Análise Comparativa entre os Algoritmos de Classificação SVM e Naive Bayes

Bruno Dezorzi
Faculdade Senac
Cascavel, PR, Brasil
brunodezorzi9@gmail.com

Roberta Vanessa Rojo Parcianello
Faculdade Senac
Cascavel, PR, Brasil
roberta.parcianello@docente.pr.senac.br

Matheus Raffael Simon
Faculdade Senac
Cascavel, PR, Brasil
matheus.simon@docente.pr.senac.br

***Abstract*—This article presents a comparative analysis of two widely used supervised learning algorithms for classification tasks: *Naive Bayes* and *Support Vector Machine (SVM)*. Experiments were conducted in *Python* programming language using the *Iris* dataset. The methodology included preprocessing steps (outlier removal, feature engineering, and normalization), model application, and performance evaluation through stratified 5-fold cross-validation, using accuracy as the main metric. Results showed that *GaussianNB* achieved perfect accuracy, while the SVM delivered consistent performance with an average accuracy of 96%. The comparison highlights not only performance differences but also the strengths and limitations of each approach in relation to the dataset's characteristics.**

***Keywords*—Machine Learning; Supervised Learning; Naive Bayes; SVM; Data Classification.**

***Resumo*—Este artigo apresenta uma análise comparativa entre dois algoritmos supervisionados amplamente utilizados em tarefas de classificação: *Naive Bayes* e *Support Vector Machine (SVM)*. Os experimentos foram conduzidos na linguagem de programação *Python* com o conjunto de dados *Iris*. O processo incluiu pré-processamento (remoção de *outliers*, engenharia de atributos e normalização), aplicação dos modelos e avaliação por validação cruzada estratificada em cinco dobras, com acurácia como principal métrica. Os resultados mostraram que o *Naive Bayes* obteve acurácia perfeita, enquanto o SVM apresentou desempenho consistente, com média de 96%. A comparação evidencia diferenças de desempenho, bem como vantagens e limitações de cada abordagem frente às características do conjunto analisado.**

***Palavras-chave*—Aprendizado de Máquina; Aprendizado Supervisionado; *Naive Bayes*; SVM; Classificação de Dados.**

I. Introdução

O aprendizado de máquina é um campo da inteligência artificial que permite que sistemas computacionais identifiquem padrões e tomem decisões com base em dados, sem a necessidade de programação explícita para cada tarefa. Essa abordagem se baseia na construção de modelos matemáticos que ajustam seus parâmetros a partir de exemplos, melhorando sua performance à medida que recebem mais informações, quanto mais informação e ajustes forem disponibilizados para o algoritmo, melhor sua eficácia. O aprendizado de máquina tem sido utilizado em áreas como reconhecimento de imagens, processamento de linguagem natural e sistemas de recomendação, o que mostra seu uso em diferentes contextos.

O objetivo deste artigo é introduzir o leitor à área de aprendizado de máquina, apresentando dois dos algoritmos mais utilizados por iniciantes: o *Support Vector Machine (SVM)* e o *Naive Bayes*. Além de abordar a base teórica sobre o que é aprendizado de máquina e seus principais tipos, o trabalho também busca demonstrar, na prática (link do repositório no Apêndice A), como esses algoritmos podem ser aplicados e comparados utilizando o conjunto de dados *Iris*. Dessa forma, a proposta é facilitar o entendimento dos conceitos e evidenciar os impactos de cada modelo durante o treinamento e a análise dos resultados.

II. Fundamentação Teórica

Aprendizado de máquina é o campo de estudo que possibilita aos computadores a habilidade de aprender sem explicitamente programá-los [1]. Os sistemas de aprendizado de máquina podem ser classificados de acordo com a quantidade e tipo de supervisão que recebem durante seu treinamento. Existem quatro categorias principais de aprendizado: supervisionado, não supervisionado, semi-supervisionado e aprendizado por reforço [2].

A. Tipos de Aprendizagem

Na aprendizagem supervisionada, busca-se estimar um resultado por meio do aprendizado a partir de exemplos previamente estabelecidos e identificados pelo algoritmo. Nesse caso, são fornecidas amostras de determinadas variáveis juntamente com seus resultados, também chamados de *labels*. Esse tipo de aprendizagem é denominado "supervisionado" porque durante o treinamento as respostas esperadas já estão presentes nos dados, permitindo que o modelo aprenda a associar entradas a saídas de forma orientada [3]. Os problemas são divididos

em Regressão ou Classificação. Exemplo: Utilizar classificação para determinar se um e-mail é um spam.

Com base nos itens de [4], alguns exemplos de algoritmos de aprendizado supervisionado incluem o K-ésimo vizinho mais próximo (*K-Nearest Neighbors - KNN*), *Naive Bayes*, Regressão Logística (*Logistic Regression*), Máquinas de Vetores de Suporte (*Support Vector Machine - SVM*), Árvore de Decisão (*Decision Tree*) e Redes Neurais (*Neural Networks*).

Diferentemente do aprendizado supervisionado, no aprendizado não supervisionado os dados de treinamento não são rotulados [2]. A tarefa desse formato de aprendizagem é aprender sozinho, rotulando os dados com base em sua semelhança e na identificação de padrões intrínsecos, sem a necessidade de um ”professor” ou rótulos previamente definidos.

Como citado em [5], alguns exemplos de algoritmos de aprendizado não supervisionado incluem Análise de Componentes Principais (*Principal Component Analysis - PCA*) e K-ésimo vizinho mais próximo (*K-Nearest Neighbors - KNN*).

Segundo [6], o aprendizado semi-supervisionado é uma abordagem específica para classificação. Enquanto classificadores tradicionais exigem dados rotulados (pares de características e rótulos) para serem treinados, esse processo de rotular as instâncias pode ser trabalhoso, caro e demorado, pois requer a atuação de especialistas. Conforme mencionado por [6], os dados não rotulados são mais fáceis de serem coletados, mas suas aplicações práticas ainda são limitadas.

O aprendizado semi-supervisionado, tal qual cita [6], resolve essa limitação ao combinar grandes volumes de dados não rotulados com dados rotulados, resultando em classificadores mais eficazes. De acordo com o autor, por demandar menos esforço humano e alcançar maior precisão, essa técnica tem se tornado uma área de grande interesse tanto na pesquisa quanto na aplicação prática.

Diante de um cenário onde a obtenção de dados e variáveis de interesse representam um processo custoso e demorado, precisa-se buscar alternativas para aproveitar as informações disponíveis. O aprendizado semi-supervisionado surge como uma alternativa promissora às abordagens de aprendizado supervisionado e não supervisionado. O objetivo é trabalhar em um cenário com problemas de rotulação parcial dos dados, onde as informações sobre a variável alvo são poucas [7].

No artigo de [7] é discorrido sobre as vantagens da utilização deste algoritmo em casos de detecção de fraudes em transações, sendo elas:

- Precisão aprimorada;
- Redução de falsos positivos;
- Aproveitamento de dados não rotulados;
- Adaptação a mudanças;
- Controle personalizado.

Com base nos itens de [7], alguns exemplos de algoritmos de aprendizado semi-supervisionado incluem o *SVM Transdutivo* (*Transductive SVM*), os *Modelos Generativos* (*Generative Models*) e o *Autotreinamento* (*Self-Training*).

O Aprendizado com Reforço tem o objetivo de melhorar o desempenho final, aumentar a acurácia na qual nenhuma entrada/saída é fornecida, mas sim *feedbacks* sobre as decisões como um meio de maximizar um sinal de recompensa, levando ao aprendizado de ações desejadas em determinados ambientes [8].

B. Classificação

Classificação é uma abordagem de mineração de dados (aprendizado de máquina) usada para prever a associação de grupos para instâncias de dados [9]. Embora haja uma variedade de técnicas disponíveis para aprendizado de máquina, a classificação é a técnica mais amplamente usada [10].

De acordo com [4], o modelo de aprendizado deve aprender (aproximar o comportamento de) uma função que mapeia um vetor (ou uma matriz) para uma das várias classes, analisando diversos exemplos de entrada e saída dessa função. O aprendizado de máquina indutivo é o processo de aprender um conjunto de regras a partir de instâncias (exemplos em um conjunto de treinamento) ou - de forma mais geral - criar um classificador que possa ser utilizado para generalizar a partir de novas instâncias. Isso significa que um classificador é um método de aprendizado supervisionado que utiliza dados rotulados para aprender e fazer previsões. Dados rotulados são registros que já possuem uma ou mais colunas com características distintas de interesse para quem realiza a análise. Cada registro nesses dados tem um rótulo associado que representa a categoria ou classe à qual pertence. O classificador usa essas informações para identificar padrões e prever a qual classe novos dados não rotulados pertencem.

Um exemplo clássico de dados rotulados é o conjunto de dados *Iris* [1] que contém amostras de flores com medições de suas características. Cada registro inclui informações sobre o comprimento e a largura das sépalas e pétalas, além da espécie correspondente indicada na última coluna. A tabela 1 apresenta 10 exemplos desses registros, destacando as características utilizadas para classificação das espécies.

O *Iris* é amplamente utilizado nos primeiros estudos em *machine learning*, pois está disponível na biblioteca *scikit-learn* e apresenta uma estrutura simples. Suas características bem definidas entre as classes tornam-no ideal para testar e comparar diferentes classificadores de forma intuitiva. Esse *dataset* serve como uma base para compreender o funcionamento dos algoritmos de aprendizado supervisionado na prática.

[1] O site do conjunto de dados se encontra no Apêndice A

TABELA I
Amostra com dez registros do conjunto de dados *Iris*

sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
6.1	2.8	4.7	1.2	versicolor
5.7	3.8	1.7	0.3	setosa
7.7	2.6	6.9	2.3	virginica
6.0	2.9	4.5	1.5	versicolor
6.8	2.8	4.8	1.4	versicolor
5.4	3.4	1.5	0.4	setosa
5.6	2.9	3.6	1.3	versicolor
6.9	3.1	5.1	2.3	virginica
6.2	2.2	4.5	1.5	versicolor
5.8	2.7	3.9	1.2	versicolor

Fonte: Adaptado de [11]

C. Os Classificadores

Neste artigo foram implementado dois algoritmos para realizar a comparação da eficácia: o algoritmo de *Naive Bayes* e *Support Vector Machine (SVM)*.

1) Naive Bayes: O *Naive Bayes* é um classificador linear baseado no teorema de *Bayes* que utiliza probabilidades condicionais para determinar a classe de um dado a partir da combinação entre a probabilidade a *priori* e a verossimilhança da classe. Esse algoritmo é conhecido como um ”aprendedor ansioso”, pois possui uma rápida capacidade de aprendizado, sendo amplamente utilizado para classificação de texto [12].

Entre os classificadores probabilísticos, ele pressupõe uma independência entre os atributos dos dados. É popular para aplicações de classificação de textos, por exemplo, para identificação de *spams*. Uma vantagem desse modelo é que, por supor uma independência entre os atributos, ele é capaz de fazer o treinamento de um modelo com um número pequeno de amostras, uma vez que o modelo não conseguirá capturar as interações entre os atributos. Esse modelo simples também pode trabalhar com dados que tenham vários atributos. Desse modo, serve como um bom modelo de base [13].

De acordo com [13], há três principais classes no *sklearn*, *GaussianNB*, *MultinomialNB* e *BernoulliNB*. A primeira pressupõe uma distribuição gaussiana (atributos contínuos com uma distribuição normal); a segunda é voltada para contagens discretas de ocorrências; e a terceira para atributos booleanos discretos

Segundo [14], o algoritmo *Naive Bayes* apresenta um desempenho eficiente em tarefas como classificação de e-mails, Processamento de Linguagem Natural (PLN) e detecção de anomalias devido à sua simplicidade e baixo custo computacional.

2) SVM (Support Vector Machine): Uma máquina de vetores de suporte é um modelo muito robusto e versátil de aprendizado de máquina, possuindo a capacidade de fazer classificações lineares ou não lineares, de regressão e até mesmo detecção de *outliers* e são particularmente adaptadas para a classificação de conjunto de dados complexos de pequeno ou médio porte [2].

Esse é um dos algoritmos mais efetivos para Classificação e que também pode ser utilizado para problemas de Regressão, apesar de menos comum. O *Support Vector Machine* pode ser aplicado em dados lineares ou não lineares. Apesar de o treinamento desses modelos costumam ser lentos, eles exigem poucos ajustes, tendem a apresentar boa acurácia e conseguem modelar fronteiras de decisão complexas e não lineares. Além disso, são menos propensos a *overfitting* se comparados com outros métodos [15].

Essencialmente, o SVM realiza um mapeamento não linear para transformar os dados de treino originais em uma dimensão maior. Nesta nova dimensão o algoritmo busca pelo hiperplano que separa os dados linearmente de forma ótima. Com um mapeamento apropriado para uma dimensão suficientemente alta, dados de duas classes podem ser sempre separados por um hiperplano. O SVM encontra este hiperplano usando vetores de suporte (exemplos essenciais para o treinamento) e margens definidas pelos vetores de suporte [15].

III. Trabalhos Relacionados

Diversos estudos aplicam técnicas de classificação ao conjunto de dados *Iris*, buscando avaliar o desempenho de diferentes algoritmos. [16] realizou um treinamento utilizando o algoritmo *SVM* onde separou 70% de dados para treinamento e 30% para teste. Utilizou *Principal Component Analysis* (PCA) para reduzir a dimensionalidade do dataset de quatro atributos para três. Seus resultados geraram dois valores: ao utilizar os parâmetros `c` e `gamma` o modelo obteve uma acurácia de 98,7%, enquanto sem esses parâmetros o resultado foi de 95,3%.

No estudo de [17], o autor usou os algoritmos *Naive Bayes* e *Decision Tree* aplicados ao conjunto de dados *Iris*. A base foi dividida em 67% para treinamento e 33% para teste. Ambos os modelos apresentaram métricas elevadas de desempenho, com destaque para o *Decision Tree*, que obteve acurácia de 100% no treinamento e 98% no teste, enquanto o *Naive Bayes* alcançou 95% e 96%, respectivamente. As métricas de precisão, revocação e *F1-score* permaneceram acima de 90% em todas as classes, indicando resultados consistentes.

No trabalho de [18], foi aplicada a metodologia de classificação baseada no algoritmo *Gaussian Naive Bayes*, utilizando o conjunto de dados *Iris* disponibilizado pelo repositório *UCI Machine Learning Repository*. Os autores estruturaram o experimento realizando a preparação dos dados, visualizações exploratórias e a divisão entre conjuntos de treino e teste e o treinamento utilizando o modelo *Naive Bayes Gaussiano*. Os resultados evidenciaram uma acurácia de aproximadamente 95%, demonstrando que o artigo adota

uma abordagem direta e didática, sem explorar o ajuste de hiperparâmetros.

Diferentemente desses trabalhos, o presente estudo realiza uma comparação direta entre *SVM* com diferentes *kernels* e *Naive Bayes*, incorporando etapas de limpeza de dados, seleção de atributos e ajuste de hiperparâmetros, com o objetivo de identificar o modelo mais eficiente para este cenário de classificação.

IV. Metodologia

Os experimentos foram conduzidos utilizando a linguagem de programação *Python* com as bibliotecas *pandas*, *scikit-learn*, *matplotlib* e *seaborn*. O *dataset* utilizado foi o *Iris Dataset*, que contém amostras de três espécies de flores: *setosa*, *versicolor* e *virginica*.

O pré-processamento dos dados incluiu:

- **Mapeamento das classes**: os rótulos numéricos foram convertidos em nomes de espécies para fins de análise exploratória e depois retornado para números com o objetivo de ajustar para o treinamento.
- **Tratamento de *outliers***: realizado por meio do método do Intervalo Interquartil (IQR), substituindo os valores considerados extremos pela média dos valores não *outliers* de cada atributo.
- **Engenharia de atributos**: criação de novas *features*, como a área da pétala (`petal_area`), a área da sépala (`sepal_area`), a proporção entre o comprimento e a largura da pétala (`petal_prop`) e da sépala (`sepal_prop`).
- **Normalização**: realizada com *StandardScaler*, ajustando todas as variáveis para média zero e desvio padrão um.

Antes da aplicação dos algoritmos de aprendizado de máquina, é essencial realizar uma etapa de análise exploratória e pré-processamento dos dados, com foco na detecção e remoção de valores discrepantes — os chamados *outliers*. Esses valores extremos podem distorcer as distribuições das variáveis e comprometer o desempenho dos classificadores.

A Fig.1 ilustra um gráfico do tipo *boxplot*, representando os dados em seu estado original, antes da aplicação de qualquer técnica de engenharia de atributos ou tratamento de *outliers*. Nota-se a presença de diversas observações fora dos limites interquartílicos (representadas pelos pontos fora das ”caixas”) o que evidência a necessidade de intervenção prévia.

A partir dessa análise inicial, foram aplicadas técnicas de engenharia de atributos, incluindo a criação das variáveis `petal_area`, `sepal_area`, `petal_prop` e `sepal_prop`, seguidas pela remoção de *outliers* com base em critérios estatísticos. O objetivo foi tratar a estrutura do conjunto de dados, garantindo maior eficiência para os modelos de classificação.

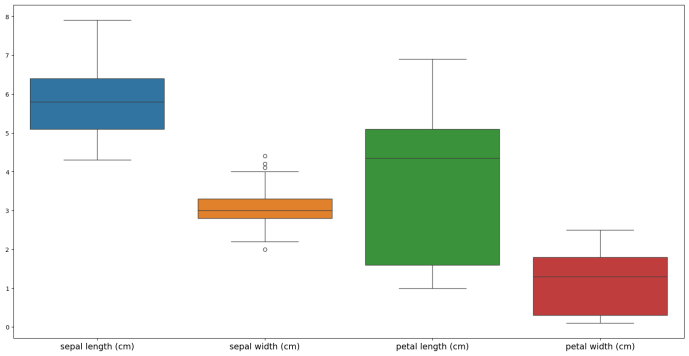

Fig. 1: Visualização do *boxplot* antes da engenharia de atributos e do tratamento de *outliers*.

Fonte: Elaborado pelos autores.

A Fig.2 apresenta o mesmo tipo de gráfico após essas etapas de tratamento. Observa-se a criação de novas *features* e a eliminação dos valores discrepantes o que indica um conjunto de dados mais limpo e preparado para a aplicação dos algoritmos de aprendizado de máquina. A engenharia de atributos se tornou necessária por conta da escassez de registros para o treinamento dos modelos, portanto, ocorreu a necessidade de ”criar” mais dados.

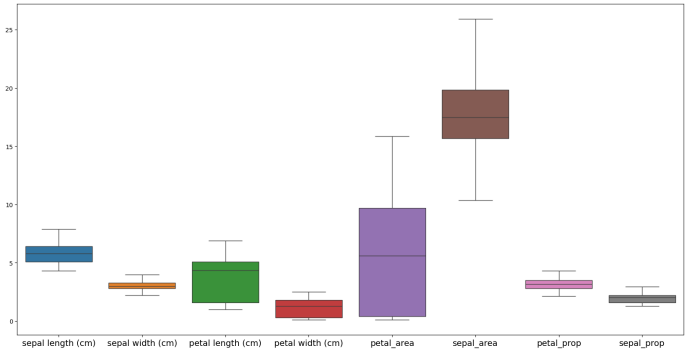

Fig. 2: Visualização do *boxplot* após a engenharia de atributos e a remoção de *outliers*.

Fonte: Elaborado pelos autores.

A base de dados foi dividida em três partes para garantir um bom treinamento e avaliação do modelo: inicialmente, 15% dos dados foram separados para o conjunto de teste que será utilizado para a avaliação final do desempenho. Dos 85% restantes, 15% foram reservados para o conjunto de validação, usado para ajustar hiperparâmetros e evitar *overfitting*. Isso resulta em aproximadamente 72,25% dos dados destinados ao treinamento do modelo, 12,75% para validação e 15% para

teste.

Os modelos principais foram:

- ***DummyClassifier***: configurado com a estratégia `most_frequent`, foi utilizado como modelo *baseline*.
- ***Support Vector Machine (SVM)***: foram testados os *kernels* `linear`, `poly`, `rbf` e `sigmoid`. O melhor desempenho foi obtido com o kernel `linear`, o qual foi posteriormente refinado com os parâmetros `C = 0.5` e `class_weight = 'balanced'`.
- ***Naive Bayes***: avaliou-se o *GaussianNB* e o *BernoulliNB*. O *GaussianNB* teve sua performance otimizada utilizando *GridSearchCV*, ajustando o parâmetro `var_smoothing` com valores logarítmicos entre 10^{-11} e 10^{-1}.

A validação dos modelos foi realizada com *cross-validation* de 5 *folds*, utilizando a acurácia como métrica principal.

V. Resultados e Discussão

Os resultados obtidos são apresentandos na Tabela II. A tabela apresenta a acurácia de cada classificador aplicado no conjunto de dados *Iris*.

TABELA II
Resultados de acurácia obtidos pelos classificadores avaliados no conjunto de dados *Iris*

Modelo	Acurácia (Validação)	Acurácia (Teste)
Dummy Classifier	0.35	0.26
SVM (linear)	0.95	0.96
SVM (poly)	0.88	0.88
SVM (rbf)	0.94	0.94
SVM (sigmoid)	0.94	0.94
Naive Bayes (GaussianNB)	1.00	1.00
Naive Bayes (BernoulliNB)	0.82	0.82
SVM (linear) ajustado	0.95	0.96

Fonte: Elaborado pelos autores

Podemos observar na Tabela II que:

- ***Dummy Classifier***: apresentou acurácia de 0.35 na validação e 0.26 no teste, evidenciando que estratégias baseadas em frequência simples não são adequadas para este conjunto de dados, afinal, ele é apenas um modelo de referência.
- ***SVM* com diferentes *kernels***:
  - `linear`: acurácia média de validação cruzada de 0.96 ($\pm$ 0.04), com 0.95 na validação e 0.96 no teste, demonstrando excelente desempenho consistente.
  - `poly`: acurácia média de 0.88 ($\pm$ 0.09), inferior ao linear, indicando que a maior complexidade não trouxe ganhos.
  - `rbf`: acurácia média de 0.94 ($\pm$ 0.03), robusta e estável.
  - `sigmoid`: acurácia média de 0.94 ($\pm$ 0.05), também consistente.
- ***Naive Bayes***:
  - *GaussianNB*: acurácia média de 0.93 ($\pm$ 0.02) na validação cruzada. Após ajuste do parâmetro `var_smoothing`, atingiu 1.00 na validação e no teste, sugerindo possível *overfitting*.
  - *BernoulliNB*: desempenho inferior, com acurácia de 0.82 ($\pm$ 0.05), mostrando menor adequação ao conjunto de dados.
- **SVM ajustado**: com *kernel* `linear`, `C = 0.5` e `class_weight = 'balanced'`, alcançou 0.95 na validação e 0.96 no teste, apresentando estabilidade e confiabilidade comparáveis ao GaussianNB ajustado.

Os valores apresentados com o símbolo $\pm$ representam o desvio padrão das acurácias obtidas nas diferentes divisões de validação cruzada *(5-fold cross-validation)*, indicando a consistência do modelo entre os subconjuntos dos dados.
Em resumo, apesar do GaussianNB ajustado alcançar acurácia perfeita, o SVM com *kernel* linear apresenta desempenho robusto e consistente, mostrando-se mais confiável para aplicação em dados reais, especialmente considerando o pequeno tamanho e baixa complexidade do Iris Dataset.

VI. Conclusão

Os experimentos realizados demonstraram que tanto o classificador *SVM* quanto o *Naive Bayes* são capazes de alcançar altos níveis de acurácia no *Iris Dataset*, especialmente após ajustes criteriosos de seus hiperparâmetros. Apesar de o *GaussianNB* ter atingido desempenho perfeito após otimização, esse resultado levanta suspeitas de *overfitting*, principalmente devido à simplicidade e ao tamanho reduzido do conjunto de dados.

O modelo *SVM* com kernel `linear`, por sua vez, apresentou resultados robustos e estáveis mesmo antes de ajustes finos, confirmando sua adequação para problemas lineares bem definidos, como é o caso do conjunto de dados utilizado. Sua consistência nos diferentes subconjuntos de validação cruzada indica maior generalização, tornando-o uma escolha confiável em contextos reais com dados similares.

Portanto, conclui-se que, embora o *GaussianNB* otimizado tenha alcançado os melhores números em termos absolutos, o *SVM linear* oferece um equilíbrio superior entre desempenho e estabilidade. Além disso, este trabalho reforça a importância do pré-processamento cuidadoso e da engenharia de atributos como etapas fundamentais para o sucesso de qualquer modelo de aprendizado de máquina, mesmo em bases de dados consideradas simples.

Em trabalhos futuros, pode haver novas oportunidades para explorar métricas adicionais de desempenho, realizar testes com

conjuntos de dados maiores e mais complexos e utilizar os parâmetros dos modelos de forma mais eficiente e elaborada, por meio de estudos sobre esses parâmetros, além de aprofundar o estudo nos algoritmos apresentados neste artigo e em outros algoritmos de aprendizado. Como apresentado, muitos desses parâmetros ainda não são totalmente compreendidos, o que representa uma oportunidade de expandir o conhecimento. Esse estudo mais detalhado sobre parâmetros, novos modelos e a construção de algoritmos permitirá aprimorar o desempenho dos modelos em futuras aplicações, além de proporcionar um entendimento mais profundo dos cálculos matemáticos e estatísticos pertencentes a esses algoritmos.

Agradecimentos

Gostaria de expressar minha gratidão a todos que me apoiaram e me orientaram durante o desenvolvimento deste artigo.

Agradeço aos meus orientadores, Profa. Roberta Parcianello e Prof. Matheus Raffael Simon pelo apoio e orientação.

Agradeço a minha faculdade por disponibilizar a iniciação científica como projeto de extensão.

E aos meus amigos, colegas e familiares por todo o apoio que foi oferecido durante a minha trajetória neste projeto.

Este trabalho não seria possível sem o apoio de todos vocês. Muito obrigado!

Referências

[1] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959.

[2] A. Géron, "Mãos à obra: aprendizado de máquina com scikit-learn," *Keras & TensorFlow: Conceitos, ferramentas e técnicas para a construção de sistemas inteligentes.*, 2021.

[3] M. J. W. Garzillo, "Classificação de tumores cerebrais com algoritmos de machine learning," Ph.D. dissertation, Instituto Politécnico de Lisboa, Escola Superior de Tecnologia da Saúde de Lisboa, 2022.

[4] F. Osisanwo, J. Akinsola, O. Awodele, J. Hinmikaiye, O. Olakanmi, J. Akinjobi *et al.*, "Supervised machine learning algorithms: classification and comparison," *International Journal of Computer Trends and Technology (IJCTT)*, vol. 48, no. 3, pp. 128–138, 2017.

[5] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, no. 1, pp. 381–386, 2020.

[6] X. J. Zhu, "Semi-supervised learning literature survey," *University of Wisconsin-Madison*, 2005.

[7] M. Almeida, "Machine learning: o que é aprendizado semi-supervisionado," https://www.alura.com.br/artigos/machine-learning-aprendizado-semi-supervisionado?srsltid=AfmBOorhmQOk9xfK_LzB6sE88sD2WMnRfJgY0loEOawtqq91jcTuReLe, 2023, acesso em: 20 nov. 2024.

[8] G. R. Schleder and A. Fazzio, "Machine learning na física, química, e ciência de materiais: Descoberta e design de materiais," *Revista Brasileira de Ensino de Física*, vol. 43, no. Suppl 1, p. e20200407, 2021.

[9] R. Tera, "O que é machine learning: o guia para entender aprendizado de máquina," https://blog.somostera.com/data-science/machine-learning, 2021, acesso em: 05 dez. 2024.

[10] F. M. Honda, H. and P. Yaohao, "Os três tipos de aprendizado de máquina," LAMFO - Laboratório de Aprendizado de Máquina em Finanças e Organizações, jul 2017, obtido em 10 de setembro de 2021.

[11] Scikit-learn, "scikit-learn: Machine learning in python," https://scikit-learn.org/stable/, 2024, acesso em: 23 ago. 2024.

[12] C. Naulak, "A comparative study of naive bayes classifiers with improved technique on text classification," *Authorea Preprints*, 2023.

[13] M. Harrison, *Machine Learning Pocket Reference*. O'Reilly Media, Inc., 2019.

[14] T. A. S. Pardo and M. d. G. V. Nunes, "Aprendizado bayesiano aplicado ao processamento de línguas naturais," 2002.

[15] T. Escovedo, *Introdução à Estatística para Ciência de Dados: Da exploração dos dados à experimentação contínua com exemplos de código em Python e R*. Aovs Sistemas de Informática Ltda., 2024.

[16] Z. F. Hussain, H. R. Ibraheem, M. Alsajri, A. H. Ali, M. A. Ismail, S. Kasim, and T. Sutikno, "A new model for iris data set classification based on linear support vector machine parameter's optimization," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 1, p. 1079, 2020.

[17] Y. Dani and M. A. Ginting, "Comparison of iris dataset classification with gaussian naïve bayes and decision tree algorithms." *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 14, no. 2, 2024.

[18] Z. Iqbal and M. Yadav, "Multiclass classification with iris dataset using gaussian naive bayes," *International Journal of Computer Science and Mobile Computing*, pp. 27–35, 2020.

Apêndice A
Conjunto de Dados e Repositório

O conjunto de dados utilizado neste trabalho é o *Iris Dataset*, disponível no site do *Scikit-learn*, acessível em: https://scikit-learn.org/1.4/auto_examples/datasets/plot_iris_dataset.html.

A versão tratada do conjunto de dados, juntamente com os códigos fonte, pode ser encontrada no repositório deste projeto: https://github.com/Bruno-Dezorzi/Iniciacao_Cientifica_ML_Classificacao_Senac.