Universidade Federal de Ouro Preto - UFOP - Escola de Minas - Colegiado do curso de Engenharia de Controle e Automação - CECAU

Gabriel Bueno Guimarães

Uma análise exploratória da influência da detecção de *outliers* na precificação de produtos em *e-commerce*

Monografia de Graduação em Engenharia de Controle e Automação

Ouro Preto, 2021

Gabriel Bueno Guimarães

Uma análise exploratória da influência da detecção de *outliers* na precificação de produtos em *e-commerce*

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como parte dos requisitos para a obtenção do Grau de Engenheiro de Controle e Automação.

Orientador: Jadson Castro Gertrudes

Coorientadora: Adrielle de Carvalho Santana

Ouro Preto, 2021

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

G963a Guimaraes, Gabriel Bueno.
Uma análise exploratória da influência da detecção de outliers na precificação de produtos em e-commerce. [manuscrito] / Gabriel Bueno Guimaraes. - 2022.
56 f.: il.: color., gráf..

Orientador: Prof. Dr. Jadson Castro Gertrudes.
Coorientadora: Profa. Dra. Adrielle de Carvalho Santana.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Escola de Minas. Graduação em Engenharia de Controle e Automação .

1. Inteligência artificial. 2. Aprendizado de máquina. 3. Algoritmos computacionais - Floresta aleatória. 4. Comércio eletrônico. I. Gertrudes, Jadson Castro. II. Santana, Adrielle de Carvalho. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 681.5

Bibliotecário(a) Responsável: Maristela Sanches Lima Mesquita - CRB-1716

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
ESCOLA DE MINAS
DEPARTAMENTO DE ENGENHARIA CONTROLE E AUTOMACAO

FOLHA DE APROVAÇÃO

Gabriel Bueno Guimarães

Uma análise exploratória da influência da detecção de *outliers* na precificação de produtos em ***e-commerce***

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Engenheiro de Controle e Automação

Aprovada em 05 de janeiro de 2022

Membros da banca

Prof. Dr. Jadson Castro Gertrudes - Orientador
Profa. Dra. Adrielle de Carvalho Santana – Coorientadora
Prof. Me. Pedro Henrique Lopes Silva - Professor Convidado
Prof. Dr. Agnaldo José da Rocha Reis - Professor Convidado

Jadson Castro Gertrudes, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 11/01/2022

Documento assinado eletronicamente por **Jadson Castro Gertrudes**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 11/01/2022, às 10:39, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0264529** e o código CRC **F91851AB**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: 3135591533 - www.ufop.br

Agradecimentos

Agradeço à minha mãe, por todo o apoio e amor incondicional. Ao meu pai, por me incentivar a estudar e alcançar meus objetivos. Aos meus amigos e a minha namorada, Maryane, que sempre estiveram comigo e me ajudaram durante toda esta grande jornada.

Aos meus orientadores, Jadson e Adrielle, que sempre me incentivavam durante todo o trabalho e estavam sempre dispostos a me ajudar. Sou grato por ter sido aluno de vocês e por ter tido a honra de ter sido orientado neste trabalho de conclusão de curso.

A grandiosa república Badalação, por todos os momentos bons, conselhos e amizade que criei durante este período de graduação.

Por fim, a Universidade Federal de Ouro Preto, que me proporcionou um ensino de qualidade, com professores de excelentíssima capacidade técnica e, acima de tudo, de forma gratuita.

"O homem que evita dúvidas, nunca terá certezas." (Johnnie Walker)

Resumo

O meio eletrônico tem se tornado mais presente na vida da sociedade e cada vez mais os consumidores buscam por comodidade e praticidade ao realizar suas compras. Surge então o *e-commerce*, um modelo de negócio tão moderno que, associado a outras tecnologias, traz consigo a possibilidade de utilização de algoritmos de aprendizado de máquina para alavancar vendas e melhorar a experiência do usuário. Em um contexto atual, onde existem inúmeras empresas que entram em um modelo de negócio tão novo quanto o *e-commerce* e encontram um ambiente extremamente competitivo, um processo eficaz de precificação de produtos é a chave entre o sucesso e o fracasso. Torna-se extremamente interessante a utilização da inteligência artificial para sanar tal problema. Tópicos de aprendizado de máquina são amplamente estudados na literatura e, na prática, treinar algoritmos com conjunto de dados que possuem um número significativo de anomalias (*outliers*) pode levar a resultados indesejados. Este trabalho tem como principal objetivo realizar uma análise exploratória para predizer o preço de itens vendidos por meio do *e-commerce*, analisando o desempenho do aprendizado de um algoritmo antes e após a remoção de *outliers*. Neste trabalho foi utilizado um modelo de Floresta Aleatória (*Random Forest*) associado a algoritmos de detecção de *outliers*, verificando e comparando os resultados obtidos através da técnica de validação cruzada. Por fim, foi possível concluir que existe uma melhora significativa quando se associa algoritmos de detecção de *outlier* a um modelo de Floresta Aleatória.

Palavras-chaves: Inteligência Artificial, Aprendizado de Máquina, Árvores Aleatórias, detecção de *outliers* e *e-commerce*.

Abstract

The electronic medium has become more present in society's life and consumers are increasingly looking for convenience and convenience when shopping. Then comes e-commerce, a business model so modern that, associated with other technologies, brings with it the possibility of using machine learning algorithms to leverage sales and improve the user experience. In a current context, where there are numerous companies that enter a business model as new as e-commerce and find an extremely competitive environment, an effective product pricing process is the key between success and failure. It becomes extremely interesting to use artificial intelligence to solve this problem. Machine learning topics are widely studied in the literature and, in practice, training algorithms with data sets that have a significant number of anomalies (outliers) can lead to undesired results. The main objective of this work is to carry out an exploratory analysis to predict the price of items sold through e-commerce, analyzing the performance of learning an algorithm before and after removing outliers. In this work, a Random Forest Random Forest model associated with outlier detection algorithms was used, verifying and comparing the results obtained through the cross-validation technique. Finally, it was possible to conclude that there is a significant improvement when associating outlier detection algorithms to a Random Forest model.

Key-words: machine learning, artificial intelligence, algorithms, outliers, e-commerce.

Lista de ilustrações

Lista de abreviaturas e siglas

LOF	Local Outlier Factor
iForest	Isolation Forest
COPOD	Copula-Based Outlier Detection
KDEOS	Kernel Density Estimation Outlier Score
HDBSCAN	Hierarchical Density-based Spatial Clustering of Applications with Noise
GLOSH	Global-Local Outlier Score from Hierarchies
EQM	Erro Quadrático Médio

Sumário

1 Introdução

Sempre foi inerente à concepção humana a ideia de que comodidade e avanços científicos são indissociáveis. A internet, produto de avanços tecnológicos da humanidade ao decorrer de sua evolução científica e bélica, trouxe aos seres humanos a possibilidade de um mundo extremamente interconectado. Desde celulares, *tablets*, *notebooks* até lâmpadas, dispositivos médicos e relógios de pulso, todos podem ser conectados à internet. Vivemos em um mundo onde tudo é extremamente rápido e prático e, claro, essa afirmativa não poderia ser diferente quando se trata do setor comercial.

Segundo Moura et al. (2015), o *e-commerce* é um modelo de negócio adaptado a uma realidade onde existem conexões intermitentes à internet e há grande necessidade, por parte dos clientes, de presteza durante o processo de compra e economia de tempo quando se trata de translado do cliente até a loja física. Trata-se, portanto, de uma nova experiência que proporciona ao cliente não só a possibilidade de um acompanhamento do seu pedido em tempo real, desde a compra até a entrega, mas uma melhor qualidade de vida e melhores preços.

De acordo com uma pesquisa realizada pela empresa brasileira Conversion (2021), o *e-commerce* no Brasil bateu recordes surpreendentes no mês de Abril de 2021, quando comparado ao mesmo mês do ano anterior. Com mais de 1,66 bilhão de acessos e um crescimento superior a 40%, ele vem se provando um modelo de negócio extremamente importante no cotidiano dos brasileiros, especialmente em tempos de pandemia.

Assim como afirma Chiavenato (2019), a internet proporciona às empresas a capacidade de tornar-se cada vez mais ágil em seus processos internos, reformulando seu modelo logístico e proporcionando aos clientes um novo modelo relacional, o B2C (*business to customer*) onde o foco é modelar as relações entre o negócio e o cliente, como um consumidor final.

O novo modelo relacional proporcionado pela internet entre empresa-cliente criados à partir do *e-commerce* traz ao cliente uma comunicação com mais presteza e elimina a necessidade de deslocamento até o local onde o vendedor se encontra. De acordo com Ramos (2015), ao se analisar pela perspectiva das empresas, que agora entram em um novo mercado, este se mostra extremamente promissor quanto a redução de custos operacionais. No entanto, também faz-se necessário acentuar que, para essas empresas, trata-se de um mundo repleto de novos desafios logísticos, relacionais e comerciais.

Pode-se dizer que agora, como nunca antes visto, o mercado consumidor procura se

basear nos mínimos detalhes para apoiar seu processo decisório. Durante uma compra cada detalhe torna-se extremamente importante, tais como: um comentário negativo realizado por um cliente após uma compra que não foi satisfatória, um preço entre produtos que se diferem na casa das unidades e um frete que se apresenta com preço mais elevado quando comparado aos concorrentes.

Ao se traçar uma linha de raciocínio por todos os exemplos que foram citados anteriormente, pode-se perceber que há um ponto em comum entre eles: todos podem apresentar características que se destoam do comum quando comparado aos seus semelhantes e, portanto, são chamados de *outliers*.

Segundo Behgounia e Zohuri (2020), em um mercado tão competitivo e que caminha de mãos dadas com a tecnologia, não é estranho observar o emprego de aprendizado de máquina no *e-commerce*. Há alguns anos, estão presentes por toda a internet empresas que utilizam inteligencia artificial para alavancar suas vendas e até mesmo prevenir perdas. Podem-se colocar à luz desta narrativa os famosos sistemas de detecção de fraudes, que se baseiam na detecção de anomalias durante o processo de compra de um determinado cliente. Existem ainda sistemas de recomendação de produtos que se baseiam em pesquisas anteriores feitas por um usuário; sistemas de análise de sentimento que utilizam Processamento de Linguagem Natural (NLP[1]) para verificar quão satisfeitos os clientes estão quanto à compra que fizeram ou sistemas que por meio de Geração de Linguagem Natural (NLG[2]) atuam como atendentes e, com presteza, conseguem resolver problemas de produtos com defeito e atraso em entregas, evitando, assim, que o comprador precise esperar por minutos ou até horas em uma fila telefônica para ser atendido.

Ao se falar sobre aprendizado de máquina (*machine learning*), sempre há espaço para discussão quando se trata de otimização do desempenho de um algoritmo. Neste trabalho, visando um melhor desempenho do algoritmo de aprendizado de máquina, são empregados métodos de detecção de *outliers*. Segundo Liu, Ting e Zhou (2008), *outliers* podem ser definidos como um ou mais dados que apresentam características muito diferentes dos demais dados de uma dada amostra enquanto que, segundo Hawkins (1980), um *outlier* é um dado que se desvia de tamanha forma dos demais dados a ponto de levantar suspeitas de que este dado foi gerado por um mecanismo diferente. Ademais, pode-se acrescentar que treinar um algoritmo com um conjunto de dados onde exista um número considerável de *outliers* deteriora o seu aprendizado.

Existem muitas variáveis que devem ser analisadas antes da venda de um produto, especialmente no *e-commerce*, um ramo tão próspero e com o crescimento acelerado. Ao se perceber que uma loja não vende somente uma mercadoria, é fácil notar que o problema é ainda mais complexo. Segundo Dotto, Moyano et al. (2008), são diversos os custos

[1] Do original, em Inglês, *Natural Language Processing*
[2] Do original, em Inglês, *Natural Language Generation*

a serem levados em consideração antes de se precificar uma mercadoria, sendo alguns deles: o custo de mercadoria adquirida, custo de armazenamento, custo da distribuição, despesas financeiras e tributárias, despesas com o pós-venda e despesas administrativas. Torna-se, então, um processo extremamente trabalhoso e às vezes, inviável, que, sem sombra de dúvidas, existe uma necessidade de ferramentas ágeis que auxiliem no processo de precificação.

Em vista do que foi dito e em como existe uma grande lacuna de ferramentas que consigam ajudar novos e antigos *players* no mercado do *e-commerce*, o presente trabalho tem como principal finalidade desenvolver um modelo de aprendizado que consiga auxiliar no processo de precificação de itens vendidos no meio eletrônico.

1.1 Objetivos

A proposta deste trabalho de conclusão de curso consiste na criação de um modelo de uma ferramenta que auxilia na precificação de mercadorias vendidas via *e-commerce* utilizando um algoritmo de aprendizado de máquina. Durante os estudos foi realizada uma comparação de técnicas de detecção de *outliers* quando associadas ao algoritmo de aprendizado a fim de analisar os impactos desses dados na qualidade do modelo encontrado.

Por objetivos específicos, este trabalho busca:

- Avaliar o desempenho da predição após empregar algoritmos de detecção de *outliers*;
- Avaliar a influência dos *outliers* na assertividade do modelo.

1.2 Justificativa

Em um mercado com crescimento acelerado e demandas cada vez maiores, a entrada de novos empreendedores sem experiência neste ramo de mercado torna-se cada vez mais comum. Neste sentido, não é estranho encontrar empresas que cometem erros fatais à sustentabilidade econômica do seu negócio e acabam declarando falência. Como grande exemplo disso, tem-se a pesquisa feita por Souza (2019), que mostra que de um universo de 10.000 empresas, cerca de 8.900 acabam vendendo seus produtos no prejuízo, resultado de um processo de precificação que não condiz com um valor sustentável de um determinado produto.

Portanto, este trabalho justifica-se em duas principais frentes. A primeira, traduz-se na criação de uma ferramenta inicial que consiga ajudar tanto novos e antigos empreendedores do ramo a precificar corretamente seus produtos, com base em informações básicas sobre o produto e em valores competitivos ofertados por concorrentes. A segunda justificativa, está relacionada ao conhecimento mais amplo que o trabalho traz ao discente

acerca de algoritmos de aprendizado de máquina e de detecção de *outliers*, bem como uma experiência mais robusta em manipulação e análise de dados, que são áreas em crescente ascensão no mundo da tecnologia e inovação.

1.3 Organização e estrutura

O restante do trabalho está organizado da seguinte forma: o Capítulo 2 traz uma revisão bibliográfica sobre todos os métodos de detecção de *outliers* que foram empregados neste trabalho, bem como uma breve introdução sobre o que são *outliers* e aprendizado de máquina. O Capítulo 3 apresenta as ferramentas e como o passo a passo do trabalho foi desenvolvido. No Capítulo 4 são apresentados os resultados de todas as técnicas empregadas neste trabalho, bem como algumas discussões relevantes acerca dos resultados obtidos. Por fim, o Capítulo 5 traz uma conclusão sobre o trabalho e sugestões para trabalhos futuros.

2 Revisão Bibliográfica

Esta seção apresenta alguns conceitos, os algoritmos e as tecnologias utilizadas no trabalho, fornecendo um entendimento mais profundo de como funcionam.

2.1 Trabalhos Relacionados

Como grandes referências de trabalhos similares, é importante ressaltar que são diversas as áreas em que se visa utilizar de algoritmos de aprendizado de máquina para a precificação de itens, desde preços de carros, casas, ações e entre outras coisas. Como este trabalho irá focar especificamente em produtos vendidos em *e-commerce*, selecionou-se dois principais trabalhos.

O primeiro trabalho, denominado *"Deep end-to-end learning for price prediction of second-hand items"* realizado por Fathalla et al. (2020), procura tentar predizer o preço de itens de segunda mão utilizando aprendizado de máquina. Para isso, os autores alimentam um modelo de *deep learning* com uma análise de sentimento de respostas de usuários de compras passadas, características específicas de cada elemento físico e por fim, fotos de cada um dos produtos. Como resultado da junção de todos esses elementos, os autores conseguiram obter 80,95% dos resultados desejados dentro da faixa de preço nas quais cada item foi vendido. Mais informações acerca da obra podem ser encontradas seguindo as referências deste trabalho. Como proposta de trabalhos futuros, os autores sugerem que exista um foco maior na extração das imagens dos itens vendidos, de forma que possa ser feita uma análise acerca da qualidade do produto, visto que por se tratarem de produtos de segunda mão, o preço pode variar de acordo com a condição do produto. Neste ponto, torna-se interessante que sejam utilizados algoritmos de detecção de *outlier* para evitar que imagens que não correspondam aos itens que estão em venda sejam introduzidas na inteligência artificial de tal forma a deteriorar seu desempenho.

O segundo trabalho, intitulado *"Predicting the final prices of online auction items"*, desenvolvido por Xuefeng et al. (2006), procura tentar precificar itens vendidos em leilões online, para isso, os autores criaram um algoritmo extremamente específico para coletar dados de leilões realizados pelo *Ebay* e preparam um *dataset* final que contém informações sobre o vendedor do item (como sua reputação), detalhes do item e um histórico do preço de cada um dos lances dados nos leilões. Com isso feito, o autor utiliza duas formas de tentar predizer os valores dos itens: a primeira consiste em tratar o processo de precificação como um problema de regressão linear e utilizar de coeficientes de regressão para tentar

predizer os preços, obtendo como resultado, um coeficiente de determinação de 0.7944, e o segundo consiste em tratar a precificação como um problema de classificação, para isso, os autores dividiram os possíveis preços dos itens em intervalos de diferentes valores, totalizando nove intervalos que foram numerados de 1 a 9 à medida em que a grandeza da classe se tornava maior. Para encontrar os valores preditos nesta segunda forma de enxergar o problema, o autor utilizou-se de redes neurais, obtendo como resultado, um coeficiente de determinação igual a 0.9129.

Apesar de ambos os trabalhos realizarem algum tipo de tratamento dos dados que serão usados em seus respectivos algoritmos de aprendizado de máquina, nenhum algoritmo de detecção de *outlier* foi utilizado durante o desenvolvimento dos trabalhos.

2.2 *Outlier*

A percepção do significado da palavra *outlier*, de forma intuitiva, já dispõe ao leitor a ideia de algo que não pertence à um determinado conjunto ou agrupamento de informações. De maneira mais técnica, como dito por Hawkins (1980), um ponto pode ser considerado como *outlier* quando se desvia muito dos demais pontos, de maneira a levantar suspeitas de que este ponto pode ter sido gerado por outros mecanismos. Já pela perspectiva de Barnett e Lewis (1984), *outlier* é uma observação, ou até mesmo um conjunto de observações, que parecem ser incosistentes com o restante do conjunto de dados.

2.3 Algoritmos de detecção de *outliers*

Segundo Campos (2015) em sua dissertação, existem duas grandes classes em que se pode dividir os algoritmos de detecção de *outliers*: algoritmos globais e algoritmos locais. Os algoritmos globais, partem do pressuposto que todo o conjunto de dados teve sua origem por meio de um único mecanismo e, portanto, não se preocupam com a localização de um dado ponto em um conjunto de dados. Os algoritmos locais, diferentemente dos globais, se preocupam com a localização de um ponto em um espaço amostral, partindo do conceito de que um conjunto de dados pode ter sido gerado por um ou mais mecanismos e buscam analisar o comportamento da vizinhança ao redor do ponto que está sendo analisado. Nesta subseção, serão apresentados alguns algoritmos de detecção de *outliers* que foram empregados neste trabalho.

O critério de escolha para a seleção dos algoritmos utilizados neste trabalho baseou-se na ideia de utilização tanto de algoritmos locais como algoritmos globais, de forma que ao se apresentar os resultados, diferenças pudessem ser percebidas em relação a como as grandes classes (globais e locais) se comportam e percebem os *outliers*. Além

disso, com o crescente interesse em tópicos de aprendizado de máquina, cada um dos algoritmos utilizados neste trabalho acabaram tornando-se extremamente populares entre a comunidade de Ciência de Dados, ganhando publicações e tutoriais de implementações em sites de renome internacional, como Towards Data Science, por exemplo.

2.3.1 *Local Outlier Factor* - LOF (Local)

Pode-se dizer que o *Local Outlier Factor* (LOF) foi um dos primeiros algoritmos de detecção de *outliers* locais que foi criado. Baseados nele, diversos outros algoritmos de detecção local foram criados e, surpreendentemente, até hoje, o LOF consegue demonstrar ótimo desempenho quando comparado a algoritmos mais recentes.

O LOF identifica um ponto p como *outlier* com base na densidade de sua vizinhança. Isto é, este ponto p, será considerado *outlier* se a densidade ao seu redor for consideravelmente menor do que a densidade ao redor do seus pontos vizinhos.

Antes de explicar o funcionamento do algoritmo, é importante descrever algumas definições levantadas por Breunig et al. (2000).

Definição 1: Seja, p e o, dois objetos distintos em um espaço de amostras,k qualquer valor natural positivo e $\mathbf{D}$ um conjunto amostral. Segundo Breunig et al. (2000), a $k - distance$ de um objeto p, denotada pelos autores como $k - distance(p)$, é descrita como a distância entre o objeto p e o objeto o, $d(p, o) \in \mathbf{D}$, de maneira que atenda a duas condições:

- Existam, no mínimo, k objetos $o' \in \mathbf{D}$, de tal maneira que $d(p, o') \leq d(p, o)$;
- No máximo, $k - 1$ objetos $o' \in \mathbf{D}$ possuem $d(p, o') \leq d(p, o)$.

Definição 2: Dada uma $k - distance$ de um ponto p, a vizinhança do ponto p irá conter todos os objetos dos quais a sua distância até o ponto p não for superior ao valor de $k - distance$.

O autor nomeia a definição 2 de $k - distance\, neighborhood$ e é denotada como $N_{k-distance(p)}(p)$. A Figura 1 ilustra o que foi apresentado nas Definições 1 e 2, mostrando que os pontos B, D e C são considerados vizinhos do ponto A, uma vez que a distância do ponto A até cada um deles não é maior do que $k - distance(A)$ de 2.

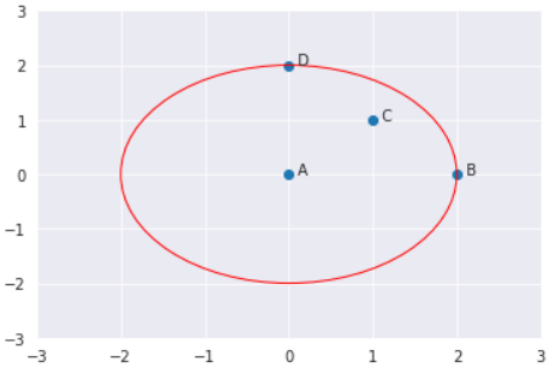


Figura 1: Exemplo de $k-distance$ com valor de $k = 2$.
Fonte: Saul (2020)

Definição 3: A distância de alcançabilidade de um objeto o ou *reachability distance*, é dada como a verdadeira distância entre o ponto p e o ponto o caso este se encontre muito distante do ponto p. Neste caso, considera-se um ponto distante todo o ponto que supere o valor de $k-distance(p)$. No caso em que os objetos são suficientemente próximos, ou seja, $d(p,o) \leq k-distance(p)$, a distância de alcançabilidade assume o valor de $k-distance(p)$. Formalmente, esta definição é escrita como:

$$reachDist_k(p,o) = \max\{k\text{-distance}(p), d(p,o)\} \tag{2.1}$$

Definição 4: A densidade de alcançabilidade local de um objeto p é definida por:

$$lrd_{MinPts}(p) = \frac{1}{\left(\sum\limits_{o \in N_{MinPts}(p)} reachDist_{MinPts}(p,o)\right)}, \tag{2.2}$$

onde N_{MinPts} significa um número mínimo, inteiro e positivo objetos próximos do ponto p. Por meio dessa fórmula, é possível concluir que, quanto menor for o valor da densidade de alcançabilidade local, menor vai ser a densidade de pontos ou, em Inglês, *clusters* perto do objeto p, ou seja, significa dizer que o *cluster* mais próximo está longe do objeto p.

Definição 5: O fator de *outlier* local ou *Local Outlier Factor (LOF)* de um objeto p é dado por:

$$LOF_{MinPts}(p) = \left(\frac{\sum\limits_{o \in N_{MinPts}(p)} \frac{reachDist_{MinPts}(p)}{reachDist_{MinPts}(o)}}{|N_{MinPts}(p)|} \right) \tag{2.3}$$

A Equação 2.3 captura o grau em que se pode afirmar que o ponto p é considerado como um *outlier*. O LOF é, então, a razão entre a média da densidade de alcançabilidade local do ponto p e os seus N_{MinPts} vizinhos mais próximos.

Com as definições explicadas nesta subseção, é possível perceber que para esse método, um ponto não será considerado como *outlier* quando a proporção da densidade de alcançabilidade local do ponto p e a proporção da densidade de alcançabilidade local dos seus N_{MinPts} vizinhos mais próximos forem aproximadamente iguais. Neste caso, o valor de LOF será próximo a 1. Geralmente, quando $LOF > 1$, um ponto é então considerado como *outlier*.

2.3.2 *Isolation Forest* - iForest (Local)

Diferentemente da maioria dos algoritmos de detecção de *outlier* que utilizam o princípio de árvores de decisão para encontrar observações que de fato pertençam a uma base dados e, ao fim de sua execução, considera que todas as demais observações que não foram selecionadas são consideradas como *outlier*, o Isolation Forest ou iForest, foca-se em isolar os *outliers*. De acordo com Liu, Ting e Zhou (2008), o algoritmo proposto tende a apresentar uma capacidade de lidar com bases de dados extremamente grandes e com inúmeros atributos irrelevantes, o que o torna extremamente interessante de ser aplicado a este trabalho.

O funcionamento do algoritmo iForest parte do entendimento de que como *outliers* são a minoria e se destoam em um conjunto de dados, estes são mais suscetíveis a serem isolados, ou seja, são mais suscetíveis a serem separados dos demais pontos. Para isso, o algoritmo iForest busca particionar um determinado conjunto de dados de maneira aleatória, selecionando aleatoriamente um atributo e por fim, definindo um valor também aleatório da proporção da divisão a ser feita sobre o conjunto de dados. Este valor, no entanto, irá variar entre o valor máximo e mínimo do atributo selecionado. Esse processo acontece recursivamente, até que todos os pontos consigam ser isolados.

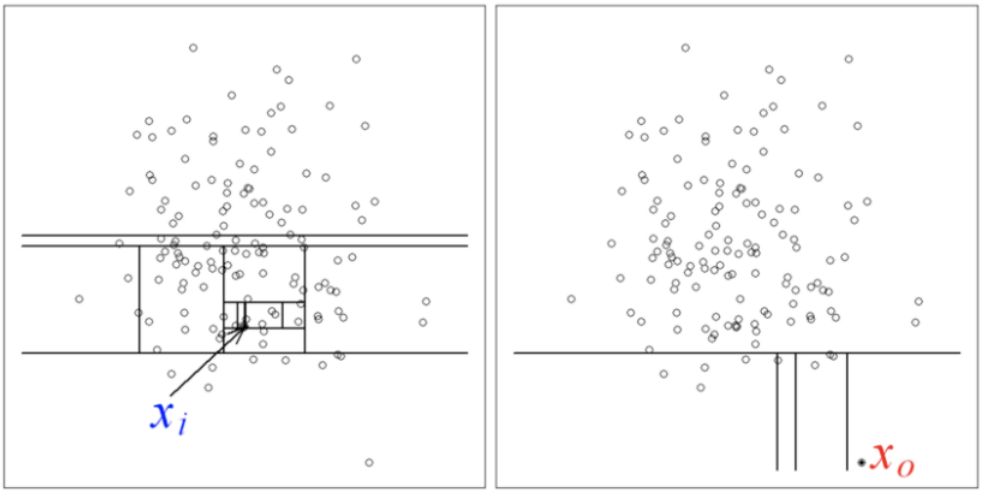


Figura 2: Exemplo de como o iForest isola objetos
Fonte: Liu, Ting e Zhou (2008).

Como está destacado na Figura 2, o ponto x_i, que é considerado como um *inlier*, requer que o algoritmo particione muito mais vezes quando comparado ao ponto x_o, que é tido como um *outlier*.

Em um aspecto um pouco mais profundo, o iForest é um algoritmo que utiliza árvores de decisão binárias. De maneira simples, pode-se dizer que árvores de decisão binárias são árvores em que cada nó tem, no máximo, dois filhos. Dito isso, uma árvore de decisão no iForest irá crescer até conseguir isolar cada ponto de um determinado conjunto de dados. Outro ponto a ser levantado é que, como ele particiona o conjunto de dados de forma aleatória, as árvores são geradas com diferentes conjuntos de partições. Portanto, define-se que um caminho médio até um ponto p, como a distância média entre o número de arestas que são atravessadas desde a raiz da árvore de decisão até o nó mais externo que, no caso, é o ponto p.

A função que determinará se um ponto p em um conjunto de dados com n instâncias será considerado como *outlier* ou não, é dada por:

$$s(p, n) = 2^{-\frac{E(h(p))}{c(n)}}, \tag{2.4}$$

onde $E(h(p))$ é o caminho médio até o ponto p entre uma coleção de árvores ou, como o próprio autor as cita, *isolation trees* e $(c(n))$ é o caminho médio até p dadas n instancias. Pode-se calcular $(c(n))$ por:

$$c(n) = 2H(n-1) - \left(\frac{2(n-1)}{n}\right), \tag{2.5}$$

onde H é o número harmônico e pode ser estimado por $ln(n) = 0.5772156649$. Com isso, é possível dizer que por meio da equação 2.4, um ponto p será considerado *outlier* se:

- Se s for um valor próximo a 1, então é provável que este ponto seja um *outlier*;
- Se s for um valor muito menor que 0.5, então é possível afirmar com certa segurança que o ponto é um *inlier*;
- Se s for um valor próximo de 0.5 ($s \simeq 0.5$), então toda a amostra não possui nenhuma anomalia distinta.

2.3.3 *Copula-Based Outlier Detection* - COPOD (Global)

Antes de se iniciar o estudo sobre o *Copula-Based Outlier Detection* (COPOD), existem algumas definições que são de suma importância para o completo entendimento do funcionamento do algoritmo.

Definição 1: Segundo Whapole e Ye (2009), define-se como função de distribuição acumulada a função que descreve a probabilidade de uma variável aleatória X assumir um valor menor ou igual a x. Matematicamente, descreve-se por:

$$F(x) = P(X \leq x) \tag{2.6}$$

Definição 2: De acordo com Cortivo (2019), uma distribuição marginal de um vetor aleatório X_j com $j = 1, 2, \ldots, n$ pode ser definida matematicamente por:

$$p_{x_j}(x_j) = P(X_j = x_j) = \sum_{x_i} p(x_i) = \sum_{x_i} P\left(X_1 = x_1, ..., X_n = x_n\right) \tag{2.7}$$

Em outras palavras, uma distribuição marginal de variáveis pode ser vista como a distribuição de probabilidade de cada uma das variáveis contida em um conjunto de multivariáveis.

Definição 3: Na visão de Li et al. (2020), cópulas são funções que permitem a separação entre a distribuição marginal e a estrutura de dependência de um conjunto de dados com multivariáveis. Ao se fazer esse processo para cada uma das variáveis, é possível descrever a dependência entre cada uma delas. Em uma visão matemática, o autor descreve que uma cópula pode ser descrita como uma função de distribuição cumulativa de um vetor aleatório, por exemplo, $U_1, U_2, U_3..., U_d$ com marginais uniformes (entre 0 e 1). A equação abaixo ilustra a visão matemática do autor:

$$C_u(u) = \mathbf{P}(U_1 \leq u_1, \ldots, U_d \leq u_d), \tag{2.8}$$

onde para uma cópula com d-variáveis, a probabilidade de um vetor j será dada por $P(U_j \leq u_j) = u_j$, onde $j \in [1, ..., d]$ e $u_j \in [0, 1]$.

Definição 4: Uma cópula empírica ou *empirical copula*, é uma cópula que é ajustada de acordo com uma função de distribuição cumulativa empírica, matematicamente expressa por:

$$\hat{F} = \mathbf{P}((-\infty, x]) = \frac{1}{n} \sum_{i=1}^{n} \prod (X_i \leq x) \tag{2.9}$$

Com essas definições, é possível avançar na descrição do algoritmo. O COPOD baseia-se em um processo de três estágios, dos quais, no primeiro, espera-se um conjunto de dados com d-dimensões, sendo escrito por $X = (X_{1,i}, X_{2,i}, X_{3,i}, X_{d,i})$ onde $i = 1, ..., n$. Uma vez que o conjunto de dados seja recebido, são construídas as funções de distribuição acumulada. Como um segundo estágio, constrói-se as cópulas empíricas. O terceiro e último passo possui alguns pontos que tornam-se interessantes em ser explicados, por tanto, passa-se a um detalhamento mais aprofundado sobre ele.

A última etapa de execução do algoritmo parte do princípio de que *outliers* são considerados um evento de cauda. Em outras palavras, como um exemplo, toma-se uma

determinada observação x_i , assumindo que esta observação atende a uma determinada função de distribuição, F_x, objetiva-se encontrar uma outra observação tão distante quanto x_i. Como *outliers* são tratados como eventos de cauda, então é necessário calcular a probabilidade de eventos à esquerda, $F_x(x_i) = \mathbf{P}(X \leq x_i)$, e à direita, $1 - F_x(x_i) = \mathbf{P}(X \geq x_i)$.

Como base de dados de grandes dimensões sofrem interferências de diversos fenômenos, os autores descrevem uma ferramenta auxiliar chamada de fator correção de distorção. Esta ferramenta procura verificar a concentração de pontos para uma determinada distribuição, quer ela esteja na cauda direita ou esquerda. Matematicamente, descreve-se por:

$$b_i = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \bar{x}_i\right)^3}{\sqrt[3]{\frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \bar{x}_i\right)^2}}, \tag{2.10}$$

onde n é o número de observações do conjunto de dados. Em termos práticos, para valores negativos de b, tem-se que a cauda direita é maior do que a cauda esquerda e, nesse caso, faz mais sentido utilizar a cópula empírica da cauda esquerda. De maneira análoga, para valores positivos de b, faz mais sentido utilizar a cópula empírica da cauda direita. Desta forma, uma cópula será composta por três componentes: a cauda esquerda, a cauda direita e o fator de correção de distorção. Outro ponto importante a se destacar é que, ainda visando a diminuição das interferências sofridas por grandes conjuntos de dados, o autor trabalha com o somatório negativo das probabilidades. Então, da equação 2.9, tem-se que:

$$-log(\tilde{C}(x)) = -\sum_{j=1}^{d} log(x_j), \tag{2.11}$$

onde d é dimensão do conjunto de dados.

Após fazer todos os cálculos quanto ao fator de correção, a probabilidade de cauda à esquerda e à direita, o COPOD retorna o valor máximo entre o logaritmo negativo obtido nas caudas à direita, à esquerda e o fator de correção. Este retorno é chamado de *score* e, para este algoritmo, quanto maior o *score* de uma determinada observação, maior será a probabilidade que esta seja um *outlier*.

2.3.4 *Kernel Density Estimation Outlier Score* - KDEOS (Local)

O *Kernel Density Estimation Outlier Score* (KDEOS) é um algoritmo que pode ser classificado como um algoritmo local, uma vez que ele busca detectar *outliers* por meio de estimativas de densidade do *kernel*. Antes de aprofundar sobre o tema, é importante pontuar algumas definições.

Definição 1: Segundo Siegel e Jr (1975), métodos estatísticos não paramétricos são métodos que se baseiam em suposições gerais acerca da informação de uma dada

amostra extraída. Nestas situações, não existem informações específicas sobre a população da qual a amostra foi extraída.

Definição 2: De acordo com Scott e Longuet-Higgins (1990), um estimador de densidade por *kernel* baseia-se no conceito de matrizes de afinidade. Estas matrizes são compostas por um vetor que representa a afinidade entre variáveis, normalmente definido por **A**, onde cada elemento a_{ij} contido em **A**, representa uma medida de similaridade entre duas observações de uma base de dados.

O estimador de densidade por *kernel* é calculado por meio de uma função chamada de função de *kernel* (PARZEN, 1962), podendo ser descrita de acordo com o tipo de problema com que se está lidando. Esta função centraliza-se em cada uma das observações de uma base de dados. Segundo Wanderley e Barbosa (2013), uma estimativa de densidade de uma observação x_i é descrita por meio de uma determinada relação espacial. Esta estimativa é denotada como $\hat{f}(x_i)$. Como dito anteriormente, a relação espacial entre as observações de um conjunto de dados é embutida por meio de uma função de *kernel*.

Após serem esclarecidas as definições citadas nesta seção, é possível compreender alguns aspectos utilizados no desenvolvimento do algoritmo. O KDEOS pode ser dividido como um algoritmo que tem duas fases, a primeira responsável por realizar a estimativa de densidade e a segunda, responsável por comparar a densidade com os vizinhos locais.

A primeira fase, consiste na escolha de uma função de *kernel*. Conforme Schubert, Zimek e Kriegel (2014), a escolha entre o tipo de *kernel* a ser adotado deveria ser tratado como um *input*, onde fica a critério de quem irá utilizar o algoritmo escolher aquele de sua preferência. Ainda na primeira parte, após a escolha da função de *kernel*, é aplicado a estimativa da densidade do *kernel*. Como se está tratando de *kernels* fixos, de acordo com os autores e também por uma pesquisa feita por Terrell e Scott (1992), a melhor estimativa de densidade do *kernel* é chamada de *ballon estimation*. Ao se transformar o *ballon estimation* em um problema com múltiplas observações, tem-se:

$$KDE_{balloon,h}(o) = \frac{1}{n} \sum_{p \in \mathrm{KNN}(o)} K_{h(o)}(o - p), \tag{2.12}$$

onde o é o ponto que está sendo analisado, n é o tamanho da amostra e p são os vizinhos próximos a esse ponto. O cálculo da largura de banda sugerida neste método baseia-se na seguinte equação:

$$h(o) = \min\{mean_{p \in kNN} d(p, o)\}, \tag{2.13}$$

onde para a escolha dos k-vizinhos mais próximos, que pode ser uma tarefa extremamente difícil de se realizar, é definido um *range* de valores de que serão testados um a um, de forma a produzir uma série de estimativas de densidade.

Na segunda etapa da execução do algoritmo, aplica-se uma normalização para padronizar os dados. Neste método, emprega-se a técnica do Z-score, matematicamente

descrita por:

$$z(x, X) := \frac{(x - \mu_x)}{\sigma_x}, \tag{2.14}$$

onde μ_x é a media de X e σ_x é o desvio padrão.

Como existem múltiplos valores de densidade uma vez que existe uma série k de densidades para um mesmo ponto, usa-se a média do valor do Z-score para retornar um *score* para uma dada observação. Então, a equação 2.14 se torna:

$$s(o) = \underset{k_{min},\ldots,k_{max}}{mean} z(KDE(o), \{KDE(p)\}_{p \in kNN(o)}). \tag{2.15}$$

Após serem atribuídos a cada uma das observações um *score*, é feita mais uma normalização de cada um desses valores, de maneira que o novo *range* desses valores fiquem entre [0,1], onde 0 significa que o ponto é considerado como *inlier* e 1 como *outlier*.

2.3.5 *Hierarchical Density-based Spatial Clustering of Applications with Noise* - HDBSCAN (Local)

De maneira similar aos outros métodos de detecção de outliers apresentados nesta seção, apresentamos alguns aspectos teóricos do HDBSCAN antes de explicar o seu funcionamento.

Definição 1: Conforme Morais (2018), uma árvore geradora miníma ou *minimum spanning tree* (MST), pode ser representada como um algoritmo que constrói grafos, conectando dois vértices, de forma a identificar as conexões mais importantes (com o menor peso). Ao final da execução do algoritmo, espera-se um grafo que possua um número mínimo de ligações que consigam manter os vértices conectados com o menor custo possível.

O HDBSCAN é considerado um algoritmo de clusterização, ou seja, pode-se dizer que o algoritmo procura agrupar elementos que possuem características comuns entre si, isolando-os dos demais outros elementos. A ideia, ao final do algoritmo, é conseguir conectar cada um desses *clusters* por meio de observações, de forma a construir, com o menor peso possível, uma MST que ligue todo o conjunto de dados, elencando cada um dos *clusters* de forma hierárquica e não paramétrica, ou seja, não existe a necessidade de definir parâmetros. Com esta visão generalista sobre o funcionamento do algoritmo, é possível prosseguir para uma explicação mais profunda sobre como funciona cada etapa de seu funcionamento.

O primeiro passo na execução do algoritmo consiste na estimativa da densidade das observações presentes no conjunto de dados. Como nos outros métodos citados nesse trabalho, o HDBSCAN utiliza da técnica dos k-vizinhos mais próximos, que consiste na criação de círculos de diferentes raios, que contenham pelo menos uma determinada quantidade de observações, denotado por min_{pts}. A Figura 3, ilustra o que foi explicado neste parágrafo para uma quantidade de 7 vizinhos mais próximos.

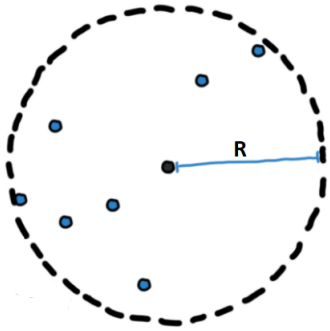
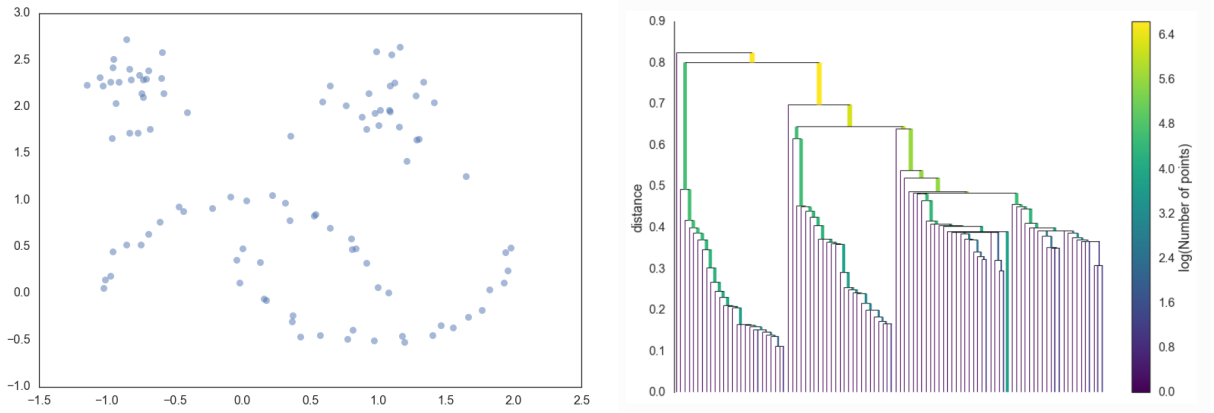


Figura 3: Exemplo com 7 vizinhos mais próximos.
Fonte: Adaptado de Berba (2020).

Retomando a etapa de construção da conexão entre os pontos de uma determinada base de dados, a intenção é construir uma MST. Para isso, as observações de cada conjunto de dados serão vistas como vértices e uma ligação entre dois vértices terá um peso igual a sua distância de alcançabilidade entre os dois pontos.

Partindo para a construção de um *cluster* organizado hierarquicamente, agora o algoritmo passa a conectar os vértices da base de dados. Para isso, a métrica de distância torna-se importante, uma vez que os vértices serão ordenados em uma fila de maneira crescente de acordo com sua respectiva distância de alcançabilidade mútua. Em seguida, é feita uma iteração por essa fila, ligando dois vértices que irão formar um *cluster*. Durante o processo de ligação entre os vértices, existem alguns pontos que não se conectam a nenhum vértice, estes são chamados de ruídos. Esse processo é feito até que toda a árvore hierárquica consiga ser construída. A Figura 4 ilustra um exemplo de uma árvore hierárquica feita para uma determinada base dados com 100 amostras aleatórias.



(a) Plot 2D de um conjunto de 100 observações aleatórias

(b) Processo de criação da árvore hierárquica

Figura 4: Construção de uma árvore hierárquica.
Fonte: Saul (2017).

O que se espera agora é que exista uma corte horizontal entre os níveis hierárquicos

da árvore, de maneira que este corte seja feito e consiga representar dados significativos. Para que isso seja feito, o algoritmo passa por um processo de condensação dos *clusters*, onde um parâmetro do algoritmo chamado de *minimum cluster size* torna-se importante. Este parâmetro irá definir o número mínimo de observações que um *cluster* deve conter para que ele possa se ramificar, caso contrário, aquele *cluster* não deverá gerar filhos, ou *clusters* hierarquicamente inferiores, e será considerado como um nó final. Ao final deste processo de condensação, é possível imaginar que grande parte dos *clusters* gerados pela Figura 4, principalmente em distâncias extremamente baixas, irão começar a se aglutinar em *clusters* maiores.

Depois da condensação dos *clusters*, o algoritmo busca por cortes na nova árvore hierárquica para seleção de grupos significativos. Para isso, cria-se uma nova métrica que determina o quão representativo um determinado *cluster* é. Para isso, define-se a seguinte equação:

$$\lambda = \frac{1}{distance}. \tag{2.16}$$

Com esta nova métrica, serão definidos dois novos termos: λ_p e $\lambda_{nascimento}$. O λ_p representa o momento em que eventualmente um ponto p, existente em um *cluster*, irá se desvincular desse. Este processo de desvinculação acontece porque a medida com que a árvore vai sendo construída, acontece o processo de iteração entre a fila que contém os valores de alcançabilidade mútua construída na fase anterior. O $\lambda_{nascimento}$, pode ser visto como o valor em que um novo *cluster* é formado. Com isso, pode-se definir o que se chama de estabilidade de um *cluster* por:

$$s = \sum_{p \in cluster} (\lambda_p - \lambda_{nascimento}) \tag{2.17}$$

Agora, parte-se para a etapa final do algoritmo, os cortes na árvore (*prunning*). O algoritmo então poderá tomar duas decisões, sendo elas:

1. Se a soma das estabilidades dos *clusters* filhos forem maiores que a estabilidade do *cluster* que os originou, então a estabilidade daquele *cluster* será o somatório das estabilidades dos *clusters* filhos e todos eles serão mantidos.

2. Em caso contrário, onde a estabilidade do *cluster* pai for maior que a soma da estabilidade dos *clusters* filhos, então todos os *clusters* filhos serão cortados e só sobrará o *cluster* pai.

Depois de realizar todo o processo de clusterização do algoritmo, é possível fazer a detecção dos *outliers*. O HDBCAN utiliza o algoritmo *Global-Local Outlier Score from*

Hierarchies (GLOSH), idealizado por Campello et al. (2015), para conseguir identificar *outliers*. A equação que define GLOSH pode ser descrita como:

$$GLOSH = 1 - \frac{\varepsilon_{max(x_i)}}{\varepsilon\left(x_i\right)} \tag{2.18}$$

Onde $\varepsilon_{(x_i)}$ representa o menor raio ao qual uma observação x_i ainda será considerada pertencente ao *cluster* que está sendo analisado pelo algoritmo no momento e $\varepsilon_{max(x_i)}$ será o valor do menor raio ao qual o *cluster* ou qualquer um de seus *clusters* filhos ainda existirão durante o processo de podagem da árvore.

O algoritmo criado por Campello et al. (2015), no entanto, retorna um score para cada observação presente em uma base de dados, ranqueando todas as observações presentes. Com este *ranking*, quanto mais próximo de 0 uma observação for classificada, maior a chance da mesma ser considerada como um *inlier*. De forma análoga, quanto mais distante de 0 e maior for o *score* de uma observação, mais provável de ser considerada como um *outlier* ela é.

Para um estudo mais aprofundado acerca do HDBSCAN e como o algoritmo utiliza de formulas matemáticas para performar os seus processos, mais informações estão disponíveis na obra de Campello et al. (2015).

2.4 Aprendizado de máquina

Nesta seção, é apresentado o método de aprendizado de máquina utilizado neste trabalho.

2.4.1 Árvores de decisão

Idealizadas por Breiman et al. (1984), árvores de decisões são modelos de aprendizado de máquina supervisionados, ou seja, para que funcionem de maneira adequada, são necessários que antes sejam fornecidos dados prévios contendo todos os atributos de um conjunto de dados, bem como os resultados da variável na qual se deseja avaliar. Nesta seção, descreve-se sobre o modelo de árvores de regressão.

2.4.1.1 Árvore de regressão

Os problemas de regressão em árvores de decisão são problemas onde se deseja predizer valores de uma variável numérica alvo, dado um conjunto de atributos. Para este tipo de problema, denomina-se por *Regression Tree*, uma árvore de decisão que se baseia em modelos de regressão.

Uma árvore de regressão, de forma simples, pode ser dito como um método que estratifica um conjunto de dados e por meio de uma sequência de iterações e de decisões que visam maximizar a assertividade do algoritmo, continua a dividir um conjunto de dados em parcelas menores até chegar à um valor final que mais se adéque ao desejado; chama-se esse resultado final de folha (*leaf*). Ou seja, esse método tenta aprender por inferência regras que constituem cada decisão tomada durante a construção da árvore de regressão.

O algoritmo, em sua aplicação para uma base de dados que possui um conjunto multivariado de atributos, busca primeiramente decidir qual será o melhor atributo que definirá o primeiro estado da árvore de decisão, chamado de raiz. A raiz nada mais é do que a base da árvore de decisão e a etapa inicial de todo o processo de "escolhas" que o algoritmo irá realizar.

Pode-se dizer que árvores de regressão fazem parte de um processo iterativo de escolhas, que sempre buscam encontrar os melhores valores possíveis para tentar predizer a variável alvo. Neste processo, o algoritmo sempre busca, por meio de diversos critérios, isolar observações presentes no conjunto de dados e a cada iteração, refinar cada vez mais a acurácia da predição da variável alvo.

Para fins explicativos, a Figura 5 mostra parte de uma árvore de decisão, de cima para baixo. Ao se observar bem, pode-se perceber que a raiz da árvore é o primeiro balão e que, assim como os outros, possuem quatro características, sendo elas: *customer_city*, *mse*,*samples* e *values*.

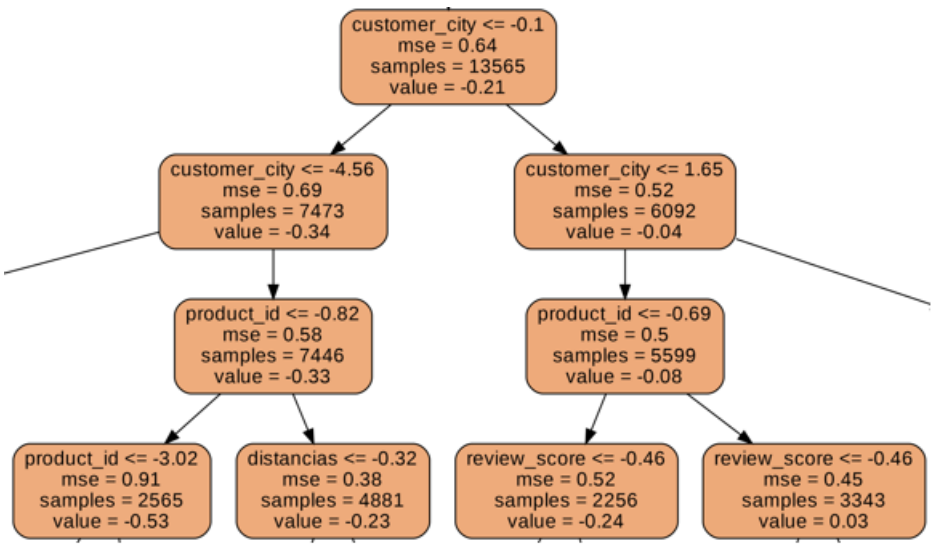


Figura 5: Construção de uma árvore de decisão.
Fonte: Elaborado pelo autor.

O primeiro elemento da linha presente em cada balão, representa um dos atributos presentes no conjunto de dados onde o algoritmo está sendo aplicado. Este atributo é

sempre selecionado de maneira comparativa, ou seja, o algoritmo sempre vai realizar métricas de seleção que comparem cada um dos atributos de maneira com que o melhor atributo sempre seja escolhido de acordo com o ramo em que se encontra na árvore de decisão. Este primeiro elemento serve como um critério, demonstrado por uma inequação, que separa as observações presentes no conjunto de dados.

À medida em que a árvore se ramifica, o algoritmo objetiva tentar encontrar os melhores valores possíveis de *mse*, uma métrica de avaliação que mede o erro quadrático médio entre as observações e que será explicada posteriormente neste Capítulo. De forma simples, quanto menor o valor de *mse*, mais próximo do valor desejado a predição se encontra.

Por fim, tem-se o parâmetro *samples* que indicam quantas observações estão contidas naquela ramificação de acordo com o critério de divisão adotado pelo algoritmo e disposto na primeira linha do balão.

Como dito anteriormente, o algoritmo tentará isolar amostras visando obter melhores predições e, quando o algoritmo chega à uma resposta final, este segmento será chamado de folha. É importante ressaltar que, à medida com que acontecem as iterações, o número de *samples* nunca vai ser menor do que o valor estabelecido por um dos parâmetros do algoritmo chamado de *min_samples_split*. Isto acontece para que o algoritmo não encontre respostas enviesadas e não ocorra um processo de sobreajuste (*overfit*) durante a fase de aprendizado da árvore.

2.4.2 Floresta Aleatória

Uma floresta aleatória ou *random forest*, é um algoritmo que se baseia no conceito de aprendizado em conjunto e utiliza de árvores de decisão para fazer isso. Este algoritmo parte do princípio de que há grandes vantagens em se aprender por meio de um conjunto de diferentes árvores de decisão.

Como é possível observar por meio da Figura 6, o algoritmo, a partir de um conjunto de dados, cria diferentes árvores de decisão, onde cada uma delas é alimentada por uma amostragem diferente de atributos/observações. Isso permite obter a diversidade de regras que podem ser utilizada na definição de um rótulo.

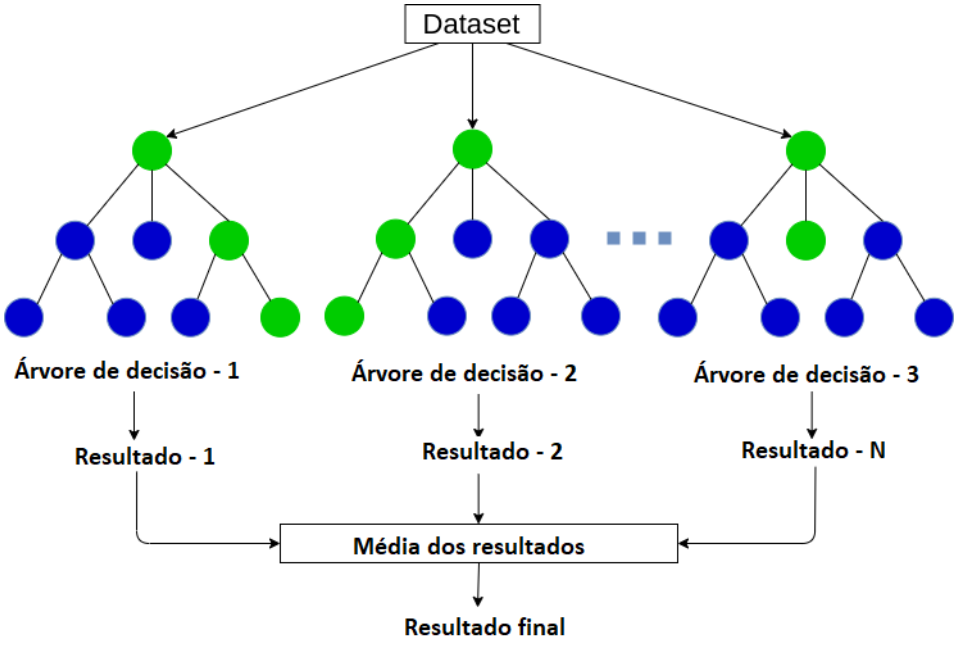


Figura 6: Construção de uma *random forest*.
Fonte: Adaptado de Ampadu (2021).

Neste sentido, é importante ressaltar que o processo de alimentar cada uma das árvores com diferentes amostras aleatórias não significa dizer que se está sub dividindo o conjunto de dados em diferentes partes que serão designadas para cada uma das árvores, mas sim, significa dizer que para um conjunto de dados contendo X observações para treino, cada árvore será alimentada com X observações, no entanto, algumas dessas observações poderão ter seus valores substituídos por outros valores repetidos. Este conceito é conhecido como *bagging* e fica mais claro com um exemplo: um conjunto de dados de treino N que contém as observações de valores $\{1, 2, 3, 4, 5, 6\}$ alimenta duas árvores, uma árvore aleatória A_1 será alimentada com $\{1, 2, 3, 5, 5, 2\}$ enquanto uma outra árvore A_2 será alimentada com $\{1, 3, 3, 4, 5, 6\}$. O que acontece é que as duas árvores recebem a mesma quantidade de observações presentes na base de dados de treino, no entanto, são passados valores repetidos para que hajam diferenças entre as árvores.

Além disso, o algoritmo também trata os atributos presentes no conjunto de dados de maneira diferente. Para cada árvore aleatória gerada, são distribuídos diferentes atributos para cada uma, de maneira a conseguir captar ainda mais diferentes perspectivas de aprendizado.

A partir deste ponto, todo o processo de execução de cada árvore é exatamente igual à árvores de regressão comuns. Cada árvore, ao fim de sua execução irá gerar diferentes resultados de predição. O resultado final de predição do algoritmo será então a média de todas as predições realizadas pelas árvores aleatórias. Para uma visão matemática mais descritiva sobre como o algoritmo funciona, mais informações estão disponíveis em Breiman (2001).

Pode-se dizer que a simplicidade de implementação de uma Floresta Aleatória, bem como a sua capacidade de conseguir lidar com a maioria dos problemas aos quais lhes são propostos, faz com que este seja um algoritmo muito utilizado por cientistas de dados. Desta forma, optou-se por utilizar este algoritmo para o desenvolvimento deste trabalho.

2.5 Técnicas de validação cruzada

Quando se fala sobre implementação de algoritmos de aprendizado de máquina, não é de se estranha ouvir falar sobre técnicas de validação cruzada. Estas técnicas objetivam avaliar a assertividade de modelos, evitando que ocorra resultados indesejados, como por exemplo, um determinado modelo conseguir predizer bem sobre os dados de treino e deixar a desejar predizendo sobre os dados de teste, este fenômeno é conhecido como *overfitting*.

Neste trabalho, a técnica de validação cruzada implementada é conhecida como K-Fold. Este método consiste em primeiramente misturar todo o conjunto de dados de maneira aleatória e então dividi-lo em k partes diferentes.

Como é possível observar pela Figura 7, para um valor de $k = 5$, todo o conjunto de dados será dividido em 5 partes. Após fazer a divisão, o algoritmo então irá iterar sobre cada uma das k partes divididas, de maneira que, a cada iteração, o subconjunto de dados em que a iteração se encontra, destacado como a parte vermelha na Figura 7, será considerada como a base de dados onde o algoritmo irá ser testado com base na métrica de avaliação selecionada e as demais outras partes divididas pelo K-Fold, destacado pela cor azul, serão consideradas como a base de dados onde o algoritmo será treinado. Ao fim da execução de todo o processo, para cada uma das iterações, serão obtidos resultados que indicam a assertividade com base na métrica de avaliação do algoritmo.

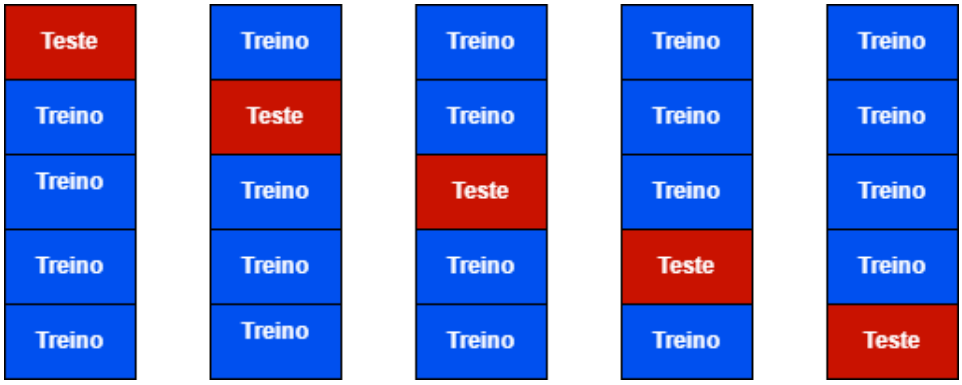


Figura 7: Exemplo de *cross validation*
Fonte: Adaptado de Santana (2020).

2.6 Técnica de padronização dos dados

Neste trabalho, utilizou-se a técnica de padronização de dados *Z-score*. De acordo com Kirkwood e Sterne (2010), o *Z-score* expressa o quão longe um valor está da média da população, ou, no caso deste trabalho, da base de dados. Esta técnica pode ser expressa matematicamente por:

$$Z - score = \frac{x - \mu}{\sigma} \tag{2.19}$$

Onde x é o ponto observado, μ e σ são a média e o desvio padrão da população, respectivamente.

2.7 Métricas de avaliação

Para a avaliação da assertividade dos algoritmos de aprendizado de máquina utilizados neste trabalho, foram usadas duas métricas: coeficiente de determinação e erro quadrático médio.

2.7.1 Coeficiente de determinação

O coeficiente de determinação ou, como é mais conhecido, R^2, é uma métrica que diz quão bem os dados preditos por um algoritmo podem ser explicados por um modelo linear. Nesta métrica, o valor de R^2 pode variar de (0, 1], onde quanto mais próximo de 1 melhor o modelo consegue predizer os dados do problema. Matematicamente, pode-se descrever a equação desta métrica por:

$$R^2 = \frac{\sum_{i=0}^{n} (\hat{y}_i - \bar{y})^2}{\sum_{i=0}^{n} (y_i - \bar{y})^2} \tag{2.20}$$

Onde $\hat{y}_i$ é o valor predito de y para a observação i, y_i é o valor da observação i, $\bar{y}$ é a média dos valores de y e n é a quantidade de observações.

2.7.2 Erro Quadrático Médio

Segundo Glen (2021), o Erro Quadrático Médio (EQM[1]) é uma métrica que busca mostrar o quão próximo um conjunto de pontos está de uma reta de regressão. Para isso, chama-se de erro a distância entre estes pontos e a linha de regressão e, em seguida, eleva-se o erro ao quadrado. Então, quanto mais próximo de 0, mais assertivo o modelo é. Matematicamente, o Erro Quadrático Médio é descrito por:

$$EQM = \frac{1}{n} \sum_{i=0}^{n} (\hat{y}_i - y_i)^2 \tag{2.21}$$

[1] Do oringinal, em Inglês, *Mean Squared Error*.

Onde $\hat{y}_i$ é o valor predito e y_i é o valor real da observação i e n é a quantidade de observações presentes no conjunto de dados.

3 Desenvolvimento

Nesta seção serão apresentadas as etapas do desenvolvimento do projeto.

3.1 Linguagem de Programação e ambiente de execução

Todo o processo de codificação usado nos experimentos deste trabalho foi feito através da linguagem de programação Python na versão 3.7.12. Python é uma linguagem de programação de alto nível, dinâmica, interpretada, modular, multiplataforma e orientada a objetos (ROVEDA, 2021). Sendo uma linguagem que está em crescente ascensão, junto com outras linguagem emergentes, como Julia, o Python apresenta um alto nível de facilidade para que novos usuários a aprendam. Estas linguagens vêm fazendo, cada vez mais, parte do cotidiano de ambientes que utilizam tópicos de aprendizado de máquina e ciência de dados.

O ambiente de execução para a elaboração desse trabalho foi o Jupyter, uma vez que trata-se de um ambiente de desenvolvimento que permite a utilização de marcadores de texto e codificação, facilitando o processo de documentação e organização do trabalho. As Figura 8, ilustra o ambiente do Jupyter.

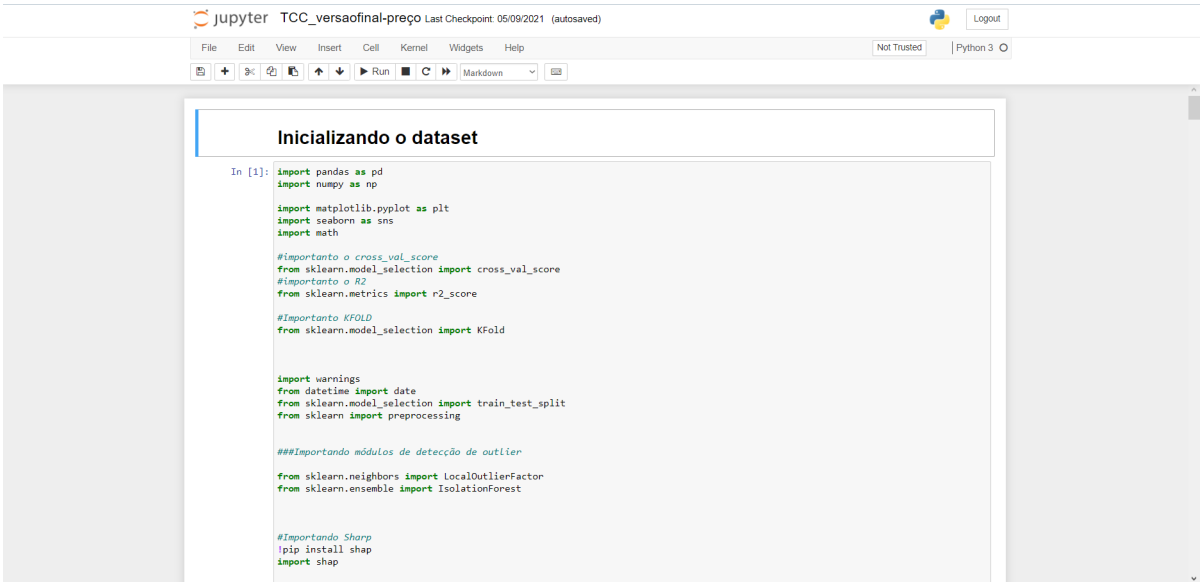


Figura 8: Ambiente de execução Jupyter
Fonte: Elaborado pelo autor.

3.2 Base de dados

Para a realização deste trabalho foi utilizada a base de dados *Brazilian E-Commerce Public Dataset by Olist*[1] disponibilizada no site Kaggle. Neste sentido, é importante ressaltar que esta base de dados contém mais de 100 mil pedidos com informações do produto, vendedor e avaliações dos compradores. Como é possível perceber, um *dataset* com tantas informações acerca dos produtos traz consigo uma grande quantidade de *features*. Por isso, a base de dados fornecida pelo ambiente foi disponibilizada em pequenos *datasets*, de maneira que houvesse um arquivo diferente para cada tipo de informação registrada. A Figura 9, ilustra as *features* presentes em cada uma das unidades de dados fornecidas pelo Kaggle.

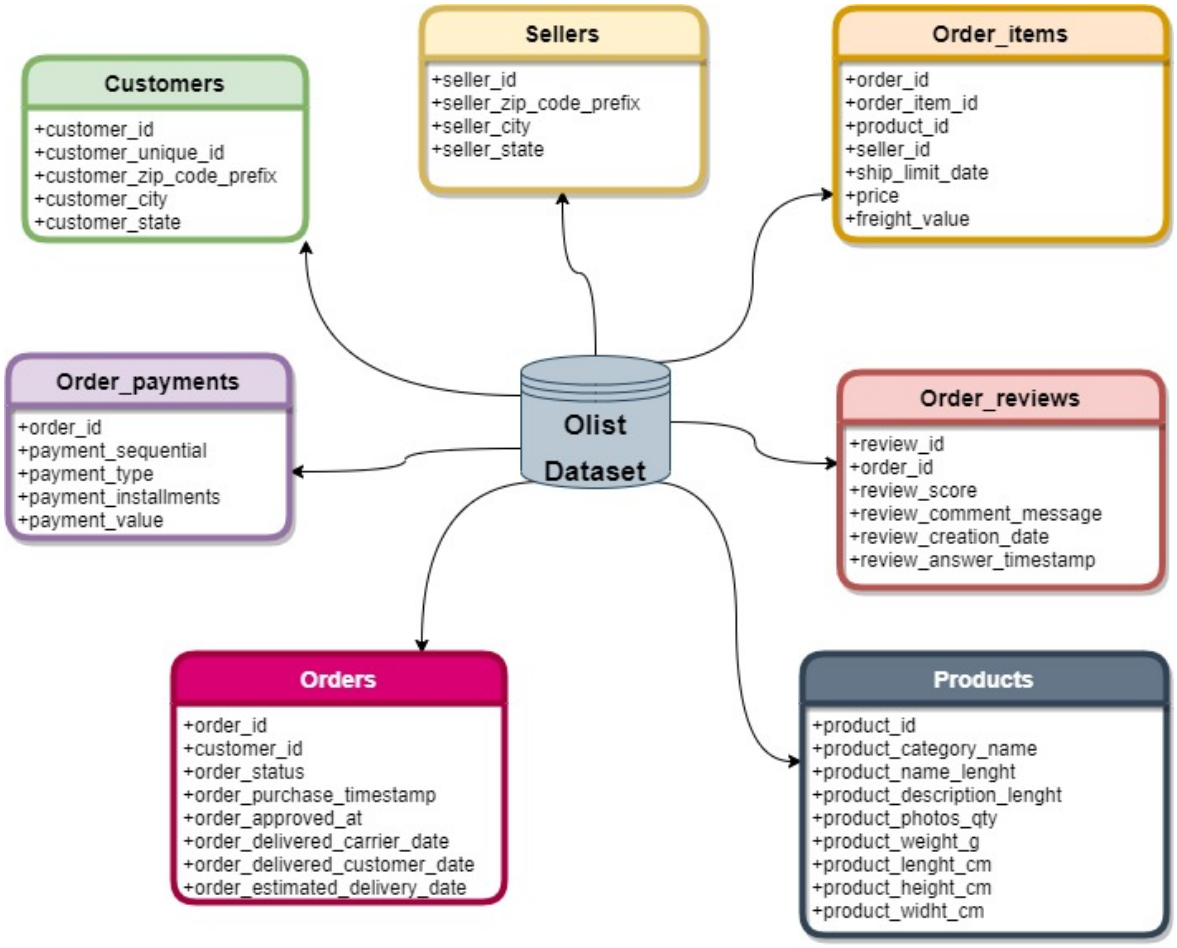


Figura 9: Base de dados Olist.
Fonte: Elaborado pelo autor

[1] Disponível em: https://www.kaggle.com/olistbr/brazilian-ecommerce

3.3 Tratamento dos dados

3.3.1 Agrupamento dos *datasets*

Uma vez que todos estes arquivos encontravam-se separados e que o interesse deste trabalho é obter a maior quantidade de informações possíveis sobre os itens vendidos no *e-commerce*, foi realizada a junção entre cada um dos *datasets* utilizando como chave de identificação as *features* que eram comuns a dois ou mais *datasets*. A Figura 10 mostra como foi feito o processo de concatenação de cada uma das bases de dados de forma a chegar à base de dados final desejada.

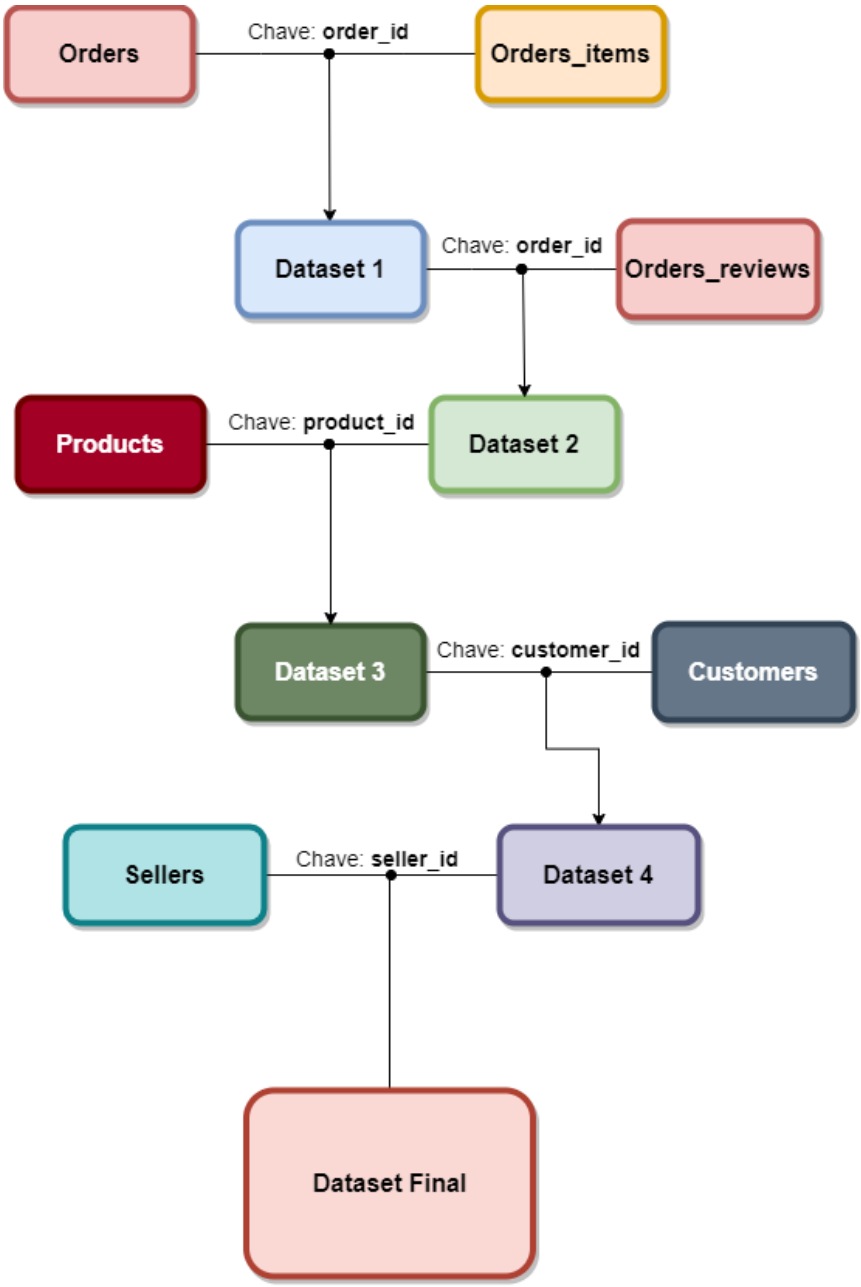


Figura 10: Agrupamento dos *datasets*.
Fonte: Elaborado pelo autor

Com todo o processo de concatenação dos dados, foi obtido uma base de dados final que foi utilizada para a realização deste trabalho. Esta base de dados é composta por 119.151 observações e 38 *features* que caracterizavam cada uma delas. Para que seja possível um melhor entendimento sobre o *dataset* final, a Figura 11 ilustra cada um dos seus atributos.

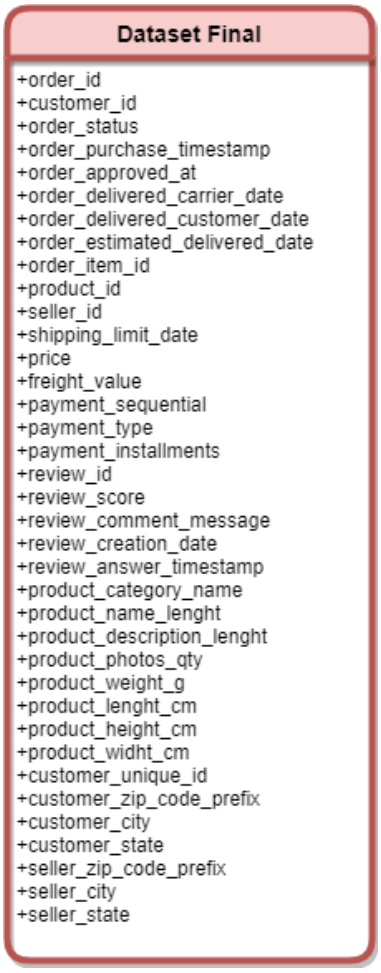


Figura 11: *Features* do *dataset* final.
Fonte: Elaborado pelo autor.

3.3.2 Manuseio dos dados

O manuseio dos dados é um dos processos mais importantes na execução de atividades relacionadas ao aprendizado de máquina. Segundo Ramos (2020), este processo consiste em transformar e manipular os dados de um *dataset* de forma a trazê-los para um contexto mais conveniente de acordo com o tipo de aplicação desejada. Dessa forma, nesta subseção serão apresentados os tratamentos que foram aplicados aos dados deste trabalho.

3.3.2.1 Dados nulos

Antes de se treinar algum algoritmo de aprendizado de máquina é necessário que seja feita uma limpeza dos dados nulos presentes na base de dados, uma vez que estes dados não serão úteis no processo de aprendizado. A biblioteca Pandas[2], amplamente usada para a manipulação de *datasets* em Python, dispõe de uma função que retira todas as linhas da base de dados que possui alguma de suas colunas com valor nulo.

Ao analisar a Figura 12, para cada uma das colunas do *dataset*, existiam diversos dados nulos. Neste ponto, antes de se excluir todas as linhas que possuiam algum tipo de dado nulo, é importante que *features* que não serão usadas no trabalho sejam apagadas para que informações que possam ser importantes para o aprendizado não sejam perdidas.

```
In [267]: df_final = df
          df_final.isna().sum()

Out[267]: order_id                          0
          customer_id                       0
          order_status                      0
          order_purchase_timestamp          0
          order_approved_at               177
          order_delivered_carrier_date   2086
          order_delivered_customer_date  3421
          order_estimated_delivery_date     0
          order_item_id                   833
          product_id                      833
          seller_id                       833
          shipping_limit_date             833
          price                           833
          freight_value                   833
          payment_sequential                3
          payment_type                      3
          payment_installments              3
          payment_value                     3
          review_id                         0
          review_score                      0
          review_comment_message        67901
          review_creation_date              0
          review_answer_timestamp           0
          product_category_name          2542
          product_name_lenght            2542
          product_description_lenght     2542
          product_photos_qty             2542
          product_weight_g                853
          product_length_cm               853
          product_height_cm               853
          product_width_cm                853
          customer_unique_id                0
          customer_zip_code_prefix          0
          customer_city                     0
          customer_state                    0
          seller_zip_code_prefix          833
          seller_city                     833
          seller_state                    833
          dtype: int64
```

Figura 12: Dados nulos presentes no *dataset* final
Fonte: Autoria própria.

Quando analisado, o *dataset* apresentava diversas *features* que se mostravam como

[2] Pacote disponível em: https://pandas.pydata.org/

códigos criptografados para que a identidade de itens, clientes e outras coisas pudessem sem preservados quando divulgados e que, para o âmbito desse trabalho, não teria utilidade. Então, colunas com *features* como: order_item_id, customer_unique_id e review_id foram apagadas utilizando a função *drop()*.

Por outro lado, comentários deixados por compradores que adquiriram produtos estavam disponíveis na base de dados e muitos deles eram comentários nulos, como pode ser visto na Figura 12. Neste sentido, esta coluna também foi apagada.

3.3.2.2 Dados duplicados

Decorrente da grande base de dados disponibilizada no ambiente Kaggle, não é surpresa que existam dados duplicados presentes. Neste *dataset* em específico, existiam dados de compras que foram realizadas pelo mesmo comprador, através de um mesmo vendedor e com o mesmo produto. Para excluir estes dados, foi utilizada a função *drop_duplicates()*, definindo os parâmetros da função que compara cada uma das linhas que possuem valores iguais para as colunas : order_item_id, review_comment_message, customer_unique_id, review_id e payment_sequential.

3.3.2.3 Categorias redundantes

A base de dados utilizada neste trabalho nominava cada tipo de item em uma dada categoria. No entanto, muitas delas eram redundantes, pois diziam respeito ao mesmo segmento de objetos que, ao fim da construção da base de dados, segmentava em diversas categorias cada um dos produtos. Para que o algoritmo de aprendizado de máquina pudesse ter um melhor desempenho, todas as categorias que diziam respeito à um mesmo segmento de objetos, foram agrupadas de maneira a preservar o máximo possível cada singularidade de cada categoria.

A Figura 13 mostra os agrupamentos feitos para diminuir a segmentação presente na base de dados. Em negrito, é destacado o nome da nova classificação dada pelo agrupamento e cada tópico dentro do quadrado representa uma *feature* que foi utilizada para sua construção.

moveis
• moveis_decoracao
• moveis_escritorio
• moveis_cozinha_area_de_servico_jantar_e_jardim
• moveis_sala
• moveis_colchao_e_estofado
• moveis_quarto
beleza_saude
• perfumaria
• beleza_saude
• fraldas_higiene
comercio
• industria_comercio_e_negocios
• agro_industria_e_comercio
construcao
• construcao_ferramentas_construcao
• casa_construcao
• construcao_ferramentas_iluminacao
• construcao_ferramentas_jardim
• construcao_ferramentas_seguranca
• construcao_ferramentas_ferramentas
bebidas_e_alimentos
• alimentos
• alimentos_bebidas
• bebidas

casa
• utilidades_domesticas
• casa_conforto
• la_cuisine
• casa_conforto_2
• portateis_casa_forno_e_cafe
• cama_mesa_banho
• portateis_cozinha_e_preparadores_de_alimentos
• eletrodomesticos
informatica_e_telefonia
• informatica_acessorios
• telefonia
• tablets_impressao_imagem
• pcs
• telefonia_fixa
• pc_gamer
• dvds_blu_ray
• cds_dvds_musicais
• audio
fashion
• fashion_underwear_e_moda_praia
• fashion_bolsas_e_acessorios
• fashion_calcados
• fashion_roupa_masculina
• fashion_esporte
• fashion_roupa_infanto_juvenil
• fashion_roupa_feminina

Figura 13: Composição das categorias após o reagrupamento.
Fonte: Elaborado pelo autor.

3.3.2.4 Variáveis Categóricas

O tratamento dos dados categóricos presentes na base de dados foi feita utilizado o procedimento chamado de *one-hot-encoding*. Este procedimento consistiu em transformar cada variável categórica em uma coluna e atribuir valores binários à ela, de forma que o valor de 1 quer dizer que uma determinada observação possui a característica daquela determinada variável categórica. Neste sentido, esta técnica foi aplicada às *features* : *payment_type* e categoria

3.3.2.5 Normalização

Antes da aplicação dos algoritmos de detecção de *outlier* e também do algoritmo de aprendizado de máquina, todos os dados foram normalizados. Este procedimento de

normalização justifica-se pela grande diferença de escala entre as grandezas de cada uma dos atributos da base dados, de tal forma que, no momento do treinamento do modelo de aprendizado de máquina, não haja nenhuma interferência causada por essa disparidade de valores. Neste trabalho, foi utilizada a técnica de normalização Z-score, com o auxilio da função estatística *stats.zscore* da bibilioteca *Scipy* [3].

3.4 Separação dos dados e treinamento do algoritmo

A base de dados foi inicialmente dividida em 70% das observações para um conjunto de treino e os outros 30% para um conjunto de teste.

3.4.1 Configuração do experimento

Os algoritmos de detecção de *outlier* utilizados neste trabalho possuem um parâmetro denominado de contaminação. Este parâmetro diz respeito à porcentagem de observações presentes em um determinado conjunto de dados que devem ser consideradas como *outlier*. Nesta linha de raciocínio e pensando que não há, nos algoritmos, métodos implementados que calculem diferentes porcentagens de contaminação de um conjunto de dados e selecione a melhor, para cada um dos algoritmos, variou-se a porcentagem de contaminação entre 2%, 5% e 10%.

Durante a execução deste trabalho, optou-se pela metodologia de se retirar os *outliers* percebidos por um determinado algoritmo de detecção antes da etapa de treinamento da Floresta Aleatória. Além disso, a Floresta Aleatória foi utilizada tanto nos casos onde era associada a algoritmos de detecção de *outlier*, como também no caso onde foi criado o *baseline*, explicado posteriormente neste trabalho.

Sobre o conjunto de treino, aplicou-se a técnica de validação cruzada *K-Fold*, com um K igual a 30. Desta forma, cada um dos algoritmos de detecção de *outlier*, sendo eles: LOF [4], HDBSCAN [5], KDEOS [6], iForest [7], COPOD [8], ao serem associados as taxas de contaminação, a técnica de validação cruzada e a técnica de aprendizado de máquina de Floresta Aleatória [9], geraram diferentes resultados com base nas métricas de avaliação R^2 e de erro quadrático médio.

Para que se pudesse ter uma percepção sobre como os algoritmos de detecção de *outlier* impactam sobre a assertividade em um processo de aprendizado de máquina,

[3] Disponível em: Scipy.Stats
[4] Disponível em: Local Outlier Factor
[5] Disponível em: Hierarchical Density-based Spatial Clustering of Applications with Noise
[6] Disponível em: Local Outlier Factor
[7] Disponível em: Isolation Forest
[8] Disponível em: Copula Based Outlier Detector
[9] Disponível em: Random Forest

aplicou-se somente a técnica de validação cruzada *K-Fold*, com um valor de K igual a 30, à Floresta Aleatória, sem nenhum outro algoritmo de detecção de *outlier*, criando uma *baseline* para o problema. Como neste trabalho, o conjunto de dados foi separado em conjunto de treino e teste, criou-se duas *baselines*, uma referente ao conjunto de treino e outra referente ao conjunto de teste.

Para cada um dos resultados obtidos neste trabalho, decorrentes das aplicações dos algoritmos, das taxas de contaminação e também da *baseline*, houveram 30 amostras de R^2 e de erro quadrático médio, graças à técnica de validação cruzada. Visando identificar o melhor resultado obtido, retirou-se a média e a variância de cada conjunto-resposta das métricas de avaliação, de maneira a classificar a melhor solução que possuiria a maior média e a menor variância para a métrica de avaliação R^2 e, no caso do EQM, a menor média e a menor variância.

A melhor solução obtida no conjunto de treino foi então aplicada ao conjunto de testes para que se pudesse validar a assertividade do modelo. Neste momento, aplicou-se novamente uma Floresta Aleatória associada ao algoritmo de detecção de *outlier* que trouxe melhor resultados ao modelo, utilizou-se as 30 amostras provenientes de cada uma das métricas de avaliação para o cálculo da média e da variância. Por fim, buscou-se investigar quais foram as características que mais influenciaram no algoritmo de aprendizado de máquina e seus respectivos pesos para esse processo.

4 Resultados

Neste capítulo, serão apresentados os resultados obtidos através da metodologia descrita no Capítulo 3.

4.1 Conjunto de treino

Como mencionado no Capítulo de desenvolvimento, o conjunto de treino utilizado neste trabalho teve por finalidade identificar qual o algoritmo de detecção de *outliers* que, ao ser associado ao algoritmo de aprendizado de máquina, gerou a melhora mais significativa de assertividade. Nesta seção, irão ser discutidos cada um dos resultados obtidos.

Analisando-se um conjunto amostral de 30 resultados obtidos através da aplicação do algoritmo de detecção de *outlier* seguidos pelo treinamento e validação do algoritmo de aprendizado, tornou-se interessante aplicar técnicas que mostrem visualmente quais são as características de cada um dos conjunto-resposta.

Nas Figuras 14 e 15, nas colunas à esquerda, tem-se um *boxplot* com o resultado do algoritmo baseado na métrica R^2 e, nas colunas à direita, o resultado baseado na métrica EQM. Graças as características do gráfico *boxplot* é possível perceber resultados interessantes sobre o conjunto-resposta. Neste sentido, torna-se interessante analisar a faixa dos resultados obtidos, destacando-se o algoritmo iForest, mostrado pela Figura 14c, que apresentou resultados de R^2 que vão desde, aproximadamente, 0.82 até 0.95. Esta característica mostra-se por meio dos eixos que perfuram a ilustração dos quartis da caixa e que, quanto mais alongados forem, maior variabilidade o conjunto-resposta possuiu.

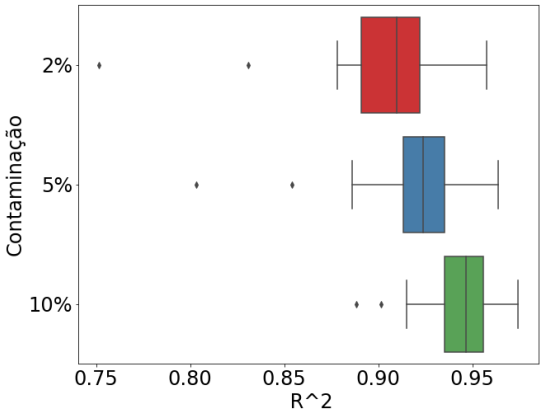
Neste caso em específico, por meio da Figura 14c é possível perceber que há uma variação do R^2 de cerca de 0.14. Este resultado representado por uma grande variância, pode ser explicada por diversos motivos. Aqui, destaca-se dois: o primeiro trata-se da maneira na qual árvores do *Isolation Forest* são geradas, uma vez que os ramos são criados a partir de uma estrutura de decisão baseada em valores de corte que, assim como mostrado na Figura 2, geram cortes horizontais e verticais que servem para isolar determinadas observações. Estes cortes podem acabar gerando regiões enviesadas que assinalam observações de maneira errônea.

O segundo motivo, de caráter mais generalista e que atinge não somente o *iForest* mas boa parte dos algoritmos de detecção de *outlier*, trata-se da taxa de contaminação do

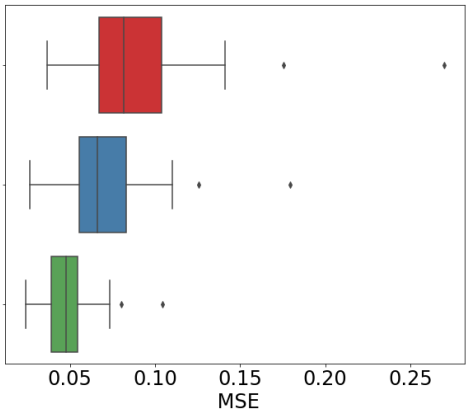
conjunto de dados, isto é, necessita-se que sejam fornecidos ao algoritmo a porcentagem de *outliers* existentes no conjunto de dados do problema. Desta forma, com base no método de procura do algoritmo e em como ele enxerga *outliers*, à medida com que se varia a taxa de contaminação, o algoritmo passa a tentar isolar observações que não seriam consideradas como *outliers*, fazendo com que o desempenho do modelo seja degradado.

Em contrapartida, é possível perceber pelas Figuras 14a e 14b que, quando associado ao algoritmo LOF, o modelo conseguiu não só dispor resultados contidos em uma pequena faixa de valores como também uma maior assertividade. Outro ponto interessante a ser levantado é que, a medida com que se aumenta a contaminação do conjunto de dados, o modelo consegue obter resultados melhores. Estes resultados são mostrados pela Figura 14a quando o *boxplot* é uma figura mais achatada e está próximo a 1 e, pela Figura 14b que, da mesma forma, apresenta uma figura mais achatada e valores próximos a 0.

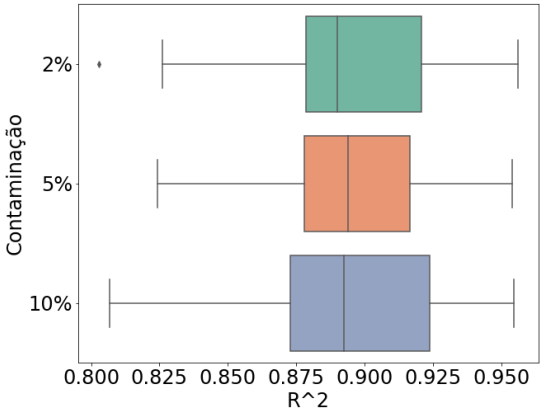
O motivo mais provável pelo qual o algoritmo conseguiu fazer com que o modelo obtivesse melhores resultados pode ser justificado pela vantagem na qual o LOF consegue identificar *outliers* locais mesmo que, apesar de próximas a *clusters*, determinadas observações possam ser identificadas e classificadas como *outliers*. O aumento da taxa de contaminação associado a melhora dos resultados induz a uma provável presença de observações que são consideradas como *outliers* locais, já que o algoritmo LOF é um algoritmo de busca local, que ao serem retiradas do conjunto de dados, fazem com que o modelo seja mais assertivo. Por se tratar de um problema com múltiplas dimensões, é difícil visualizar *clusters* formados pelas observações.



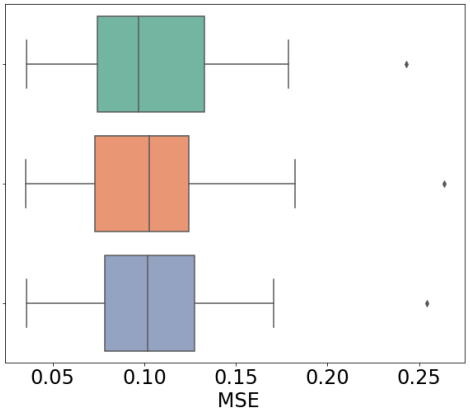
(a) LOF: R^2



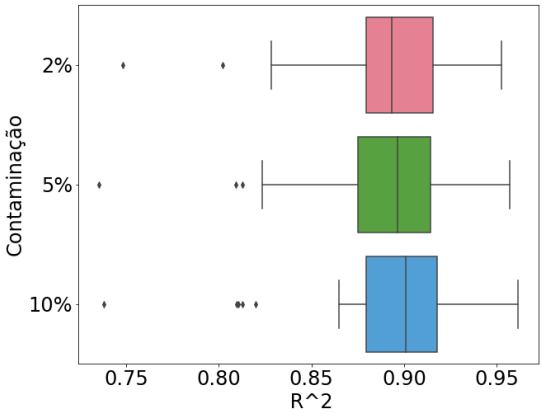
(b) LOF: EQM



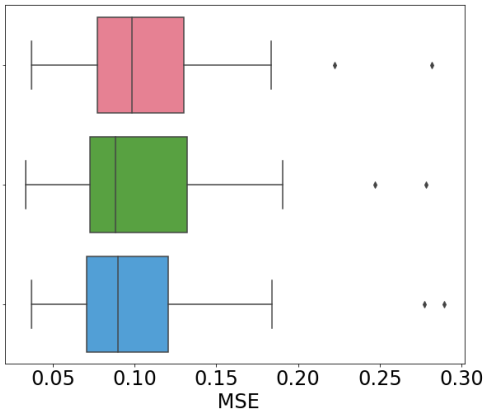
(c) iForest: R^2



(d) iForest: EQM



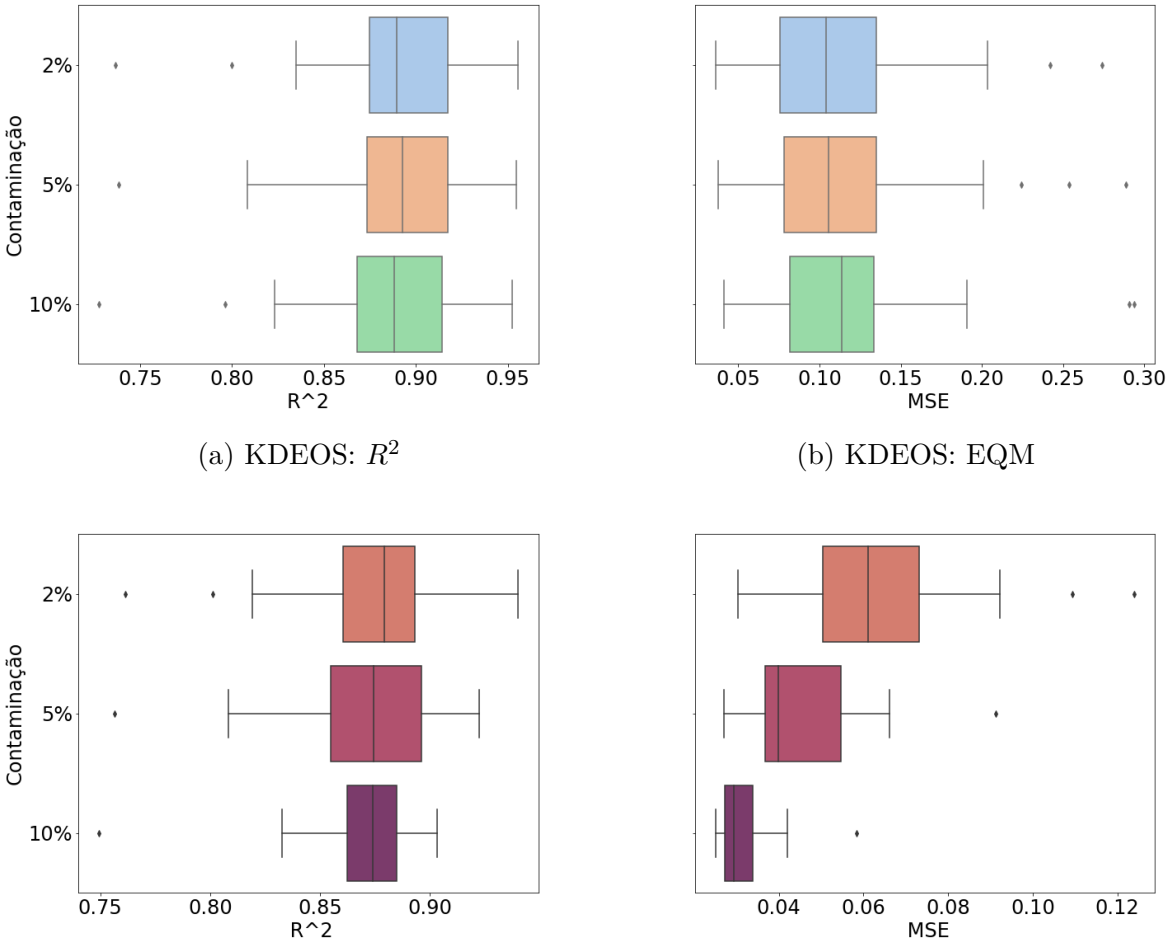
(e) HDBSCAN: R^2



(f) HDBSCAN: EQM

Figura 14: *Box plot* obtido dos 30 resultados gerados pelo *cross validaton*, para cada uma das métricas de avaliação e os algoritmos LOF, iForest e HDBSCAN para detecção de *outliers*

Fonte: Elaborado pelo autor.



(a) KDEOS: R^2

(b) KDEOS: EQM

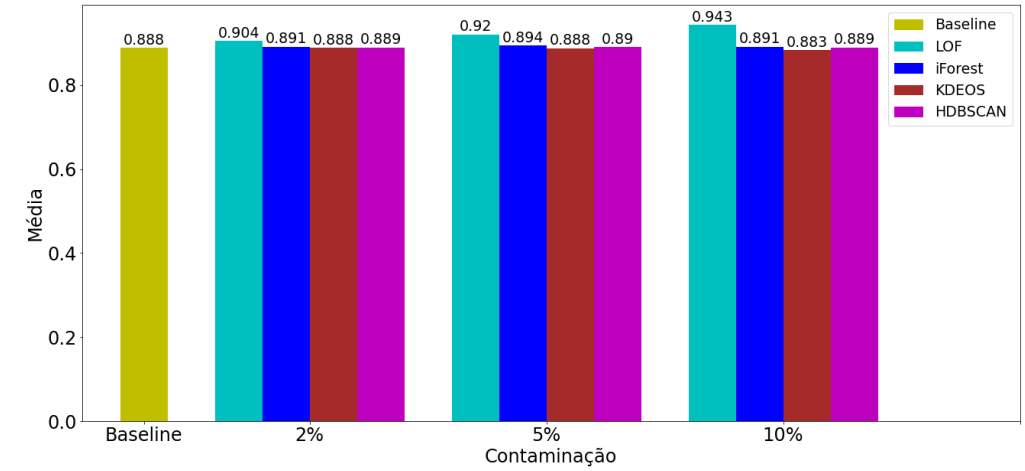
(c) COPOD: R^2

(d) COPOD: EQM

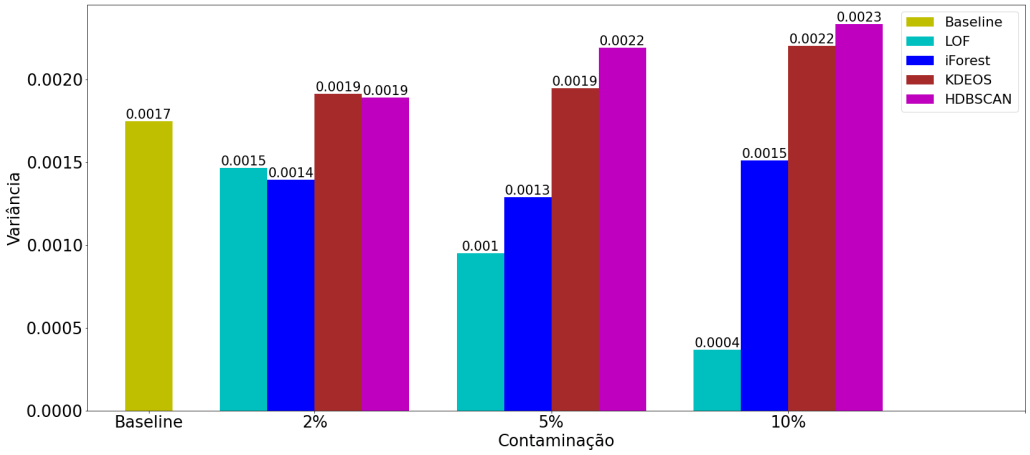
Figura 15: *Box plot* obtido dos 30 resultados gerados pelo *cross validaton*, para cada uma das métricas de avaliação e os algoritmos KDEOS e COPOD
Fonte: Elaborado pelo autor.

Para fins comparativos e analíticos, escolheu-se plotar somente os resultados médios de R^2 do modelo gerados pela *Random Forest*, uma vez que ambas as métricas de avaliação tendem a apresentar resultados semelhantes. Neste sentindo, foi feito um gráfico de barras demonstrado na Figura 16. Na Figura 16a, estão dispostos os valores referentes a média de R^2 para cada dos resultados dos modelos quando associado aos algoritmos de detecção de *outlier*. É possível perceber que, comparado ao *baseline*, o modelo quando associado ao LOF consegue uma melhora na média de resultados obtidos pelo modelo, principalmente quando se usa um índice de contaminação de 10% . A Figura 16b demonstra que a variabilidade dos resultados do modelo também sofre uma melhora, isto é, pequenos valores de variância corroboram com a tese de que os valores de R^2 para os diferentes conjuntos de treino e teste criados pelo *K-Fold* não sofrem grandes variações. Por outro lado, é possível perceber que quando se olha para o algoritmo HDBSCAN, principalmente com 10% de contaminação, existe uma grande variabilidade entre os resultados do modelo, que também pode ser vista

nas Figuras 14e e 14f, ilustrada pelos pontos assinalados fora da área de abrangência das caixas e que acabam gerando esse resultado de variabilidade.



(a) Média de valores obtidos em R^2



(b) Variância dos valores de R^2

Figura 16: *Gráfico de barras feito por meio dos 30 resultados de R^2 gerados pelo cross validaton e cada um dos algoritmos de detecção outlier*
Fonte: Elaborada pelo autor.

O possível motivo pelo qual o algoritmo HDBSCAN não conseguiu trazer melhores resultados pode se resumir ao parâmetro *minimum cluster size*. Neste trabalho, adotou-se o valor padrão da biblioteca de 5, permitindo com que existissem *clusters* extremamente pequenos que, durante a etapa na qual o algoritmo busca conectar dois *clusters* para que se forme uma hierarquia, fizeram com que não fossem atendidos os requisitos mínimos, desta forma, sendo considerados como *outliers*. Outro parâmetro, chamado de *min_samples*, um pouco menos intuitivo, mas que afeta significativamente o resultado do algoritmo, procura determinar o número mínimo de observações presentes na vizinhança para que um

determinado ponto possa ser definido como um ponto central de um *cluster*. Em outras palavras, quanto maior o valor deste parâmetro, mais observações serão considerada como *outliers* e cada um dos *clusters* serão consequentemente mais densos. Como é definido por padrão da bibioteca, não existe um valor mínimo de observações que precisam estar ao redor de um ponto para que ele seja considerado como central, o que pode levar o algoritmo a criar *clusters* pouco densos e que contenham *outliers*.

Outro ponto interessante a ser analisado diz respeito a importância a qual o algoritmo está dando para cada um dos atributos presentes na base de dados para a predição dos preços. A Figura 17 mostra, por meio de uma escala de cores, a importância de cada atributo, de forma que é possível identificar que existe uma grande importância para os atributos *freight_value* e *payment_installments* bem como também para atributos que foram criados a partir da técnica de *one-hot-encoding*, mostrando que um bom tratamento dos dados impacta positivamente no resultado do modelo.

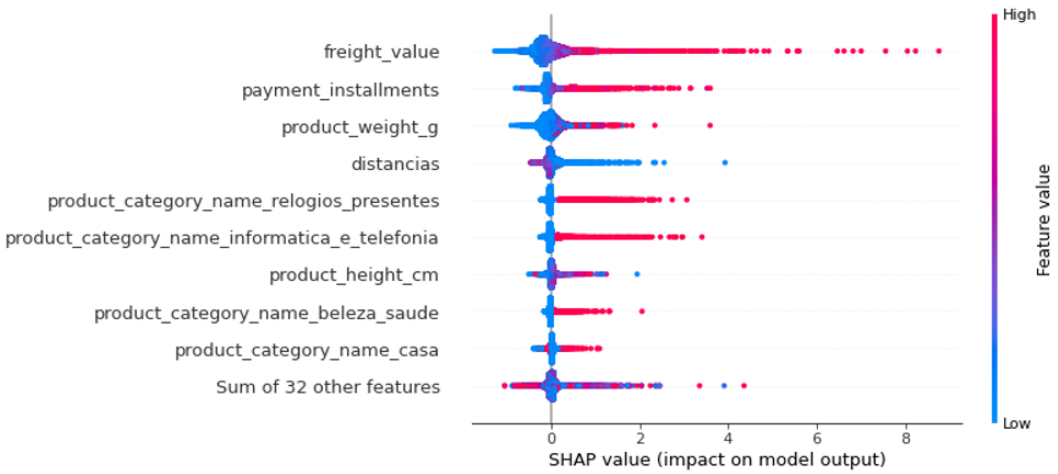


Figura 17: Importância dos atributos.
Fonte: Elaborado pelo autor.

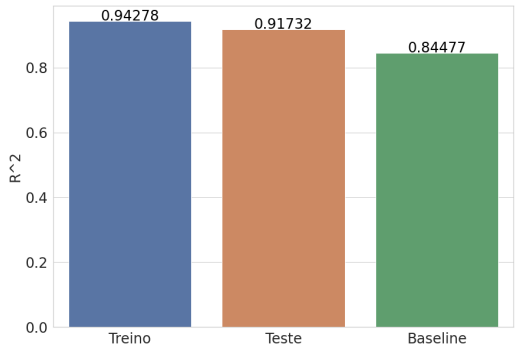
4.1.1 Escolha do algoritmo de detecção de *outliers*

Com base no que foi discutido nesta seção, procurou-se selecionar o algoritmo de detecção de *outlier* que aparentou trazer não só uma melhora nos resultados mas como também agregar menor variabilidade de resultados do modelo. Desta forma e conforme os argumentos expostos, o algoritmo LOF foi selecionado para ser aplicado no conjunto de testes do problema.

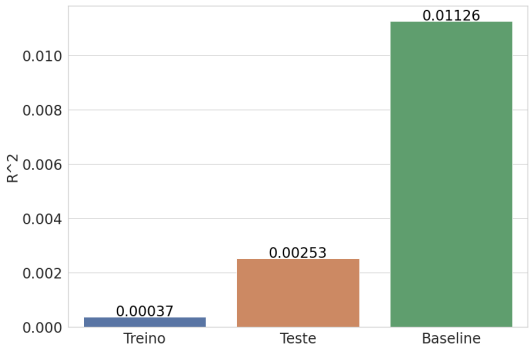
4.2 Conjunto de testes

Para validar os resultados obtidos no conjunto de testes e confirmar que não houve um *overfit* do modelo, os 30% restantes das observações do *dataset* foram utilizados para validar o modelo.

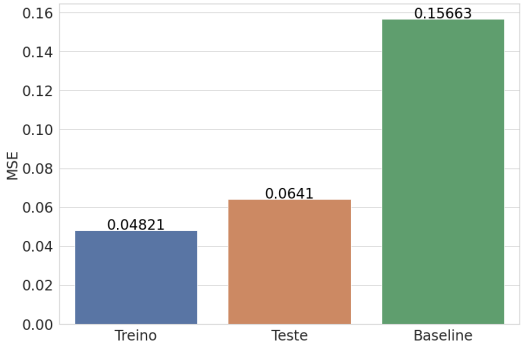
A Figura 18 mostra os resultados obtidos no conjunto de teste de forma a comparar os resultados nos conjuntos de treino e também a um *baseline*. Como pode ser observado, apesar de que quando comparados aos resultados do conjunto de treino, o conjunto de teste apresente uma piora nos resultados, há uma melhora significativa nos valores das métricas de avaliação quando se comparados à um modelo em que não foi aplicado nenhum algoritmo de detecção de *outlier*. Outro ponto importante a se destacar é que a variância dos resultados obtidos pelo modelo ainda se apresenta como valores menores quando se comparado ao *baseline*. Estes resultados obtidos no conjunto de teste mostram que não houve um *overfit* sobre o modelo e que a inteligência artificial está conseguindo generalizar de forma satisfatória os preços dos produtos vendidos no *e-commerce*.



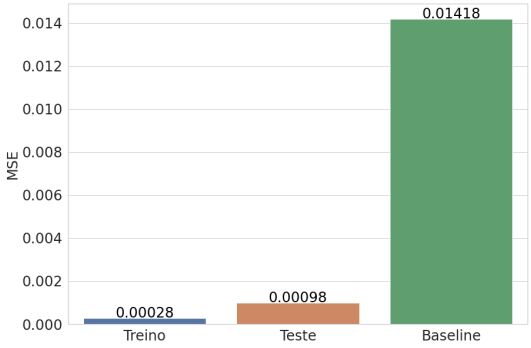
(a) Média de valores obtidos em R^2 para o conjunto de testes



(b) Variância dos valores de R^2 para o conjunto de testes



(c) Média dos valores de EQM para o conjunto de testes



(d) Variância valores de EQM para o conjunto de testes

Figura 18: *G*ráfico de barras feito sobre o conjunto de testes do *dataset* com base nas métricas de avaliação

Fonte: Elaborada pelo autor.

5 Conclusão

A proposta deste trabalho de conclusão de curso consistiu na criação de um modelo de uma ferramenta que auxilia na precificação de mercadorias vendidas via e-commerce utilizando um algoritmo de aprendizado de máquina.

Ao fim deste trabalho, foi possível concluir que métodos de detecção de *outlier* conseguem trazer grandes benefícios para modelos de aprendizado de máquina, por vezes aumentando a sua assertividade bem como diminuindo a variabilidade dos dos seus resultados. No entanto, torna-se interessante levantar que há uma grande necessidade de que sejam feitos testes, aplicando-se diretamente algoritmos de detecção de *outlier* sobre o conjunto de dados e escolhendo-se aquele que melhor identifica a distribuição das observações e melhora o resultado do modelo.

Além disso, o modelo obtido para precificação de itens em *e-commerce*, mostrou-se como uma poderosa ferramenta para que vendedores consigam ingressar neste mercado que conecta tecnologia e empreendedorismo em um só lugar. Desta forma, através dessa ferramenta, vendedores conseguirão se resguardar contra erros de precificação e conseguirão elevar o seu negócio a um novo nível.

Por fim, este trabalho proporcionou ao discente um conhecimento não só sobre o funcionamento teórico de algoritmos de detecção de *outlier*, como também proporcionou a experiência de sua aplicação e seus benefícios para um modelo de aprendizado de máquina.

5.1 Sugestões para trabalho futuros

Como sugestão para trabalhos futuros, pode-se alterar os hiperparâmetros da Floresta Aleatória e realizar um comparativo entre eles, bem como aplicar algoritmos que se utilizam de redes neurais para e verificar se há uma melhora significativa no processo de precificação ou, também, adicionar imagens ao conjunto de dados e avaliar características individuais dos itens, como marca, cor, etc

Referências

AMPADU, H. *Random Forests Understanding*. 2021. Disponível em: <https://ai-pool-.com/a/s/random-forests-understanding>. Acesso em: 03th October 2021. Citado 2 vezes nas páginas 8 e 31.

BARNETT, V.; LEWIS, T. Outliers in statistical data. *Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics*, 1984. 1984. Citado na página 17.

BEHGOUNIA, F.; ZOHURI, B. Machine learning driven an e-commerce. *International Journal of Computer Science and Information Security (IJCSIS)*, 2020. v. 18, n. 10, 2020. Citado na página 13.

BERBA, P. *A gentle introduction to HDBSCAN and density-based clustering*. 2020. Disponível em: <https://towardsdatascience.com/a-gentle-introduction-to-hdbscan-and-density-based-clustering-5fd79329c1e8>. Acesso em: 23th September 2021. Citado na página 26.

BREIMAN, L. Random forests. *Machine learning*, 2001. Springer, v. 45, n. 1, p. 5–32, 2001. Citado na página 31.

BREIMAN, L. et al. *Classification and regression trees*. [S.l.]: Routledge, 1984. Citado na página 28.

BREUNIG, M. M. et al. Lof: identifying density-based local outliers. In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. [S.l.: s.n.], 2000. p. 93–104. Citado na página 18.

CAMPELLO, R. J. et al. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2015. ACM New York, NY, USA, v. 10, n. 1, p. 1–51, 2015. Citado na página 28.

CAMPOS, G. O. *Estudo, avaliação e comparação de técnicas de detecção não supervisionada de outliers*. Tese (Doutorado) — Universidade de São Paulo, 2015. Citado na página 17.

CHIAVENATO, I. *Introdução à Teoria Geral da Administração*. [S.l.]: Manole, 2019. 621p. Citado na página 12.

CONVERSION. *Relatório E-commerce no Brasil*. 2021. Disponível em: <https:-//www.conversion.com.br/wp-content/uploads/2021/05/Maio_21-Abril-Relatorio-E-commerce-no-Brasil.pdf>. Acesso em: 30th June 2021. Citado na página 12.

CORTIVO, Z. D. *Modelos probabilísticos*. [S.l.]: InterSaberes, 2019. 202p. Citado na página 22.

DOTTO, D. M. R.; MOYANO, C. A. M. et al. Estudos organizacionais: desafios contemporâneos: volume ii. 2008. EDUNISC, 2008. Citado na página 13.

FATHALLA, A. et al. Deep end-to-end learning for price prediction of second-hand items. *Knowledge and Information Systems*, 2020. Springer, v. 62, n. 12, p. 4541–4568, 2020. Citado na página 16.

GLEN, S. *Mean Squared Error: Definition and Example*. 2021. Disponível em: <https://www.statisticshowto.com/probability-and-statistics/statistics-definitions-/mean-squared-error/>. Acesso em: 5th October 2021. Citado na página 33.

HAWKINS, D. M. *Identification of outliers*. [S.l.]: Springer, 1980. Citado 2 vezes nas páginas 13 e 17.

KIRKWOOD, B. R.; STERNE, J. A. *Essential medical statistics*. [S.l.]: John Wiley & Sons, 2010. Citado na página 33.

LI, Z. et al. Copula-based outlier detection. In: *Proceedings of the 2020 IEEE International Conference on Data Mining (ICDM), Sorrento, Italy*. [S.l.: s.n.], 2020. p. 17–20. Citado na página 22.

LIU, F. T.; TING, K. M.; ZHOU, Z.-H. Isolation forest. In: IEEE. *2008 eighth ieee international conference on data mining*. [S.l.], 2008. p. 413–422. Citado 2 vezes nas páginas 13 e 20.

MORAIS, S. *Árvore de extensão mínima (Minimal Spanning Tree) com ações da Ibovespa*. 2018. Disponível em: <https://medium.com/@samuelmorais_73757/árvore-de-extensão-mínima-minimal-spanning-tree-com-ações-da-ibovespa-60df7496d678>. Acesso em: 23th September 2021. Citado na página 25.

MOURA, S. W. de O. et al. O crescimento do ecommmerce entre brasil e china. 2015. 2015. Citado na página 12.

PARZEN, E. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 1962. JSTOR, v. 33, n. 3, p. 1065–1076, 1962. Citado na página 24.

RAMOS, E. *E-commerce*. [S.l.]: Editora FGV, 2015. Citado na página 12.

RAMOS, R. *Data Wrangling: Preparando os Dados*. 2020. Disponível em: <https:/-/oestatistico.com.br/data-wrangling-dados/>. Acesso em: 11th October 2021. Citado na página 38.

ROVEDA, U. *O que é Python, para que serve e por que aprender?* 2021. Disponível em: <https://kenzie.com.br/blog/o-que-e-python/>. Acesso em: 03th October 2021. Citado na página 35.

SANTANA, R. *Validação Cruzada: Aprenda de forma simples como usar essa técnica*. 2020. Disponível em: <https://minerandodados.com.br/validacao-cruzada-aprenda-de-forma-simples-como-usar-essa-tecnica/>. Acesso em: 4th October 2021. Citado 2 vezes nas páginas 8 e 32.

SAUL, N. *How HDBSCAN Works*. 2017. Disponível em: <https://github.com/scikit-learn-contrib/hdbscan/blob/master/docs/how_hdbscan_works.rst>. Acesso em: 25th September 2021. Citado na página 26.

SAUL, N. *Local Outlier Factor (LOF) — Algorithm for outlier identification*. 2020. Disponível em: <https://towardsdatascience.com/local-outlier-factor-lof-algorithm-for-outlier-identification-8efb887d9843>. Acesso em: 01th October 2021. Citado na página 19.

SCHUBERT, E.; ZIMEK, A.; KRIEGEL, H.-P. Generalized outlier detection with flexible kernel density estimates. In: SIAM. *Proceedings of the 2014 SIAM International Conference on Data Mining*. [S.l.], 2014. p. 542–550. Citado na página 24.

SCOTT, G. L.; LONGUET-HIGGINS, H. C. Feature grouping by'relocalisation'of eigenvectors of the proximity matrix. In: CITESEER. *BMVC*. [S.l.], 1990. p. 1–6. Citado na página 24.

SIEGEL, S.; JR, N. J. C. *Estatística não-paramétrica para ciências do comportamento*. [S.l.]: Artmed Editora, 1975. Citado na página 23.

SOUZA, L. *O que é Precificação de Produtos? Entenda todo o processo para chegar no preço e lucro ideal*. 2019. Disponível em: <https://conteudo.precocerto.co/precificacao/>. Acesso em: 7th September 2021. Citado na página 14.

TERRELL, G. R.; SCOTT, D. W. Variable kernel density estimation. *The Annals of Statistics*, 1992. JSTOR, p. 1236–1265, 1992. Citado na página 24.

WANDERLEY; BARBOSA, M. F. *Estudos em Estimação de Densidade por Kernel: Métodos de Seleção de Características e Estimação do*. Tese (Doutorado) — UNIVERSIDADE FEDERAL DE MINAS GERAIS, 2013. Citado na página 24.

WHAPOLE, R. H. R. E.; YE, K. *Probabilidade Estatística: para engenharia e ciências - 8ª edição*. [S.l.]: Editora Pearson, 2009. 494p. Citado na página 22.

XUEFENG, L. et al. Predicting the final prices of online auction items. *Expert Systems with Applications*, 2006. Elsevier, v. 31, n. 3, p. 542–550, 2006. Citado na página 16.