_time4
print("Tempo de execução:",tempo_execucao4)
```

```
Tempo de execução: 1.2730729579925537
```

Fonte: Autor

Figura 40: Código para criação do gráfico de comparação dos novos tempos de execução entre os algoritmos

```
algoritmos2 = ['FP-Growth', 'Apriori']

# Tempos de execução correspondentes
tempos2 = [tempo_execucao4, tempo_execucao3]

# Criar o gráfico de barras
plt.figure(figsize=(8, 6))
plt.bar(algoritmos2, tempos2, color=['blue', 'green'])
plt.xlabel('Algoritmo')
plt.ylabel('Tempo de Execução (segundos)')
plt.title('Comparação de Tempo de Execução: FP-Growth vs. Apriori')
plt.ylim(0, max(tempos2) * 1.2)  # Defina o limite superior do eixo y
plt.show()
```



Fonte: Autor

Após a nova aplicação dos algoritmos em uma base maior, foi possível perceber através dos resultados, que o *Apriori* pode ser muito lento em relação ao *FP-Growth* quando se trata de grandes bases de dados. Neste último caso o *Apriori* demorou cerca de 200 segundos para ser executado, enquanto o *Fp-Growth* demorou apenas cerca de 1,3 segundos, um tempo muito próximo do alcançado anteriormente com a base menor. Isso é um indicativo que o processamento da estrutura de árvore do algoritmo *FP-Growth* pode ser muito mais eficiente do que o *Apriori* quando se trata de grandes bases de dados.

3.5 APLICAÇÃO DO ALGORITMO DE CLUSTERIZAÇÃO

Uma nova análise que pode trazer resultados interessantes, é aplicar o algoritmo de associação em diferentes clusters de clientes em relação a receita gerada por eles, para

observar se os padrões de compra mudam de acordo com os diferentes clusters, por exemplo, é possível que o padrão de compra de clientes que fazem compras de baixo valor seja diferente do padrão de clientes que realizam compras de alto valor.

Para isso é necessário criar uma base que contenha o Id de cada cliente e o respectivo valor já gasto por cada um no site, para que seja feita a clusterização sobre os valores de pagamento. Foi preciso adicionar o atributo de valores de pagamento presente na base de dados *pagamentos* para a base *clientes*, e para fazer isso foi necessário que essas bases tivessem uma chave em comum que servisse de parâmetro para utilizar na função *merge()* da biblioteca *Pandas* para realizar a junção da coluna. A base de dados *pedidos* possui os atributos *order_id* e *customer_id*, a base de dados *clientes* também possui o atributo *customer_id*, e a base *pagamentos* possui o atributo *order_id*, dessa forma foi possível fazer manipulações para adicionar o atributo de pagamentos na base de clientes, e depois formar a base *receita_cliente* composta pelos Id dos clientes e valores de pagamento, o que é destacado a seguir.

Figura 41: Código para criação da base receita_cliente

```
clientes = pd.merge(clientes, pedidos[['customer_id','order_id']], on = 'customer_id', how='left')
clientes = pd.merge(clientes, pagamentos[['order_id','payment_value']], on = 'order_id', how='left')
receita_cliente = clientes.groupby('customer_unique_id').payment_value.sum().reset_index()
```

Fonte: Autor

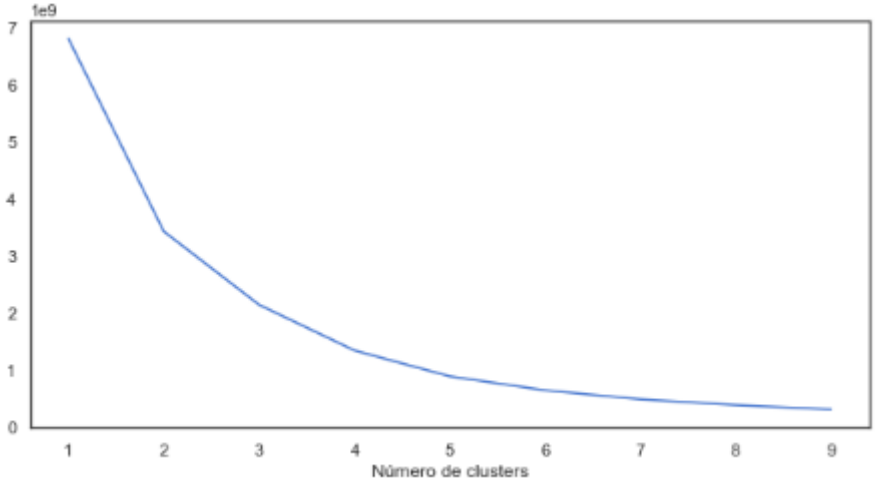
Para realizar a clusterização, primeiramente é interessante definir o número de clusters desejados para se passar como parâmetro quando utilizar o algoritmo *K-means*. Uma maneira de fazer isso é através da métrica da soma dos quadrados dos erros (SSE), pois ela mede a variabilidade dentro de cada agrupamento. É calculada a soma dos quadrados das distâncias entre cada ponto de dado e o centróide do cluster ao qual ele pertence. Quanto menor o valor do SSE, mais compactos são os clusters, o que indica que os pontos de dados dentro de cada cluster estão mais próximos uns dos outros, o que é desejável para uma segmentação eficaz.

Foi criado um gráfico da curva da soma dos quadrados dos erros em função do número de clusters, e o ponto onde o SSE começa a diminuir mais lentamente é frequentemente considerado um indicativo do número adequado de clusters. Para fazer isso, foi necessário utilizar o algoritmo *K-means* presente na biblioteca *sklearn*. No código inicialmente foi definida uma variável dicionário chamada "*sse*" para armazenar os valores obtidos do SSE, e posteriormente foi criada uma nova base chamada *receita_cliente2* para armazenar somente

os valores de pagamento. Foi criado um *loop* de 1 a 10 usando a variável "*k*" que é passada como parâmetro no algoritmo *K-means* para definir o número de clusters, dessa forma o algoritmo foi executado 10 vezes para agrupar os valores de pagamento, cada vez com um número de clusters, e o resultado da soma dos quadrados dos erros foi calculada pelo atributo *inertia_* e armazenado na variável "*sse*". Dessa forma, foi possível criar o gráfico, e ao analisá-lo perceber que o número ideal são 6 clusters.

Figura 42: Código para criação do gráfico da curva da soma dos quadrados dos erros em função do número de clusters

```
from sklearn.cluster import KMeans

sse={}
receita_cliente2 = receita_cliente[['payment_value']]
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, max_iter=1000).fit(receita_cliente2)
    receita_cliente2["clusters"] = kmeans.labels_
    sse[k] = kmeans.inertia_

plt.figure(figsize=(10, 5))
plt.plot(list(sse.keys()), list(sse.values()))
plt.xlabel("Número de clusters")
plt.show()
```



Fonte: Autor

Após definir o número de clusters, foi aplicado o algoritmo *K-means* sobre o atributo *payment_value* da base *receita_cliente*, e em seguida foi criada uma nova coluna nessa base chamada *RevenueCluster* que recebeu os resultados obtidos do método *predict()*, que é usado para prever a qual cluster cada cliente pertence com base em sua receita. Foi optado utilizar a clusterização, pois a base de dados utilizada não possui dados rotulados em cada instância com uma etiqueta de classe atribuída, portanto foi utilizado o algoritmo *K-means* para agrupar

esses dados em clusters com base em semelhanças intrínsecas, se baseando em medidas de similaridade entre os pontos de dados.

Figura 43: Código para a aplicação do algoritmo *K-means*

```
#aplicando o kmeans
kmeans = KMeans(n_clusters=6)
kmeans.fit(receita_cliente[['payment_value']])
receita_cliente['RevenueCluster'] = kmeans.predict(receita_cliente[['payment_value']])
receita_cliente.head()
```

	customer_unique_id	payment_value	RevenueCluster
0	0000366f3b9a7992bf8c76cfdf3221e2	141.90	0
1	0000b849f77a49e4a4ce2b2a4ca5be3f	27.19	0
2	0000f46a3911fa3c0805444483337064	86.22	0
3	0000f6ccb0745a6a4b88665a16c9f078	43.62	0
4	00050ab1314c0e55a6ca13cf7181fecf	35.38	0

Fonte: Autor

Dessa forma a base *receita_cliente* agora possui uma coluna com o cluster de cada cliente, porém ainda é necessário entender melhor os resultados obtidos da clusterização, portanto foi criada uma nova base de dados para fazer essa análise, chamada *receita_cliente3* que recebe os dados do atributo *payment_value* da base *receita_cliente*, que foram agrupados com base no atributo *RevenueCluster*, e em seguida foi utilizado o método *describe()* para gerar estatísticas descritivas para cada cluster separadamente.

Figura 44: Código para criação da base receita_cliente3

```
receita_cliente3= receita_cliente.groupby('RevenueCluster')['payment_value']
receita_cliente3.describe()
```

RevenueCluster	count	mean	std	min	25%	50%	75%	max
0	66082.0	82.309728	37.227030	0.00	51.1300	77.030	112.0675	159.22
1	23752.0	235.920741	64.587294	159.23	183.1800	216.640	276.4475	406.87
2	4750.0	577.056244	133.444171	406.92	460.0825	549.690	669.9750	891.34
3	1210.0	1204.065769	240.615149	891.86	1005.7500	1149.970	1366.0150	1798.01
4	294.0	2392.623061	622.163587	1803.24	1993.2375	2153.350	2624.3175	4809.44
5	8.0	8090.416250	2471.190122	6081.54	6873.3225	7102.095	8066.9775	13664.08

Fonte: Autor

Com esses resultados é possível perceber algumas características da base de clientes da empresa. Cerca de 69% dos clientes estão presentes no cluster 0, em que a média de receita gerada é cerca de apenas 82 reais, e o valor máximo gasto por um cliente desse grupo é de

159 reais. Os clientes presentes nos clusters 3,4 e 5 possuem uma média de compra acima dos 1000 reais, porém esses clientes representam somente cerca de 1,6% do total de clientes.

Figura 45: Código para obter a soma da receita total e a receita dos clusters 3,4 e 5.

```
soma_total = receita_cliente['payment_value'].sum()
soma_total

16008872.12

clusters_selecionados = [3, 4, 5]
soma_clusters_3_4_5 = receita_cliente[receita_cliente['RevenueCluster'].isin(clusters_selecionados)]['payment_value'].sum()
soma_clusters_3_4_5

2227747.38
```

Fonte: Autor

A receita gerada por todos os clientes é de cerca de 16 milhões de reais, e os clientes pertencentes aos clusters 3,4 e 5 que são apenas 1,6% do total, geram uma receita de cerca de 2,2 milhões de reais, que é próximo de 14% da receita total, o que demonstra que apesar de formarem um grupo pequeno de clientes eles possuem uma certa representatividade na receita, e provavelmente as estratégias de marketing e de vendas devem ser adequadas para esses diferentes grupos de clientes. Isso também mostra que as regras de associação geradas anteriormente podem sofrer mudanças de acordo com o cluster do cliente, o que leva a uma nova análise que será feita no tópico a seguir.

3.6 APLICAÇÃO DO ALGORITMO DE ASSOCIAÇÃO EM DIFERENTES CLUSTERS

Para realizar essa análise, foi necessário adicionar o atributo *RevenueCluster* pertencente a base de dados *receita_cliente*, na base *itens_pedidos*, que é utilizada como fonte para a criação da matriz de transações em que o algoritmo de associação será aplicado, pois nela estão os atributos *order_id* e *product_category_name*. Para isso foi necessário realizar algumas manipulações nas bases de dados, como adicionar o atributo *order_id* pertencente a base *pedidos* na base *receita_cliente*, para ser usado como parâmetro para adicionar o atributo *RevenueCluster* na base *itens_pedidos*, e novamente a função *merge()* foi utilizada.

Figura 46: Código para adicionar os clusters na tabela *itens_pedidos*.

```
receita_cliente = pd.merge(receita_cliente, pedidos[['customer_id','order_id']], on = 'customer_id', how='left')
itens_pedidos = pd.merge(itens_pedidos, receita_cliente[['order_id','RevenueCluster']], on = 'order_id', how='left')
```

Fonte: Autor

Com a base *itens_pedidos* possuindo o atributo *RevenueCluester*, foi possível criar novas bases de dados de transações contendo os dados de *order_id* e *product_category_name*

referentes a cada um dos clusters obtidos no tópico anterior. Para isso foi necessário filtrar a base *itens_pedidos* de acordo com o índice de cada cluster, e posteriormente foram criadas novas bases chamadas *base_cluster_0*, *base_cluster_1*, *base_cluster_2*, *base_cluster_3*, *base_cluster_4* e *base_cluster_5* através da função *groupby( )*.

Figura 47: Código para criação das bases de transações referente a cada cluster

```
# Criação das bases de transações para cada cluster
transacoes_cluster_0 = itens_pedidos[itens_pedidos['RevenueCluster'] == 0]
base_cluster_0 = transacoes_cluster_0.groupby("order_id").product_category_name.unique()
transacoes_cluster_1 = itens_pedidos[itens_pedidos['RevenueCluster'] == 1]
base_cluster_1 = transacoes_cluster_1.groupby("order_id").product_category_name.unique()
transacoes_cluster_2 = itens_pedidos[itens_pedidos['RevenueCluster'] == 2]
base_cluster_2 = transacoes_cluster_2.groupby("order_id").product_category_name.unique()
transacoes_cluster_3 = itens_pedidos[itens_pedidos['RevenueCluster'] == 3]
base_cluster_3 = transacoes_cluster_3.groupby("order_id").product_category_name.unique()
transacoes_cluster_4 = itens_pedidos[itens_pedidos['RevenueCluster'] == 4]
base_cluster_4 = transacoes_cluster_4.groupby("order_id").product_category_name.unique()
transacoes_cluster_5 = itens_pedidos[itens_pedidos['RevenueCluster'] == 5]
base_cluster_5 = transacoes_cluster_5.groupby("order_id").product_category_name.unique()
```

Fonte: Autor

Logo após foi necessário passar essas bases de dados criadas para o formato de lista com a função *tolist()*, para que fosse possível criar as novas matrizes *one-hot* utilizadas para a aplicação do algoritmo de associação. Em seguida foram criadas matrizes *one-hot* para cada base de cluster, da mesma maneira que foram criadas no tópico 3.2

Figura 48: Código para criação das matrizes *one-hot*

```
# Passando as bases criadas para lista para poder aplicar a one-hot

base_cluster_0 = base_cluster_0.tolist()
base_cluster_1 = base_cluster_1.tolist()
base_cluster_2 = base_cluster_2.tolist()
base_cluster_3 = base_cluster_3.tolist()
base_cluster_4 = base_cluster_4.tolist()
base_cluster_5 = base_cluster_5.tolist()

# criando matriz one_hot para base de transações do cluster 0
encoder = TransactionEncoder()

encoder.fit(base_cluster_0)

onehot_c0 = encoder.transform(base_cluster_0)

onehot_c0 = pd.DataFrame(onehot_c0, columns = encoder.columns_)

# criando matriz one_hot para base de transações do cluster 1
encoder = TransactionEncoder()

encoder.fit(base_cluster_1)

onehot_c1 = encoder.transform(base_cluster_1)

onehot_c1 = pd.DataFrame(onehot_c1, columns = encoder.columns_)

# criando matriz one_hot para base de transações do cluster 2
encoder = TransactionEncoder()

encoder.fit(base_cluster_2)

onehot_c2 = encoder.transform(base_cluster_2)

onehot_c2 = pd.DataFrame(onehot_c2, columns = encoder.columns_)

# criando matriz one_hot para base de transações do cluster 3
encoder = TransactionEncoder()

encoder.fit(base_cluster_3)

onehot_c3 = encoder.transform(base_cluster_3)

onehot_c3 = pd.DataFrame(onehot_c3, columns = encoder.columns_)

# criando matriz one_hot para base de transações do cluster 4
encoder = TransactionEncoder()

encoder.fit(base_cluster_4)

onehot_c4 = encoder.transform(base_cluster_4)

# criando matriz one_hot para base de transações do cluster 5
encoder = TransactionEncoder()

encoder.fit(base_cluster_5)

onehot_c5 = encoder.transform(base_cluster_5)

onehot_c5 = pd.DataFrame(onehot_c5, columns = encoder.columns_)
```

Fonte: Autor

Após essa etapa foi possível realizar a aplicação do algoritmo *Apriori* em cada base de transações *one_hot* para obter as matrizes de conjuntos de categorias de produto e seus respectivos suportes de cada cluster. Essas matrizes chamadas *frequent_item_sets_c0*,

frequent_item_sets_c1, *frequent_item_sets_c2*, *frequent_item_sets_c3*, *frequent_item_sets_c4* e *frequent_item_sets_c5* foram feitas para serem utilizadas pelo algoritmo de geração de associações.

Figura 49: Código para a aplicação do algoritmo *Apriori*

```
# aplicando APRIORI para a  base de transações cluster 0
frequent_item_sets_c0 = apriori(onehot_c0, use_colnames = True, min_support = 0.00005, max_len = 2)

# aplicando APRIORI para a  base de transações cluster 1
frequent_item_sets_c1 = apriori(onehot_c1, use_colnames = True, min_support = 0.00005, max_len = 2)

# aplicando APRIORI para a  base de transações cluster 2
frequent_item_sets_c2 = apriori(onehot_c2, use_colnames = True, min_support = 0.00005, max_len = 2)

# aplicando APRIORI para a  base de transações cluster 3
frequent_item_sets_c3 = apriori(onehot_c3, use_colnames = True, min_support = 0.00005, max_len = 2)

# aplicando APRIORI para a  base de transações cluster 4
frequent_item_sets_c4 = apriori(onehot_c4, use_colnames = True, min_support = 0.00005, max_len = 2)

# aplicando APRIORI para a  base de transações cluster 5
frequent_item_sets_c5 = apriori(onehot_c5, use_colnames = True, min_support = 0.00005, max_len = 2)
```

Fonte: Autor

Logo em seguida o algoritmo que gera as regras de associação foi aplicado em cada uma das bases criadas anteriormente, utilizando diferentes parâmetros e obtendo diferentes resultados de padrões de compra para os diferentes clusters. Para o cluster 0 foi utilizado o parâmetro confiança sendo superior a 0,003. Nesse cluster os valores da confiança foram baixos nas associações, e não obteve nenhuma associação com a influência positiva. Obtendo os seguintes resultados.

Figura 50: Código para aplicação da regra de associação na base referente ao cluster 0

```
# aplicando a regra de associação para obter os padrões de compra do cluster 0

associacoes_c0 = association_rules(frequent_item_sets_c0, metric = 'confidence', min_threshold = 0.003)
associacoes_c0
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(bebes)	(brinquedos)	0.029922	0.041650	0.000139	0.004642	0.111442	-0.001107
1	(brinquedos)	(bebes)	0.041650	0.029922	0.000139	0.003335	0.111442	-0.001107
2	(bebes)	(cama_mesa_banho)	0.029922	0.100074	0.000108	0.003610	0.036074	-0.002886

Fonte: Autor

Para o cluster 1 foi utilizado o parâmetro confiança sendo superior a 0,1 e o parâmetro influência superior a 0. Obtendo os seguintes resultados.

Figura 51: Código para aplicação da regra de associação na base referente ao cluster 1

```
# aplicando a regra de associação para obter os padrões de compra do cluster 1
associacoes_c1 = association_rules(frequent_item_sets_c1, metric = 'confidence', min_threshold = 0.1)
associacoes_c1 = associacoes_c1[associacoes_c1['leverage'] > 0.00]
associacoes_c1
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(artes)	(moveis_decoracao)	0.000648	0.074679	0.000173	0.266667	3.570833	0.000124
1	(casa_conforto)	(cama_mesa_banho)	0.007520	0.098924	0.001556	0.206897	2.091472	0.000812
2	(tablets_impressao_imagem)	(cool_stuff)	0.000346	0.050521	0.000086	0.250000	4.948460	0.000069

Fonte: Autor

Para o cluster 2 foi utilizado o parâmetro confiança sendo superior a 0,19 e o parâmetro influência superior a 0. Obtendo os seguintes resultados.

Figura 52: Código para aplicação da regra de associação na base referente ao cluster 2

```
# aplicando a regra de associação para obter os padrões de compra do cluster 2
associacoes_c2 = association_rules(frequent_item_sets_c2, metric = 'confidence', min_threshold = 0.19)
associacoes_c2 = associacoes_c2[associacoes_c2['leverage'] > 0.00]
associacoes_c2
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(alimentos_bebidas)	(esporte_lazer)	0.001064	0.060851	0.000213	0.200000	3.286713	0.000148
1	(bebidas)	(beleza_saude)	0.001489	0.103191	0.000426	0.285714	2.768778	0.000272
2	(casa_conforto)	(cama_mesa_banho)	0.004255	0.069787	0.001277	0.300000	4.298780	0.000980

Fonte: Autor

Para o cluster 3 foi utilizado o parâmetro confiança sendo superior a 0,1 e o parâmetro influência superior a 0. Além desses parâmetros também foi utilizado o suporte consequente sendo maior que 0,05, pois essa base não possui uma grande quantidade de transações como nas bases dos clusters anteriores, e esse parâmetro mede a porcentagem das vezes que a categoria do produto consequente esteve presente no total das transações. Dessa forma, para um melhor resultado foi especificado que a categoria de produto consequente deve estar presente em pelo menos 5% das transações dessa base. Obtendo os seguintes resultados.

Figura 53: Código para aplicação da regra de associação na base referente ao cluster 3

```
# aplicando a regra de associação para obter os padrões de compra do cluster 3
associacoes_c3 = association_rules(frequent_item_sets_c3, metric = 'confidence', min_threshold = 0.1)
associacoes_c3 = associacoes_c3[associacoes_c3['leverage'] > 0.00]
associacoes_c3 = associacoes_c3[associacoes_c3['consequent support'] > 0.05]
associacoes_c3
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(audio)	(relogios_presentes)	0.001701	0.116497	0.00085	0.500000	4.291971	0.000652
7	(construcao_ferramentas_iluminacao)	(esporte_lazer)	0.005952	0.068027	0.00085	0.142857	2.100000	0.000445
9	(eletronicos)	(informatica_acessorios)	0.004252	0.080782	0.00085	0.200000	2.475789	0.000507

Fonte: Autor

Para o cluster 4 foi utilizado o parâmetro confiança sendo superior a 0,1 e o parâmetro influência superior a 0. Obtendo os seguintes resultados.

Figura 54: Código para aplicação da regra de associação na base referente ao cluster 4

```
# aplicando a regra de associação para obter os padrões de compra do cluster 4
associacoes_c4 = association_rules(frequent_item_sets_c4, metric = 'confidence', min_threshold = 0.1)
associacoes_c4 = associacoes_c4[associacoes_c4['leverage'] > 0.00]
associacoes_c4
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(utilidades_domesticas)	(cama_mesa_banho)	0.021978	0.021978	0.003663	0.166667	7.583333	0.00318
1	(cama_mesa_banho)	(utilidades_domesticas)	0.021978	0.021978	0.003663	0.166667	7.583333	0.00318

Fonte: Autor

Para o cluster 5 não foi possível gerar regras de associação, mesmo reduzindo o valor do parâmetro de confiança para 0,00001. Isso se deve a baixa quantidade de transações presentes nessa base de dados.

Com os resultados obtidos, foi possível perceber que a utilização da clusterização de forma prévia para a aplicação de algoritmos de associação, pode ser muito útil para aprimorar os resultados, assim como foi observado por Rodrigues (2023) no exemplo citado no capítulo 2.3, em que foi possível descobrir associações que são mais relevantes e específicas para o contexto de cada grupo obtido através da clusterização. Essa aplicação permite personalizar a análise de associação para diferentes segmentos dos dados. Cada cluster possui características distintas, e analisar as regras de associação em cada grupo específico pode levar a insights mais relevantes e a estratégias mais personalizadas, que podem ser usadas para direcionar esforços de marketing, otimizar a gestão de estoque e realizar ofertas aos clientes.

4. CONCLUSÃO

Foi possível obter diferentes conclusões ao longo deste trabalho. Inicialmente foi realizada uma análise exploratória dos dados, que permitiu entender melhor a natureza dos dados e extrair algumas informações relevantes. Por exemplo, 35,2% de todos os pedidos são realizados por apenas 10 cidades de um total de 4119 cidades, e 74% dos pagamentos são realizados com o cartão de crédito. Posteriormente, foram aplicados os algoritmos *Apriori* e *FP-Growth* em uma base de transações. Com esses métodos, foi possível identificar conjuntos de itens frequentes para serem utilizados na criação de regras de associação, que indicaram a coocorrência de categorias de produtos nas transações. Essas regras podem ser úteis para entender padrões de compra e o comportamento do consumidor.

Para avaliar o desempenho dos algoritmos, foi feito um estudo comparativo de tempo de execução entre o *Apriori* e o *FP-Growth*. Foi possível observar que o *FP-Growth* demonstrou uma eficiência superior em relação ao *Apriori*, especialmente quando aplicado a grandes volumes de dados. Essa descoberta sugere que a escolha do algoritmo pode ser crucial ao lidar com bases de dados extensas. Como extensão da análise de regras de associação, foi realizada uma clusterização dos valores de pagamentos. Isso permitiu agrupar transações semelhantes com base em seus montantes. A clusterização ofereceu uma visão mais granular dos dados, destacando segmentos distintos de clientes.

Após a clusterização, foram aplicados novamente os algoritmos de mineração de regras de associação a cada cluster. Essa abordagem permitiu identificar associações e padrões de compra específicos para cada grupo de transações. Foi possível perceber que as regras de associação variavam significativamente entre os clusters, destacando a importância da segmentação para a análise de mercado.

As conclusões deste trabalho têm informações que podem ser consideradas relevantes para estratégias de negócios e tomada de decisões. A análise de regras de associação pode ser aplicada para melhorar a personalização de ofertas, otimizar o gerenciamento de estoque e aprimorar a experiência do cliente. Além disso, o uso da clusterização pode permitir uma abordagem mais direcionada a diferentes segmentos de clientes.

5. REFERÊNCIAS

AGRAWAL, R.; IMIELIŃSKI, T.; SWAMI, A.. **Mining association rules between sets of items in large databases**. ACM SIGMOD Record, 22(2), 207–216, 1993.

AGGARWAL, C. C.; YU, P. S. **A new approach to online generation of association rules**. IEEE Transactions on Knowledge and Data Engineering, 13(4), 527–540, 2001.

AMARAL, Fernando. **Introdução à ciência de dados: mineração de dados e big data**. 1. ed. rev. [S. l.]: Starlin Altas book, 2016.

AALST, W. V. D**. Data science in action.** In: Process Mining. [S.l.: s.n.]. p. 3–23, 2016.

ALZUBI, J.; NAYYAR, A.; KUMAR, A. **Machine Learning from Theory to Algorithms: An Overview**. Journal of Physics: Conference Series, 1142, 012012, 2018.

AWS. **O que são arquivos de log?**. 2023. Disponível em: < https://aws.amazon.com/pt/what-is/log files/#:~:text=Um%20log%20de%20altera%C3%A7%C3%B5es%20%C3%A9,na%20configura%C3%A7%C3%A3o%20de%20um%20sistema.>. Acesso em: 28 Jul. 2023.

BERRY, M. W.; MOHAMED, A.; YAP, B. W**. Supervised and Unsupervised Learning for Data Science.** Unsupervised and Semi-Supervised Learning, 2020.

BORGELT, C. **An implementation of the FP-growth algorithm**. Proceedings of the 1st International Workshop on Open Source Data Mining Frequent Pattern Mining Implementations - OSDM ’05, 2005.

CAO, L.. **Data Science**. ACM Computing Surveys, 50(3), 1–42, 2017.

CARDOSO, Pedro Henrique. **Ciência de dados aplicada a dados governamentais abertos sob a ótica da Ciência da Informação**. Dissertação de Mestrado, [s. l.], 2019.

DB-Engines. **DB-Engines Ranking**. 2023. Disponível em:
< https://db-engines.com/en/ranking.>. Acesso em: 28 Jul. 2023.

DEMŠAR, Janez et al**. Orange: data mining toolbox in Python**. the Journal of machine Learning research, v. 14, n. 1, p. 2349-2353, 2013.

DENG, L.; YU, D**. Deep Learning: Methods and Applications**. Foundations and Trends in Signal Processing. 7 (3–4): 197–387, 2014.
DE MELO CRUZ, Wander Luis. **Crescimento do e-commerce no Brasil: desenvolvimento, serviços logísticos e o impulso da pandemia de Covid-19**. GeoTextos, 2021.

DIAS, M. B.; LOCHER, D.; LI, M.; EL-DEREDY, W.; LISBOA, P. J. G. **The value of personalised recommender systems to e-business**. Proceedings of the 2008 ACM Conference on Recommender Systems - RecSys '08, (2008).

EL NAQA, I.; MURPHY, M. J. **What Is Machine Learning?** Machine Learning in Radiation Oncology, 3–11, 2015.

FREITAS NUNES, Diogo Henriques. **Um breve estudo sobre o algoritmo K-means**. 2016. 60 f. Dissertação (Mestrado em Matemática) - Departamento de Matemática - Faculdade de Ciências e Tecnologia de Coimbra. Coimbra, 2016.

HAN, J.; PEI, J.; YIN, Y.; MAO, R**. Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach**. Data Mining and Knowledge Discovery, 8(1), 53–87, 2004.

HOFMANN, T. **Unsupervised learning by probabilistic latent semantic analysis.** Machine Learning, 42, 177–196, 2001.

IME. **Aula 24 - Introdução à Python (Parte 3)**.2020. Disponível em: < https://www.ime.usp.br/~kellyrb/mac0216_2020/mac0216_aula24.html>. Acesso em: 28 Jul. 2023.

JACKSON, Joyce. **Data mining; a conceptual overview**. Communications of the Association for Information Systems, v. 8, n. 1, p. 19, 2002.

KOTSIANTIS, S. B. **Supervised machine learning: A review of classification techniques**. Informatica, 31, 249–268, 2007.

LAUDON, K.C.; TRAVER, C.G**. E-commerce business.technology.society** (4th Ed), Pearson, India, 2008.

MCCREA, N. **An Introduction to Machine Learning Theory and Its Applications: A Visual Tutorial with Examples**. Retrieved September 2, 2015.

MAHESH, Batta. **Machine learning algorithms-a review.** International Journal of Science and Research (IJSR).[Internet] 9: 381-386, 2020.

MARSON, M. D. **A industrialização brasileira antes de 1930**: uma contribuição sobre a evolução da indústria de máquinas e equipamentos no estado de São Paulo, 1900-1920, 2014.

MING-SYAN, C.; JIAWEI, H.; YU, P. S**. Data mining: an overview from a database perspective.** IEEE Transactions on Knowledge and Data Engineering, 8(6), 866–883, 1996.

MÜLLER, O.; FAY, M.; BROCKE, J.V. **The effect of big data and analytics on firm performance: an econometric analysis considering industry characteristics**, Journal of Management Information Systems, Vol. 35 No. 2, pp. 488-509, 2018.

ORACLE. **What is a relational database**. 2023 . Disponível em:

< https://www.oracle.com/br/database/what-is-a-relational-database/>. Acesso em: 28 Jul. 2023.

OSMAN, Abdullahi Sidow. **Data mining techniques**, 2019.

PANJAITAN, S.; SULINDAWATY, A. M.; LINDAWATI, S.; WATRIANTHOS, R.; SIHOTANG, H. T.; SINAGA, B. **Implementation of Apriori Algorithm for Analysis of Consumer Purchase Patterns**. Journal of Physics: Conference Series, 2019.

PEDREGOSA, Fabian. **Scikit-learn: Machine Learn-ing in Python**, Journal of Machine Learning Re-search, 12, 2011.

PORCU, V. **Python for Data Mining Quick Syntax Reference**, 2018.

PRADEEP, I. K.; BHASKAR, M. J.; SATYANARAYANA, B. **Data Science and Deep Learning Applications in the e-Commerce Industry: A Survey**. Indian Journal of Computer Science and Engineering, v. 11, n. 5, 2020.

RACHID, A. D. **Clustering prediction techniques in defining and predicting customers defection: The case of e-commerce context**. International Journal of Electrical and Computer Engineering 8.4, 2367, 2018.

RODRIGUES, E. M.; GOUVEIA, R. M. M.; ALBUQUERQUE, G. A.; BATISTA, M. C. M. **Aprendizado de máquina para agrupamento e associação de dados do ensino superior público brasileiro**. Revista de Sistemas e Computação, Salvador, v. 13, n. 1, p. 20-33, jan./abr, 2023.

ROSENBLATT, F. **The perceptron: A probabilistic model for information storage and organization in the brain**. Psychological Review, 65(6), 386–408, 1958.

UCR. **Engineering data analysis: why it matters**. 2023. Disponível em:

< https://engineeringonline.ucr.edu/blog/engineering-data-analysis/#:~:text=Data%20analysis%20involves%20gathering%20and,efficiency%20of%20an%20engineering%20project.>. Acesso em: 28 Jul. 2023.

URNE, A. C. **Impact of E-commerce on Consumer Buying Behaviour: A Review of Existing Literature**. SaiBalaji International Journal of Management Science-A journal of Contemporary Research, 2020.

SAABITH, S.; VINOTHRAJ, T.; FAREEZ, M. **A review on Python libraries and Ides for Data Science.** Int. J. Res. Eng. Sci**.**, v. 9, n. 11, p. 36-53, 2021.

SAGIROGLU, S.; SINANC, D. (2013). **Big data: A review.** 2013 International Conference on Collaboration Technologies and Systems (CTS), 2013.

SCIELO. **A industrialização brasileira antes de 1930**, São Paulo, out/dez. 2014. Disponível em:<http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0101-41612015000400753>. Acesso em: 5 Jun. 2023.

STANCIN, I.; JOVIC, A**. An overview and comparison of free Python libraries for data mining and big data analysis**. 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2019.

TENG, X.; GONG, Y. **Research on Application of Machine Learning in Data Mining**. IOP Conference Series: Materials Science and Engineering, 392(6), 062202, 2018.

TROILO, M.; BOUCHET, A.; URBAN, T.L.; SUTTON, W.A. **Perception, reality, and the adoption of business analytics: evidence from North American professional sport organizations**, Omega, Vol. 59, pp. 72-83, 2016.

VAIDYA, N.; KHACHANE, A. R**. Recommender systems-the need of the ecommerce ERA**. 2017 International Conference on Computing Methodologies and Communication (ICCMC), 2017.

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
FACULDADE DE ENGENHARIA

Termo de Declaração de Autenticidade de Autoria

Declaro, sob as penas da lei e para os devidos fins, junto à Universidade Federal de Juiz de Fora, que meu Trabalho de Conclusão de Curso do Curso de Graduação em Engenharia de Produção é original, de minha única e exclusiva autoria. E não se trata de cópia integral ou parcial de textos e trabalhos de autoria de outrem, seja em formato de papel, eletrônico, digital, áudio-visual ou qualquer outro meio.
Declaro ainda ter total conhecimento e compreensão do que é considerado plágio, não apenas a cópia integral do trabalho, mas também de parte dele, inclusive de artigos e/ou parágrafos, sem citação do autor ou de sua fonte.
Declaro, por fim, ter total conhecimento e compreensão das punições decorrentes da prática de plágio, através das sanções civis previstas na lei do direito autoral[1] e criminais previstas no Código Penal[2], além das cominações administrativas e acadêmicas que poderão resultar em reprovação no Trabalho de Conclusão de Curso.

Juiz de Fora, 14 de Dezembro de 2023.

PEDRO REIS CABREIRA
NOME LEGÍVEL DO ALUNO (A)

201649057
Matrícula

Pedro Reis Cabreira
ASSINATURA

020 339 266 30
CPF

[1] LEI N° 9.610, DE 19 DE FEVEREIRO DE 1998. Altera, atualiza e consolida a legislação sobre direitos autorais e dá outras providências.
[2] Art. 184. Violar direitos de autor e os que lhe são conexos: Pena - detenção, de 3 (três) meses a 1 (um) ano, ou multa.