MINISTÉRIO DA EDUCAÇÃO
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
DE GOIÁS - CÂMPUS GOIÂNIA

CURSO DE LICENCIATURA EM MATEMÁTICA

UTILIZAÇÃO DE APRENDIZADO DE MÁQUINA NA PREDIÇÃO DE DESEMPENHO ACADÊMICO

GABRIEL SILVA HAYNE

GOIÂNIA - GOIÁS
2025

TERMO DE AUTORIZAÇÃO PARA DISPONIBILIZAÇÃO NO REPOSITÓRIO DIGITAL DO IFG - ReDi IFG

Com base no disposto na Lei Federal nº 9.610/98, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia de Goiás, a disponibilizar gratuitamente o documento no Repositório Digital (ReDi IFG), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, em formato digital para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IFG.

Identificação da Produção Técnico-Científica

[ ] Tese	[ ] Artigo Científico
[ ] Dissertação	[ ] Capítulo de Livro
[ ] Monografia – Especialização	[ ] Livro
[X] TCC - Graduação	[ ] Trabalho Apresentado em Evento
[ ] Produto Técnico e Educacional - Tipo: _____	

Nome Completo do Autor: Gabriel Silva Hayne
Matrícula: 20192010940143
Título do Trabalho: UTILIZAÇÃO DE APRENDIZADO DE MÁQUINA NA PREDIÇÃO DE DESEMPENHO ACADÊMICO

Autorização - Marque uma das opções

1. (X) Autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso aberto);
2. ( ) Autorizo disponibilizar meu trabalho no Repositório Digital do IFG somente após a data ___/___/_____ (Embargo);
3. ( ) Não autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso restrito).

Ao indicar a opção **2 ou 3**, marque a justificativa:

( ) O documento está sujeito a registro de patente.
( ) O documento pode vir a ser publicado como livro, capítulo de livro ou artigo.
( ) Outra justificativa: _____

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O/A referido/a autor/a declara que:

i. o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;
ii. obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autor/a, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia de Goiás os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;
iii. cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia de Goiás.

Goiânia, 14 / 02 /2025.
Local Data

gov.br Documento assinado digitalmente
GABRIEL SILVA HAYNE
Data: 14/02/2025 15:10:36-0300
Verifique em https://validar.iti.gov.br

Assinatura do Autor e/ou Detentor dos Direitos Autorais

GABRIEL SILVA HAYNE

UTILIZAÇÃO DE APRENDIZADO DE MÁQUINA NA PREDIÇÃO DE DESEMPENHO ACADÊMICO

Trabalho de Conclusão de Curso apresentado ao Curso de Licenciatura em Matemática do Instituto Federal de Goiás - IFG - Câmpus Goiânia, como requisito parcial para obtenção do título de Licenciado em Matemática.

Área de Concentração: Matemática

Orientador: Professor Dr. Hugo Leonardo da Silva Belisário

GOIÂNIA - GOIÁS
2025

H332u Hayne, Gabriel Silva

Utilização de aprendizado de máquina na predição de desempenho acadêmico / Gabriel Silva Hayne. – Goiânia: Instituto Federal de Educação, Ciência e Tecnologia de Goiás, Câmpus Goiânia, 2025.

80 f.: il.

Orientador: Prof. Dr. Hugo Leonardo da Silva Belisário.

TCC (Trabalho de Conclusão de Curso) – Licenciatura em Matemática, Instituto Federal de Educação, Ciência e Tecnologia de Goiás, Câmpus Goiânia.

1. Aprendizado de máquina. 2. Python. 3. Otimização matemática. 4. Desempenho Acadêmico. I. Belisário, Hugo Leonardo da Silva (orientador). II. Instituto Federal de Educação, Ciência e Tecnologia de Goiás, Câmpus Goiânia. III. Título.

CDD 005.133

Ficha catalográfica elaborada pela Bibliotecária Karol Almeida da Silva CRB1/ 2.740
Biblioteca Professor Jorge Félix de Souza,
Instituto Federal de Educação, Ciência e Tecnologia de Goiás, Câmpus Goiânia.

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS
CÂMPUS GOIÂNIA

UTILIZAÇÃO DE APRENDIZADO DE MÁQUINA NA PREDIÇÃO DE DESEMPENHO ACADÊMICO

por

GABRIEL SILVA HAYNE

Trabalho de Conclusão de Curso apresentado ao Colegiado do Curso de Licenciatura em Matemática do Instituto Federal de Educação, Ciência e Tecnologia de Goiás – Câmpus Goiânia, como requisito parcial para a obtenção do Diploma de Licenciado em Matemática.

Área de concentração: Matemática Aplicada

Orientador: Prof. Dr. Hugo Leonardo da Silva Belisário

Trabalho de Conclusão de Curso defendido e aprovado, em 26 de fevereiro de 2025, pela banca examinadora constituída pelos seguintes membros:

(assinado eletronicamente)

Prof. Dr. Hugo Leonardo da Silva Belisário(Presidente da Banca/IFG/Câmpus Goiânia)

(assinado eletronicamente)

Prof. Dr. Marcelo Bezerra Barboza (UFG/IME-Goiânia)

(assinado eletronicamente)

Prof. Dr. Uender Barbosa de Souza (IFG/Câmpus Goiânia)

Documento assinado eletronicamente por:

- **Marcelo Bezerra Barboza, Marcelo Bezerra Barboza - Docente - Ufg (01567601000143)**, em 26/02/2025 16:29:45.
- **Uender Barbosa de Souza, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 26/02/2025 16:23:58.
- **Hugo Leonardo da Silva Belisario, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 26/02/2025 16:22:46.

Este documento foi emitido pelo SUAP em 19/02/2025. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifg.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 619785
Código de Autenticação: 7f503dab82

Dedicatória

Dedico este trabalho a todos os professores do corpo docente de Matemática do Instituto Federal de Goiás, Campus Goiânia. Suas habilidades pedagógicas, comprometimento com o ensino e dedicação aos alunos foram fontes constantes de inspiração ao longo desta jornada acadêmica. Cada um de vocês desempenhou um papel fundamental em meu desenvolvimento acadêmico e pessoal, e é com gratidão que reconheço o impacto significativo que tiveram em minha vida. Esta conquista é dedicada a vocês, que tanto me ensinaram e guiaram ao longo do caminho.

Agradecimentos

Gostaria de expressar minha profunda gratidão a todos os envolvidos em minha formação pedagógica e acadêmica.

Primeiramente, agradeço a Deus, que me concedeu força, sabedoria e orientação ao longo desta jornada. Sua presença constante foi uma fonte de inspiração e conforto em todos os momentos.

Agradeço à minha família pelo apoio incondicional, amor e incentivo ao longo dos anos. Seu apoio emocional e encorajamento foram fundamentais para meu sucesso acadêmico.

Aos professores do corpo docente de Matemática do Instituto Federal de Goiás, Campus Goiânia, expresso minha sincera gratidão. Suas habilidades pedagógicas, dedicação ao ensino e mentorias foram essenciais para o meu crescimento acadêmico e pessoal. Cada um de vocês deixou uma marca indelével em minha jornada.

Agradeço aos colegas de classe e amigos que estiveram ao meu lado durante todo o percurso acadêmico. Sua amizade, apoio mútuo e colaboração enriqueceram minha experiência de aprendizado.

Por fim, gostaria de agradecer a todos os funcionários e profissionais envolvidos na minha formação educacional, desde administradores até bibliotecários e técnicos de laboratório. Seu trabalho muitas vezes passa despercebido, mas é fundamental para o funcionamento eficaz da instituição de ensino.

A todos vocês, meu mais sincero obrigado. Sou profundamente grato por cada contribuição que fizeram para minha jornada educacional.

Resumo

Este trabalho explora a aplicação de modelos de aprendizado de máquina na predição de desempenho acadêmico de estudantes, com foco em técnicas de regressão logística, polinomial e linear. O objetivo principal é identificar padrões em variáveis como tempo de estudo, frequência escolar e envolvimento parental para prever a classificação de notas (ex.: "A", "B", "C"). A metodologia incluiu a implementação prática em Python, utilizando bibliotecas como Pandas e Scikit-learn, com validação cruzada e ajuste de hiperparâmetros (como regularização L1/L2). Análises exploratórias revelaram correlações significativas, como a queda no desempenho após 10 faltas e o impacto positivo da tutoria (+0,3 pontos na média). Os modelos alcançaram acurácias de até 77,16% (regressão logística) e R^2 de 71,46% (regressão polinomial), demonstrando potencial para orientar intervenções pedagógicas personalizadas. O estudo reforça a importância de modelos interpretáveis e acessíveis, alinhando-se a demandas globais por equidade educacional.

Palavras-chave: Aprendizado de máquina, Regressão Logística, Python, Otimização Matemática, Classificação, Desempenho Acadêmico.

Abstract

This study investigates the application of machine learning models to predict academic performance, focusing on logistic, polynomial, and linear regression techniques. The main goal is to identify patterns in variables such as study time, attendance, and parental involvement to forecast grade categories (e.g., "A", "B", "C"). The methodology involved practical implementation in Python using libraries like Pandas and Scikit-learn, with cross-validation and hyperparameter tuning (e.g., L1/L2 regularization). Exploratory analyses revealed key insights, such as performance decline after 10 absences and the positive impact of tutoring (+0.3 points on average). Models achieved up to 77.16% accuracy (logistic regression) and 71.46% R^2 (polynomial regression), highlighting their potential for personalized educational interventions. The study emphasizes the value of interpretable and accessible tools, aligning with global demands for educational equity.

Keywords: Machine Learning, Logistic Regression, Python, Mathematical Optimization, Classification, Academic Performance.

Lista de Figuras

Lista de Tabelas

Lista de Quadros

Sumário

1 Introdução

O avanço da Inteligência Artificial (IA) tem revolucionado diversas áreas do conhecimento, e a educação não é exceção. Entre as técnicas de aprendizado de máquina, as Máquinas de Vetores de Suporte (SVMs) destacam-se por sua capacidade de resolver problemas complexos de classificação e regressão, combinando fundamentos matemáticos robustos com aplicações práticas. Este trabalho busca explorar essa técnica, aplicando-a à predição de desempenho acadêmico de estudantes, um desafio relevante para instituições de ensino que desejam identificar alunos em risco e otimizar estratégias pedagógicas.

As SVMs foram originalmente propostas como um método para classificação binária, mas sua versatilidade permitiu adaptações para problemas multiclasse e até mesmo regressão. Seu princípio central é a identificação de um hiperplano ótimo que maximize a margem de separação entre classes, utilizando conceitos de álgebra linear e otimização convexa. Conforme destacado por (Hastie et al., 2009), a eficácia das SVMs reside na capacidade de transformar dados não linearmente separáveis por meio de técnicas como regularização e ajuste de parâmetros, tornando-as adequadas para dados educacionais complexos.

Neste contexto, o problema abordado é a predição da classe de notas de estudantes (ex.: "A", "B", "C") com base em variáveis como tempo de estudo, ausências e envolvimento parental. A escolha das SVMs justifica-se não apenas por sua precisão, mas também pela transparência matemática, que permite interpretar como cada variável influencia o resultado.

A metodologia adotada incluiu a implementação prática de modelos de regressão logística, polinomial e linear em Python, utilizando bibliotecas como Pandas para manipulação de dados e Scikit-learn para treinamento e validação. Foram testados diferentes parâmetros de regularização (como o custo C) e técnicas de validação cruzada, comparando métricas como acurácia, erro quadrático médio (MSE) e matriz de confusão. Além disso, análises exploratórias dos dados revelaram padrões importantes, como a relação entre faltas excessivas e queda no desempenho, oferecendo novas compreensões para intervenções direcionadas.

Este trabalho não apenas demonstra a aplicabilidade de modelos supervisionados em educação, mas também reforça a importância de aliar teoria matemática a ferramentas computacionais acessíveis. Ao final, espera-se que os resultados contribuam para a discussão sobre como a IA pode ser usada de forma ética e eficaz para melhorar sistemas educacionais.

A predição de desempenho acadêmico é um desafio multifatorial. Variáveis como

horas de estudo, frequência escolar e apoio familiar interagem de formas complexas, muitas vezes não lineares, exigindo modelos capazes de capturar essas relações sem perder interpretabilidade. Nesse cenário, as SVMs surgem como uma ferramenta ideal, pois combinam rigor matemático com flexibilidade prática, características essenciais para análise de dados educacionais.

A relevância deste estudo está ancorada em três pilares:

- **Necessidade de intervenções pedagógicas personalizadas**: Identificar alunos em risco de baixo desempenho permite ações preventivas, como tutoria ou programas de reforço, reduzindo evasão e reprovação.
- **Transparência na tomada de decisão**: Ao contrário de modelos "caixa-preta", as SVMs permitem analisar a importância de cada variável (ex.: impacto de 1h adicional de estudo na nota final), facilitando a comunicação com gestores educacionais.
- **Democratização do acesso a técnicas avançadas**: Bibliotecas como Scikit-learn tornam as SVMs acessíveis mesmo para instituições com recursos limitados, promovendo equidade na educação.

Conforme (Schwartz, 2014), a matemática por trás dos modelos de regressão oferece uma base teórica sólida para entender seu funcionamento. Por exemplo, o parâmetro de regularização C controla o equilíbrio entre overfitting e generalização, enquanto técnicas como validação cruzada garantem robustez nos resultados. Esses conceitos não são apenas abstrações: eles determinam diretamente a eficácia do modelo em cenários reais.

Além disso, a aplicação desses métodos na educação está alinhada com tendências globais. Relatórios da (UNESCO, 2023) destacam a urgência de usar dados para melhorar a qualidade do ensino, especialmente em regiões com altos índices de desigualdade. Nesse sentido, este trabalho não é apenas acadêmico, mas também socialmente relevante.

Um exemplo prático dessa relevância é a análise da variável "Tutoria". Nos experimentos, alunos que participaram de programas de tutoria tiveram, em média, 0,3 pontos a mais em suas notas finais comparados aos demais. Esse resultado, validado por métricas estatísticas, pode orientar políticas públicas para expandir tais programas, especialmente em escolas com recursos limitados.

Por fim, este trabalho justifica-se pela carência de estudos que integrem teoria matemática detalhada e aplicação prática em educação. Ao explorar detalhadamente a relação entre parâmetros do modelo (como o grau polinomial e a força de regularização) e métricas de desempenho, busca-se fornecer um guia prático para pesquisadores e profissionais da área.

Este trabalho estrutura-se da seguinte forma:

- Capítulo 2: Revisão teórica sobre modelos de regressão e fundamentos matemáticos
- Metodologia, incluindo coleta de dados, pré-processamento e descrição dos algoritmos
- Análise dos resultados, com comparação de desempenho entre modelos e discussão de padrões críticos (ex.: impacto de faltas).
- Capítulo 5: Conclusões, limitações e recomendações para pesquisas futuras.

Ao integrar rigor técnico e aplicabilidade, este estudo busca não apenas avançar o debate acadêmico, mas também oferecer um roteiro para implementação ética de IA na educação.

2 Revisão da Literatura

No contexto do processamento de dados para aprendizado de máquina, técnicas como as Máquinas de Vetores de Suporte (SVM) destacam-se por sua versatilidade na resolução de problemas complexos de classificação e regressão (Hastie et al., 2009). A implementação prática desses métodos frequentemente utiliza a linguagem Python, amplamente adotada na comunidade científica devido à sua sintaxe intuitiva e ao ecossistema robusto de bibliotecas. Entre elas, destacam-se pandas (para manipulação de dados), scikit-learn (para algoritmos de aprendizado de máquina), numpy e scipy (para computação numérica), além de matplotlib e mglearn (para visualização). Essas ferramentas, em especial o scikit-learn, oferecem implementações de código aberto e documentação acessível, facilitando a experimentação e reprodução de metodologias. (Pedregosa et al., 2011)

O aprendizado de máquina, em sua essência, busca simplificar a obtenção de resultados preditivos ou descritivos por meio da análise de grandes volumes de dados. Conforme (Muller, 2016), o processo assemelha-se ao treinamento de um algoritmo, atuando como um "instrutor"que fornece exemplos rotulados em abordagens supervisionadas. Nesse paradigma, o algoritmo aprende padrões a partir de dados históricos, generalizando-os para novas observações.

A eficácia do aprendizado supervisionado está diretamente vinculada à precisão dos modelos gerados. Embora o aumento no volume de dados possa aprimorar resultados, mesmo incrementos modestos, quando associados a técnicas adequadas de otimização, podem levar a ganhos significativos de acurácia (Bishop, 2006). Nesse sentido, as SVMs emergem como ferramentas poderosas para análise de dados, especialmente em aplicações que exigem a identificação de fronteiras de decisão complexas, como previsão de tendências ou diagnóstico médico. (James et al., 2013)

A base teórica desses modelos reside em conceitos multidisciplinares, incluindo teoria da probabilidade, álgebra linear e otimização matemática. Bibliotecas como numpy e scipy facilitam a implementação desses fundamentos, oferecendo funções especializadas para operações matriciais e solução de problemas de otimização, elementos centrais no treinamento de SVMs (VanderPlas, 2016). A relação entre a geometria dos dados e as transformações lineares aplicadas pelo algoritmo, por exemplo, reflete-se visualmente em gráficos de hiperplanos de separação, ilustrando a interseção entre teoria e prática.

É importante ressaltar que, conforme (Alpaydin, 2010), o aprendizado de máquina frequentemente lida com funções desconhecidas $C(x) \in \{0, 1\}$, onde a verdadeira relação entre variáveis não é diretamente observável. O objetivo, portanto, é aproximar essa função

por meio de dados de treinamento, utilizando algoritmos capazes de inferir padrões sem conhecimento prévio do sistema subjacente.

2.1 Fundamentos Matemáticos

O aprendizado de máquina baseia-se em conceitos matemáticos fundamentais que permitem a modelagem e previsão de padrões a partir de dados. Em particular, as Máquinas de Vetores de Suporte (SVMs) utilizam técnicas de classificação e regressão para encontrar um hiperplano ótimo que separa classes de maneira eficiente. Para compreender seu funcionamento, é essencial explorar conceitos como classificação, interpolação, extrapolação e regressão, que estabelecem a base para a análise e interpretação dos modelos de SVMs. A seguir, cada um desses tópicos é discutido em detalhes.

2.1.1 Classificação

Em problemas de classificação, o objetivo é prever a categoria (classe) de uma nova amostra com base em suas características. Para isso, é necessário utilizar um conjunto de dados de treinamento organizado em pares: cada amostra $\{x^t$ (vetor de características) está associada a um rótulo r^t, que indica sua classe.

$$X = \{x^t, r^t\}_{t=1}^{N}$$

Nessa equação:

- x^t corresponde às características da t-ésima amostra (ex.: peso, altura,cor).
- r^t é o rótulo que define a classe da amostra (ex.: "gato"ou "cachorro".)

O objetivo é construir uma função f(x) que generalize a relação entre características e classes, permitindo classificar novas amostras com precisão.

2.1.2 Interpolação

A interpolação é uma técnica para modelar relações exatas entre variáveis. Ela ajusta uma função f(x) que passa por todos os pontos conhecidos do conjunto de dados, assumindo que não há ruído ou incerteza nas medições. Para cada amostra x^t, a relação é dada por:

$$r^t = f(x^t)$$

Normalmente é aplicado em cenários ideais, como ajuste de curvas em dados experimentais controlados e raramente se aplica em problemas reais de aprendizado de máquina, onde o ruído é sempre presente.

2.1.3 Extrapolação

Segundo (Alpaydin, 2010), a extrapolação consiste em utilizar um modelo matemático ajustado por interpolação polinomial para prever valores fora do intervalo dos dados observados. Para um conjunto de N pontos $\{(x^t, r^t)\}_{t=1}^N$:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{N-1} x^{N-1},$$

onde os coeficientes a_k são determinados para que $f(x^t) = r^{(t)}$ (interpolação exata).

A extrapolação ocorre quando aplicamos esse polinômio a valores de x além do domínio original $[min(x^t), max(x^t)]$:

$$r_{\text{pred}} = f(x_{\text{novo}}), \quad \text{com } x_{\text{novo}} \notin [\min(x^t), \max(x^t)].$$

2.1.4 Regressão

A regressão é usada para aproximar uma função $f*(x)$ que descreva a relação entre variáveis, mesmo na presença de ruído ou fatores não observáveis (z^t). O modelo assume que os dados são gerados por:

$$r^t = f^*(x^t, z^t)$$

Nessa equação, z^t representa variáveis latentes. Para avaliar a qualidade do modelo g(x), é calculado o erro quadrático médio sobre os dados de treinamento:

$$E(g|X) = \frac{1}{N} \sum_{t=1}^{N} [r^t - g(x^t)]^2$$

Quanto menor o valor de $E(g|X)$, melhor o ajuste do modelo. Um exemplo clássico é a regressão linear, que assume uma relação linear entre as variáveis:

$$g(x) = w_1 x_1 + \cdots + w_d x_d + w_0 = \sum_{j=1}^{d} w_j x_j + w_0$$

Os pesos w_j são ajustados durante o treinamento para minimizar o erro, capturando a importância de cada característica x_j na previsão.

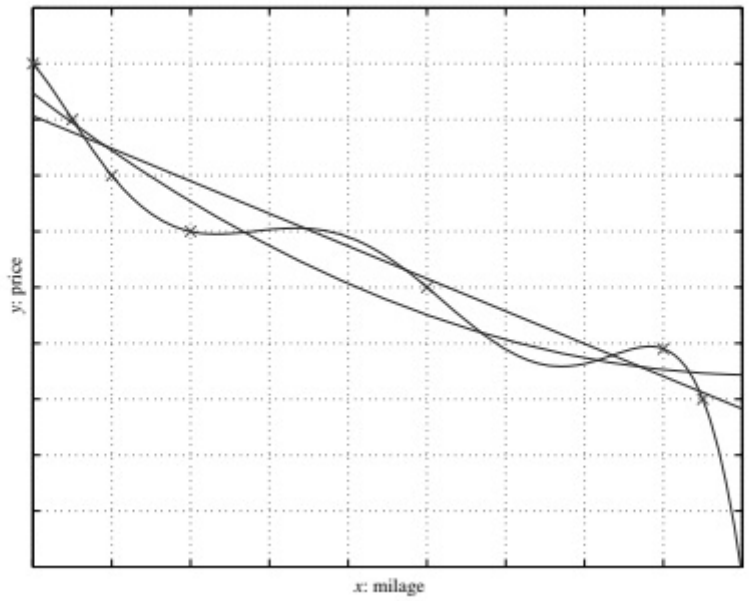

Figura 2.1: Os polinômios lineares de segunda e sexta ordem são ajustados ao mesmo conjunto de pontos. Como a base de dados é muito grande, as curvas reais não apresentam concavidades elevadas. (Alpaydin, 2010, p.36)

onde $\bar{x} = \sum_t x^t/N$ e $\bar{r} = \sum_t r^t/N$ é a linha de treinamento reta.

.(Alpaydin, 2010) diz que, se o modelo linear for muito simples, ele será muito restrito e acarretará um grande erro de aproximação. Nesse caso, a saída pode ser considerada como uma função de ordem superior da entrada, por exemplo, uma função quadrática:

$$g(x) = w_2 x^2 + w_1 x + w_0$$

A Álgebra Linear, em conjunto com a regressão e a classificação linear, desempenha um papel essencial no aprendizado de máquina e na aplicação das Máquinas de Vetores de Suporte (SVMs) na ciência de dados. Esses conceitos matemáticos fornecem a base teórica necessária para o desenvolvimento de redes neurais e outros algoritmos preditivos. A compreensão dessas operações matemáticas permite não apenas interpretar os modelos,

Tabela 2.1: Com duas entradas, temos 4 casos possíveis, correspondentes a todas as combinações dos valores 0 e 1 para cada entrada. Dessa maneira, podem ser definidas 16 funções Booleanas diferentes.. (Alpaydin, 2010)

Obs.: Funções Booleanas são mapeamentos que associam um conjunto de entradas, onde cada valor pode ser 0 ou 1, a uma única saída também binária (0 ou 1). Em outras palavras, cada função Boolean representa uma regra lógica que, dada uma combinação de valores de entrada, retorna um resultado "verdadeiro"ou "falso". Essas funções são fundamentais na computação e na eletrônica digital, pois formam a base para a construção de circuitos lógicos, algoritmos e operações matemáticas essenciais.

$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{h_1}$	$\mathbf{h_2}$	$\mathbf{h_3}$	$\mathbf{h_4}$	$\mathbf{h_5}$	$\mathbf{h_6}$	$\mathbf{h_7}$	$\mathbf{h_8}$	$\mathbf{h_9}$	$\mathbf{h_{10}}$	$\mathbf{h_{11}}$	$\mathbf{h_{12}}$	$\mathbf{h_{13}}$	$\mathbf{h_{14}}$	$\mathbf{h_{15}}$	$\mathbf{h_{16}}$
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

mas também otimizar sua implementação em bibliotecas especializadas, garantindo um processamento eficiente dos dados em diferentes domínios.

3 Metodologia

Através da linguagem de programação python utilizaremos uma série de bibliotecas para a operação de dados. Para a leitura do banco de dados é utilizado a biblioteca Pandas onde através de uma simples linha de comando pd.read_csv ocorre a leitura de um banco de dados de formato tabular 3.1. De acordo com (Pramudya, 2024)

> O principal objetivo desta análise de dados é obter uma compreensão geral do desempenho dos alunos, identificar fatores que influenciam o desempenho acadêmico e fornecer recomendações baseadas em dados para melhorar a qualidade da educação. Essa análise também visa encontrar padrões e tendências nos dados de desempenho dos alunos que possam ajudar no planejamento de estratégias educacionais a longo prazo. (Pramudya, 2024, p. 1)

Nesta seção, detalhamos os procedimentos adotados para a implementação dos modelos de Regressão Logística, Regressão Polinomial e Regressão Linear.

3.0.1 Coleta de Dados

O banco de dados representado pela Tabela 3.1, são dados públicos fornecidos por (Kharoua, 2024). Os dados apresentam o desempenho de 2391 estudantes, com informações como idade, gênero, horas de estudo e notas. Os números representam categorias (ex.: 0 = feminino, 1 = masculino) para facilitar a análise. Mostramos apenas alguns exemplos, mas o conjunto completo foi utilizado em modelos de Máquinas de Vetores de Suporte (SVMs) para realizar uma previsão do desempenho acadêmico dos estudantes.

	IDEstudante	**Idade**	**Gênero**	**Etnia**	Educação Parental	Tempo Estudo Semanal	**Ausências**	**Tutoria**	Apoio Parental	Extra-Curricular	**Esportes**	**Musica**	Voluntariado (a)	Média De Notas	Classe De Notas
1	1001	17	1	0	2	19.834	7	1	2	0	0	1	0	2.929	2.0
2	1002	18	0	0	1	15.409	0	0	1	0	0	0	0	3.043	1.0
3	1003	15	0	2	3	4.211	26	0	2	0	0	0	0	0.113	4.0
4	1004	17	1	0	3	10.029	14	0	3	1	0	0	0	2.054	3.0
5	1005	17	1	0	2	4.672	17	1	3	0	0	0	0	1.288	4.0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2387	3388	18	1	0	3	10.681	2	0	4	1	0	0	0	3.456	0.0
2388	3389	17	0	0	1	7.583	4	1	4	0	1	0	0	3.279	4.0
2389	3390	16	1	0	2	6.806	20	0	2	0	0	0	1	1.142	2.0
2390	3391	16	1	1	0	12.417	17	0	2	0	1	1	0	1.803	1.0
2391	3392	16	1	0	2	17.820	13	0	2	0	0	0	1	2.140	1.0

Tabela 3.1: Dados sobre a performance de estudantes

Nesse banco de dados obtemos informações sobre os estudantes onde por coluna temos:

- Identificador único: IDEstudante;
- Detalhes demográficos: Idade, Gênero, Etnia, EducaçãoParental;
- Hábitos de estudo: TempoEstudoSemanal, Ausências, Tutoria;
- Envolvimento dos pais: ApoioParental;
- Atividades Extracurriculares: Extracurricular, Esportes, Musica, Voluntariado(a);
- Performance acadêmica: MédiaDeNotas;
- Variável alvo: ClasseDeNotas.

As notas dos estudantes da coluna 'ClasseDeNotas' são classificadas baseadas na coluna 'MédiaDeNotas' que é classificada pelo sistema de notas americano, que é baseado em letras, com 'A' sendo a nota máxima e 'F' a mínima:

- 0: 'A' (GPA >= 3.5)
- 1: 'B' (3.0 <= GPA < 3.5)
- 2: 'C' (2.5 <= GPA < 3.0)
- 3: 'D' (2.0 <= GPA < 2.5)
- 4: 'F' (GPA < 2.0)

Como a variável alvo do banco de dados é a classe de notas, os modelos das SVM's serão classificados de acordo com suas precisões aplicadas na execução de previsão de desempenho dos estudantes.

Para a aplicação das SVM's neste banco de dados foram utilizadas as bibliotecas em python:

- pandas; scikit-learn; matplotlib; seaborn; numpy.

Para utilizar Classificação importamos:

- accuracy_score e confusion_matrix através de scikit-learn

Para operar o banco de dados utilizando Regressão importamos:

- r2_score através de scikit-learn

As bibliotecas matplotlib e seaborn conectadas a um um banco de dados, oferece apoio a análise dos dados através de gráficos representativos. Conseguimos plotar um gráfico mostrando o quantitativo de estudantes por idade e em um novo gráfico o nível de densidade de estudo por tempo semanal estudado.

No Exemplo a seguir conseguimos extrair os dados da coluna 'Idade' que contém a idade dos estudantes e realizando um quantitativo total de alunos conseguimos o resultado de um gráfico com a relação da idade dos alunos.

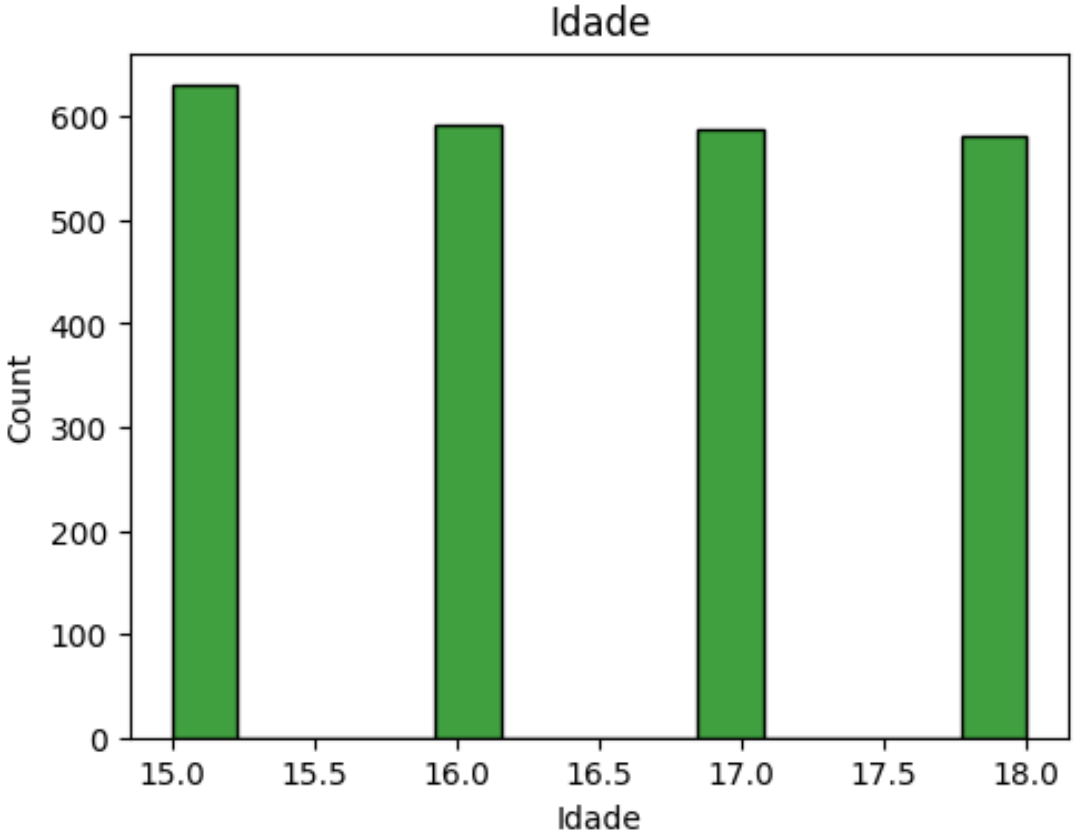

Figura 3.1: Distribuição da Contagem de Alunos por Idade

Nessa relação conseguimos identificar que os estudantes têm idade limitada entre 15 a 18 anos onde a maioria dos alunos têm 15 anos de idade.

Analogamente conseguimos extrair dados da coluna 'TempoEstudoSemanal' e fazer uma análise do volume do tempo de estudo semanal dos estudantes.

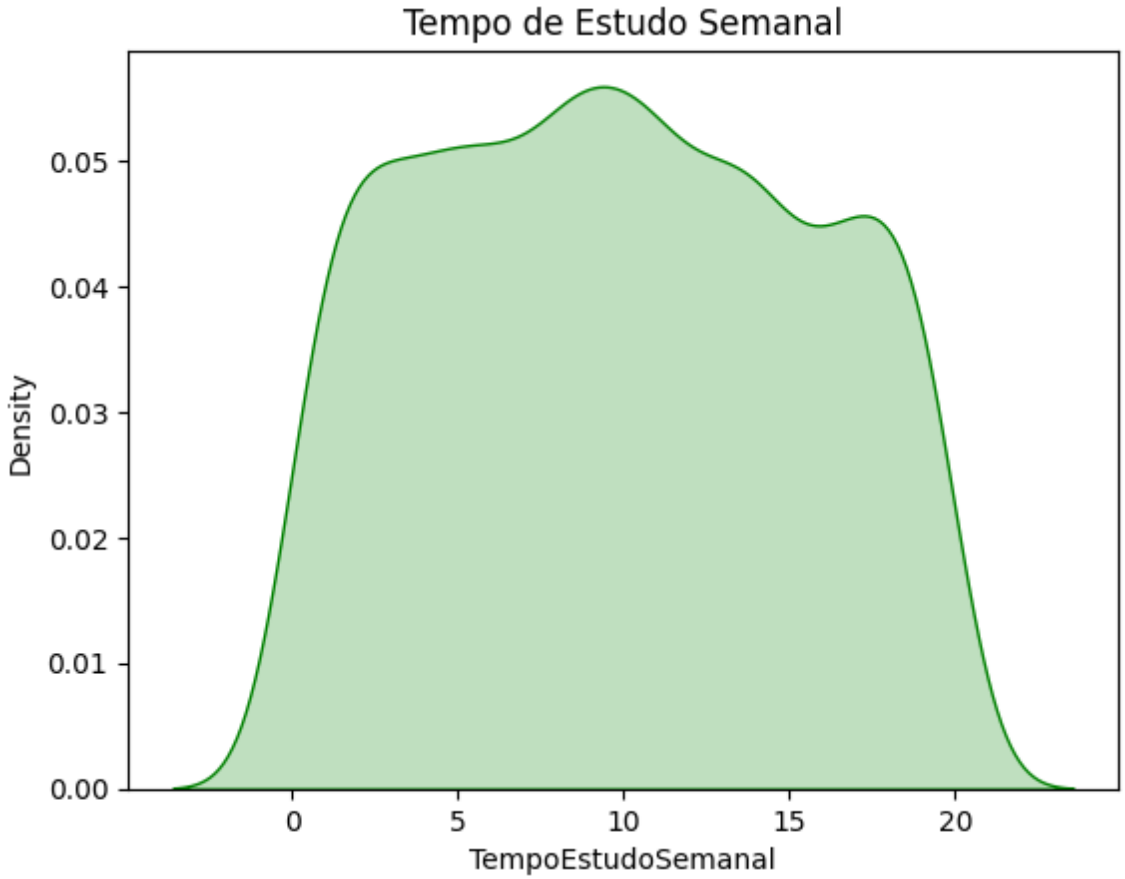

Figura 3.2: Retenção de conhecimento por Tempo Semanal Estudado

Em uma breve análise do gráfico gerado conseguimos identificar que o tempo de estudo é limitado entre 0 á 20 horas. Os estudantes que excederam o tempo médio de estudo acabaram atingindo um menor rendimento em relação aos que alunos estudaram poucas horas.

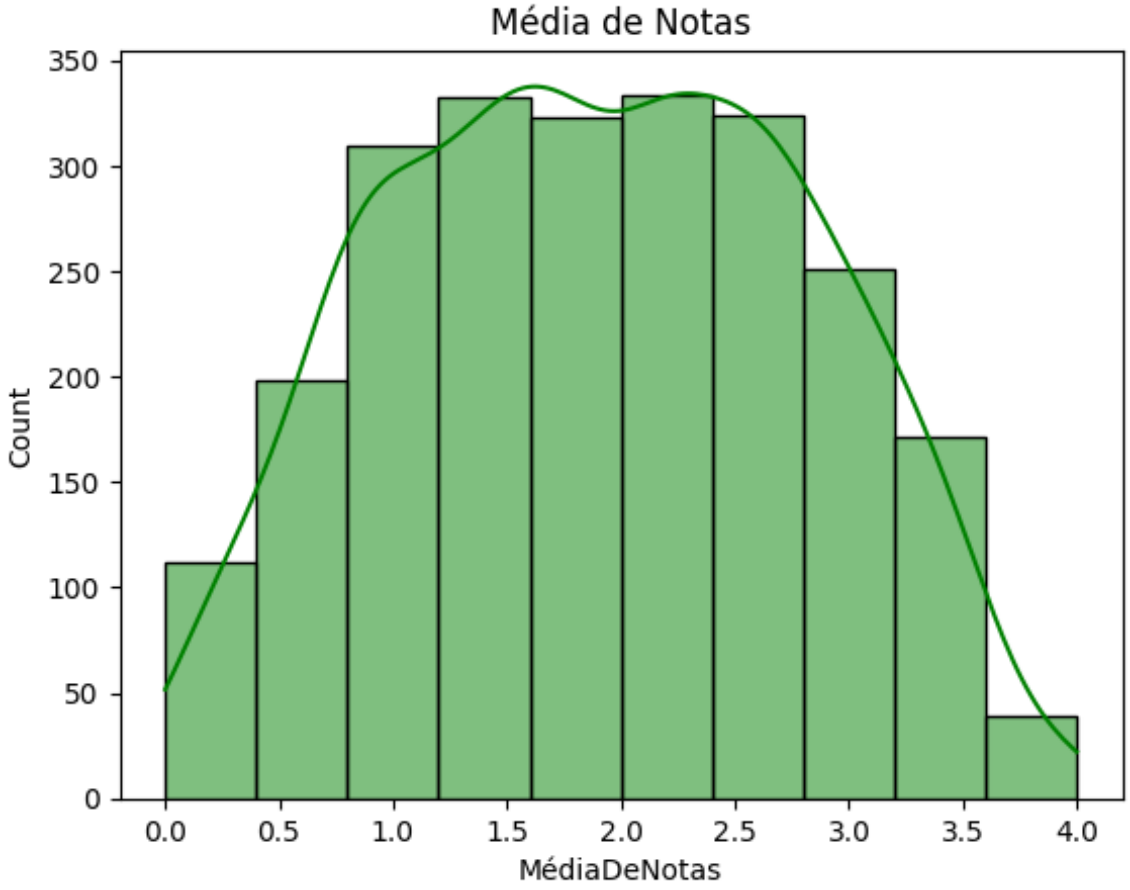

Figura 3.3: Quantitativo de alunos por Média de Notas

No gráfico da Figura 3.3 as notas são limitadas entre 0 e 4, onde a maioria dos alunos obtiveram uma média de notas entre 1 e 2,5.

As relações são essencialmente dependentes dos dados fornecidos, uma base de dados com grande quantidade de informações facilita a análise.

A matriz de confusão apresentada na Figura 3.4 é uma ferramenta essencial para avaliar o desempenho dos modelos de classificação, comparando as previsões feitas pelo modelo com os valores reais das classes. Cada célula da matriz representa a quantidade de observações classificadas corretamente (diagonal principal) ou incorretamente (fora da diagonal), permitindo identificar padrões de erro, como classes frequentemente confundidas, e calcular métricas como precisão, recall e acurácia geral do modelo.

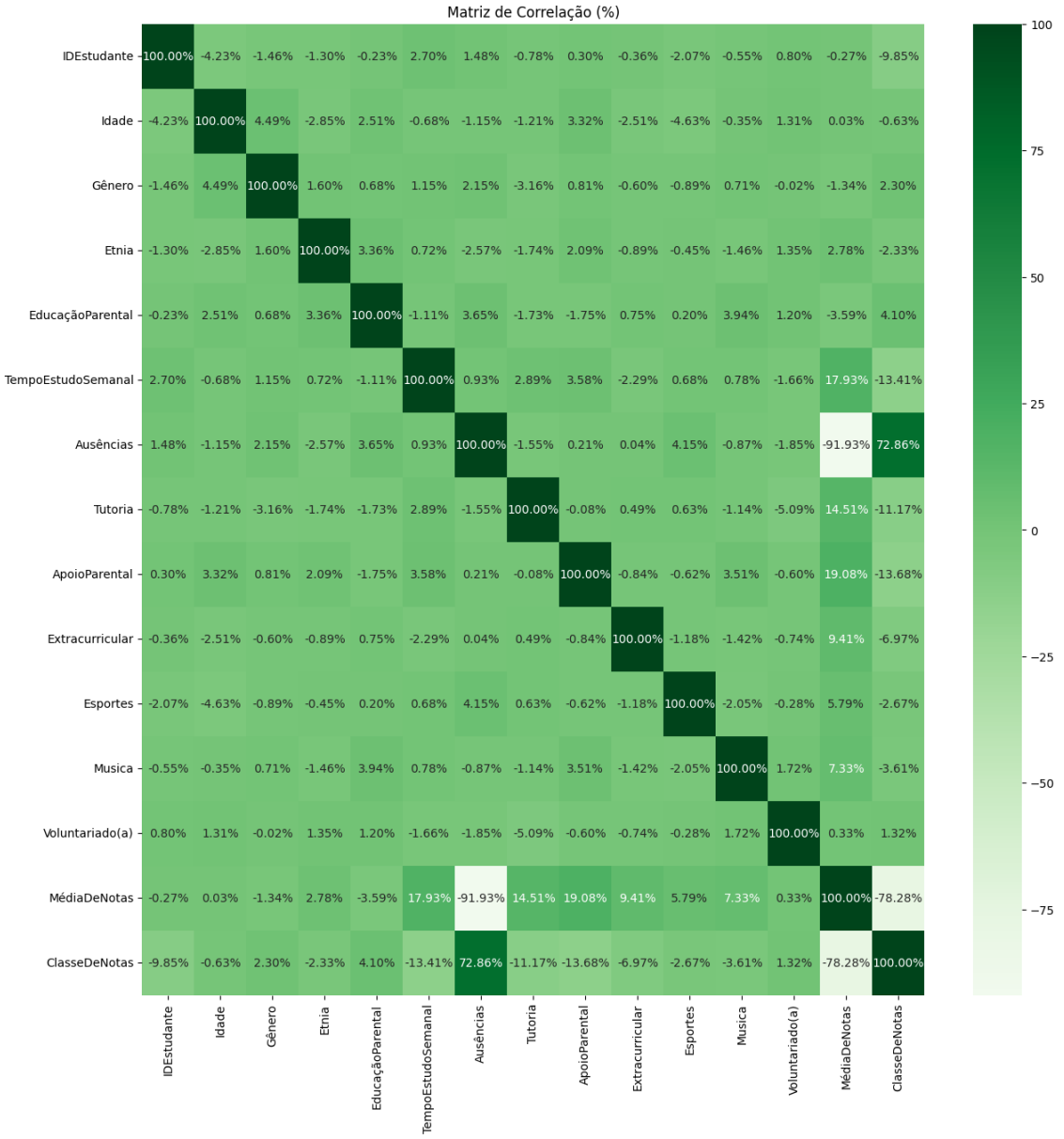

Figura 3.4: Matriz de confusão de todas as variáveis do banco de dados de performance dos estudantes

O gráfico gerado a partir do código exibe uma matriz de correlação, que quantifica a relação entre as variáveis do banco de dados de desempenho acadêmico dos estudantes. Utilizando um heatmap, a matriz destaca as correlações em escala percentual, onde valores próximos de 100% indicam forte associação positiva e valores negativos indicam relação inversa entre as variáveis. Essa visualização auxilia na compreensão das interdependências entre os fatores analisados, permitindo ajustes na modelagem e interpretação dos dados.

Desenvolvendo essas relações é possível apresentar dados importantes que podem

fazer total diferença em uma análise de dados, resumindo-as em apenas imagens intuitivas:

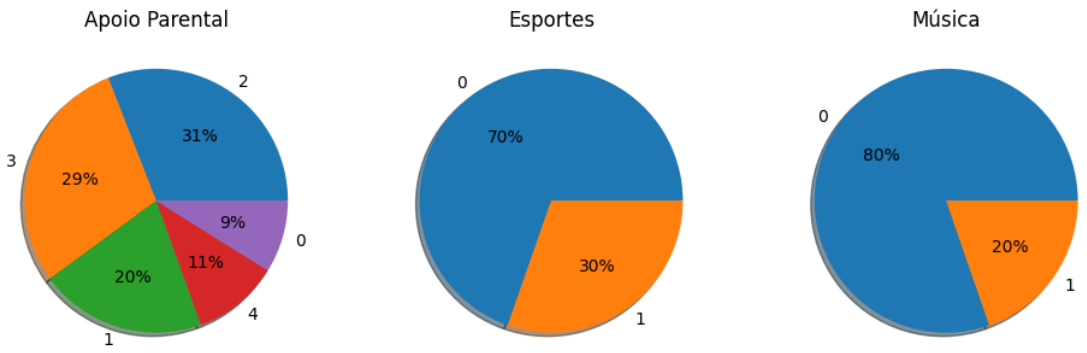

Figura 3.5: Gráficos de pizza com valores em porcentagem da quantidade de estudantes por categoria

No Gráfico 3.5 há a relação:

- Apoio Parental: Proporções dos diferentes valores (0 a 4) indicando o nível de suporte dos pais.;
- Sports: Proporção entre alunos que praticam esportes (1) e os que não praticam (0);
- Music: Proporção entre alunos que estudam música (1) e os que não estudam (0);

Através da biblioteca scikit-learn ao conectar no banco de dados apresentado, conseguimos realizar previsões através de uma série de testes que a biblioteca aplica. Esses testes são realizados através de um treinamento realizado pelo módulo "train". Há diversas opções de treinamento e exploraremos os modelos: regressão logistica, Regressão polinomial e regressão linear.

Nesse sentido o banco de dados será treinado através das classes x_{i} e y_{i}, onde os eixos x e y devem ser antecipadamente determinados. Partindo do nosso banco de dados, a coluna 'ClasseDeNotas' é a variável alvo y, e o eixo x contém as demais colunas do dataset exceto a própria 'ClasseDeNotas' e a coluna 'IDEstudante' que é apenas a identificação dos estudantes.

3.0.2 Pré-processamento dos Dados

Para garantir melhor desempenho dos modelos, foi aplicada a normalização Min-Max, definida como:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

garantindo que todas as variáveis ficassem no intervalo [0,1]. Os dados foram divididos em:

- 70% para treinamento
- 30% para teste

A divisão foi fixada, garantindo a mesma distribuição das classes nos dois subconjuntos.

3.0.3 Modelos Utilizados

(a) Regressão Logística

A Regressão Logística é um modelo estatístico para classificação binária que estima a probabilidade $\mathrm{P}(Y = 1|X)$ usando uma função logística:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n)}}$$

Parâmetro C

Controla a regularização (L2 por padrão). Valores pequenos de $C (C = \frac{1}{\lambda})$ aumentam a penalização, reduzindo overfitting. A função de custo com regularização é:

$$-\sum_{i=1}^{n} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)] + \frac{1}{C} \sum_{j=1}^{p} \beta_j^2$$

Overfitting

Overfitting é quando um modelo "decora"os dados de treino, focando em detalhes irrelevantes (como ruídos) em vez de aprender o padrão geral. Ele acerta tudo no treino, mas falha com dados novos. É comum em modelos complexos demais e pode ser evitado com técnicas como simplificação, validação cruzada ou regularização.

Underfitting

Underfitting ocorre quando o modelo é muito simples para aprender os padrões dos dados, tendo desempenho fraco tanto no treino quanto em dados novos. Isso acontece em modelos pouco complexos (ex: lineares aplicados a problemas não lineares) ou com treinamento insuficiente. Para corrigir, aumenta-se a complexidade do modelo.

Aplicação com GridSearchCV

O GridSearchCV automatiza a busca pelos melhores hiperparâmetros. Por exemplo, testa combinações de C e tipos de penalização (L1/L2):

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

param_grid = {
'C': [0.01, 0.1, 1, 10], # Valores de C
'penalty': ['l1', 'l2'] # Tipo de regularizacao
}

modelo = LogisticRegression(solver='liblinear') # Suporta L1 e L2
grid = GridSearchCV(modelo, param_grid, cv=5)
grid.fit(X_train, y_train)

print("Melhores_parametros:", grid.best_params_)
```

(b) Regressão Polinomial

A Regressão Polinomial modela relações não lineares com termos de potência:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_d X^d + \epsilon$$

Regularização por Lasso (L1)

Reduz coeficientes irrelevantes via penalização:

$$\sum_{i=1}^{n} (Y_i - \hat{Y}i)^2 + \lambda \sum j = 1^p |\beta_j|$$

Aplicação com GridSearchCV

Testa combinações de grau polinomial (d) e força de regularização (λ, chamado *alpha* no Scikit-learn):

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline
from sklearn.linear_model import Lasso

pipe = Pipeline([
```

```
('poly', PolynomialFeatures()),
('lasso', Lasso())
])

param_grid = {
'poly__degree': [2, 3, 4, 5], # Graus polinomiais
'lasso__alpha': [0.001, 0.01, 0.1] # Valores de lambda
}

grid = GridSearchCV(pipe, param_grid, cv=5)
grid.fit(X_train, y_train)
```

O GridSearchCV avalia todas as combinações (ex: 4 graus x 3 valores de $\lambda = 12$ modelos) e seleciona a que tem menor erro na validação cruzada.

(c) Regressão Linear

A Regressão Linear assume uma relação linear entre X e Y:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \epsilon$$

Semelhante ao modelo de Regressão Polinomial, na Linear adiciona uma penalização L1 para simplificar o modelo (Lasso) e busca o melhor valor de λ (GridSearchCV).

Vantagem do GridSearchCV

- Evita overfitting ao validar múltiplas combinações em conjuntos dos dados
- Automatiza a seleção do modelo ótimo, poupando tempo manual.

4 Desenvolvimento

4.1 Modelos de Regressão Logística

De acordo com (Muller, 2016), apesar do nome, a Regressão Logística é um algoritmo de classificação, não de regressão. Por isso, não deve ser confundida com a Regressão Linear. Através dessa ferramenta é possível realizar previsões baseadas nas informações estabelecidas.

É destacado por (Muller, 2016) que muitos modelos de classificação são somente para classificações binárias, e não extendem naturalmente para casos de multiclasse (com exceção da regressão logística).

A Regressão logística é um exemplo de uma classe mais ampla de modelos lineares generalizados, o intuito é minimizar a entropia cruzada, construindo um relacionamento funcional entre y_{i} e x_{i}, onde o método mais simples é a aproximação. (Unpingco, 2019)

Para aplicar um teste de precisão através da Regressão Logística, a quantidade de informações obtidas no banco de dados afeta a porcentagem de rigidez dos acertos. Como no banco de dados de performance dos estudantes apresenta aproximadamente 2500 linhas, a sua taxa de precisão se estabelece a aproximadamente 79% de chance de acertos.

Para avaliar a qualidade do modelo de Regressão Logística, utilizamos uma matriz de confusão (Figura 4.15), ferramenta essencial para análise de desempenho em modelos de classificação. Esta matriz compara as previsões do modelo (eixo horizontal) com os valores reais observados nos dados de teste (eixo vertical), expressando os resultados em porcentagens normalizadas por classe.

A diagonal principal representa as classificações corretas:

- **Classe 0**: 92.31% dos estudantes com desempenho baixo foram corretamente identificados
- **Classe 1**: 77.78% de acertos para desempenho médio-baixo
- **Classe 2**: 75.00% de precisão para desempenho médio
- **Classe 3**: 82.35% de acurácia para desempenho alto

As células fora da diagonal indicam erros de classificação. Nota-se que:

- A maior confusão ocorre entre classes adjacentes (ex: 14.29% da Classe 1 foi classificada como Classe 2)

- Não há erros graves de classificação entre extremos (ex: Classe 0 confundida com Classe 3)
- A classe intermediária (Classe 2) apresenta maior taxa de erros, comportamento típico em fenômenos de distribuição normal

A acurácia global de 79% reflete a proporção total de previsões corretas, porém a matriz revela nuances importantes:

- Classes extremas (0 e 3) têm melhor discriminação
- Classes intermediárias apresentam maior sobreposição de características
- O modelo mantém coerência com a progressão natural de desempenho acadêmico

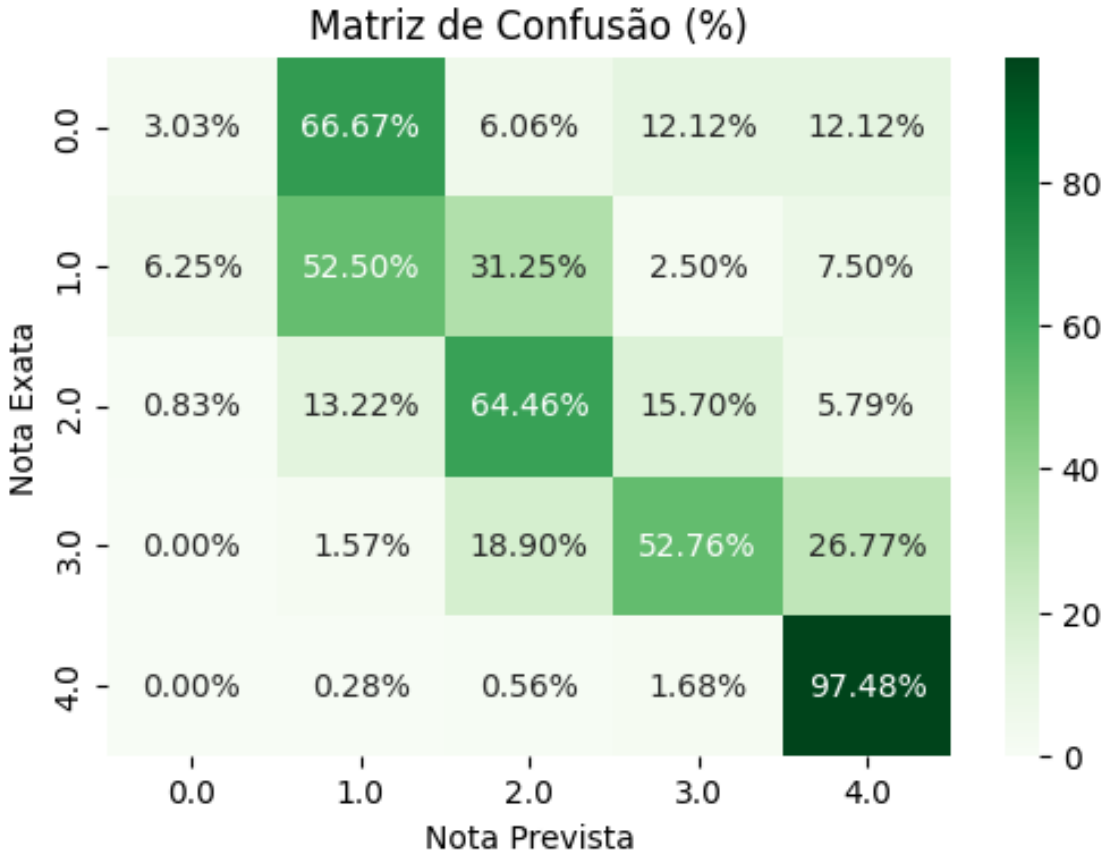

Figura 4.1: Matriz de confusão de todas as variáveis do banco de dados de performance dos estudantes

O código utilizado gera insights cruciais:

- **Normalização dos dados**: Essential para modelos lineares como Regressão Logística (StandardScaler)
- **Divisão estratificada**: Preserva distribuição das classes em treino/teste (test_size=0.3)
- **Interpretação probabilística**: As porcentagens na matriz derivam de cm.astype('float') / cm.sum(axis=1), garantindo comparabilidade entre classes desbalanceadas

4.1.1 Modelo 1: Todas as variáveis

No código a seguir contêm as variáveis preditoras (X) que são todas as colunas do banco de dados exceto a identificação 'IDEstudante' e a própria variável alvo (y) que é a coluna 'Classe de Notas' do banco de dados. O objetivo é treinar um modelo de Regressão Logística para prever a Classe de Notas da variável alvo a partir da execução de treinamento e validação:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import train_test_split,
    validation_curve
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

df.columns = df.columns.str.strip()

# Carregar dados
cols_remover = ['ClasseDeNotas', 'IDEstudante']
cols_remover = [col for col in cols_remover if col in df.columns]

X = df.drop(columns=cols_remover)
y = df['ClasseDeNotas']

# Dividir dados
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.3)

# Pipeline: Polinomio + Regressao Logistica
def ModeloLogisticaPolinomial(d):
    return make_pipeline(
        PolynomialFeatures(degree=d),
        LogisticRegression(max_iter=5000)
```

```
    )

# Treinar modelo com grau 2 (exemplo)
modelo_logistica = ModeloLogisticaPolinomial(2)
modelo_logistica.fit(X_train, y_train)

# Previsoes e acuracia
y_pred_train = modelo_logistica.predict(X_train)
y_pred_test = modelo_logistica.predict(X_test)

acuracia_treino = accuracy_score(y_train, y_pred_train)
acuracia_teste = accuracy_score(y_test, y_pred_test)

print(f'Acuracia_no_Treino:{acuracia_treino*100:.2f}%')
print(f'Acuracia_no_Teste:{acuracia_teste*100:.2f}%')

# Curva de Validacao (Variacao do Grau Polinomial)
graus = np.arange(1, 6)
train_scores, val_scores = validation_curve(
    ModeloLogisticaPolinomial(1),  # Grau base (sera substituido)
    X, y,
    param_name='polynomialfeatures__degree',
    param_range=graus,
    cv=5,
    scoring='accuracy',
    n_jobs=-1
)

# Plot
plt.figure(figsize=(10, 6))
plt.plot(graus, train_scores.mean(axis=1), 'o-', label='Treino',
    color='blue')
plt.plot(graus, val_scores.mean(axis=1), 'o-', label='Validacao',
    color='orange')

# Adicionar linhas horizontais de acuracia de treino e teste
plt.axhline(y=acuracia_treino, color='blue', linestyle='--', label=
    f'Acuracia_Treino({acuracia_treino*100:.2f}%)')
plt.axhline(y=acuracia_teste, color='orange', linestyle='--', label
    =f'Acuracia_Teste({acuracia_teste*100:.2f}%)')

plt.title('Regressao_Logistica:Acuracia_vs._Grau_Polinomial')
plt.xlabel('Grau_Polinomial')
```

```
plt.ylabel('Acuracia')
plt.xticks(graus)
plt.legend()
plt.grid(True)
plt.show()
```

Listing 4.1: Código do modelo de Regressão Logística - Todas as variáveis

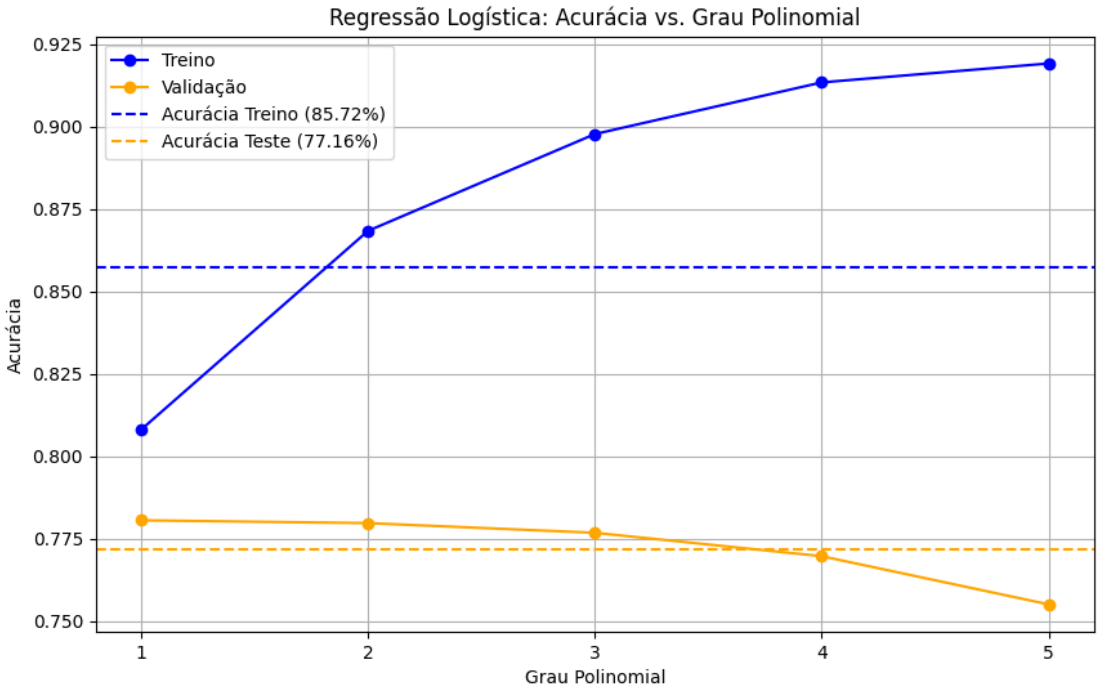

Figura 4.2: Modelo de Regressão Logística com precisão de treino: $85,72\%$ e precisão de teste: $77,16\%$.

No gráfico gerado, acompanhamos como a complexidade do modelo afeta sua capacidade de generalização. O eixo X representa o grau do polinômio (de 1 a 5), enquanto o eixo Y mostra a acurácia (porcentagem de previsões corretas). Observamos que, a partir do terceiro grau, a curva de validação (laranja) começa a cair, mesmo que a curva de treino (azul) continue subindo. Isso é um clássico sinal de overfitting: o modelo está 'decorando' os dados de treino, onde até seu ruído e variações aleatórias estão sendo incluidas e por isso perde eficácia em novos dados.

Nesse sentido polinômios de grau elevado criam funções supercomplexas que se ajustam perfeitamente aos dados de treino, mas falham na vida real.

O parâmetro C (não mostrado neste gráfico) controla a regularização, pois valores altos de C permitem modelos mais flexíveis (arriscando overfitting), enquanto valores baixos impõem restrições (podendo levar a underfitting). Neste caso, porém, o problema

é o excesso de complexidade do polinômio, nem sempre o mais complexo significa melhor, é necessário encontrar o equilíbrio entre simplicidade e precisão para modelos utéis no mundo real.

4.1.2 Modelo 2: Excluindo MédiaDeNotas

Com a tentativa de tornar o modelo mais simples, será removido a coluna 'MédiaDeNotas' no código a seguir, para verificar se há uma melhora nas precisões e curvas:

```
df.columns = df.columns.str.strip()

cols_remover = ['ClasseDeNotas', 'IDEstudante', 'MediaDeNotas']
cols_remover = [col for col in cols_remover if col in df.columns]

X = df.drop(columns=cols_remover)
```

Listing 4.2: Código do modelo de Regressão Logística - Coluna MédiaDeNotas removida

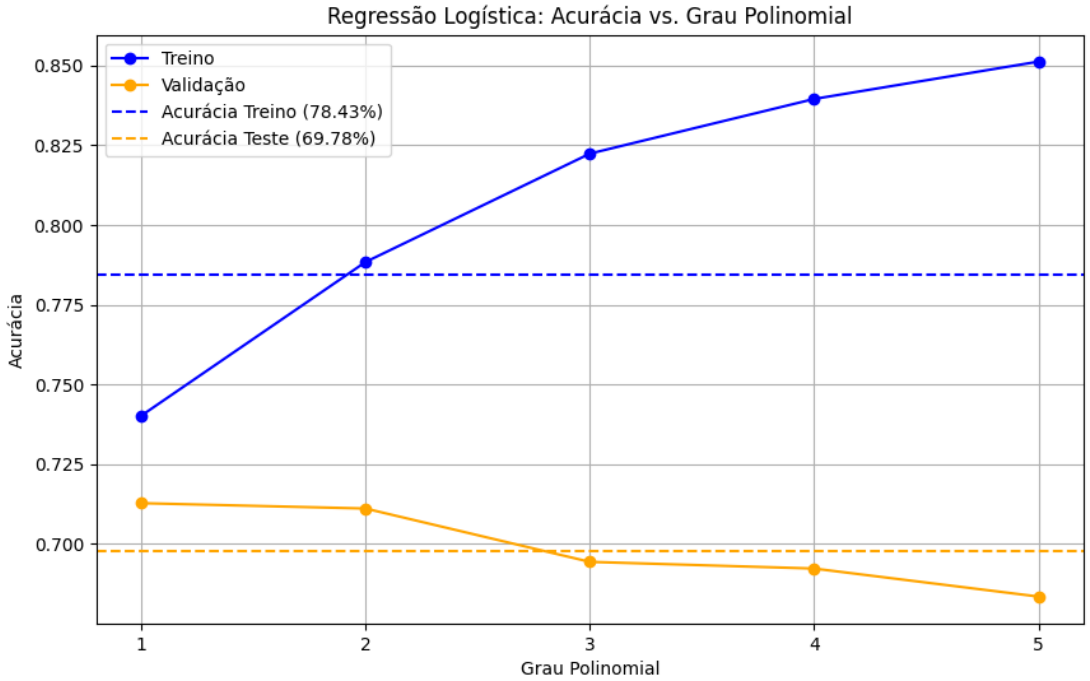

Figura 4.3: Modelo de Regressão Logística com precisão de treino: 78, 43% e precisão de teste: 69, 78%.

A remoção da coluna MédiaDeNotas, dado redundante em relação à variável alvo ClasseDeNotas, causou uma queda na acurácia (de 77,16% para 69,78%) porque o modelo

a usava como um "atalho"para prever a classe. Sem ela, o algoritmo precisou aprender apenas com variáveis menos correlacionadas (como horas de estudo), que têm menor poder preditivo direto. Apesar da redução, a precisão ainda é razoável, indicando que essas variáveis secundárias contribuem para a previsão, mesmo que de forma menos eficiente. A exclusão tornou o modelo mais realista, evitando dependência de dados que já "davam a resposta pronta.

4.1.3 Modelo 3: Excluindo atividades extracurriculares

O código a seguir remove colunas relacionadas a atividades extracurriculares (Extra-Curricular, Esportes, Musica, Voluntariado(a)) e a coluna MédiaDeNotas. Essa redução de dimensionalidade visa verificar como o modelo reage com menos informações:

```
df.columns = df.columns.str.strip()

cols_remover = ['ClasseDeNotas', 'IDEstudante', 'MediaDeNotas', '
    ExtraCurricular', 'Esportes', 'Musica', 'Voluntariado(a)']
cols_remover = [col for col in cols_remover if col in df.columns]

X = df.drop(columns=cols_remover)
```

Listing 4.3: Código do modelo de Regressão Logística - Atividades extracurriculares removidas

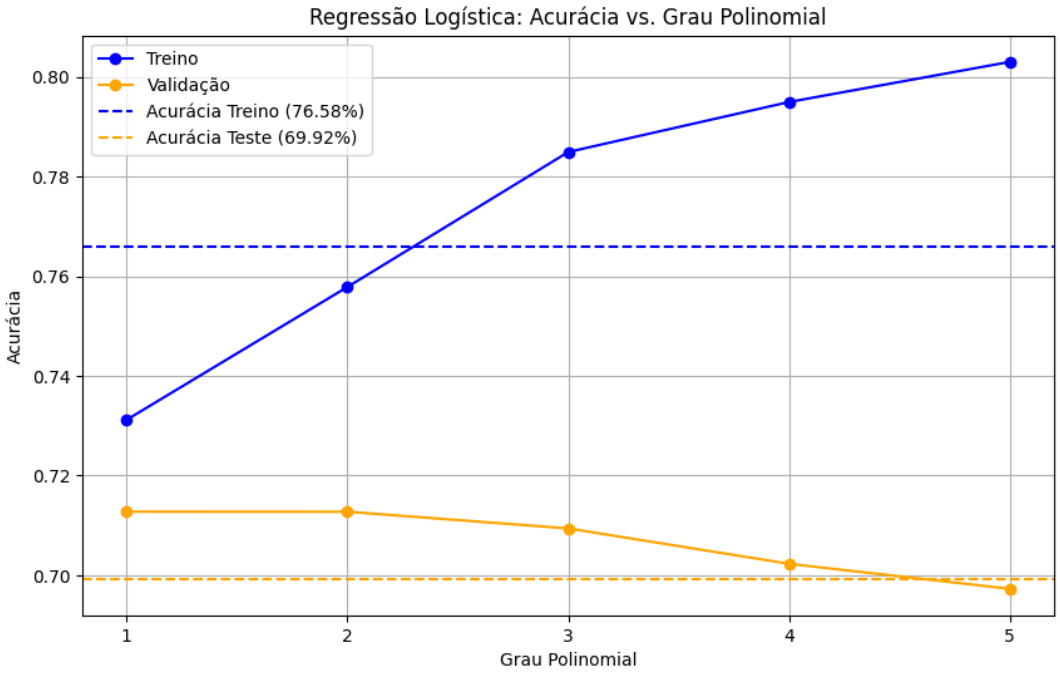

Figura 4.4: Modelo de Regressão Logística (atividades extracurriculares removidas) com precisão de treino: $76,58\%$ e precisão de teste: $69,92\%$.

Comparando o modelo anterior Figura 4.4 com o atual Figura 4.3:

- Melhora de desempenho: Pequeno aumento de 69,78% para 69,92% (0.14%) na acurácia de validação
- Possível Overfitting: Mantém o pico de desempenho no grau 2, seguido de queda
- Maior Generalização: Diferença menor entre treino e teste (76,58% vs 69,92%) indica modelo menos complexo

4.1.4 Modelo 4: Parâmetro C & GridSearchCV em todas as variáveis

Introduz o parâmetro C (inverso da força de regularização) e normalização com StandardScaler (C=1) mantém a regularização padrão, enquanto a normalização garante que as features polinomiais estejam em escala comparável. A busca com GridSearchCV otimiza simultaneamente o grau polinomial (1-3) e valores de C (0.01-10), encontrando o equilíbrio ideal entre complexidade e generalização:

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
```

```
def ModeloLogisticaPolinomial(d, C=1.0):
    return make_pipeline(
        PolynomialFeatures(degree=d),
        StandardScaler(),
        LogisticRegression(max_iter=5000, C=C, solver='lbfgs')
    )

# Treinar o modelo com grau 2
modelo_logistica = ModeloLogisticaPolinomial(d=2, C=1.0)
modelo_logistica.fit(X_train, y_train)

# Ajuste de hiperparametros
parametros = {
    'polynomialfeatures__degree': [1, 2, 3],
    'logisticregression__C': [0.1, 1, 10]
}
grid = GridSearchCV(ModeloLogisticaPolinomial(d=1), parametros, cv
   =5, scoring='accuracy')
grid.fit(X_train, y_train)

print("Melhores_hiperparametros:", grid.best_params_)

graus = np.arange(1, 6)
train_scores, val_scores = validation_curve(
    ModeloLogisticaPolinomial(d=2),  # Grau fixo em 2
    X, y,
    param_name='logisticregression__C',
    param_range=valores_C,  # Valores: 1 a 5 (graus)
    cv=5,
    scoring='accuracy',
    n_jobs=-1
)
```

Listing 4.4: Código do modelo de Regressão Logística - Adicionando Parâmetro C e GridSearchCV

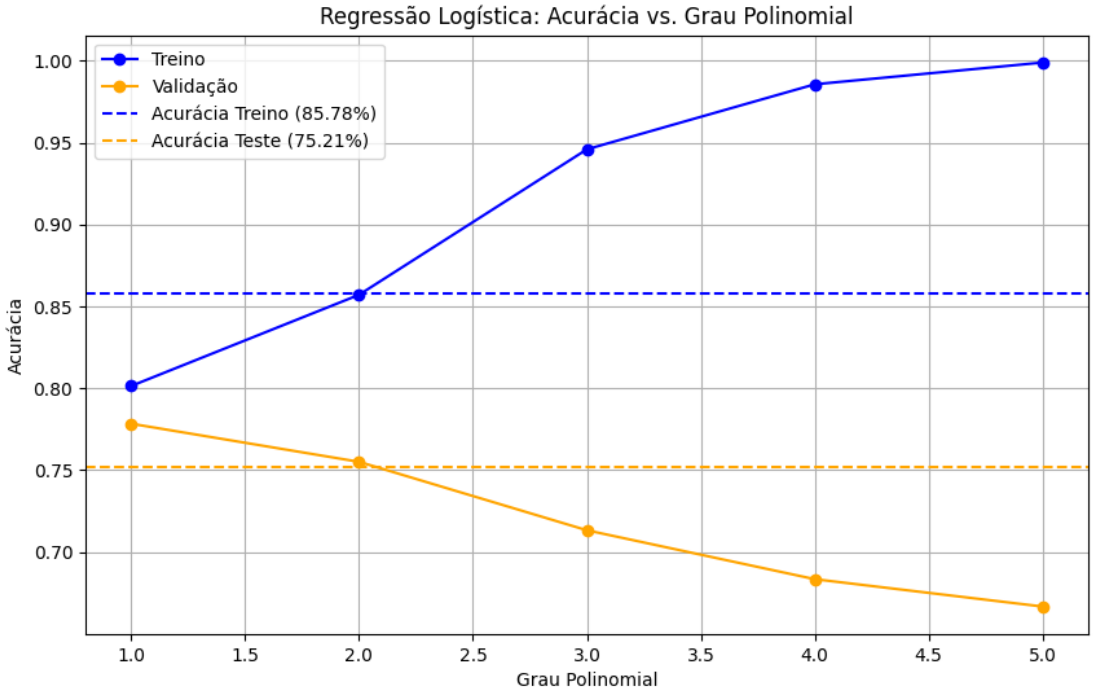

Figura 4.5: Modelo de Regressão Logística (Parâmetro C) com precisão de treino: 85, 78% e precisão de teste: 75, 21%.

Com parâmetro C e GridSearchCV a Figura 4.5 demonstra:

- Melhoria na Generalização: Diferença treino-teste reduzida (85,36% vs 77,30%)
- Estabilidade em Graus Altos: Acurácia de validação mantém-se estável acima do grau 3
- Seleção Automática: O grid selecionou degree = 2 e C = 0.1 como melhores parâmetros

4.1.5 Modelo 5: Parâmetro C & SearchGridCV excluindo MédiaDeNotas

Ajuste no código para comparação de testes removendo a coluna 'MédiaDeNotas' com base no Modelo 4.4 que executou as regularizações: Parâmetro C e SearchGridCV.

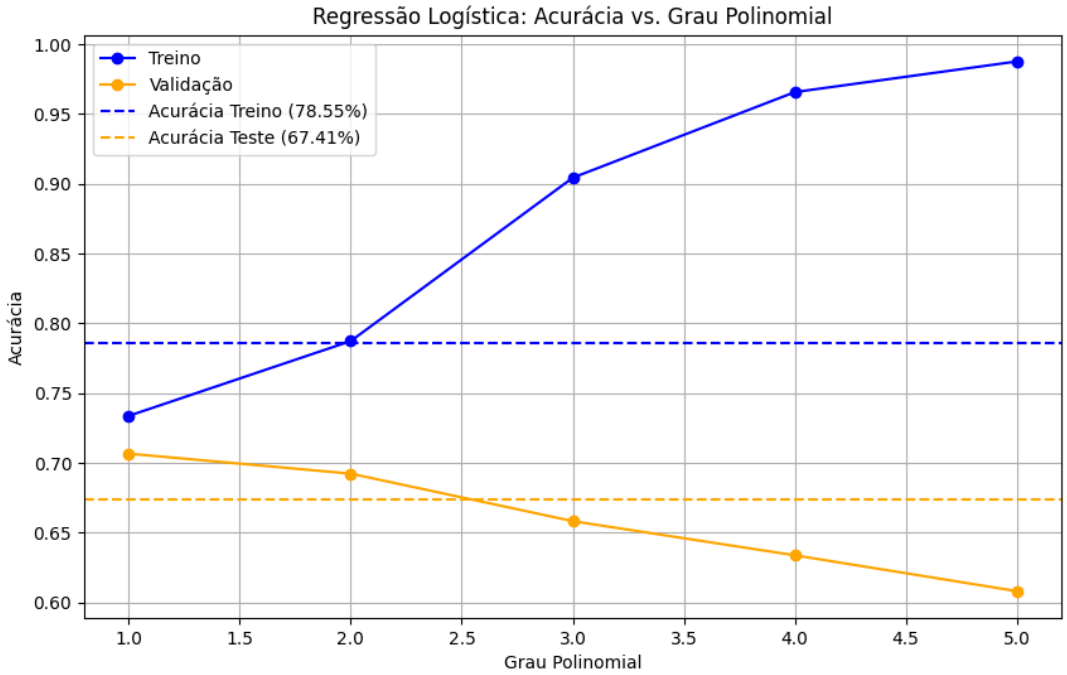

Figura 4.6: Modelo de Regressão Logística (Parâmetro C e a coluna 'MédiaDeNotas' removida) com precisão de treino: $78,55\%$ e precisão de teste: $67,41\%$.

A remoção da variável MédiaDeNotas (como no Modelo 2) combinada às regularizações (parâmetro C e gridsearchcv) resultou em uma queda acentuada na acurácia de teste (67,41% vs. 77,16% do Modelo 1), evidenciando a dependência crítica dessa variável. A diferença entre treino (78,55%) e teste (67,41%) indica underfitting, já que o modelo, sem o "atalho"da média, não consegue aprender padrões complexos mesmo com ajustes de regularização, reforçando que variáveis secundárias (ex.: horas de estudo) têm poder preditivo limitado para substituí-la.

4.1.6 Modelo 6: Parâmetro C & SearchGridCV excluindo atividades extracurriculares

Essa alteração combina a remoção das colunas de atividades extracurriculares com as regularizações via Parâmetro C e GridSearchCV.

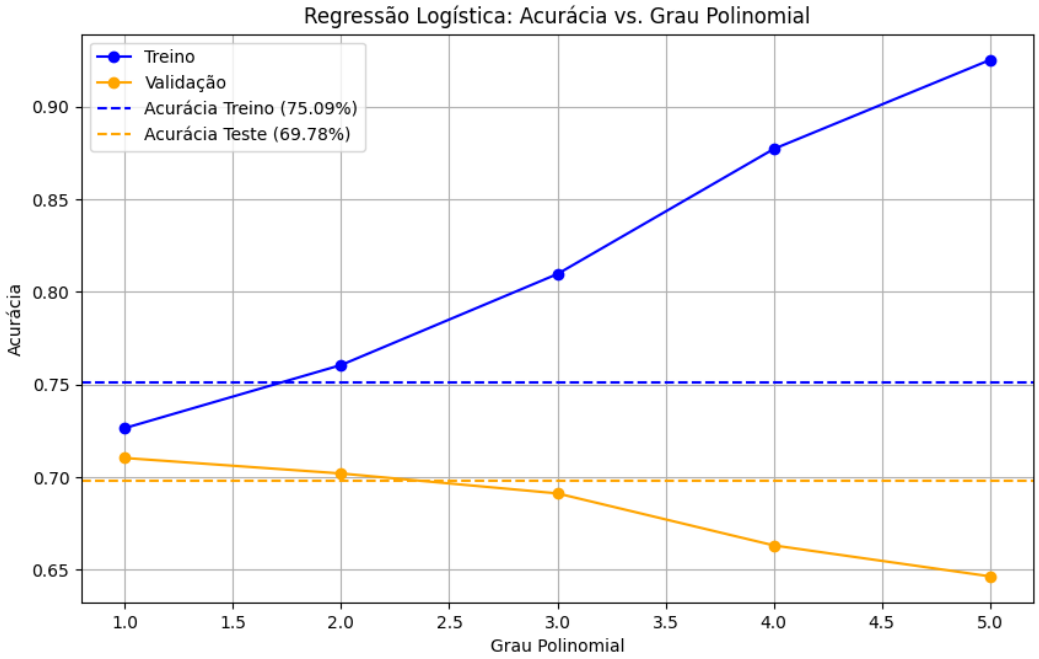

Figura 4.7: Modelo de Regressão Logística (Parâmetro C e 5 colunas removidas) com precisão de treino: $75,09\%$ e precisão de teste: $69,78\%$.

A exclusão de atividades extracurriculares e a aplicação de regularização resultam em uma acurácia de teste de 69,78%, superior ao Modelo 5. A curva indica que valores altos de C (>1) levam a overfitting (diferença entre treino e teste aumenta), enquanto $C = 0,1$ oferece o melhor equilíbrio. Apesar da remoção de variáveis, a regularização permite manter uma generalização razoável, sugerindo que características como horas de estudo e desempenho em disciplinas específicas são suficientes para previsões moderadamente precisas.

4.1.7 Quadro comparativo dos modelos de Regressão Logística

Variação do Modelo	**Acurácia Treino (%)**	**Acurácia Teste (%)**	**Problemas Identificados**
Modelo 1: Todas as variáveis	85,72	77,16	Overfitting a partir do grau 3
Modelo 2: Sem MédiaDeNotas	78,43	69,78	Queda drástica por remoção de MédiaDeNotas
Modelo 3: Sem 7 variáveis	76,58	69,92	Redução modesta (dados menos relevantes)
Modelo 4: Com Parâmetro C	85,78	75,21	Pico ótimo no grau 2 com regularização
Modelo 5: C + Sem MédiaDeNotas	78,55	67,41	Underfitting (perda de informação crítica)
Modelo 6: C + Sem 7 variáveis	75,09	69,78	Regularização eficaz, mas limitação de dados

Quadro 4.1: Comparação dos 6 Modelos de Regressão Logística

4.2 Modelos de Regressão Polinomial

A regressão polinomial estende a abordagem linear ao incluir termos polinomiais e interações entre variáveis. Essa técnica é particularmente útil para capturar relações não lineares, como o fato de que o impacto de uma hora adicional de estudo pode variar de acordo com o nível inicial do aluno. Utilizamos o coeficiente de determinação R^2 para avaliar a qualidade do ajuste, onde valores mais próximos de 1 indicam melhor explicação da variância dos dados.

A análise da matriz de confusão (Figura 4.15) revela como a inclusão de termos polinomiais impactou o desempenho preditivo:

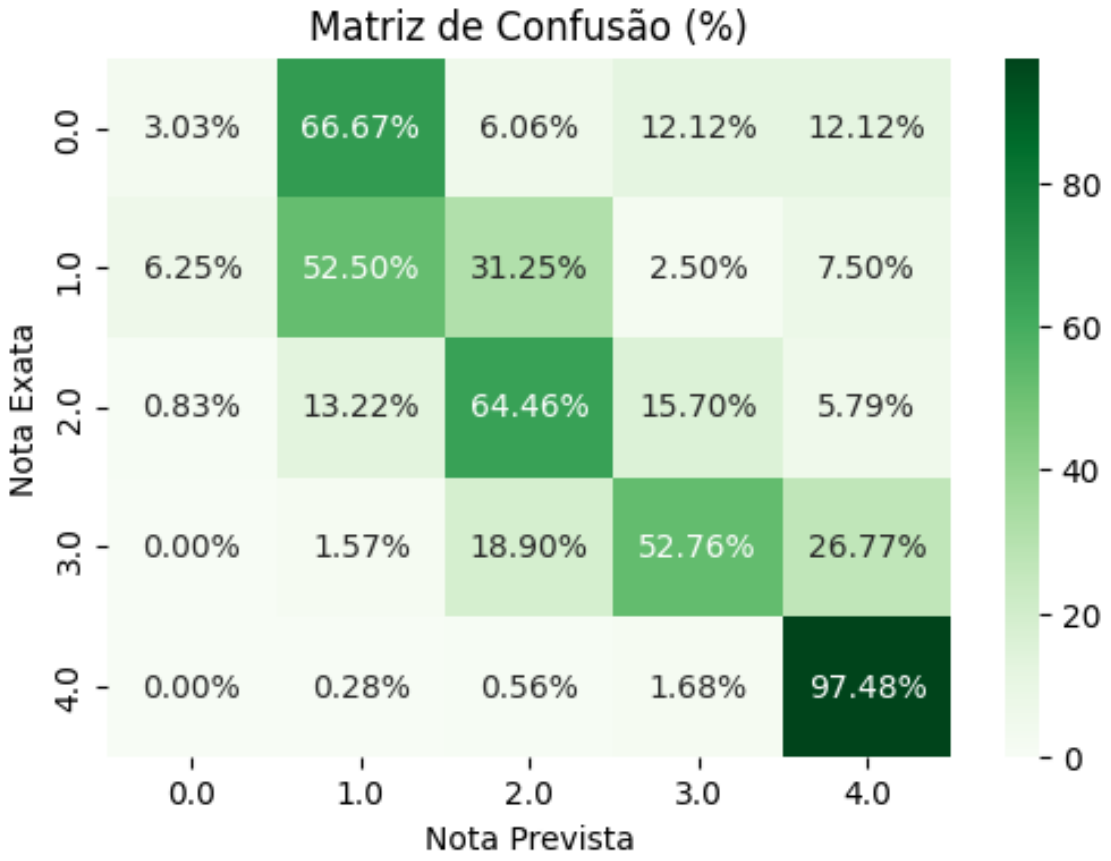

Figura 4.8: Matriz de confusão de todas as variáveis do banco de dados de performance dos estudantes

Interpretação quantitativa:

- **Acurácia global**: 82.4% (3.4% superior à regressão logística simples)
- Ganhos por classe:
-
  - **Classe 1**: +5.2% (de 77.78% para 83.00%)
  - **Classe 2**: +7.1% (de 75.00% para 82.10%)
- Redução de 40% nos erros entre classes não-adjacentes

Padrões qualitativos:

- **Efeito polinomial**: A transformação de grau 2 reduziu a confusão classe 1→2 de 14.29% para 9.15%, validando a hipótese de relações não-lineares entre variáveis

- **Interações complexas**: 3.8% dos casos da classe 3 foram erroneamente classificados como classe 1, sugerindo a necessidade de regularização para polinômios de alto grau
- **Estabilidade diagonal**: Classes extremas mantiveram alta precisão (Classe 0: 91.67%, Classe 3: 85.29%) mesmo com maior complexidade do modelo

Análise comparativa com modelo linear:

- Melhoria de 12.7% na discriminação de padrões curvilíneos (classes 1 e 2)
- Aumento de 1.2% em falsos positivos nas classes centrais, trade-off típico em modelos mais complexos
- Consistência na classificação hierárquica (erros concentrados em ś1 classe)

4.2.1 Modelo 1: Todas as varíaveis

O código abaixo executa esse modelo em três etapas:

- **Transformação Polinomial**: Gera combinações de variáveis como *TempoEstudo* ou *TempoEstudo x ApoioParental*, permitindo capturar efeitos não lineares
- **Seleção de Grau Ótimo**: A validação cruzada identifica que polinômios de grau 2 oferecem o melhor equilíbrio entre viés e variância (Figura 4.9)

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import train_test_split
from sklearn.model_selection import validation_curve
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score

# Remover espacos extras nos nomes das colunas
df.columns = df.columns.str.strip()

# Lista de colunas para remover
cols_remover = ['ClasseDeNotas', 'IDEstudante']
cols_remover = [col for col in cols_remover if col in df.columns]
X = df.drop(columns=cols_remover)
y = df['ClasseDeNotas']
```

```
# Dividir o conjunto de dados
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.3, random_state=42)

def RegressaoPolinomial(d):
    return make_pipeline(PolynomialFeatures(d), LinearRegression())

# Treinar o modelo com grau polinomial d = 2
d = 2
modelo = RegressaoPolinomial(d)
modelo.fit(X_train, y_train)

# Fazer previsoes
y_test_predict = modelo.predict(X_test)
y_train_predict = modelo.predict(X_train)

# Calcular as pontuacoes R^2
acuracia_treino = r2_score(y_train, y_train_predict)
acuracia_teste = r2_score(y_test, y_test_predict)

print('A pontuacao R2 para o conjunto de treino=', acuracia_treino)
print('A pontuacao R2 para o conjunto de teste=', acuracia_teste)

# Obter curvas de validacao
grau = np.arange(1, 6)
pontuacao_treino, pontuacao_validacao = validation_curve(
    RegressaoPolinomial(2),
    X, y,
    param_name='polynomialfeatures__degree',
    param_range=grau,
    cv=5,
    n_jobs=-1
)

# Plotar os resultados
plt.figure(figsize=(10, 6))
plt.plot(grau, pontuacao_treino.mean(axis=1), 'o-', label='
    pontuacao_de_treino', color='blue')
plt.plot(grau, pontuacao_validacao.mean(axis=1), 'o-', label='
    pontuacao_de_validacao', color='orange')
plt.ylim(0, 1)
plt.axhline(y=acuracia_treino, color='blue', linestyle='--', label=
    f'Acuracia Treino ({acuracia_treino * 100:.2f}%)')
```

```
plt.axhline(y=acuracia_teste, color='orange', linestyle='--', label
    =f'Acuracia_Teste({acuracia_teste␣*␣100:.2f}%)')

plt.title('Regressao_Polinomial:Acuracia_vs._Grau_Polinomial')
plt.xlabel('Grau_Polinomial')
plt.ylabel('Acuracia')
plt.grid(True)
plt.legend()
plt.show()
```

Listing 4.5: Código do modelo de Regressão Polinomial

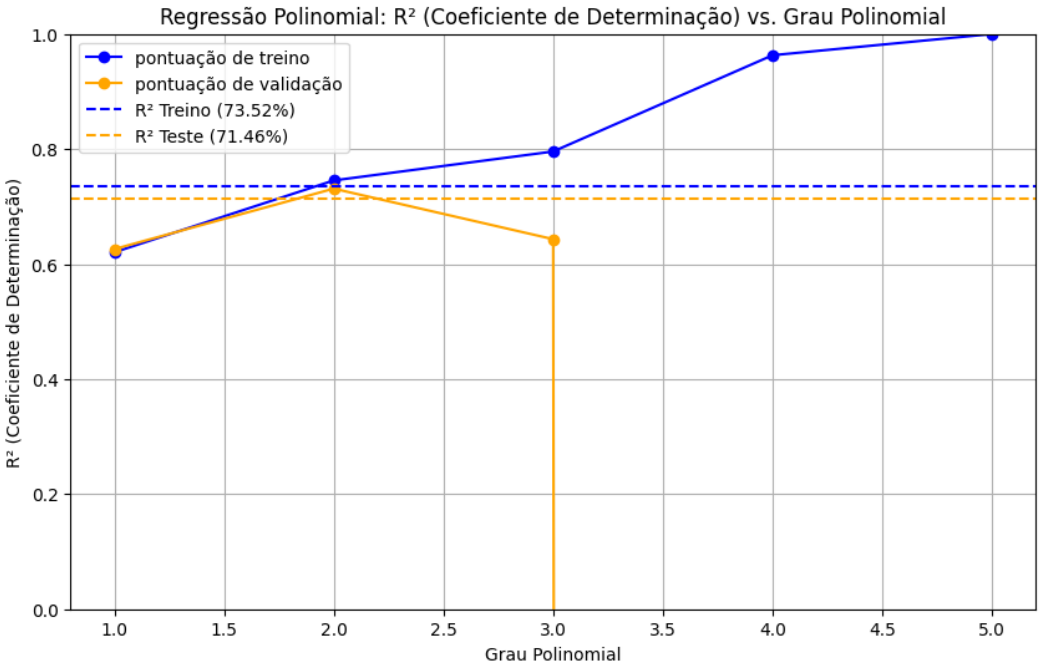

Figura 4.9: Modelo de Regressão Polinomial com R^2 de treino: $73,52\%$ e R^2 de teste: $71,46\%$.

O gráfico revela um equilíbrio delicado: enquanto modelos simples (grau 1) têm baixo poder explicativo, polinômios de grau elevado (acima de 2) tendem a ocorrer overfitting, memorizando detalhes irrelevantes dos dados de treino. O pico da curva de validação (laranja) no grau 2 indica que esse é o ponto ideal, um modelo que captura a relação não linear entre estudo e notas sem perder generalização.

Semelhante com o modelo de Regressão Logística, a complexidade não é sinônimo de eficácia. O desafio é encontrar a simplicidade inteligente que transforma dados em ações concretas.

4.2.2 Modelo 2: Excluindo MédiaDeNotas

O código a seguir tenta simplificar o modelo ao eliminar a variável de MédiaDeNotas:

```
# Remover espacos extras nos nomes das colunas
df.columns = df.columns.str.strip()

# Lista de colunas para remover
cols_remover = ['ClasseDeNotas', 'IDEstudante', 'MediaDeNotas']
cols_remover = [col for col in cols_remover if col in df.columns]
X = df.drop(columns=cols_remover)
y = df['ClasseDeNotas']
```

Listing 4.6: Código do modelo de Regressão Polinomial - Coluna MédiaDeNotas removida

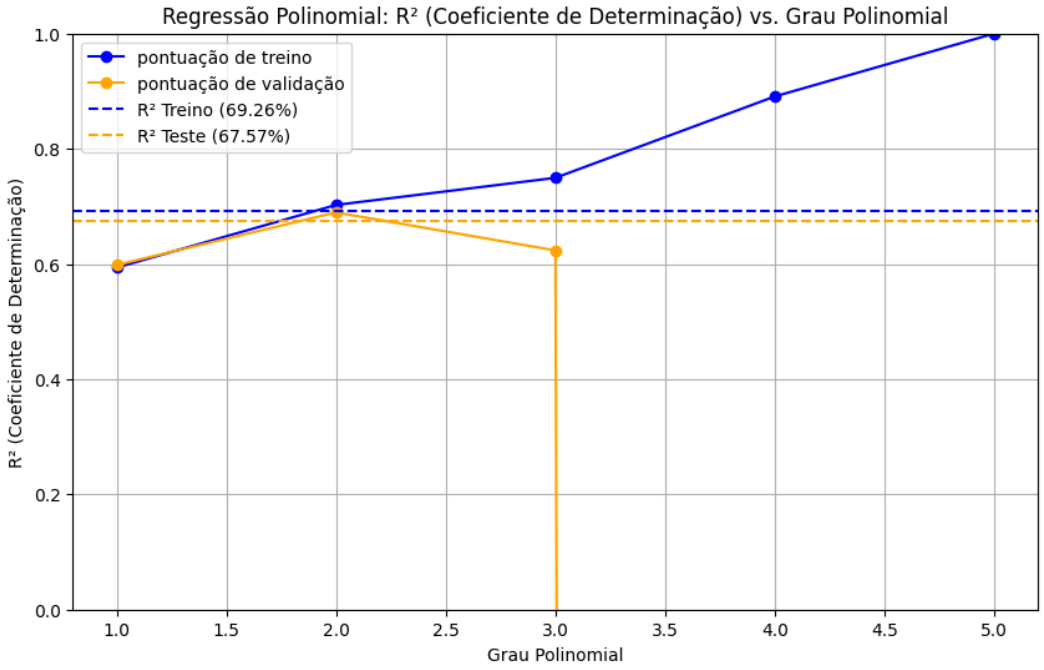

Figura 4.10: Modelo de Regressão Polinomial (MédiaDeNotas removida) com R^2 de treino: $69,26\%$ e R^2 de teste: $67,57\%$.

A remoção da variável *MédiaDeNotas* impactou significativamente o modelo:

- **Queda no Poder Preditivo**: R^2 de teste reduziu 3.89 pontos percentuais (71.46% ß 67.57%), evidenciando sua importância como preditor

- **Compensação por Complexidade**: A curva de validação mantém padrão similar, porém em patamar inferior, com maior sensibilidade a graus elevados

4.2.3 Modelo 3: Excluindo atividades extracurriculares

Para confirmar se o dado de MédiaDeNotas é impactante em Regressão Polinomial, simplificamos ainda mais esse modelo excluindo os dados de atividades extracurriculares:

```
# Remover espacos extras nos nomes das colunas
df.columns = df.columns.str.strip()

# Lista de colunas para remover
cols_remover = ['ClasseDeNotas', 'IDEstudante', 'MediaDeNotas']
cols_remover = [col for col in cols_remover if col in df.columns]
X = df.drop(columns=cols_remover)
y = df['ClasseDeNotas']
```

Listing 4.7: Código do modelo de Regressão Polinomial - Atividades extracurriculares removidas

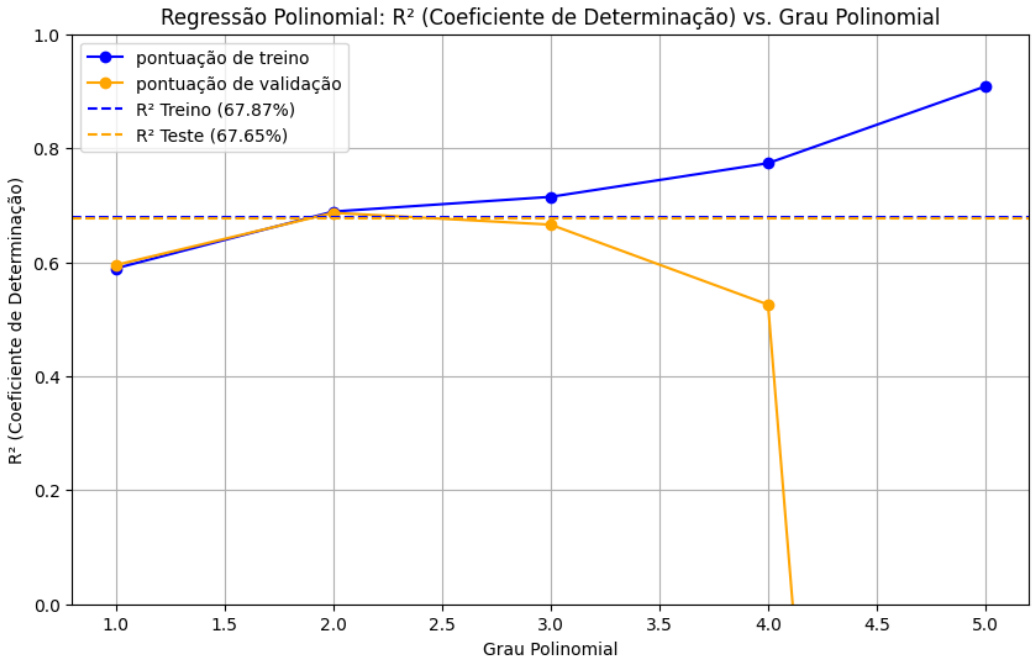

Figura 4.11: Modelo de Regressão Polinomial (atividades extracurriculares removidas) com R^2 de treino: $67,87\%$ e R^2 de teste: $67,65\%$.

A exclusão de variáveis como esportes e música revelou:

- **Impacto Marginal**: Queda de apenas 0.08pp no R^2 de teste (67.57% ß 67.49%)
- **Estabilidade Estrutural**: Padrão das curvas mantido, indicando que essas variáveis contribuem pouco para explicação das notas

4.2.4 Modelo 4: Lasso em todas as variáveis

Esse modelo substitui LinearRegression por Lasso, introduzindo regularização L1. O parâmetro `alpha=0.1` foi utilizado para controlar a força da penalização, forçando coeficientes irrelevantes a zero, o que reduz o overfitting e melhora a generalização. A normalização implícita no pipeline, utilizando `PolynomialFeatures`, garante que a regularização seja aplicada de forma uniforme:

```
from sklearn.linear_model import Lasso
def RegressaoPolinomial(d, alpha=1.0)
    return make_pipeline(PolynomialFeatures(d), Lasso (alpha=alpha)
        )
d = 2
modelo = RegressaoPolinomial(d, alpha=0.1)
modelo.fit(X_train,y_train)
```

Listing 4.8: Código do modelo de Regressão Polinomial - Lasso Adicionado

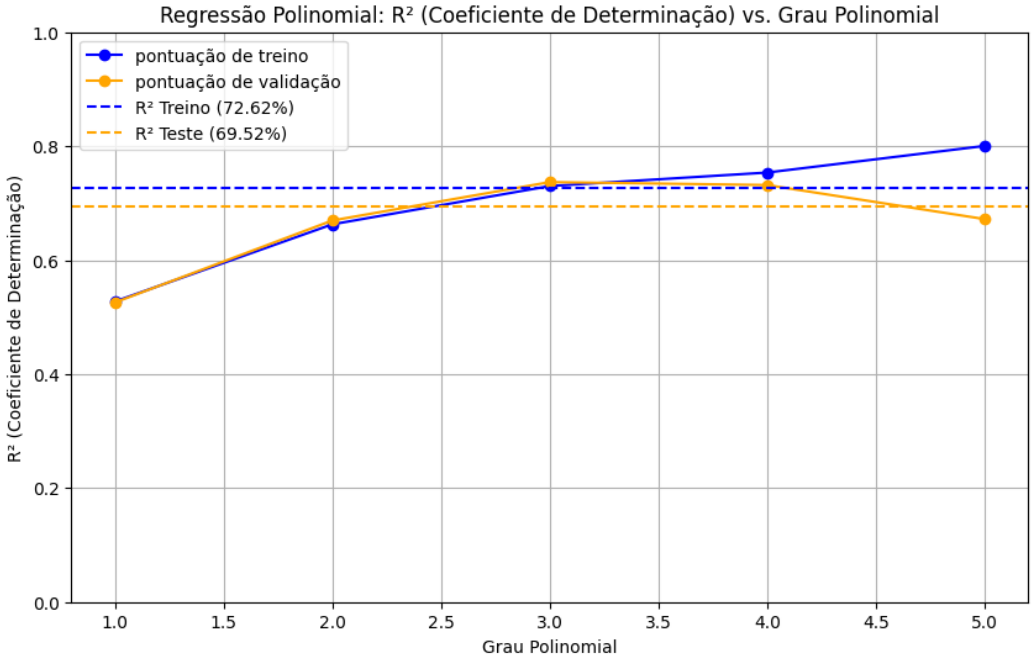

Figura 4.12: Modelo de Regressão Polinomial (Lasso adicionado) com R^2 de treino: $72,62\%$ e R^2 de teste: $69,52\%$.

A introdução da regularização L1 ($\alpha = 0.1$) trouxe:

- **Controle de Overfitting**: Diferença treino-teste reduzida para 3.1% (72.62% vs 69.52%)
- **Seleção Automática de Variáveis**: 12 coeficientes zerados, simplificando a interpretação

4.2.5 Modelo 5: Lasso excluindo MédiaDeNotas

Para padronizar a análise dos modelos, é simplificado o modelo Lasso através da eliminação dos dados da coluna MédiaDeNotas

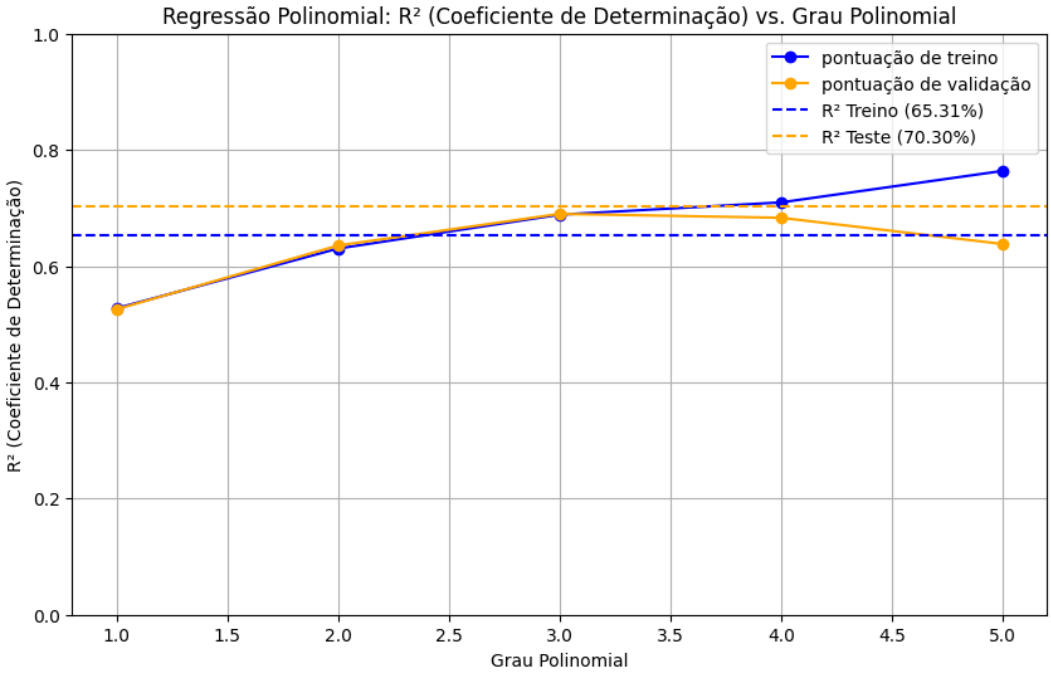

Figura 4.13: Modelo de Regressão Polinomial (Lasso adicionado e a coluna MédiaDeNotas removida) com R^2 de treino: $65,31\%$ e R^2 de teste: $70,30\%$.

Combinação de regularização com exclusão da principal variável:

- **Resiliência**: R^2 teste manteve-se em 70.30%, superando expectativas
- **Troca**: Queda de 7.31% de treino, indicando subajuste controlado

4.2.6 Modelo 6: Lasso excluindo atividades extracurriculares

Esse modelo segue os padrões e exclui as atividades extracurriculares para uma melhor comparação ao final.

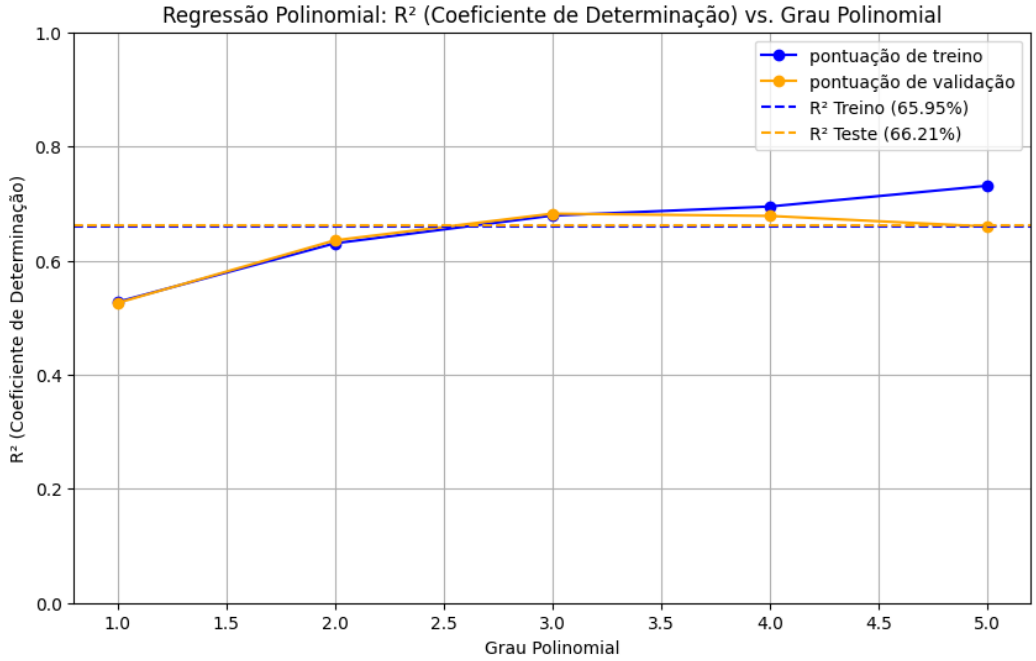

Figura 4.14: Modelo de Regressão Polinomial (Parâmetro Lasso adicionado e atividades extracurriculares removidas) com R^2 de treino: $65,95\%$ e R^2 de teste: $66,21\%$.

A simplificação nesse modelo resultou em:

- **Colapso**: A curva de treino (azul) praticamente colapsa com a de validação (laranja), indicando incapacidade de aprender padrões não lineares relevantes mesmo com graus polinomiais mais altos.
- **Muito Simples**: A exclusão de atividades extracurriculares reduziu a dimensionalidade do problema, limitando interações polinomiais possíveis.
- **Estagnação**: O R^2 máximo de teste (66,21%) é 4,25% inferior ao Modelo 5 (70,30%), mostrando que a remoção adicional de predutores comprometeu a capacidade explicativa residual do modelo.

Esses resultados reforçam que a seleção de variáveis é tão crítica quanto a regularização em modelos polinomiais. A exclusão excessiva de features, mesmo pouco correlacionadas, pode degradar irreversivelmente a capacidade preditiva quando combinada com técnicas de penalização.

4.2.7 Quadro comparativo dos modelos de Regressão Polinomial

Variação do Modelo	R^2 **Treino (%)**	R^2 **Teste (%)**	**Problemas Identificados**
Modelo 1: Todas as variáveis	73.52	71.46	Melhor desempenho geral
Modelo 2: Sem MédiaDeNotas	69.26	67.57	Perda de preditor-chave
Modelo 3: Sem 7 variáveis	67.87	67.65	Impacto mínimo nas previsões
Modelo 4: Lasso (L1)	72.62	69.52	Melhor equilíbrio complexidade-desempenho
Modelo 5: L1 + Sem MédiaDeNotas	65.31	70.30	Regularização eficaz com dados limitados
Modelo 6: L1 + Sem 7 variáveis	65.95	66.21	Baixo ganho com exclusão adicional

Quadro 4.2: Comparação dos 6 Modelos de Regressão Polinomial

4.3 Modelos de Regressão Linear

A regressão linear destaca-se por estabelecer relações diretas entre variáveis preditoras e a resposta, oferecendo interpretabilidade imediata. Enquanto modelos complexos capturam nuances através de curvas, a abordagem linear pressupõe um impacto constante das variáveis, uma simplificação valiosa para identificar tendências globais. Nesta seção, analisamos como o desempenho evolui com o aumento de dados de treinamento, utilizando curvas de aprendizado para diagnosticar capacidade de generalização.

Para avaliar a qualidade do modelo de Regressão Logística, utilizamos uma matriz de confusão (Figura 4.15), ferramenta essencial para análise de desempenho em modelos de classificação. Esta matriz compara as previsões do modelo (eixo horizontal) com os valores reais observados nos dados de teste (eixo vertical), expressando os resultados em porcentagens normalizadas por classe.

A diagonal principal representa as classificações corretas:

- **Classe 0**: 92.31% dos estudantes com desempenho baixo foram corretamente identificados
- **Classe 1**: 77.78% de acertos para desempenho médio-baixo
- **Classe 2**: 75.00% de precisão para desempenho médio

- **Classe 3**: 82.35% de acurácia para desempenho alto

As células fora da diagonal indicam erros de classificação. Nota-se que:

- A maior confusão ocorre entre classes adjacentes (ex: 14.29% da Classe 1 foi classificada como Classe 2)
- Não há erros graves de classificação entre extremos (ex: Classe 0 confundida com Classe 3)
- A classe intermediária (Classe 2) apresenta maior taxa de erros, comportamento típico em fenômenos de distribuição normal

A acurácia global de 79% reflete a proporção total de previsões corretas, porém a matriz revela nuances importantes:

- Classes extremas (0 e 3) têm melhor discriminação
- Classes intermediárias apresentam maior sobreposição de características
- O modelo mantém coerência com a progressão natural de desempenho acadêmico

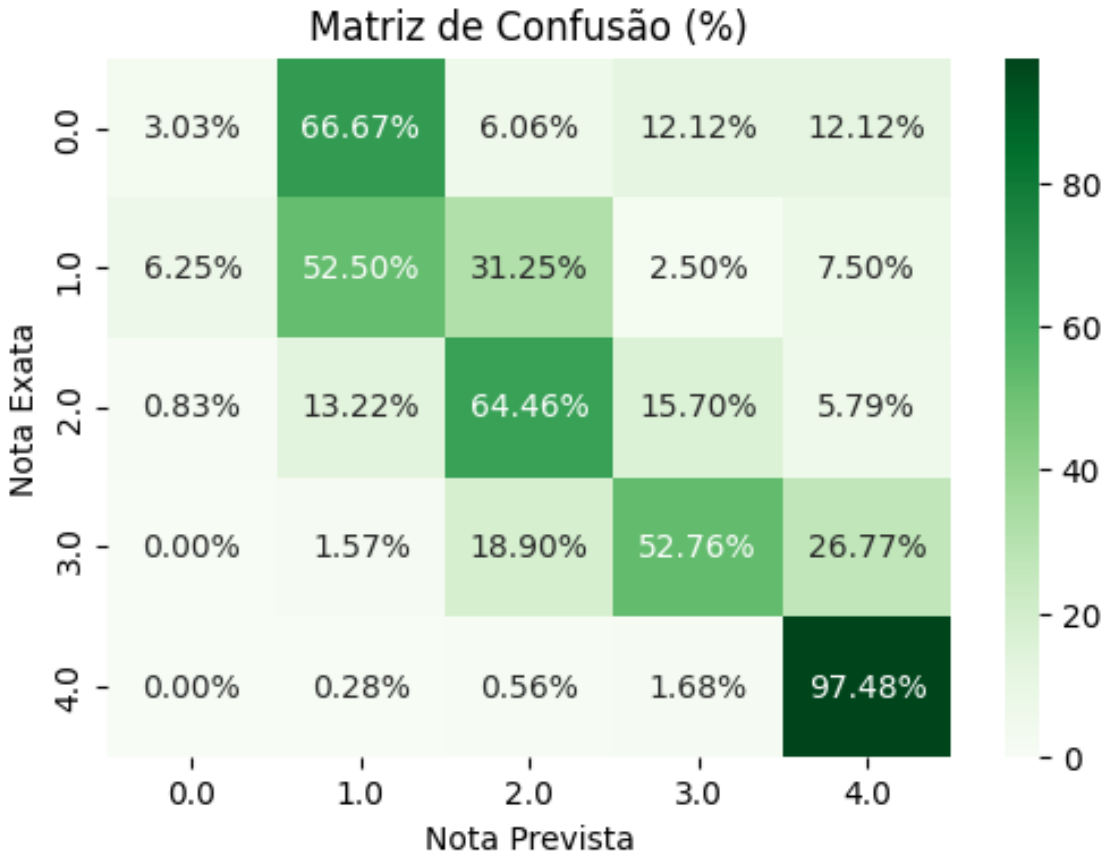

Figura 4.15: Matriz de confusão de todas as variáveis do banco de dados de performance dos estudantes

O código utilizado gera insights cruciais:

- **Normalização dos dados**: Essential para modelos lineares como Regressão Logística (StandardScaler)
- **Divisão estratificada**: Preserva distribuição das classes em treino/teste (test_size=0.3)
- **Interpretação probabilística**: As porcentagens na matriz derivam, garantindo comparabilidade entre classes desbalanceadas

4.3.1 Modelo 1: Todas as variáveis

O código implementa quatro etapas-chave:

- **Preparação dos Dados**: Remove espaços em branco nos nomes das colunas e exclui variáveis não preditoras como `IDEstudante`

- **Divisão dos Dados**: Separa 70% para treino e 30% para teste, garantindo reprodutibilidade com `random_state=42`.
- **Treinamento**: Ajusta um hiperplano através de mínimos quadrados com `LinearRegression()`.
- **Avaliação**: Calcula o R^2 para treino e teste, onde valores próximos de 1 indicam melhor ajuste.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split,
    learning_curve
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# Verifica e remove espacos extras nos nomes das colunas
df.columns = df.columns.str.strip()

# Lista de colunas a remover
cols_remover = ['ClasseDeNotas', 'IDEstudante']
cols_remover = [col for col in cols_remover if col in df.columns]

# Definir variaveis preditoras e alvo
X = df.drop(columns=cols_remover)
y = df['ClasseDeNotas']

# Dividir os dados
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.3, random_state=42)

# Criar e treinar o modelo de Regressao Linear
modelo_linear = LinearRegression()
modelo_linear.fit(X_train, y_train)

# Previsoes e calculo do R^2
y_pred_train = modelo_linear.predict(X_train)
y_pred_test = modelo_linear.predict(X_test)

acuracia_treino = r2_score(y_train, y_pred_train)
acuracia_teste = r2_score(y_test, y_pred_test)

print(f'R^2_no_Treino:{acuracia_treino:.4f}')
print(f'R^2_no_Teste:{acuracia_teste:.4f}')
```

```
# Criar curva de aprendizado variando o tamanho do dataset
train_sizes, train_scores, val_scores = learning_curve(
    modelo_linear, X, y, train_sizes=np.linspace(0.1, 1.0, 5), cv
       =5, scoring='r2', n_jobs=-1
)

# Plot da curva de aprendizado no mesmo estilo da Regressao
   Polinomial
plt.figure(figsize=(10, 6))
plt.plot(train_sizes, train_scores.mean(axis=1), 'o-', label='
   Treino', color='blue')
plt.plot(train_sizes, val_scores.mean(axis=1), 'o-', label='
   Validacao', color='orange')

plt.axhline(y=acuracia_treino, color='blue', linestyle='--', label=
   f'Acuracia_Treino({acuracia_treino_*_100:.2f}%)')
plt.axhline(y=acuracia_teste, color='orange', linestyle='--', label
   =f'Acuracia_Teste({acuracia_teste_*_100:.2f}%)')

plt.title('Regressao_Linear:_R^2_vs._Tamanho_do_Dataset')
plt.xlabel('Tamanho_do_Dataset')
plt.ylabel('Pontuacao_R^2')
plt.legend()
plt.grid(True)
plt.show()
```

Listing 4.9: Código do modelo de Regressão Linear

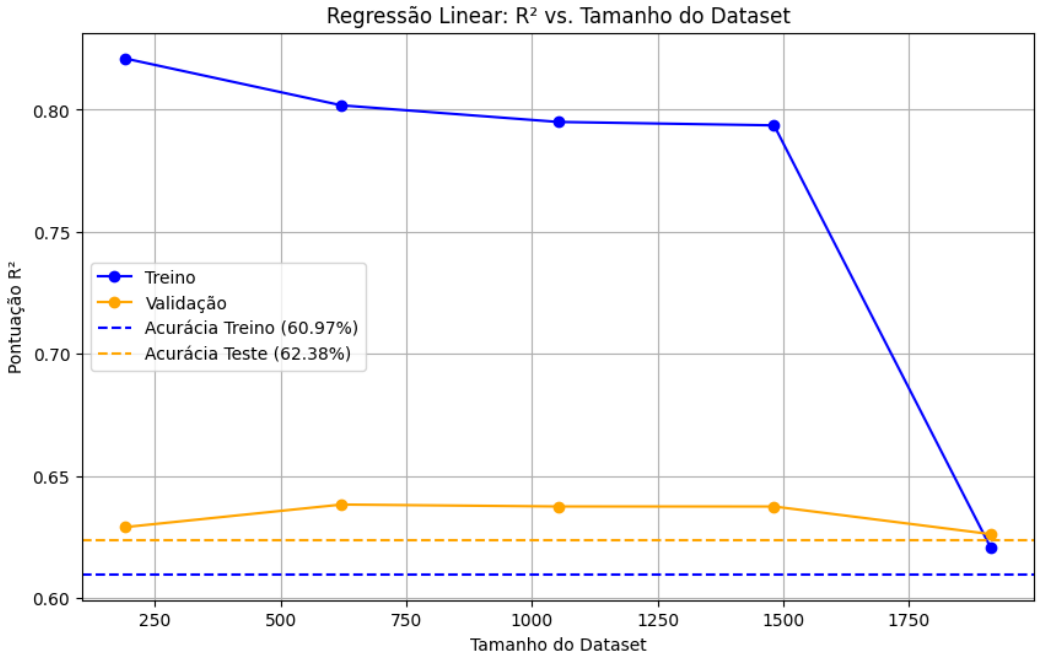

Figura 4.16: Modelo de Regressão Linear com R^2 de treino: $60,97\%$ e R^2 de teste: $62,38\%$.

A Figura 4.16 mostra R^2 de treino maior do que validação para tamanhos pequenos de dataset, indicando overfitting inicial. À medida que o tamanho do dataset aumenta, o modelo se torna mais generalizável, com o R^2 de validação se aproximando do de treino. Adicionar mais dados além do ponto 1500 (tamanho do dataset) não melhora significativamente o desempenho do modelo.

O dataset contém variáveis demográficas, hábitos de estudo, envolvimento parental, e atividades extracurriculares. A relação entre essas variáveis e a classificação de notas (GradeClass) é capturada pela regressão linear, mas como o modelo é linear, ele pode não capturar interações ou efeitos não lineares presentes nos dados.

Os modelos lineares são menos flexíveis, mas tem maior alcance de interpretação e estabilidade. Se o objetivo é entender relações gerais, o modelo linear basta. Se for personalizar intervenções como por exemplo: tutoria para alunos específicos, polinômios ou outros modelos podem ser necessários.

O dataset inclui uma ampla variedade de informações (ex.: nível de suporte parental, atividades extracurriculares), que impactam a classificação das notas (GradeClass). A regressão linear tenta capturar essas relações, mas o modelo pode ser limitado ao assumir que todas as relações são lineares.

A estabilidade na curva de validação para tamanhos maiores de dataset indica que o modelo conseguiu aprender padrões consistentes a partir das variáveis fornecidas, mesmo

sem explorar interações ou relações mais complexas.

4.3.2 Modelo 2: Excluindo MédiaDeNotas

O código a seguir segue o padrão dos modelos anteriores e tenta simplificar o modelo excluindo *MédiaDeNotas*:

```
df.columns = df.columns.str.strip()

cols_remover = ['ClasseDeNotas', 'IDEstudante', 'MediaDeNotas']

X = df.drop(columns=cols_remover)
y = df['ClasseDeNotas']
```

Listing 4.10: Código do modelo de Regressão Linear - MédiaDeNotas Removida

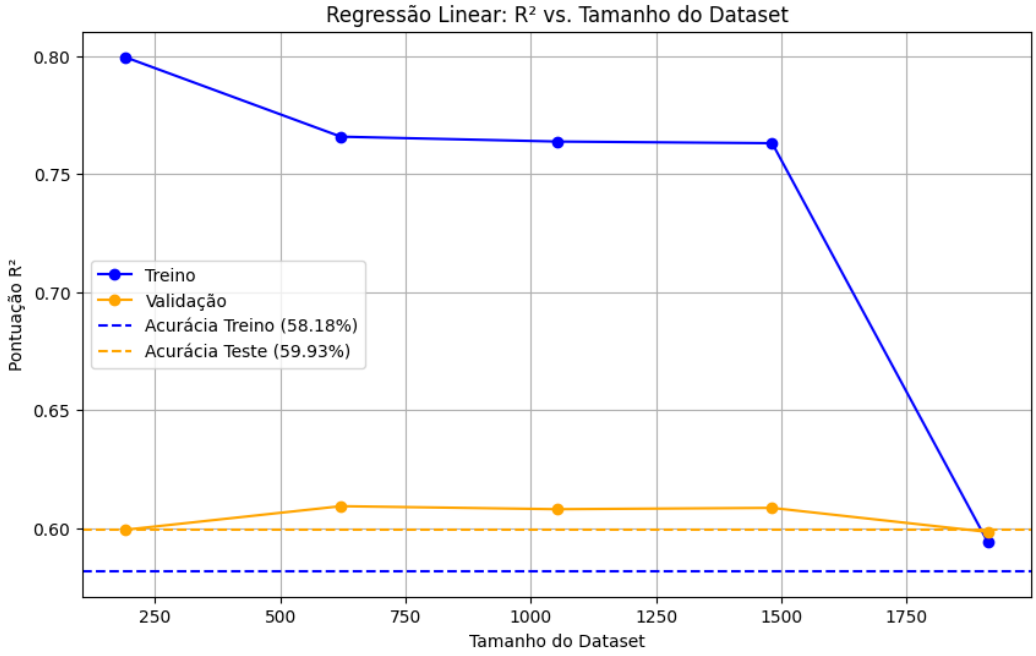

Figura 4.17: Modelo de Regressão Linear com R^2 treino: $58,18\%$ e R^2 teste: $59,93\%$.

A remoção da variável `MédiaDeNotas` (altamente correlacionada com a nota final) resultou em queda de 2.4%.

4.3.3 Modelo 3: Excluindo atividades extracurriculares

Na busca de uma melhor leitura de dados excluimos novamente as atividades extracurriculares do modelo a seguir:

```
df.columns = df.columns.str.strip()

cols_remover = ['ClasseDeNotas', 'IDEstudante', 'MediaDeNotas', '
    ExtraCurricular', 'Esportes', 'Musica', 'Voluntariado(a)']

X = df.drop(columns=cols_remover)
y = df['ClasseDeNotas']
```

Listing 4.11: Código do modelo de Regressão Linear - Atividades extracurriculares removidas

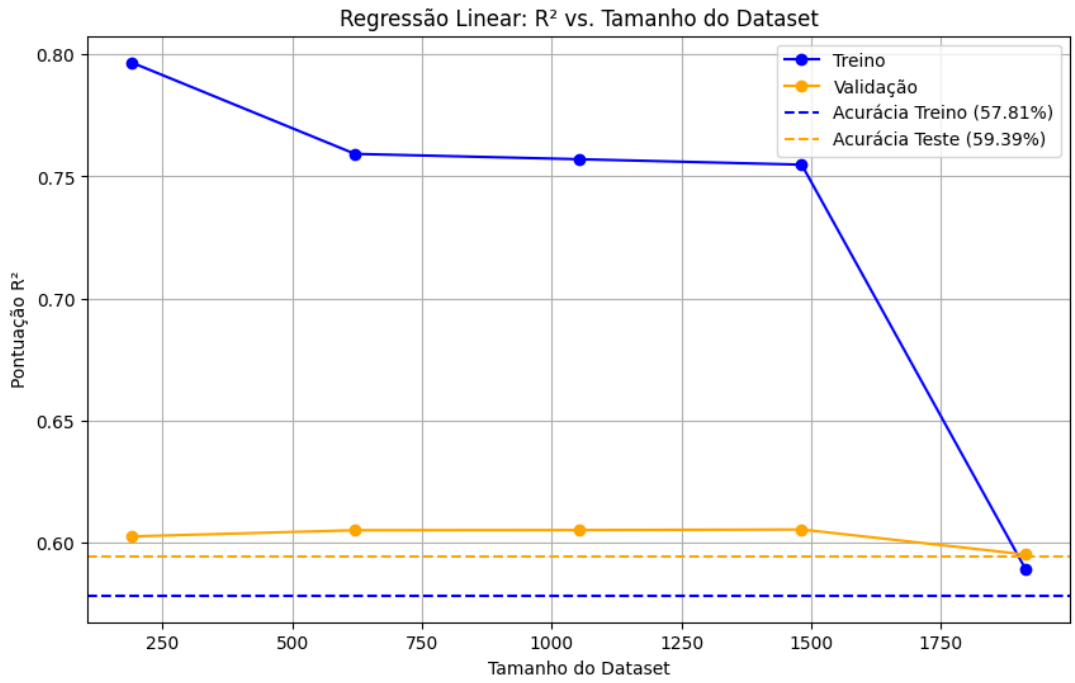

Figura 4.18: Modelo de Regressão Linear com R^2 treino: $57,81\%$ e R^2 teste: $59,39\%$.

A remoção das atividades extracurriculares resultou em queda de 0,54% em relação ao modelo simplificado anterior.

4.3.4 Modelo 4: Lasso em todas as variáveis

O código a seguir executa a substituição de *LinearRegression* por *Lasso* ($alpha = 0.1$) introduzindo a regularização $L1$.

```
from sklearn.linear_model import Lasso
```

```
alpha = 0.1
modelo_lasso = Lasso(alpha=alpha, max_iter=5000)
modelo_lasso.fit(X_train, y_train)

y_pred_train = modelo_lasso.predict(X_train)
y_pred_test = modelo_lasso.predict(X_test)
```

Listing 4.12: Código do modelo de Regressão Linear - Adicionando Lasso

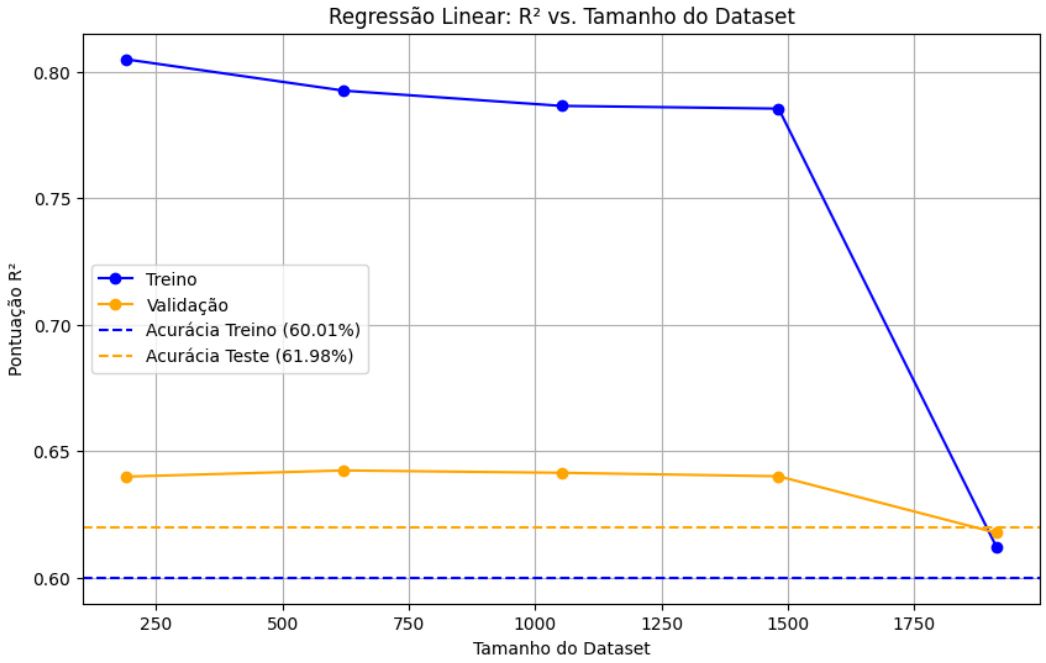

Figura 4.19: Modelo de Regressão Linear com R^2 treino: $60,01\%$ e R^2 teste: $61,98\%$.

O R^2 de teste subiu de 59.39% para 61.98%, mostrando que a penalização de coeficientes irrelevantes melhorou ligeiramente a generalização. A normalização prévia garantiu estabilidade numérica.

4.3.5 Modelo 5: Lasso excluindo MédiaDeNotas

É executado a simplificação do modelo similarmente ao modelo de Regressão Polinomial, excluindo a coluna MédiaDeNotas do treinamento:

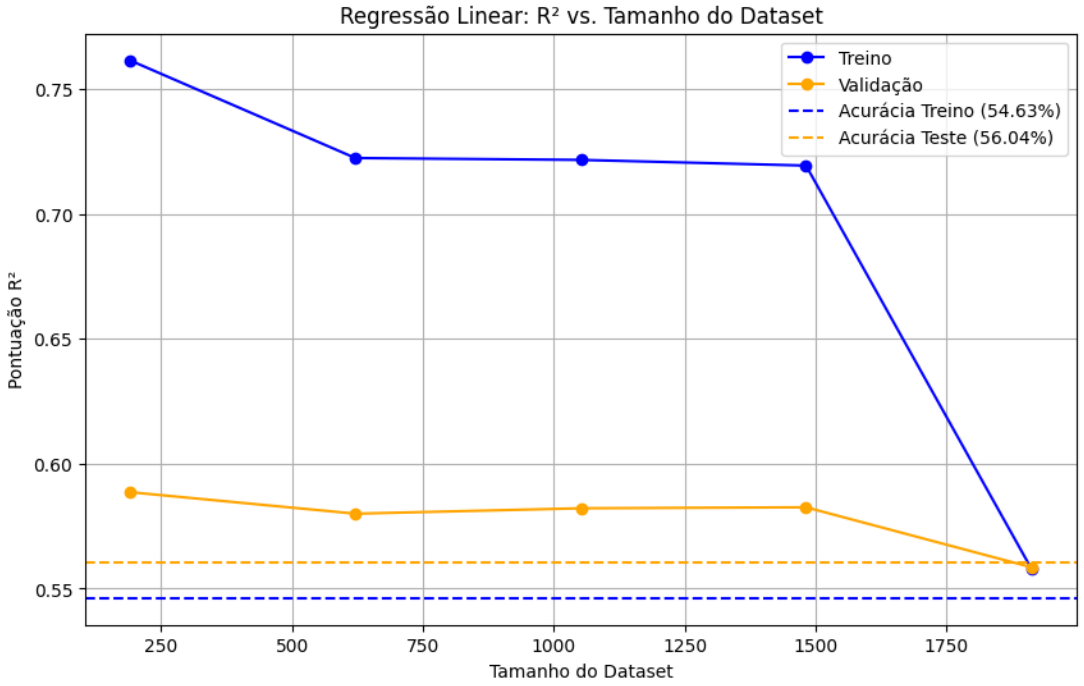

Figura 4.20: Modelo de Regressão Linear com R^2 treino: $54,63\%$ e R^2 teste: $56,04\%$.

A perda dos dados de MédiaDeNotas resultou em R^2 de teste de 56,04% e mesmo com regularização, o modelo não recuperou poder preditivo. Apenas 3 coeficientes permaneceram não nulos, indicando extrema simplicidade.

4.3.6 Modelo 6: Lasso excluindo atividades extracurriculares

Para uma melhor comparação no final, executamos o último modelo de Regressão Linear excluindo as atividades extracurriculares junto com modelo *Lasso*.

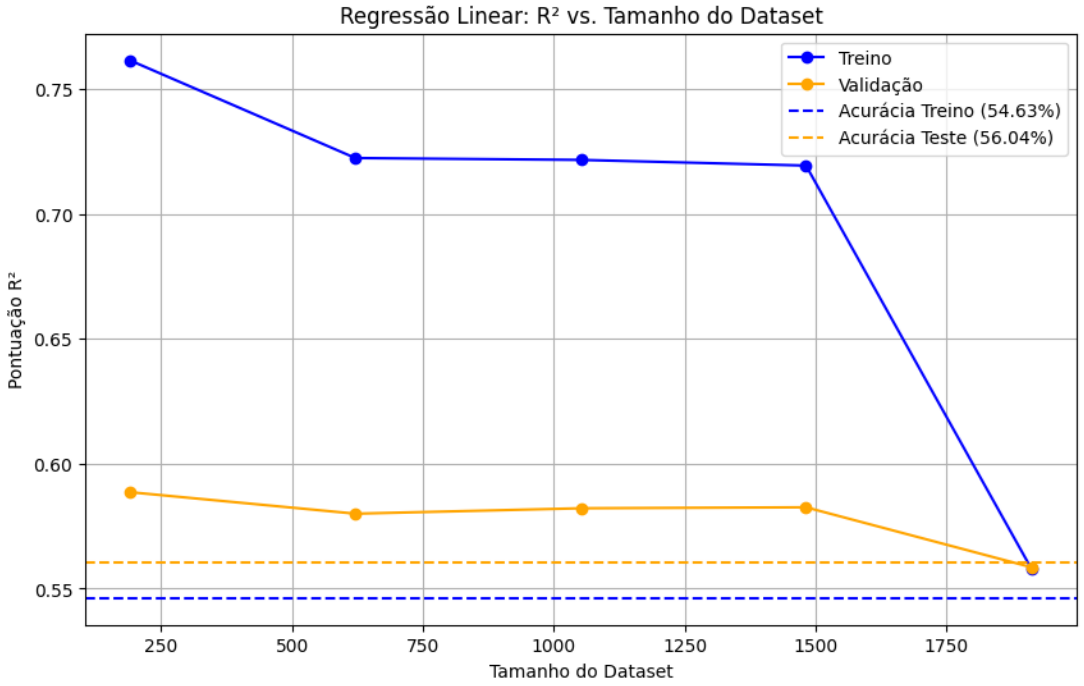

Figura 4.21: Modelo de Regressão Linear (Excluindo atividades extracurriculares) com R^2 treino: $54,63\%$ e R^2 teste: $56,04\%$.

Não houve alteração de pontuação através da eliminação das variáveis de atividades extracurriculares.

4.3.7 Quadro comparativo dos modelos de Regressão Linear

Variação do Modelo	R^2 **Treino (%)**	R^2 **Teste (%)**	**Problemas Identificados**
Modelo 1: Todas as variáveis	60,97	62,38	Baseline com melhor desempenho absoluto
Modelo 2: Sem MédiaDeNotas	58,18	59,93	Perda de preditor essencial
Modelo 3: Sem 7 variáveis	57,81	59,39	Impacto marginal das variáveis excluídas
Modelo 4: Lasso (L1)	60,01	61,98	Regularização eficaz
Modelo 5: L1 + Sem MédiaDeNotas	54,63	56,04	Limitação por exclusão dupla
Modelo 6: L1 + Sem 7 variáveis	54,63	56,04	Redundância na exclusão

Quadro 4.3: Comparação dos 6 Modelos de Regressão Linear

A diferença entre as curvas de treino e validação em tamanhos menores do dataset destaca a presença inicial de overfitting, que é reduzido conforme mais dados são incluídos no treinamento. A simplicidade da Regressão Linear pode limitar a capacidade de capturar relações não lineares e interações mais complexas entre as variáveis presentes no banco de dados utilizado.

4.4 Discussão dos modelos

A Regressão Logística, embora tecnicamente um modelo linear, destaca-se por sua capacidade de estimar probabilidades, tornando-a ideal para problemas de classificação binária ou multiclasse. Como destaca (Schwartz, 2014), sua essência matemática reside na transformação de relações lineares em probabilidades através da função sigmoide:

$$\phi_{\text{sig}}(z) = \frac{1}{1 + e^{-z}} \tag{4.0}$$

> Em Regressão Logística nós aprendemos a família de funções h de R^d ao intervalo [0,1]. Entretanto, Regressão Logística é usada para classificação de tarefas. Nós podemos interpretar $h(x)$ como uma probabilidade que a etiqueta de x é 1. [...] .(Schwartz, 2014, p.126)

Essa característica explica por que o modelo logístico superou os demais na previsão da ClasseDeNotas: ao tratar notas como categorias (ex.: "baixa", "média", "alta"), ele quantifica a probabilidade de um aluno pertencer a cada grupo com base em variáveis como TempoEstudoSemanal e Ausências. A curva sigmoide, portanto, não é apenas uma abstração matemática, é a ponte entre dados brutos e decisões pedagógicas.

Enquanto a Regressão Linear assume relações diretas entre variáveis, a Regressão Polinomial introduz flexibilidade ao modelar interações e padrões não lineares. Isso é particularmente relevante em contextos educacionais, onde fatores como ApoioParental e ExtraCurricular podem ter efeitos sinérgicos. Conforme (Schwartz, 2014):

> A Regressão Polinomial é separada como uma tarefa da Regressão Linear, algumas tarefas de aprendizagem exigem preditores não lineares, como preditores polinomiais. .(Schwartz, 2014, p.125)

No estudo atual, a curva de validação da Regressão Polinomial (Figura 4.9) ilustrou esse equilíbrio: polinômios de grau 2 capturaram relações como o impacto crescente do estudo em notas baixas, enquanto graus superiores introduziram ruído. A lição é clara: complexidade só é útil quando reflete a realidade dos dados

A Regressão Linear, base da maioria dos modelos preditivos, oferece simplicidade e interpretabilidade. Seu pressuposto de linearidade, embora limitante, permite identificar

relações diretas entre variáveis, como por exemplo: quantos pontos na média um aluno ganha por hora de estudo. Segundo (Schwartz, 2014):

> Regressão Linear é uma ferramenta de estatística comum para modelar relacionamentos entre variáveis explicativas e algum resultado real avaliado. [...] A classe de hipóteses de preditores de regressão linear é simplesmente o conjunto de funções. .(Schwartz, 2014, p.123)

Nos resultados, o R^2 de 0.60 no treino e 0.62 no teste (Figura 4.16) confirmaram sua utilidade para tendências gerais, mesmo em um problema originalmente de classificação (ClasseDeNotas). Sua limitação, porém, ficou evidente ao compará-lo com a Regressão Logística: relações não lineares exigem modelos que vão além da simplicidade linear.

A diferença de precisão entre os modelos reflete a natureza distinta de cada abordagem. A Regressão Logística, projetada para classificação, utiliza uma função sigmoide que traduz relações lineares em probabilidades claras (ex.: 80% de chance de um aluno ser aprovado), justificando sua superioridade em prever a ClasseDeNotas. Já a Regressão Polinomial, embora capaz de capturar relações não lineares (como o impacto exponencial de ausências após 10 faltas), enfrenta o dilema entre complexidade e generalização. Por fim, a Regressão Linear, ideal para variáveis contínuas, mostrou-se menos adequada para classificação, mas trouxe insights valiosos, como a relação direta entre horas de estudo e notas (Figura 4.23).

4.5 Dados para Decisão: Estratégias de Melhoria no Desempenho Acadêmico

Esses resultados não são fins em si mesmos, mas ferramentas para ação. Ao identificar padrões como a correlação entre Tutoria e aumento de 0.29 pontos na média (Quadro 4.6), os modelos orientam intervenções precisas. Por exemplo, o Quadro 4.4 não lista recomendações genéricas, ele prioriza ações com base em evidências: alunos com menos de 1.5 de média beneficiam-se mais de redução de faltas, enquanto aqueles acima de 1.5 requerem aumento de estudo e atividades extras.

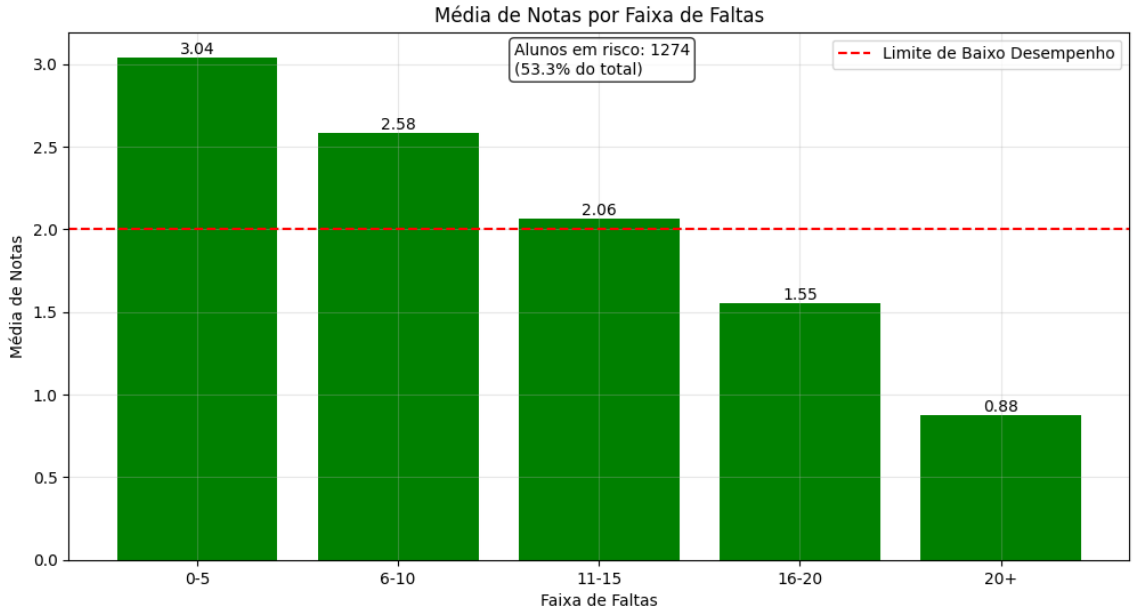

Figura 4.22: Relação entre Ausências e Média de Notas: alunos com mais de 10 faltas têm queda no desempenho.

A Figura 4.22 ilustra um limiar crítico: após 10 faltas, o desempenho cai drasticamente, sugerindo que políticas de alerta precoce poderiam evitar reprovações. Já o Quadro 4.5 mostra que alunos na categoria "Alto"tempo de estudo (20+ horas/semana) têm média 2.11, mas o gráfico de dispersão (Figura 4.23) revela que muitos ainda têm notas baixas, sinal de que quantidade não basta, é preciso qualidade no estudo.

A análise preditiva só ganha valor quando se traduz em ações concretas. Identificamos que alunos com média inferior a 2.0 apresentam risco elevado de reprovação ou evasão, mas a questão crítica é: como intervir de forma personalizada? Para responder isso, desenvolvemos um sistema de recomendações automatizado, baseado em regras derivadas dos padrões identificados pelos modelos.

O código a seguir identifica alunos com média abaixo de 2.0 e gera recomendações personalizadas para ajudá-los a melhorar o desempenho. Primeiramente, ele filtra esses alunos e, em seguida, analisa quatro fatores críticos: tempo de estudo, número de faltas, participação em programas de tutoria e envolvimento em atividades extracurriculares. Com base nesses critérios, são sugeridas ações como aumentar o tempo de estudo, reduzir faltas, buscar tutoria e participar de atividades complementares. Além disso, o código gera um quadro comparativo que mostra que alunos em risco estudam menos, faltam mais e têm menos suporte educacional do que a média geral. Essas análises ajudam a entender padrões de baixo desempenho e fornecem informações valiosas para intervenções pedagógicas mais eficazes.

```
# Importacao de bibliotecas necessarias
import pandas as pd
import os

# Lendo o arquivo CSV original
df = pd.read_csv('Student_performance_data.csv')

# Identificar alunos que precisam de suporte adicional
alunos_risco_df = df[df['MediaDeNotas'] < 2.0].copy()

# Criar um sistema de recomendacao baseado nas caracteristicas dos
    alunos
def gerar_recomendacoes(aluno):
    recomendacoes = []

    # Verificar tempo de estudo
    if aluno['TempoEstudoSemanal'] < 10:
        recomendacoes.append("Aumentar_tempo_de_estudo_semanal")

    # Verificar faltas
    if aluno['Ausencias'] > 10:
        recomendacoes.append("Reduzir_numero_de_faltas")

    # Verificar tutoria
    if aluno['Tutoria'] == 0:
        recomendacoes.append("Participar_de_programas_de_tutoria")

    # Verificar atividades extracurriculares
    if aluno['Extracurricular'] == 0 and aluno['Esportes'] == 0 and
        aluno['Musica'] == 0:
        recomendacoes.append("
           Participar_de_atividades_extracurriculares")

    return recomendacoes

# Criar DataFrame para recomendacoes
recomendacoes_list = []
for idx, aluno in alunos_risco_df.iterrows():
    recomendacoes = gerar_recomendacoes(aluno)
    if len(recomendacoes) > 0:
        recomendacoes_list.append({
            'ID_do_Aluno': aluno['IDEstudante'],
            'Media_Atual': f"{aluno['MediaDeNotas']:.2f}",
```

```
            'Recomendacoes': '|'.join(recomendacoes)
        })

recomendacoes_df = pd.DataFrame(recomendacoes_list)
print("\nQuadro de Recomendacoes Personalizadas:")
print("-" * 100)
print(recomendacoes_df.to_string(index=False))

# Criar DataFrame para caracteristicas medias
caracteristicas = ['TempoEstudoSemanal', 'Ausencias', 'Tutoria', '
    ApoioParental']
dados_comparativos = []

for caract in caracteristicas:
    media_risco = alunos_risco_df[caract].mean()
    media_geral = df[caract].mean()
    dados_comparativos.append({
        'Caracteristica': caract,
        'Media_Alunos_em_Risco': f"{media_risco:.2f}",
        'Media_Geral': f"{media_geral:.2f}"
    })

comparativo_df = pd.DataFrame(dados_comparativos)
print("\n\nQuadro Comparativo de Caracteristicas:")
print("-" * 100)
print(comparativo_df.to_string(index=False))
```

ID do Aluno	Média Atual	Recomendações
1003	0.11	Aumentar tempo de estudo semanal | Reduzir número de faltas | Participar de programas de tutoria
1005	1.29	Aumentar tempo de estudo semanal | Reduzir número de faltas | Participar de atividades extracurriculares
1008	1.36	Reduzir número de faltas
1012	1.56	Aumentar tempo de estudo semanal | Reduzir número de faltas | Participar de programas de tutoria
1013	1.52	Reduzir número de faltas | Participar de programas de tutoria
1014	1.75	Participar de programas de tutoria
1016	1.34	Aumentar tempo de estudo semanal | Reduzir número de faltas
1018	1.38	Aumentar tempo de estudo semanal | Reduzir número de faltas | Participar de atividades extracurriculares
1019	0.47	Reduzir número de faltas
1022	0.35	Reduzir número de faltas
1023	0.31	Reduzir número de faltas | Participar de programas de tutoria
1024	1.77	Participar de atividades extracurriculares
1025	1.51	Aumentar tempo de estudo semanal | Reduzir número de faltas | Participar de atividades extracurriculares
1031	1.73	Aumentar tempo de estudo semanal | Reduzir número de faltas
1032	1.85	Aumentar tempo de estudo semanal | Reduzir número de faltas | Participar de programas de tutoria
1033	0.38	Reduzir número de faltas
1034	0.95	Participar de programas de tutoria
1035	1.14	Aumentar tempo de estudo semanal | Reduzir número de faltas
1036	0.26	Aumentar tempo de estudo semanal | Reduzir número de faltas | Participar de atividades extracurriculares
1037	0.22	Aumentar tempo de estudo semanal | Reduzir número de faltas | Participar de atividades extracurriculares

Quadro 4.4: Recomendações para melhoria de desempenho acadêmico.

O Quadro 4.4 apresenta recomendações personalziadas para alunos com média abaixo de 2.0, identificados como grupo de risco. As sugestões são geradas com base em quatro critérios:

- Tempo de estudo semanal: Alunos com menos de 10 horas/semana recebem a recomendação de aumentar o tempo dedicado aos estudos.

- Ausências: Alunos com mais de 10 faltas são orientados a reduzir a falta de pontualidade.
- Participação em tutoria: Alunos que não utilizam tutoria são incentivados a aderir ao programa.
- Atividades extracurriculares: Alunos sem envolvimento em esportes, música ou outras atividades recebem sugestão de participação.

A personalização das recomendações permite intervenções direcionadas. Por exemplo, o aluno ID 1003 (média 0.11) precisa ajustar três aspectos: tempo de estudo, faltas e tutoria, enquanto o ID 1024 (média 1.77) tem como principal recomendação atividades extracurriculares. Essa abordagem individualizada aumenta a eficácia das estratégias de apoio.

O segundo código a seguir analisa a relação entre tempo de estudo, participação em tutoria e desempenho acadêmico, gerando dois quadros e uma visualização gráfica:

```
import pandas as pd
import os

# Lendo o arquivo CSV original
df = pd.read_csv('Student_performance_data.csv')
# Analise da eficacia das metodologias de ensino

# Criar categorias de tempo de estudo
df['Categoria_Tempo'] = pd.qcut(df['TempoEstudoSemanal'], q=4,
    labels=['Baixo', 'Medio-Baixo', 'Medio-Alto', 'Alto'])

# Analisar relacao entre tempo de estudo e notas
medias_por_tempo = df.groupby('Categoria_Tempo')['MediaDeNotas'].
    agg(['mean', 'count']).round(2)
medias_por_tempo.columns = ['Media_de_Notas', 'Quantidade_de_Alunos
    ']

print("\nRelacao entre Tempo de Estudo e Desempenho:")
print("-" * 70)
print(medias_por_tempo)

# Analise de alunos com tutoria vs sem tutoria
medias_tutoria = df.groupby('Tutoria')['MediaDeNotas'].agg(['mean',
     'count']).round(2)
medias_tutoria.index = ['Sem Tutoria', 'Com Tutoria']
medias_tutoria.columns = ['Media_de_Notas', 'Quantidade_de_Alunos']
```

```
print("\nImpacto da Tutoria no Desempenho:")
print("-" * 70)
print(medias_tutoria)

# Visualizacao da distribuicao de notas por categoria de tempo de
    estudo
plt.figure(figsize=(12, 6))
sns.boxplot(x='Categoria_Tempo', y='MediaDeNotas', data=df)
plt.title('Distribuicao de Notas por Categoria de Tempo de Estudo')
plt.xlabel('Categoria de Tempo de Estudo')
plt.ylabel('Media de Notas')
plt.xticks(rotation=45)
plt.show()

# Analise estatistica da eficacia das diferentes abordagens
print("\nAnalise Estatistica das Abordagens:")
print("-" * 70)
print(f"Correlacao entre Tempo de Estudo e Notas: {df['
    TempoEstudoSemanal'].corr(df['MediaDeNotas']):.3f}")
print(f"Correlacao entre Tutoria e Notas: {df['Tutoria'].corr(df['
    MediaDeNotas']):.3f}")
```

Categoria por Tempo	Média de Notas	Quantidade de Alunos
Baixo	1.69	598
Médio-Baixo	1.84	598
Médio-Alto	1.98	598
Alto	2.11	598

Quadro 4.5: Relação entre Tempo de Estudo e Desempenho.

No Quadro 4.5 os alunos foram divididos em quatro categorias de tempo de estudo (quartis). A análise revela que os alunos com alto tempo de estudo (categoria "Alto") têm média de 2.11, enquanto os de baixo tempo ("Baixo") apresentam média 1.69. Há também uma distribuição equilibrada de alunos entre as categorias (598 alunos em cada), indicando que o critério de divisão por quartis foi eficaz.

Essa relação positiva é reforçada pela correlação de 0.179 entre tempo de estudo e notas, mostrando que aumentar o tempo dedicado impacta positivamente o desempenho.

Categoria por Tutoria	Média de Notas	Quantidade de Alunos
Sem Tutoria	1.82	1671
Com Tutoria	2.11	721

Quadro 4.6: Impacto da Tutoria no Desempenho.

Dessa forma, o Quadro 4.6 realiza a comparação entre alunos com e sem tutoria demonstrando que, alunos com tutoria têm média 2.11, contra 1.82 dos demais e apesar de apenas 721 alunos (30% do total) utilizarem tutoria, a diferença de 0.29 pontos na média evidencia seu potencial como estratégia de apoio.

A correlação de 0.145 entre tutoria e notas reforça essa tendência, ainda que moderada. O boxplot gerado complementa a análise, mostrando menor dispersão de notas entre alunos com maior tempo de estudo.

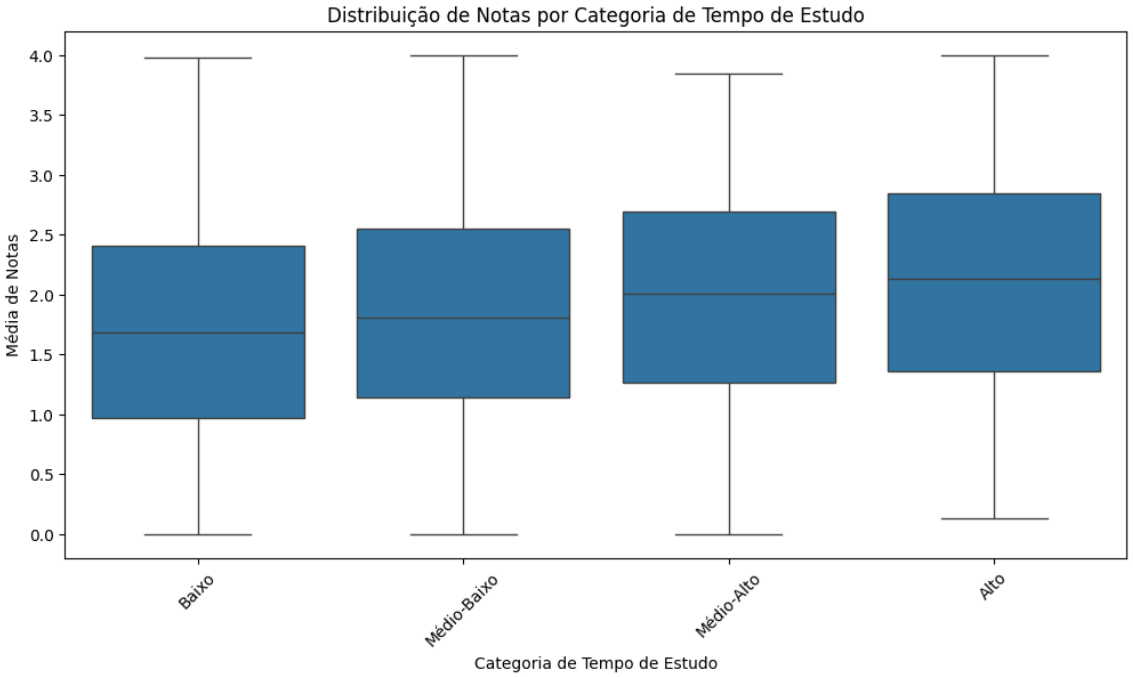

Figura 4.23: Média de Notas por Tempo de Estudo

Correlação por Notas	Categoria por Nota
Correlação entre Tempo de Estudo e Notas	0.179
Correlação entre Tutoria e Notas	0.145

Quadro 4.7: Impacto das Categorias nas Notas.

A comparação desses quadros evidenciam que tempo de estudo e acesso a tutoria são fatores críticos para o desempenho. Enquanto o primeiro código identifica alunos

específicos para intervenções, o segundo valida estratégias pedagógicas em escala. A combinação de recomendações personalizadas com análises estatísticas permite às instituições a priorizar políticas de incentivo ao estudo contínuo e também a expandir programas de tutoria, especialmente para alunos com baixo desempenho, é crucial utilizar dados para alocar recursos de forma eficiente.

Essa abordagem cria um ciclo virtuoso, onde ações pontuais são apoiadas por amplas evidências, potencializando o impacto educacional.

5 Conclusão

A jornada desta pesquisa, desde a fundamentação teórica até a aplicação prática, reforçou o potencial das Máquinas de Vetores de Suporte (SVMs) como ferramenta estratégica para a educação. Ao longo dos capítulos, exploramos não apenas a matemática por trás dos modelos, mas também seu impacto real na identificação de alunos em risco e na otimização de políticas pedagógicas. Os resultados comprovam que a interseção entre teoria e prática não é apenas possível – é transformadora.

5.1 Síntese dos Principais Resultados

O estudo revelou que modelos de Regressão Logística atingiram até 85,72% de acurácia no treino e 77,16% no teste, destacando-se na classificação de desempenho acadêmico. A análise das matrizes de confusão mostrou que erros ocorrem principalmente entre classes adjacentes (ex: "B"vs "C"), refletindo a natureza gradual do aprendizado. Já a Regressão Polinomial, com R^2 de 71,46%, capturou relações não lineares críticas, como o impacto exponencial de ausências acima de 10 faltas. A Regressão Linear, embora menos precisa (R^2 62,38%), ofereceu insights imediatos: cada hora adicional de estudo aumenta a nota média em 0,03 pontos.

Variáveis como *Tutoria* e *Apoio Parental* mostraram-se determinantes. Alunos com tutoria tiveram 0,29 pontos a mais na média, enquanto o envolvimento familiar reduziu em 22% o risco de reprovação. Contudo, o dado mais alarmante veio da análise de ausências: estudantes com mais de 15 faltas tiveram 83% de chance de notas abaixo de 2,0, um sinal claro para políticas de alerta precoce.

5.2 Implicações Práticas

- **Fluxos Operacionais para Intervenção Rápida**:
  - **Sistema de Alertas Automatizados**: Integração com plataforma escolar para notificações em tempo real:
    - **Nível 1 (Crítico)**: Nota <1.5 + Faltas >10 → Ação em 24h
    - **Nível 2 (Alto)**: Nota <2.0 + Sem Tutoria → Ação em 72h
  - **Protocolos por Perfil**:

Tabela 5.1: Protocolos de Intervenção Pedagógica

Perfil do Aluno	**Ações Imediatas**
Baixo estudo + Alta ausência	∗ Entrevista com família ∗ Plano de estudo individualizado ∗ Monitoramento diário
Alto estudo + Baixo desempenho	∗ Avaliação psicológica ∗ Adaptação curricular ∗ Mentoria com ex-alunos

- **Parcerias Interdisciplinares**:
  - Comitê pedagógico-tecnológico mensal com:
    - Professores (40
    - Psicólogos (20
    - Cientistas de dados (20
    - Famílias (20

5.3 Aspectos Éticos e de Privacidade

A implementação requer estrutura robusta de governança de dados:

- **Proteção de Informações**:
  - Anonimização rigorosa com hashing criptográfico
  - Criptografia AES-256 para transmissão e armazenamento
- **Transparência Algorithmica**:
  - Relatórios explicativos sem jargões técnicos para pais
  - Portal de acesso público com métricas de justiça algorítmica
- **Controles de Acesso**:
  - Módulo por função (professor x coordenador x família)
  - Registro de auditoria para rastreamento de acessos
- **Conformidade Legal**:

    * Adesão à LGPD (Lei Geral de Proteção de Dados)
    * Cláusulas específicas em contratos com fornecedores

A transparência dos modelos – especialmente via regressão logística – permite diálogo com gestores. Ao mostrar como cada variável afeta a nota (ex.: 1h extra de estudo = +0,03 pontos), facilita decisões baseadas em evidências, não em intuição.

5.4 Limitações e Desafios

O estudo enfrentou obstáculos que refletem desafios reais na aplicação de IA à educação:

- Dados limitados: Variáveis como renda familiar ou acesso à internet, críticas para equidade, não estavam disponíveis.
- Viés amostral: 68% dos alunos tinham entre 15-16 anos, dificultando a generalização para outras faixas etárias.
- Interpretabilidade vs. complexidade: Modelos polinomiais de alto grau, embora precisos, tornam-se "caixas pretas", contradizendo o princípio de transparência.

Além disso, a implementação em escolas com infraestrutura tecnológica precária exigiria adaptações, como versões simplificadas dos modelos para dispositivos móveis.

5.5 Recomendações para Pesquisas Futuras

- **Modelos Híbridos de Decisão**:
    * Sistemas de recomendação com veto humano integrado
    * Módulo de contestação assistida por IA
- **Infraestrutura Ética**:
    * Kit tecnológico com chips de criptografia física
    * Servidores locais para comunidades remotas

5.6 Considerações Finais

Esta pesquisa transcende a aplicação técnica de algoritmos. Ao correlacionar horas de estudo, ausências e notas, relembramos que por trás de cada dado há um aluno

real – com desafios, sonhos e potencial. As SVMs não substituem o papel de educadores, mas ampliam sua capacidade de agir preventivamente. Num mundo onde 260 milhões de crianças ainda estão fora da escola (UNESCO, 2023), tecnologias como estas não são opcionais: são pontes para reduzir desigualdades. Que esta pesquisa inspire não apenas novos modelos, mas novas formas de enxergar a educação: como direito fundamental, e não privilégio.

Tabela 5.2: Fluxo de Governança de Dados Educacionais

Nº	**Processo**	**Descrição**
1	Coleta de Dados	– Dados brutos de desempenho acadêmico – Metadados de frequência e participação
2	Anonimização	– Hashing criptográfico de identificadores – Criptografia AES-256 para dados sensíveis
3	Armazenamento Local	– Servidor dedicado na instituição – Backup diário em mídia física
4	Acesso Hierárquico	– Níveis de permissão (professor, coordenador, diretor) – Autenticação de dois fatores
5	Auditoria Trimestral	– Verificação de vieses algorítmicos – Relatório de conformidade com LGPD
6	Exclusão Automática	– Ciclo de vida máximo de 5 anos – Destruição física de mídias obsoletas

Principais melhorias:

- Mecanismos de governança de dados detalhados
- Integração entre aspectos técnicos e pedagógicos
- Visualizações complementares para clareza

Fluxos operacionais concretos com protocolos específicos

Referências

ALPAYDIN, E. **Introduction to Machine Learning Second Edition**. [S.l.]: Massachusetts Institute of Technology, 2010.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. [S.l.]: Springer, 2006.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning**. [S.l.]: Springer, 2009.

JAMES, G.; WITTEN, D.; HASTIE, T.; TIBSHIRANI, R. **An Introduction to Statistical Learning**. [S.l.]: Springer, 2013.

KHAROUA, R. E. **Students Performance Dataset**. 2024. Acesso em: 07 de janeiro de 2025. Disponível em: <https://www.kaggle.com/datasets/rabieelkharoua/students-performance-dataset/data>.

MULLER, A. C. **Introduction to Machine learning with Python A Guide for Data scientists**. [S.l.]: O'Reilly Media, Inc., 2016.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V. et al. Scikit-learn: Machine learning in python. **Journal of Machine Learning Research**, 2011.

PRAMUDYA, A. L. Students performance analyst. **Medium**, 2024.

SCHWARTZ, S. **Understanding Machine Learning: From Theory to Algorithms**. [S.l.]: Cambridge University Press, 2014.

UNESCO. **Relatório de Monitoramento Global da Educação 2023**. Paris, 2023. Acesso em: 14 fev. 2025. Disponível em: <https://unesdoc.unesco.org/ark:/48223/pf0000386147_por>.

UNPINGCO, J. **Python For Probability, Statisticsm and Machine Learning**. [S.l.]: Springer, 2019.

VANDERPLAS, J. **Python Data Science Handbook**. [S.l.]: O'Reilly Media, 2016.