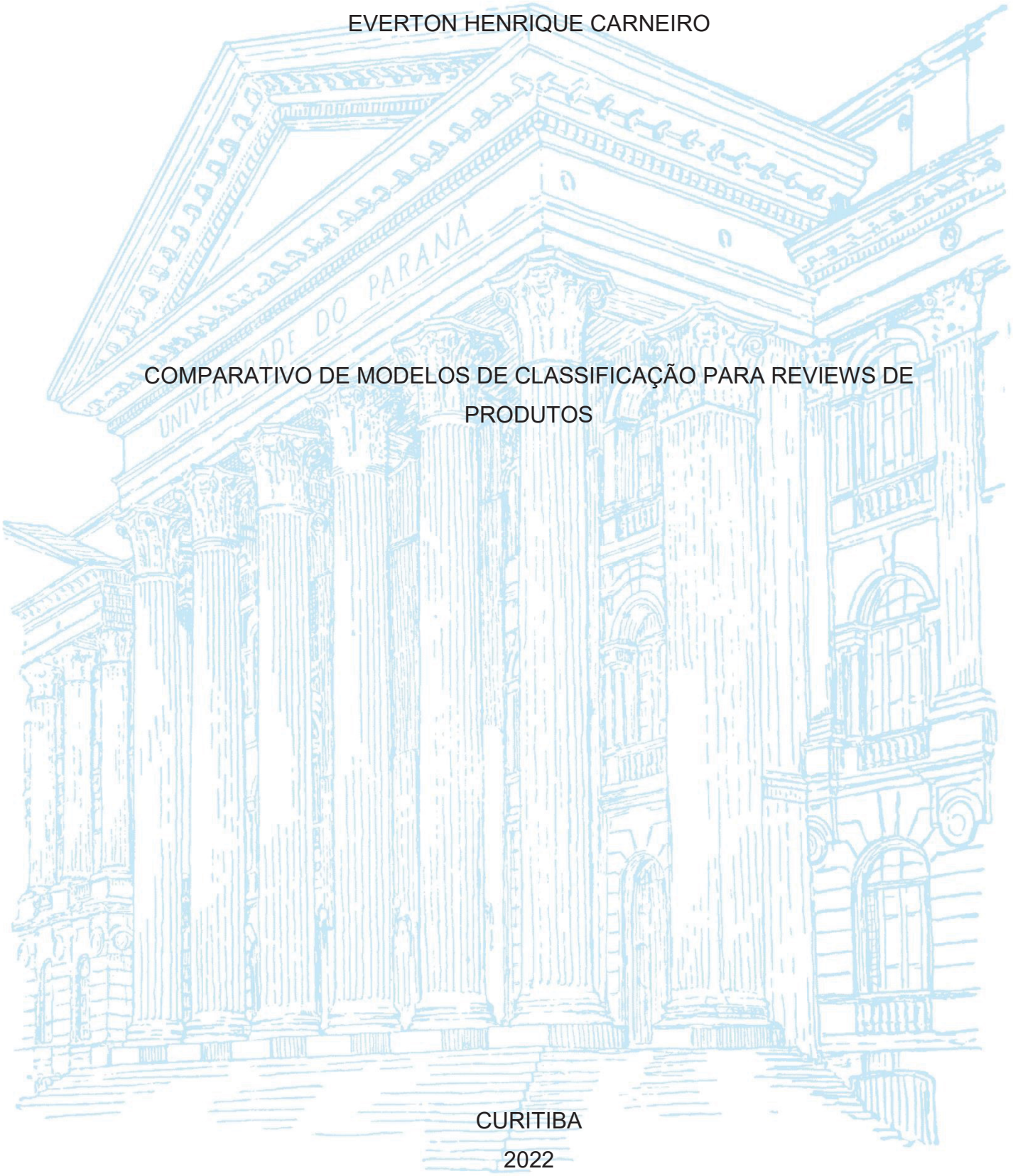
UNIVERSIDADE FEDERAL DO PARANÁ

EVERTON HENRIQUE CARNEIRO



COMPARATIVO DE MODELOS DE CLASSIFICAÇÃO PARA REVIEWS DE PRODUTOS

CURITIBA

2022

EVERTON HENRIQUE CARNEIRO

COMPARATIVO DE MODELOS DE CLASSIFICAÇÃO PARA REVIEWS DE PRODUTOS

Trabalho de Conclusão de Curso apresentado ao curso de Especialização em Inteligência Artificial Aplicada, Setor de Educação Profissional e Tecnológica, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Especialista em Inteligência Artificial Aplicada.

Orientador: Prof. Dr. Razer Anthom Nizer Rojas Montaño

CURITIBA

2022

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INTELIGÊNCIA ARTIFICIAL APLICADA da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **EVERTON HENRIQUE CARNEIRO** intitulada: **Comparativo De Modelos De Classificação Para Reviews De Produtos**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de especialista está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 29 de Novembro de 2022.

RAZER ANTHOM NIZER ROJAS MONTAÑO
Presidente da Banca Examinadora

JAIME WOJCIECHOWSKI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Comparativo de Modelos de Classificação Para Reviews de Produtos

Everton Henrique Carneiro
Especialização em Inteligência Artificial Aplicada
Universidade Federal do Paraná (UFPR)
Curitiba, Brasil
everton.carneiro@ufpr.br

Razer Anthom Nizer Rojas Montaño
Especialização em Inteligência Artificial Aplicada
Universidade Federal do Paraná (UFPR)
Curitiba, Brasil
razer@ufpr.br

***Resumo*—A pandemia da COVID-19 mudou os hábitos dos consumidores. Várias lojas digitais foram abertas, fazendo com que muitos clientes migrassem do ambiente físico para o digital. Isso se deu por conta da segurança, conveniência, economia de preço, economia de tempo e pela imensa quantidade de opções de produtos e vendedores. Com o aumento dos usuários, a disputa dessas lojas digitais por esses clientes também cresceu. Um dos fatores mais importantes na tomada de decisão de compra dos usuários é a avaliação de produtos. No entanto, pode ser difícil obter respostas, e principalmente, respostas com qualidade em relação as avaliações de produtos. Este artigo tem como objetivo desenvolver um comparativo entre quatro modelos de aprendizado de máquina, *Random Forest*, SVM (*Support Vector Machine*), MLP (*MultiLayer Perceptron*) e KNN (*K-Nearest Neighbors*). O objetivo desses modelos é o enriquecimento de dados de avaliações de produtos, isto é, *reviews* de produto, realizando a classificação das *reviews* de produtos como positivas ou negativas.**

***Index Terms*—Aprendizado de Máquina, SVM, MLP, KNN, Florestas Aleatórias, Avaliação de Produtos**

***Abstract*—The COVID-19 pandemic has changed consumer habits. Several digital stores were opened, causing many customers to migrate from the physical to the digital domain. This was due to safety, price savings, time savings and also due to the immense amount of products. With the increase in users, the dispute of these e-commerces for these customers grew as well. One of the most important factors in users' purchase decision making is product reviews. However, it can be difficult to get answers, and especially, quality answers in relation to reviews. This article aims to develop a comparative between four machine learning models, forest learning, SVM (Support Vector Machine), MLP (Multi Layer Perceptron) and KNN (K-Nearest Neighbors). The purpose of these models is to enrich product reviews data, performing a classification of product reviews as positive or negative.**

***Index Terms*—Machine Learning, SMV, MLP, KNN, Random Forest, Product Reviews**

I. Introdução

A pandemia da COVID-19 (*coronavirus disease* 2019) mudou os hábitos de consumo das pessoas. Muitas lojas digitais foram abertas e os consumidores passaram a optar por consumir produtos e serviços digitalmente [37]. Uma pesquisa recente mostrou que entre os respondentes, 11% da geração Z, 10% da geração Y, 12% da geração X e 5% dos *Baby Boomers* [1] compraram pela primeira vez durante a pandemia da COVID-19. Desta forma, 66% da geração Z, 68% da geração Y, 73% da geração X e 68% dos *Baby Boomers* adotaram as compras online após o aumento acentuado de lojas digitais devido a pandemia [21]. Tal aumento de possíveis clientes tem como consequência a intensificação da concorrência entre essas lojas online.

Um dos fatores que influencia significativamente a decisão de compra dos usuários são as *reviews* de produtos [5]. Uma pesquisa mostra que para 51% dos respondentes, as *reviews* de produtos são um dos maiores influenciadores na decisão de compra [7], ficando atrás somente do fator preço. Por conta disso, se faz necessário que as empresas aumentem a quantidade e qualidade das suas *reviews* de produto [42]. Um problema recorrente são *reviews* não respondidas ou respondidas de forma incompleta [24], ou seja, *reviews* com baixa qualidade. Essas *reviews* incompletas podem diminuir o potencial de venda de um produto e atrapalhar a experiência de outros usuários, além de trazerem um retorno incompleto do investimento em disparo de solicitação de *reviews* para os clientes.

A. Objetivo

Este estudo tem como objetivo treinar quatro modelos de aprendizado de máquina com a finalidade de analisar e comparar seus desempenhos em relação a classificação de *reviews* de produtos. Os modelos escolhidos para esta análise são *Random Forest*, *Support Vector Machine*, *Multilayer Perceptron* e *K-Nearest Neighbors*.

Desta forma, é esperado desenvolver uma ferramenta que auxilie empresas de varejo online a enriquecer seus dados de *reviews* de produto e por meio disso, diminuir custos envolvidos em relação a essas *reviews* e possibilitar com que os clientes dessas empresas tomem suas decisões de compra de maneira mais assertiva.

Tendo em vista esse objetivo geral, este artigo tem os seguintes objetivos específicos:

- Treinar 4 modelos de aprendizado de máquina distintos.

[1] *Baby boomers são pessoas nascidas entre 1946 e 1964, geração X são pessoas nascidas entre 1965 e 1980, geração Y são pessoas nascidas entre 1981 e 1996 e geração Z são pessoas nascidas entre 1997 e 2010 [64].*

- Comparar os resultados dos modelos.
- Conseguir um *Recall* e um *F1_Score* maior que 85% em pelo menos um dos modelos.

B. Estrutura do Artigo

A organização deste artigo consiste em cinco seções. Na seção um é apresentada a introdução deste trabalho, na qual contém uma contextualização, o problema abordado, os objetivos gerais e os objetivos específicos.

Na seção dois são apresentados os conceitos, fundamentação teórica, ferramentas e técnicas necessárias para o entendimento do trabalho proposto, ou seja, possui uma descrição desde o que é um *e-commerce* e a importância das *reviews* para esses *ecommerces* até os conceitos de aprendizado de máquina e uma breve explicação dos modelos utilizados neste trabalho.

Posteriormente, a seção três define e explica os materiais e métodos utilizados no desenvolvimento do projeto. Nesta seção são descritos as etapas desde a configuração do ambiente e apresentação das ferramentas utilizadas, passando pela coleta, explicação, exploração e preparação dos dados e por fim descrevendo as técnicas dos testes e métricas avaliativas escolhidas para este trabalho.

Na seção quatro, são apresentados e discutidos os resultados obtidos no decorrer da seção três. Por fim, a seção cinco encerra o trabalho resumindo os resultados, além disso, sugere maneiras para implementação do projeto e apresenta possibilidades de expansão do trabalho.

II. Fundamentação Teórica

Esta seção apresenta a fundamentação teórica dos conceitos e técnicas utilizadas no desenvolvimento deste trabalho.

A. E-commerce

Na década de 1990, a principal área de investimento nas bolsas de valores era a tecnologia, mais especificamente, a *internet*. Apenas tendo um "*.com*" no final do nome da empresa, as ações das empresas cresciam de 10% a 20% [39]. Isso combinado com a desregulamentação como meio de transmissão de dados na mesma época, a *internet* passou a mudar a vida das pessoas e a forma com que os seres humanos interagem [39].

Um dos adventos que a *internet* possibilitou e que vem mudando a vida das pessoas desde a década de 1990 são os *e-commerces*. Existem várias definições para a palavra *e-commerce*, Paul May em seu livro *The Business of E-commerce* comenta que a definição mais comum é: "a venda de itens por meio da *World Wide Web (www)*", ou seja, a *internet* [43].

Hoje existem *e-commerces* focados em diversas áreas como produtos eletrônicos, móveis e decoração para casa, roupas e até alimentação [3]. Atualmente no mundo os principais *e-commerces* são Amazon, Ebay, Shopee entre outros como ilustrado na Tabela I com uma estimativa de suas respectivas visitas mensais [34].

O Brasil possui mais de 200 milhões de habitantes e dentre esses, 150 milhões usam *internet*, tornando-o o quarto maior mercado de internet no mundo, os seus principais marketplaces são Mercado Livre, Americanas, OLX, entre outros. Como ilustrado na Tabela II, com uma estimativa de suas respectivas visitas mensais [17].

Tabela I:
TOP 10 *E-COMMERCES* DO MUNDO

Nome	Região	Visitas Mês
Amazon	Global	5.69B
eBay	Global	2.98B
Shopee	Sudeste da Ásia	631.19M
Rakuten	Global	590.84M
AliExpress	Global	526.4M
Wallmart	América do Norte	514.03M
Mercado Libre	América Latina	446.97M
Etsy	Global	397.5M
Taobao	China	333.14M
Wildberries	Russia	279.8M

Fonte: [34]

Tabela II:
TOP 10 *E-COMMERCES* DO BRASIL

Nome	Visitas Mês
Mercado Livre	260.93M
Americanas	134.58M
OLX	99.78M
Amazon Brasil	54.93M
Magazine Luiza	50.15M
Casas Bahia	36.35M
Netshoes	35.79M
Submarino	33.31M
Shoptime	19.46M
Dafiti	17.38M

Fonte: [17]

Os *e-commerces* tiveram um crescimento bastante significativo durante a pandemia do COVID-19, além do benefício de realizar as compras na segurança das suas casas, os usuários também conseguem poupar tempo, dinheiro, comparar preços de forma mais rápida e eficiente, conveniência, privacidade e um número infinitamente maior de opções de produtos [22]. Em uma pesquisa recente, 43% dos respondentes compraram em *e-commercers* durante a pandemia, em contraponto apenas 12% compravam antes da pandemia da COVID-19 [20]. A mesma pesquisa mostra que as compras recorrentes também aumentaram, passando de 9.8% para 25% pós-pandemia.

B. Reviews de Produto

Reviews de produto e notas são ferramentas populares para auxiliar a tomada de decisão de um cliente e também auxiliam vendedores *onlines* a construírem uma reputação confiável no mercado digital [23].

As pesquisas sobre os produtos de forma geral são enviadas para os clientes alguns dias após a entrega do produto ou serviço. Hoje, a fim de facilitar as respostas dos clientes, essas *Reviews* são enviadas por aplicativos como *WhatsApp*, SMS, e também por *e-mail*, dentre outras ferramentas de comunicação.

KPI, do inglês, *Key Performance Indicator*, isto é, indicador chave de desempenho, é uma ferramenta de performance.

Um KPI avalia o sucesso de uma organização, projeto ou uma atividade. O KPI simplifica a avaliação de desempenho para um ou mais critérios de indicadores chaves [41]. As *Reviews* de produto geram basicamente 3 KPIs importantes para uma companhia: a quantidade total de *Reviews*, taxa de recomendação de um produto e a quantidade média de estrelas de um produto.

Atualmente existem diversos trabalhos em relação a avaliação de produtos com técnicas de aprendizado de máquina como, por exemplo, classificação de sentimentos com base em *reviews* de produtos [52], identificação de *reviews* de produtos falsas [53], classificação de *reviews* de produtos [54], entre outros trabalhos.

C. Aprendizado de Máquina

Aprendizado de máquina, conhecido também como, *machine learning*, nasceu na década de 1960 como uma área de Inteligência Artificial e tinha o objetivo de aprender padrões baseados em dados [19]. Existem diversas técnicas de aprendizado de máquina, as mais populares são: aprendizado supervisionado, aprendizado não supervisionado, aprendizado semi-supervisionado, aprendizado por reforço e aprendizado por transferência [9].

Nas últimas décadas, as técnicas e aplicações de aprendizado de máquina tiveram um crescimento acentuado por conta do aumento do potencial computacional de gerar e processar dados [40]. Por conta disso, hoje temos aplicações de aprendizado de máquina em diversas áreas [35], como saúde, indústria, mercado de vendas, agricultura, entre outras.

1) Aprendizado Supervisionado: O Aprendizado Supervisionado acontece quando os dados de treinamento são fornecidos juntamente com as respostas desejadas. Essas respostas, também conhecidas como *target*, são usadas como ”supervisores” das previsões, possibilitando ajustes das previsões com base nos erros [16]. Exemplos de modelos supervisionados são: Máquina de Vetores de Suporte (SVM), Regressão Logística e Linear, *Naive Bayes*, Árvores de Decisão, entre outros.

2) Aprendizado Não Supervisionado: De forma contrária ao aprendizado supervisionado, no aprendizado não supervisionado, os dados não possuem as respostas para o modelo usar como referência no aprendizado [27]. Por exemplo, se for passado para um algoritmo de aprendizado não supervisionado dados sobre apartamentos, como tamanho, preço e região, o modelo pode separá-los como alto padrão e baixo padrão sem possuir previamente essas separações, isto é, sem supervisão. Alguns modelos de aprendizado não supervisionados são:*K-means*, redução de dimensionalidade, decomposição de valor singular, agrupamento hierárquico, entre outros.

D. Classificação

Classificação é uma técnica de *machine learning* e hierarquicamente está dentro das técnicas de aprendizado supervisionado como ilustrado na Figura 1.

A técnica de classificação é utilizada para prever a classe a qual um objeto pertence [38]. Para isso, como comentado na subseção anterior, o algoritmo de classificação recebe dados com as classes conhecidas e a partir desses dados, o algoritmo encontra as características que geram as classes.

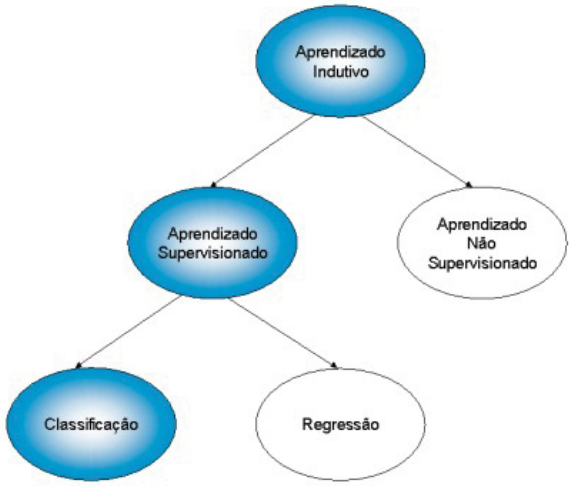


Fig 1. Hierarquia do Aprendizado. Fonte: [28]

Fig 2. Exemplo Classificação de *Spam*. Fonte: [16]

Um exemplo comum [16] é o filtro de *spam* de *e-mails*. Nesse contexto, um algoritmo de classificação recebe um conjunto de dados de *e-mails* com as classificações de *spam* e não *spam* e a partir dessa variável resposta aprende a classificar novos *e-mails* como ilustrado na Figura 2.

Os algoritmos de classificação possuem diversas possibilidades de aplicação. Além de classificação de *spam*, é usado também na classificação de câncer de mama [2], classificação de gênero musical [4], classificar de qualidade de água [30], classificação de sentimento [44], dentre outras diversas aplicações.

E. Floresta Aleatória

Floresta Aleatória, do inglês, *Random Forest*, é um algoritmo de aprendizado de máquina supervisionado que combina vários algoritmos de árvores de decisão para prever ou classificar o valor de uma variável [6].

Árvore de decisão é uma técnica poderosa e bastante usada em diversas áreas como aprendizado de máquina, processamento de imagem e identificação de padrão [8]. Uma árvore de decisão é composta por diversos nós. Dentre esses, existe o nó raíz (*root*), que possui maior nível hierárquico. Abaixo dele existem os nós filhos ou de separação (*split*), que consistem efetivamente em segmentar o caminho. Os nós filhos podem ter um ou mais filhos e seus filhos podem possuir filhos também.

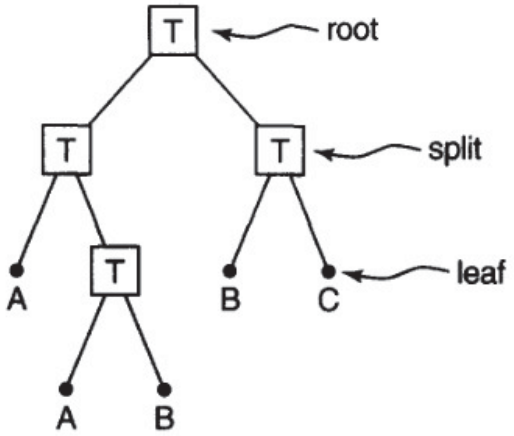


Fig 3. Exemplo Árvore de Decisão. Fonte: [14]

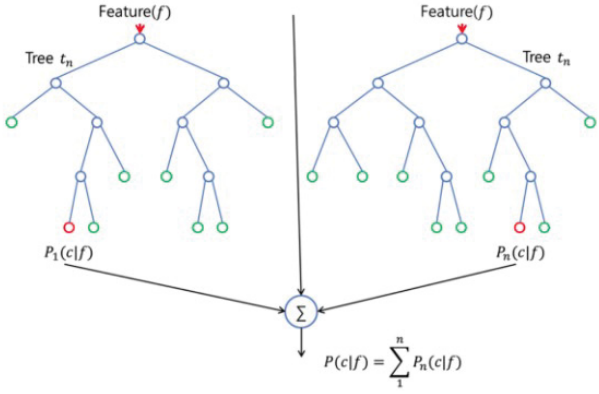


Fig 4. Exemplo Floresta Aleatória. Fonte: [10]

Por fim, o nó que não pode possuir filho, é o nó resposta, ou folha (*leaf*). Em uma árvore de decisão, cada nó interno, filhos, possui somente um nó pai e dois ou mais nós descendentes [14], como demonstrado na Figura 3.

O algoritmo de *Random Forest* cria um fluxo no qual uma condição é verificada e caso seja correspondida, o fluxo segue para o próximo ramo, e caso essa condição não seja correspondida, segue para um outro ramo, desta forma até o fim da árvore. A aleatoriedade do algoritmo se dá no momento em que essas árvores são criadas, pois ao invés de utilizar um método definido, o algoritmo escolhe de maneira aleatória as variáveis e então faz os cálculos baseado nessas amostras escolhidas. Isso é feito novamente, excluindo as amostras selecionadas anteriormente, até o último nó [33], como ilustrado na Figura 4.

A técnica *Random Forest* tem sido aplicada em trabalhos de diversas áreas como, por exemplo, classificação de imagem [55], previsão de diabetes [56], classificação de folhas de plantas [57], entre diversos outros.

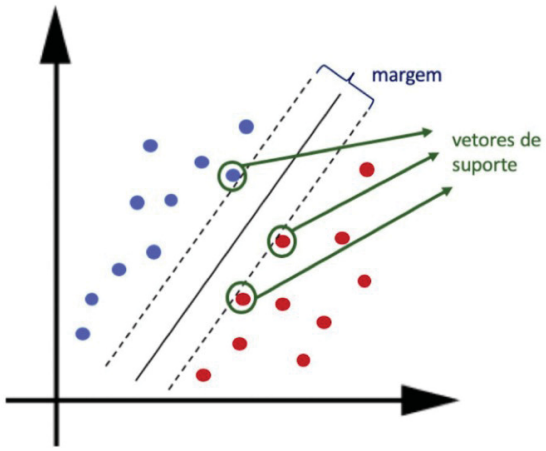


Fig 5. Exemplo SVM. Fonte: [11]

F. Máquina de Vetores de Suporte (SVM)

Máquina de Vetores de Suporte, do inglês, *Support Vector Machine* (SVM), é um algoritmo de aprendizado de máquina bastante popular. Em geral, o algoritmo SVM é usado bastante para problemas de classificação, porém pode ser usado também para problemas de regressão [26].

O *support vector machine* basicamente faz um mapeamento não linear com o objetivo de transformar os dados de treinamento em uma dimensão maior, caso necessário, com o objetivo de encontrar um hiperplano que separe esses dados de forma linear [11]. O SVM calcula esse hiperplano por meio de vetores de suporte e margens definidas por esses vetores como ilustrado na Figura 5.

Existem diversos hiperplanos e margens que podem separar um conjunto de dados em duas classes. Geralmente o classificador que possui os melhores resultados é chamado de ”classificador linear de margem máxima”, e portanto, é ele que tende a ser escolhido como melhor classificador [11].

Em geral, o treinamento do algoritmo *support vector machine* tende a ser lento, porém os classificadores geralmente precisam de poucos ajustes e apresentam bons resultados pois conseguem modelar bem fronteiras complexas e que não são lineares [13].

A técnica *Support Vector Machine* tem sido aplicada em trabalhos de diversas áreas como, por exemplo, categorização de textos [58], previsões de ações de mercado [59], previsão de doenças cardíacas [60], entre diversos outros.

G. Redes Neurais

Inicialmente as redes neurais foram desenvolvidas com o objetivo de simular os processos cerebrais de um ser humano [32]. Assim se tornou uma ferramenta tecnológica com aplicabilidade em diversas tarefas em áreas como saúde [25], meio ambiente [31], reconhecimento de padrões em imagens [1], entre diversas outras.

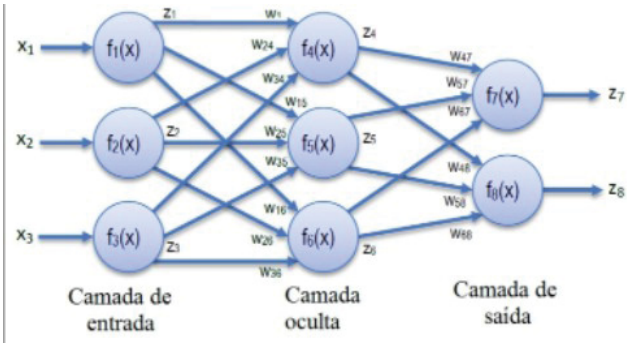


Fig 6. Exemplo Redes Neurais. Fonte: [29]

Redes neurais, do inglês, *neural networks*, são modelos que consistem em unidades de processamento, que são chamados de neurônios artificiais, como ilustrado na Figura 6.

Segundo [18], redes neurais artificiais são bastante eficientes em mapear entradas e saídas de sistemas não lineares e simular sistemas complexos. As redes neurais fazem uma generalização dos resultados, isto é, desenvolvem respostas para padrões que não necessariamente foram providos em seu treinamento.

As redes neurais artificiais possuem três tipos de camadas, a camada de entrada que é responsável por receber os dados, camadas ocultas que tem a função de extrair informações desses dados e por fim a camada de saída que possui a responsabilidade de concluir e apresentar o resultado final [18]. Tais redes também possuem um algoritmo de retropropagação que tem a responsabilidade de ajustar os pesos sinápticos com o objetivo a alcançar o resultado alvo durante as etapas de treinamento [29], como pode ser visto na Figura 6.

Uma rede neural artificial do tipo *Multilayer Perceptron* consiste basicamente em um conjunto de neurônios que formam uma camada de entrada, uma ou mais camadas intermediárias ou ocultas e, por fim, uma camada de saída. O dado então é propagado camada a camada até a obtenção do resultado final [12] como ilustrado na Figura 7.

A técnica *Multilayer Perceptron* tem sido aplicada em trabalhos de diversas áreas como, por exemplo, previsão de produção de alimento [61], previsões de diabetes [61], detecção de anomalias em *cyber* segurança [62], entre diversos outros.

H. K-Nearest Neighbors (KNN)

O *K-Nearest Neighbors*, ou seja, K-vizinhos mais próximos, é um algoritmo de aprendizado de máquina supervisionado bastante utilizado para problemas de classificação [45]. Foi proposto originalmente em 1967 [46]. Cover e Hart mostram que assumindo que todas as instâncias estando em um determinado plano dimensional, no qual ”m”é o número de atributos, por meio de uma função de distância é possível determinar a proximidade de uma instância da outra. Desta forma, é possível classificar as instâncias com a classe que mais aparece nos K-vizinhos mais próximos [46] como ilustrado na Figura 8.

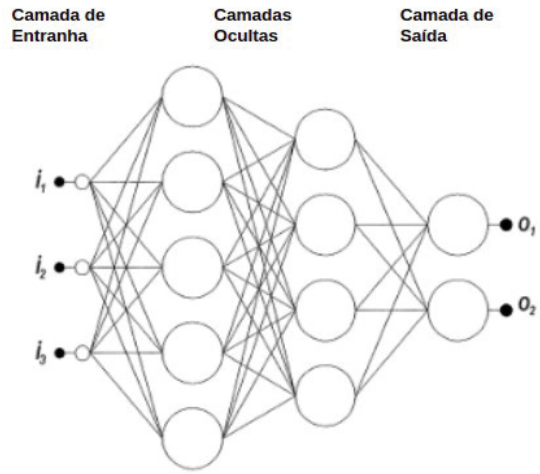


Fig 7. Exemplo Perceptron Multicamadas. Fonte: [15]

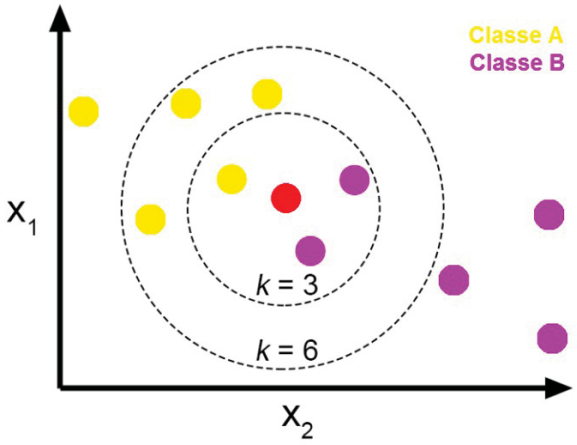


Fig 8. Exemplo KNN. Fonte: [47]

Quando o valor de K-vizinhos é pequeno, a classificação fica bastante sensível para as regiões próximas, o que pode gerar um *overfiting* para o modelo. Contudo, com um valor alto para K-vizinhos a classificação fica menos sensível a ruídos, porém com o K alto demais, o classificador pode sofrer um *underfiting*.

A técnica *K-Nearest Neighbors* tem sido aplicada em trabalhos de diversas áreas como, por exemplo, detecção de câncer de pele [48], detecção de expressões faciais [49], análise de sentimentos [50], entre diversos outros.

III. Materiais e Métodos

Nesta seção são apresentadas as etapas realizadas para o desenvolvimento da comparação de modelos de classificação para base de *review* de produtos. As etapas são:

1) Definição das Ferramentas e Ambiente.
2) Definição da Base de Dados.

3) Exploração dos Dados.
4) Preparação dos Dados para Modelagem.
5) Testes e Validação dos Parâmetros e Algoritmos.
6) Definição de Métricas de Avaliação

A. *Definição da Ferramentas e Ambiente*

Todos os preparos, testes e comparações propostas foram realizados com a linguagem de programação *Python* na versão 3.8. Para a manipulação dos dados, o principal módulo utilizado foi o Pandas. Para construção, treinamento e validação dos modelos, foi utilizado o módulo *Scikit-Learn*. Por fim, para apresentação visual dos dados e resultados, foram utilizados os módulos *Seaborn* e *Matplotlib*. Todos os experimentos foram realizados no *Jupyter Notebook* executado localmente.

B. *Definição da Base de Dados*

A base escolhida para utilização no desenvolvimento do comparativo é uma base de dados de avaliação de produtos. Os dados são abertos na *internet* e se encontram descentralizados em um site de comércio *online* na área de bens para o lar. A base com os dados compilados foi disponibilizada pelo varejista online no modelo ".xlsx".

O conjunto de dados consiste em uma base de 387.328 registros, isto é, avaliações de produto, e possui 14 colunas. As colunas e suas descrições se encontram na Tabela III.

Tabela III:
DESCRIÇÃO DO CONJUNTO DE DADOS.

Coluna	Descrição
id	identificador do registro
id_produto	id do produto avaliado
rate	nota atribuída ao produto (1 a 5)
text	comentário em relação ao produto
up_votes_count	total de curtidas da avaliação
down_votes_count	total de descurtidas da avaliação
data_review	data da avaliação
data_envio	data do envio da avaliação
product_name	nome do produto
product_average_rate	nota média do produto (1 a 5)
product_total_recommends	total de avaliações do produto
product_total_opinions	total de opiniões do produto
product_links	*links* do produto
recommends	recomendação (1 ou 0)

Fonte: o autor (2022)

C. *Exploração dos Dados*

O conjunto de dados utilizado contempla avaliações realizadas no período entre outubro de 2019 e julho de 2022, possuindo os meses com mais quantidade de avaliações, maio, junho e julho de 2020 como demonstrado na Figura 9.

Seguindo a etapa de exploração dos dados, foi observado que a coluna *rate*, isto é, a nota atribuída a cada avaliação, possuía uma distribuição desproporcional, como pode ser visto na Figura 10, concentrando na nota 5 aproximadamente 60% dos registros.

A variável dependente também possui uma desproporcionalidade em relação a distribuição dos registros, como pode ser

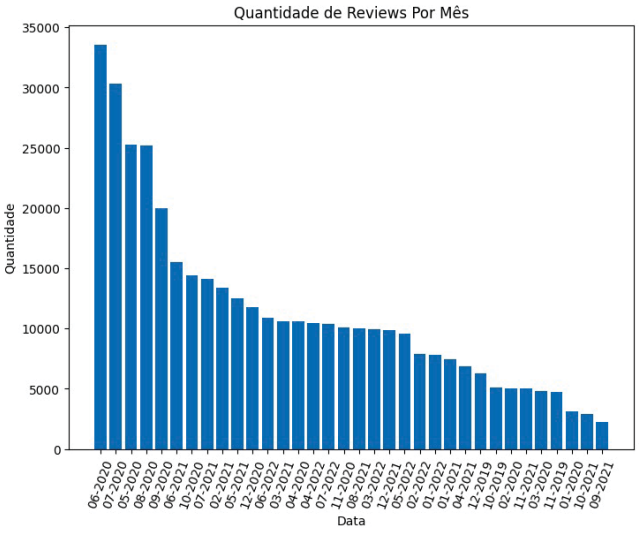


Fig 9. Fonte: o autor (2022)

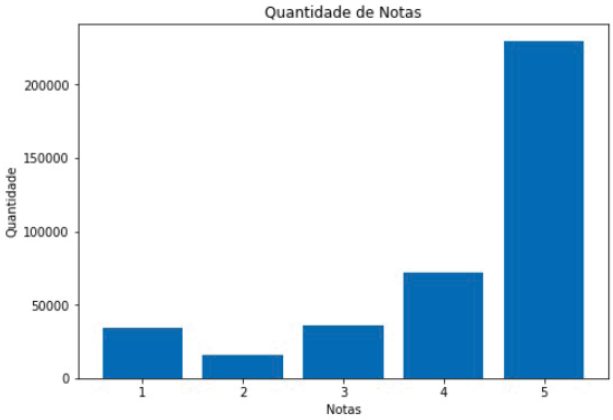


Fig 10. Fonte: o autor (2022)

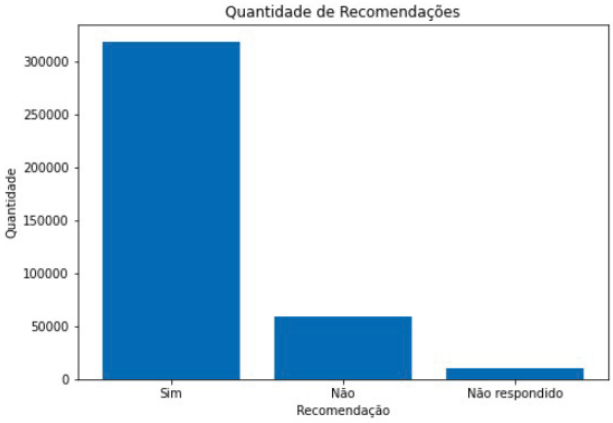


Fig 11. Fonte: o autor (2022)

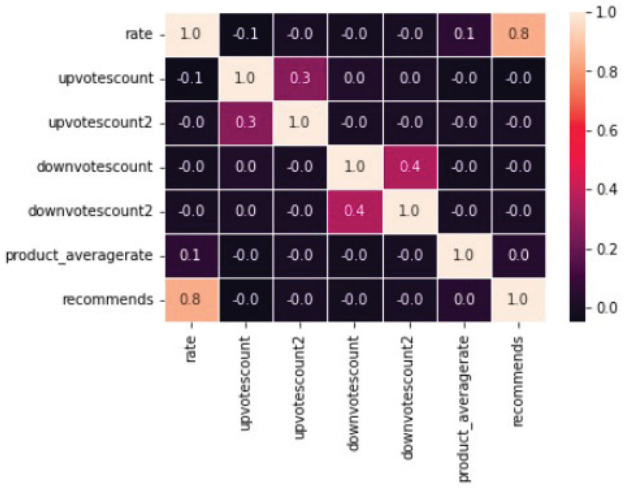


Fig 12. Análise de Correlação. Fonte: o autor (2022)

visto na Figura 11. Possuindo, inclusive, 2.46% dos registros sem respostas.

Em seguida, foi realizado uma análise de correlação entre as variáveis e foi possível observar que existe uma forte correlação entre a variável *rate* e a variável dependente *recommends*, como ilustrado na Figura 12.

D. Preparação dos Dados Para Modelagem

Para a preparação dos dados para modelagem, primeiramente foram removidos os registros com a variável dependente sem resposta. Desta forma, o conjunto de dados passou a ter somente registros ou com o valor ”Sim” ou com o valor ”Não” na váriavel *recommends*.

Como visto na subseção ”Exploração dos Dados”, o conjunto de dados utilizado possui a necessidade de ajustes pois têm alguns campos com valores desbalanceados.

A primeira coluna ajustada foi a *rate*. A coluna *rate* possui uma distribuição desigual, como pode ser visto na Tabela IV. Para esse ajuste, a coluna *rate* foi balanceada com base no valor com menor quantidade. Portanto, todos os valores foram, aleatoriamente, amostrados em 14601 registros.

Tabela IV:
DISTRIBUIÇÃO *RATE*

Nota	Quantidade	%
5	227551	60.27%
4	70253	18.61%
3	33054	8.75%
2	14601	3.87%
1	32107	8.5%

Fonte: o autor (2022)

Desta mesma maneira, a variável dependente, *recommends*, foi balanceada com base no valor com menor quantidade de registros, que neste caso foi o valor ”Não”, como visto na Tabela V. Portanto, a base ficou com, aleatoriamente, 33.385 registros para recomendações com o valor ”Sim”e todos os registros de recomendações com o valor ”Não”.

Tabela V:
DISTRIBUIÇÃO *RECOMMENDS*

Recomendação	Quantidade	%
Sim	39620	54.27%
Não	33385	45.73%

Fonte: o autor (2022)

Além disso, a coluna *recommends* foi transformada para valores binários, isto é, o valor ”Sim” recebeu o valor ”1” e o valor ”Não” recebeu o valor ”0”. Essa transformação foi realizada para que a coluna pudesse ser operada nos testes dos modelos da subseção ”Testes e Avaliação dos Parâmetros e Algoritmos Escolhidos”.

Por fim, foram criadas mais duas características para a etapa de ”Testes e Avaliação dos Parâmetros e Algoritmos Escolhidos”. A primeira foi com base na coluna *up_votes_count*, que foi elevada ao quadrado. A segunda, da mesma maneira, foi o quadrado da coluna *down_votes_count*. A intenção foi reforçar o peso desses campos, pois os dois têm a finalidade de validação ou reprovação da *review* dada pelo usuário.

E. Testes e Validação dos Parâmetros e Algoritmos

Os quatros modelos escolhidos foram *Random Forest*, SVM (Máquina de Vetores de Suporte), MLP (*Perceptron Multicamadas*) e KNN (*K-Nearest Neighbors*).

Primeiramente, foi utilizada a metodologia *GridSearch* por meio da função *GridSearchCV* do módulo *sklearn*. O *GridSearch* consiste em um técnica de automatização de testes dos hiperparâmetros de modelos. O *GridSearchCV* faz testes de modo sistêmico de diversas possibilidades de combinações dos hiperparâmetros. Após cada teste, realiza a validação de modelos e armazena os resultados em um objeto.

Foram utilizados *F1 Score* e *Recall* como métricas de validação do *GridSearchCV*, sempre priorizando *Recall* por conta do propósito do modelo como explicado na subseção seguinte, ”Definição de Métricas de Avaliação”.

Neste trabalho, foi utilizado o *GridSearchCV* com combinações numéricas de limites inferiores e superiores e a cada interação, esse limite foram reajustados. Por exemplo, primeiramente foi realizado um teste com o modelo *Random Forest* com 100 e 1000 estimadores. Como o *GridSearchCV* apontou que o melhor modelo foi o com 1000, foi realizado outro teste com 500 a 1500. Essa metodologia foi utilizada até ser obtido hiperparâmetros próximo aos ideais para os modelos testados. Além dos testes automatizados, o *GridSearchCV* possui um método de *Cross Validation*. Nesta etapa foi utilizada a técnica *Hold-out* com segmentação de 80% do conjunto de dados para treino e 20% para teste. Posteriormente os melhores resultados de cada modelo, passaram também pelo método *RepeatedKFold* para fins de comparação.

Na Tabela VI, Tabela VII, Tabela VIII e Tabela IX, é possível ver os hiperparâmetros testados para os modelos *Random Forest*, Máquina de Vetores de Suporte, Perceptron Multicamadas e KNN respectivamente.

Tabela VI:
HIPERPARÂMETROS *RANDOM FOREST*

Hiperparâmetros	Valores
n_estimators	100 a 1900
max_features	*sqrt, log2*
criterion	*gini, entropy, log_loss*

Fonte: o autor (2022)

Tabela VII:
HIPERPARÂMETROS *SVM*

Hiperparâmetros	Valores
C	0.02 a 0.008
kernel	*rbf, poly, sigmoid, linear*
degree	1 a 3
gamma	0.001 a 0.00005

Fonte: o autor (2022)

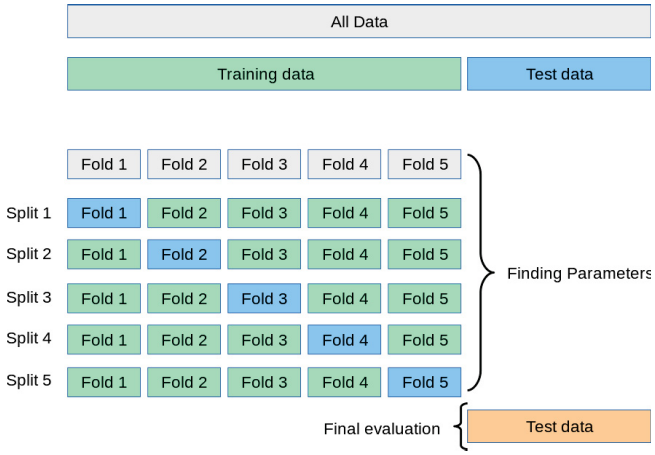


Fig 13. Exemplo *K-fold* Fonte: [36]

Para três dos melhores resultados de cada modelo obtido pelo *GridSearch*, foi utilizado o método de avaliação *RepeatedKFold* para eliminar possíveis vieses. O método *K-fold* é um método de reamostragem que consiste em dividir o conjunto de dados em K partes iguais. Sempre utilizando uma das porções para teste e as demais para treino. Esse processo é repetido K vezes, desta forma, a cada interação os conjuntos de teste e treino são diferentes, como pode ser visto na Figura 13. O método *RepeatedKFold* consiste em repetir o *KFold* N vezes. Durante as interações, os resultados foram armazenados em vetores e, posteriormente, foi calculada a média desses vetores com o objetivo de diminuir distorções. Os testes foram realizados com K igual 10, isto é, dez subdivisões e esse processo foi repetido 5 vezes.

É importante ressaltar que quanto maior o K definido, o viés tende a diminuir, porém a variância tende a aumentar. Desta mesma forma, quanto mais baixo o K definido, o viés será maior e a variância menor.

Para treinamento foram utilizadas as mesmas características para os quatros modelos. As características foram:

- *rate*
- *up_votes_count*
- *up_votes_count2*
- *down_votes_count*
- *down_votes_count_2*
- *product_average_rate*

Tabela VIII:
HIPERPARÂMETROS *MLP*

Hiperparâmetros	Valores
hidden_layer_sizes	Diversas arquiteturas
activation	*identity, logistic, tanh, relu*
solver	*lbfgs, sgd, adam*
alpha	0.0001 a 0.005
learning_rate	*constant,invscaling, adaptive*

Fonte: o autor (2022)

Tabela IX:
HIPERPARÂMETROS *KNN*

Hiperparâmetros	Valores
algorithm	*auto, ball_tree, kd_tree, brute*
metric	*euclidean, manhattan*
n_neighbors	1 a 19
weights	*uniform, distance*

Fonte: o autor (2022)

F. Definição de Métricas de Avaliação

As principais métricas escolhidas para a comparação dos modelos foram *recall* e *F1_score*.

O *recall*, ou seja, sensibilidade, indica as previsões positivas em relação a todas as previsões que realmente são positivas, como pode ser lido na Fórmula 1.

$$recall = \frac{TP}{TP + FN} \tag{1}$$

O *Recall* é bastante útil quando se faz necessário ver os falsos negativos, isto é, indica o quanto dos casos positivos o modelo está classificando corretamente. Essa métrica foi escolhida pois é importante não atribuir uma classificação positiva para uma avaliação de produto negativa, pois pode influenciar a compra de um produto ruim para um cliente, gerando problemas e custos para a empresa.

A segunda métrica escolhida foi o *F1_score*, que é, basicamente, uma maneira de observar o *Recall* e a precisão juntas. A precisão é definida pela relação entre quantidade de precisões classificadas como positivas e o total de previsões positivas como pode ser visto na Fórmula 2.

$$precision = \frac{TP}{TP + FP} \tag{2}$$

O *F1_score* é determinado pela média harmônica entre a precisão e o *Recall*, como ilustrado na Fórmula 3. Uma das características do *F1_score* é que se uma das variáveis for próximo de zero, o *F1_score* será baixo. Consequentemente, se a precisão e o *Recall* foram altas, o *F1_score* também será alto.

$$F1Score = 2 * \frac{precision * recall}{precision + recall} \quad (3)$$

É importante ressaltar que métricas como acurácia, precisão e especificidade também foram analisadas, porém com menor importância, considerando o propósito do modelo. As métricas foram calculadas por meio da função *Metrics* do módulo *Scikit Learn* [51].

IV. Resultados e Discussões

Nesta seção são apresentados os resultados obtidos por meio dos métodos e ferramentas apresentados na seção três deste artigo.

Primeiramente, como discutido na seção três deste artigo, os algoritmos passaram pela metodologia *Grid Search* com o objetivo de encontrar os melhores hiperparâmetros que gerassem os melhores classificadores segundo as métricas apresentadas na seção anterior priorizando o *Recall* (*recall*). Posteriormente, as melhores combinações de hiperparâmetros passaram pela metodologia *Cross Validation Repeated KFold* a fim de submeter as melhores combinações de hiperparâmetros encontrados na primeira etapa para uma validação mais robusta.

A. Grid Search

1) Random Forest: O primeiro algoritmo a ser testado foi o *Random Forest*. Nesta primeira execução, foram testadas 60 combinações diferentes de hiperparâmetros para o modelo. As 10 melhores combinações de hiperparâmetros e seus respectivos resultados estão demonstradas na Tabela X.

Como pode ser visto na Tabela X, os resultados obtidos na primeira execução do *Grid Search*, possuem todos os valores iguais tanto para *Recall* quanto para *F1_score*.

Os hiperparâmetros para os números de estimadores ficou entre 100 e 300, portanto, foi realizada mais uma execução do *Grid Search* com números de estimadores entre 100 e 300. Para os demais hiperparâmetros, os valores foram mantidos pois todos apareceram entre os melhores classificadores.

Portanto, foram testados mais 30 diferentes combinações para o algoritmo *Random Forest*. As 10 melhores combinações de hiperparâmetros da segunda execução e seus respectivos resultados estão demonstradas na Tabela XI.

Como demonstrado na Tabela X, os resultados das melhores 10 combinações obtiveram os mesmos resultados para *Recall* e *F1_Score*. Já na segunda execução do *Grid Search*, os valores de *Recall* e *F1_Score* não mudaram para as 6 melhores combinações de hiperparâmetros, como demonstrado Tabela XI. Porém, com o valor 150 para o número de estimadores, tanto o *Recall* quanto o *F1_Score* aumentaram de 0.9332 para 0.9335 e 0.8858 para 0.8859 respectivamente.

2) SVM: Posteriormente o algoritmo *support vector machine* foi testado. Na primeira execução do *Grid Search*, foram testadas 32 combinações diferentes de hiperparâmetros para o modelo. As 10 melhores combinações de hiperparâmetros e seus respectivos resultados estão demonstrados na Tabela XII.

Tabela X:
1º *GRIDSEARCH RANDOM FOREST*

Hiperparâmetros	*Recall*	*F1 Score*	*Accuracy*
criterion : gini max features : sqrt n estimators : 100	0.9332	0.8858	0.8800
criterion : entropy max features : sqrt n estimators : 100	0.9332	0.8858	0.8800
criterion : log loss max features : log2 n estimators : 300	0.9332	0.8858	0.8800
criterion : log loss max features : log2 n estimators : 100	0.9332	0.8858	0.8800
criterion : log loss max features : sqrt n estimators : 300	0.9332	0.8858	0.8800
criterion : log loss max features : sqrt n estimators : 100	0.9332	0.8858	0.8800
criterion : entropy max features : log2 n estimators : 300	0.9332	0.8858	0.8800
criterion : gini max features : sqrt n estimators : 300	0.9332	0.8858	0.8800
criterion : entropy max features : sqrt n estimators : 300	0.9332	0.8858	0.8800
criterion : entropy max features : log2 n estimators : 100	0.9332	0.8858	0.8800

Fonte: o autor (2022)

Tabela XI:
2º *GRIDSEARCH RANDOM FOREST*

Hiperparâmetros	*Recall*	*F1 Score*	*Accuracy*
criterion : log loss max features : log2 n estimators : 150	0.9335	0.8859	0.8799
criterion : gini max features : log2 n estimators : 150	0.9335	0.8859	0.8799
criterion : entropy max features : log2 n estimators : 150	0.9335	0.8859	0.8799
criterion : gini max features : sqrt n estimators : 150	0.9335	0.8859	0.8799
criterion : log loss max features : sqrt n estimators : 150	0.9335	0.8859	0.8799
criterion : entropy max features : sqrt n estimators : 150	0.9335	0.8859	0.8799
criterion : log loss max features : sqrt n estimators : 200	0.9333	0.8859	0.8800
criterion : gini max features : sqrt n estimators : 200	0.9333	0.8859	0.8800
criterion : entropy max features : log2 n estimators : 200	0.9333	0.8859	0.8800
criterion : log loss max features : log2 n estimators : 200	0.9333	0.8859	0.8800

Fonte: o autor (2022)

Tabela XII:
1º *GRIDSEARCH SVM*

Hiperparâmetros	*Recall*	*F1 Score*	*Accuracy*
C : 0.01 *degree : 2* *gamma : 0.0005* *kernel : poly*	1	0.6667	0.5
C : 0.01 *degree : 2* *gamma : 0.00005* *kernel : poly*	1	0.6667	0.5
C : 0.01 *degree : 3* *gamma : 0.0005* *kernel : poly*	1	0.6667	0.5
C : 0.01 *degree : 3* *gamma : 0.00005* *kernel : poly*	1	0.6667	0.5
C : 0.05 *degree : 2* *gamma : 0.0005* *kernel : poly*	1	0.6667	0.5
C : 0.05 *degree : 2* *gamma : 0.00005* *kernel : poly*	1	0.6667	0.5
C : 0.05 *degree : 3* *gamma : 0.0005* *kernel : poly*	1	0.6667	0.5
C : 0.05 *degree : 3* *gamma : 0.00005* *kernel : poly*	1	0.6667	0.5
C : 0.01 *degree : 2* *gamma : 0.0005* *kernel : linear*	0.9503	0.8899	0.8824
C : 0.01 *degree : 2* *gamma : 0.00005* *kernel : sigmoid*	0.9503	0.8899	0.8824

Fonte: o autor (2022)

Tabela XIII:
2º *GRIDSEARCH SVM*

Hiperparâmetros	*Recall*	*F1 Score*	*Accuracy*	Execução
C : 0.008 *degree : 2* *gamma : 0.0001* *kernel : poly*	1	0.6667	0.5	1
C : 0.008 *degree : 2* *gamma : 0.0005* *kernel : poly*	1	0.6667	0.5	2
C : 0.008 *degree : 2* *gamma : 0.001* *kernel : poly*	1	0.6667	0.5	3
C : 0.008 *degree : 3* *gamma : 0.0001* *kernel : poly*	1	0.6667	0.5	4
C : 0.008 *degree : 3* *gamma : 0.0005* *kernel : poly*	1	0.6667	0.5	5
C : 0.008 *degree : 3* *gamma : 0.001* *kernel : poly*	1	0.6667	0.5	6
C : 0.008 *degree : 1* *gamma : 0.0001* *kernel : poly*	0.9898	0.7692	0.7031	13
C : 0.008 *degree : 1* *gamma : 0.0005* *kernel : poly*	0.9898	0.7692	0.7031	14
C : 0.008 *degree : 1* *gamma : 0.0001* *kernel : sigmoid*	0.9503	0.8899	0.8824	16
C : 0.008 *degree : 1* *gamma : 0.0005* *kernel : linear*	0.9503	0.8899	0.8824	17

Fonte: o autor (2022)

Como pode ser visto na Tabela XII, as 8 melhores combinações de hiperparâmetros obtiveram os mesmo resultados para *recall* e *F1_score*. Na nona e décima combinação, os valores de *Recall* baixaram de 100% para 95%, porém os valores de *F1_score* subiram de 67% para 89%.

Como nos 10 melhores valores para *Kernel* não apareceu o valor ”rfb”, para a segunda execução do *Grid Search*, ele foi removido. Para os valores de *degree*, foi incluso também o valor ”1”. Desta mesma forma, foi incluso o valor ”0.001” para *gamma* para a segunda execução do *Grid Search*. Por fim, para os valores de ”C”, foi incluso o valor ”0.008”.

Portanto, foram testadas mais 70 combinações diferentes para o algoritmo *support vector machine*. As melhores combinações de hiperparâmetros da segunda execução do *Grid Search* e seus respectivos resultados estão demonstrados na Tabela XIII.

Além dos 6 primeiros resultados da Tabela XIII, as primeiras 12 melhores combinações de hiperparâmetros na segunda execução do *Grid Search* obtiveram os mesmos resultados de *Recall* e *F1_score* e seus valores respectivamente foram 100% e 67%. Os resultados 13 e 14 foram as primeiras amostras a apresentarem valores diferentes dos primeiros resultados, obtendo 99% de *Recall* e 77% de *F1_score*. Nas execuções 16 e 17, os valores de *Recall* foram 95% e os valores de *F1_score* foram 89%.

Em relação aos hiperparâmetros da segunda execução do *Grid Search*, todas as combinações demonstradas na Tabela XIII tiveram o valor de ”C”de ”0.008”. Da mesma forma que a primeira execução do *Grid Search*, dentre as melhores combinações de hiperparâmetros, a maioria teve a valor *”poly”* para o parâmetro *”kernel”*. Por fim, para os valores de *degree* e *gamma*, todos os valores testados apareceram nas combinações que obtiveram os melhores resultados.

3) MLP: Da mesma forma, o algoritmo *Multilayer Perceptron* foi testado. Na primeira execução do *Grid Search*, foram testadas 756 combinações diferentes de hiperparâmetros para o modelo. As 10 melhores combinações de hiperparâmetros e seus respectivos resultados estão demonstrados na Tabela XIV.

Tabela XIV:
1º *GRIDSEARCH MLP*

Hiperparâmetros	*Recall*	*F1 Score*	*Accuracy*
activation : logistic *alpha : 0.001* *hidden_layer_sizes : (10, 30, 10)* *learning_rate : invscaling* *solver : sgd*	1	0.6667	0.5
activation : logistic *alpha : 0.0001* *hidden_layer_sizes : (3,3)* *learning_rate : invscaling* *solver : sgd*	1	0.6667	0.5
activation : logistic *alpha : 0.005* *hidden_layer_sizes : (50, 100, 50)* *learning_rate : invscaling* *solver : sgd*	1	0.6667	0.5
activation : logistic *alpha : 0.005* *hidden_layer_sizes : (3,3)* *learning_rate : invscaling* *solver : sgd*	1	0.6667	0.5
activation : logistic *alpha : 0.0001* *hidden_layer_sizes : (50, 50, 50)* $learning_rate$: $constant$ *solver : sgd*	0.9999	0.6667	0.5
activation : logistic *alpha : 0.005* *hidden_layer_sizes : (10, 30, 10)* $learning_rate$: $constant$ *solver : sgd*	0.9998	0.6678	0.5
activation : relu *alpha : 0.005* *hidden_layer_sizes : (3,3)* *learning_rate : invscaling* *solver : sgd*	0.9958	0.7094	0.55
activation : logistic *alpha : 0.001* *hidden_layer_sizes : (20,)* *learning_rate : invscaling* *solver : sgd*	0.9950	0.7196	0.6634
activation : identity *alpha : 0.001* *hidden_layer_sizes : (3,3)* *learning_rate : invscaling* *solver : sgd*	0.9804	0.7530	0.7377
activation : logistic *alpha : 0.0001* *hidden_layer_sizes : (20,)* *learning_rate : invscaling* *solver : sgd*	0.9787	0.7663	0.7580

Fonte: o autor (2022)

Como pode ser visto na Tabela XIV, as 4 melhores combinações de hiperparâmetros obtiveram os mesmos resultados para *Recall* e *F1_score*. Nas demais execuções, os valores de *Recall* diminuíram gradualmente até as 97% e o valor de *F1_score* aumentou gradualmente até 77%.

Em relação ao hiperparâmetro *activation*, na maioria das melhores execuções os valores ficaram entre *logistic* e *relu*, portanto os valores *tanh* e *identity* foram removidos da segunda execução. Para os valores de *alpha*, foram adicionados os valores ”0.003” e ”0.0005” e os valores ”0.005” e ”0.0001” foram removidos da segunda execução. Além das melhores configurações para o parâmetro *hidden_layer_sizes*, foram testadas outras configurações de *hidden_layer_sizes*. Em relação ao parâmetro de *solver*, o valor *adam* foi removido da segunda execução pois não apareceu nenhuma vez nas melhores combinações de hiperparâmetros da primeira execução. Por fim, para o parâmetro *learning_rate* o valor ”*adaptive*”foi removido da segunda execução pois também não apareceu nas melhores combinações de hiperparâmetros da primeira execução.

Portanto, foram testadas mais 144 combinações diferentes de hiperparâmetros para o algoritmo *Multilayer Perceptron*. As melhores combinações da segunda execução do *Grid Search* e seus respectivos resultados estão demonstrados na Tabela XV.

Tabela XV:
2º *GRIDSEARCH MLP*

Hiperparâmetros	*Recall*	*F1 Score*	*Accuracy*	Execução
activation : relu *alpha : 0.001* *hidden_layer_sizes : (3, 3, 2)* *learning_rate : invscaling* *solver : sgd*	1	0.6666	0.5	1
activation : logistic *alpha : 0.003* *hidden_layer_sizes : (10,)* *learning_rate : invscaling* *solver : sgd*	1	0.6667	0.5	2
activation : logistic *alpha : 0.003* *hidden_layer_sizes : (10, 30, 10)* *learning_rate : invscaling* *solver : sgd*	1	0.6666	0.5	3
activation : logistic *alpha : 0.001* *hidden_layer_sizes : (3, 3, 3)* *learning_rate : invscaling* *solver : sgd*	1	0.6666	0.5	4
activation : relu *alpha : 0.003* *hidden_layer_sizes : (3, 3, 2)* *learning_rate : constant* *solver : lbfgs*	1	0.6666	0.5	5
activation : relu *alpha : 2* *hidden_layer_sizes : (3, 3, 3)* *learning_rate : invscaling* *solver : lbfgs*	1	0.6667	0.5	6
activation : relu *alpha : 2* *hidden_layer_sizes : (3, 3, 3)* *learning_rate : invscaling* *solver : sgd*	0.9998	0.6666	0.5	17
activation : relu *alpha : 0.001* *hidden_layer_sizes : (10,)* *learning_rate : invscaling* *solver : sgd*	0.9981	0.6826	0.6370	18
activation : logistic *alpha : 0.001* *hidden_layer_sizes : (10,)* *learning_rate : invscaling* *solver : sgd*	0.9542	0.8356	0.81	24
activation : relu *alpha : 0.001* *hidden_layer_sizes : (3, 3, 3)* *learning_rate : constant* *solver : sgd*	0.9491	0.8881	0.8825	26

Fonte: o autor (2022)

Além dos 6 primeiros resultados da Tabela XV, as primeiras 15 melhores combinações de hiperparâmetros na segunda execução do *Grid Search* obtiveram os mesmos resultados de *Recall* e *F1_score* e seus valores respectivamente foram 100% e 67%. Os resultados 17 e 18 foram as primeiras amostras a apresentarem valores diferentes dos primeiros resultados, obtendo 99% de *Recall* e o mesmo valor de 67% de *F1_score*. Nas execuções 24 e 26, os valores de *Recall* foram 95% e os valores de *F1_score* foram 84% e 89% respectivamente.

Em relação aos hiperparâmetros da segunda execução do *Grid Search*, todos os valores testados para os parâmetros do modelo apareceram nas melhores execuções como demonstrado na Tabela XV.

4) KNN: Por fim, o algoritmo *K-Nearest Neighbors* foi testado. Na primeira execução do *Grid Search*, foram testadas 64 combinações diferentes de hiperparâmetros para o modelo. As 10 melhores combinações de hiperparâmetros e seus respectivos resultados estão demonstrados na Tabela XVI.

Tabela XVI:
1º *GRIDSEARCH KNN*

Hiperparâmetros	*Recall*	*F1 Score*	*Accuracy*
algorithm : brute metric : manhattan n_neighbors : 5 weights : uniform	0.9340	0.8871	0.8812
algorithm : brute metric : euclidean n_neighbors : 5 weights : uniform	0.9340	0.8871	0.8812
algorithm : kd_tree metric : manhattan n_neighbors : 5 weights : uniform	0.9329	0.8868	0.8810
algorithm : auto metric : euclidean n_neighbors : 5 weights : uniform	0.9329	0.8868	0.8810
algorithm : kd_tree metric : euclidean n_neighbors : 5 weights : uniform	0.9329	0.8868	0.8810
algorithm : auto metric : manhattan n_neighbors : 5 weights : uniform	0.9329	0.8868	0.8810
algorithm : ball_tree metric : euclidean n_neighbors : 5 weights : uniform	0.9326	0.8866	0.8808
algorithm : ball_tree metric : manhattan n_neighbors : 5 weights : uniform	0.9326	0.8866	0.8808
algorithm : brute metric : manhattan n_neighbors : 5 weights : distance	0.9320	0.8862	0.8803
algorithm : brute metric : euclidean n_neighbors : 5 weights : distance	0.9320	0.8862	0.8803

Fonte: o autor (2022)

Como pode ser visto na Tabela XVI, as duas melhores combinações de hiperparâmetros obtiveram os mesmos resultados para *recall* e *F1_score*. A partir da terceira combinação, os valores de *Recall* diminuíram de 93,40% para 93,20%, porém os valores de *F1_score* subiram de 88,62% para 88,71%.

Como nos 10 melhores valores para *n_neighbors* apareceu somente o valor "5", as fronteiras foram alteradas para valores próximos de 5 (3, 4, 5, 6, 7). Como nas 8 melhores combinações o valor de *wights* foi "*uniform*", o valor "*distance*" foi removido da segunda execução do *Grid Search*. Para os demais parâmetros, nada foi alterado, pois todas as combinações aparecem nas melhores combinações de hiperparâmetros.

Portanto, foram testadas mais 40 combinações diferentes para o algoritmo *K-Nearest Neighbors*. As melhores combinações de hiperparâmetros da segunda execução do *Grid Search* e seus respectivos resultados estão demonstrados na Tabela XVII.

Tabela XVII:
2º *GRIDSEARCH KNN*

Hiperparâmetros	*Recall*	*F1 Score*	*Accuracy*
algorithm : brute metric : manhattan n_neighbors : 5 weights : uniform	0.9340	0.8871	0.8812
algorithm : brute metric : euclidean n_neighbors : 5 weights : uniform	0.9340	0.8871	0.8812
algorithm : ball_tree metric : euclidean n_neighbors : 3 weights : uniform	0.9340	0.8875	0.8812
algorithm : ball_tree metric : manhattan n_neighbors : 3 weights : uniform	0.9340	0.8875	0.8812
algorithm : auto metric : euclidean n_neighbors : 3 weights : uniform	0.9334	0.8874	0.8811
algorithm : kd_tree metric : manhattan n_neighbors : 3 weights : uniform	0.9334	0.8874	0.8811
algorithm : kd_tree metric : euclidean n_neighbors : 3 weights : uniform	0.9334	0.8874	0.8811
algorithm : auto metric : manhattan n_neighbors : 3 weights : uniform	0.9334	0.8874	0.8811
algorithm : auto metric : manhattan n_neighbors : 5 weights : uniform	0.9330	0.8868	0.8810
algorithm : auto metric : euclidean n_neighbors : 5 weights : uniform	0.9330	0.8868	0.8810

Fonte: o autor (2022)

Como pode ser visto na Tabela XVII, as duas melhores

combinações de hiperparâmetros permaneceram as mesmas da primeira execução do *Grid Search*. Contudo, as combinações que obtiveram o melhor resultado de *F1_score* foram a 3ª e a 4ª combinações melhores, com valores de *n_neighbors* igual a "3", e *algotithm* igual a "*ball_tree*". Da mesma forma das melhores combinações de *Recall*, os valores de *metric* variaram entre "*manhattan*"e "*euclidean*".

B. Cross Validation Repeated KFold

1) Random Forest: Primeiramente, foi validado a melhor combinação de hiperparâmetros da primeira execução do *Grid Search*, isto é, o parâmetro *criterion* recebeu o valor "*gini*", o parâmetro *max_features* recebeu o valor "*sqrt*"e por fim o parâmetro *n_estimators* recebeu o valor "100". Os resultados dessa combinação de hiperparâmetro estão demonstrados na Tabela XVIII.

Tabela XVIII:
1º *COMBINAÇÃO RF*

Métricas	Valores
Acurácia	0.8819
Recall	0.8315
F1_Score	0.8756
Precisão	0.9247
Especificidade	0.9323

Fonte: o autor (2022)

Posteriormente, foi validado a melhor combinação de hiperparâmetros da segunda execução do *Grid Search*, isto é, o parâmetro *criterion* recebeu o valor "*log_loss*", o parâmetro *max_features* recebeu o valor "*log2*"e por último o parâmetro *n_estimators* recebeu o valor "150". Os resultados dessa combinação de hiperparâmetro estão demonstrados na Tabela XIX.

Tabela XIX:
2º *COMBINAÇÃO RF*

Métricas	Valores
Acurácia	0.8819
Recall	0.8313
F1_Score	0.8756
Precisão	0.9249
Especificidade	0.9325

Fonte: o autor (2022)

Por fim, foi validado a segunda melhor combinação de hiperparâmetros da segunda execução do *Grid Search*, isto é, o parâmetro *criterion* recebeu o valor "*gini*", o parâmetro *max_features* recebeu o valor "*log2*" e por fim o parâmetro *n_estimators* recebeu o valor "150". Os resultados dessa combinação de hiperparâmetro estão demonstrados na Tabela XX.

2) SVM: Primeiramente, foi validado a melhor combinação de hiperparâmetros da primeira execução do *Grid Search*, isto é, o parâmetro *C* recebeu o valor "*0.01*", o parâmetro *degree* recebeu o valor "2", o parâmetro *gamma* recebeu o valor "0.0005" e por fim o parâmetro *kernel* recebeu o valor

Tabela XX:
3º *COMBINAÇÃO RF*

Métricas	Valores
Acurácia	0.8818
Recall	0.8818
F1_Score	0.8815
Precisão	0.8857
Especificidade	0.9324

Fonte: o autor (2022)

"*poly*". Os resultados dessa combinação de hiperparâmetro estão demonstrados na Tabela XXI.

Tabela XXI:
1º *COMBINAÇÃO SVM*

Métricas	Valores
Acurácia	0.4942
Recall	0.54
F1_Score	0.5586
Precisão	0.5736
Especificidade	0.46

Fonte: o autor (2022)

Posteriormente, foi validada a melhor combinação de hiperparâmetros da segunda execução do *Grid Search*, isto é, o parâmetro *C* recebeu o valor "*0.008*", o parâmetro *degree* recebeu o valor "2", o parâmetro *gamma* recebeu o valor "0.0001" e por fim o parâmetro *kernel* recebeu o valor "*poly*". Os resultados dessa combinação de hiperparâmetro estão demonstrados na Tabela XXII.

Tabela XXII:
2º *COMBINAÇÃO SVM*

Métricas	Valores
Acurácia	0.49
Recall	0.54
F1_Score	0.6619
Precisão	0.4946
Especificidade	0.46

Fonte: o autor (2022)

Por último, foi validado a décima sexta melhor combinação de hiperparâmetro da segunda execução do *Grid Search* pois foi a combinação que apresentou melhor resultado de *F1_score*. Nesta execução o parâmetro *C* recebeu o valor "*0.008*", o parâmetro *degree* recebeu o valor "1", o parâmetro *gamma* recebeu o valor "0.0001" e por fim o parâmetro *kernel* recebeu o valor "*sigmoid*". Os resultados dessa combinação de hiperparâmetros estão demonstrados na Tabela XXIII.

3) MLP: Primeiramente, foi validado a melhor combinação de hiperparâmetros da primeira execução do *Grid Search*, isto é, o parâmetro *activation* recebeu o valor "*logistic*", o parâmetro *alpha* recebeu o valor "0.001", o parâmetro *hidden_layer_sizes* recebeu o valor "(10, 30, 10)", o parâmetro *learning_rate* recebeu o valor *invscaling* e por fim o parâmetro *solver* recebeu o valor "*sgd*". Os resultados dessa combinação de hiperparâmetros estão demonstrados na Tabela XXIV.

Tabela XXIII:
3º *COMBINAÇÃO SVM*

Métricas	Valores
Acurácia	0.49
Recall	0.54
F1_Score	0.6619
Precisão	0.4946
Especificidade	0.46

Fonte: o autor (2022)

Tabela XXIV:
1º *COMBINAÇÃO MLP*

Métricas	Valores
Acurácia	0.4982
Recall	0.5145
F1_Score	0.5991
Precisão	0.5022
Especificidade	0.4851

Fonte: o autor (2022)

Posteriormente, foi validado a melhor combinação de hiperparâmetros da segunda execução do *Grid Search*, isto é, o parâmetro *activation* recebeu o valor "*relu*", o parâmetro *alpha* recebeu o valor "0.001", o parâmetro *hidden_layer_sizes* recebeu o valor "(3, 3, 2)", o parâmetro *learning_rate* recebeu o valor "*invscaling*" e por fim o parâmetro *solver* recebeu o valor "*sgd*". Os resultados dessa combinação de hiperparâmetros estão demonstrados na Tabela XXV.

Tabela XXV:
2º *COMBINAÇÃO MLP*

Métricas	Valores
Acurácia	0.6190
Recall	0.5495
F1_Score	0.6628
Precisão	0.5023
Especificidade	0.6850

Fonte: o autor (2022)

Por fim, foi validado a vigésima sexta combinação de hiperparâmetro da segunda execução do *Grid Search* pois dentre as melhores combinações apresentadas na Tabela XV, é a combinação que apresentou maior valor de *F1_Score*. Nesta combinação o parâmetro *activation* recebeu o valor "*relu*", o parâmetro *alpha* recebeu o valor "0.001", o parâmetro *hidden_layer_sizes* recebeu o valor "(3, 3, 3)", o parâmetro *learning_rate* recebeu o valor "*constant*" e por fim o parâmetro *solver* recebeu o valor "*sgd*". Os resultados dessa combinação de hiperparâmetros estão demonstrados na Tabela XXVI.

4) KNN: Primeiramente, foi validado a melhor combinação de hiperparâmetros da primeira execução do *Grid Search*, isto é, o parâmetro *algorithm* recebeu o valor "*brute*", o parâmetro *metric* recebeu o valor "*manhattan*", o parâmetro *n_neighbors* recebeu o valor "5" e por fim o parâmetro *weights* recebeu o valor "*uniform*". Os resultados dessa combinação de hiperparâmetros estão demonstrados na Tabela XXVI.

Tabela XXVI:
3º *COMBINAÇÃO MLP*

Métricas	Valores
Acurácia	0.8140
Recall	0.7934
F1_Score	0.7988
Precisão	0.8858
Especificidade	0.8361

Fonte: o autor (2022)

Tabela XXVII:
1º *COMBINAÇÃO KNN*

Métricas	Valores
Acurácia	0.5685
Recall	0.5685
F1_Score	0.4794
Precisão	0.7168
Especificidade	0.9821

Fonte: o autor (2022)

Posteriormente, foi validado a terceira melhor combinação de hiperparâmetros da segunda execução do *Grid Search* pois foi a combinação que obteve o maior valor de *F1_score*. Nesta execução o parâmetro *algorithm* recebeu o valor "*ball_tree*", o parâmetro *metric* recebeu o valor "*euclidean*", o parâmetro *n_neighbors* recebeu o valor "3" e por fim o parâmetro *weights* recebeu o valor "*uniform*". Os resultados dessa combinação de hiperparâmetros estão demonstrados na Tabela XXVIII.

Tabela XXVIII:
2º *COMBINAÇÃO KNN*

Métricas	Valores
Acurácia	0.5649
Recall	0.5650
F1_Score	0.4729
Precisão	0.7151
Especificidade	0.9827

Fonte: o autor (2022)

Por fim, foi validado a quinta melhor combinação de hiperparâmetro da segunda execução do *Grid Search* pois dentre as melhores combinações apresentadas na Tabela XVII, é a combinação que apresentou o segundo melhor valor de *Recall*. Nesta combinação o parâmetro *algorithm* recebeu o valor "*auto*", o parâmetro *metric* recebeu o valor "*euclidean*", o parâmetro *n_neighbors* recebeu o valor "3" e por fim o parâmetro *weights* recebeu o valor "*uniform*". Os resultados dessa combinação de hiperparâmetros estão demonstrados na Tabela XXIV.

C. Resultados

Como visto nos resultados da subseção *Cross Validation Repeated KFold*, todas as combinações de hiperparâmetros de todos os modelos obtiveram um resultado inferior ao esperado na subseção *Grid Search*.

Na Tabela XXX, pode ser visto de forma consolidada, em ordem decrescente, um comparativo dos resultados alcançados

Tabela XXIX:
3º *COMBINAÇÃO KNN*

Métricas	Valores
Acurácia	0.5649
Recall	0.5650
F1_Score	0.4729
Precisão	0.7151
Especificidade	0.9827

Fonte: o autor (2022)

na etapa do *Cross Validation Repeated KFold* com base na métrica *Recall*.

Tabela XXX:
COMPARAÇÃO DOS RESULTADOS CV

Execução	Algoritmo	*Recall*	*F1_score*	*Accuracy*
3	Random Forest	88.18%	88.15%	88.18%
1	Random Forest	83.15%	87.56%	88.19%
2	Random Forest	83.13%	87.56%	88.19%
3	MLP	79.34%	79.88%	81.40%
1	KNN	56.85%	47.94%	56.85%
2	KNN	56.50%	47.29%	56.49%
3	KNN	56.50%	47.29%	56.49%
2	MLP	54.95%	66.28%	61.90%
1	SVM	54%	55.86%	49.42%
2	SVM	54%	66.19%	49%
3	SVM	54%	66.19%	49%
1	MLP	51.45%	59.91%	49.82%

Fonte: o autor (2022)

Segundo as métricas estabelecidas, isto é, *F1_Score* e *Recall*, o melhor classificador encontrado foi a terceira combinação de hiperparâmetros testados do *Random Forest* obtendo um valor de 88.18% de *Recall* e 88.15% de *F1_Score*, como mostrado na Tabela XXX.

Os outros dois melhores modelos encontrados foram a primeira e a segunda combinação de hiperparâmetros do algoritmo *Random Forest* que obtiveram um *Recall* de 83.15% e 83.13% respectivamente. Os valores de *F1_Score* para essas duas combinações de hiperparâmetros foi 87.56% que por sua vez também foi o melhor valor de *F1_Score* encontrado depois do melhor modelo.

Em relação aos demais modelos (*KNN*, *MLP* e *SVM*), todos diminuíram seus resultados em comparação a etapa de *Grid Search*, dentre esses os melhores resultados obtidos foram na terceira combinação de hiperparâmetros do *MLP*, com *F1_score* de 79.88% e *Recall* de 79.34%.

Os classificadores que obtiveram os piores resultados foram o classificador gerado a partir da primeira combinação de hiperparâmetros do algoritmo *MLP*, e a segunda e terceira combinação de hiperparâmetros do algoritmo *SVM*. Essas combinações ficaram com uma valor menor que 55% de *Recall* e um valor inferior a 67% de *F1_score*.

V. Considerações Finais

Este trabalho teve como objetivo comparar os resultados de 4 modelos de aprendizado de máquina que tinham por sua vez o objetivo de classificar a avaliação de produto em positivas ou negativas. Os classificadores gerados nesta análise têm a função de enriquecer de dados de avaliação de produtos de *e-commerce*. Os modelos escolhidos para essa análise foram *Random Forest*, *Support Vector Machine*, *Multilayer perceptron* e o *K-Nearest Neighbors*.

Com base nos experimentos realizados foi possível alcançar dois dos três objetivos específicos propostos neste artigo, isto é, treinar três modelos de aprendizado de máquina distintos e comparar seus resultados.

Além disso, foi possível também alcançar o terceiro objetivo específico que consiste em obter um *F1_Score* e um *Recall* superior a 85%. Apesar de grande parte dos hiperparâmetros testados na etapa de *Grid Search* terem alcançado este objetivo nos três modelos testados, apenas a terceira combinação de hiperparâmetros do algoritmo *Random Forest* alcançou este objetivo após a etapa de *Cross Validation Repeated KFold*. Os melhores resultados obtidos foram 88.18% de *Recall* e 88.15% de *F1_Score*.

Ficou claro neste trabalho a importância da técnica de validação cruzada, pois antes da aplicação desta técnica, grande parte das combinações de hiperparâmetros testados obtiveram altos valores de *Recall* e *F1_Score*. Os resultados diminuíram após os classificadores passarem por outra técnica de validação.

A. Trabalhos Futuros

Pensando em trabalhos futuros, existem diversas possibilidades a serem executadas a fim de melhorar os resultados obtidos neste artigo.

Uma dessas possibilidades seria treinar outros modelos de aprendizado de máquina além dos 4 modelos propostos. Outra possibilidade é aumentar a quantidade de testes de hiperparâmetros na etapa do *Grid Search* que foram poucos explorados devido a limitações de tempo de execução.

A fim de expandir o trabalho atual, também é possível aumentar a base de treinamento com mais dados de *reviews* de produtos de outros *e-commerces*. Na mesma linha, ainda é possível explorar outras características de avaliação de produto, como comentários dos clientes em relação a sua experiência com o produto. Além da utilização das características provenientes das bases de avaliação de produto, ainda há a possibilidade da criação de novas características.

Por fim, outra forma de dar continuidade para este trabalho é desenvolver de fato um classificador e implementar este classificador em alguma etapa no fluxo de uma empresa. Esta etapa pode ser na ingestão desses dados de *reviews* no banco de dados ou também em uma etapa secundária de transformação desses dados. Outra possibilidade é desenvolver uma aplicação que quando acontece o envio de uma ou mais *reviews* de produto não classificadas, a aplicação devolve essas *reviews* com os valores classificados pelo modelo escolhido.

Referências

[1] ALVES, W. A. L.; ARAÚJO, S. A. D.; LIBRANTZ, A. F. H. Reconhecimento de padrões de texturas em imagens digitais usando uma rede neural artificial híbrida. Exacta, v. 4, n. 2, p. 325–332, 2006.

[2] AMRANE, M. et al. Breast cancer classification using machine learning. [S.l.], 2018. 1–4 p.

[3] ANJUM, B.; TIWARI, R. Economic and social impacts of e-commerce. Anjum, B., Tiwari, 2011.

[4] BAHULEYAN, H. Music genre classification using machine learning techniques. arXiv preprint arXiv:1804.01149, 2018.

[5] BANERJEE, S.; BHATTACHARYYA, S.; BOSE, I. Whose online reviews to trust? understanding reviewer trustworthiness and its impact on business. Decision Support Systems, v. 96, p. 17–26, 2017. ISSN 0167-9236. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0167923617300155⟩.

[6] BREIMAN, L. Random forests. Machine learning, Springer, v. 45, n. 1, p. 5–32, 2001.

[7] CHANNELADVISOR. 2021 E-Commerce Consumer Survey Results. 2021. Dispon ıvel em: ⟨https://www.channeladvisor.com/about/news-events/press-releases/channeladvisor-announces-2021-e-commerce-consumer-survey-results/⟩.

[8] CHARBUTY, B.; ABDULAZEEZ, A. Classification based on decision tree algorithm for machine learning. Journal of Applied Science and Technology Trends, v. 2, n. 01, p. 20–28, 2021.

[9] CHINNAMGARI, S. K. R Machine Learning Projects: Implement supervised, unsupervised, and reinforcement learning techniques using R 3.5. [S.l.]: Packt Publishing Ltd, 2019.

[10] DONGES, N. Random Forest Classifier: A Complete Guide to How It Works in Machine Learning. 2018. Acesso em: 24 out. 2022. Disponível em: ⟨https://builtin.com/data-science/random-forest-algorithm⟩.

[11] ESCOVEDO, T.; KOSHIYAMA, A. Introdução a Data Science: Algoritmos de Machine Learning e métodos de análise. [S.l.]: Casa do Código, 2020.

[12] FERREIRA, A. et al. Um estudo sobre previsão da demanda de encomendas utilizando uma rede neural artificial. Blucher Marine Engineering Proceedings, v. 2, n. 1, p. 353–364, 2016.

[13] FINE, S.; SCHEINBERG, K. Efficient svm training using low-rank kernel representations. Journal of Machine Learning Research, v. 2, n. Dec, p.243–264, 2001.

[14] FRIEDL, M. A.; BRODLEY, C. E. Decision tree classification of land cover from remotely sensed data. Remote sensing of environment, Elsevier, v. 61, n. 3, p. 399–409, 1997.

[15] GARDNER, M. W.; DORLING, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmospheric environment, Elsevier, v. 32, n. 14-15, p. 2627–2636, 1998.

[16] GÉRON, A. Mãos à Obra: Aprendizado de Máquina com Scikit-Learn TensorFlow. [S.l.]: Alta Books, 2019.

[17] GUIDE, E. Top 10 Ecommerce Sites in Brazil. 2018. Acesso em: 24 out. 2022. Disponívvel em ⟨https://ecommerceguide.com/top/top-10-ecommerce-sites-in-brazil/⟩.

[18] HAYKIN, S. Redes neurais: princípios e prática. [S.l.]: Bookman Editora, 2001.

[19] IZBICKI, R.; SANTOS, T. M. dos. Aprendizado de máquina: uma abordagem estatística. [S.l.]: Rafael Izbicki, 2020.

[20] JÍLKOV A, P.; KRÁLOV A, P. Digital consumer behaviour and ecommerce trends during the covid-19 crisis. International Advances in Economic Research, Springer, v. 27, n. 1, p. 83–85, 2021.

[21] KIM, R. Y. The impact of covid-19 on consumers: Preparing for digital sales. IEEE Engineering Management Review, v. 48, n. 3, p. 212–218, 2020.

[22] KITUKUTHA, N. M.; VASA, L.; OLÁH, J. The impact of covid-19 on the economy and sustainable e-commerce. In: Forum Scientiae Oeconomia. [S.l.: s.n.], 2021. v. 9, n. 2, p. 47–72.

[23] LACKERMAIR, G.; KAILER, D.; KANMAZ, K. Importance of online product reviews from a consumer's perspective. Advances in economics and business, v. 1, n. 1, p. 1–5, 2013.

[24] LEE, E.J.; SHIN, S. Y. When do consumers buy online product reviews? effects of review quality, product type, and reviewer's photo. Computers in human behavior, Elsevier, v. 31, p. 356–366, 2014.

[25] LIU, Y. et al. Uso de rede neural percéptron multi-camadas na classificacão de patologias cardíacas. Trends in Computational and Applied Mathematics, v. 9, n. 2, p. 255–264, 2008.

[26] LORENA, A. C.; CARVALHO, A. C. de. Introdução as máquinas de vetores suporte. Sao Carlos-SP, 2003.

[27] MOHAMMED, M.; KHAN, M. B.; BASHIER, E. B. M. Machine learning: algorithms and applications. [S.l.]: Crc Press, 2016.

[28] MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. Sistemas inteligentes-Fundamentos e aplicações, Manole, v. 1, n. 1, p. 32, 2003.

[29] MOREIRA, K. D. S.; PENEDO, A. S. T. Seleção de portifolios: uma análise comparativa dos cinco fatores de fama e french e redes neurais artificiais. Enfoque: Reflexão Contábil, Universidade Estadual de Maringá, v. 37, n. 2, p. 141–155, 2018.

[30] NASIR, N. et al. Water quality classification using machine learning algorithms. Journal of Water Process Engineering, Elsevier, v. 48, p. 102920, 2022.

[31] OLIVEIRA, B. A. S. et al. Avaliação de uma rede neural artificial como estimador temporal pluviométrico no sistema de abastecimento cantareira. Revista de Informática Aplicada, v. 14, n. 1, 2018.

[32] PENNA, M. L. F. Rede neural artificial para detecção de sobremortalidade atribuível à cólera no ceará. Revista de saude publica, SciELO Public Health, v. 38, n. 3, p. 351–357, 2004.

[33] PRONI, C. Desenvolvimento de modelos neurais e florestas aleatórias para estudo de tintas da indústria gráfica. Universidade Estadual Paulista (Unesp), 2022.

[34] RETAILER, W. The World's Top Online Marketplaces 2022. 2022. Acesso em: 24 out. 2022. Disponível em: ⟨https://www.webretailer.com/marketplaces-worldwide/online-marketplaces/⟩.

[35] SARKER, I. H. Machine learning: Algorithms, real-world applications and research directions. SN Computer Science, Springer, v. 2, n. 3, p. 1–21, 2021.

[36] SCIKIT-LEARN. Cross-validation: evaluating estimator performance. 2022. Acesso em: 10 out. 2022. Disponível em: ⟨https://scikit-learn.org/stable/modules/cross validation.html⟩.

[37] SILVA, R. F. d. O aumento do comércio eletrônico em tempos de pandemia: uma análise da evolução das relações de consumo online. 2021. Disponível em: ⟨https://bibliodigital.unijui.edu.br:8443/xmlui/handle/123456789/7421⟩.

[38] SOOFI, A. A.; AWAN, A. Classification techniques in machine learning: applications and issues. Journal of Basic Applied Sciences, v. 13, p.459–465, 2017.

[39] TSAGKIAS, M. et al. Challenges and research opportunities in ecommerce search and recommendations. In: ACM NEW YORK, NY, USA. ACM SIGIR Forum. [S.l.], 2021. v. 54, n. 1, p. 1–23.

[40] ZHANG, Y. New advances in machine learning. [S.l.]: BoD–Books on Demand, 2010.

[41] ZHOU, H.; HE, Y. Comparative study of okr and kpi. In: 2018 International Conference On E-Commerce And Contemporary Economic Development (Eced 2018), DEStech Transactions on Economics Business and Management. [S.l.: s.n.], 2018.

[42] ZHOU, M.; LIU, M.; TANG, D. Do the characteristics of online consumer reviews bias buyers' purchase intention and product perception?

[43] MAY, P. The business of ecommerce: From corporate strategy to technology. [S.l.]: Cambridge University Press, 2000. v. 1

[44] PANG, Bo; LEE, Lillian; VAITHYANATHAN, Shivakumar. Thumbs up? Sentiment classification using machine learning techniques. arXiv preprint cs/0205070, 2002.

[45] K. Thirunavukkarasu, A. S. Singh, P. Rai and S. Gupta, "Classification of IRIS Dataset using Classification Based KNN Algorithm in Supervised Learning,"2018 4th International Conference on Computing Communication and Automation (ICCCA), 2018, pp. 1-4, doi: 10.1109/CCAA.2018.8777643.

[46] T. Cover and P. Hart, "Nearest neighbor pattern classification,"in IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21-27, January 1967, doi: 10.1109/TIT.1967.1053964.

[47] JOSÉ, I. KNN (K-Nearest Neighbors) 1. 2018. Dispon ıvel em: https://medium.com/brasil-ai/knn-k-nearest-neighbors-1-e140c82e9c4e.

[48] MURUGAN, A.; NAIR, S. Anu H.; KUMAR, K. P. Detection of skin cancer using SVM, random forest and kNN classifiers. Journal of medical systems, v. 43, n. 8, p. 1-9, 2019.

[49] DINO, Hivi Ismat; ABDULRAZZAQ, Maiwan Bahjat. Facial expression classification based on SVM, KNN and MLP classifiers. In: 2019 International Conference on Advanced Science and Engineering (ICOASE). IEEE, 2019. p. 70-75.

[50] DEY, Lopamudra et al. Sentiment analysis of review datasets using naive bayes and k-nn classifier. arXiv preprint arXiv:1610.09982, 2016.

[51] SCIKIT-LEARN. Metrics. 2022. Acesso em: 10 out. 2022. Disponível em: ⟨https://scikit-learn.org/stable/search.html?q=metrics⟩

[52] CUI, Hang; MITTAL, Vibhu; DATAR, Mayur. Comparative experiments on sentiment classification for online product reviews. In: AAAI. 2006. p. 30.
[53] FAYAZ, Muhammad et al. Ensemble machine learning model for classification of spam product reviews. Complexity, v. 2020, 2020.
[54] SAGAR, Shuvashish Paul et al. PRCMLA: Product Review Classification Using Machine Learning Algorithms. In: Proceedings of International Conference on Trends in Computational and Cognitive Engineering. Springer, Singapore, 2021. p. 65-75.
[55] BOSCH, Anna; ZISSERMAN, Andrew; MUNOZ, Xavier. Image classification using random forests and ferns. In: 2007 IEEE 11th international conference on computer vision. Ieee, 2007. p. 1-8.
[56] VIJIYAKUMAR, K. et al. Random forest algorithm for the prediction of diabetes. In: 2019 IEEE international conference on system, computation, automation and networking (ICSCAN). IEEE, 2019. p. 1-5.
[57] PANKAJA, K.; SUMA, V. Plant leaf recognition and classification based on the whale optimization algorithm (WOA) and random forest (RF). Journal of The Institution of Engineers (India): Series B, v. 101, n. 5, p. 597-607, 2020.
[58] JOACHIMS, Thorsten. Text categorization with support vector machines: Learning with many relevant features. In: European conference on machine learning. Springer, Berlin, Heidelberg, 1998. p. 137-142.
[59] REDDY, V. Kranthi Sai. Stock market prediction using machine learning. International Research Journal of Engineering and Technology (IRJET), v. 5, n. 10, p. 1033-1035, 2018.
[60] SINGH, Archana; KUMAR, Rakesh. Heart disease prediction using machine learning algorithms. In: 2020 international conference on electrical and electronics engineering (ICE3). IEEE, 2020. p. 452-457.
[61] NOSRATABADI, Saeed et al. Prediction of food production using machine learning algorithms of multilayer perceptron and ANFIS. Agriculture, v. 11, n. 5, p. 408, 2021.
[62] HASAN, Md Kamrul et al. Diabetes prediction using ensembling of different machine learning classifiers. IEEE Access, v. 8, p. 76516-76531, 2020.
[63] TEOH, T. T. et al. Anomaly detection in cyber security attacks on networks using MLP deep learning. In: 2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE). IEEE, 2018. p. 1-5.
[64] GERAÇÃO XYZ: características e como aprendem. Disponível em: https://beieducacao.com.br/geracoes-x-y-z-e-alfa-como-cada-uma-se-comporta-e-aprende. Acesso em: 8 dez. 2022.