UNIVERSIDADE FEDERAL DO TRIÂNGULO MINEIRO- UFTM

MESTRADO PROFISSIONAL EM MATEMÁTICA EM REDE NACIONAL- PROFMAT

Dissertação de Mestrado

Aprendizado de Máquina: Previsão de Evasão com Python e Recurso Didático e Tecnológico com Geogebra

Jonathan Marques Dias

Uberaba - Minas Gerais
13 de Novembro de 2024

Aprendizado de Máquina: Previsão de Evasão com Python e Recurso Didático e Tecnológico com Geogebra

Jonathan Marques Dias

Dissertação de Mestrado apresentada à Comissão Acadêmica Institucional do PROFMAT-UFTM como requisito parcial para obtenção do título de Mestre em Matemática.

Orientador: Prof. Dr. Osmar Aléssio.

Uberaba - Minas Gerais
13 de Novembro de 2024

Catalogação na fonte: Biblioteca da Universidade Federal do Triângulo Mineiro

Dias, Jonathan Marques
D532a Aprendizado de máquina: previsão de evasão com python e recurso didático e tecnológico com Geogebra / Jonathan Marques Dias. -- 2024.
247 p. : il., graf., tab.

Dissertação (Mestrado Profissional em Matemática em Rede Nacional) -- Universidade Federal do Triângulo Mineiro, Uberaba, MG, 2024
Orientador: Prof. Dr. Osmar Aléssio

1. Evasão universitária. 2. Aprendizado do computador. 3. Redes neurais (Computação). 4. GeoGebra (Software). I. Aléssio, Osmar. II. Universidade Federal do Triângulo Mineiro. III. Título.

CDU 378.4-057.88:004.4

Leila Aparecida Anastácio - CRB-6/2513

JONATHAN MARQUES DIAS

Aprendizado de Máquina: Previsão de Evasão com Python e Recurso Didático e Tecnológico com Geogebra

Dissertação apresentada ao Programa de Pós-Graduação em Mestrado Profissional em Matemática, área de concentração Matemática da Universidade Federal do Triângulo Mineiro como requisito parcial para obtenção do título de mestre

Uberaba, 13 de novembro de 2024

Banca Examinadora:

Dr. Osmar Aléssio – Orientador
Universidade Federal do Triângulo Mineiro

Dr. Leandro Cruvinel Lemes
Universidade Federal do Triângulo Mineiro

Dr. Airton Monte Serrat Borin Júnior
Instituto Federal do Triângulo Mineiro

no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 e no art. 34 da Portaria Reitoria/UFTM nº 215, de 16 de julho de 2024.

Documento assinado eletronicamente por **LEANDRO CRUVINEL LEMES**, **Professor do Magistério Superior**, em 04/12/2024, às 10:34, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 e no art. 34 da Portaria Reitoria/UFTM nº 215, de 16 de julho de 2024.

Documento assinado eletronicamente por **OSMAR ALESSIO**, **Professor do Magistério Superior**, em 06/12/2024, às 12:46, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 e no art. 34 da Portaria Reitoria/UFTM nº 215, de 16 de julho de 2024.

A autenticidade deste documento pode ser conferida no site http://sei.uftm.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1388922** e o código CRC **B5736845**.

A minha família, em especial a meu pai que sempre acreditou e me apoiou, e mesmo não estando aqui, sei que está torcendo e cuidando de mim.

Agradecimentos

Agradeço a minha esposa, Caroline, que esteve ao meu lado em todos os momentos mais importantes da minha vida, sempre me apoiando e acreditanto incondicionalmente em mim. Muito obrigado você é um dos principais motivos para eu conseguir chegar até aqui.

Agradeço a meus pais, Roseli e Arnaldo, que sempre acreditaram em mim, e que mesmo sem ter tido a oportunidade de estudar mais, sempre me motivaram a estudar, torcendo para que eu obtivesse sucesso. Muito obrigado pelo amor incondicional, e por serem pais incríveis, tudo que sou e conquistei foi graças a vocês.

Agradeço a minha irmã, Jeniffer, uma das pessoas que mais acredita em mim, a professora que eu mais admiro e uma das pessoas que esteve mais presente na minha vida, sempre me incentivando. Obrigado por ser tão incrível e sempre me apoiar.

Agradeço ao meu orientador, Dr. Osmar Aléssio, a quem admiro muito como pessoa e como professor. Obrigado por todos os ensinamentos, por toda sua dedicação e por acreditar em mim, sem você essa conquista não seria possível.

Agradeço a toda minha família, amigos e professores, que estiveram comigo nessa jornada e que de alguma forma contribuiram para a conclusão desse mestrado.

Agradeço a Deus por me abençoar e colocar as pessoas certas no meu caminho.

"Não é o conhecimento, mas o ato de aprender, não a posse mas o ato de chegar lá, que concede a maior satisfação."

- Carl Friedrich Gauss

Resumo

Este trabalho foi dividido em duas partes: A primeira aborda o aprendizado de máquina com algoritmos de aprendizagem supervisionada de classificação de uma forma teórica e prática, com aplicação no estudo de Evasão do curso de licenciatura em Matemática e a segunda parte aborda o entendimento simples e a matemática por trás de funcionamento de dois algoritmos de aprendizado de máquina para utilizá-lo como recurso didático tecnológico no ensino médio via software Geogebra. Para a Evasão no Ensino Superior, trabalhamos com alguns algoritmos supervisionados de classificação mais simples e populares: Redes Neurais Multilayer Perceptron e Redes Neurais Recorrentes, Árvores de Decisão, k-Vizinhos Mais Próximos e Máquina de Vetores de Suporte. A evasão é a saída antecipada antes da conclusão do curso, por desistência ocasionada por vários motivos, representando, portanto, condição terminativa de insucesso em relação ao objetivo de concluir um curso de graduação. A evasão no curso presencial de licenciatura em Matemática da Universidade Federal do Triângulo Mineiro-UFTM, ocorre com maior frequência, nos primeiros quatro períodos (semestres) do curso, contudo nota-se que a evasão do quinto período em diante também ocorre, definido como evasão tardia, isto é, nos últimos períodos. A previsão evasão acadêmica de sucesso ou fracasso em um programa de graduação pode ser baseada em padrões de sequência por semestres de: médias de notas, sexo do aluno, idade, cotas, aprovações em disciplinas, porcentagem de faltas, etc. Como observado, dados podem ter estrutura temporal (notas, faltas, etc) ou atemporal (sexo, cotas, etc). Neste trabalho utilizamos atributos temporais e por períodos semestrais com resultados de previsão em torno de 88% de acerto em relação a métrica f1score. Para o entendimento simples e a matemática por trás dos algoritmos de classificação: k-Vizinhos Mais Próximos-KNN e Árvores de Decisão, construímos dois aplicativos no software Geogebra com a finalidade da introdução do conhecimento de aprendizado de máquina na formação educacional do aluno e como recurso didático tecnológico no aprendizado da matemática por trás dos algoritmos.

Palavras-chave: Evasão; Aprendizado de Máquina; Redes Neurais; Geogebra.

Abstract

This work is divided into two parts: The first addresses machine learning with supervised learning algorithms for classification in both theoretical and practical terms, with an application in the study of dropout rates in the Mathematics Education degree program. The second part covers a simple understanding and the mathematics behind the functioning of two machine learning algorithms, to use them as a technological didactic resource in high school education through the GeoGebra software. For Higher Education dropout, we worked with some of the simplest and most popular supervised classification algorithms: Multilayer Perceptron Neural Networks and Recurrent Neural Networks, Decision Trees, k-Nearest Neighbors, and Support Vector Machines. Dropout is defined as the premature exit before the completion of the course, due to withdrawal caused by various reasons, thus representing a terminating condition of failure in relation to the goal of completing a degree. Dropout in the Mathematics Education program at the Federal University of Triângulo Mineiro (UFTM) occurs most frequently within the first four semesters of the course. However, it is noted that dropout also occurs from the fifth semester onward, defined as late dropout, i.e., in the final semesters. The prediction of academic dropout, whether success or failure in a degree program, can be based on sequence patterns across semesters such as: grade point averages, student gender, age, quotas, course completions, percentage of absences, etc. As observed, data can have a temporal structure (grades, absences, etc.) or be non-temporal (gender, quotas, etc.). In this work, we used temporal attributes and semester-based periods, achieving prediction results with around 88% accuracy in relation to the f1-score metric. For the simple understanding and the mathematics behind the classification algorithms: k-Nearest Neighbors (KNN) and Decision Trees, we built two applications using the GeoGebra software. These applications aim to introduce students to the concept of machine learning in their educational development and serve as a technological teaching resource to facilitate the learning of the mathematics behind these algorithms..

.

Keywords: Dropout, Machine Learning, Neural Networks, Geogebra.

Sumário

Lista de ilustrações

Lista de tabelas

1 INTRODUÇÃO

Em um futuro breve a Inteligência Artificial-IA estará presente em quase tudo que fizermos no dia a dia e o setor educacional precisa estar conectado com as novidades que a tecnologia oferece, neste sentido, é de suma importância introduzir o pensamento computacional no ensino básico, pois ajudará os alunos a desenvolverem habilidades que são altamente relevantes para futuras oportunidades em novas profissões que estão surgindo no mercado.

> "O pensamento computacional é composto por quatro habilidades-chave: decomposição, reconhecimento de padrões, abstração de padrões e projeto de algoritmo. A decomposição envolve a divisão de problemas complexos em partes gerenciáveis, ensinando os alunos a avaliar e entender todas as etapas necessárias. O reconhecimento de padrões, por sua vez, incentiva a busca por soluções com base em experiências passadas, promovendo eficiência e aprendizado com erros. A generalização e abstração de padrões são habilidades cruciais para identificar detalhes relevantes em um problema e ignorar informações irrelevantes. O projeto de algoritmo consiste em definir as etapas e regras necessárias para alcançar um resultado desejado.". (BOUCINHA, 2017)

A Política Nacional de Educação Digital - PNDE apresenta em um dos eixos principais a Educação Digital Escolar que visa a inserção da educação digital nos ambientes escolares, em todos os níveis e modalidades, a partir do estímulo ao letramento digital e informacional e à aprendizagem de computação, de programação, de robótica e de outras competências digitais, englobando o pensamento computacional como um dos objetivos. A Base Nacional Comum Curricular -BNCC, que é um documento normativo que define o conjunto orgânico e progressivo de aprendizagens essenciais que todos os alunos devem desenvolver ao longo das etapas e modalidades da Educação Básica. Nela há duas menções do termo "pensamento computacional", das quais destaca-se o trecho a seguir:

> "No Ensino Médio, na área de Matemática e suas Tecnologias, os estudantes devem utilizar conceitos, procedimentos e estratégias não apenas para resolver problemas, mas também para formulá-los, descrever dados, selecionar modelos matemáticos e desenvolver o pensamento computacional, por meio da utilização de diferentes recursos da área." (BRASIL, 2018, p. 470)

Pensando em abordar o tema de aprendizado de máquina, este trabalho foi dividido em duas partes: A primeira aborda o aprendizado de máquina com algoritmos de aprendizagem

supervisionada de classificação de uma forma teórica e prática, com aplicação no estudo de Evasão do curso de licenciatura em Matemática e a segunda parte aborda o entendimento simples e a matemática por trás de funcionamento de dois algoritmos de aprendizado de máquina para utilizá-lo como recurso didático tecnológico no ensino médio via software Geogebra.

Evasão no Ensino Superior, segundo INEP é a saída antecipada, antes da conclusão do ano, série ou ciclo, por desistência (independentemente do motivo), representando, portanto, condição terminativa de insucesso em relação ao objetivo de promover o aluno a uma condição superior a de ingresso, no que diz respeito à ampliação do conhecimento, ao desenvolvimento cognitivo, de habilidades e de competências almejadas para o respectivo nível de ensino. Obviamente, a interrupção do programa em decorrência de falecimento do discente não pode ser atribuída como insucesso, dado que, de forma geral, se trata de caso fortuito e não se pode presumir uma intencionalidade do indivíduo em interromper o curso, cessá-lo ou uma incapacidade do indivíduo de manter-se no programa educacional (BRASIL).

Os Censos educacionais, independentemente da sua metodologia de produção, recolhem informações de rendimento e da situação dos discentes ao final de um período letivo, as quais permitem calcular indicadores de rendimento escolar para diferentes unidades de agregação que compõem o sistema de ensino. Tais indicadores mensuram a movimentação dos discentes entre períodos letivos subsequentes, permitindo o cálculo de indicadores de fluxo ou trajetória educacional, além de expressarem relações entre rendimento escolar do aluno e sua trajetória em um determinado nível educacional, sua movimentação entre unidades educacionais integrantes do respectivo sistema de ensino, ou a interrupção prematura dessa trajetória (antes da conclusão esperada do respectivo nível de ensino).

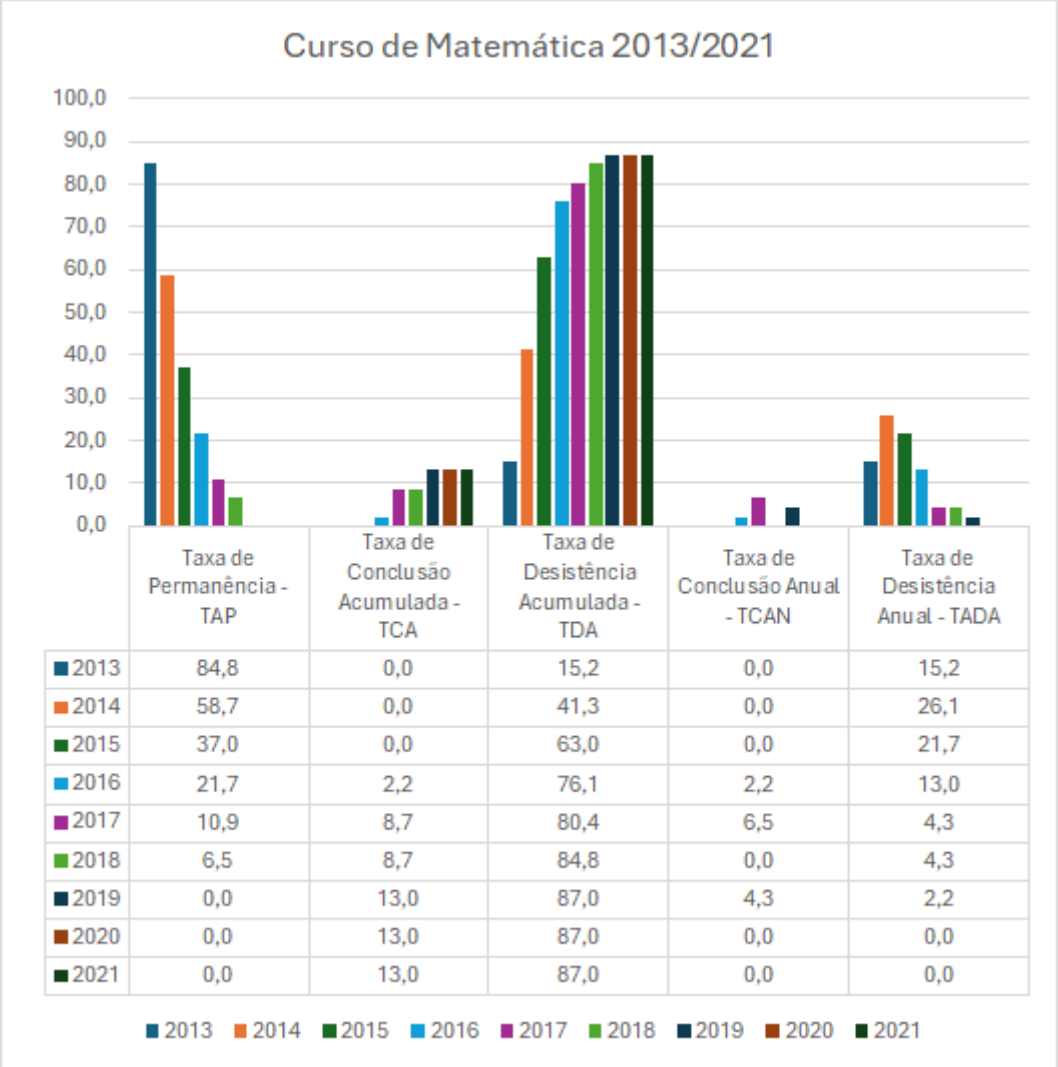
O Censo Superior, pesquisa censitária realizada anualmente pelo Inep em parceria com as instituições de ensino superior (IES), abrange os cursos de graduação e os sequenciais de formação específica. Até a edição de 2008, as estatísticas desta pesquisa referentes a alunos (matrículas, ingressantes e concluintes) eram coletadas agregadas por curso, o que inviabilizava o cálculo direto de indicadores de trajetória acadêmica. Isso permitia apenas algumas medidas aproximadas para expressar a eficácia na educação superior, como o cálculo do percentual de conclusão a partir da razão entre o número de concluintes de um ano e o de ingressantes quatro anos antes, considerando o tempo médio de formação superior de quatro anos, conforme divulgado pelo Resumo Técnico do Censo da Educação Superior 2008 (BRASIL)

A coleta de dados individuais de alunos, utilizada nos levantamentos censitários da educação superior a partir de 2009, possibilitou a compilação de uma base de dados longitudinal de discentes, conferindo precisão à informação ao nível individual e ampliando as possibilidades de análises, tendo os estudantes como a menor unidade básica de informação.

A figura 1 representa os dados de taxa de sucesso, Taxa de Permanência, Taxa

de Conclusão Acumulada, Taxa de Desistência Acumulada (Percentual do número de estudantes que desistiram (desvinculado ou transferido) do curso j até o ano t (acumulado) em relação ao número de ingressantes do curso j no ano T, subtraindo-se o número de estudantes falecidos do curso j do ano T até o ano t.), Taxa de Conclusão Anual, Tempo Médio de Conclusão do curso de Licenciatura em Matemática da UFTM.

Figura 1 – Indicadores de trajetórias do curso Licenciatura em Matemática UFTM



	Taxa de Permanência - TAP	Taxa de Conclusão Acumulada - TCA	Taxa de Desistência Acumulada - TDA	Taxa de Conclusão Anual - TCAN	Taxa de Desistência Anual - TADA
2013	84,8	0,0	15,2	0,0	15,2
2014	58,7	0,0	41,3	0,0	26,1
2015	37,0	0,0	63,0	0,0	21,7
2016	21,7	2,2	76,1	2,2	13,0
2017	10,9	8,7	80,4	6,5	4,3
2018	6,5	8,7	84,8	0,0	4,3
2019	0,0	13,0	87,0	4,3	2,2
2020	0,0	13,0	87,0	0,0	0,0
2021	0,0	13,0	87,0	0,0	0,0

Fonte:INEP

O fenômeno da evasão escolar é complexo e envolve desde questões relacionadas às condições objetivas para permanência na universidade (questões financeiras, tempo, saúde e problemas familiares) até questões subjetivas (dificuldades de aprendizagem, dificuldades de adaptação à proposta pedagógica e à cultura acadêmica, falta de identificação com o curso e dúvidas quanto à inserção no mundo do trabalho após a formação). Todos esses elementos impactam na decisão do aluno de evadir. Por isso, qualquer análise que utilize apenas dados quantitativos absolutos para compreender o fenômeno da evasão, corre o risco de simplificar um problema multifatorial, como relatado por Filho & Lobo.

> De modo geral, as instituições, públicas e privadas, dão como principal razão da evasão a falta de recursos financeiros para o estudante prosseguir nos estudos. É, também, o que o estudante declara quando perguntado sobre a principal razão da evasão. No entanto, verifica-se nos estudos existentes que essa resposta é uma simplificação, uma vez que as questões de ordem acadêmica, as expectativas do aluno em relação à sua formação e a própria integração do estudante com a instituição constituem, na maioria das vezes, os principais fatores que acabam por desestimular o estudante a priorizar o investimento de tempo ou financeiro, para conclusão do curso. Ou seja, ele acha que o custo-benefício do "sacrifício" para obter um diploma superior na carreira escolhida não vale mais a pena (Filho & Lobo).

A Comissão Especial de Estudos sobre a Evasão nas Universidades Públicas Brasileiras (1996) expande esse conceito ao definir três tipos de evasão: a) a evasão do curso, onde o acadêmico se desvincula do curso, o que pode ocorrer por não realização da rematrícula, por trancamento de matrícula, por mudança de curso ou por cancelamento de matrícula por parte da instituição por não atendimento a alguma norma; b) evasão da instituição, que se caracteriza pelo desligamento do vínculo do acadêmico não somente com o curso ao qual está matriculado mas também com a instituição de ensino; c) evasão do sistema, que é o desligamento permanente ou temporário do acadêmico com o ensino superior como um todo. Dentre as três modalidades trazidas por Rosa (2014), optamos por analisar "'a evasão do curso"' com a abordagem da "'evasão do aluno"' (Filho *et al.*).

Em 2015, o Fórum de Pró-reitores de Planejamento e Administração FORPLAD propôs o uso de uma fórmula para o cálculo da evasão anual, a qual foi baseada no trabalho de Filho & Lobo. A evasão anual é o percentual de acadêmicos que não se formou e não voltou a se matricular naquele ano. A equação 5 apresenta a fórmula utilizada para o cálculo do Índice de Evasão dos Cursos de Graduação no primeiro ano do período p:

$$E_{vp} = \left(1 - \frac{M_p - I_p}{M_{p-1} - C_{p-1}}\right) \times 100 \tag{1.1}$$

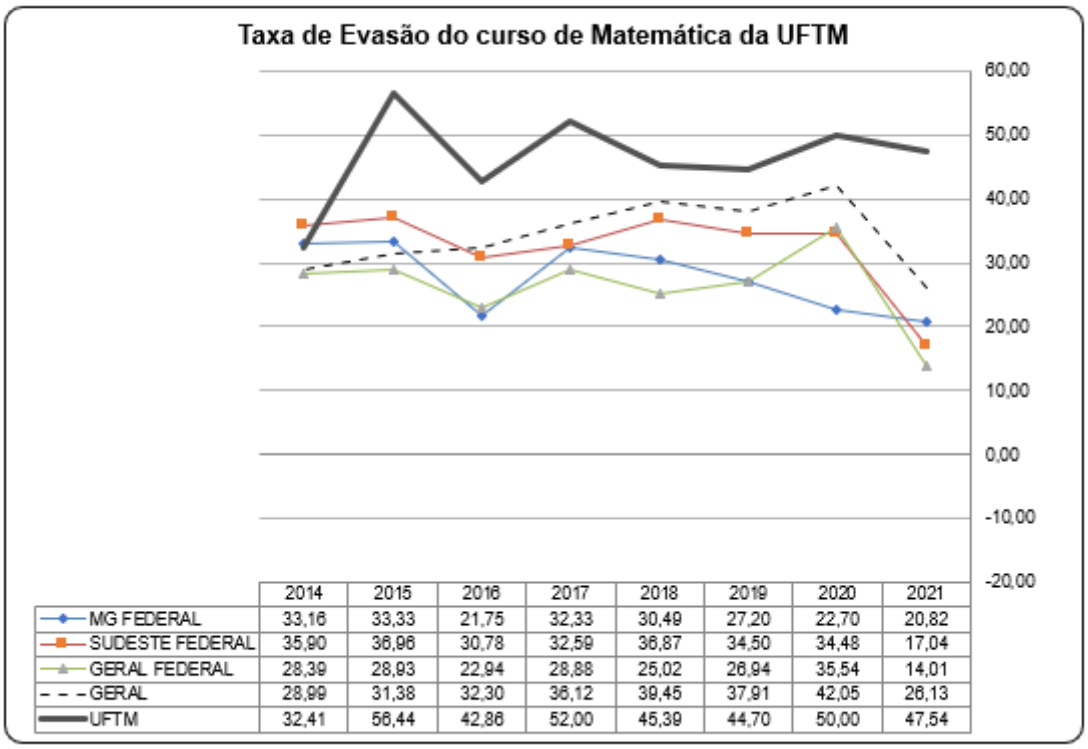
Na equação 1.1, M_p é o número de matriculados no período p; I_p é o número de ingressantes no período p; M_{p-1} é o número de matriculados no período anterior; C_{p-1} é o número de concluintes no período anterior.

A taxa de evasão não terá os mesmos valores da taxa de desistência, pois a **taxa de desistência** é o percentual do número de estudantes que saíram (desvinculado ou transferido) do curso j no ano t em relação ao número de estudantes ingressantes no curso j do ano T, subtraindo-se o número de estudantes falecidos do curso j até o ano t e a **taxa de evasão** é o percentual de acadêmicos que não se formaram e não voltaram a se matricular naquele ano, isto é, a equação permite identificar o fluxo de estudantes considerando aqueles efetivamente matriculados e não todos os estudantes potenciais para

as vagas ofertadas. A equação 1.1 apresenta a fórmula utilizada para o cálculo do Índice de Evasão dos Cursos de Graduação no primeiro ano do período p.

A figura 2 mostra os índices de taxa de evasão no curso de Licenciatura em Matemática. Estes dados foram enviados aos cursos de licenciatura pelo setor da PROPLAN.

Figura 2 – Taxa de evasão Licenciatura em Matemática



	2014	2015	2016	2017	2018	2019	2020	2021
MG FEDERAL	33,16	33,33	21,75	32,33	30,49	27,20	22,70	20,82
SUDESTE FEDERAL	35,90	36,96	30,78	32,59	36,87	34,50	34,48	17,04
GERAL FEDERAL	28,39	28,93	22,94	28,88	25,02	26,94	35,54	14,01
GERAL	28,99	31,38	32,30	36,12	39,45	37,91	42,05	26,13
UFTM	32,41	56,44	42,86	52,00	45,39	44,70	50,00	47,54

Fonte: PROPLAN/UFTM

Como indicam a taxa de evasão e ou a taxa de desistência acumulada do curso, terminar o curso já não é realidade para muitos que entram no curso, trazendo prejuízos tanto ao aluno quanto ao sistema da educação básica em relação a formação de professores, sem contar com o investimento federal que é necessário para a formar professores para a educação básica. Isso tem chamado a atenção de pesquisadores e entidades públicas que se empenham em entender as razões e os fatores que afetam o desempenho do estudante, de modo a elaborar meios para diminuir a ocorrência da evasão ou desistência, atenuando os seus danos.

O fenômeno da evasão é afetado diretamente pelo desempenho do aluno, pois, antes de evadir, sinais anunciam a eminência do evento. Por seu lado, o desempenho é influenciado pelo processo de aprendizagem que acontece ao longo do tempo, como apontado Molenaar (2014). Para ela, considerar a temporalidade nos trabalhos que analisam a vida acadêmica de um indivíduo tende a proporcionar elevação do grau de qualidade das investigações,

dado que as pesquisas tradicionais (não temporais) podem apresentar limitações se não analisarem adequadamente os eventos sequenciais da carreira dos discentes.

> Knight, Wise A & Chen em 2017 destacam que o aspecto temporal ainda não tem sido estudado em profundidade nos trabalhos envolvendo educação. Mahzoon *et al.* et all em 2018, ao aplicar abordagens preditivas de análise temporal da aprendizagem sobre o comportamento de acadêmicos em Computação, utilizando dados diversificados, mostrou que essa perspectiva provê melhores resultados que as convencionais. Trata-se de um trabalho focado na análise de dados de uma universidade americana para a predição acadêmica somente de alunos matriculados em um programa de graduação em Ciência da Computação. No entanto, considerando as particularidades dos procedimentos realizados, a abordagem desenvolvida pelo autor acima poderia ser estendida para o contexto das universidades brasileiras, por se basear, pelo menos parcialmente, em dados que também estão disponíveis nessas instituições. Knight, Wise A & Chen e Mahzoon *et al.* apud Vieira em 2021

De acordo com Melo em 2019, a utilização de redes neurais do tipo Long Short-Term Memory (LSTM) pode ser uma estratégia eficaz para analisar dados de alunos semestre a semestre. Um modelo LSTM que considere o histórico acadêmico completo, desde o ingresso do estudante até o período atual, tem potencial para alcançar maior precisão na identificação de tendências de evasão e na previsão do momento em que ela pode ocorrer.

Neste trabalho, demonstraremos que padrões temporais entre semestres podem fornecer informações valiosas sobre a experiência do aluno. Para isso, os dados de treino, validação e teste foram separados de forma não aleatória, evitando a mistura de informações entre alunos diferentes. A previsão de sucesso ou fracasso (evasão) em um curso de graduação pode ser realizada com base em padrões temporais relacionados a atributos como: notas, faltas, aprovações, reprovações, quantidade de disciplinas cursadas, entre outros fatores ao longo dos semestres. Utilizaremos Redes Neurais Multilayer Perceptron e Redes Neurais Recorrentes, Árvores de Decisão, k-Vizinhos Mais Próximos e Máquina de Vetores de Suporte. A evasão se dá mais nos primeiros períodos, porém ela pode ocorrer mas tardiamente também, lá pelo quinto, sexto e sétimo e até oitavo períodos. Para comprovar os benefícios de modelos que utilizam dados sequenciais com uma estrutura temporal, coletaremos e analisaremos dados do curso de Licenciatura em Matemática da UFTM em um modelo de sequência entre semestres. Os modelos serão avaliados pelas medidas de Acurácia, Especificidade, F1-Score e Sensitividade. Os algoritmos foram escolhidos em virtude da popularidade do bom desempenho em classificação.

A segunda parte deste trabalho é utilizar o aprendizado de máquina como recurso didático e tecnológico utilizando o Geogebra que favoreça a plena formação educacional, contribuindo com as mudanças e formulações das novas estratégias e diretrizes pedagógicas. Buscando aproximar os alunos do ensino médio de algumas técnicas de aprendizagem de

máquina, neste trabalho seguimos a ideia proposta no trabalho de Brandão em 2024 que propõe utilizar aprendizado de máquina no ensino médio, diferente da proposta dele, que era utilizar o SVM ("Support Vector Machine"), criamos atividades de aprendizado de máquina no Geogebra, similares as criadas por Petrics em 2020 e Pantaloni em 2023, pensando em facilitar a compreensão dos alunos do funcionamento dos algoritmos de predição KNN e árvore de decisão. Em Petrics (2020) ele trabalhou com distância euclidiana, como o KNN em Python também trabalha com distâncias não euclidianas e com inspiração nos trabalhos de Leivas, Portella & Souza em 2017; Luz em 2023; Kaleff & Nascimento em 2004 introduzimos a distância de Manhattan.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será abordado os principais temas que serão utilizados neste trabalho, primeiramente será descrito o tema aprendizado de máquina e os algoritmos de classificação: KNN, SVM, Árvore de Decisão, Rede Neural de Multiplas Camadas -MLP e LSTM . O texto foi baseado nos seguintes trabalhos (SILVA,), (KOLI,), (COUTINHO,) e (BALDUíNO, 2023).

A próxima seção 2.1 foram baseadas na referência Silva e a seção 2.2 foram baseadas nas referências(KOLI,), (COUTINHO,) e (BALDUíNO, 2023).

2.1 APRENDIZADO DE MÁQUINA

Machine Learning (ou aprendizado de máquina) é uma inteligência artificial que permite que computadores tomem decisões com a ajuda de algoritmos. Esses algoritmos reconhecem padrões e se tornam capazes de fazer predições. De forma bem simples, o Machine Learning se baseia na construção e no uso de algoritmos que "aprendem" a partir dos dados.

Falamos que uma máquina "aprende" com os dados quando ela consegue melhorar seu desempenho com o aumento de informação recebida. Dessa forma, utilizam-se observações e soluções calculadas anteriormente para realizar previsões de problemas utilizando dados distintos dos já observados.

Os problemas de Machine Learning são divididos em três subáreas principais:

1. Classificação: baseia-se em prever a categoria de uma observação dada. Aqui, procura-se estimar um "classificador" que gere como saída a classificação qualitativa de um dado não observado com base em dados de entrada (que abrangem observações com classificações já definidas). Exemplo: um classificador que utilize dados não observados de um paciente e classifique-o como doente ou não-doente.

2. Regressão: de forma similar a classificação, utiliza dados de entrada (preditores) já observados para prever uma resposta. A grande diferença é que, neste caso, procura-se estimar um valor numérico e não uma classificação de uma observação. Exemplo: estimar um modelo que utilize a idade e os anos de escolaridade de um indivíduo não-observado anteriormente para tentar prever seu salário. Utiliza-se como base desse modelo: idades, anos de escolaridades e salários de diversos indivíduos já observados anteriormente.

3. Agrupamento: também conhecido como "Clustering", tem como objetivo agrupar observações em grupos conhecidos como "clusters". Essas observações apresentam

similaridades dentro de seu cluster e diferenças em relação aos demais clusters formados. Diferente da Classificação, não é realizada a rotulação dos clusters, fazendo com que não exista uma clusterização errada ou certa. A clusterização utilizada resulta em diferentes tipos de clusters, e a escolha dessas técnicas deve ser previamente analisada pelo pesquisador. Exemplo: agrupar fotos de animais similares em clusters, sem ter o conhecimento prévio de qual animal está sendo apresentado.

Essas três subáreas podem ser divididas em dois tipos de aprendizado:

- Aprendizado supervisionado, abrange as subáreas da Classificação e Regressão. Utiliza dados de treinamento que você fornece ao algoritmo incluem as soluções desejadas, chamadas de rótulos, possibilitando comparar as previsões das respostas/rótulos com os rótulos reais.
- Aprendizado não-supervisionado, é onde a subárea do Clustering se encaixa. Como você pode imaginar, os dados de treinamento não são rotulados, suas técnicas não necessitam rótulos de suas observações e seu objetivo principal é encontrar grupos onde suas observações apresentem comportamento similar.

2.1.1 Principais Desafios do Aprendizado de Máquina

Devemos tomar cuidado com a inadequação de algoritmos para o problema proposto.

1. Quantidade insuficiente de dados de treinamento (a maioria dos algoritmos de aprendizado de máquina ainda precisam de uma grande quantidade de dados);
2. Dados de treinamento não representativos (dados de treinamento precisam ser representativos para que se consiga generalizar o modelo);
3. Dados de baixa qualidade (dados que contenham erros, outiliers e ruídos pode comprometer o modelo);
4. Características Irrelevantes (o sucesso do aprendizado de máquina parte de uma boa seleção das características);
5. Sobreajustando os dados de treinamento (modelo pode funcionar muito bem para os dados de treinamento e não generalizar bem o problema)
6. Subajustando os dados de treinamento (modelo utilizado é muito simples para generalizar os dados);

2.1.1.1 Quantidade Insuficiente de Dados

Em geral é necessário quantidade razoável de dados para que os algoritmos trabalhem como esperado. No caso de evasão que leva em conta a temporalidade dos dados, a previsão de evasão nos primeiros semestres terão menos dados que a previsão dos dados da metade do curso em diante.

2.1.1.2 Não representativos

Os conjuntos de dados devem ser o mais representativos possível do mundo real. O treinamento em conjuntos de dados não representativos pode causar desempenho ruim quando o modelo é solicitado a fazer previsões do mundo real. Imagine que queremos um modelo que classifique quem desistirá do curso no i-ésimo semestre e o treinamos com um conjunto de dados que contém 80% de alunos que desistiram no i-ésimo semestre. Quando esse modelo for usado no mundo real, ele pode ter dificuldade de reconhecer os alunos que continuaram além do i-ésimo semestre ou que se terminarm o curso, colando o grau.

2.1.1.3 Dados de baixa qualidade

Naturalmente, dados de treinamento cheios de erros, outliers ou ruídos farão com que o modelo tenha dificuldade para detectar os padrões básicos e como resultado, é menos provável que funcione bem, afetando negativamente a capacidade de generalização do modelo. Por exemplo, um modelo de classificação evasão treinado com um conjunto de dados com muitos exemplos rotulados de forma incorreta como evadido, pode ser incapaz de distinguir adequadamente entre os alunos que evadiram ou que colaram o grau.

2.1.1.4 Características Irrelevantes

Características irrelevantes são aqueles atributos que não têm impacto significativo na saída do modelo e quando são incluídas no treinamento, podem prejudicar a precisão do sistema. Por exemplo, o uso do atributo de qual período do aluno esta matriculado, parece não contribuir para sabermos se uma pessoa pode ou não evadir, portanto não faria sentido usar esse atributo (característica).

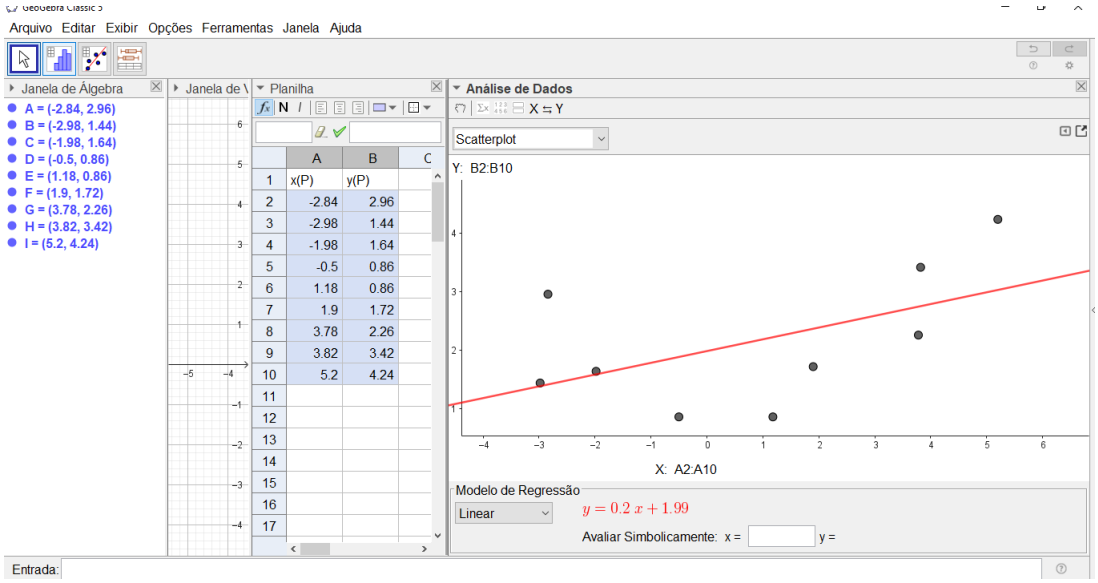
2.1.1.5 Sobreajustes e Subajustes

São elementos chave para a avaliação de modelos no Machine Learning. Ao utilizarmos o aprendizado supervisionado, temos como objetivo principal a predição dos dados, o que pode ocasionar erros de predição. Esses erros são divididos em erros irreduzíveis (“noise”) e reduzíveis. Estes são divisíveis em erros gerados pelo Bias e erros gerados pela Variância.

Subajustamento: ocorre quando o modelo não funciona bem nem com os dados de treino e nem com os dados novos ou dados de teste. Isso acontece quando o modelo

é muito simples para para o aprendizado da estrutura subjacente dos dados de treinamento. As previsões tentem a ser imprecisas, gerando um fenômeno conhecido como **Subajustamento**.

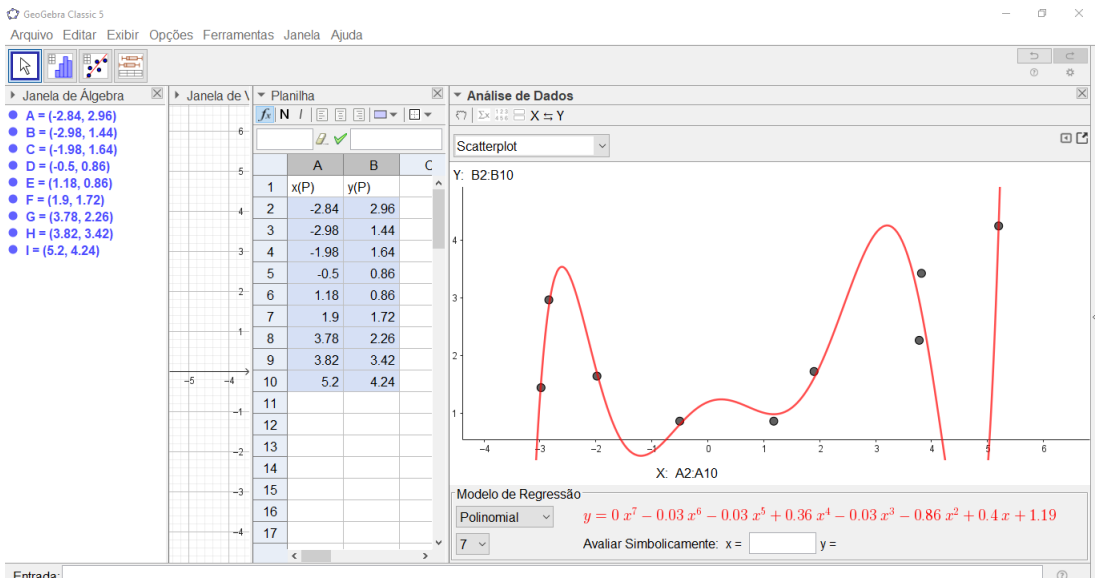
Figura 3 – Subajustamento



Fonte: Próprio Autor

Sobreajustamento ocasionado quando o modelo funciona bem com os dados de treinamento, porém tem um desempenho ruim com os dados novos ou dados de teste, isto é, o modelo não generaliza bem o prolema. Assim não consegue predizer bem dados menos específicos, gerando um fenômeno conhecido como **Sobreajustamento**.

Figura 4 – Subajustamento

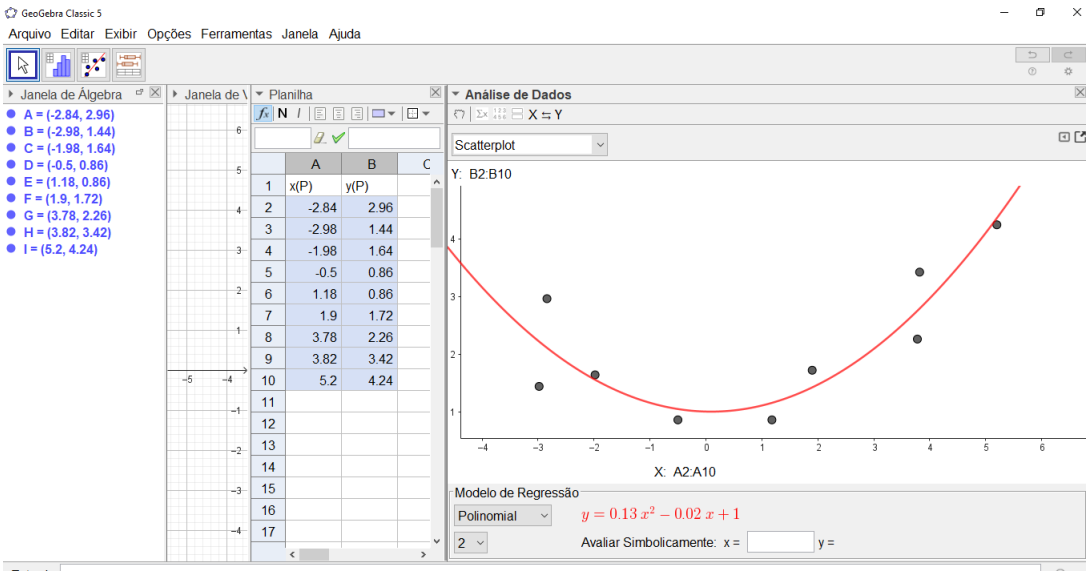


Fonte: Próprio Autor

Modelo Ajustado Quando o modelo se adéqua de forma moderada ao dado de

treinamento e tem uma boa performa no dados de teste, isto é, ele é tem uma boa generalização do modelo.

Figura 5 – Ajustado



Fonte: Próprio Autor

2.1.2 Medida de Desempenho

Como medir se meu modelo está bem ajustado? Isso dependerá do objetivo da técnica que utilizaremos: Classificação, Regressão ou Clustering.

2.1.2.1 Classificação

Para a classificação, a **acurácia** e o **erro** são as medidas básicas de desempenho.Quanto maior a acurácia do modelo, maior serão os acertos e menores serão os erros cometidos. Ela pode ser calculada da seguinte forma:

$$Acuracia = \frac{Observacoes\ classificadas\ corretamento}{Numero\ total\ de\ Observacoes\ Classificadas} \tag{2.1}$$

$$Erro = 1 - Acuracia \tag{2.2}$$

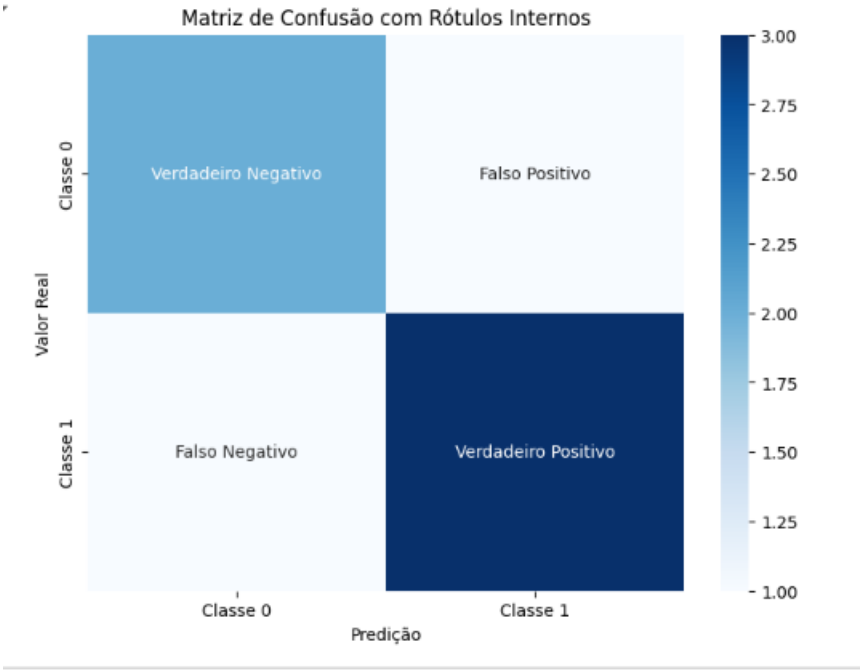
2.1.2.2 Matriz de Confusão

Matriz de confusão é uma tabela resumida do número de previsões corretas e incorretas. Ela é especialmente útil quando se trata de problemas de classificação binária, embora também possa ser adaptada para problemas de classificação multiclasse. Neste trabalho utilizaremos a matriz confusão para classificação binária.

Para avaliarmos nosso modelo de forma mais detalhada, utilizamos a Matriz de Confusão (“Confusion Matrix”). Nela, utilizamos classificação binária (Sim ou Não) dos rótulos do modelo e dos dados reais observados. Dessa forma, ao utilizarmos o modelo de classificação em dados já observados, conseguimos verificar o quanto que o modelo conseguiu predizer corretamente. A Matriz de Confusão é composta por 4 valores:

- Verdadeiro Positivo (VP): Quantidade de amostras foram preditas como positivo e o rótulo real é, de fato, positivo; (classificação correta da classe Positivo;)
- Falso Negativo (FN): Quantidade de amostras com rótulos verdadeiros e que foram preditos como falso ((Erro Tipo II): erro em que o modelo previu a classe Negativo quando o valor real era classe Positivo)
- Falso Positivo (FP): Quantidade de amostras preditas como verdadeiro, mas que, na realidade, eram falsos;((Erro Tipo I): erro em que o modelo previu a classe Positivo quando o valor real era classe Negativo;)
- Verdadeiro Negativo (VN): Quantidade de amostras previstas como negativo e o rótulo real é, de fato, negativo. (classificação correta da classe Negativo.)

Figura 6 – Matriz de Confusão



Fonte: Elaborado no Colab Python

Para classe negativa (classe 0) e a classe positiva (classe 1), podemos calcular a precisão da seguinte forma:

Definindo a matriz de confusão com as classes:

- **Verdadeiro Negativo (TN):** Classe 0 corretamente prevista como 0.
- **Falso Positivo (FP):** Classe 0 incorretamente prevista como 1.
- **Falso Negativo (FN):** Classe 1 incorretamente prevista como 0.
- **Verdadeiro Positivo (TP):** Classe 1 corretamente prevista como 1.

2.1.2.3 Acurácia

Expressa a quantidade de predições corretas em relação a todas as predições realizadas. Ainda que essa métrica seja particularmente útil quando a base de dados é balanceada, ou seja, quando há, aproximadamente, a mesma quantidade de amostras por classe, ela raramente é ponderada como métrica única para se avaliar um classificador.

$$Acuracia = \frac{no.\ total\ de\ precicoes\ corretas}{no.\ total\ de\ predicoes} = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.3}$$

2.1.2.4 Precisão

Calcula a proporção de verdadeiros positivos em relação a todas as predições positivas, incluindo as falsas positivas e os reais acertos, avaliando, assim, a assertividade do modelo quando ele decide fazer uma classificação positiva.

Fórmula para precisão Para a classe 0:

$$precisao_0 = \frac{VN}{VN + FN} \tag{2.4}$$

Fórmula para precisão Para a classe 1:

$$precisao_1 = \frac{VP}{VP + FP} \tag{2.5}$$

2.1.2.5 Recall ou Revocação ou Sensibilidade

Indica a proporção de amostras positivas em relação a quantidade de amostras classificadas corretamente pelo modelo.

A matriz de confusão terá a seguinte forma para uma classificação binária

Para classe 0

$$recall = \frac{VN}{VN + FP} \tag{2.6}$$

para classe 1

$$recall = \frac{VP}{VP + FN} \tag{2.7}$$

2.1.2.6 F1-Score

É definida pela média harmônica entre a precisão e a revocação, deste modo, seu valor dependerá dos resultados dessas duas métricas,

$$f1 - score = \frac{2 * precisao * recall}{precisao + recall} \tag{2.8}$$

2.1.3 Taxas de erros e acertos

Com base nos resultados da matriz de confusão da classificação binária, outros valores podem ser calculados:

2.1.3.1 Taxa Positiva Verdadeira (True Positive Rate – TPR), ou sensibilidade.

Taxa Positiva Verdadeira (True Positive Rate – TPR), ou sensibilidade. É a proporção de resultados corretamente classificados como positivo no resultado do modelo. Este resultado é comparado com todos os valores definidos como positivos na amostra, sendo calculado como

$$TPR = \frac{TP}{TP + FN}$$

2.1.3.2 Taxa Negativa Verdadeira (True Negative Rate – TNR), ou especificidade.

Taxa Negativa Verdadeira (True Negative Rate – TNR), ou especificidade. É a proporção de resultados classificados como negativos fora de todas as instâncias que não eram originalmente negativos. Pode ser calculado com

$$TPR = \frac{TN}{TP + FN}$$

2.1.3.3 Taxa de Falsos Positivos (False Positive Rate – FPR).

É calculada por

$$FPR = 1 - TNR$$

2.1.3.4 Taxa de Falsos Negativos (False Negative Rate – FNR)

tem a formalização na equação

$$FNR = 1 - TPR$$

Um classificador que tenha bom desempenho dará um alto TPR e TNR, mas baixos FPR e FNR.

Para classificadores que fornecem saídas probabilísticas, a sensibilidade (TPR) pode ser aumentada diminuindo o limiar de P, mas isso aumenta automaticamente o FPR. A curva de característica de operação do receptor (Receiver Operating Characteristic – ROC) e a área sob a curva ROC (Area Under the ROC curve – AUC), por exemplo, são usados para comparar o desempenho de algoritmos. Contudo esse desempenho é representado na faixa de limites normalizados entre 0 e 1. A curva ROC ideal tende ao canto superior esquerdo, resultando em alta TPR e baixa FPR. E por isso o máximo valor possível para AUC é 1.

2.1.3.5 Regressão

As métricas para avaliação das predições da rede foram a erro médio absoluto *Mean Absolute Error* (MAE); o erro percentual médio *Mean Absolute Percentage Error* (MAPE); a média do erro quadrado *Mean Squared Error* (MSE), é o mesmo erro avaliado pela função de perda, no entanto aqui ele não possui a função de ajudar na configuração da rede, somente auxiliar na avaliação do desempenho desta; a raiz quadrada do quadrado da média de erro *Root Mean Squred Error* (RMSE); e por fim o erro R quadrado R^2.

2.1.3.5.1 Erro médio absoluto - MAE

O calculado é dado pela seguinte fórmula:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.9}$$

É utilizado o modulo dos erros, pois há casos em que o erro negativo pode zerar o positivo fazendo com que o valor deste erro seja menor que o real, passando uma ideia de que o modelo é mais preciso do que de fato é. Esta métrica aponta o quão distante está a previsão do valor real, logo, quanto mais próximo de 0 o resultado for, melhor é o modelo, porém, não se pode afirmar a sua eficácia com base em somente uma métrica.

2.1.3.5.2 Erro percentual médio - MAPE

Este cálculo é feito pela seguinte fórmula:

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i} \tag{2.10}$$

Assim como a maioria das métricas quanto mais próximo de 0 significa que melhor foi o aprendizado do modelo. Enquanto a métrica anterior trazia a média da distância de previsão em valores absolutos, aqui essa distância é representada em valores percentuais.

2.1.3.5.3 Média do erro quadrado - MSE

Já a média do erro quadrado a fórmula:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.11}$$

Semelhante ao MAE 2.9, porem por haver a exponenciação do erro, dá um maior peso aos maiores erros. O uso desta métrica deixa o modelo sensível a valores discrepantes, *outliers*.

2.1.3.5.4 Raiz quadrada da média de erro quadrado - RMSE

A raiz quadrada da média do erro quadrado possui a fórmula:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.12}$$

Assim como o erro anterior MSE, este erro também é sensível a *outliers*, porém a diferença deste com o MSE é que neste o valor retornado está na mesma unidade de medida que o valor estudado, assim como o MAE.

2.1.3.5.5 R quadrado - R^2

A métrica R quadrado é representada equação:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y_i})^2} \tag{2.13}$$

Nesta fórmula $\bar{y}$ representa a média dos valores reais. Esta métrica descreve o quanto o modelo é confiante e consegue explicar os dados, o resultado geralmente varia entre 0 e 1, porém, há a possibilidade desta métrica retornar um valor negativo.

2.2 ALGORITMOS SUPERVISIONADOS

Os algoritmos supervisionados são técnicas de aprendizado de máquina que utilizam dados rotulados para treinar modelos capazes de fazer previsões. Esses algoritmos trabalham a partir de um conjunto de treinamento, no qual cada entrada (dados de entrada) está associada a uma saída conhecida (rótulo ou resposta). O objetivo é ensinar o modelo a reconhecer padrões e relações entre as entradas e as saídas, de modo que ele possa prever corretamente os rótulos de novos dados ainda não observados.

2.2.1 Algoritmos de classificação

Baseia-se em prever uma classe (ou classes) de uma observação dada. Aqui, procura-se estimar um "classificador" que gere como saída a classificação qualitativa de um dado não observado com base em dados de entrada (que abrangem observações com classificações já definidas). Pode se dizer que algoritmos de classificação atribuem rótulos aos dados classificando-os em classes.

São exemplos de algoritmos de classificação:k-vizinhos mais próximos (KNN- K-*Nearest Neighbors*), Árvore de decisão(*Decision Tree*), Máquina de vetores de suporte (SVM -*Support Vector Machines*),Regressão Logística e Redes Neurais.

2.2.1.1 Inteligência Artificial Explicável (XAI)

De acordo com (SOLUTIONS,), um algoritmo pode ser considerado como pertencente à classe XAI se seguir três princípios: transparência, interpretabilidade e explicabilidade. A transparência é garantida se os processos que calculam os parâmetros do modelo e produzem os resultados puderem ser descritos e justificados. A interpretabilidade descreve a possibilidade de entender o modelo e apresentar como ele toma decisões de uma forma compreensível para o ser humano. A explicabilidade refere-se à capacidade de decifrar por que uma determinada observação recebeu um valor específico. Na prática, esses três termos estão intimamente ligados e são frequentemente usados de forma intercambiável, na ausência de consenso sobre suas definições precisas.

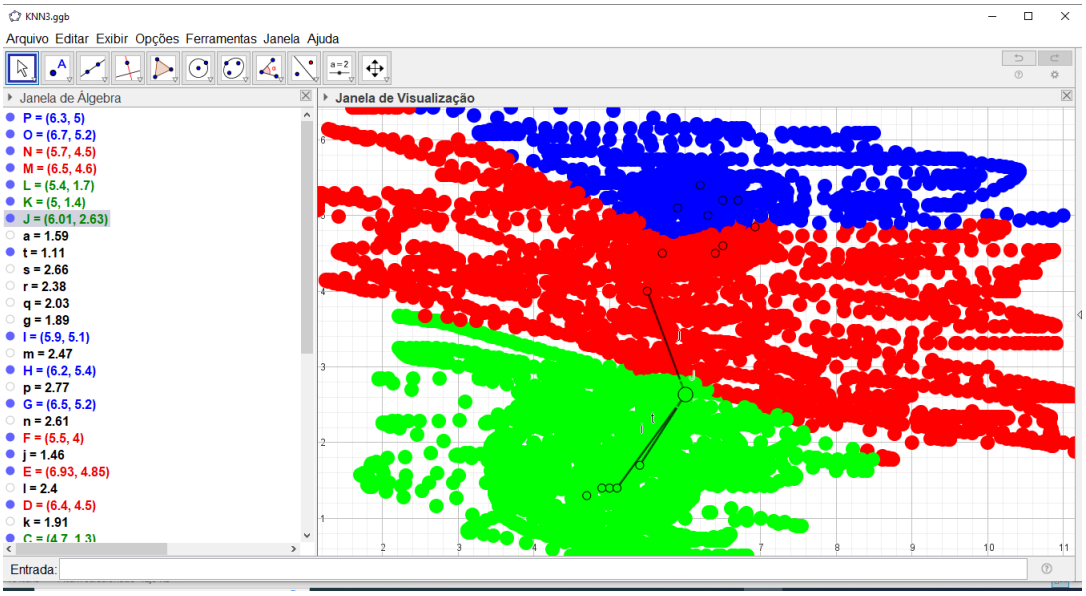
Os algoritmos k-vizinhos mais próximos, Árvore de Decisão, Máquina de vetores de suporte (SVM -*Support Vector Machines*),Regressão Logística e Redes Neurais Simples são exemplos de algoritmos XAI.

2.2.1.2 k-vizinhos mais próximos

O algoritmo k-vizinhos mais próximos (K-Nearest Neighbors-KNN) é um dos algoritmos de classificação mais simples de compreensão e implementação. Sua ideia consiste em, dado uma instância desconhecida, procurar pelos k vizinhos mais próximos a ela em um conjunto de dados previamente conhecido, segundo uma medida de distância pré-estabelecida. A classe do novo objeto será assumida como o voto majoritário entre os seus k vizinhos. O KNN verifica as k instâncias mais próximas e classifica conforme a classe da maioria.

Os parâmetros utilizados no algoritmo são o K, o peso dos atributos e o método de cálculo das distâncias. Caso o K seja um valor muito pequeno deixa o algoritmo mais suscetível a ruídos, um valor muito alto deixa as fronteiras entre as classes menos definidas (PEDREGOSA et al., 2011). O peso dos atributos pode ser uniforme, por distância ou customizado. A fórmula do cálculo das distâncias também é um parâmetro, a fórmula de cálculo mais comum é a Distância Euclidiana.

Figura 7 – KNN, k=3



Fonte: Próprio Autor

A próxima seção foi baseada na referência Koli.

2.2.1.3 Árvore de Decisão

Uma árvore de decisão (Decision Trees) é um algoritmo de aprendizagem supervisionado não paramétrico . Possui uma estrutura hierárquica em árvore, que consiste em um nó raiz, ramos, nós internos e nós folha.

As árvores de decisão são a base para muitos algoritmos clássicos de aprendizado de máquina, como "Random Forests", "Bagging"e "Boosted Decision Trees". Sua ideia era representar os dados como uma árvore onde cada nó interno denota um teste em um atributo (basicamente uma condição), cada ramo representa um resultado do teste e cada nó folha (nó terminal) contém um rótulo de classe.

Tipos de árvores de decisão

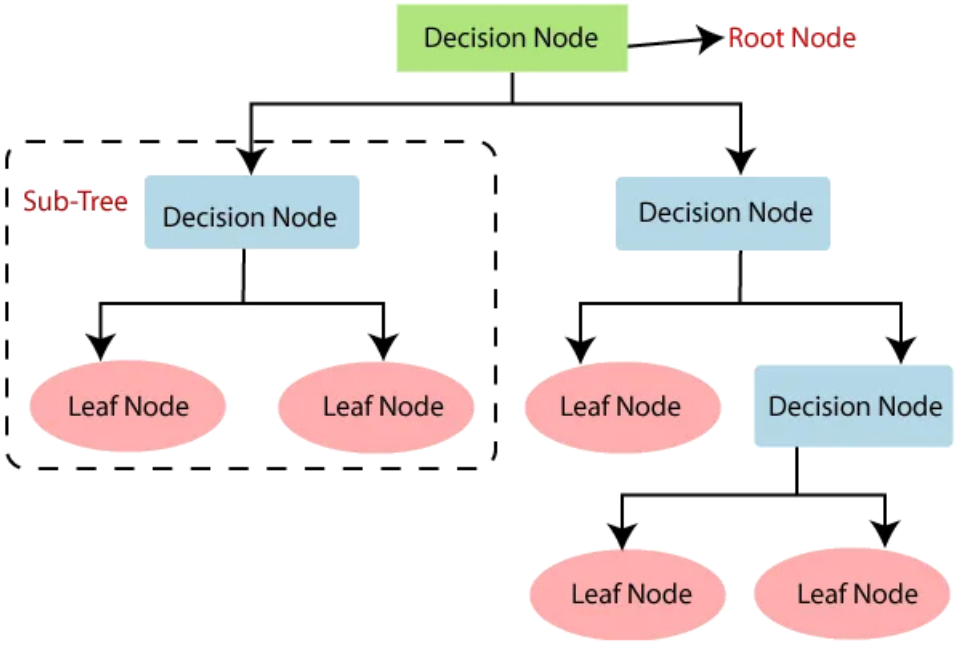
- CART (Árvores de Classificação e Regressão) usa Índice de Gini (Classificação) como métrica.
- ID3 (Dicotomizador Iterativo 3) usa a função Entropia e ganho de informação como métricas.

Antes de aprender mais sobre árvores de decisão, vamos nos familiarizar com algumas terminologias.

- Nó Raiz -é o nó presente no início de uma árvore de decisão;
- Nó de decisão - é o nó que obtemos após dividir o nó raiz;

- Nó Folha - é o nó onde a divisão adicional não é possível;
- Ramo / Subárvore -é uma subseção da árvore de decisão; e
- Poda - Eliminação de algumas partes inferiores (poda) da árvore de decisão.

Figura 8 – Árvore de Decisão



Fonte: Medium

Por que usar árvores de decisão? Existem vários algoritmos em aprendizado de máquina. Abaixo estão os dois motivos para usar a árvore de decisão:

As árvores de decisão geralmente imitam a capacidade de pensamento humano ao tomar uma decisão, por isso são fáceis de entender. A lógica por trás da árvore de decisão pode ser facilmente compreendida porque mostra uma estrutura semelhante a uma árvore.

A próxima seção foi baseada na referência Coutinho e Lorena & Carvalho

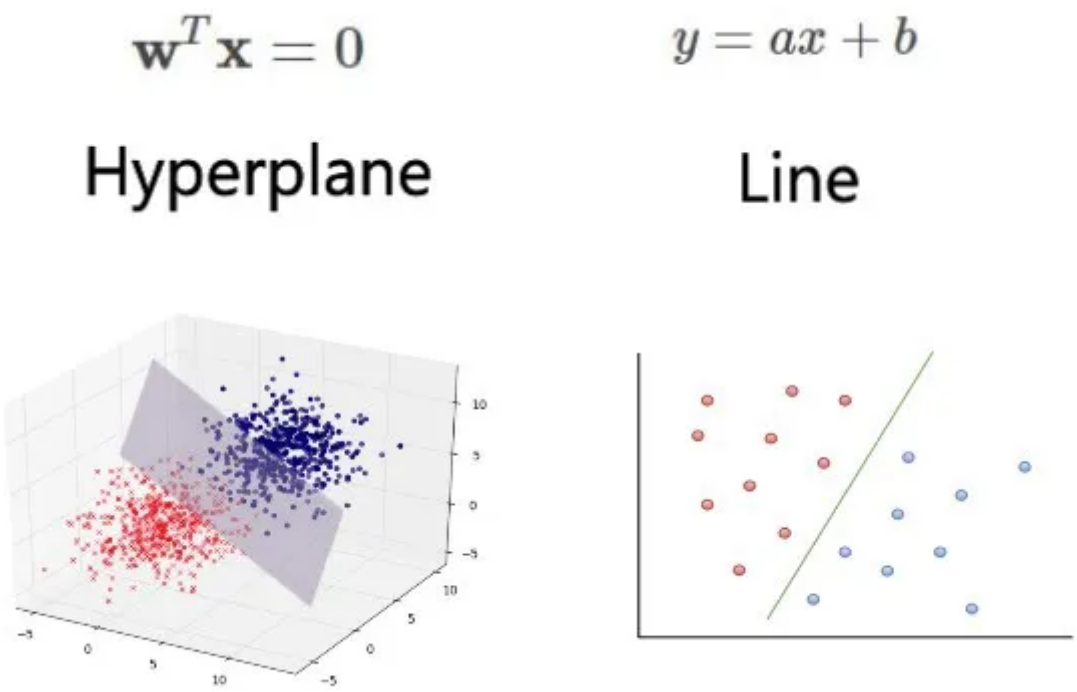
2.2.1.4 Máquina de vetores de suporte

Máquina de vetores de suporte ("Support Vector Machine-SVM) é um algoritmo de aprendizado de máquina supervisionado que pode ser usado para desafios de classificação ou regressão. Seu foco maior é no treinamento e classificação de um conjunto de dados.

O SVM é um algoritmo que busca uma linha de separação entre duas classes distintas analisando os dois pontos, um de cada grupo, mais próximos da outra classe. Isto é, o SVM escolhe a reta — também chamada de hiperplano em maiores dimensões— entre dois grupos que se distancia mais de cada um (no caso abaixo, a reta vermelha).

Primeiramente, cada dado é representado por um vetor (x_i, y_i), em que x_i é um vetor das coordenadas (features) de cada dado e y_i é a classe a qual o dado pertence (1 ou -1).

Figura 9 – SVM



Fonte: Medium (COUTINHO,)

Assim como uma reta é representada por $y = ax + b$, um hiperplano é representado por $\mathbf{w} \cdot \mathbf{x} + b = 0$, em que o $\mathbf{w}$ é um vetor normal ao hiperplano, que determina sua direção, b é responsável por deslocar linearmente o hiperplano no espaço, sem alterar sua direção, e o conjunto de pontos pertencentes ao hiperplano são todos aqueles pontos $\mathbf{x}$ que satisfazem à equação apresentada.

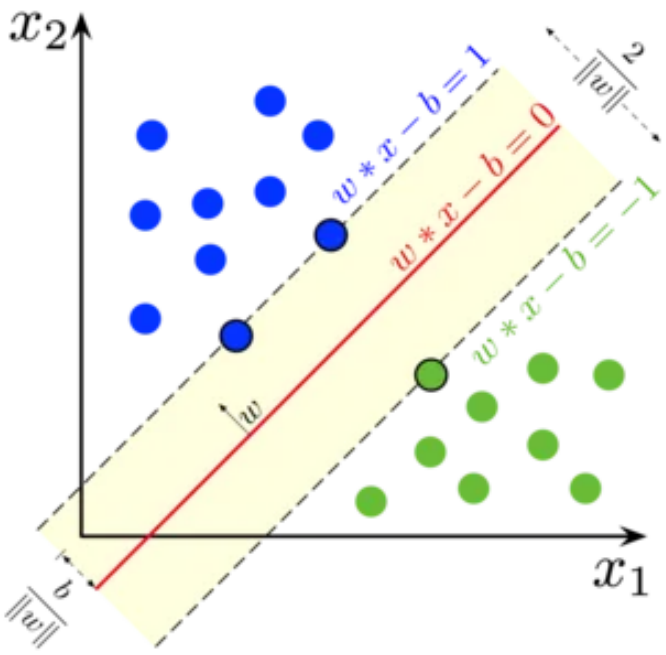
Mas por que $\mathbf{w} \cdot \mathbf{x} + b = 0$ forma um hiperplano? Bem, $\mathbf{w} \cdot \mathbf{x}$ é o produto escalar do vetor $\mathbf{w}$ com o vetor $\mathbf{x}$ (distância da origem até x), que será igual a zero quando $\mathbf{w}$

e $\mathbf{x}$ forem ortogonais. Dessa forma, os pontos $\mathbf{x}$ que satisfazem $\mathbf{w} \cdot \mathbf{x} = 0$ formarão um hiperplano que passa pela origem. Ao adicionar o b à equação, criaremos novos hiperplanos de mesma direção porém deslocados pra fora da origem.

E, como nosso alvo é achar o hiperplano que melhor separe as duas classes, isso significa que, matematicamente, precisamos encontrar o $\mathbf{w}$ e o b ideais.

Agora, para determinar o hiperplano ideal de separação entre duas classes, primeiro é necessário encontrar os hiperplanos que estão no limite de cada grupo (as margens). Para tal, utilizaremos os vetores suporte dos pontos de cada classe mais próximos do hiperplano de separação por onde as margens devem passar. As representações destes são $\mathbf{w} \cdot \mathbf{x} + b = 1$ e $\mathbf{w} \cdot \mathbf{x} + b = -1$ para suas respectivas classes, como mostrado na imagem abaixo.

Figura 10 – SVM



Fonte: Wikimedia Commons.

Assim, todos os dados da classe 1 satisfazem a inequação $\mathbf{w} \cdot \mathbf{x} + b \geq 1$ e todos os dados da classe -1 satisfazem a inequação $\mathbf{w} \cdot \mathbf{x} + b \leq -1$. Juntando essas duas inequações temos que $y_i \cdot (\mathbf{w} \cdot x_i + b) \geq 1$.

Como a distância de um hiperplano até a origem é dada por $\frac{b}{|\mathbf{w}|}$, a distância entre nossos dois hiperplanos será $\frac{(1-b)}{|\mathbf{w}|} - \frac{(-1-b)}{\mathbf{w}|}$, cujo resultado é $\frac{2}{|\mathbf{w}|}$. Já que o hiperplano ideal é aquele cuja distância a cada grupo é a maior possível, nosso w ideal é aquele valor para o qual $\frac{2}{|\mathbf{w}|}$ seja máximo.

Portanto, o $\mathbf{w}$ ideal será o menor w que satisfaça a inequação $y_i \cdot (\mathbf{w} \cdot x_i + b) \geq 1$ para todos os pontos i. A partir disso, utilizamos algum algoritmo de otimização como

gradiente descendente ou Multiplicadores de Lagrange para encontrar o $\mathbf{w}$ e o b do melhor hiperplano.

2.2.1.5 Regressão Linear Múltipla

O modelo de regressão linear múltipla com n variáveis, em geral é modelado por uma a variável dependente ou resposta Y pode estar relacionada com k variáveis explicativas ou independentes $X_1, X_2, ..., X_n$. O modelo para cada Y é dado por

$$Y = \alpha_0 + \alpha_1 X_1 + \alpha_2 X_2 + \cdots + \alpha_n \ X_n + \epsilon_i$$

.

Para cada variável dependente ou resposta y_i, com n variáveis explicativas ou independentes $x_{i1}, x_{i2}, ..., x_{in}$.

$$y_i = \alpha_0 + \alpha_1 x_{i1} + \alpha_2 x_{i2} + \cdots + \alpha_n \ x_{in} + \epsilon_i$$

Definindo $x_{i0} = 1$, $i = 1, \cdots, n$, temos,

$$y_i = \sum_{j=0}^{n} \alpha_j \ x_{ij} + \epsilon_i$$

onde ϵ_i para $i \in \{1, 2, \cdots, n\}$ são independentes e cada um tem uma distribuição de probabilidade normal com média 0 e variância σ^2 igual a 1e os parâmetros $\alpha_j, \ j = 0, \cdots, n$ são chamados de coeficientes de regressão. Este modelo descreve um hiperplano no espaço n-dimensional. Assim, se tivermos $x_{i1}, x_{i2}, ..., x_{in}$ e de alguma forma conhecer todos os valores dos parâmetros $\alpha_0, \alpha_1, \cdots, \alpha_n$, poderíamos encontrar o que y_i. Claro, não sabemos $\alpha_0, \alpha_1, \cdots, \alpha_n$em aplicações reais, mas podemos estimar esses parâmetros usando estimação de mínimos quadrados ou outros procedimentos mais complicados, se desejado.

Minimizando a soma dos quadrados dos erros ϵ_i, temos:

$$Q(\alpha_0, \cdots, \alpha_n) = \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} [y_i - \hat{y}_i]^2$$

em relação a $\alpha_0, \cdots, \alpha_n$ obtemos os estimadores de mínimos quadrados $\hat{\alpha}_0, \cdots, \hat{\alpha}_n$, de modo que $\hat{y}_i$ seja um valor de estimado de y_i

$$\hat{y}_i = \hat{\alpha}_0 + \hat{\alpha}_1 x_{i1} + \hat{\alpha}_2 x_{i2} + \cdots + \hat{\alpha}_n \ x_{in}$$

e

$$\epsilon_i = y_i - \hat{y}_i$$

[1]são os resíduos.

[1]

2.2.1.6 Regressão Logística

Seja Y_i a variável de resposta para observações $i = 1, \cdots, n$ das variáveis explicativas $x_{i1}, \cdots, x_{in}$ que são medidas na i-ésima observação. Relacionamos as variáveis explicativas com a variável resposta através do modelo para cada Y_i é dado por

$$Y_i = \alpha_0 + \alpha_1 x_{i1} + \alpha_2 x_{i2} + \cdots + \alpha_n \; x_{in} + \epsilon_i$$

onde ϵ_i para $i = 1, \cdots, n$ são independentes e cada um tem uma distribuição de probabilidade normal com média 0 e variância σ^2.

Tomando a esperança de Y_i, temos

$$E(Y_i) = \alpha_0 + \alpha_1 x_{i1} + \alpha_2 x_{i2} + \cdots + \alpha_n \; x_{in}$$

Estas equações do modelo permite que diferentes valores possíveis de Y_i ou $E(Y_i)$ sejam funções de um conjunto de variáveis explicativas. Assim, se tivermos $x_{i1}, \cdots, x_{in}$ e de alguma forma conhecermos os valores $\alpha_0, \cdots, \alpha_n$, poderíamos encontrar o que Y_i seria em média.

Como dito em Regressão Linear Múltipla, em geral, não conhecemos $\alpha_0, \cdots, \alpha_n$, mas podemos estimar.

No contexto de respostas binárias, a quantidade que queremos estimar é a probabilidade de sucesso, π. Sejam Y_i variáveis de resposta binárias independentes para observações $i \in \{1, 2, ..., n\}$, onde um valor de 1 denota um sucesso e um valor de 0 denota uma falha. Uma distribuição normal obviamente não é mais um modelo apropriado para Y_i. Em vez disso, uma distribuição de Bernoulli descreve Y_i muito bem, mas agora permitimos que o parâmetro de probabilidade de sucesso π_i seja diferente para cada observação. Assim,

$$P(Y_i = y_i) = \pi_i^{y_i}(1 - \pi_i)^{1-y_i}$$

para $y_i = 0$ ou $y_i = 1$ é a função de probabilidade para Y_i.

Para encontrar o estimador de máxima verossimilhança de π_i, a função de verossimilhança é

$$L(\pi_1, \cdots \pi_n | y_1, \cdots, y_n) = P(Y_1 = y_1) \times \cdots \times P(Y_n = y_n) = \prod_{i=1}^{n} \pi_i^{y_i}(1 - \pi_i)^{1-y_i}$$

Observe que existem n parâmetros diferentes para as n observações diferentes. Isso significa que o modelo está saturado: há tantos parâmetros quanto observações. As estimativas de parâmetros neste caso são as mesmas que os dados, $\hat{\pi} = 0 \; ou \; 1$, então não há ganho trabalhando com este modelo.

A ideia é relacionar π_i as variáveis $x_{i1}, \cdots, x_{in}$ propondo uma função matemática. A função $\pi(x_{i1}, \cdots, x_{ik})$ é o modelo

$$\pi_i(x_{i1}, \cdots, x_{ik}) = \frac{\exp\{\alpha_0 + \alpha_1 x_{i1} + \alpha_2 x_{i2} + \cdots + \alpha_k\ x_{ik}\}}{1 + \exp\{\alpha_0 + \alpha_1 x_{i1} + \alpha_2 x_{i2} + \cdots + \alpha_k\ x_{ik}\}}$$

Um modelo de regressão logística também pode ser escrito como

$$\log(\frac{\pi_i}{1-\pi_i}) = \alpha_0 + \alpha_1 x_{i1} + \alpha_2 x_{i2} + \cdots + \alpha_k\ x_{ik}$$

Denotando $logit(\pi_i) = \log(\frac{\pi_i}{1-\pi_i})$, tem-se

$$logit(\pi_i) = \alpha_0 + \alpha_1 x_{i1} + \alpha_2 x_{i2} + \cdots + \alpha_k x_{ik}$$

A estimação por máxima verossimilhança é amplamente utilizada para determinar os parâmetros do modelo de regressão logística. Esse método envolve substituir o modelo na expressão correspondente e aplicar o logaritmo natural para derivar a função de log-verossimilhança:

Para encontrar o estimador de máxima verossimilhança de π_i, a função de verossimilhança é

$$log(L(\pi_1, \cdots \pi_n | y_1, \cdots, y_n)) = log(\prod_{i=1}^{n} \pi_i^{y_i}(1-\pi_i)^{1-y_i})$$

$$\begin{aligned} log(L(\pi_1, \cdots \pi_n | y_1, \cdots, y_n)) &= \sum_{i=1}^{n}(y_i(\alpha_0 + \alpha_1 x_{i1} + \alpha_2 x_{i2} + \cdots + \alpha_k\ x_{ik}) \\ &- \log(1 + exp(\alpha_0 + \alpha_1 x_{i1} + \alpha_2 x_{i2} + \cdots + \alpha_k\ x_{ik})) \end{aligned}$$

Tomamos as derivadas da expressão acima em relação a $\alpha_0, \cdots, \alpha_n$, iguale-os a 0 e resolva-os simultaneamente para obter as estimativas dos parâmetros $\hat{\alpha}_0, \cdots, \hat{\alpha}_n$. Quando as estimativas dos parâmetros são incluídas no modelo, podemos obter a probabilidade estimada de sucesso como

$$\hat{\pi}_i(x_{i1}, x_{i2}, ..., x_{ik}) = \frac{\exp\{\hat{\alpha}_0 + \hat{\alpha}_1 x_{i1} + \hat{\alpha}_2 x_{i2} + \cdots + \hat{\alpha}_k\ x_{ik}\}}{1 + \exp\{\hat{\alpha}_0 + \hat{\alpha}_1 x_{i1} + \hat{\alpha}_2 x_{i2} + \cdots + \hat{\alpha}_k\ x_{ik}\}}$$

2.2.2 Redes Neurais Artificiais

A próxima seção foi baseada na referência Balduíno.

Uma Rede Neural Artificial (RNA) tem seu funcionamento baseado nos neurônios do cérebro, em outras palavras, o conceito que define a RNA seria a interpretação matemática do funcionamento do cérebro e por meio dela é possível a aplicação do aprendizado de máquina.

Assim como o cérebro humano a rede neural possui neurônios e sinapses e formas semelhantes de transmitir informação. Estas semelhanças são próximas sob uma interpretação prática. Existem diferentes modelos de redes neurais, essa variedade se justifica tanto pelos

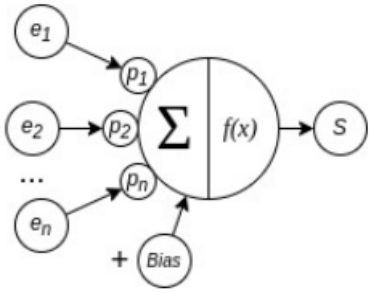
vários problemas computacionais quanto pela evolução destas. Não cabe ao escopo deste trabalho um detalhamento de diferentes modelos de redes neurais ou da evolução histórica destes estudos, o que se segue é uma apresentação didática do funcionamento básico de uma rede neural com os modelos primitivos e a discriminação do modelo de rede neural *Long Shrot Term Memory* (LSTM).

2.2.2.1 Rede neural simplificada

O modelo simplificado de uma RNA é o *perceptron*, sua arquitetura consiste em uma camada de neurônios de entrada, responsável pelo recebimento de cada informação. A quantidade de neurônios desta camada é equivalente com a quantidade de informações que serão passadas à rede por vez. Além desta, uma camada de neurônios de saída, responsável pela conclusão da rede, esta camada pode conter desde um, a vários neurônios, dependendo do problema computacional que a rede foi desenvolvida para resolver. Todos os neurônios da camada de entrada são ligados a cada neurônio da camada de saída. Estas ligações recebem o nome de sinapses. O funcionamento desta rede acontece justamente pela capacidade das sinapses atribuírem diferentes pesos para cada ligação, e através do resultado destas atribuições com a soma de um determinado número b, chamado bias, tirarem conclusões a cerca de cada informação. Um conceito matemático semelhante a este processo de atribuição de pesos é a multiplicação de matrizes.

Um neurônio artificial tem entrada pode ser interpretada como uma matriz linha $I = (e_j),\ j \in \{1 \cdots n\}$ e os pesos uma matriz coluna $P = (p_j),\ j \in \{1 \cdots n\}$ (cujos dados nós desconhecemos), resultando $x = \sum_{j=1}^{n} p_j e_j + b$ e aplicando a função de ativação produzimos a saída $S = f(x)$.

Figura 11 – Neurônio Artificial



Fonte: Adaptado de Geron (2019).

$$\begin{bmatrix} e_1 & e_2 . & . & . & e_n \end{bmatrix} \times \begin{bmatrix} p_1 \\ p_2 \\ . \\ . \\ p_n \end{bmatrix} + b = \begin{bmatrix} X \end{bmatrix} \Rightarrow S = f(X) \tag{2.14}$$

2.2.2.2 Multilayer Perceptron

Seja dada uma rede neural com n=2 entradas de duas camadas ocultas com m=3 neurônios cada camada e l=2 saídas. Seja dada as entradas $I = (e_j),\ j \in \{1 \cdots n = 2\}$ e os pesos uma matriz coluna $P = (p_j),\ j \in \{1 \cdots m = 6\}$, resultando em m dados

$$x_1 = p_1 e_1 + p_4 e_2 + b_1,$$

$$x_2 = p_2 e_1 + p_5 e_2 + b_1,$$

$$x_3 = p_3 e_1 + p_6 e_2 + b_1,$$

e aplicando a função de ativação produzimos m saídas da primeira camada oculta

$$y_1^{(1)} = f(x_1),$$
$$y_2^{(1)} = f(x_2),$$
$$y_3^{(1)} = f(x_3),$$

agora novamente aplica-se os pesos da segunda camada oculta $W = (w_j),\ j \in \{1 \cdots m = 9\}$ em y_i produzimos

$$x_7 = w_1 y_1^{(1)} + w_4 y_2^{(1)} + w_7 y_3^{(1)} + b_2,$$
$$x_8 = w_2 y_1^{(1)} + w_5 y_2^{(1)} + w_8 y_3^{(1)} + b_2,$$
$$x_9 = w_3 y_1^{(1)} + w_6 y_2^{(1)} + w_9 y_3^{(1)} + b_2,$$

e aplicando a função de ativação produzimos m saídas da segunda camada oculta

$$y_1^{(2)} = f(x_7),$$
$$y_2^{(2)} = f(x_8),$$
$$y_3^{(2)} = f(x_9),$$

agora novamente aplica-se os pesos da segunda camada oculta $Q = (q_j),\ j \in \{1 \cdots m = 6\}$ em y_i produzimos

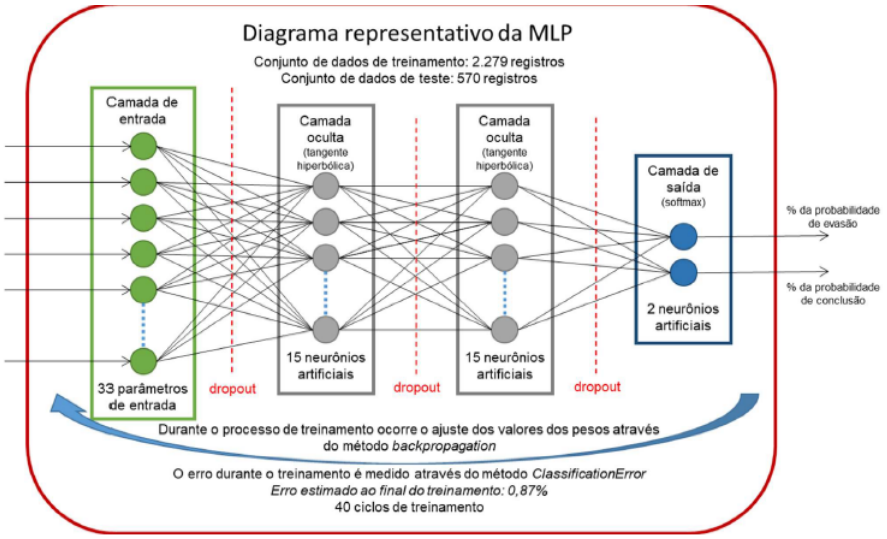
$$x_{10} = q_1 y_1^{(2)} + q_3 y_2^{(2)} + q_5 y_3^{(2)} + b_3,$$
$$x_{11} = q_2 y_1^{(2)} + q_4 y_2^{(2)} + q_6 y_3^{(2)} + b_3,$$

finalmente aplicamos a função de ativação f

$$S_1 = f(x_{10}),$$

$$S_2 = f(x_{11}).$$

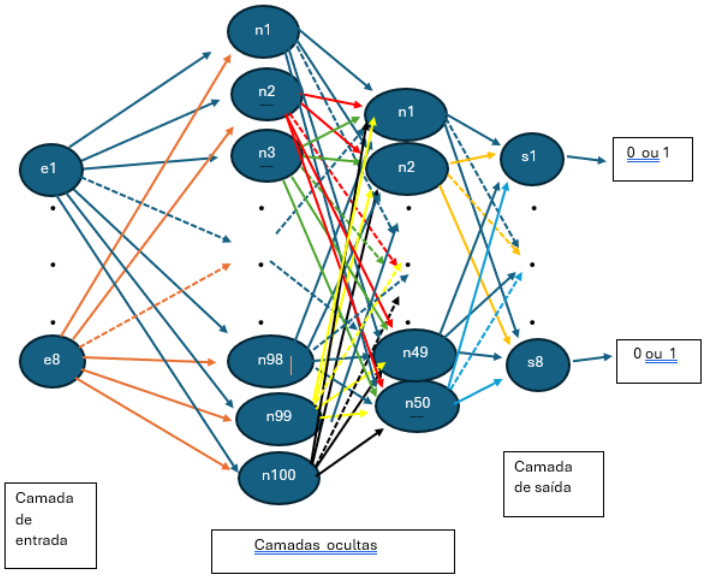
Figura 12 – Rede neural Multilayer Perceptron



Fonte: Alan Lopes (MELO, 2019).

O modelo consiste em 8 neurônios na camada de entrada, 100 neurônios na primeira camada oculta e 50 neurônios na segunda camada oculta 8 neurônios na camada de saída.

Figura 13 – Entrada de dados e portão de esquecimento.



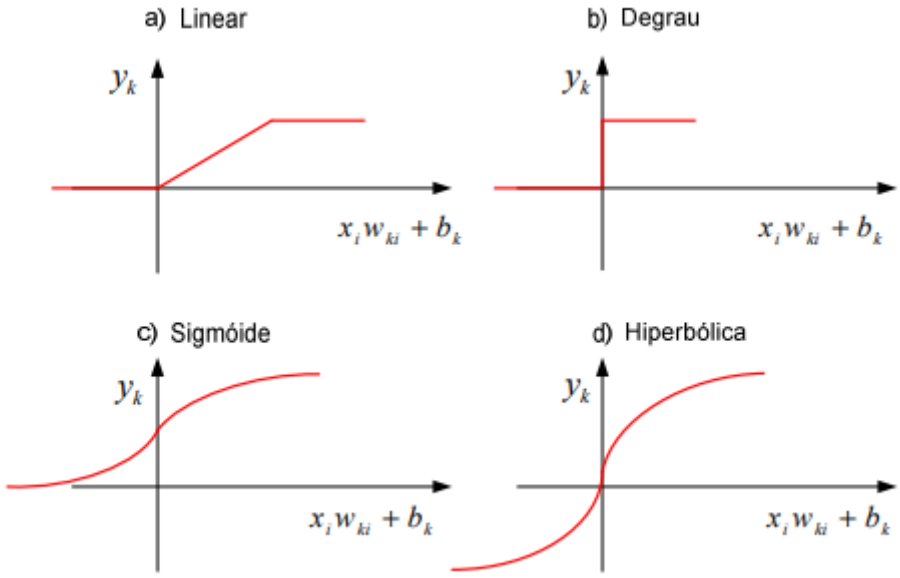
Fonte: Próprio Autor

2.2.2.2.1 Algumas funções de Ativação

As funções de ativação são um elemento extremamente importante das redes neurais artificiais. Elas basicamente decidem se um neurônio deve ser ativado ou não. Ou seja, se

a informação que o neurônio está recebendo é relevante para a informação fornecida ou deve ser ignorada. Veja na fórmula abaixo como a função de ativação é mais uma camada matemática no processamento.

Figura 14 – Funções de Ativação



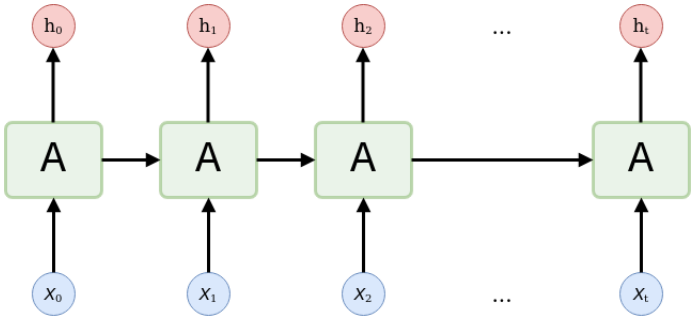
Fonte: Adaptado de Geron (2019).

2.2.2.3 Redes neurais complexas

Esta seção foi baseada nas referências (JUNIOR, 2019) e no TCC de Vinícios Antônio Passos Balduíno (BALDUíNO, 2023)

Com o desenvolvimento de novas arquiteturas de redes um novo conceito também surgiu, as redes neurais recorrentes RNR. Enquanto no exemplo de rede anterior a informação era tratada linearmente, entrando pela camada de entrada e seguindo as várias camadas ocultas que a rede pode apresentar até a camada de saída. Em uma RNR a informação não necessariamente é tratada de maneira linear, os neurônios da camada de entrada podem receber informação da camada de saída, bem como os neurônios das camadas ocultas podem trocar informação entre si. Arquiteturas assim expõe a rede a mais dados, o que pode implicar na melhor distribuição de pesos para as informações.

Figura 15 – Rede neural recorrente

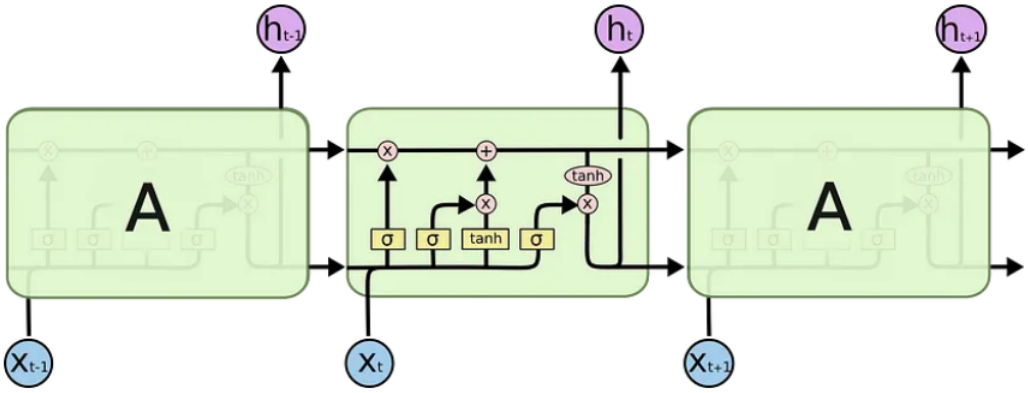


Fonte: TCC Vinícios (BALDUíNO, 2023)

Na RNR a cada interação a informação da camada anterior influencia tanto quanto a informação que entra pela camada de entrada. Devido a isso, a informação de camadas distantes acaba sendo diluída no decorrer das interações e pode ser que a informação que entra tenha sua relevância diminuida.

Graças à sua arquitetura, a rede LSTM se diferencia bastante das redes neurais convencionais, não possuindo camadas de neurônios e sim, células de memória. Estas células possuem sinapses assim como os neurônios das redes anteriores, porém tais sinapses não possuem a tarefa de fazer atribuição de pesos e sim transmitir a informação. A distribuição de pesos pela rede LSTM acontece dentro das próprias células de memória, assim como mostra a figura 16.

Figura 16 – Arquitetura de célula do tipo LSTM.



Fonte: Medium (JUNIOR, 2019)

Pode se observar na figura 16 que a célula de memória LSTM possui elementos dentro dela. Estes elementos possibilitam que ocorram manipulações das informações dentro da célula.

- As setas possuem a função de carregar a informação de forma numeral e concatenando-as aos pesos.

- Os retângulos amarelos são as funções de ativação, possuem o papel de transformar os valores anteriores. Nesta arquitetura é representada as fungões sigmoid 2.15 que sempre gera resultados entre 0 e 1 e tangente hiperbólica 2.16 gerando resultados entre -1 e 1.

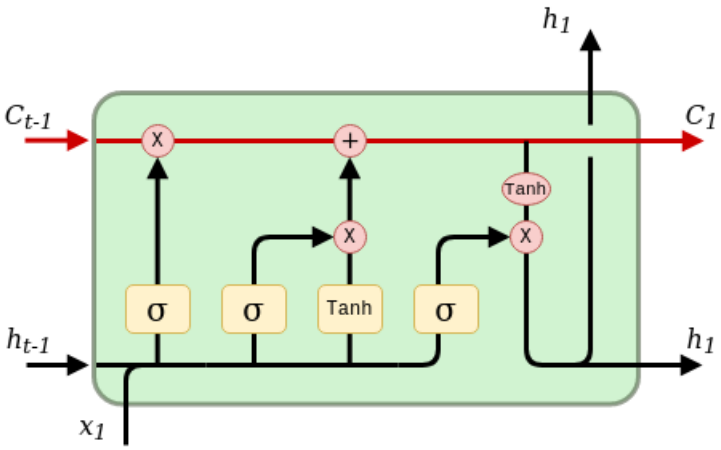
$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{2.15}$$

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.16}$$

- As elipses rosas dentro da célula de memória são responsáveis pela concatenação das informações.

Cada célula de memória desta está conectada com a sua anterior e posterior, isso faz com que a informação seja transmitida entre elas. O caminho superior na célula, destacado na figura 17 é o estado da memória, ele é responsável por armazenar todas as conclusões que a rede tira a partir dos dados. O estado da memória de uma célula pode ser passado para a seguinte, se passado isso faz com que a célula atual utilize das conclusões da célula anterior para chegar nas próprias. Podemos observar que o estado da memória está conectado com todos os demais processos da célula, logo cada informação que a célula recebe e devolve influencia e é influenciada por ele.

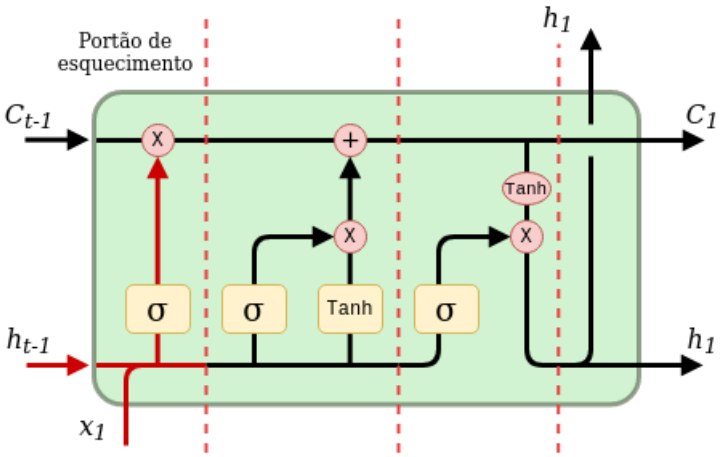
Figura 17 – Estado da memória da célula LSTM.



Fonte: TCC Vinícios (BALDUíNO, 2023)

Na parte inferior esquerda da arquitetura existe entrada de dados, tanto dos novos, quanto os dados de saída da célula anterior e são concatenados. A primeira interação dos dados com a célula acontece no portão de esquecimento, destacado na figura 21, para este portão é atribuído a letra f de *forget*. O portão de esquecimento determina o quanto de informação será perdida do estado da memória anterior, para que assim possa receber novas informações.

Figura 18 – Entrada de dados e portão de esquecimento.



Fonte: TCC Vinícios (BALDUíNO, 2023)

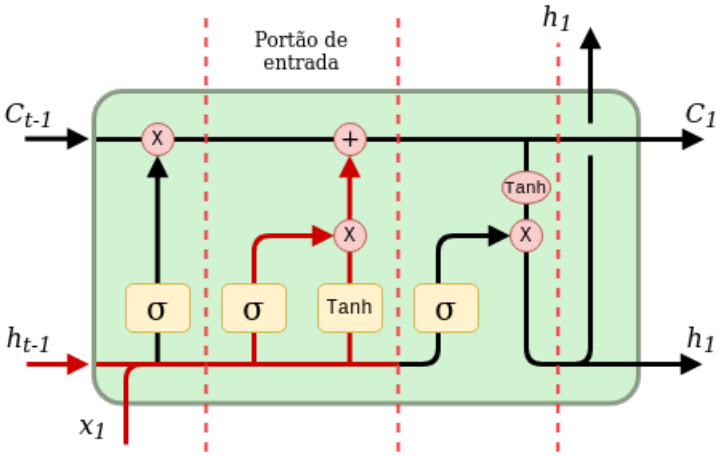
A equação 2.17 descreve o funcionamento do primeiro portão sobre os dados.

$$f_t = \sigma(W_f \times [h_{t-1}, x_t] + b_f) \tag{2.17}$$

W_e é a matriz de pesos do portão de esquecimento, b_e é bias do portão de esquecimento, ele é somado ao resultado a fim de auxiliar no balanceamento. A saída desta equação é um vetor com números número entre 0 e 1, este vetor logo é multiplicado pelo estado de memória anterior, fazendo com que a informação seja retirada do estado da memória.

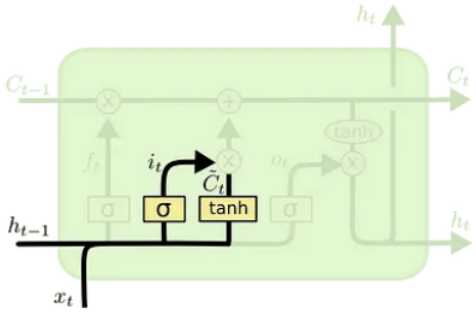
Após o portão de esquecimento tem o de entrada apresentado na figura 19. Este portão possui duas partes, para a primeira é aplicado uma função sigmoid nos dados, esta parte é representada pela letra i de *input*, equação 2.21. Na segunda parte é aplicado uma função tangente hiperbólica nos mesmos dados esta parte é representada pela letra $\hat{C}$ por se tratar de um estado de memória parcial, equação 2.19. Neste portão são selecionadas as informações que serão armazenadas no estado da memória e o quanto elas influenciarão o estado.

Figura 19 – Portão de entrada.



Fonte:TCC Vinícios (BALDUíNO, 2023)

Figura 20 – Entrada de dados e portão de esquecimento.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Fonte: Medium (JUNIOR, 2019)

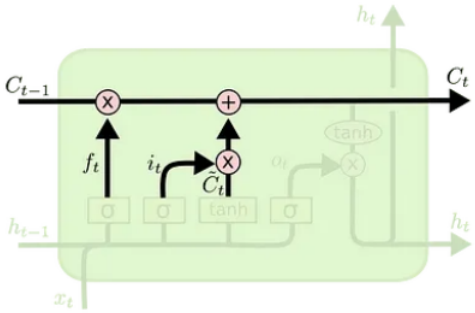
O funcionamento deste portão é descrito pelas duas equações seguintes.

$$i_t = \sigma(W_i \times [h_{t-1}, x_t] + b_i) \tag{2.18}$$

$$\hat{C}_t = tanh(W_{\hat{C}} \times [h_{t-1}, x_t] + b_{\hat{C}}) \tag{2.19}$$

Em ambos os casos W e b representam respectivamente, a matriz de pesos e o bias. As saídas das equações são multiplicada e o vetor resultante deste portão possui números entre 0 e 1, e em seguida este vetor é concatenado com o estado de memória anterior, formando assim o estado atual da memória, equação 2.20.

Figura 21 – Entrada de dados e portão de esquecimento.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

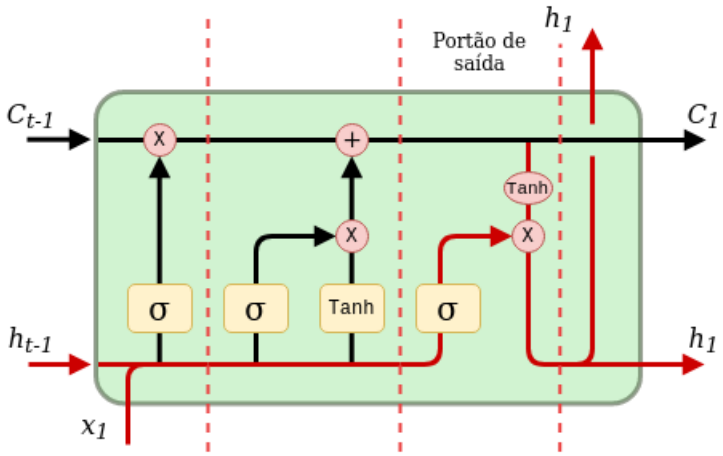
Fonte: Medium
$https://medium.com/@web2ajax/redes-neurais-recorrentes-lstm-b90b720dc3f6$

$$C_t = f_t \times C_{t-1} + i_t \times \hat{C}t \tag{2.20}$$

Para concluir a interação com a célula, os dados precisam passar pelo portão de saída, apresentado na figura 22. Neste momento a informação que entrou no começo da célula sofre todas as distribuições de peso conforme o estado da célula atual, gerando assim novos

dados a serem passados para a próxima célula ou a conclusão do processo, caso seja a ultima célula.

Figura 22 – Portão de saída.



Fonte:TCC Vinícios Antônio Passos Balduíno (BALDUíNO, 2023)

O funcionamento deste portão é descrito pelas duas equações seguintes.

$$o_t = \sigma(W_i \times [h_{t-1}, x_t] + b_i) h_t = o_t \times tanh(C_t) \tag{2.21}$$

2.3 CORRELAÇÃO

O texto desta seção foi baseado no livro (FáVERO; BELFIORE, 2017) e nos textos publicados na internet de testes de normalidade em (FERRAZ, ; AMORIM,) e testes de correlação (BATISTA,).

Uma correlação descreve a associação entre duas variáveis, ou seja, como duas variáveis mudam em conjunto, porém a existência de uma correlação significante entre duas variáveis não implica necessariamente que exista uma relação causal entre ambas.

Vamos destacar dois tipos de correlação:

1. Coeficiente de correlação de Pearson;
2. Coeficiente de correlação de Spearman; e
3. Coeficiente de correlação de Kendall.

Os coeficientes de correlação são números reais entre -1 e 1, isto é, quando existe a correlação ela pode ser negativa ou positiva.

Após calcular o coeficiente de correlação, podemos interpretar seu valor da seguinte maneira:

- Se o valor de r está próximo de -1, indica uma correlação negativa perfeita, ou seja, quando uma variável aumenta, a outra diminui de forma linear.
- Se o valor de r está próximo de 1, indica uma correlação positiva perfeita, ou seja, quando uma variável aumenta, a outra também aumenta de forma linear.
- Se o valor de r está próximo de 0, indica uma ausência de correlação linear entre as variáveis.

2.3.1 Correlação de Pearson

A correlação de Pearson avalia a relação linear entre duas variáveis contínuas. Uma relação é linear quando a mudança em uma variável é associada a uma mudança proporcional na outra variável.

Para realizar a correlação de Pearson, é necessário cumprir o seguinte:

- A escala de medição deve ser uma escala ou relação de intervalo;
- As variáveis devem ser aproximadamente distribuídas;
- A associação deve ser linear;
- Não deve haver valores atípicos nos dados, isto é, outliers.

Importância da correlação de Pearson na análise de dados:

- **Identificação de tendências:** O coeficiente de correlação de Pearson nos ajuda a identificar se existe uma relação linear entre duas variáveis. Isso nos permite entender tendências e padrões em nossos dados.
- **Previsão:** Com base na correlação, podemos usar uma variável para prever o comportamento de outra. Isso é especialmente útil em modelos de aprendizado de máquina, onde a correlação pode ser um indicador importante para o desempenho do modelo.
- **Seleção de atributos:** A correlação de Pearson pode ajudar na seleção de atributos relevantes. Se duas variáveis estão altamente correlacionadas, podemos considerar a remoção de uma delas para evitar redundância e melhorar a eficiência do modelo.
- **Análise de causalidade:** Embora a correlação não indique causalidade direta entre as variáveis, ela pode ser um primeiro passo para investigar relacionamentos mais profundos e fornecer "‘insights"’ para futuras pesquisas.

Definição 2.3.1 (**Correlação**). *O coeficiente de correlação de Pearson que mede a dependência linear entre duas variáveis é definido como:*

$$\rho = \rho(X,Y) = Corr(X,Y) = \frac{Cov(X,Y)}{\sigma(X)\sigma(Y)} \tag{2.22}$$

A coeficiente de correlação de Pearson é dada pelo

$$\rho = \frac{Cov(X,Y)}{\sigma(X)\sigma(Y)}$$

onde $Cov(X,Y)$ é a covariância entre X e Y e σ é a variância. Porém, embora seja desconhecido, o coeficiente populacional ρ pode ser estimado através do coeficiente amostral r, calculado a partir de uma amostra retirada de uma população normal bidimensional (Bussab e Morettin, 2002).

Para se avaliar se a correlação populacional é nula ou não, são construídas as hipóteses nula (Ho) e alternativa (Ha), e as hipóteses são arranjadas como segue:

$$\begin{cases} H_o : \rho = 0 \\ H_a : \rho \neq 0 \end{cases}$$

Em Ho, as variáveis são independentes, logo a correlação entre elas é nula; e em Ha, as variáveis são dependentes, e então, existe alguma correlação não nula. O Teste de Correlação Momento Produto de Pearson, mais conhecido como Teste de Correlação de Pearson, é adequado para testar tais hipóteses. O teste é baseado na estatística r dada na equação 2.24

$$r = r(X,Y) = \frac{\sum_{i=1}^{N}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{(\sum_{i=1}^{N}(X_i - \overline{X})^2)}\sqrt{(\sum_{i=1}^{N}(Y_i - \overline{Y})^2)}} \tag{2.23}$$

- $-1 \leq \rho_{x,y} \leq 1$;
- $\rho_{x,y} = \rho_{y,x}$;
- $\rho_{x,y} = 0$ quando as duas variáveis não são correlacionada.

2.3.2 Correlação de Spearman

A correlação de Spearman é uma medida de correlação não paramétrica que avalia a relação monotônica entre duas variáveis, ou seja, não assume que as variáveis estão distribuídas normalmente. É comumente usada quando as variáveis apresentam uma relação não-linear ou quando há presença de outliers.

A correlação de Spearman avalia a relação monotônica entre duas variáveis contínuas ou ordinais. Em uma relação monotônica, as variáveis tendem a mudar juntas mas não necessariamente a uma taxa constante. O coeficiente de correlação de Spearman baseia-se nos valores classificados de cada variável, em vez de os dados brutos.

A correlação de Spearman é muito usada para avaliar relações envolvendo variáveis ordinais. Por exemplo, você poderia usar a correlação de Spearman para avaliar se a ordem na qual os funcionários executam um teste está relacionada ao número de meses de emprego.

Posto é a posição do valor quando se colocam por ordem valores (possivelmente reais) ou categorias (com escala ordinal). Por exemplo, se $X = (4, 8, 2, 5\}$ então $R(X) = \{2, 4, 1, 3\}$.

$$\begin{aligned} \rho_s = \rho(R(X), R(Y)) &= \frac{Cov(R(X), R(Y))}{\sigma(R(X))\sigma(R(Y))} \\ &= \frac{\sum_{i=1}^{N}(R(X_i) - R(\overline{X}))(R(Y_i) - R(\overline{Y}))}{\sqrt{(\sum_{i=1}^{N}(R(X_i) - R(\overline{X}))^2)}\sqrt{(\sum_{i=1}^{N}(R(Y_i) - R(\overline{Y}))^2)}} \end{aligned}$$

onde

- ρ denota o usual coeficiente de correlação de Pearson, mas aplicado as variáveis nos postos;
- $Cov(R(X), R(Y))$ é a covariância das variáveis nos postos;
- $\sigma(R(X))$ e $\sigma(R(Y))$ são os desvios padrão das variáveis nos postos.
- $R(X_i)$ e $R(Y_i)$ são as ordens das variáveis X e Y, respectivamente.

Uma fórmula fácil para calcular o coeficiente de Spearman é dada por:

$$\rho_s = \frac{\sum_{i=1}^{n}(R(X_i) - \frac{n+1}{2})(R(Y_i) - \frac{n+1}{2})}{\frac{n(n^2+1)}{12}} \tag{2.24}$$

onde n é o número de pares (X_i, Y_i).

2.4 CORRELAÇÃO DE PEARSON × SPEARMAN

Para se utilizar a correlação de Pearson os dados:

- O coeficiente de correlação de Pearson não diferencia entre variáveis independentes e variáveis dependentes, isto é, a correlação entre X e Y e igual a correlação entre Y e X.
- A correlação entre X e Y exige que as variáveis sejam quantitativas (continuas ou discretas);
- Os valores observados das variáveis X e Y precisam estar normalmente distribuídos. Este pressuposto é especialmente importante em amostras pequenas ($n \leq 30$), devido ao teorema central do limite;
- necessário uma análise sobre a presença de outliers (valores atípicos nos dados), o coeficiente é fortemente afetado pela presença deles.
- Necessário a independência das observações, ou seja a ocorrência de uma observação X_1 não influencia a ocorrência de outra observação X_2.

A violação desses pressupostos pode comprometer os resultados, levando o pesquisador a concluir que não existe relação entre as variáveis, enquanto existe relação ou o contrário poderia concluir que existe relação entre as variáveis, enquanto não existe relação.

O coeficiente de correlação de Pearson (r) é fortemente influenciado pela média da distribuição. Por esse motivo, um dos pressupostos centrais para que essa medida seja adequadamente utilizada é de que as observações obedeçam a uma distribuição normal. Existem testes estatísticos e observações gráficas disponíveis para averiguar em que medida as observações estão normalmente distribuídas.

2.4.1 Correlação de Kendell

Considere (X_i, Y_i) pares de observações das variáveis aleatórias conjuntas X e Y respectivamente, tal que todos os valores de X_i e Y_i sejam únicos. Qualquer par de observação (X_i, Y_i) e (X_j, Y_j), em que $i \neq j$, é concordante se as classificações de ambos os elementos concordarem uma com a outra, isto é, se $X_i > X_j$ e $Y_i > Y_j$ ou se $X_i < X_j$ e

$Y_i < Y_j$. Elas são discordantes se $X_i > X_j$ e $Y_i < Y_j$ ou $X_i < X_j$ e $Y_i > Y_j$. Se $X_i = X_j$ ou $Y_i = Y_j$, o par não é nem concordante, nem discordante.

O estimador do coeficiente de correlação por postos de Kendall é dado pela fórmula:

$$\tau_k = \frac{(quantidade\ de\ pares\ concordantes) - (quantidade\ de\ pares\ discordantes)}{\frac{n(n^2+1)}{12}} \quad (2.25)$$

ou

$$\tau = \sum_{pares(i,j), i<j} \frac{\sum (R(X_i) - R(X_j) \sum (R(Y_i) - R(Y_j))}{\frac{n(n^2+1)}{2}} \quad (2.26)$$

onde

- $R(X_i)$ e $R(Y_i)$ são as ordens das variáveis X e Y, respectivamente.
- n é o número de elementeos aos quais se atribuíram postos em X e Y.

2.4.2 Correlação Ponto-bisserial

Existem três tipos de correlações mais conhecidas e frequentemente utilizadas, que são a correlação de Pearson r_{XY}, a correlação de Spearman ρ_s e correlação de Kendall τ. Cada uma delas possuem suas particularidades e melhores casos de utilização. Mas elas possuem algo em comum: não podem ser utilizadas com variáveis categóricas a nível nominal.

Pela própria definição, uma variável categórica nominal é caracterizada pelo nome da categoria à qual pertence, por exemplo, conclusão ou não do curso, status civil (solteiro, casado, divorciado, etc) e sexo(homem ou mulher). Variáveis deste tipo não podem ser quantificadas por números, porém é comum representá-las por códigos numéricos, do tipo 0=evasão e 1=conclusão. Mas códigos numéricos são apenas uma maneira abreviada de se referir às categorias e não tem qualquer sentido considerá-los como valores em um estudo de correlação.

Quando a variável resposta (label ou Target) é dicotômica (binária). Utiliza-se a correlação Ponto-Bisserial.

O coeficiente de correlação ponto bisserial (r_{pb}) é um coeficiente de correlação utilizado quando uma variável (por exemplo, Y) é dicotômica; Y pode ser "naturalmente"dicotômica, como se o lançamento de uma moeda resulta em cara ou coroa, ou uma variável dicotomizada artificialmente. Na maioria das situações não é aconselhável dicotomizar variáveis artificialmente. Quando uma nova variável é dicotomizada artificialmente a nova variável dicotômica pode ser concebido como tendo uma continuidade subjacente. Se este for o caso, uma correlação bisserial seria o cálculo mais apropriado.

A correlação ponto-bisserial é matematicamente equivalente à correlação de Pearson (produto-momento) correlação, isto é, se temos uma variável X medida continuamente e

uma variável Y dicotômica, $r_{XY} = r_{pb}$. Isso pode ser demonstrado através da atribuição de dois valores numéricos diferentes para a variável dicotômica.

Para calcular r_{pb}, suponha que a variável dicotômica Y assuma os valores 0 e 1. Se o conjunto de dados for dividido em dois grupos, o grupo 1, em que Y recebeu o valor "1"e o grupo 2, em que Y recebeu o valor "0", então o coeficiente de correlação ponto-bisserial é calculado da seguinte forma:

$$r_{pb} = \frac{M_1 - M_0}{s_n}\sqrt{\frac{n_1 n_0}{n^2}} \tag{2.27}$$

em que s_n é o desvio padrão utilizado quando os dados estão disponíveis para todos os membros da população:

$$s_n = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2} \tag{2.28}$$

sendo M_1 o valor médio da variável contínua X para totos os pontos de dados do grupo 1, e M_0 o valor médio da variável continua X para todos os pontos de dados do grupo 2. Além disso, n_1 é o número de pontos de dados no grupo 1, n_0 é o número de pontos de dados no grupo 2 e n é o tamanho total da amostra. Esta fórmula é uma fórmula que foi derivada a partir da fórmula para r_{XY}.

2.4.3 Normalidade dos Dados

Podemos observar a normalidade dos dados de duas formas: de maneira visual e através de testes estatísticos.

2.4.3.1 Métodos Visuais

São métodos para uma verificação rápida da normalidade dos dados, porém não garantem que a distribuição seja normal.

Utilizaremos o python para visualizar o histograma. Para isso devemos iniciar algumas bibliotecas

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import statsmodels.api as sm
5 from matplotlib import pyplot as plt
```

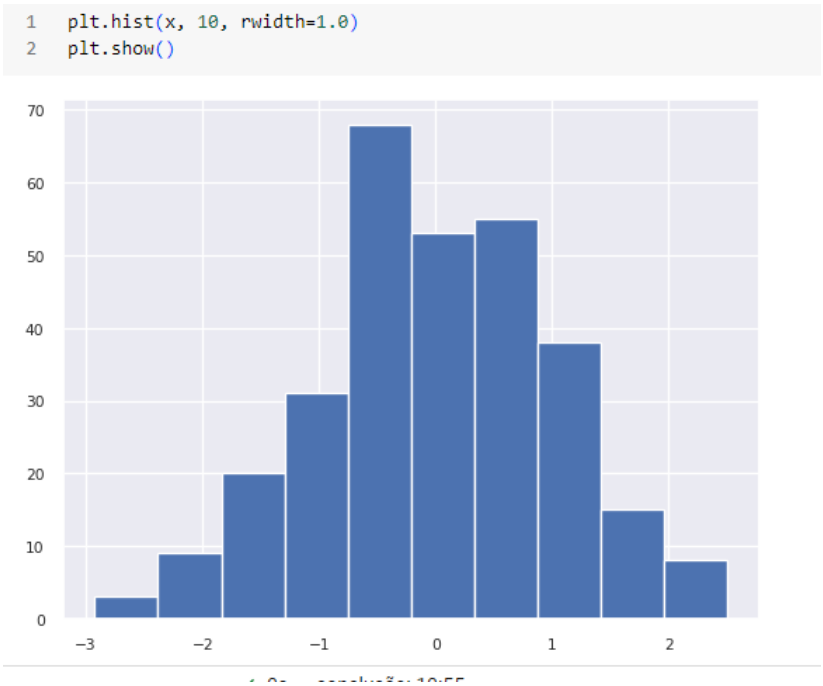
2.4.3.1.1 Histograma

O histograma é uma forma de visualização da frequência de uma variável.

Vamos agora simular dois conjuntos de dados, um gerado a partir de uma distribuição normal

```
x = np.random.normal(size=300)
plt.hist(x, 10, rwidth=1.0)
plt.show()
```

Figura 23 – Evasão depois do segundo período



Fonte: Produzido pelo autor tirados dos dados do SISCAD/UFTM

e outro gerado a partir de uma distribuição uniforme.

```
y = np.random.uniform(size=300)
plt.hist(x, 10, rwidth=1.0)
plt.show()
```

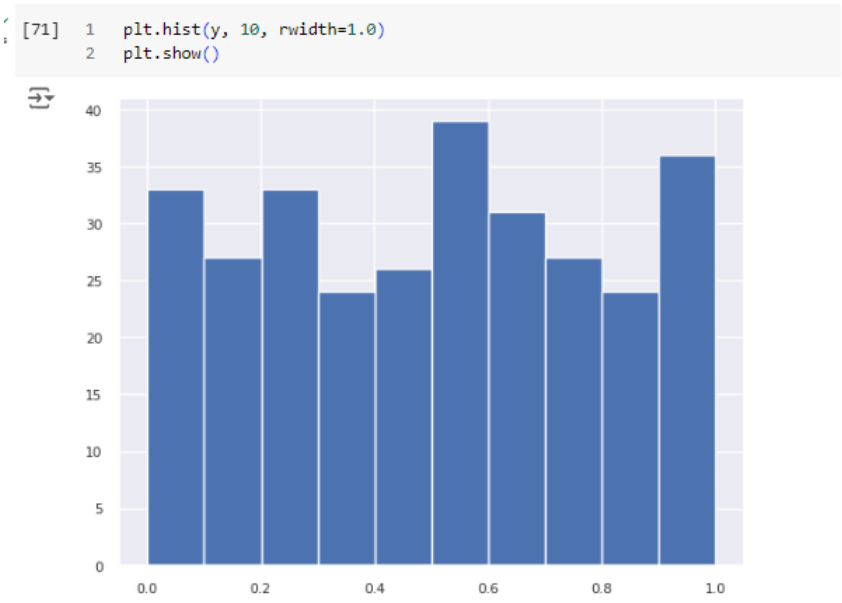
2.4.3.1.2 QQ plot

O gráfico Quantile-Quantile plot - QQ plot, serve para verificar se os dados seguem uma distribuição específica. No eixo vertical (y) colocamos as posições reais de nossos dados, já no eixo horizontal (x) são colocadas as posições esperadas (valores teóricos) caso os dados seguissem a distribuição escolhida.

Se as distribuições são semelhantes, então os pontos deverão cair em cima da linha vermelha (diagonal). Caso contrário, eles estarão bem afastados.

```
x = np.random.normal(size=300)
sm.qqplot(x,line='s')
```

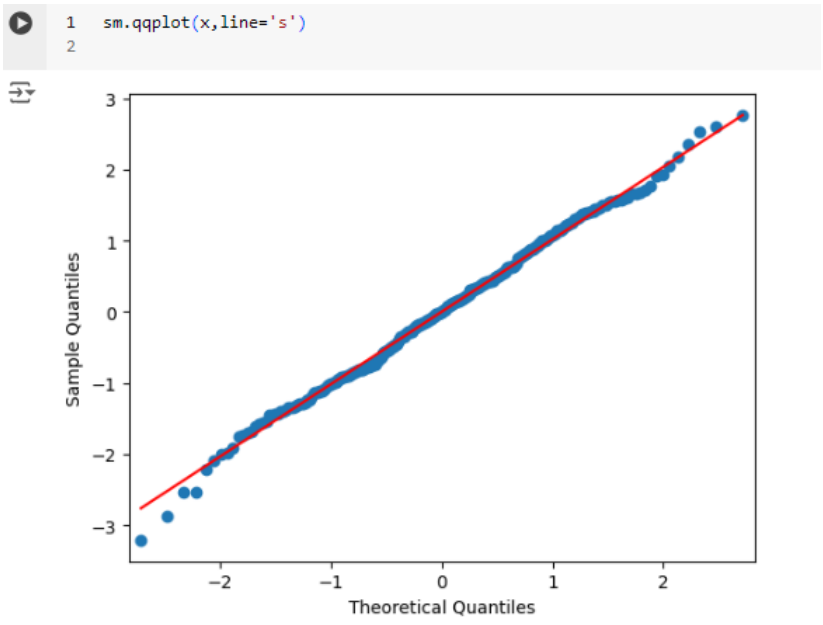
Figura 24 – Evasão depois do segundo período



Fonte: Produzido pelo autor tirados dos dados do SISCAD/UFTM

QQ plot da distribuição normal

Figura 25 – Evasão depois do segundo período

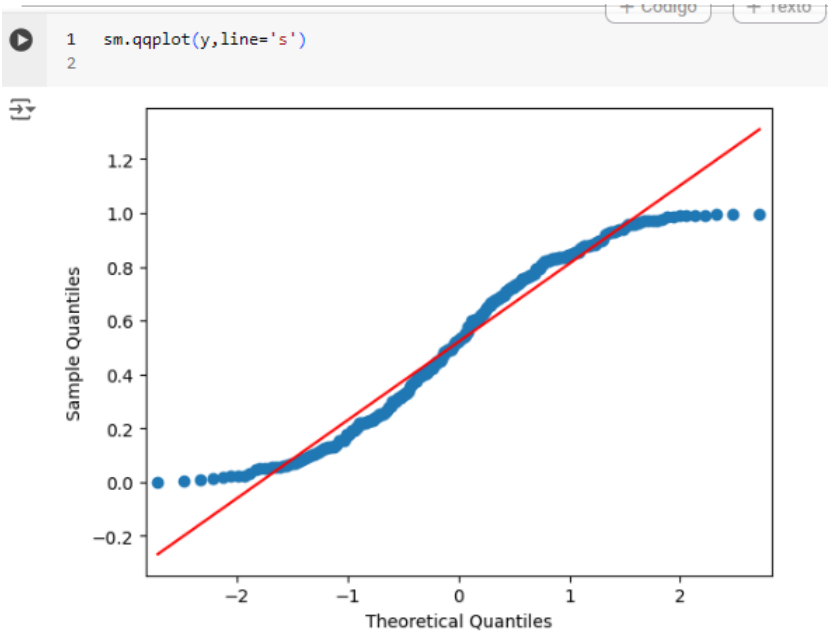


Fonte: Produzido pelo autor tirados dos dados do SISCAD/UFTM

QQ plot da distribuição uniforme

```
y = np.random.uniform(size=300)
sm.qqplot(x,line='s')
```

Figura 26 – Evasão depois do segundo período



Fonte: Produzido pelo autor tirados dos dados do SISCAD/UFTM

2.4.3.2 Teste Estatísticos da Normalidade dos Dados

Para verificar a normalidade univariada dos dados, os testes mais utilizados são os de Kolmogorov-Smirnov e de Shapiro-Wilk.

2.4.3.2.1 Kolmogorov-Smirnov

O teste de Kolmogorov-Smirnov (K-S) é um teste de aderência, isto é, compara a distribuição de frequências acumuladas de um conjunto de valores amostrais com a uma distribuição teórica. O objetivo é testar se os valores amostrais são oriundos de uma população com suposta distribuição teórica ou esperada, neste caso a distribuição normal. A estatística do teste é o ponto de maior diferença entre as duas distribuições.

Para utilizar o teste de K-S, a média e o desvio-padrão da população devem ser conhecidos. Para pequenas amostras, o tese perde potência, de modo que deve ser utilizado em amostras grandes ($n \geq 30$).

O teste de K-S assume as seguintes hipóteses:

H_0 : a amostra provém de uma população com distribuição normal $N(\mu, \sigma)$

H_1 : a amostra não provém de uma população com distribuição normal $N(\mu, \sigma)$

Seja $F_{esp}(X)$ uma função de distribuição esperada (normal) de frequências relativas acumuladas da variável X, em que $F_{esp}(X) \approx N(\mu, \sigma)$, e $F_{obs}(X)$ a distribuição de frequências relativas acumuladas observada da variável X. O objetivo é testar se $F_{obs}(X) = F_{esp}(X)$, contra a alternativa de que $F_{obs}(X) \neq F_{esp}(X)$.

A estatística do teste é:

$$D_{cal} = max\left\{\left|F_{esp}(X_i) - F_{obs}(X_i)\right|; \left|F_{esp}(X_i) - F_{obs}(X_{i-1})\right|\right\},\ para\ i = 1, \cdots, n.$$

em que:

$F_{esp}(X_i)$: frequência relativa acumulada esperada na categoria i; $F_{obs}(X_i)$: frequência relativa acumulada observada na categoria i; $F_{obs}(X_{i-1})$: frequência relativa acumulada observada na categoria i-1;

Os valores críticos da estatística de Kolmogorov-Smirnov (D_c) estão numa tabela G que fornece os valores críticos de D_c tal que $P(D_{cal} > D_c) = \alpha$ (para um teste unilateral à direita). Para que a hipótese nula H_0 seja rejeitada, o valor da estatística (D_{cal}) deve pertencer à região crítica, isto é, $D_{cal} > D_c$; caso contrário, não rejeitamos H_0.

O $Pvalue$ (probabilidade associada ao valor da estatística calculada D_{cal} a partir da amostra) também pode ser obtido da tabela G. Neste caso, rejeitamos H_0 se $Pvalue \leq \alpha$.

No caso do teste, um resultado não significativo $Pvalue > 0.05$ indica normalidade, caso contrário $Pvalue \leq 0.05$ indica que o pressuposto de normalidade foi violado.

Kolmogorov-Smirnov no Python

```
1 import statsmodels
2 statsmodels.stats.diagnostic.lilliefors(df['QtDisAprovadasSemestre'
    ], dist='norm', pvalmethod='table')
```

2.4.3.2.2 Shapiro-Wilk

O teste de Shapiro-Wilk é um teste estatístico mais popular para o cálculo da normalidade. Ele pode ser utilizado independente do número de amostras. No entanto, caso a amostra seja pequena (entre 4 a 30 registros), é preferível utilizar este teste.

O teste de Shapiro-Wilk assume as seguintes hipóteses: H_0 : a amostra provém de uma população com distribuição normal $N(\mu, \sigma)$
H_1 : a amostra não provém de uma população com distribuição normal $N(\mu, \sigma)$

Ele avalia dados de uma amostra com a hipótese nula de que o conjunto de dados é normalmente distribuído. Um valor p grande indica que o conjunto de dados é distribuído normalmente, um valor p baixo indica que não é distribuído normalmente.

O teste de Shapiro-Wilk testa a hipótese nula que uma amostra $x_1,\ x_2, \cdots,\ x_n$, retirada de uma população, tem distribuição normal.

Para calcular o valor da estatística W, dada a amostra aleatória, de tamanho n, deve-se proceder da seguinte maneira:

1. Obter uma amostra ordenada: $x_1 \leq x_2 \leq \cdots \leq x_n$;

2. Calcular a soma de quadrado dos desvios: $S^2 = \sum_{i=1}^{n}(x_i - \bar{x})^2$

3. Uma vez que se tem o número de amostras (n) coletadas, tem-se que:

[3.1] se n é par, $n = 2k$, calcule: $b = \sum_{i=1}^{n} a_{n-i+1}(x_{n-i+1} - x_i)$

[3.2] se n é impar, $n = 2k + 1$, calcule: $b = a_n(x_n - x_1) + \cdots + a_{k+2}(x_{n+2} - x_k)$, em que o valor de x_{k+1}, que é a mediana, não entra no cálculo de b.

4. Calcule: $W = \dfrac{b^2}{S^2}$

5. Para tomada de decisão a respeito da normalidade dos dados, compara-se o valor calculado de W com o valor tabelado $W_{n,\alpha}$, obtido da Tabela **Shapiro_prob**. Se o valor calculado W for menor que o tabelado, rejeita-se a hipótese de normalidade ao nível α de significância.

Shapiro-Wilk no Python

Primeiro de tudo, vamos importar NumPy e Matplotlib .

importmatplotlib.pyplotasplt

```
import statsmodels.api as sm
from scipy.stats.stats import pearsonr, spearmanr
import itertools
from scipy.stats import shapiro
```

Agora temos que importar a função que calcula o valor p de um teste de Shapiro-Wilk. É a função "shapiro" no scipy.stats.

```
from scipy.stats import shapiro
```

Vamos agora simular dois conjuntos de dados, um gerado a partir de uma distribuição normal e outro gerado a partir de uma distribuição uniforme.

```
x = np.random.normal(size=300)
plt.hist(x, 10, rwidth=1.0)
plt.show()
shapiro(x)
ShapiroResult(statistic=0.9958346486091614, pvalue
   =0.6103981137275696)
```

```
y = np.random.uniform(size=300)
plt.hist(y, 10, rwidth=1.0)
plt.show()
shapiro(y)
ShapiroResult(statistic=0.9537638425827026, pvalue
   =3.945431359397844e-08)
```

2.4.4 Significância da Correlação

A correlação entre duas variáveis ocorreu mesmo de fato ou seria uma coincidência?

Em estatística, aborda-se a questão da significância de um resultado usando-se o conceito de hipótese nula.

O teste de Significância assume as seguintes hipóteses:
H_o : o resultado estatístico foi obtido apenas por acaso, devido a flutuações probabilísticas dos eventos sendo medidos e não devido a um efeito real que cause o resultado.
H_a : o resultado estatístico não foi obtido apenas por acaso, isto é, o valor do coeficiente de correlação é significante.

Se a hipótese nula estiver correta e o resultado obtido for devido ao acaso, qual a probabilidade de que ele ocorra?

Se a probabilidade da hipótese nula ocorrer é suficientemente baixa ($p_{value} \leq 0,05$), devemos rejeitar a hipótese nula e dizer que o resultado estatístico da correlação é significante, agora se a probabilidade da hipótese nula for ($p_{value} > 0,05$), não se pode rejeitar a hipótese nula, isto é, porém não se pode concluir que a correlação não é significante, somente dizer que tem uma probabilidade p_{value} de que a hipótese nula seja verdadeira, isto é, que o resultado pode ter ocorrido ao acaso.

Quando o tamanho da amostra é pequeno, é possível que obter um pareamento perfeito entre as duas variáveis apenas pelo acaso, então nestes casos precisamos ficar atentos aos valores da correlação, por outro lado, quando n for grande, mesmo valores baixos do coeficiente de correlação são difíceis de ser obtidos por mero acaso e, portanto, são considerados significantes.

2.4.4.1 Teste de Significância para o Coeficiente de Correlação de Pearson

Quando se coleta uma amostra de n pares de valores das variáveis (X, Y) e se calcula o seu coeficiente de correlação r, o que se quer saber é se esse valor de r é significante.

O teste de Significância para o Coeficiente r de Correlação de Pearson:
H_o : $\rho = 0$ não existe correlação para a população das variáveis X e Y (o que implicaria que o valor obtido para r ocorreu por mero acaso)
H_a : $\rho \neq 0$ existe correlação para a população das variáveis X e Y

O valor de r 2.23 é usado para estimar o coeficiente de correlação ρ 2.22 e o teste de significância desse valor consiste em assumir:
H_o : $\rho = 0$ (ausência de correlação)
H_a : $\rho \neq 0$ (correlação significante)

- Escolha o nível de significância α, geralmente $\alpha = 5\%$
- Consulte a tabela da distribuição t de Student para se obter o valor de $t(gl)$ para o valor de α escolhido e $gl = n - 2$;

- Calcule a variável $t_0 = r\sqrt{\frac{n-2}{1-r^2}}$;
- Se $t_0 > t(gl)$ ou $t_0 < -t(gl)$, rejeita-se a H_0, caso contrário não se rejeita H_0.

Se a probabilidade de obter o valor de r for menor que um certo valor crítico ($p_{value} \leq 0.05$) rejeita-se a H_0 e assume como mais provável a hipótese alternativa, segundo a qual $\rho \neq 0$.

Condição para o teste de significância do coeficiente de correlação de Pearson entre X e Y a distribuição de probabilidade conjunta para a população das variáveis X e Y é normal bidimensional.

Observação: quando se trabalha com amostras de n pares de valores (X_i, Y_i) onde $n \geq 30$, a condição de normalidade das duas variáveis é satisfeita.

2.4.4.2 Teste de Significância para o Coeficiente de Correlação de Spearman

O coeficiente de correlação de Spearman é um tipo de medida estatística não paramétrica, isto significa que não há restrições de normalidade para o teste do coeficiente de correlação de Spearman.

O valor de r_s 2.24 é usado para estimar o coeficiente de correlação ρ 2.22 e o teste de significância desse valor consiste em assumir:
$H_o : \rho = 0$ (ausência de correlação)
$H_a : \rho \neq 0$ (correlação significante)

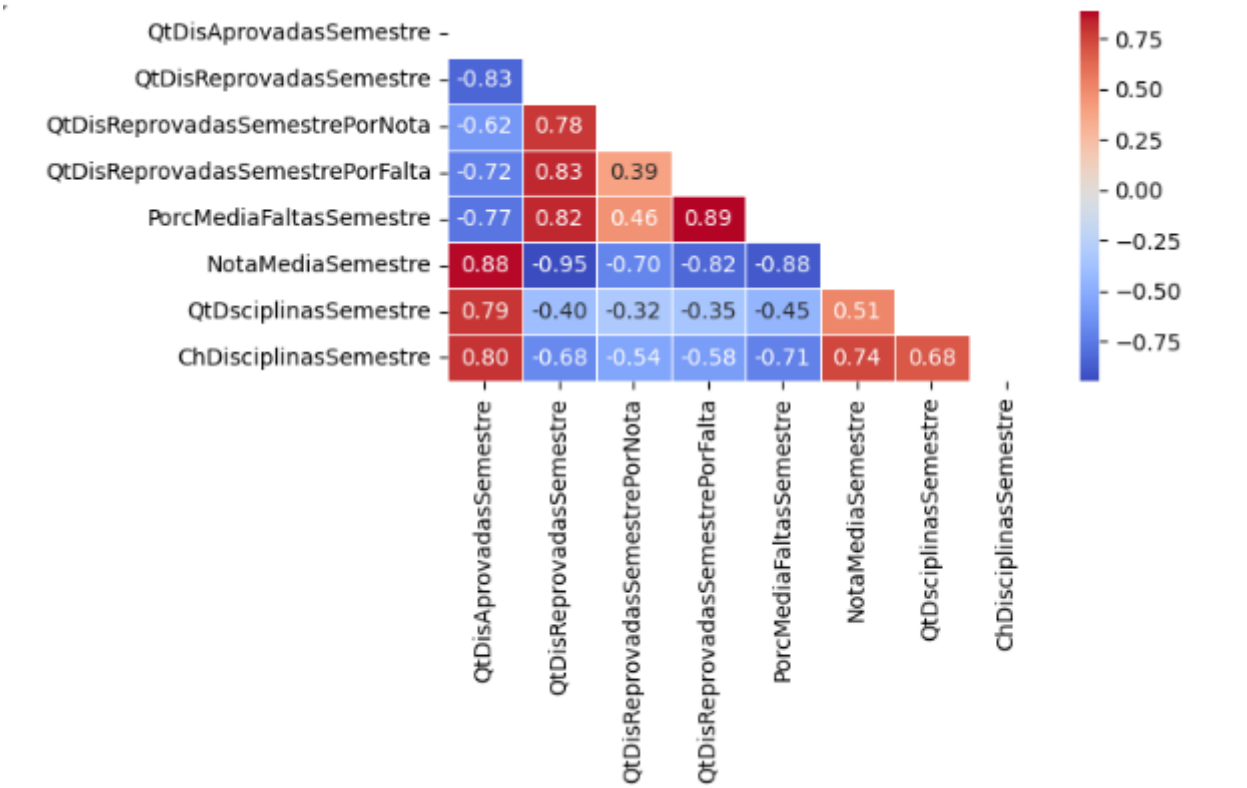
- Escolha o nível de significância α, geralmente $\alpha = 5\%$
- Consulte a tabela da distribuição t de Student para se obter o valor de $t(gl)$ para o valor de α escolhido e $gl = n - 2$;
- Calcule a variável $t_0 = r_s\sqrt{\frac{n-2}{1-(r_s)^2}}$;
- Se $t_0 > t(gl)$ ou $t_0 < -t(gl)$, rejeita-se a H_0, caso contrário não se rejeita H_0.

2.5 MATRIZ DE CORRELAÇÃO

A correlação bivariada é um bom começo para análise de correlação, porém as vezes precisamos analisar mais do que duas variáveis. Uma correlação de muitas variáveis pode ser retratada por uma matriz de correlação. Uma matriz de correlação é uma matriz que representa a correlação de cada par de variável. Uma matriz de correlação exibe o grau de correlação entre variáveis. Cada célula a_{ij} da matriz representa a correlação entre a variável X_i com a variável X_j. A cor da célula indica o grau de correlação entre essas duas medidas.

```
corr_matrix = df.corr("spearman")
f, ax = plt.subplots(figsize=(16,8))
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))
sns.heatmap(corr_matrix,mask=mask, annot=True, fmt='.2f', linewidth
   =0.4,
            annot_kws={"size": 10}, cmap='coolwarm', ax=ax)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.show()
```

Figura 27 – Matriz de Correlação de Spearman



Fonte: Produzido pelo autor

O coeficiente de correlação de Spearman com os valors do p_{value}

```
correlations = {}
columns = df.columns.tolist()
for col_a, col_b in itertools.combinations(columns, 2):
    correlations[col_a + '__' 'TARGET'] = spearmanr(df.loc[:, col_a
       ], df.loc[:,'TARGET'])
result = df.from_dict(correlations, orient='index')
result.columns = ['PCC', 'p-value']
print(result.sort_index())
```

```
                                                PCC        p-value
ChDisciplinasSemestre__TARGET              0.646349   7.187900e-19
NotaMediaSemestre__TARGET                  0.756993   9.039033e-29
PorcMediaFaltasSemestre__TARGET           -0.705720   1.307028e-23
QtDisAprovadasSemestre__TARGET             0.701615   3.030727e-23
QtDisReprovadasSemestrePorFalta__TARGET   -0.621230   3.641782e-17
QtDisReprovadasSemestrePorNota__TARGET    -0.547133   6.246907e-13
QtDisReprovadasSemestre__TARGET           -0.704177   1.795978e-23
QtDsciplinasSemestre__TARGET               0.480785   6.224142e-10
```

2.6 PRÉ-PROCESSAMENTO DOS DADOS

Atualmente existem duas técnicas para dimensionamento dos dados, sendo a primeira a normalização e a segunda a padronização. A diferença entre ambas consiste que padronizar as variáveis irá resultar em uma média igual a 0 e um desvio padrão igual a 1. Já normalizar tem como objetivo colocar as variáveis dentro do intervalo de 0 e 1, caso tenha resultado negativo -1 e 1.

2.6.1 **Reescala dos dados**

Quando observarmos que os valores das variáveis atributos estão em uma escala bem distinta, isto pode causar um grande problema no treino do nosso modelo uma vez o atributo **ChDisciplinasSemestre** possui uma escala muito maior que o atributo **QtDisAprovadasSemestre** que terá uma influência consequentemente muito maior no resultado. Além disso podemos ter alguns possíveis outliers os quais queremos minimizar o impacto em nossa solução. Para solucionar esses problemas poderíamos usar diversas técnicas de estatística e algumas métricas como quartis, desvio padrão e variância ou podemos ser muito mais espertos e já usar inúmeras ferramentas do **sklearn** que já aplicam todas essas técnicas a fim de obter uma melhor solução para o nosso problema através da reescala dos dados.

StandardScaler: Esse método age sobre as colunas dos dados que estamos pré-processando.A documentação oficial nos traz que o StandardScaler "'padroniza as features removendo a média e a escala a uma variância a uma unidade"'. Em outras palavras, isso significa que para cada feature, a média seria 0, e o Desvio Padrão seria 1. Em resumo, o método subtrai do valor em questão a média da coluna e divide o resultado pelo desvio padrão:

$$\frac{x_i - \mu(x)}{\sigma}$$

Esse método trabalha melhor em dados com distribuição normal porém vale a tentativa para outros tipos de distribuições, além disso podemos deixar como dica que esse método resulta em ótimos frutos quando usado em conjunto com algoritmos como Linear Regression e Logistic Regression.

MinMaxScaler: Assim como o StandardScaler, esse método também atua sobre as colunas dos dados que estamos pré-processando. A documentação oficial nos traz que o MinMaxScaler "dimensiona o conjunto de dados de modo que todos os valores de recursos estejam no intervalo [0, 1] ou no intervalo [-1,1] se houver valores negativos nos dados. Esta escala também comprime todos os inliers na faixa [0,0.005]".

De forma simples, o MinMaxScaler subtrai o valor em questão pelo menor valor da coluna e então divide pela diferença entre o valor máximo e mínimo:

$$\frac{x_i - min(x)}{max(x) - min(x)}$$

RobustScaler: Ao contrário do que vimos até o momento, essa reescala não é baseada em média ou desvio padrão, mas sim em percentis. Devido a isso o diferencial desse método é que isso nos garante um bom tratamento dos outliers. Porém, é importante notar que os outliers ainda estarão presentes nos dados reescalados, mas o seu impacto negativo será reduzido.

Em seu método, o RobustScaler subtrai a média do valor em questão e então divide o resultado pelo segundo quartil:

$$\frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)}$$

QuantileTransformer: Assim como o RobustScaler atua sobre as colunas e também trata os outliers com uso de quartis. Este método aplica uma transformação não linear de modo que a função de densidade de probabilidade de cada característica será mapeada para uma distribuição uniforme ou gaussiana. Neste caso, todos os dados, incluindo outliers, serão mapeados para uma distribuição uniforme com o intervalo [0, 1], tornando outliers indistinguíveis de inliers.

PowerTransformer: Atua sobre as colunas e assim como o Quantile procura transformar os valores em uma distribuição mais normal, sendo indicado em situações onde uma

distribuição normal é desejada para os dados, além disso esse método ainda suporta os métodos de transformação

Box-Cox (dataset com dados positivos) e Yeo-Johnson (dataset com dados positivos e negativos).

$$\begin{cases} \dfrac{(x_i+1)^\lambda - 1}{\lambda} & se \quad \lambda \neq 0,\ x_i \geq 0 \\ ln(x_i+1) & se \quad \lambda = 0,\ x_i \geq 0 \\ \dfrac{(-x_i+1)^\lambda - 1}{\lambda} & se \quad \lambda \neq 0,\ x_i < 0 \\ ln(-x_i+1) & se \quad \lambda = 0,\ x_i < 0 \end{cases}$$

MaxAbsScaler: Use-o quando quiser dimensionar dados esparsos. Escala cada recurso pelo seu valor absoluto máximo. Semelhante ao MinMaxScaler quando valores positivos estão presentes. Sensível a outliers.

$$\frac{x_i}{max(x)}$$

Podemos utilizar cada transformação nas seguintes situações:

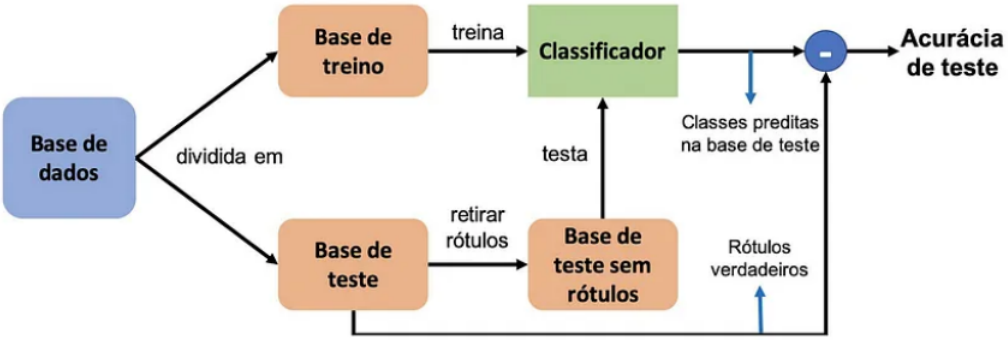
- **MinMaxScaler:** Dados não seguem uma distribuição normal e se o desvio padrão for pequeno;
- **StandardScaler:** Dados seguem uma distribuição normal ou quando se deseja uma distribuição mais próxima da normal e é uma boa combinação com algoritmos como Linear Regression e Logistic Regression;
- **RobustScaler:** Quando deseja reduzir os impactos dos outiliers.
- **QuantileTransformer:** Quando deseja reduzir os impactos dos outiliers e trata os outiliers mais agressivo que o RobustScaler; e
- **PowerTransformer:** Quando é necessário transformar os dados em uma distribuição mais próximo de uma normal; e
- **MaxAbsScaler:** Semelhante ao MinMaxScaler quando valores positivos estão presentes.

Observação sobre reescala: Não existe uma receita de bolo clara para a escolha da melhor forma de se reescalar os dados, portanto a melhor forma é comparar os resultados.

2.7 DIVISÃO DOS DADOS

Avaliar uma série de métricas para decidirmos se um modelo de aprendizado de máquina tem uma performance satisfatória ou não, precisamos calcular estas métricas com base em testes realizados em um conjunto de dados diferente daquele que foi utilizado para o treinamento. Portanto o modelo não pode "ver"os dados de teste durante o treinamento; isto é, a fazer uma prova sabendo de antemão quais são as questões e quais são as respostas. O propósito do teste é avaliar como o modelo se comporta com dados nunca vistos antes, simulando uma situação de uso real.

Figura 28 – Dados não normalizados



Fonte: Medium

Quando estamos dividindo um dataset em treino e teste, precisamos lembrar que estamos tirando informações importantes que o modelo poderia se beneficiar. Por isso não queremos colocar muita informação no set de teste. Entretanto quanto menor o set de treino, mais impreciso será o resultado. Dividir um dataset em treino e teste é tudo uma questão de equilibrar essa troca. Na pratica as divisões mais comuns são de 60:40, 70:30, or 80:20, dependendo do tamanho inicial do dataset. Entretanto datasets com grandes quantidades de dados podem ser divididos em 90:10 ou até 99:1, se um dataset contém mais de 100.000 exemplos, será tranquilo tirarmos apenas 10.000 dele, para que possamos testar e ter uma melhor estimativa.

2.7.1 **Hold-out**

O método mais simples de testar o modelo consiste em separar, de forma aleatória, uma parcela dos dados para testar o modelo, e utilizar o restante para treinamento. Ou seja, os testes são feitos com dados que o modelo não viu anteriormente. Isso é conhecido como hold-out.

A imagem abaixo ilustra o método **hold out**, em que o conjunto de dados é dividido em 70% de segmentos de treinamento e 30% de teste.

Figura 29 – Dados não normalizados

Fonte: Fonte (SILVA, 2023)

Porém, o hold-out tem alguns problemas que se tornam mais ou menos perceptíveis de acordo com o tamanho e a qualidade do dataset:

- Observações muito importantes do dataset podem acabar sendo separadas para teste, prejudicando o treinamento do modelo e consequentemente piorando o resultado dos testes;
- As métricas podem depender da forma como os dados foram separados para treino e teste;
- Caso o dataset seja pequeno, dividir os dados entre treino e teste se torna inviável. Normalmente usamos datasets com centenas ou milhares de registros para treinar um modelo. Tudo depende do problema que está sendo resolvido;

A seguir serão apresentados algumas formas de separação de dados:

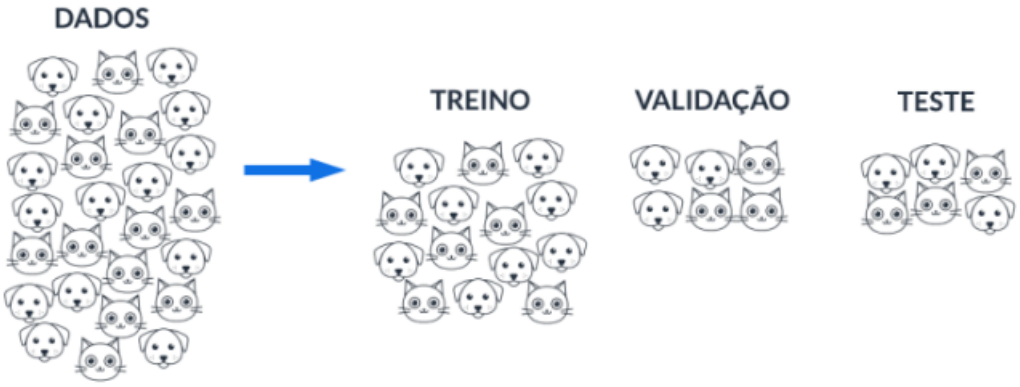
2.7.2 Aleatória simples

Nessa situação, de grande volume de dados, a separação de amostras deve ser por amostragem aleatória simples. Isso não significa colocar as primeiras X linhas na amostra de treino, as próximas Y linhas na amostra de validação e as restantes Z linhas na amostra de teste. Se os dados estiverem ordenados, a chance de uma determinada amostra ser povoada com dados de uma única categoria pode ser alta.

2.7.3 Aleatória Estratificada

Nessa situação, de pouco volume de dados, a separação de amostra pode ser categoria-por-categoria. Isso significa separar aleatoriamente da categoria predominante, (i.e., gato), X linhas para amostra de treino, Y linhas para amostra de validação e Z linhas para

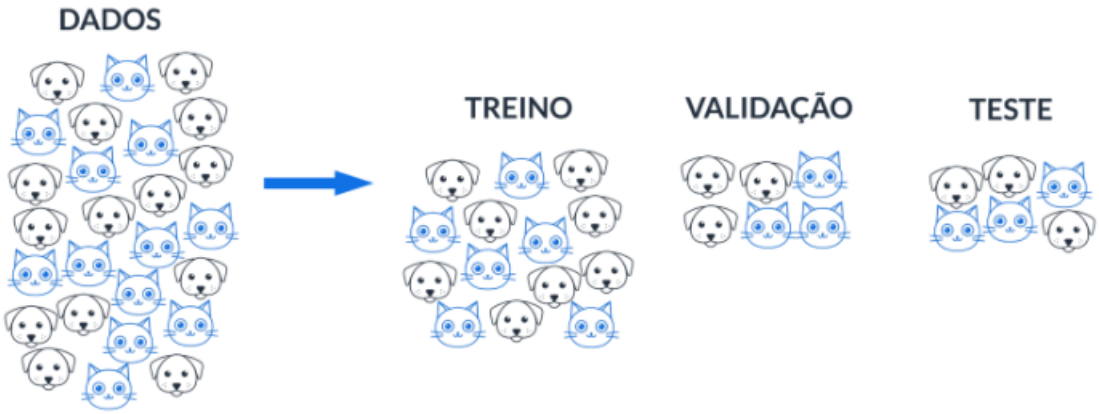
Figura 30 – Aleatório Simples



Fonte: (SILVA, 2023)

H]

Figura 31 – Amostra Estratificada



Fonte: (SILVA, 2023)

amostra de teste. E fazer o mesmo procedimento para as demais categorias. Isso se chama amostragem aleatória estratificada.

2.7.4 Amostras por períodos no tempo.

Além da divisão em conjuntos de treinamento e teste, é importante também considerar a aleatoriedade na seleção dos dados. A aleatoriedade ajuda a garantir que o modelo seja treinado e testado em diferentes amostras, o que contribui para uma avaliação mais robusta do desempenho do modelo.

Figura 32 – Amostras por períodos no tempo

Fonte: (SILVA, 2023)

2.7.5 Balanceamento dos Dados

Conjunto de dados com mais de 50% das entradas pertencendo a uma única classe são considerados desbalanceados. A maioria dos algoritmos de aprendizado de máquina tem melhor desempenho com conjuntos de dados equilibrados, pois buscam otimizar a precisão geral da classificação ou medidas relacionadas. Em casos de dados desbalanceados, os limites de decisão estabelecidos pelos algoritmos tendem a favorecer a classe majoritária, levando a uma classificação incorreta da classe minoritária.

Para resolver esse problema, é necessário utilizar métricas que levem em consideração o desbalanceamento e aplicar técnicas para tratar essa questão. As técnicas de amostragem, como "Undersampling"e "Oversampling", são métodos comuns para lidar com o desequilíbrio de classes.

De modo geral, o "under-sampling"consiste em remover exemplos da classe majoritária para tornar a proporção entre as classes mais equilibrada. Já o "over-sampling"ou "SMOTE (Synthetic Minority Oversampling Technique)", por sua vez, consiste em gerar novos exemplos para a classe minoritária, de forma a aumentar sua representatividade no conjunto de dados.

2.7.6 Validação Cruzada: K-Fold

Enquanto o K-Fold Cross-Validation divide o conjunto de dados em vários subconjuntos para treinar e testar iterativamente o modelo, o método Train-Test Split divide o conjunto de dados em apenas duas partes: uma para treinamento e outra para teste. O método Train-Test Split é simples e rápido de implementar, mas a estimativa de desempenho pode ser altamente dependente da divisão específica, o que leva a uma alta variação nos resultados.

- K — Significa o número de subdivisões (iguais) que nós fizemos;
- Fold — Significa cada um dos blocos de cada K.

A imagem mostra uma validação cruzada 5 vezes, em que o conjunto de dados é dividido em cinco partes, com cada parte servindo como um conjunto de teste em uma das cinco iterações, garantindo que cada segmento seja usado tanto para treinamento quanto para teste.

Figura 33 – Dados não normalizados

Fonte: Medium (JUNIOR, 2019)

O sklearn fornece algumas funções de validação cruzada, como por exemplo a $cross\_val\_score$

A função $cross\_val\_score$ retorna uma lista com o resultado de cada um dos testes, utilizando a métrica padrão do modelo: que geralmente é a acurácia, para modelos de classificação, e R^2 para os modelos de regressão.

Para obter o resultado final, costumamos calcular a média aritmética entre os resultados individuais de cada teste. Podemos também calcular a consistência do modelo entre as iterações para avaliar a variância. Quanto menor o desvio padrão, maior é a consistência do modelo (= baixa variância).

A validação cruzada k-fold é uma boa maneira de avaliar como o modelo se comporta diante de variações nos samples de treinamento, e ajuda a evitar um dos problemas principais do hold-out: influência da divisão dos dados na métrica. Testes empíricos mostram que valores bastante confiáveis para K são 5 e 10, pois estes valores não causam erros de teste altos nem de bias e nem de variância.

3 REVISÃO DE LITERATURA

3.1 REVISÃO DA LITERATURA SOBRE O PROBLEMA DE EVASÃO

Neste capítulo, apresentamos o conhecimento sobre a literatura, no caso trabalhos de dissertação, teses e artigos sobre o assunto de evasão. Muitos estudos sobre a previsão de evasão escolar já foram publicados, foram encontrados vários estudos que utilizam métodos de classificação e redes neurais, e alguns que utilizam métodos de séries temporais. No contexto do problema de evasão e retenção escolar, é possível verificar algumas abordagens propostas na literatura para análise do problema. Aqui vamos citar alguns trabalhos.

Mahzoon *et al.* em 2018, apresentou um modelo baseado em relações temporais de dados heterogêneos como base para a construção de modelos preditivos. Neste trabalho foi mostrado como os padrões temporais dentro e entre semestres podem fornecer informações sobre a experiência do aluno. Por exemplo, em um modelo semestral, a previsão da nota final do curso pode ser baseada em atividades semanais e envios registrados no sistema de gerenciamento de aprendizagem (Learning Management System - LMS). No modelo entre semestres, a previsão de sucesso ou fracasso em um programa de graduação pode ser baseada em padrões de sequência de notas e atividades em vários semestres. Os benefícios do modelo de dados sequenciais incluem estrutura temporal, segmentação, contextualização e narrativa. Para demonstrar esses benefícios, coletaram e analisaram dados de 10 anos de alunos em um modelo de sequência entre semestres e usaram dados em um curso introdutório à ciência da computação para construir um modelo de sequência dentro do semestre. Resultados para os dois modelos de sequência mostraram que a análise baseada no modelo de dados de sequência pode alcançar maior precisão preditiva do que modelos não temporais com os mesmos dados.

Melo em 2019, identificou em sua pesquisa um padrão de informações a respeito dos alunos que pode ser relacionado à possibilidade de evasão do ensino superior. Com este estudo, ele criou uma ferramenta automatizada, baseada na técnica de mineração de dados utilizando-se redes neurais artificiais, capazes de identificar os alunos com tendência à evasão. O modelo criado foi capaz de identificar $63,8\%$ dos alunos que evadiram. Dentre todos os alunos analisados durante a validação do modelo, o modelo foi capaz de identificar corretamente a situação de $70,5\%$ dos alunos (entre evadidos e não evadidos). Também foi possível definir que o modelo consegue identificar os alunos com tendência à evasão com 36 dias de antecedência da ocorrência da evasão, na média. Foi proposto um mecanismo de geração de alertas, mediante a produção de um relatório que pode ser disponibilizado

diretamente na ferramenta de gestão acadêmica (SISCAD). Dessa forma, os agentes da Instituição, como os coordenadores de curso e os servidores da Pró-Reitoria de Assuntos Comunitários e Estudantis (PROACE) da UFTM, poderão atuar junto aos alunos com essa tendência de evasão e tentar revertê-la.

Askinadze & Conrad em 2019, trataram sobre a evasão dos alunos do curso de ciências da computação de uma universidade da Alemanha, visto que a maioria dos desistentes abandonam o curso nos quatro primeiros semestres, a pesquisa busca identificar esses alunos mais cedo possível para poderem iniciar uma intervenção. Para fazer as previsões foram analisados os exames semestrais dos alunos, não apenas o exame do último semestre, mas todos os anteriores de cada aluno. Com essa análise de dados buscaram a possível criação de um modelo de previsão de evasão. O autor mostrou que distâncias de uma série temporal podem ser utilizadas no núcleo de um modelo SVM (Support Vector Machine) para a previsão de desistência.

Santos, Martins & Plastino em 2021, aplicou técnicas de Mineração de Dados, mais especificamente, técnicas de classificação, para prever e tentar evitar a evasão. Os modelos preditivos são gerados apenas com base no desempenho dos estudantes nas disciplinas cursadas. São criados n diferentes modelos, dos quais o i-ésimo modelo,$1 \leq i \leq n$, é capaz de prever, ao fim do i-ésimo semestre de um estudante, se ele ou ela irá evadir os se formar no futuro. Os experimentos realizados com uma base de dados real, sobre estudantes da Universidade Federal Fluminense, mostraram que os modelos são capazes de atingir acurácia preditiva entre $79,31\%$ *e* $98,25\%$.

Souza *et al.* em 2021, fizeram uma revisão de literatura sobre o tema evasão na Educação Superior, levantamento de dados no sistema acadêmico da instituição e nos setores que registram a vida escolar dos estudantes e questionários. O estudo concluiu que o fenômeno da evasão é complexo, multifacetado, social e historicamente referendado, pois envolve desde questões relacionadas às condições objetivas de permanência na universidade, como questões financeiras, tempo, saúde, problemas familiares, até condições subjetivas como dificuldades de aprendizagem, dificuldades de adaptação à proposta pedagógica e à cultura acadêmica, falta de identificação com o curso e dúvidas quanto à inserção no mundo do trabalho após a formação. O estudo propôs o fortalecimento de ações institucionais que impactem na permanência e no sucesso acadêmico, além da implementação de um projeto de tutoria. Também indicou a necessidade de definir uma política institucional de combate à evasão com responsabilidades dos diferentes setores e profissionais da universidade, recursos e formas de acompanhamento articulado.

Filho, Vinuto & Leal em 2021, publicaram um artigo sobre técnicas de construção de modelos de séries temporais para predição de taxas periódicas de evasão dos alunos do Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE). Os autores dividiram o processo de análise de dados em quatro etapas. Na etapa 1 utilizaram a plataforma do IFCE para coletar os dados e construir as séries temporais, referentes a quantidade de

alunos nos períodos semestrais de 2009.1 a 2017.2 de 9 Campis do instituto, com o intuito de prevenir e melhorar as tomadas de decisão para combater a evasão . Na etapa 2 foi feita a análise exploratória das séries temporais, foram aplicados técnicas de predição, análise de gráficos e foram feitas as validações dos modelos, para esse tratamento de dados foi utilizado a linguagem R. Na etapa 3 foram selecionados os modelos ARIMA, Regressão Linear (TSLM) e redes neurais (NNAR), no qual foi analisado que o modelo ARIMA apresenta melhores resultados. Por fim na Etapa 4 são analisados os resultados, foram utilizados as medidas de desempenho, raiz quadrada do erro médio quadrático (RMSE), e o erro médio percentual absoluto (MAPE).

Souza em 2021, apresentou um estudo sobre séries temporais, relacionadas a índices de evasão e retenção escolar no ensino médio profissional, visando a identificação das características peculiares a estas séries e, baseado neste estudo, propor uma abordagem baseada em redes neurais artificiais, do tipo multicamadas, para prever este tipo particular de série temporal. Para o processo de aprendizagem, foi utilizado o algoritmo de retropropagação do erro (back propagation). Uma análise experimental é conduzida com a abordagem proposta utilizando séries temporais relacionadas aos índices de evasão e retenção do Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE), no período de 2009 a 2018. Nestes experimentos, são utilizadas as medidas erro médio quadrático (mean squared error, MSE) e erro médio absoluto percentual (mean absolute percentage error, MAPE) para avaliar o desempenho preditivo e os testes de Friedman e Tukey para validá-lo estatisticamente. Os resultados alcançados indicam que a abordagem proposta é capaz de prever eficientemente estas séries no período avaliado, sendo opções viáveis para previsão de índices de evasão e retenção escolar em instituições de ensino médio profissional.

Vieira em 2021, publicou um trabalho sobre predição de evasão acadêmica, realizado na Universidade Federal de Goiás (UFG), entre os anos de 2009 a 2020. Nesse trabalho, no processo de coleta de dados, as informações referentes aos alunos foram armazenadas em um *datalake* no formato CSV, essas informações eram organizadas em dois conjuntos tabulares associados ao ingresso dos alunos e ao desemprenho dos mesmos. O primeiro conjunto citado eram formados por informações de aspectos gerais, demográficos e relacionados a trajetória dos alunos. Já o outro conjunto era formado por informações relacionadas às ocorrências de matrícula nas disciplinas. Esse processo de coleta de dados foi importante para poder identificar inconsistências nos dados, que puderam ser corrigidos na fase de preparação, alguns dos problemas encontrados foram: grande quantidade de atributos com informações faltantes; atributos e registros duplicados; inconsistência na formatação de valores decimais; registros em branco; e campos multivalorados.

Rodrigues em 2022, investigaram modelos de classificação preditivos que auxiliam na identificação de alunos que são propensos à evadir do curso. Foram obtidos dados dos alunos de Engenharia de Computação e Sistemas de Informação da UFOP, dados os quais foram tratados, analisados e caracterizados a fim de revelar o comportamento dos

alunos e identificar as principais variáveis que impactam a evasão. Durante o processo de caracterização dos dados, foram realizadas análises de correlação, de componentes principais e de regressão linear múltipla para definir as variáveis que melhor se ajustavam para treinamento e teste dos modelos. Em seguida, foram construídos e validados modelos preditivos por meio de técnicas de aprendizado de máquina como Regressão Logística, XGBoost, SVM, Árvores de Decisão e Random Forest. Por fim, foi realizada a etapa de análise preditiva, onde foram aplicados experimentos com diferentes configurações de treino e teste. Os modelos alcançaram resultados satisfatórios, alcançando mais de 95% de acurácia na predição da evasão de discentes em determinados cenários. Dessa forma, o trabalho permite melhor entendimento acerca dos fatores que impactam a evasão no ICEA para os cursos de computação. Além disso, abrem-se oportunidades de investigação da evasão em cursos de outras áreas de conhecimento ou em modelos generalizados, o que pode permitir uma comparação e avaliação de qual nível de generalização é oportuno para o ajuste dos modelos.

Tavenard em 2022, o buscaram um alinhamento temporal que minimiza a distância euclidiana entre as séries alinhadas. Em outras palavras podemos dizer que o DTW é equivalente a minimizar a distância euclidiana entre as séries temporais alinhadas em todos os alinhamentos temporais admissíveis. Analisando as sequências e a série temporal o autor analisou quatro modelos de previsão diferentes, utilizando o classificador de séries temporais SVM, ele analisou três abordagens, utilizando o Kernel GDTW (*Gaussian Dynamic Time Warping)*, o Kernel WSD (Word Sense Disambiguation), e utilizando um Kernel de redes neurais RBF (*Radial-Basis Function*). Além desses também foi utilizado um modelo RF (Random Forest).

Barbosa em 2022, utilizou algoritmos de regressão logística, árvore de decisão, k-Nearest Neighbours, Support Vector Machine e random forest. A regressão logística, a árvore de decisão e o random forest permitiram identificar que as variáveis mais significativas foram categoria de cota, forma de ingresso, setor de estudo, e considerando o **primeiro semestre**: índice de rendimento , carga curricular, número de reprovações por frequência e notas. Os melhores resultados de predição foram obtidos pelos algoritmos random forest com AUC de $0,863$ e acurácia de $0,734$ e o SVM com AUC de $0,847$ e acurácia de $0,741$.

Rodrigues em 2022, investigou modelos de classificação preditivos que auxiliam na identificação de alunos que são propensos à evadir do curso. Foram obtidos dados dos alunos de Engenharia de Computação e Sistemas de Informação da UFOP, dados os quais foram tratados, analisados e caracterizados a fim de revelar o comportamento dos alunos e identificar as principais variáveis que impactam a evasão. Durante o processo de caracterização dos dados, foram realizadas análises de correlação, de componentes principais e de regressão linear múltipla para definir as variáveis que melhor se ajustavam para treinamento e teste dos modelos. Em seguida, foram construídos e validados modelos preditivos por meio de técnicas de aprendizado de máquina como Regressão Logística,

XGBoost, SVM, Árvores de Decisão e Random Forest. Por fim, foi realizada a etapa de análise preditiva, onde foram aplicados experimentos com diferentes configurações de treino e teste. Os modelos alcançaram resultados satisfatórios, alcançando mais de 95% de acurácia na predição da evasão de discentes em determinados cenários.

Viana, Santana & Rabêlo em 2022 estou evasão utilizando modelos treinados para cada janela semestral. Os experimentos foram realizados com dados dos cursos de Computação e Sistemas de Informação da Universidade Federal do Piauí (UFPI). Neste trabalho realizou o processo de coleta, pré-processamento, transformação, classificação e validação em dados de discentes do curso de Computação de uma instituição de ensino federal. A arquitetura separa os dados por período, do 1° ao 6° período, com a proposta de criar um modelo treinado para cada. Cada modelo é construído com dados de ingresso fixos e dados acadêmicos cumulativos e variáveis. Ou seja, os dados acadêmicos mudam para cada modelo, pois são dados cumulativos até aquele momento do curso. Importante ressaltar que o modelo referente ao sexto período é utilizado como modelo para prever todos os discentes que concluam do sexto período em diante. Os resultados forão bem promissores, com valores de acurácia superiores a 86% na validação cruzada. Esse valor sobe a cada período classificado, com valor ótimo no quinto período na maioria dos algoritmos, chegando à $95,7\%$. Os algoritmos possuem comportamentos diferentes para cada período, podendo considerar algoritmos específicos para cada. Enquanto o RF possui melhor acurácia quando gerado o modelo para o 1° e 2° período, o melhor algoritmo para o 4° e 6° período é o ET e MLP. Nesses casos, pode-se gerar uma arquitetura onde para cada período seja atribuído um modelo específico que possua maior taxa de acurácia.

Martins *et al.* em 2023 os autores propuseram modelos de previsão de evasão, especialmente a evasão tardia. Um banco de dados de 30.000 alunos foi usado para treinar algoritmos de aprendizado de máquina supervisionados. Seus desempenhos são então discutidos em termos de acurácia e f1-score.

Oliveira & Medeiros em 2024 avaliou um modelo preditivo para identificar alunos propensos à evasão, utilizando dados de um modelo semestral de autoavaliação dos cursos de graduação da Universidade Federal da Paraíba (UFPB). Utilizando a mineração de dados educacionais e a metodologia CRISP-EDM, o estudo analisou a relação entre evasão escolar e autoavaliação institucional, seguido de análise exploratória e preparação dos dados para classificação. Diversas técnicas de modelagem, como Árvore de Decisão, Floresta Aleatória e Máquinas de Vetores de Suporte, foram aplicadas, sendo os modelos avaliados por métricas de desempenho, revelando uma acurácia de $87,97\%$, precisão de $91,72\%$, recall de $91,67\%$ e f-score de $91,57\%$ na identificação de alunos com alta probabilidade de evasão.

3.2 REVISÃO DA LITERATURA SOBRE A INTELIGÊNCIA ARTIFICIAL E GEOGEBRA NO ENSINO MÉDIO

Já pensando na Educação básica mais especificamente no Ensino Médio, temos segundo a Base Nacional Comum Curricular (BNCC), que os alunos ao chegarem aos anos finais, sejam estimulados a desenvolver o pensamento computacional, por meio da interpretação e da elaboração de algoritmos, incluindo também aqueles que podem ser representados por fluxogramas. Ainda de acordo com a BNCC no Ensino Médio o foco deve ser a construção de uma visão integrada da Matemática, aplicada à realidade, em diferentes contextos. Consequentemente, quando a realidade é a referência, é preciso levar em conta as vivências do dia a dia dos estudantes, impactados de diversas maneiras pelos avanços tecnológicos, pelas exigências do mercado de trabalho, pela potencialidade das mídias sociais, entre outros. Nesse contexto, destaca-se ainda a importância do recurso a tecnologias digitais e aplicativos tanto para a investigação matemática como para dar continuidade ao desenvolvimento do pensamento computacional, iniciado na etapa anterior. Além disso as Diretrizes Curriculares Nacionais para o Ensino Médio (DCNEM) cita no artigo 12 que nos itinerário formativos da área de Matemática e sua tecnologias, de acordo com a oferta de ensino, pode ser feito um aprofundamento de conhecimentos estruturantes para aplicação de diferentes conceitos matemáticos na área de inteligência artificial, programação, robótica, entre outros.

Alguns trabalhos relacionados a esse último tema, utilizam métodos de predição no Ensino médio como ferramenta para o desenvolvimento do pensamento computacional do aluno.

Em 2024 Brandão publicou um trabalho, no qual é apresentado aos alunos do Ensino Médio o algoritmo Support Vector Machine (SVM). Inicialmente foram propostos atividades teóricas aos alunos, a primeira atividade proposta foi sobre o problema da flor de íris, o objetivo nessa atividade era apresentar aos alunos uma forma de calcular a equação da reta que representa a fronteira de decisão do SVM. para a realização dessa atividade é necessário a apresentação de algumas informações para que os alunos possam compreender a atividade de acordo com os conteúdos já aprendidos por eles. A segunda atividade apresentada foi o truque do kernel, teve como objetivo encontrar a função de decisão do SVM por meio do truque do kernel, para a realização dessas duas atividades os alunos precisam ter conhecimento prévio sobre operações básicas, multiplicação de matrizes e produto escalar. O autor também elaborou uma atividade com o intuito de trazer a teoria para a pratica computacional, nessa atividade o objetivo é criar um modelo que seja capaz de predizer o número escondido na imagem com acurácia aceitável. Essa atividade foi divida em seis etapas, no qual mediado pelo professor, os alunos desenvolvem o modelo de predição.

Em 2004 Kaleff & Nascimento neste artigo apresentam-se atividades apoiadas nos

Parâmetros Curriculares Nacionais e no Modelo de van Hiele para o desenvolvimento do pensamento geométrico. Desenvolveram-se atividades que buscam levar o aluno, desde os primeiros ciclos do Ensino Fundamental, a construir conceitos que o possibilitarão a se orientar no espaço e a transpor algumas dificuldades que têm surgido em situações de aprendizagem introdutórias ao ensino das Geometrias não-Euclidianas. Apresentam-se atividades de um tipo especial de Geometria: do Táxi ou Pombalina. Criam-se situações pedagógicas que permitem apresentar a negação de um axioma euclidiano, as quais envolvem jogos, tabelas, maquetes, mapas e esquemas gráficos. PALAVRAS-CHAVE / Atividades introdutórias; Geometrias não-Euclidianas; Geometria do táxi; materiais concretos.

Em 2023 Luz o autor utilizou o Google Earth e o GeoGebra no ensino da Geometria do Táxi e da Geometria Euclidiana que teve como público alvo os alunos da 3ª série do ensino médio da rede pública de Teresina-PI. Na sua execução foi associada as teorias matemáticas da geometria e as funcionalidades dos softwares Google Earth e GeoGebra. O objetivo foi criar situações problemas em sala de aula e visualizar a sua resolução nos softwares. A introdução traz um breve contexto histórico dos postulados de Euclides e da tentativa de matemáticos como Gauss, Bolyai, Lobachevsky, Reimann e Hermann Minkowski que esfossaram-se para demostrar o V postulado a partir dos quatro primeiros porém não obtiveram sucesso. Contudo, estes estudos serviram de base para a criação das geometrias não euclidianas como a geometria hiperbólica, esférica e do táxi. Em seguida, são vistas as noções básicas de métrica e espaços métricos, métrica da geometria euclidiana e da geometria do táxi, a relação entre o cálculo das distâncias entre dois pontos, a construção das circunferências e a determinação do π nestas geometrias. A pesquisa de campo foi iniciada com a avaliação diagnóstica, esta buscou verificar o conhecimento prévio que os alunos possuíam sobre o tema. Em seguida foi realizada a análise das suas respostas, elemento fundamental para a construção dos planos de aula e a estrutura da execução das atividades. Os alunos utilizaram os software Google Earth e GeoGebra para comprovar e verificar todas as construções realizadas em sala de aula.

Em 2017 Leivas, Portella & Souza os autores apresentaram um recorte de uma pesquisa orientada pelo primeiro autor, envolvendo dois professores da rede pública de ensino no Brasil, sendo que o segundo autor trabalhou com três estudantes do Ensino Fundamental na aquisição de conhecimentos básicos sobre Geometria Hiperbólica, especificamente utilizando o GeoGebra na construção do modelo de Poincaré. A terceira autora investigou, juntamente com três estudantes do Ensino Médio, conhecimentos sobre Geometria Elíptica, mais designadamente aqui relatando a investigação na relação entre ângulo central e ângulo inscrito na circunferência. O objetivo do projeto foi investigar possibilidades didáticas de introduzir conteúdos dessas geometrias, com o uso de software, na escola básica brasileira. Os resultados da pesquisa mostraram ser possível desenvolver com os dois grupos de estudantes conteúdos que, na maioria das vezes, nem chegam a ser de conhecimento do professor de Matemática em formação.

Em 2020 Petrics; Petrics o autor publicou três atividades no site do geogebra.org, um era sobre KNN utilizando distância euclidiana, redes neurais artificiais simples e árvore de decisão e em 2023 Pantaloni o autor publicou duas atividades no site do geogebra.org que reproduzia o algoritmo do KNN para $k \in \{2, 3, \cdots, 20\}$ utilizando distância Euclidiana e aplicou no ploblema da Flor de Iris.

Buscando aproximar os alunos do ensino médio de algumas técnicas de aprendizagem de máquina, neste trabalho seguimos a ideia proposta no trabalho de Brandão (2023) que propõe utilizar aprendizado de máquina no ensino médio, diferente da proposta dele, que era utilizar o SVM ("Support Vector Machine"), criamos atividades de aprendizado de máquina no Geogebra, similares as criadas por Petrics (PETRICS, 2020a; PETRICS, 2020b) e (PANTALONI, 2023), pensando em facilitar a compreensão dos alunos do funcionamento dos algoritmos de predição KNN e árvore de decisão. Em Petrics (2020) ele trabalhou com distância euclidiana, como o KNN em Python também trabalha com distâncias não euclidianas e com inspiração nos trabalhos (LUZ, 2023; LEIVAS; PORTELLA; SOUZA, 2017; KALEFF; NASCIMENTO, 2004) introduzimos a distância de Manhattan no KNN.

4 MATERIAL E MÉTODO

A programação dos modelos abaixo foram feitas no Colab Python, veja:
Parte estatística dos dados Link para o Google Colab

4.1 MATERIAL

A obtenção dos dados foi realizada através de solicitação de acesso via memorando para os setores responsáveis pela posse dos dados (Pró-Reitoria de Ensino – PROENS) e pela armazenagem (Departamento de Tecnologia da Informação – DTI). Após esse processo foi feita a análise do projeto pelo Comitê de Ética. Após o recebimento das autorizações será disponibilizada uma cópia do banco de dados do Sistema Acadêmico da UFTM (SISCAD).

O conjunto de dados solicitado consiste das seguintes informações:

- Cidade do campus: Uberaba.
- Instituto (ICENE): instituto no qual o aluno está matriculado;
- Tipo do Curso: Licenciatura Plena;
- Área do conhecimento: Ciências Exatas e da Terra
- Curso: Licenciatura em Matemática
- Idade: idade do aluno no momento de entrada do aluno da instituição;
- Tipo de ingresso (Enem, vestibular, outros processos seletivos, etc.)
- Quantidade de reprovações por semestre;
- Quantidade de reprovações por falta por semestre;
- Quantidade de reprovações por nota por semestre;
- Quantidade de aprovações por semestre;
- Quantidade de trancamentos por semestre;
- Quantidade de disciplinas matriculados por semestre;
- Quantidade de carga horária por semestre;
- Quantidade de falta média por semestre;

- Nota média por semestre.
- Última ocorrência: Matriculado, Concluinte ou Evadido

4.2 MÉTODOS

Com a definição do problema e com os dados obtidos partimos para a análise e a aplicação das técnicas de mineração dos dados e avaliação dos dos resultados. Segue o esquema a ser seguido:

- Problema a ser resolvido;
- Base dos dados;
- Pré-processar os dados, fazendo uma limpeza e transformação dos dados;
- Seleção do modelo analítico e do algoritmo de processamento;
- Aplicação do modelo selecionado aos dados obtidos e transformados;
- Separar uma parte dos dados existentes em dois conjuntos de dados, chamados de conjunto de treinamento e conjunto de teste;
- Prever evasão ou não; e
- Avaliação dos Resultados.

4.2.1 **Problema a ser resolvido**

Pretendemos mostrar neste trabalho como os padrões temporais entre semestres podem fornecer informações sobre a experiência do aluno. Num primeiro momento, pretendíamos utilizar informação dentro dos semestres, isto é, informações mensais para a previsão de sucesso ou fracasso em um programa de graduação pode ser baseada em padrões de sequência de notas, trancamentos, faltas e atividades em mensais. Dados obtidos mensalmente produziriam uma maior quantidade de informação temporal dos dados, porém, devido o tempo que tínhamos para terminar o trabalho e também o tempo para que o Departamento de Tecnologia e Informática da UFTM -DTI nos fornecesse as informações na forma de dados mensais, optamos por obter dados semestrais, então começamos a trabalhar com vários algoritmos de classificação: Regressão Logística, Árvores de Decisão, k-Vizinhos Mais Próximos, Máquina de Vetores de Suporte, Redes Neurais e LSTM. Os modelos serão avaliados pelas medidas de Acurácia, Especificidade, F1-Score e Sensitividade.

4.2.2 Base de Dados

4.2.2.1 Dados Consultados do Banco de Dados do Sistema de Controle Acadêmico

Na etapa de pré-processamento foram realizadas consultas ao banco de dados do Sistema de Controle Acadêmico, ver Tabela 1. Alguns atributos que não possuem variação temporal, foram eliminados. Os atributos selecionados para a construção dos novos atributos propostos foram marcados como "SIM" e os que não foram selecionados como "NÃO";

```
df = pd.read_csv('/content/Dados.csv')
```

Tabela 1 – DESCRIÇÃO DOS DADOS E ATRIBUTOS RECOLHIDOS DO SISCAD.

DESCRIÇÃO DOS DADOS E ATRIBUTOS RECOLHIDOS DO SISCAD			
Atributo	Descrição	Tipo de Dado	Selecionado
Aluno	Número de matrícula em código	inteiro	Não
Ano	Ano que foi obtido os dados	inteiro	Não
Semestre	O Semestre do ano que foi obtido os dados	inteiro	Não
Período	O período do curso que o aluno estava	inteiro	Não
Sexo	Sexo do aluno	Texto	Não
Idade	Idade do aluno na início do curso	inteiro	Não
Forma de Ingresso	Forma de Ingresso do Aluno	Texto	Não
QtDisCanceladasSemestre	Quantidade de disciplinas canceladas por semestre	inteiro	Não
QtDisAprovadasSemestre	Quantidade de disciplinas Aprovadas por semestre	inteiro	Sim
QtDisReprovadasSemestre	Quantidade de disciplinas Reprovadas por semestre	inteiro	Sim
QtDisReprovadasSemestrePorNota	Quantidade de disciplinas reprovadas por nota por semestre	inteiro	Sim
QtDisReprovadasSemestrePorFalta	Quantidade de disciplinas reprovadas por falta por semestre	inteiro	Sim
PorcMediaFaltasSemestre	Porcentagem Média de Faltas por Semestre	Número real (float)	Sim
NotaMediaSemestre	Nota média por semestre	Número real (float)	Sim
QtDsciplinasSemestre	Quantidade de disciplinas matriculadas por Semestre	inteiro	Sim
ChDisciplinasSemestre	Carga Horária no semestre	inteiro	Sim
UltimaOcorrência	Informações sobre o aluno por semestre: matriculado, trancado, etc	texto	Não
TARGET	Informações se o aluno evadiu ou não	inteiro	Sim

Fonte: Próprio Autor

4.2.2.2 Seleção dos Atributos

Os atributos escolhidos foram os que nos trazem informações variantes com o tempo, isto é, eles carregam informações temporais, exceto do TARGET. Os atributos Idade e Quantidade de Trancamentos sejam variantes com o tempo, não foram escolhidos, pois no caso do atributo Idade, no trabalho (SOUZA *et al.*, 2021), foi detectado que idade não seria relevante para a evasão dos alunos do curso de licenciatura em Matemática da UFTM.

> Qual a idade média de quem evadiu? Os dados mostraram que não há uma diferença significativa entre a média das idades dos alunos que abandonaram o curso e a dos que concluíram, ou seja, não há relação entre idade e evasão para os cursos LCB, LF, LM e LQ. Para LEC, observa-se uma diferença maior de idade média entre os concluintes e os que abandonaram. (Souza, et all, 2018, pag. 14)

No caso do atributo quantidade de disciplinas canceladas por semestre ou quantidade de trancamento do curso, embora seja uma atributo importante para prever evasão, conforme os trabalhos (SOUZA *et al.*, 2021) e Ramon Nóbrega.

> Quem evadiu possui em média quantos trancamentos? O número médio de trancamentos entre aqueles que concluíram os cursos do ICENE no período analisado foi 0,23. Entre os alunos que abandonaram os cursos LCB, LF, LM e LQ, o valor foi 0,47, ou seja, um pouco superior ao dobro do primeiro valor. Portanto, a média de trancamento entre os alunos que evadem é maior e também pode ser utilizada para o diagnóstico precoce dos alunos que tendem a evadir. (Souza, et all, 2018, pag. 14)

Fizemos a opção de não escolher este atributo, pois da forma que nos foi fornecido os dados, quando o aluno faz o trancamento Geral do curso não aparece a quantidade de disciplinas canceladas e acreditamos que isso poderia distorcer este atributo quantidade de disciplinas canceladas por semestre.

Ao final desta etapa, produzimos conjuntos de dados com oito atributos $(X_1, X_2, \cdots, X_8)$ (características) e mais a variável $Y = TARGET$

```
df.dtypes
QtDisAprovadasSemestre                int64
QtDisReprovadasSemestre               int64
QtDisReprovadasSemestrePorNota        int64
QtDisReprovadasSemestrePorFalta       int64
PorcMediaFaltasSemestre             float64
NotaMediaSemestre                   float64
QtDsciplinasSemestre                  int64
ChDisciplinasSemestre               float64
```

Tabela 2 – DESCRIÇÃO DOS DADOS E ATRIBUTOS RECOLHIDOS DO SISCAD.

DESCRIÇÃO DOS DADOS E ATRIBUTOS RECOLHIDOS DO SISCAD			
Atributo	Descrição	Tipo de Dado	Selecionado
QtDisAprovadasSemestre	Quantidade de disciplinas Aprovadas por semestre	inteiro	Sim
QtDisReprovadasSemestre	Quantidade de disciplinas Reprovadas por semestre	inteiro	Sim
QtDisReprovadasSemestrePorNota	Quantidade de disciplinas reprovadas por nota por semestre	inteiro	Sim
QtDisReprovadasSemestrePorFalta	Quantidade de disciplinas reprovadas por falta por semestre	inteiro	Sim
PorcMediaFaltasSemestre	Porcentagem Média de Faltas por Semestre	Número real (float)	Sim
NotaMediaSemestre	Nota média por semestre	Número real (float)	Sim
QtDsciplinasSemestre	Quantidade de disciplinas matriculadas por Semestre	inteiro	Sim
ChDisciplinasSemestre	Carga Horária no semestre	Número real (float)	Sim
TARGET	Informações se o aluno evadiu ou não	inteiro	Sim

Fonte: Próprio Autor

```
10 TARGET                                int64
11 dtype: object
```

4.2.3 Base de dados de treinamento e teste

Como a implantação do SISCAD ocorreu em torno de 2013, agora em 2023 podemos analisar um ciclo completo de 9 anos (tempo de integralização máxima de um curso de Licenciatura), então serão feitas verificações utilizando-se os dados dos alunos que tiveram seu último contato com a UFTM a partir de 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021 e 2022. Os dados obtidos com informações precisas de "concluído"ou "evadido"nos levou a 450 alunos no período de 2014 até 2022. Se considerarmos subconjunto de dados do dataset principal, por exemplo, se considerarmos alunos que ficaram até o primeiro período, teremos o conjunto inteiro, isto é, 450 alunos, agora se considerarmos alunos que ficaram pelo menos até o segundo período, a quantidade de aluno cai para 256 e se

considerarmos alunos que ficaram pelo menos até o terceiro período, a quantidade de aluno cai para 186 e se considerarmos alunos que ficaram até o quarto período, a quantidade de alunos cai para 148 e se considerarmos alunos que ficaram pelo menos até o quinto período, a quantidade de aluno cai para 120 e se considerarmos alunos que ficaram pelo menos até o sexto período, a quantidade de aluno cai para 100 e se considerarmos alunos que ficaram pelo menos até o sétimo período, a quantidade de aluno cai para 94 e por fim se considerarmos alunos que ficaram pelo menos até o oitavo período, a quantidade de aluno cai para 78. Se considerarmos alunos que ficaram pelo menos até o nono período, a quantidade de aluno cai para 51 e se considerarmos alunos que ficaram pelo menos até o décimo período, a quantidade de aluno cai para 36.

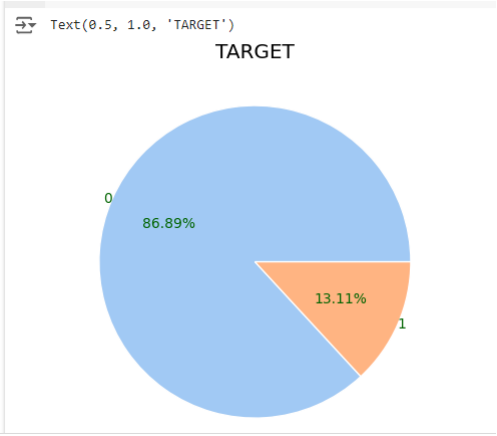
Neste trabalho iremos avaliar mais ainda nos capítulos posteriores, 6 algoritmos de aprendizado de máquina para detectar desistência (evasão) e/ou conclusão do curso. Os algoritmos escolhidos foram: Regressão Logística, Máquina de Vetores de Suporte, k-Vizinhos Mais Próximos-KNN, Árvores de Decisão, Multilayer Perceptron-MLP e LSTM. Estes algoritmos serão aplicados em 24 datasets diferentes. Os 24 conjunto de dados diferentes, vem da divisão do dataset principal em 8 datasets $(df1, df2, \cdots, df8)$, onde dfk indica o conjunto de dados dos alunos que cursaram pelo menos até o k-ésimo semestre e destes 8 datasets obtivemos 24 conjuntos de dados (8 desbalanceados, 8 balanceados pela técnica subamonstragem e 8 balanceados pela técnica sobreamostragem).

4.2.3.1 Estatísticas dos alunos evadidos do Curso de Licenciatura em Matemática da UFTM

As formas de evasão são: abandono, cancelamento administrativo, cancelamento de matrícula a pedido pelo aluno, transferência externa para outra instituição e Jubilamento.

Se considerarmos todos os alunos que efetivaram a matrícula e cursaram no mínimo o primeiro período do curso de matemática dos anos de 2014 até 2022, nos teremos um universo de 450 alunos matriculados, pelos dados fornecidos pelo SISCAD/UFTM, temos a informação de que $86,88\%$ tiveram taxa de insucesso, isto é, não concluíram o curso. Que está de acordo com as informações do INEP contidas na Figura 1, isto é, a taxa de insucesso ou taxa de desistência acumulada do curso de matemática no universo de 2013 até 2021 é em torno de $87,0\%$.

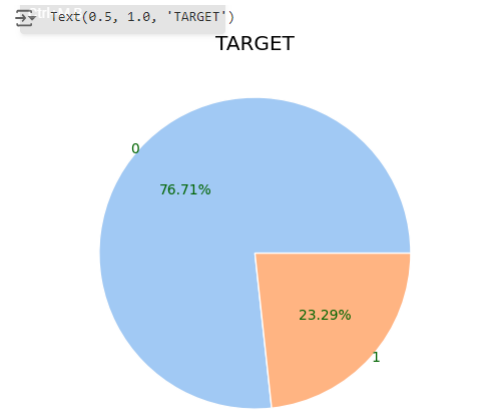
Figura 34 – Taxa de desistência depois do primeiro Período



Fonte: Produzido pelo autor tirados dos dados do SISCAD/UFTM

Quando considerarmos somente os alunos que cursaram no mínimo até o **segundo período** do curso de matemática no espaço dos anos de 2014 até 2022, nós teremos um universo de 256 alunos e a taxa de insucesso é de $76,71\%$, isto é, alunos que chegaram até o final do segundo período tem uma probabilidade de $76,71\%$ de não concluírem o curso.

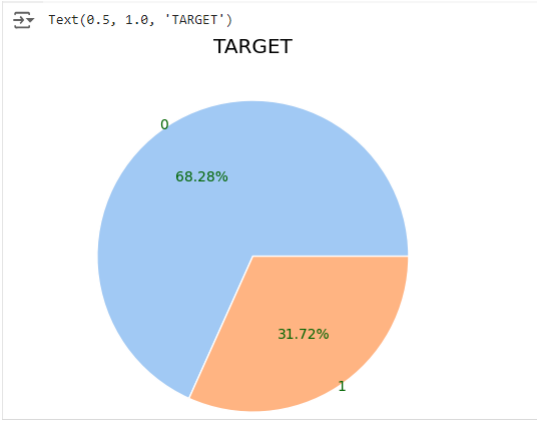
Figura 35 – Taxa de desistência depois do segundo período



Fonte: Produzido pelo autor tirados dos dados do SISCAD/UFTM

Quando considerarmos somente os alunos que cursaram no mínimo até o **terceiro período** do curso de matemática no espaço dos anos de 2014 até 2022, nós teremos um universo de 186 alunos e a taxa de insucesso é de $68,28\%$, isto é, alunos que chegaram até o final do terceiro período tem uma probabilidade de $68,28\%$ de não concluírem o curso.

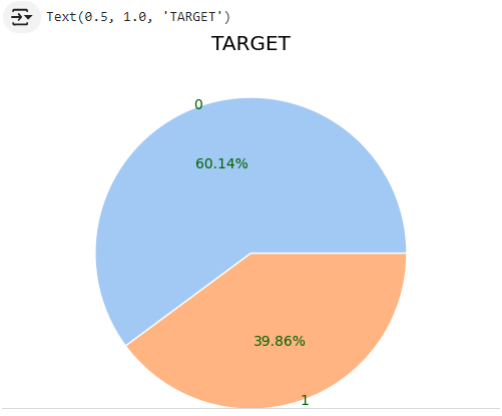
Figura 36 – Taxa de desistência depois do terceiro período



Fonte: Produzido pelo autor tirados dos dados do SISCAD/UFTM

Quando considerarmos somente os alunos que cursaram no mínimo até o **quarto período** do curso de matemática no espaço dos anos de 2014 até 2022, nós teremos um universo de 148 alunos e a taxa de insucesso é de $60,14\%$, isto é, alunos que chegaram até o final do quarto período tem uma probabilidade de $60,14\%$ de não concluírem o curso.

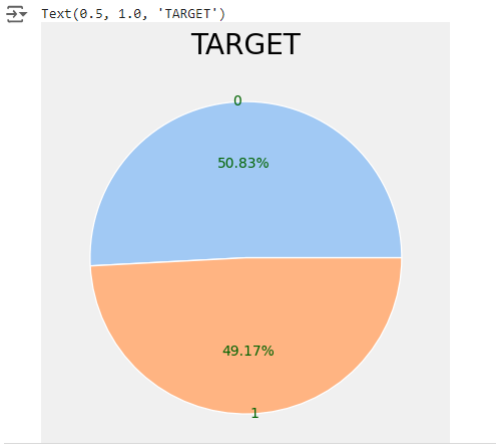
Figura 37 – Taxa de desistência depois do quinto período



Fonte: Produzido pelo autor tirados dos dados do SISCAD/UFTM

Quando considerarmos somente os alunos que cursaram no mínimo até **quinto período** do curso de matemática no espaço dos anos de 2014 até 2022, nós teremos um universo de 120 alunos e a taxa de insucesso é de $50,83\%$, isto é, alunos que chegaram até o final do quinto período tem uma probabilidade de $50,83\%$ de não concluírem o curso.

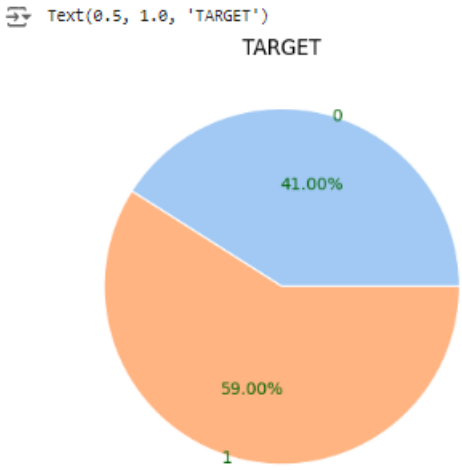
Figura 38 – Taxa de desistência depois do quinto período



Fonte: Produzido pelo autor tirados dos dados do SISCAD/UFTM

Quando considerarmos somente os alunos que cursaram no mínimo até o **sexto período** do curso de matemática no espaço dos anos de 2014 até 2022, nós teremos um universo de 100 alunos e a taxa de insucesso é de 41%, isto é, alunos que chegaram até o final do sexto período tem uma probabilidade de 41% de não concluírem o curso.

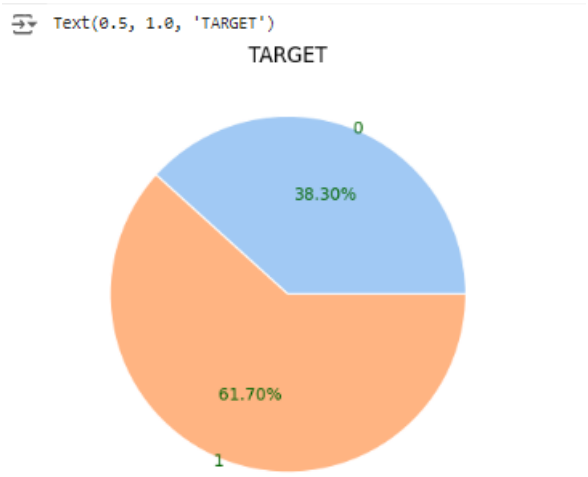
Figura 39 – Taxa de desistência depois do sexto período



Fonte: Produzido pelo autor tirados dos dados do SISCAD/UFTM

Quando considerarmos somente os alunos que cursaram no mínimo até o **sétimo período** do curso de matemática no espaço dos anos de 2014 até 2022, nós teremos um universo de 94 alunos e a taxa de insucesso é de $38,29\%$, isto é, alunos que chegaram até o final do sétimo período tem uma probabilidade de $38,29\%$ de não concluírem o curso.

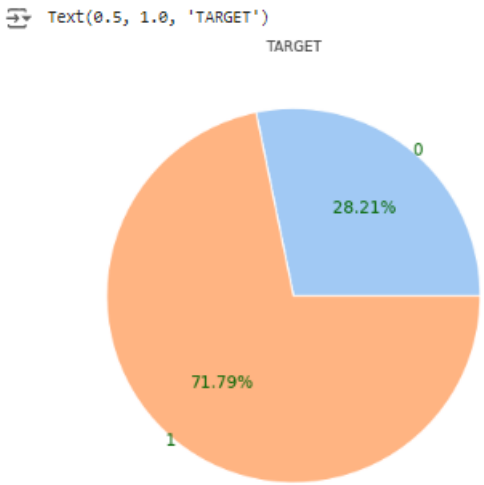
Figura 40 – Taxa de desistência depois do sétimo período



Fonte: Produzido pelo autor tirados dos dados do SISCAD/UFTM

Quando considerarmos somente os alunos que cursaram no mínimo até o **oitavo período** do curso de matemática no espaço dos anos de 2014 até 2022, nós teremos um universo de 78 alunos e a taxa de insucesso de $28,2\%$, isto é, alunos que chegaram até o final do sétimo período tem uma probabilidade de $28,2\%$ de não concluírem o curso.

Figura 41 – Taxa de desistência depois do oitavo período



Fonte: Produzido pelo autor tirados dos dados do SISCAD/UFTM

Obs: Só lembrando que índice de taxa de evasão oficial do MEC não é igual a taxa de insucesso (desistência), isto é, a taxa de evasão de um curso não é simplesmente pegar a razão entre a quantidade dos alunos que desistiram num certo período versus quantidade de alunos que poderiam ser matriculados naquele mesmo período, a fórmula usada para calcular evasão é dada pela equação 1.1. Esta fórmula mede um fluxo semestral ou anual dos alunos que poderiam se matricular naquele semestre menos os ingressantes daquele período versus a quantidade de alunos que poderiam ter se matriculados no período anterior menos os concluintes do período anterior.

Segundo informações da PROPLAN, ver a Figura 2, o índice de evasão do ano de 2014 do curso de Matemática da UFTM é de $32,41\%$ e se considerarmos todos os alunos que efetivaram a matrícula no ano de 2014 (1S e 2S), pelos dados que nos foram fornecidos pelo DTI/SISCAD, tivemos 37 alunos matriculados no ano de 2014, com 29 alunos não concluintes (desistências) e 8 alunos concluintes, isto nos fornece uma taxa de insucesso ou taxa de desistência do ano de 2014 de $78,38\%$.

4.2.3.2 Estatística dos Dados

Considerando os dados fornecidos fornecidos pelo DTI/SISCAD dos alunos do curso de matemática dos anos de 2014 até 2022, nós calculamos a média dos dados de cada atributo, isso nos fornece informação de todos os períodos. Vamos verificar as estatísticas Veja na tabela abaixo os dados estatísticos das variáveis atributos dos 450 alunos que cursaram pelo menos o **primeiro período** do curso de licenciatura em Matemática.

Tabela 3 – Estatísticas do Primeiro Período

QtDisAprovadasSemestre				QtDisReprovadasSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	450	59	391	Quantidade	450	59	391
Média	2.85	6.16	2.35	Média	3.06	0.18	3.49
dp	2.79	1.26	2.61	dp	2.62	0.60	2.54
min	0.0	3.0	0.0	min	0.0	0.0	0.0
25%	0.0	5.0	0.0	25%	0.0	0.0	1.0
50%	2.5	6.0	1.0	50%	3.0	0.0	4.0
75%	5.0	7.0	5.0	75%	5.0	0.0	5.0
max	8.0	8.0	8.0	max	8.0	3.0	8.0
QtDisReprovadasSemestrePorNota				QtDisReprovadasSemestrePorFalta			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	450	59	391	Quantidade	450	59	391
Média	0.70	0.11	0.79	Média	2.33	0.06	2.68
dp	1.02	0.49	1.04	dp	2.57	0.31	2.58
min	0.0	0.0	0.0	min	0.0	0.0	0.0
25%	0.0	0.0	0.0	25%	0.0	0.0	0.0
50%	0.0	0.0	0.0	50%	1.0	0.0	2.0
75%	1.0	0.0	1.0	75%	5.0	0.0	4.0
max	7.0	3.00	7.0	max	8.0	2.00	8.0
PorcMediaFaltasSemestre				NotaMediaSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	450	59	391	Quantidade	450	59	391
Média	0.34	0.04	0.38	Média	4.39	8.37	3.79
dp	0.34	0.04	0.34	dp	3.53	0.89	3.39
min	0.0	0.0	0.0	min	0.0	5.40	0.0
25%	0.04	0.02	0.06	25%	0.28	7.96	0.04
50%	0.18	0.03	0.29	50%	5.0	8.49	3.42
75%	0.65	0.05	0.75	75%	7.82	8.95	7.10
max	1.0	0.19	1.0	max	10.0	9.74	10.0
QtDsciplinasSemestre				ChDisciplinasSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	450	59	391	Quantidade	450	59	391
Média	5.92	6.35	5.84	Média	361.33	373.47	359.50
dp	1.46	1.21	1.48	dp	54.06	27.81	56.78
min	3.37	4.0	1.0	min	60.0	270.00	60.00
25%	5.0	5.0	5.0	25%	363.75	375.00	360.00
50%	6.0	6.0	6.0	50%	375.00	390.00	375.00
75%	7.0	8.0	7.0	75%	390.00	390.00	390.0
max	8.0	8.0	8.0	max	495.00	405.00	495.00

Fonte: Próprio Autor

Vamos verificar as estatísticas dos 250 alunos que cursaram pelo menos o **segundo período** do curso de licenciatura em Matemática. Veja os dados estatísticos das variáveis atributos.

Tabela 4 – Estatísticas do Segundo Período

QtDisAprovadasSemestre				QtDisReprovadasSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	250	59	191	Quantidade	250	59	191
Média	3.57	5.39	3.01	Média	1.79	0.37	2.23
dp	1.94	0.83	1.84	dp	1.64	0.63	1.61
min	0.0	2.0	0.0	min	0.0	0.0	0.0
25%	2.0	5.0	1.5	25%	0.5	0.0	1.0
50%	4.0	5.5	3.0	50%	1.5	0.0	2.0
75%	5.0	6.0	4.5	75%	3.0	0.5	3.5
max	7.0	7.0	6.5	max	7.5	2.5	7.5
QtDisReprovadasSemestrePorNota				QtDisReprovadasSemestrePorFalta			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	250	59	191	Quantidade	250	59	191
Média	0.7	0.27	0.83	Média	1.09	0.10	1.39
dp	0.72	0.51	0.73	dp	1.41	0.33	1.47
min	0.0	0.0	0.0	min	0.0	0.0	0.0
25%	0.0	0.0	0.5	25%	0.0	0.0	0.0
50%	0.5	0.0	0.5	50%	0.5	0.0	1.0
75%	1.0	0.5	1.5	75%	2.0	0.0	2.5
max	3.0	2.5	3.0	max	7.0	2.00	7.0
PorcMediaFaltasSemestre				NotaMediaSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	250	59	191	Quantidade	250	59	191
Média	0.20	0.04	0.25	Média	5.77	8.12	5.05
dp	0.21	0.04	0.22	dp	2.58	1.06	2.48
min	0.0	0.0	0.0	min	0.0	4.67	0.0
25%	0.04	0.02	0.06	25%	3.89	7.52	3.2
50%	0.1	0.03	0.16	50%	6.2	8.18	5.43
75%	0.32	0.05	0.4	75%	7.91	8.97	7.05
max	0.94	0.23	0.94	max	9.7	9.7	9.47
QtDsciplinasSemestre				ChDisciplinasSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	250	59	191	Quantidade	251	59	192
Média	5.37	5.77	5.25	Média	345.26	380.33	334.49
dp	1.02	0.65	1.08	dp	56.11	25.41	58.52
min	2.0	4.50	2.0	min	135.0	315.00	135.00
25%	4.62	5.50	4.5	25%	322.5	375.00	300.00
50%	5.5	5.5	5.0	50%	375.0	382.50	352.50
75%	6.0	6.5	6.0	75%	382.50	390.00	382.5
max	7.5	7.0	7.5	max	502.5	502.5	442.50

Fonte: Próprio Autor

Vamos verificar as estatísticas dos 186 alunos que cursaram pelo menos o **terceiro período** do curso de licenciatura em Matemática. Veja os dados estatísticos das variáveis atributos.

Tabela 5 – Estatísticas do Terceiro Período

QtDisAprovadasSemestre				QtDisReprovadasSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	186	59	127	Quantidade	186	59	127
Média	3.83	5.58	3.02	Média	1.54	0.36	2.09
dp	1.87	0.82	1.65	dp	1.46	0.56	1.42
min	0.0	2.67	0.0	min	0.0	0.0	0.0
25%	2.33	5.0	1.83	25%	0.33	0.0	1.0
50%	4.0	5.67	3.0	50%	1.0	0.0	2.0
75%	5.33	6.0	4.33	75%	2.33	0.67	3.0
max	6.67	6.67	6.67	max	6.67	2.33	6.67
QtDisReprovadasSemestrePorNota				QtDisReprovadasSemestrePorFalta			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	186	59	127	Quantidade	186	59	127
Média	0.63	0.45	0.79	Média	0.89	0.08	1.27
dp	0.65	0.0	0.67	dp	1.24	0.24	1.34
min	0.0	0.0	0.0	min	0.0	0.0	0.0
25%	0.0	0.00	0.33	25%	0.0	0.0	0.0
50%	0.33	0.00	0.67	50%	0.33	0.0	1.0
75%	1.0	0.33	1.0	75%	1.33	0.0	2.0
max	3.67	2.33	3.67	max	6.33	1.33	6.33
PorcMediaFaltasSemestre				NotaMediaSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	186	59	127	Quantidade	186	59	127
Média	0.16	0.04	0.22	Média	6.15	8.18	5.20
dp	0.18	0.04	0.19	dp	2.32	0.91	2.17
min	0.0	0.0	0.0	min	0.0	5.39	0.0
25%	0.04	0.02	0.08	25%	4.39	7.72	4.00
50%	0.09	0.03	0.17	50%	6.7	8.21	5.49
75%	0.22	0.06	0.33	75%	8.03	8.91	6.95
max	0.92	0.18	0.92	max	9.72	9.72	8.95
QtDsciplinasSemestre				ChDisciplinasSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	186	59	127	Quantidade	186	59	127
Média	5.37	5.94	5.11	Média	337.98	382.88	317.12
dp	1.01	0.57	1.06	dp	58.70	32.49	56.5
min	2.33	4.67	2.33	min	165.0	310.00	165.00
25%	4.67	5.50	4.33	25%	305.0	365.00	280.00
50%	5.67	6.0	5.33	50%	350.0	390.00	320.00
75%	6.0	6.33	6.0	75%	380.00	390.000	355.0
max	7.33	6.67	7.33	max	490.0	490.00	445.00

Fonte: Próprio Autor

Vamos verificar as estatísticas dos 148 alunos que cursaram pelo menos o **quarto período** do curso de licenciatura em Matemática. Veja os dados estatísticos das variáveis atributos.

Tabela 6 – Estatísticas do Quarto Período

QtDisAprovadasSemestre				QtDisReprovadasSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	148	59	89	Quantidade	148	59	89
Média	4.12	5.62	3.12	Média	1.23	0.32	1.84
dp	1.76	0.9	1.46	dp	1.17	0.51	1.1
min	0.25	2.25	0.25	min	0.0	0.0	0.0
25%	2.5	5.25	2.25	25%	0.25	0.0	1.0
50%	4.5	5.75	3.0	50%	1.0	0.0	1.75
75%	5.5	6.25	4.25	75%	2.06	0.5	2.75
max	6.75	6.75	6.5	max	5.25	2.5	5.25
QtDisReprovadasSemestrePorNota				QtDisReprovadasSemestrePorFalta			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	148	59	89	Quantidade	148	59	89
Média	0.57	0.24	0.79	Média	0.65	0.08	1.03
dp	0.61	0.43	0.62	dp	0.9	0.18	0.98
min	0.0	0.0	0.0	min	0.0	0.0	0.0
25%	0.0	0.0	0.25	25%	0.0	0.0	0.25
50%	0.5	0.0	0.75	50%	0.25	0.0	0.75
75%	0.75	0.25	1.0	75%	1.06	0.0	1.75
max	3.25	2.5	3.25	max	4.75	1.00	4.75
PorcMediaFaltasSemestre				NotaMediaSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	148	59	89	Quantidade	148	59	89
Média	0.14	0.04	0.21	Média	6.59	8.28	5.47
dp	0.14	0.36	0.15	dp	2.00	0.84	1.75
min	0.01	0.01	0.01	min	1.09	5.56	1.09
25%	0.04	0.02	0.09	25%	5.25	7.86	4.09
50%	0.08	0.03	0.18	50%	7.0	8.34	5.61
75%	0.21	0.05	0.31	75%	8.06	8.89	6.94
max	0.77	0.16	0.77	max	9.65	9.65	8.83
QtDsciplinasSemestre				ChDisciplinasSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	148	59	89	Quantidade	148	59	89
Média	5.39	5.98	4.99	Média	333.38	377.15	304.36
dp	1.03	0.61	1.06	dp	66.38	57.08	55.48
min	2.0	4.0	2.0	min	33.75	33.75	146.25
25%	4.75	5.62	4.25	25%	288.75	371.25	270.00
50%	5.5	6.25	5.25	50%	343.25	397.50	315.00
75%	6.25	6.37	5.75	75%	386.56	397.50	341.5
max	7.0	7.0	6.75	max	453.75	453.75	401.25

Fonte: Próprio Autor

Vamos verificar as estatísticas dos 120 alunos que cursaram pelo menos o **quinto período** do curso de licenciatura em Matemática. Veja os dados estatísticos das variáveis atributos.

Tabela 7 – Estatísticas do Quinto Período

QtDisAprovadasSemestre				QtDisReprovadasSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	120	59	61	Quantidade	120	59	61
Média	4.49	5.76	3.25	Média	1.01	0.31	1.68
dp	1.75	0.97	1.44	dp	1.03	0.46	0.99
min	0.4	2.4	0.4	min	0.0	0.0	0.0
25%	3.0	5.3	2.2	25%	0.2	0.0	1.0
50%	5.0	5.8	3.0	50%	0.6	0.2	1.6
75%	5.8	6.4	4.6	75%	1.6	0.4	2.4
max	7.8	7.8	6.2	max	4.8	2.2	4.8
QtDisReprovadasSemestrePorNota				QtDisReprovadasSemestrePorFalta			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	120	59	61	Quantidade	120	59	61
Média	0.50	0.24	0.76	Média	0.49	0.07	0.9
dp	0.55	0.40	0.56	dp	0.76	0.15	0.89
min	0.0	0.0	0.0	min	0.0	0.0	0.0
25%	0.0	0.0	0.4	25%	0.0	0.0	0.2
50%	0.4	0.2	0.6	50%	0.2	0.0	0.6
75%	0.8	0.4	1.0	75%	0.65	0.0	1.4
max	2.4	2.2	2.4	max	4.4	0.8	4.4
PorcMediaFaltasSemestre				NotaMediaSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	120	59	61	Quantidade	120	59	61
Média	0.12	0.04	0.19	Média	6.99	8.34	5.68
dp	0.13	0.03	0.14	dp	1.88	0.79	1.69
min	0.0	0.0	0.01	min	1.13	5.77	1.13
25%	0.03	0.01	0.09	25%	5.84	7.99	4.29
50%	0.06	0.03	0.14	50%	7.43	8.42	5.9
75%	0.14	0.05	0.3	75%	8.38	8.97	7.08
max	0.6	0.13	0.6	max	9.6	9.6	8.28
QtDsciplinasSemestre				ChDisciplinasSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	120	59	61	Quantidade	120	59	61
Média	5.5	6.07	4.94	Média	346.75	392.44	302.55
dp	1.01	0.74	0.93	dp	64.45	41.40	50.43
min	2.2	4.0	2.2	min	153.0	276.00	153.00
25%	5.0	5.6	4.2	25%	303.0	367.50	273.00
50%	5.6	6.2	5.0	50%	355.50	402.00	309.00
75%	6.25	6.6	5.6	75%	395.25	418.50	342.0
max	7.8	7.8	6.4	max	468.0	468.0	381.00

Fonte: Próprio Autor

Vamos verificar as estatísticas dos 100 alunos que cursaram pelo menos o **sexto período** do curso de licenciatura em Matemática. Veja os dados estatísticos das variáveis atributos.

Tabela 8 – Estatísticas do Sexto Período

QtDisAprovadasSemestre				QtDisReprovadasSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	100	59	41	Quantidade	100	59	41
Média	4.75	5.71	3.36	Média	0.81	0.32	1.51
dp	1.6	0.89	1.37	dp	0.88	0.45	0.89
min	0.83	2.5	0.83	min	0.0	0.0	0.0
25%	3.5	5.33	2.33	25%	0.12	0.0	0.83
50%	5.17	5.67	3.33	50%	0.41	0.17	1.5
75%	6.0	6.25	4.67	75%	1.37	0.33	2.0
max	7.33	7.33	6.0	max	3.5	2.00	3.5
QtDisReprovadasSemestrePorNota				QtDisReprovadasSemestrePorFalta			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	100	59	41	Quantidade	100	59	41
Média	0.46	0.24	0.78	Média	0.33	0.07	0.69
dp	0.52	0.39	0.52	dp	0.54	0.13	0.69
min	0.0	0.0	0.0	min	0.0	0.0	0.0
25%	0.0	0.0	0.33	25%	0.0	0.0	0.17
50%	0.33	0.17	0.67	50%	0.17	0.0	0.5
75%	0.67	0.33	1.17	75%	0.33	0.17	1.0
max	2.0	2.00	2.0	max	2.33	0.67	2.33
PorcMediaFaltasSemestre				NotaMediaSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	100	59	41	Quantidade	100	59	41
Média	0.09	0.04	0.16	Média	7.34	8.32	5.94
dp	0.1	0.03	0.12	dp	1.67	0.79	1.61
min	0.0	0.0	0.01	min	1.8	5.84	1.81
25%	0.02	0.02	0.07	25%	6.41	7.93	4.91
50%	0.05	0.03	0.12	50%	7.88	8.38	5.99
75%	0.11	0.05	0.24	75%	8.55	8.94	7.4
max	0.47	0.17	0.47	max	9.63	9.63	8.26
QtDsciplinasSemestre				ChDisciplinasSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	100	59	41	Quantidade	100	59	41
Média	5.56	8.32	4.87	Média	355.85	396.86	296.82
dp	0.9	0.79	0.79	dp	64.97	39.48	46.19
min	3.5	5.84	3.5	min	195.0	290.0	195.00
25%	5.0	7.93	4.33	25%	305.0	373.75	267.50
50%	5.67	8.38	5.0	50%	370.0	402.5	305.00
75%	6.33	8.94	5.5	75%	407.5	422.5	322.5
max	7.33	9.63	6.33	max	472.5	472.5	392.50

Fonte: Próprio Autor

Vamos verificar as estatísticas dos 94 alunos que cursaram pelo menos o **sétimo período** do curso de licenciatura em Matemática. Veja os dados estatísticos das variáveis atributos.

Tabela 9 – Estatísticas do Sétimo Período

QtDisAprovadasSemestre				QtDisReprovadasSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	94	58	36	Quantidade	94	58	36
Média	4.75	5.72	3.23	Média	0.78	0.32	1.54
dp	1.6	0.82	1.33	dp	0.85	0.40	0.85
min	0.71	2.71	0.71	min	0.0	0.0	0.14
25%	3.5	5.29	2.14	25%	0.14	0.0	1.43
50%	5.29	5.71	3.0	50%	0.43	0.14	2.07
75%	5.86	6.14	4.43	75%	1.29	0.43	2.07
max	7.0	7.00	5.71	max	3.43	1.71	3.43
QtDisReprovadasSemestrePorNota				QtDisReprovadasSemestrePorFalta			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	94	58	36	Quantidade	94	58	36
Média	0.42	0.22	0.75	Média	0.34	0.08	0.76
dp	0.47	0.34	0.49	dp	0.56	0.15	0.71
min	0.0	0.0	0.0	min	0.0	0.0	0.0
25%	0.0	0.0	0.39	25%	0.0	0.0	0.14
50%	0.29	0.14	0.57	50%	0.14	0.0	0.5
75%	0.57	0.29	1.03	75%	0.43	0.14	1.07
max	1.86	1.71	1.86	max	2.43	0.86	2.43
PorcMediaFaltasSemestre				NotaMediaSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	94	58	36	Quantidade	94	58	36
Média	0.09	0.04	0.18	Média	7.33	8.29	5.78
dp	0.11	0.03	0.13	dp	1.7	0.75	1.67
min	0.01	0.01	0.04	min	1.55	6.21	1.55
25%	0.02	0.02	0.07	25%	6.52	7.88	4.73
50%	0.05	0.03	0.15	50%	7.84	8.35	5.96
75%	0.11	0.05	0.25	75%	8.58	8.85	7.35
max	0.5	0.15	0.5	max	9.63	9.63	8.07
QtDsciplinasSemestre				ChDisciplinasSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	94	58	36	Quantidade	94	58	36
Média	5.56	6.04	4.78	Média	352.98	401.86	289.76
dp	0.87	0.58	0.66	dp	67.96	36.17	46.56
min	3.57	4.29	3.57	min	197.14	310.71	197.14
25%	4.89	5.71	4.25	25%	306.43	379.82	268.39
50%	5.71	6.07	4.86	50%	370.71	402.86	286.07
75%	6.29	6.43	5.17	75%	413.03	432.32	312.85
max	7.0	7.0	6.14	max	467.14	467.14	372.86

Fonte: Próprio Autor

Vamos verificar as estatísticas dos 78 alunos que cursaram pelo menos o **oitavo período** do curso de licenciatura em Matemática. Veja os dados estatísticos das variáveis atributos.

Tabela 10 – Estatísticas do Oitavo Período

QtDisAprovadasSemestre				QtDisReprovadasSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	78	56	22	Quantidade	78	56	22
Média	4.92	5.61	3.15	Média	0.65	0.33	1.48
dp	1.42	0.73	1.2	dp	0.75	0.4	0.81
min	1.12	2.87	1.12	min	0.0	0.0	0.25
25%	4.37	5.25	2.25	25%	0.03	0.0	0.88
50%	5.25	5.75	3.2	50%	0.38	0.25	1.37
75%	5.87	6.0	4.27	75%	0.88	0.53	2.09
max	6.75	6.75	5.0	max	3.12	1.5	3.12
QtDisReprovadasSemestrePorNota				QtDisReprovadasSemestrePorFalta			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	78	56	22	Quantidade	78	56	22
Média	0.37	0.23	0.72	Média	0.25	0.08	0.69
dp	0.4	0.32	0.39	dp	0.48	0.14	0.74
min	0.0	0.0	0.25	min	0.0	0.0	0.0
25%	0.0	0.0	0.38	25%	0.0	0.0	0.15
50%	0.25	0.12	0.62	50%	0.12	0.0	0.44
75%	0.62	0.38	1.09	75%	0.25	0.12	0.97
max	1.5	1.5	1.5	max	2.62	0.75	2.62
PorcMediaFaltasSemestre				NotaMediaSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	78	56	22	Quantidade	78	56	22
Média	0.07	0.04	0.17	Média	7.55	8.2	5.86
dp	0.08	0.03	0.11	dp	1.47	0.75	1.53
min	0.01	0.01	0.04	min	2.3	6.4	2.3
25%	0.02	0.02	0.08	25%	7.1	7.74	5.25
50%	0.04	0.03	0.13	50%	7.9	8.3	6.35
75%	0.09	0.05	0.23	75%	8.6	8.82	7.05
max	0.5	0.14	0.5	max	9.4	9.4	7.7
QtDsciplinasSemestre				ChDisciplinasSemestre			
Estatísticas	Todos	Formandos	Evadidos	Estatísticas	Todos	Formandos	Evadidos
Quantidade	78	56	22	Quantidade	78	56	22
Média	5.55	5.91	4.61	Média	369.27	399.75	291.71
dp	0.79	0.49	0.64	dp	61.68	33.69	46.89
min	3.5	4.3	3.5	min	217.5	303.75	217.50
25%	5.02	5.75	4.13	25%	337.5	380.60	265.75
50%	5.8	6.0	4.6	50%	382.45	394.75	290.55
75%	6.1	6.25	5.02	75%	420.87	423.75	328.12
max	6.77	6.75	5.8	max	455.6	455.6	375.00

Fonte: Próprio Autor

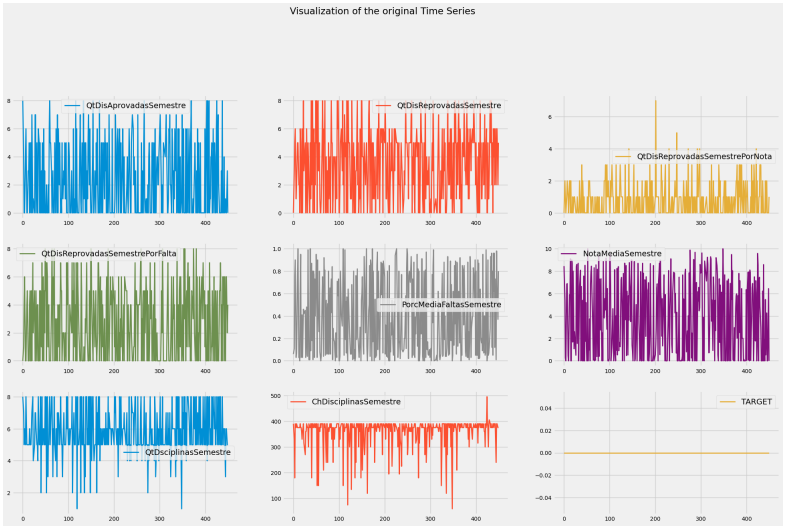
4.2.3.3 Visualização Gráfica dos Dados

Considerando os alunos que cursaram pelo menos o **primeiro período** do curso de Licenciatura em Matemática, identificamos um total de 450 alunos. A seguir, apresentamos os gráficos relacionados às variáveis e atributos analisados.

Figura 42 – Gráfico Variáveis Atributos para os concluintes

Fonte: Produzido pelo autor tirados dos dados do SISCAD/UFTM

Figura 43 – Gráfico Variáveis Atributos para os desistentes



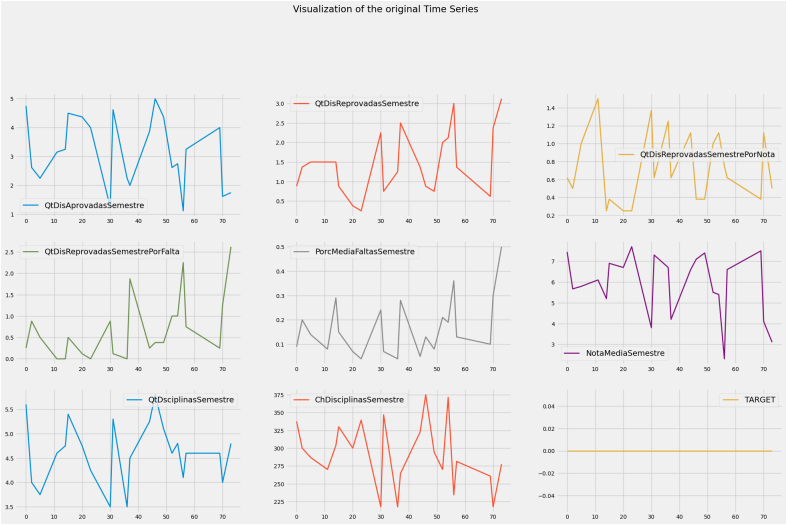
Fonte: Produzido pelo autor tirados dos dados do SISCAD/UFTM

Focando nos alunos que cursaram pelo menos o **oitavo período** do curso de Licenciatura em Matemática, identificamos um total de 78 alunos. Abaixo, apresentamos os gráficos referentes às variáveis e atributos analisados.

Figura 44 – Gráfico Variáveis Atributos para os concluintes

Fonte: Produzido pelo autor tirados dos dados do SISCAD/UFTM

Figura 45 – Gráfico Variáveis Atributos para os desistentes



Fonte: Produzido pelo autor tirados dos dados do SISCAD/UFTM

4.2.3.4 Correlação dos Dados

Existem três tipos de correlações mais conhecidas e frequentemente utilizadas, que são a correlação de Pearson r_{XY}, a correlação de Spearman ρ_s e correlação de Kendall τ. Cada uma delas possuem suas particularidades e melhores casos de utilização. Mas elas possuem algo em comum: não podem ser utilizadas com variáveis categóricas a nível nominal.

Pela própria definição, uma variável categórica nominal é caracterizada pelo nome da categoria à qual pertence, por exemplo, conclusão ou não do curso, status civil (solteiro, casado, divorciado, etc) e sexo(homem ou mulher). Variáveis deste tipo não podem ser quantificadas por números, porém é comum representá-las por códigos numéricos, do tipo 0=evasão e 1=conclusão. Mas códigos numéricos são apenas uma maneira abreviada de se referir às categorias e não tem qualquer sentido considerá-los como valores em um estudo de correlação.

Quando a variável resposta(label ou Target) é dicotômica (binária), utiliza-se a correlação Ponto-Bisserial.

A correlação ponto-bisserial é matematicamente equivalente à correlação de Pearson (produto-momento), isto é, se temos uma variável X medida continuamente e uma variável Y dicotômica, $r_{XY} = r_{pb}$. Isso pode ser demonstrado através da atribuição de dois valores numéricos diferentes para a variável dicotômica.

```
import scipy.stats as stats
def realizar_teste_ponto_bisserial(variavel, nome_variavel):
  x=variavel
  y=df['TARGET']
  estatistica, valor_p = stats.pointbiserialr(x,y)
  print(f"Test ponto-bisseral para {nome_variavel}:")
  print(f"Estatistica: {estatistica}")
  print(f"Valor p: {valor_p}")
  if valor_p > 0.05:
    print(f"A {nome_variavel} parece seguir uma distribuicao normal.")
  else:
    print(f"A {nome_variavel} nao parece seguir uma distribuicao normal.")
  print("\n")

variaveis = [
    ('QtDisAprovadasSemestre', df['QtDisAprovadasSemestre']),
    ('QtDisReprovadasSemestre', df['QtDisReprovadasSemestre']),
    ('QtDisReprovadasSemestrePorNota', df['QtDisReprovadasSemestrePorNota']),
    ('QtDisReprovadasSemestrePorFalta', df['
```

```
        QtDisReprovadasSemestrePorFalta']),
    ('PorcMediaFaltasSemestre', df['PorcMediaFaltasSemestre']),
    ('NotaMediaSemestre', df['NotaMediaSemestre']),
    ('QtDsciplinasSemestre', df['QtDsciplinasSemestre']),
    ('ChDisciplinasSemestre', df['ChDisciplinasSemestre'])]
for nome, variavel in variaveis:
  realizar_teste_ponto_bisserial(variavel,nome)
```

Só para exemplificar os resultados, iremos mostrar a correlação entre os atributos $QtDisAprovadasSemestre \times TARGET$ e $QtDisReprovadasSemestre \times TARGET$

```
Test ponto-bisseral para QtDisAprovadasSemestre:
Estatistica: 0.549605662723384
Valor p: 1.1806490358626214e-48
A QtDisAprovadasSemestre n\~ao parece seguir uma distribui\c  c\~ao
    normal.


Test ponto-bisseral para QtDisReprovadasSemestre:
Estat\'istica: -0.5092872532293459
Valor p: 6.524932638767928e-41
A QtDisReprovadasSemestre n\~ao parece seguir uma distribui\c c\~ao
    normal.
```

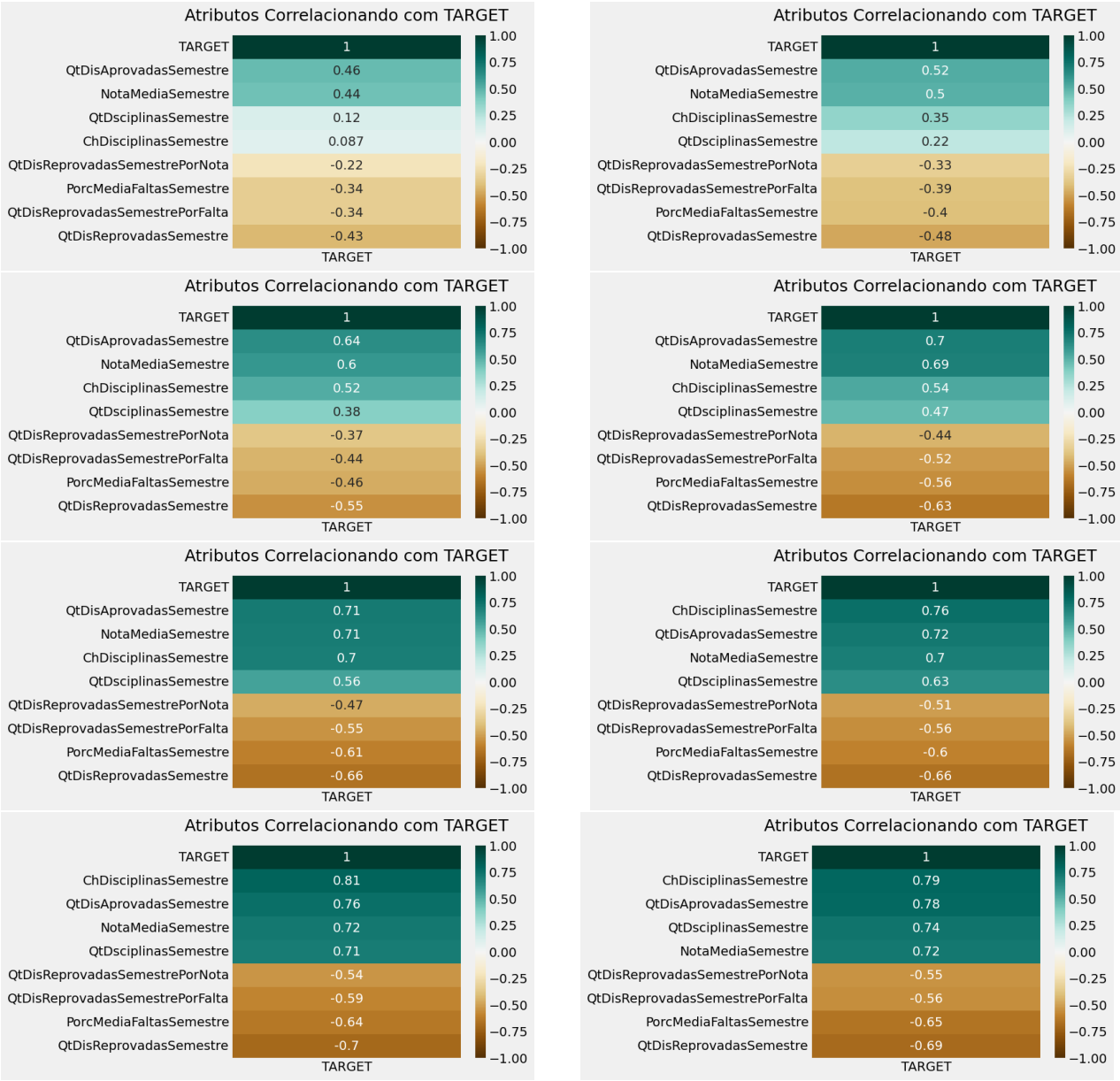
Como a correlação de Ponto-Bisserial é matematicamente equivalente à correlação de Pearson, então segue as correlação de Pearson para o caso dos dados de alunos que cursaram pelo menos o primeiro, segundo, terceiro, quarto, quinto, sexto, sétimo ou oitavo período.

4.2.3.5 Matriz de correlação de Pearson

A matriz de correlação de Person em Python.

```
corr_matrix = df.corr("pearson")
f, ax = plt.subplots(figsize=(16,8))
# Seleccionar a metade superior da matriz de correla\c c\~ao
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))
sns.heatmap(corr_matrix,mask=mask, annot=True, fmt='.2f', linewidth
    =0.4,
            annot_kws={"size": 10}, cmap='coolwarm', ax=ax)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.show()
```

Figura 46 – Os coeficientes de correlação de Pearson das variáveis atributos com a variável Target (taxa de sucesso) dos alunos que cursaram pelo menos o **primeiro, segundo, ... ,sétimo e oitavo** período, respectivamente.



Fonte: Produzido pelo autor tirados dos dados do SISCAD/UFTM

4.3 PRÉ-PROCESSAMENTO DOS DADOS

Pré-processar os dados, fazendo uma limpeza e transformação dos dados.

4.3.1 Remoção de valores nulos e outliers

Todos os atributos foram verificados quanto a incidência de valores nulos, não foi encontrado nenhum registro com valores nulos.

```
faltantes1=df1aux.isnull().sum()
faltantes2=df2aux.isnull().sum()
faltantes3=df3aux.isnull().sum()
faltantes4=df4aux.isnull().sum()
faltantes5=df5aux.isnull().sum()
faltantes6=df6aux.isnull().sum()
faltantes7=df7aux.isnull().sum()
faltantes8=df8aux.isnull().sum()
```

Por exemplo, os dados dos alunos que fizeram até o oitavo (8°) período.

```
faltantes8.head()
QtDisAprovadasSemestre  0
QtDisReprovadasSemestre 0
QtDisReprovadasSemestrePorNota  0
QtDisReprovadasSemestrePorFalta 0
PorcMediaFaltasSemestre 0
NotaMediaSemestre   0
QtDsciplinasSemestre    0
ChDisciplinasSemestre   0
TARGET  0
dtype: int64
```

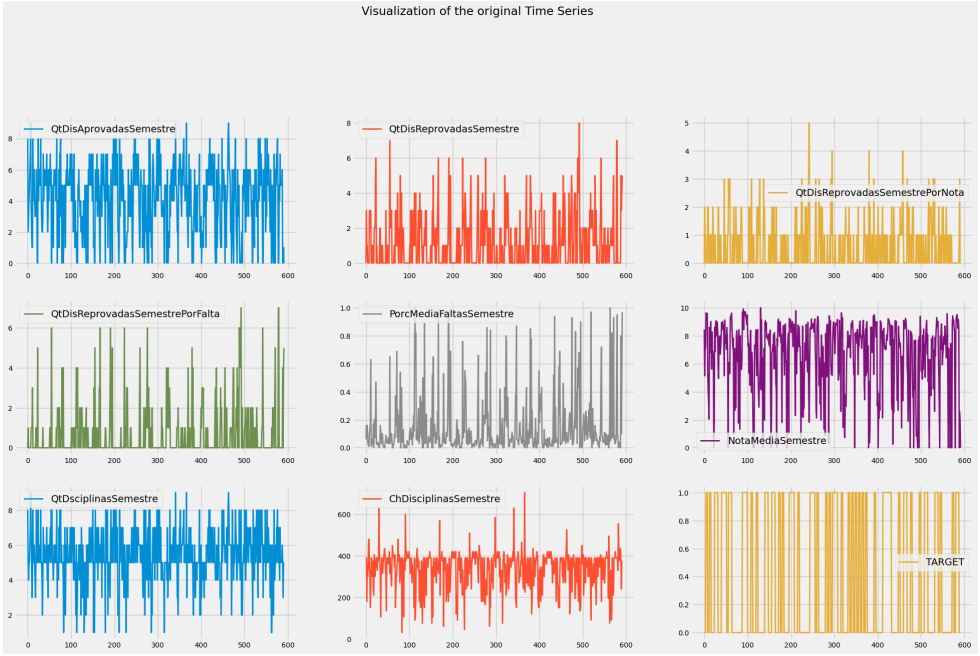
4.3.2 Remoção de outliers

Verifica a incidência de pouco outliers

Problemas causados pelos outliers:

A presença de outliers pode causar sérios problemas nas análises de dados e desenvolvimento de modelos. Viés nos resultados e impacto nas métricas descritivas como média, mediana e desvio padrão. Risco de overfitting de modelos devido à diferença dos outliers em relação ao padrão dos dados. Consequências dos outliers, como impacto na capacidade de generalização dos modelos.

Figura 47 – Gráfico Variáveis Atributos para os concluintes



Fonte: Produzido pelo autor tirados dos dados do SISCAD/UFTM

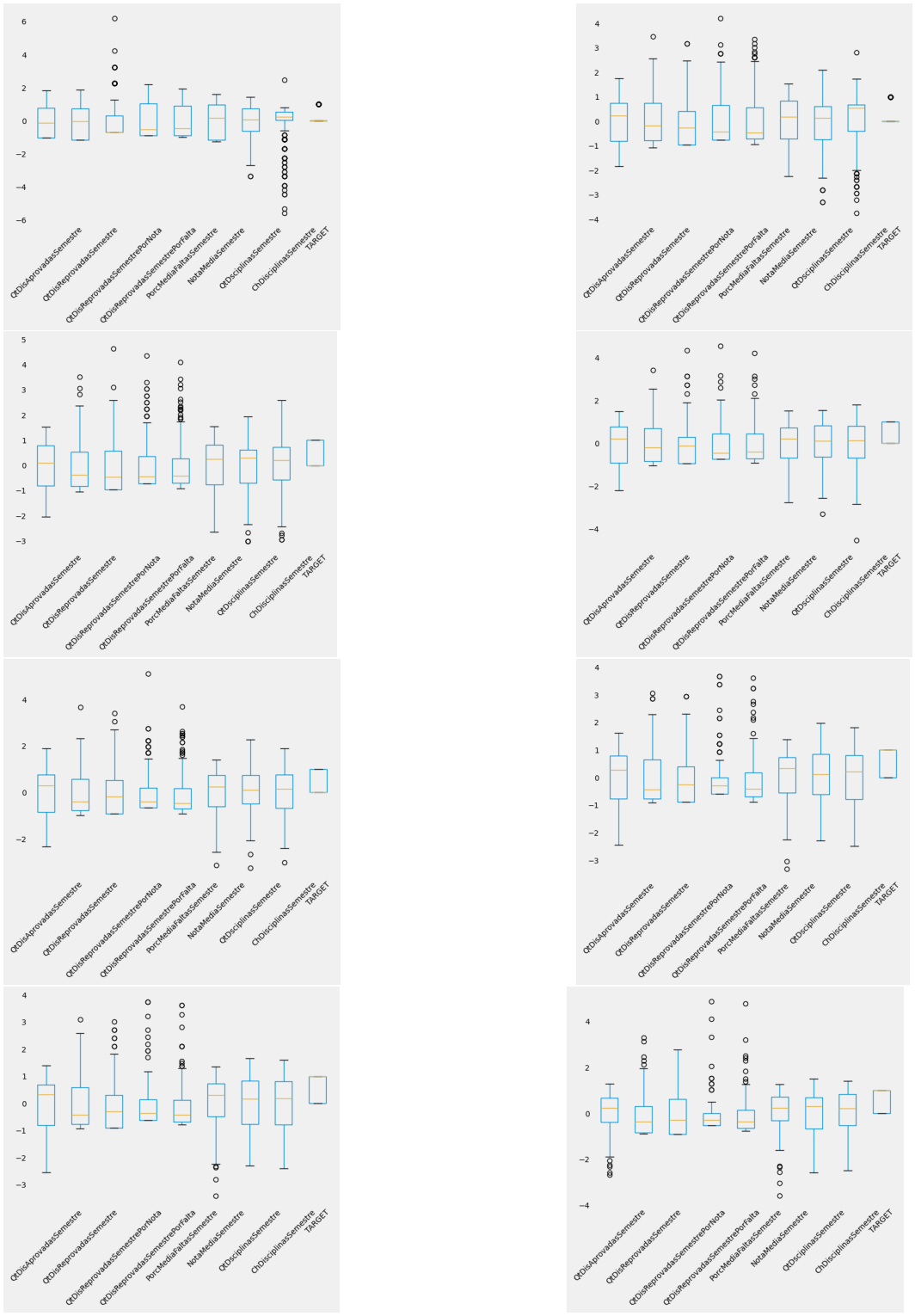
Identificando outliers com boxplots: Uma forma visual muito utilizada para detectar a presença de outliers é o boxplot (ou gráfico de caixa). Nele, os quartis e a mediana dividem os dados em 4 partes iguais. Os valores acima do limite superior (Q3 + 1,5 x intervalo interquartilar) ou abaixo do limite inferior (Q1 – 1,5 x intervalo interquartilar) são considerados outliers.

O boxplot é uma ferramenta visual eficaz para identificar outliers em conjuntos de dados. Os quartis e a mediana são utilizados para dividir os dados em partes iguais. Valores acima do limite superior ou abaixo do limite inferior são considerados outliers

Removendo outliers: Uma vez identificados os outliers, uma abordagem comum é simplesmente removê-los do conjunto de dados, descartando os pontos problemáticos antes das análises.

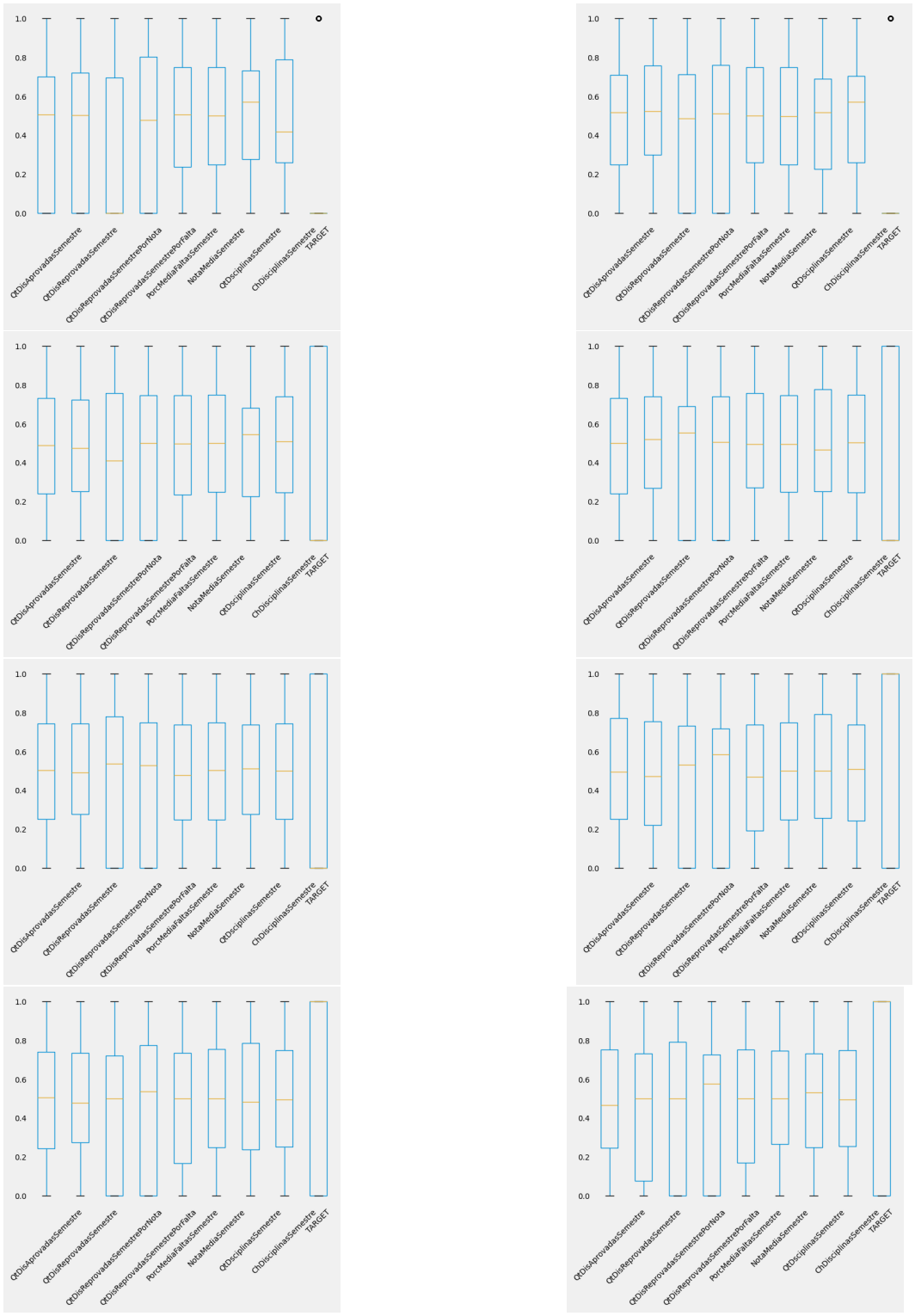
A remoção dos outliers é uma abordagem comum para tratá-los em conjuntos de dados. Descartar os pontos problemáticos antes das análises pode ajudar a manter a integridade dos resultados. A remoção dos outliers é uma etapa importante no pré-processamento de dados.

Figura 48 – Aplicando a normalização **StandardScaler** para os dados dos alunos que cursaram pelo menos o **primeiro, segundo, ... ,sétimo e oitavo** período, respectivamente.



Fonte: Produzido pelo autor tirados dos dados do SISCAD/UFTM

Figura 49 – Aplicando a normalização **QuantileTransformer** para os dados dos alunos que cursaram pelo menos o **primeiro, segundo, ... ,sétimo e oitavo** período, respectivamente.



Fonte: Produzido pelo autor tirados dos dados do SISCAD/UFTM

4.3.3 Transformações

Embora a suavização dos outliers foi melhor aplicando QuantileTransformer, não necessariamente obtemos um resultado melhor do que aplicar o StanderScaler.

```
# Normalizar os dados
#scaler = Normalizer()
#scaler = MinMaxScaler(feature_range=(0,1))
scaler = StandardScaler()
#scaler = RobustScaler()
#scaler = PowerTransformer()
#scaler = QuantileTransformer()

df1aux[features] = scaler.fit_transform(df1[features])
df2aux[features] = scaler.fit_transform(df2[features])
df3aux[features] = scaler.fit_transform(df3[features])
df4aux[features] = scaler.fit_transform(df4[features])
df5aux[features] = scaler.fit_transform(df5[features])
df6aux[features] = scaler.fit_transform(df6[features])
df7aux[features] = scaler.fit_transform(df7[features])
df8aux[features] = scaler.fit_transform(df8[features])
```

4.3.4 Separação dos Dados

4.3.4.1 Regressão Logística, Máquina de Vetores de Suporte, k-Vizinhos Mais Próximos-KNN e Árvores de Decisão

Para os algoritmos Regressão Logística, Máquina de Vetores de Suporte, k-Vizinhos Mais Próximos-KNN, Árvores de Decisão aplicamos um **for** para tirar a média e desvio padrão. Tiramos do train-test-split o ramdom-state. Aplicamos 30 testes utilizando a proporção 85% para dados de treino e validação e 15% e para dados teste. Dos 85% para dados de treino e validação, separamos 80% para dados de treino e validação e 20%, dando uma porcentagem dos dados originais de 68% para dados de treino, 17% para validação e 15% e para dados teste.

Divisão dos dados

```
features = df.columns.drop('TARGET')
features
previsores = df[features]
alvo = df.drop(features, axis=1)['TARGET']
for i in range(0,10):
# Dividir em dados de treino/valida\c c\~ao e teste
```

```
X_treino_val, X_teste, y_treino_val, y_teste = train_test_split(
    previsores, alvo, train_size=0.85)
# Dividir novamente os dados de treino/valida\c c\~ao em treino e
    valida\c c\~ao
X_treino, X_val, y_treino, y_val = train_test_split(X_treino_val,
    y_treino_val, test_size=0.2)
```

4.3.4.1.1 Multilayer Perceptron-MLP

Para os algoritmos Multilayer Perceptron-MLP também aplicamos um **for** para tirar a média e desvio padrão e tiramos do train-test-split o ramdom-state, porém fizemos um for com 10 iterações. Justificativa foi pela demora em rodar o colab. Aplicamos 5 testes utilizando a proporção 80% para dados de treino e validação e 20% para dados de teste. Dos 80% para treino e validação, fica 75% para treino e 25% para validação. Com base nos resultados dos modelos aplicados nos conjuntos de dados para **testes**, tivemos os seguintes resultados com intervalos de confiança de 95%.

Divisão dos dados

```
features = df.columns.drop('TARGET')
features
previsores = df[features]
alvo = df.drop(features, axis=1)['TARGET']
for i in range(0,10):
# Dividir em dados de treino/validação e teste
X_treino_val, X_teste, y_treino_val, y_teste =
             train_test_split(previsores, alvo, train_size=0.80)
# Dividir novamente os dados de treino/validação em treino e validação
X_treino, X_val, y_treino, y_val =
            train_test_split(X_treino_val, y_treino_val, test_size=0.2)
```

Modelos utilizados

```
model = Sequential()
model.add(Dense(100, input_dim=n_features,kernel_initializer='
    normal',
          activation='relu'))
model.add(Dense(2, kernel_initializer='normal',activation='softmax'
    ))
model.summary()
model.compile(optimizer=SGD(), loss='categorical_crossentropy',
    metrics=['accuracy'])
```

```
model.fit(X_treino, y_treino, batch_size=1, epochs= 100,
            validation_data = (X_val,y_val), verbose=1)
```

4.3.4.1.2 LSTM

Para os algoritmos *Long Short-Term Memory*(LSTM) fizemos três divisões dos dados para calcular a média e desvio padrão das métricas accuracy, precision, recall e f1-score.

A divisão dos dados foram mais ou menos a proporção 80% para dados de treino e 20% e para dados teste. Isso se deve pela temporalidade dos dados, não podemos separar de qualquer maneira, pois uma separação sem olhar a quantidade de semestre, pode por dado de uma aluno junto com outro aluno. Não separamos os dados de treino em treino e validação, isso se deve pela dificuldade de separar os datsets com temporalidade.

Divisão dos dados

```
#d=0, dividir os dados em treino [0,80%] e teste [80%,100%]
#d=1, dividir os dados em treino [20%,100%] e teste [0,20%]
#d=2, dividir os dados em treino [10%,90%] e teste [0,10%]U
   [90%,100%]
```

Para o treinamento do LSTM utilizamos o modelo:

```
model = Sequential()
model.add(LSTM(10, input_shape=(X_train.shape[1], X_train.shape[2])
   ,
          return_sequences=True))
model.add(LSTM(5, return_sequences=False))
model.add(Dense(8))
# Compile the model
model.compile(loss='mean_absolute_error', optimizer='Adam',
              metrics=['acc'])
historico = model.fit(X_train, Y_train, validation_data = (X_test,
   Y_test),
 epochs=100, batch_size=1, verbose=1)
```

4.3.5 Balanceamento dos Dados

Como o conjunto de dados com mais de 50% das entradas pertencendo a uma única classe são considerados desbalanceados, temos que os nossos datasets separados por alunos cursados até o i-ésimo período são todos desbalanceados.

Tabela 11 – Quantidade de alunos formandos e desistentes(evadidos) por semestre

datasets	Todos	Formandos	Evadidos/Desistentes
1S	450	59	391
2S	250	59	191
3S	186	59	127
4S	148	59	89
5S	120	59	61
6S	100	59	41
7S	94	58	36
8S	78	56	22

Fonte:Próprio Autor

Para resolver esse problema, podemos utilizar as técnicas de amostragem *UnderSampler* e *OverSampler* para lidar com o desequilíbrio de classes.

Para aplicar a técnica de *undersampling* e *oversampling*, utilizaremos o algoritmo *RandomUnderSampler* e *RandomOverSampler*, respectivamente. O *RandomUnderSampler* remove instâncias de maneira aleatória e *RandomOverSampler* cria dados sintéticos. O mesmo está disponível na biblioteca *imbalanced-learn*.

```
# importar as bibliotecas necess rias
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import RandomOverSampler
```

O método *RandomUnderSampler* é um método de sub-amostragem de dados usado para tratar datasets desequilibrados. O objetivo é diminuir a quantidade de amostras da classe majoritária até que ela seja equivalente à classe minoritária

Aplicamos a técnica de *RandomUnderSampling* aos **dados de treino** separados pelo técnica *Holdout* no conjunto de dados utilizando a função *train_test_split*.

```
from sklearn.model_selection import train_test_split
X_treino, X_teste, y_treino, y_teste = train_test_split(
                                       previsores, alvo,
                                       stratify=alvo, random_state
                                          =0,



desempenho_cross_val = cross_val_score(estimator = classifier,X =
   X_treino,
                                       y = y_treino,scoring = '
                                          roc_auc',cv = 5)
```

A função *fit_resample* ajusta o objeto *RandomUnderSampler* aos dados os retorna balanceados.

```
  # criando uma inst ncia do RandomUnderSampling
    rus = RandomUnderSampler(random_state=42,sampling_strategy = '
       majority')
    # balanceando os dados
    X_treino, y_treino = rus.fit_resample(X_treino, y_treino)
```

O método *RandomOverSampler* é um método de sobre-amostragem de dados usado para tratar *datasets* desequilibrados. O objetivo é aumentar a quantidade de amostras da classe minoritária até que ela seja equivalente à classe majoritária

Aplicamos a técnica de *RandomUnderSampling* aos **dados de treino** separados pelo técnica *Holdout* no conjunto de dados utilizando a função *train_test_split*.

```
from sklearn.model_selection import train_test_split
X_treino, X_teste, y_treino, y_teste = train_test_split(
                                       previsores, alvo,
                                       stratify=alvo, random_state
                                          =0,



desempenho_cross_val = cross_val_score(estimator = classifier,X =
   X_treino,
                                       y = y_treino,scoring = '
                                          roc_auc',cv = 5)
```

```
# criando uma inst ncia do RandomUnderSampling
rus = RandomUnderSampler(random_state=42, sampling_strategy = '
   majority')
```

A função *fit_resample* ajusta o objeto *RandomUnderSampler* aos dados os retorna balanceados.

```
 # criando uma inst ncia do RandomOverSampling
    oversample = RandomOverSampler(random_state=42,
       sampling_strategy='minority')
```

```
3     # balanceando os dados
4     X_treino, y_treino = oversample.fit_resample(X_treino, y_treino
          )
```

4.4 SISTEMA DE PREVISÃO DE EVASÃO

Queremos prever se um aluno do curso de licenciatura em Matemática da UFTM irá evadir (desistir) do curso em algum j-ésimo semestre, onde $1° \leq j \leq 8°$. Teremos 8 modelos e o k-ésimo modelo, $1 \leq k \leq 8$ é utilizado para prever se um aluno que entrou no curso há k semestres irá no futuro, se formar ou evadir do curso. Os bancos de dados de treinamento e teste (BTTs) para o k-ésimo modelo são obtidos pelas informações dos atributos listados na tabela 12 de todos os alunos que já tenham formados ou evadidos no período de 2014 até 2022.

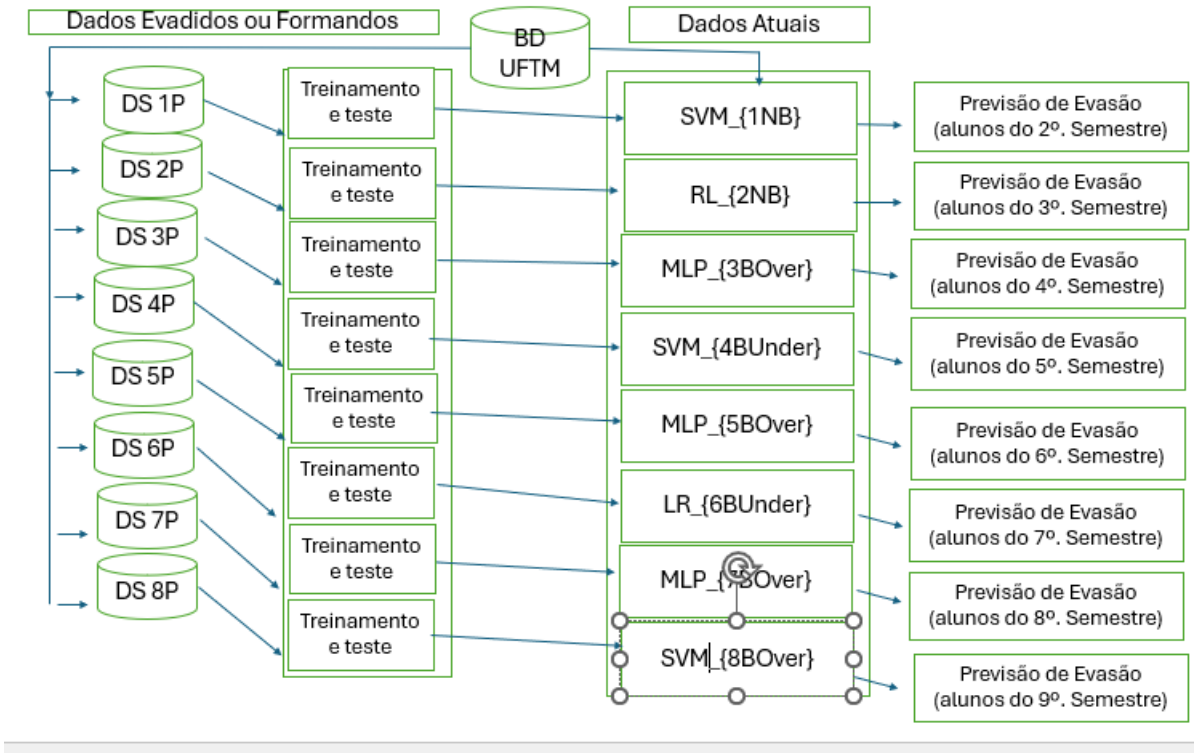
Tabela 12 – Atributos

DESCRIÇÃO DOS DADOS E ATRIBUTOS RECOLHIDOS DO SISCAD			
Atributo	Descrição	Tipo de Dado	Selecionado
QtDisAprovadasSemestre	Quantidade de disciplinas Aprovadas por semestre	inteiro	Sim
QtDisReprovadasSemestre	Quantidade de disciplinas Reprovadas por semestre	inteiro	Sim
QtDisReprovadasSemestrePorNota	Quantidade de disciplinas reprovadas por nota por semestre	inteiro	Sim
QtDisReprovadasSemestrePorFalta	Quantidade de disciplinas reprovadas por falta por semestre	inteiro	Sim
PorcMediaFaltasSemestre	Porcentagem Média de Faltas por Semestre	Número real (float)	Sim
NotaMediaSemestre	Nota média por semestre	Número real (float)	Sim
QtDsciplinasSemestre	Quantidade de disciplinas matriculadas por Semestre	inteiro	Sim
ChDisciplinasSemestre	Carga Horária no semestre	inteiro	Sim
TARGET	Informações se o aluno evadiu ou não	inteiro	Sim

Fonte: Próprio Autor.

A figura 50 representa os processos de geração, teste e utilização do Sistema de Previsão de Evasões. O banco de dados de treinamento e teste (BTTs).

Figura 50 – Sistema de Previsão de Evasões com 8 modelos



Fonte: Proprio Autor

5 PREVISÃO DE EVASÃO NO CURSO DE LICENCIATURA EM MATEMÁTICA DA UFTM

A programação dos modelos abaixo foram feitas no Colab Python, veja:

RL, SVC, KNN e DT Link para o Google Colab
MLP Link para o Google Colab
LSTM Link para o Google Colab

5.1 MODELAGEM E ESCOLHA DO MODELO

Foram selecionadas alguns algoritmos de aprendizagem de máquina: Regressão Logística, Máquina de Vetores de Suporte, k-Vizinhos Mais Próximos-KNN, Árvores de Decisão, Redes Neurais Simples e Recorrentes do tipo *long short-therm memory (LSTM)*. Os 4 primeiros pela simplicidade e rapidez de executar, MLP por achar que seria interessante e LSTM uma aposta nos dados temporais.

Separamos em quatro etapas:

- Primeira etapa: Rodar os algoritmos Regressão Logística, Máquina de Vetores de Suporte, k-Vizinhos Mais Próximos-KNN, Árvores de Decisão, redes neurais MLP e LSTM nos dados de treino e validação.
- Segunda etapa: Melhor desempenho do recall para a classe 0 (desistentes/evadidos) em cada dataset balenceado ou não.
- Terceira etapa: Tunar o melhor modelo
- Quarta e final etapa: Aplicar o modelo tunado nos dados de teste.

5.1.1 Regressão Logística, Máquina de Vetores de Suporte, k-Vizinhos Mais Próximos-KNN, Árvores de Decisão, MLP e LSTM

A seguir para cada modelo será apresentado o processo de tratamento dos modelos aplicando um **for** sem **random-state** e criando divisões de dados treino e teste

5.1.1.1 Regressão Logística, Máquina de Vetores de Suporte, k-Vizinhos Mais Próximos-KNN e Árvores de Decisão

Para os algoritmos Regressão Logística, Máquina de Vetores de Suporte, k-Vizinhos Mais Próximos-KNN, Árvores de Decisão aplicamos um **for** para tirar a média e desvio padrão. Tiramos do train-test-split o ramdom-state. Aplicamos 30 testes utilizando a proporção 85% para dados de treino e validação e 15% e para dados teste. Dos 85% para dados de treino e validação, separamos 80% para dados de treino e validação e 20%, dando uma porcentagem dos dados originais de 68% para dados de treino, 17% para validação e 15% e para dados teste.

Divisão dos dados

```
features = df.columns.drop('TARGET')
features
previsores = df[features]
alvo = df.drop(features, axis=1)['TARGET']
for i in range(0,10):
# Dividir em dados de treino/valida  o e teste
X_treino_val, X_teste, y_treino_val, y_teste = train_test_split(
    previsores, alvo, train_size=0.85)
# Dividir novamente os dados de treino/valida  o em treino e
    valida  o
X_treino, X_val, y_treino, y_val = train_test_split(X_treino_val,
    y_treino_val, test_size=0.2)
```

Modelos utilizados

```
LogisticRegression()
SVC()
KNeighborsClassifier()
DecisionTreeClassifier()
```

5.1.1.2 Multilayer Perceptron-MLP

Para os algoritmos Multilayer Perceptron-MLP também aplicamos um **for** para tirar a média e desvio padrão e tiramos do *train-test-split* o *random-state*, porém fizemos um for com 10 iterações. Justificativa foi pela demora em rodar o colab. Aplicamos 5 testes utilizando a proporção 80% para dados de treino e validação e 20% para dados de teste. Dos 80% para treino e validação, fica 75% para treino e 25% para validação. Com base nos resultados dos modelos aplicados nos conjuntos de dados para **testes**, tivemos os seguintes resultados com intervalos de confiança de 95%.

Divisão dos dados

```
features = df.columns.drop('TARGET')
features
previsores = df[features]
alvo = df.drop(features, axis=1)['TARGET']
for i in range(0,10):
# Dividir em dados de treino/valida  o e teste
X_treino_val, X_teste, y_treino_val, y_teste =
              train_test_split(previsores, alvo, train_size=0.80)
# Dividir novamente os dados de treino/valida  o em treino e
   valida  o
X_treino, X_val, y_treino, y_val =
             train_test_split(X_treino_val, y_treino_val, test_size
                =0.2)
```

Modelos utilizados

```
model = Sequential()
model.add(Dense(100, input_dim=n_features,kernel_initializer='
   normal',
        activation='relu'))
model.add(Dense(2, kernel_initializer='normal',activation='softmax'
   ))
model.summary()
model.compile(optimizer=SGD(), loss='categorical_crossentropy',
   metrics=['accuracy'])
model.fit(X_treino, y_treino, batch_size=1, epochs= 100,
           validation_data = (X_val,y_val), verbose=1)
```

5.1.1.3 LSTM

Para os algoritmos *Long Short-Term Memory*(LSTM) fizemos três divisões dos dados para calcular a média e desvio padrão das métricas accuracy, precision, recall e f1-score.

A divisão dos dados foram mais ou menos a proporção 80% para dados de treino e 20% e para dados teste. Isso se deve pela temporalidade dos dados, não podemos separar de qualquer maneira, pois uma separação sem olhar a quantidade de semestre, pode por dado de uma aluno junto com outro aluno. Não separamos os dados de treino em treino e validação, isso se deve pela dificuldade de separar os datsets com temporalidade.

Divisão dos dados

```
#d=0, dividir os dados em treino [0,80%] e teste [80%,100%]
#d=1, dividir os dados em treino [20%,100%] e teste [0,20%]
```

```
#d=2, dividir os dados em treino [10%,90%] e teste [0,10%]U
    [90%,100%]
```

Para o treinamento do LSTM utilizamos o modelo:

```
model = Sequential()
model.add(LSTM(10, input_shape=(X_train.shape[1], X_train.shape[2])
    ,
          return_sequences=True))
model.add(LSTM(5, return_sequences=False))
model.add(Dense(8))
# Compile the model
model.compile(loss='mean_absolute_error', optimizer='Adam',
              metrics=['acc'])
historico = model.fit(X_train, Y_train, validation_data = (X_test,
    Y_test),
 epochs=100, batch_size=1, verbose=1)
```

5.1.1.4 Conjunto de dados 1P -Treino e Validação

Dados de alunos que cursaram pelo menos o **primeiro período**: 450 alunos ⇒ 59 concluintes e 391 desistentes.

Dados de Treinamento e validação Não Balanceados -NB: 360 alunos, 46 concluintes e 314 desistentes;

Dados de Treinamento Não Balanceados -NB: 288 alunos, 35 concluintes e 253 desistentes.

Dados de validação Não Balanceados -NB: 72 alunos, 11 concluintes e 61 desistentes.

Tabela 13 – RL, SVM, KNN e DT aplicados nos dados de validação não balanceados

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.55, 0.65)	(0.84, 0.89)	(0.88, 0.92)	(0.93, 0.97)	(0.91, 0.94)
1	LR	(0.55, 0.65)	(0.84, 0.89)	(0.33, 0.57)	(0.16, 0.34)	(0.21, 0.4)
0	SVM	(0.5, 0.51)	(0.82, 0.87)	(0.82, 0.87)	**(1.0, 1.0)**	(0.9, 0.93)
1	SVM	(0.5, 0.51)	(0.82, 0.87)	(0.0, 0.31)	(0.0, 0.02)	(0.0, 0.04)
0	KNN	(0.6, 0.69)	(0.83, 0.88)	(0.9, 0.92)	(0.9, 0.95)	(0.9, 0.93)
1	KNN	(0.6, 0.69)	(0.83, 0.88)	(0.31, 0.59)	(0.29, 0.44)	(0.3, 0.49)
0	DT	(0.6, 0.73)	(0.83, 0.89)	(0.89, 0.94)	(0.91, 0.95)	(0.9, 0.94)
1	DT	(0.6, 0.73)	(0.83, 0.89)	(0.31, 0.59)	(0.29, 0.52)	(0.3, 0.54)
0	MLP	(0.57, 0.71)	(0.9, 0.93)	(0.9, 0.93)	(0.84, 0.88)	(0.87, 0.96)
1	MLP	(0.57, 0.71)	(0.22, 0.47)	(0.23, 0.54)	(0.84, 0.88)	(0.19, 0.54)

Fonte: Próprio Autor.

Dados de Treinamento Balanceados -Under :70 alunos, 35 concluintes e 35 desistentes.

Dados de validação Não Balanceados -NB: 70 alunos, 35 concluintes e 35 desistentes.

Tabela 14 – RL, SVM, KNN, DT eMLP aplicados nos dados de validação balanceados Under

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.79, 0.86)	(0.71, 0.79)	(0.97, 1.0)	(0.67, 0.77)	(0.8, 0.86)
1	LR	(0.79, 0.86)	(0.71, 0.79)	(0.3, 0.41)	(0.87, 0.99)	(0.45, 0.56)
0	SVM	(0.81, 0.88)	(0.78, 0.84)	(0.97, 0.99)	(0.77, 0.83)	(0.86, 0.9)
1	SVM	(0.81, 0.88)	(0.78, 0.84)	(0.3, 0.46)	(0.83, 0.95)	(0.45, 0.6)
0	KNN	(0.78, 0.86)	(0.72, 0.81)	(0.96, 0.99)	(0.69, 0.79)	(0.8, 0.88)
1	KNN	(0.78, 0.86)	(0.72, 0.81)	(0.31, 0.44)	(0.86, 0.95)	(0.46, 0.6)
0	DT	(0.7, 0.81)	(0.72, 0.81)	(0.92, 0.97)	(0.72, 0.83)	(0.81, 0.89)
1	DT	(0.7, 0.81)	(0.72, 0.81)	(0.28, 0.37)	(0.62, 0.85)	(0.4, 0.48)
0	MLP	(0.8, 0.88)	(0.87, 0.91)	(0.95, 0.99)	(0.79, 0.85)	(0.78, 0.85)
1	MLP	(0.8, 0.88)	(0.5, 0.6)	(0.36, 0.48)	(0.79, 0.85)	(0.78, 0.94)

Fonte: Próprio Autor.

Dados de Treinamento Balanceados -Over : 506 alunos, 253 concluintes e 253 desistentes.

Dados de validação Não Balanceados -NB: 70 alunos, 35 concluintes e 35 desistentes.

Tabela 15 – RL, SVM, KNN e DT aplicados nos dados de validação balanceados Over

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.8, 0.87)	(0.77, 0.83)	(0.97, 0.99)	(0.76, 0.83)	(0.85, 0.9)
1	LR	(0.8, 0.87)	(0.77, 0.83)	(0.36, 0.45)	(0.82, 0.93)	(0.5, 0.6)
0	SVM	(0.82, 0.89)	(0.8, 0.85)	(0.97, 0.99)	(0.78, 0.84)	(0.87, 0.9)
1	SVM	(0.82, 0.89)	(0.8, 0.85)	(0.33, 0.5)	(0.83, 0.98)	(0.48, 0.64)
0	KNN	(0.68, 0.79)	(0.76, 0.82)	(0.91, 0.96)	(0.77, 0.85)	(0.85, 0.89)
1	KNN	(0.68, 0.79)	(0.76, 0.82)	(0.28, 0.45)	(0.54, 0.77)	(0.38, 0.54)
0	DT	(0.59, 0.73)	(0.8, 0.85)	(0.91, 0.94	(0.85, 0.9)	(0.88, 0.92)
1	DT	(0.59, 0.73)	(0.8, 0.85)	(0.21, 0.41)	(0.3, 0.57)	(0.25, 0.46)
0	MLP	(0.82, 0.9)	(0.89, 0.91)	(0.97, 1.0)	(0.83, 0.86)	(0.81, 0.85)
1	MLP	(0.82, 0.9)	(0.44, 0.66)	(0.3, 0.51)	(0.83, 0.86)	(0.8, 0.98)

Fonte: Próprio Autor.

Tabela 16 – Intervalo de confiança de $LSTM_{1NB}$

Modelo	target	accuracy e std	precision e std	recall e std	f1score e std
LSTM	0	(0.65, 0.92)	(0.12, 1.0)	(0.13, 1.0)	(0.12, 1.0)
LSTM	1	(0.65, 0.92)	(0.08, 0.74)	(0.0, 0.89)	(0.01, 0.72)

Fonte: Próprio Autor.

Tabela 17 – Melhor de 1P

Target/Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0 SVM_{1NB}	(0.5, 0.51)	(0.82, 0.87)	(0.82, 0.87)	**(1.0, 1.0)**	(0.9, 0.93)
1 SVM_{1NB}	(0.5, 0.51)	(0.82, 0.87)	(0.0, 0.31)	(0.0, 0.02)	(0.0, 0.04)
0 $MLP_{1BUnder}$	(0.8, 0.88)	(0.87, 0.91)	(0.95, 0.99)	(0.79, 0.85)	(0.78, 0.85)
1 $MLP_{1BUnder}$	(0.8, 0.88)	(0.5, 0.6)	(0.36, 0.48)	(0.79, 0.85)	(0.78, 0.94)
0 MLP_{1BOver}	(0.82, 0.9)	(0.89, 0.91)	(0.97, 1.0)	(0.83, 0.86)	(0.81, 0.85)
1 MLP_{1BOver}	(0.82, 0.9)	(0.44, 0.66)	(0.3, 0.51)	(0.83, 0.86)	(0.8, 0.98)

Fonte: Próprio Autor.

5.1.1.5 Conjunto de dados 2P -Treino e Validação

Dados de alunos que cursaram pelo menos o **primeiro período**: 200 alunos ⇒ 46 concluintes e 154 desistentes.

Dados de Treinamento e validação Não Balanceados -NB: 200 alunos, 46 concluintes e 154 desistentes;

Dados de Treinamento Não Balanceados -NB: 160 alunos, 36 concluintes e 124 desistentes;

Dados de validação Não Balanceados -NB: 40 alunos, 10 concluintes e 30 desistentes;

Tabela 18 – RL, SVM, KNN e DT aplicados nos dados de validação não balanceados

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.76, 0.81)	(0.89, 0.92)	(0.88, 0.92)	(0.84, 0.87)	(0.9, 0.93)
1	LR	(0.76, 0.81)	(0.62, 0.71)	(0.66, 0.76)	(0.84, 0.87)	(0.6, 0.71)
0	SVM	(0.72, 0.77)	(0.86, 0.89)	(0.84, 0.88)	(0.79, 0.83)	(0.87, 0.91)
1	SVM	(0.72, 0.77)	(0.57, 0.66)	(0.61, 0.71)	(0.79, 0.83)	(0.54, 0.65)
0	KNN	(0.69, 0.76)	(0.86, 0.89)	(0.85, 0.89)	(0.79, 0.83)	(0.86, 0.9)
1	KNN	(0.69, 0.76)	(0.51, 0.62)	(0.53, 0.64)	(0.79, 0.83)	(0.51, 0.64)
0	DT	(0.65, 0.7)	(0.82, 0.86)	(0.82, 0.87)	(0.73, 0.78)	(0.81, 0.87)
1	DT	(0.65, 0.7)	(0.45, 0.53)	(0.46, 0.58)	(0.73, 0.78)	(0.45, 0.56)
0	MLP	(0.69, 0.75)	(0.52, 0.61)	(0.5, 0.64)	(0.77, 0.81)	(0.5, 0.66)
1	MLP	(0.69, 0.75)	(0.52, 0.61)	(0.5, 0.64)	(0.77, 0.81)	(0.5, 0.66)

Fonte: Próprio Autor.

Dados de Treinamento Balanceados -Under :72 alunos, 36 concluintes e 36 desistentes;

Dados de validação Não Balanceados -NB: 40 alunos, 10 concluintes e 30 desistentes; e

Tabela 19 – RL, SVM, KNN e DT aplicados nos dados de validação balanceados Under

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.81, 0.86)	(0.82, 0.86)	(0.97, 0.99)	(0.76, 0.81)	(0.71, 0.78)
1	LR	(0.81, 0.86)	(0.6, 0.7)	(0.46, 0.57)	(0.76, 0.81)	(0.89, 0.97)
0	SVM	(0.79, 0.83)	(0.81, 0.84)	(0.94, 0.96)	(0.75, 0.79)	(0.71, 0.75)
1	SVM	(0.79, 0.83)	(0.62, 0.69)	(0.49, 0.56)	(0.75, 0.79)	(0.86, 0.92)
0	KNN	(0.79, 0.83)	(0.81, 0.84)	(0.94, 0.97)	(0.74, 0.78)	(0.7, 0.75)
1	KNN	(0.79, 0.83)	(0.59, 0.66)	(0.45, 0.53)	(0.74, 0.78)	(0.85, 0.92)
0	DT	(0.7, 0.76)	(0.77, 0.82)	(0.88, 0.92)	(0.7, 0.75)	(0.68, 0.75)
1	DT	(0.7, 0.76)	(0.53, 0.61)	(0.42, 0.51)	(0.7, 0.75)	(0.7, 0.79)
0	MLP	(0.81, 0.86)	(0.85, 0.87)	(0.96, 0.98)	(0.79, 0.82)	(0.76, 0.78)
1	MLP	(0.81, 0.86)	(0.58, 0.75)	(0.45, 0.62)	(0.79, 0.82)	(0.85, 0.94)

Fonte: Próprio Autor.

Dados de Treinamento Balanceados -Over : 248 alunos, 124 concluintes e 124 desistentes.

Dados de validação Não Balanceados -NB: 40 alunos, 10 concluintes e 30 desistentes.

Tabela 20 – RL, SVM, KNN e DT aplicados nos dados de validação balanceados Over

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.78, 0.83)	(0.81, 0.85)	(0.94, 0.97)	(0.75, 0.79)	(0.71, 0.76)
1	LR	(0.78, 0.83)	(0.59, 0.67)	(0.46, 0.55)	(0.75, 0.79)	(0.84, 0.91)
0	SVM	(0.8, 0.86)	(0.83, 0.87)	(0.94, 0.97)	(0.78, 0.82)	(0.75, 0.79)
1	SVM	(0.8, 0.86)	(0.64, 0.71)	(0.51, 0.59)	(0.78, 0.82)	(0.85, 0.93)
0	KNN	(0.76, 0.81)	(0.79, 0.84)	(0.92, 0.95)	(0.73, 0.78)	(0.7, 0.76)
1	KNN	(0.76, 0.81)	(0.59, 0.65)	(0.47, 0.53)	(0.73, 0.78)	(0.8, 0.88)
0	DT	(0.64, 0.69)	(0.84, 0.86)	(0.83, 0.87)	(0.75, 0.79)	(0.83, 0.87)
1	DT	(0.64, 0.69)	(0.41, 0.5)	(0.41, 0.53)	(0.75, 0.79)	(0.43, 0.52)
0	MLP	(0.72, 0.82)	(0.82, 0.88)	(0.93, 0.95)	(0.73, 0.82)	(0.73, 0.83)
1	MLP	(0.72, 0.82)	(0.44, 0.64)	(0.31, 0.55)	(0.73, 0.82)	(0.7, 0.82)

Fonte: Próprio Autor.

Tabela 21 – Intervalo de confiança de $LSTM_{2NB}$

Modelo	target	accuracy e std	precision e std	recall e std	f1score e std
LSTM	0	(0.86, 0.9)	(0.95, 0.96)	(0.86, 0.90)	(0.91, 092)
LSTM	1	(0.86, 0.9)	(0.62, 0.80)	(0.86, 0.90)	(0.73, 0.84)

Fonte: Próprio Autor.

Tabela 22 – Melhor de 2P

Target/ Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0 LR_{2NB}	(0.76, 0.81)	(0.89, 0.92)	(0.88, 0.92)	**(0.84, 0.87)**	(0.9, 0.93)
1 LR_{2NB}	(0.76, 0.81)	(0.62, 0.71)	(0.66, 0.76)	(0.84, 0.87)	(0.6, 0.71)
0 $MLP_{2BUnder}$	(0.81, 0.86)	(0.85, 0.87)	(0.96, 0.98)	(0.79, 0.82)	(0.76, 0.78)
1 $MLP_{2BUnder}$	(0.81, 0.86)	(0.58, 0.75)	(0.45, 0.62)	(0.79, 0.82)	(0.85, 0.94)
0 SVM_{2BOver}	(0.8, 0.86)	(0.83, 0.87)	(0.94, 0.97)	(0.78, 0.82)	(0.75, 0.79)
1 SVM_{2BOver}	(0.8, 0.86)	(0.64, 0.71)	(0.51, 0.59)	(0.78, 0.82)	(0.85, 0.93)

Fonte: Próprio Autor.

5.1.1.6 Conjunto de dados 3P -Treino e Validação

Dados de alunos que cursaram pelo menos o **primeiro período**: 186 alunos ⇒ 59 concluintes e 127 desistentes.

Dados de Treinamento e validação Não Balanceados -NB: 200 alunos, 46 concluintes e 154 desistentes;

Dados de Treinamento Não Balanceados -NB: 160 alunos, 36 concluintes e 124 desistentes;

Dados de validação Não Balanceados -NB: 40 alunos, 10 concluintes e 30 desistentes;

Tabela 23 – RL, SVM, KNN e DT aplicados nos dados de validação não balanceados

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.8, 0.89)	(0.83, 0.9)	(0.88, 0.95)	(0.85, 0.93)	(0.87, 0.92)
1	LR	(0.8, 0.89)	(0.83, 0.9)	(0.69, 0.85)	(0.72, 0.89)	(0.71, 0.85)
0	SVM	(0.82, 0.91)	(0.8, 0.89)	(0.9, 1.0)	(0.79, 0.87)	(0.85, 0.92)
1	SVM	(0.82, 0.91)	(0.8, 0.89)	(0.59, 0.77)	(0.83, 0.98)	(0.7, 0.84)
0	KNN	(0.8, 0.87)	(0.78, 0.89)	(0.88, 0.95)	(0.76, 0.94)	(0.83, 0.92)
1	KNN	(0.8, 0.87)	(0.78, 0.89)	(0.59, 0.85)	(0.74, 0.89)	(0.67, 0.81)
0	DT	(0.73, 0.84)	(0.75, 0.84)	(0.8, 0.91)	(0.77, 0.91)	(0.8, 0.89)
1	DT	(0.73, 0.84)	(0.75, 0.84)	(0.56, 0.79)	(0.63, 0.82)	(0.61, 0.75)
0	MLP	(0.71, 0.85)	(0.84, 0.94)	(0.8, 0.91)	(0.77, 0.91)	(0.88, 0.99)
1	MLP	(0.71, 0.85)	(0.6, 0.8)	(0.68, 0.95)	(0.77, 0.91)	(0.52, 0.73)

Fonte: Próprio Autor.

Dados de Treinamento Balanceados Under -BUnder: 72 alunos, 36 concluintes e 36 desistentes;

Dados de validação Balanceados Under -BUnder: 40 alunos, 10 concluintes e 30 desistentes;

Tabela 24 – RL, SVM, KNN e DT aplicados nos dados de validação balanceados Under

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.83, 0.91)	(0.8, 0.88)	(0.94, 1.0)	(0.74, 0.84)	(0.83, 0.9)
1	LR	(0.83, 0.91)	(0.8, 0.88)	(0.57, 0.74)	(0.89, 1.0)	(0.69, 0.84)
0	SVM	(0.83, 0.9)	(0.83, 0.89)	(0.91, 0.97)	(0.82, 0.9)	(0.88, 0.92)
1	SVM	(0.83, 0.9)	(0.83, 0.89)	(0.65, 0.82)	(0.8, 0.93)	(0.74, 0.84)
0	KNN	(0.76, 0.85)	(0.76, 0.85)	(0.85, 0.94)	(0.69, 0.87)	(0.76, 0.89)
1	KNN	(0.76, 0.85)	(0.76, 0.85)	(0.59, 0.78)	(0.75, 0.89)	(0.67, 0.8)
0	DT	(0.67, 0.8)	(0.67, 0.8)	(0.8, 0.9)	(0.66, 0.83)	(0.73, 0.84)
1	DT	(0.67, 0.8)	(0.67, 0.8)	(0.47, 0.72)	(0.61, 0.84)	(0.54, 0.75)
0	MLP	(0.78, 0.87)	(0.82, 0.88)	(0.88, 0.96)	(0.78, 0.86)	(0.75, 0.83)
1	MLP	(0.78, 0.87)	(0.71, 0.82)	(0.64, 0.75)	(0.78, 0.86)	(0.79, 0.94)

Fonte: Próprio Autor.

Dados de Treinamento Balanceados Over -BOver: 248 alunos, 124concluintes e 124 desistentes;

Dados de validação Balanceados Over -BOver: 40 alunos, 10 concluintes e 30 desistentes;

Tabela 25 – RL, SVM, KNN e DT aplicados nos dados de validação balanceados Over

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.83, 0.89)	(0.82, 0.88)	(0.91, 0.99)	(0.76, 0.86)	(0.84, 0.9)
1	LR	(0.83, 0.89)	(0.82, 0.88)	(0.66, 0.8)	(0.85, 0.98)	(0.76, 0.85)
0	SVM	(0.83, 0.9)	(0.79, 0.87)	(0.95, 1.0)	(0.73, 0.83)	(0.83, 0.9)
1	SVM	(0.83, 0.9)	(0.79, 0.87)	(0.58, 0.71)	(0.92, 1.0)	(0.72, 0.81)
0	KNN	(0.82, 0.86)	(0.81, 0.86)	(0.91, 0.97)	(0.76, 0.88)	(0.85, 0.9)
1	KNN	(0.82, 0.86)	(0.81, 0.86)	(0.58, 0.76)	(0.79, 0.93)	(0.7, 0.79)
0	DT	(0.73, 0.84)	(0.76, 0.87)	(0.81, 0.91)	(0.82, 0.91)	(0.82, 0.91)
1	DT	(0.73, 0.84)	(0.76, 0.87)	(0.63, 0.8)	(0.62, 0.78)	(0.63, 0.77)
0	MLP	(0.86, 0.94)	(0.87, 0.94)	(0.91, 0.98)	(0.86, 0.93)	(0.85, 0.92)
1	MLP	(0.86, 0.94)	(0.82, 0.9)	(0.77, 0.87)	(0.86, 0.93)	(0.85, 0.97)

Fonte: Próprio Autor.

Tabela 26 – Intervalo de confiança de $LSTM_{3NB}$

Modelo	target	accuracy e std	precision e std	recall e std	f1score e std
LSTM	0	(0.81, 0.90)	(0.88, 0.90)	(0.86, 0.95)	(0.87, 0.93)
LSTM	1	(0.81, 0.90)	(0.68, 0.86)	(0.63, 0.79)	(0.68, 0.82)

Fonte: Próprio Autor.

Tabela 27 – Melhor de 3P

Target/Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0 LR_{3NB}	(0.8, 0.89)	(0.83, 0.9)	(0.88, 0.95)	(0.85, 0.93)	(0.87, 0.92)
1 LR_{3NB}	(0.8, 0.89)	(0.83, 0.9)	(0.69, 0.85)	(0.72, 0.89)	(0.71, 0.85)
0 $SVM_{3BUnder}$	(0.83, 0.9)	(0.83, 0.89)	(0.91, 0.97)	(0.82, 0.9)	(0.88, 0.92)
1 $SVM_{3BUnder}$	(0.83, 0.9)	(0.83, 0.89)	(0.65, 0.82)	(0.8, 0.93)	(0.74, 0.84)
0 MLP_{3BOver}	(0.86, 0.94)	(0.87, 0.94)	(0.91, 0.98)	**(0.86, 0.93)**	(0.85, 0.92)
1 MLP_{3BOver}	(0.86, 0.94)	(0.82, 0.9)	(0.77, 0.87)	(0.86, 0.93)	(0.85, 0.97)

Fonte: Próprio Autor.

5.1.1.7 Conjundo de dados 4P -Treino e Validação

Dados de alunos que cursaram pelo menos o **quarto período**: 148 alunos, 59 formandos e 89 evadidos.

Dados de Treinamento Não Balanceados -NB: 118 alunos, 45 concluintes e 73 desistentes;

Dados de Treinamento Não Balanceados -NB: 94 alunos, 34 concluintes e 60 desistentes;

Dados de validação Não Balanceados -NB: 24 alunos, 11 concluintes e 13 desistentes;

Tabela 28 – RL, SVM, KNN e DT aplicados nos dados de validação Não balanceados

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.85, 0.9)	(0.87, 0.92)	(0.91, 0.95)	(0.85, 0.9)	(0.83, 0.9)
1	LR	(0.85, 0.9)	(0.8, 0.87)	(0.76, 0.85)	(0.85, 0.9)	(0.85, 0.92)
0	SVM	(0.83, 0.89)	(0.85, 0.9)	(0.86, 0.92)	(0.84, 0.88)	(0.83, 0.89)
1	SVM	(0.83, 0.89)	(0.8, 0.87)	(0.78, 0.85)	(0.84, 0.88)	(0.82, 0.9)
0	KNN	(0.83, 0.88)	(0.86, 0.9)	(0.86, 0.92)	(0.83, 0.87)	(0.84, 0.9)
1	KNN	(0.83, 0.88)	(0.78, 0.84)	(0.76, 0.85)	(0.83, 0.87)	(0.8, 0.88)
0	DT	(0.83, 0.88)	(0.81, 0.86)	(0.83, 0.89)	(0.79, 0.83)	(0.78, 0.85)
1	DT	(0.78, 0.84)	(0.74, 0.81)	(0.71, 0.81)	(0.79, 0.83)	(0.76, 0.85)
0	MLP	(0.79, 0.9)	(0.83, 0.95)	(0.87, 0.95)	(0.8, 0.93)	(0.79, 0.96)
1	MLP	(0.79, 0.9)	(0.75, 0.89)	(0.73, 0.95)	(0.8, 0.93)	(0.74, 0.9)

Fonte: Próprio Autor.

Dados de Treinamento Balanceados Under -BUnder: 68 alunos, 34 concluintes e 34 desistentes;

Dados de validação Não Balanceados -NB: 24 alunos, 11 concluintes e 13 desistentes;

Tabela 29 – RL, SVM, KNN e DT aplicados nos dados de validação balanceados Under

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.84, 0.88)	(0.84, 0.88)	(0.88, 0.94)	(0.83, 0.87)	(0.79, 0.86)
1	LR	(0.84, 0.88)	(0.8, 0.85)	(0.74, 0.83)	(0.83, 0.87)	(0.86, 0.92)
0	SVM	(0.86, 0.91)	(0.87, 0.92)	(0.9, 0.94)	(0.86, 0.91)	(0.85, 0.91)
1	SVM	(0.86, 0.91)	(0.83, 0.89)	(0.8, 0.87)	(0.86, 0.91)	(0.87, 0.92)
0	KNN	(0.81, 0.86)	(0.82, 0.87)	(0.85, 0.92)	(0.81, 0.85)	(0.79, 0.85)
1	KNN	(0.81, 0.86)	(0.77, 0.83)	(0.72, 0.8)	(0.81, 0.85)	(0.81, 0.89)
0	DT	(0.8, 0.84)	(0.82, 0.87)	(0.83, 0.9)	(0.79, 0.84)	(0.79, 0.88)
1	DT	(0.79, 0.84)	(0.72, 0.79)	(0.7, 0.81)	(0.79, 0.84)	(0.76, 0.84)
1	MLP	(0.82, 0.89)	(0.8, 0.88)	(0.77, 0.94)	(0.81, 0.9)	(0.79, 0.9)

Fonte: Próprio Autor.

Dados de Treinamento Balanceados Over -BOver: 120 alunos, 60 concluintes e 60 desistentes;

Dados de validação Não Balanceados -NB: 25 alunos, 9 concluintes e 16 desistentes;

Tabela 30 – RL, SVM, KNN e DT aplicados nos dados de validação balanceados Over

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.85, 0.89)	(0.86, 0.89)	(0.92, 0.96)	(0.84, 0.87)	(0.8, 0.84)
1	LR	(0.85, 0.89)	(0.78, 0.84)	(0.69, 0.79)	(0.83, 0.87)	(0.88, 0.95)
0	SVM	(0.82, 0.88)	(0.83, 0.89)	(0.88, 0.95)	(0.82, 0.87)	(0.79, 0.86)
1	SVM	(0.82, 0.88)	(0.78, 0.85)	(0.74, 0.82)	(0.82, 0.87)	(0.83, 0.92)
0	KNN	(0.81, 0.86)	(0.82, 0.87)	(0.86, 0.92)	(0.81, 0.86)	(0.79, 0.85)
1	KNN	(0.81, 0.86)	(0.76, 0.83)	(0.71, 0.79)	(0.81, 0.86)	(0.82, 0.89)
0	DT	(0.81, 0.86)	(0.84, 0.88)	(0.84, 0.91)	(0.81, 0.85)	(0.82, 0.89)
1	DT	(0.81, 0.86)	(0.75, 0.81)	(0.72, 0.82)	(0.81, 0.85)	(0.77, 0.86)
0	MLP	(0.78, 0.85)	(0.82, 0.91)	(0.8, 0.89)	(0.82, 0.88)	(0.83, 0.95)
1	MLP	(0.78, 0.85)	(0.71, 0.84)	(0.74, 0.92)	(0.82, 0.88)	(0.66, 0.83)

Fonte: Próprio Autor.

Tabela 31 – Intervalo de confiança de $LSTM_{4NB}$

Modelo	target	accuracy e std	precision e std	recall e std	f1score e std
LSTM	0	(0.76, 0.9)	(0.81, 0.93)	(0.77, 0.91)	(0.78, 0.91)
LSTM	1	(0.76, 0.9)	(0.70, 0.83)	(0.76, 0.88)	(0.74, 0.86)

Fonte: Próprio Autor.

Tabela 32 – Melhor de 4P

Target/Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0 LR_{4NB}	(0.85, 0.9)	(0.87, 0.92)	(0.91, 0.95)	(0.85, 0.9)	(0.83, 0.9)
1 LR_{4NB}	(0.85, 0.9)	(0.8, 0.87)	(0.76, 0.85)	(0.85, 0.9)	(0.85, 0.92)
0 $SVM_{4BUnder}$	(0.86, 0.91)	(0.87, 0.92)	(0.9, 0.94)	**(0.86, 0.91)**	(0.85, 0.91)
1 $SVM_{4BUnder}$	(0.86, 0.91)	(0.83, 0.89)	(0.8, 0.87)	(0.86, 0.91)	(0.87, 0.92)
0 LR_{4BOver}	(0.85, 0.89)	(0.86, 0.89)	(0.92, 0.96)	(0.84, 0.87)	(0.8, 0.84)
1 LR_{4BOver}	(0.85, 0.89)	(0.78, 0.84)	(0.69, 0.79)	(0.83, 0.87)	(0.88, 0.95)

Fonte: Próprio Autor.

5.1.1.8 Conjunto de dados 5P -Treino e Validação

Dados de alunos que cursaram pelo menos o **quinto período**: 120 alunos, 59 formandos e 61 evadidos.

Dados de Treinamento e validação Não Balanceados -NB: 96 alunos, 46 concluintes e 50 desistentes;

Dados de Treinamento Não Balanceados -NB: 76 alunos, 36 concluintes e 40 desistentes; e ;

Dados de validação Não Balanceados -NB: 20 alunos, 10 concluintes e 10 desistentes;

Tabela 33 – RL, SVM, KNN e DT aplicados nos dados de validação Não balanceados

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.83, 0.92)	(0.83, 0.92)	(0.86, 0.97)	(0.79, 0.89)	(0.83, 0.92)
1	LR	(0.83, 0.92)	(0.83, 0.92)	(0.79, 0.9)	(0.85, 0.97)	(0.82, 0.93)
0	SVM	(0.85, 0.95)	(0.86, 0.95)	(0.85, 0.97)	(0.83, 0.96)	(0.85, 0.95)
1	SVM	(0.85, 0.95)	(0.86, 0.95)	(0.83, 0.96)	(0.84, 0.97)	(0.84, 0.95)
0	KNN	(0.79, 0.9)	(0.79, 0.9)	(0.78, 0.92)	(0.8, 0.92)	(0.8, 0.9)
1	KNN	(0.79, 0.9)	(0.79, 0.9)	(0.76, 0.91)	(0.77, 0.9)	(0.77, 0.89)
0	DT	(0.76, 0.86)	(0.73, 0.85)	(0.64, 0.85)	(0.74, 0.95)	(0.7, 0.84)
1	DT	(0.76, 0.86)	(0.73, 0.85)	(0.75, 0.95)	(0.71, 0.85)	(0.75, 0.85)
0	MLP	(0.81, 0.93)	(0.77, 0.93)	(0.8, 0.93)	(0.82, 0.94)	(0.75, 0.96)
1	MLP	(0.81, 0.93)	(0.85, 0.94)	(0.84, 0.97)	(0.82, 0.94)	(0.84, 0.94)

Fonte: Próprio Autor.

Dados de Treinamento Balanceados -Under :72 alunos, 36 concluintes e 36 desistentes;

Dados de validação Não Balanceados -NB: 20 alunos, 10 concluintes e 10 desistentes;

Tabela 34 – RL, SVM, KNN e DT aplicados nos dados de validação balanceados Under

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.83, 0.92)	(0.83, 0.91)	(0.85, 0.95)	(0.75, 0.92)	(0.81, 0.9)
1	LR	(0.83, 0.92)	(0.83, 0.91)	(0.78, 0.92)	(0.87, 0.97)	(0.84, 0.92)
0	SVM	(0.84, 0.94)	(0.84, 0.94)	(0.86, 0.97)	(0.8, 0.94)	(0.84, 0.94)
1	SVM	(0.84, 0.94)	(0.84, 0.94)	(0.8, 0.95)	(0.84, 0.97)	(0.83, 0.94)
0	KNN	(0.8, 0.9)	(0.8, 0.9)	(0.86, 0.93)	(0.73, 0.89)	(0.79, 0.91)
1	KNN	(0.8, 0.9)	(0.8, 0.9)	(0.74, 0.89)	(0.86, 0.93)	(0.8, 0.9)
0	DT	(0.72, 0.83)	(0.73, 0.85)	(0.73, 0.87)	(0.65, 0.89)	(0.7, 0.86)
1	DT	(0.72, 0.83)	(0.73, 0.85)	(0.7, 0.88)	(0.69, 0.87)	(0.71, 0.83)
0	MLP	(0.79, 0.89)	(0.73, 0.85)	(0.72, 0.83)	(0.8, 0.88)	(0.71, 0.94)
1	MLP	(0.79, 0.89)	(0.83, 0.91)	(0.81, 0.98)	(0.8, 0.88)	(0.82, 0.9)

Fonte: Próprio Autor.

Dados de Treinamento Balanceados -Over : 80 alunos, 40 concluintes e 40 desistentes

Dados de validação Não Balanceados -NB: 20 alunos, 10 concluintes e 10 desistentes;

Tabela 35 – RL, SVM, KNN e DT aplicados nos dados de validação balanceados Over

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.81, 0.92)	(0.82, 0.93)	(0.81, 0.96)	(0.77, 0.94)	(0.8, 0.93)
1	LR	(0.81, 0.92)	(0.82, 0.93)	(0.78, 0.95)	(0.81, 0.94)	(0.81, 0.92)
0	SVM	(0.8, 0.94)	(0.79, 0.94)	(0.72, 0.91)	(0.82, 0.99)	(0.77, 0.94)
1	SVM	(0.8, 0.94)	(0.79, 0.94)	(0.82, 0.99)	(0.75, 0.91)	(0.79, 0.94)
0	KNN	(0.78, 0.88)	(0.79, 0.88)	(0.82, 0.96)	(0.67, 0.87)	(0.75, 0.89)
1	KNN	(0.78, 0.88)	(0.79, 0.88)	(0.72, 0.87)	(0.84, 0.95)	(0.79, 0.88)
0	DT	(0.73, 0.86)	(0.73, 0.86)	(0.72, 0.83)	(0.76, 0.94)	(0.74, 0.87)
1	DT	(0.73, 0.86)	(0.73, 0.86)	(0.72, 0.93)	(0.67, 0.81)	(0.7, 0.85)
0	MLP	(0.89, 0.95)	(0.88, 0.95)	(0.85, 0.95)	(0.88, 0.95)	(0.9, 0.97)
1	MLP	(0.89, 0.95)	(0.9, 0.94)	(0.91, 0.98)	(0.88, 0.95)	(0.86, 0.94)

Fonte: Próprio Autor.

Tabela 36 – Intervalo de confiança de $LSTM_{5NB}$

Modelo	target	accuracy e std	precision e std	recall e std	f1score e std
LSTM	0	(0.80, 0.89)	(0.75, 0.9)	(0.80, 0.89)	(0.81, 0.90)
LSTM	1	(0.80, 0.89)	(0.75, 0.9)	(0.80, 0.91)	(0.81, 0.90)

Fonte: Próprio Autor.

Tabela 37 – Melhor de 5P

Target/Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0 SVM_{5NB}	(0.85, 0.95)	(0.86, 0.95)	(0.85, 0.97)	(0.83, 0.96)	(0.85, 0.95)
1 SVM_{5NB}	(0.85, 0.95)	(0.86, 0.95)	(0.83, 0.96)	(0.84, 0.97)	(0.84, 0.95)
0 $SVM_{5BUnder}$	(0.84, 0.94)	(0.84, 0.94)	(0.86, 0.97)	(0.8, 0.94)	(0.84, 0.94)
1 $SVM_{5BUnder}$	(0.84, 0.94)	(0.84, 0.94)	(0.8, 0.95)	(0.84, 0.97)	(0.83, 0.94)
0 MLP_{5BOver}	(0.89, 0.95)	(0.88, 0.95)	(0.85, 0.95)	**(0.88, 0.95)**	(0.9, 0.97)
1 MLP_{5BOver}	(0.89, 0.95)	(0.9, 0.94)	(0.91, 0.98)	(0.88, 0.95)	0.86, 0.94)

Fonte: Próprio Autor.

5.1.1.9 Conjunto de dados 6P -Treino e Validação

Dados de alunos que cursaram pelo menos o **sexto período**: 100 alunos, 59 formandos e 41 evadidos.

Dados de Treinamento e validação Não Balanceados -NB: 85 alunos, 47 concluintes e 38 desistentes;

Dados de Treinamento Não Balanceados -NB: 68 alunos, 39 concluintes e 29 desistentes;

Dados de validação Não Balanceados -NB: 17 alunos, 8 concluintes e 9 desistentes;

Tabela 38 – RL, SVM, KNN e DT aplicados nos dados de validação Não balanceados

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.83, 0.88)	(0.78, 0.87)	(0.84, 0.94)	(0.87, 0.92)	(0.74, 0.86)
1	LR	(0.85, 0.91)	(0.89, 0.93)	(0.85, 0.92)	(0.87, 0.92)	(0.93, 0.97)
0	SVM	(0.83, 0.88)	(0.85, 0.91)	(0.85, 0.94)	(0.88, 0.93)	(0.83, 0.93)
1	SVM	(0.88, 0.93)	(0.89, 0.94)	(0.88, 0.95)	(0.88, 0.93)	(0.9, 0.96)
0	KNN	(0.82, 0.89)	(0.79, 0.87)	(0.75, 0.85)	(0.82, 0.89)	(0.83, 0.91)
1	KNN	(0.82, 0.89)	(0.84, 0.9)	(0.87, 0.94)	(0.82, 0.89)	(0.8, 0.88)
0	DT	(0.77, 0.82)	(0.7, 0.78)	(0.69, 0.82)	(0.76, 0.82)	(0.71, 0.83)
1	DT	(0.77, 0.82)	(0.79, 0.84)	(0.79, 0.87)	(0.76, 0.82)	(0.78, 0.86)
0	MLP	(0.79, 0.92)	(0.72, 0.89)	(0.74, 0.95)	(0.79, 0.91)	(0.67, 0.91)
1	MLP	(0.79, 0.92)	(0.81, 0.92)	(0.74, 0.93)	(0.79, 0.91)	(0.87, 0.97)

Fonte: Próprio Autor.

Dados de Treinamento Balanceados -Under : 54 alunos, 28 concluintes e 28 desistentes; e

Dados de validação Não Balanceados -NB: 17 alunos, 8 concluintes e 9 desistentes;

Tabela 39 – RL, SVM, KNN e DT aplicados nos dados de validação balanceados Under

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.77, 0.82)	(0.84, 0.9)	(0.81, 0.9)	(0.88, 0.92)	(0.85, 0.94)
1	LR	(0.88, 0.93)	(0.89, 0.94)	(0.91, 0.96)	(0.88, 0.92)	(0.88, 0.94)
0	SVM	(0.86, 0.91)	(0.83, 0.9)	(0.8, 0.89)	(0.86, 0.91)	(0.86, 0.93)
1	SVM	(0.86, 0.91)	(0.87, 0.92)	(0.9, 0.95)	(0.86, 0.91)	(0.84, 0.91)
0	KNN	(0.86, 0.91)	(0.81, 0.89)	(0.78, 0.87)	(0.85, 0.91)	(0.84, 0.93)
1	KNN	(0.85, 0.91)	(0.87, 0.92)	(0.9, 0.95)	(0.86, 0.91)	(0.85, 0.9)
0	DT	(0.82, 0.87)	(0.75, 0.84)	(0.74, 0.85)	(0.82, 0.88)	(0.78, 0.88)
1	DT	(0.82, 0.87)	(0.84, 0.9)	(0.86, 0.93)	(0.82, 0.88)	(0.82, 0.9)
0	MLP	(0.79, 0.91)	(0.61, 0.91)	(0.52, 0.9)	(0.71, 0.91)	(0.85, 0.97)
1	MLP	(0.79, 0.91)	(0.77, 0.92)	(0.88, 0.98)	(0.71, 0.91)	(0.67, 0.91)

Fonte: Próprio Autor.

Dados de Treinamento Balanceados -Over : 74 alunos, 37 concluintes e 37 desistentes

Dados de validação Não Balanceados -NB: 17 alunos, 8 concluintes e 9 desistentes;

Tabela 40 – RL, SVM, KNN e DT aplicados nos dados de validação balanceados Over

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.86, 0.91)	(0.83, 0.9)	(0.85, 0.92)	(0.87, 0.92)	(0.81, 0.9)
1	LR	(0.86, 0.91)	(0.89, 0.93)	(0.88, 0.93)	(0.87, 0.92)	(0.89, 0.95)
0	SVM	(0.88, 0.92)	(0.84, 0.9)	(0.8, 0.89)	(0.87, 0.92)	(0.87, 0.96)
1	SVM	(0.88, 0.92)	(0.88, 0.92)	(0.9, 0.97)	(0.87, 0.92)	(0.85, 0.92)
0	KNN	(0.86, 0.91)	(0.84, 0.9)	(0.8, 0.89)	(0.86, 0.91)	(0.89, 0.95)
1	KNN	(0.86, 0.91)	(0.86, 0.91)	(0.91, 0.96)	(0.86, 0.91)	(0.8, 0.9)
0	DT	(0.75, 0.82)	(0.67, 0.78)	(0.68, 0.8)	(0.76, 0.83)	(0.67, 0.81)
1	DT	(0.75, 0.82)	(0.79, 0.86)	(0.78, 0.88)	(0.76, 0.83)	(0.8, 0.87)
0	MLP	(0.83, 0.91)	(0.83, 0.9)	(0.76, 0.96)	(0.82, 0.91)	(0.83, 0.97)
1	MLP	(0.83, 0.91)	(0.8, 0.91)	(0.86, 0.97)	(0.82, 0.91)	(0.72, 0.96)

Fonte: Próprio Autor.

Tabela 41 – Intervalo de confiança de $LSTM_{6NB}$

Modelo	target	accuracy e std	precision e std	recall e std	f1score e std
LSTM	0	(0.83, 0.98)	(0.81, 1.0)	(0.73, 0.99)	(0.78, 0.98)
LSTM	1	(0.83, 0.98)	(0.83, 0.98)	(0.87, 0.99)	(0.86, 0.97)

Fonte: Próprio Autor.

Tabela 42 – Melhor de 6P

Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0 SVM_{6NB}	(0.83, 0.88)	(0.85, 0.91)	(0.85, 0.94)	(0.88, 0.93)	(0.83, 0.93)
1 SVM_{6NB}	(0.88, 0.93)	(0.89, 0.94)	(0.88, 0.95)	(0.88, 0.93)	(0.9, 0.96)
0 $LR_{6BUnder}$	(0.77, 0.82)	(0.84, 0.9)	(0.81, 0.9)	**(0.88, 0.92)**	(0.85, 0.94)
1 $LR_{6BUnder}$	(0.88, 0.93)	(0.89, 0.94)	(0.91, 0.96)	(0.88, 0.92)	(0.88, 0.94)
0 SVM_{6BOver}	(0.88, 0.92)	(0.84, 0.9)	(0.8, 0.89)	(0.87, 0.92)	(0.87, 0.96)
1 SVM_{6BOver}	(0.88, 0.92)	(0.88, 0.92)	(0.9, 0.97)	(0.87, 0.92)	(0.85, 0.92)

Fonte: Próprio Autor.

5.1.1.10 Conjunto de dados 7P -Treino e Validação

Dados de alunos que cursaram pelo menos o **sétimo período**: 94 alunos, 58 formandos e 36 evadidos.

Dados de Treinamento e validação Não Balanceados -NB: 79 alunos, 51 concluintes e 28 desistentes;

Dados de Treinamento Não Balanceados -NB: 64 alunos, 38 concluintes e 26 desistentes;

Dados de validação Não Balanceados -NB: 16 alunos, 12 concluintes e 4 desistentes;

Tabela 43 – RL, SVM, KNN e DT aplicados nos dados de validação Não balanceados

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.82, 0.88)	(0.76, 0.85)	(0.86, 0.95)	(0.84, 0.9)	(0.69, 0.82)
1	LR	(0.82, 0.88)	(0.87, 0.92)	(0.83, 0.9)	(0.84, 0.9)	(0.92, 0.97)
0	SVM	(0.87, 0.92)	(0.82, 0.89)	(0.83, 0.94)	(0.87, 0.92)	(0.8, 0.89)
1	SVM	(0.87, 0.92)	(0.89, 0.93)	(0.86, 0.93)	(0.87, 0.92)	(0.91, 0.96)
0	KNN	(0.88, 0.93)	(0.83, 0.9)	(0.79, 0.88)	(0.87, 0.92)	(0.88, 0.96)
1	KNN	(0.88, 0.93)	(0.89, 0.93)	(0.91, 0.97)	(0.87, 0.92)	(0.85, 0.92)
0	DT	(0.8, 0.89)	(0.74, 0.85)	(0.76, 0.86)	(0.82, 0.89)	(0.74, 0.89)
1	DT	(0.8, 0.89)	(0.84, 0.91)	(0.85, 0.94)	(0.82, 0.89)	(0.83, 0.9)
0	MLP	(0.82, 0.9)	(0.8, 0.9)	(0.86, 0.97)	(0.83, 0.93)	(0.73, 0.88)
1	MLP	(0.82, 0.9)	(0.84, 0.95)	(0.81, 0.94)	(0.83, 0.93)	(0.87, 0.98)

Fonte: Próprio Autor.

Dados de Treinamento Balanceados Under -BUnder: alunos, concluintes e desistentes;

Dados de validação Balanceados Under -BUnder: alunos, concluintes e desistentes;

Tabela 44 – RL, SVM, KNN e DT aplicados nos dados de validação balanceados Under

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.86, 0.91)	(0.82, 0.89)	(0.88, 0.95)	(0.88, 0.92)	(0.77, 0.87)
1	LR	(0.86, 0.91)	(0.89, 0.94)	(0.86, 0.92)	(0.88, 0.92)	(0.92, 0.97)
0	SVM	(0.85, 0.9)	(0.8, 0.87)	(0.84, 0.92)	(0.86, 0.91)	(0.77, 0.88)
1	SVM	(0.85, 0.9)	(0.89, 0.93)	(0.87, 0.93)	(0.86, 0.91)	(0.9, 0.95)
0	KNN	(0.85, 0.91)	(0.79, 0.88)	(0.74, 0.86)	(0.85, 0.9)	(0.85, 0.95)
1	KNN	(0.85, 0.91)	(0.86, 0.91)	(0.91, 0.97)	(0.85, 0.9)	(0.81, 0.89)
0	DT	(0.8, 0.88)	(0.76, 0.85)	(0.73, 0.84)	(0.8, 0.88)	(0.81, 0.91)
1	DT	(0.8, 0.88)	(0.81, 0.89)	(0.85, 0.93)	(0.8, 0.88)	(0.78, 0.88)
0	MLP	(0.83, 0.93)	(0.62, 0.87)	(0.48, 0.81)	(0.79, 0.92)	(0.85, 1.0)
1	MLP	(0.83, 0.93)	(0.84, 0.93)	(0.94, 1.0)	(0.79, 0.92)	(0.74, 0.89)

Fonte: Próprio Autor.

Dados de Treinamento Balanceados Over -BOver: 248 alunos, 124concluintes e 124 desistentes;

Dados de validação Balanceados Over -BOver: 40 alunos, 10 concluintes e 30 desistentes;

Tabela 45 – RL, SVM, KNN e DT aplicados nos dados de validação balanceados Over

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.84, 0.9)	(0.79, 0.87)	(0.81, 0.91)	(0.85, 0.91)	(0.77, 0.88)
1	LR	(0.84, 0.9)	(0.87, 0.93)	(0.85, 0.92)	(0.85, 0.91)	(0.89, 0.95)
0	SVM	(0.85, 0.9)	(0.82, 0.89)	(0.85, 0.93)	(0.86, 0.91)	(0.79, 0.87)
1	SVM	(0.85, 0.9)	(0.87, 0.92)	(0.86, 0.92)	(0.86, 0.91)	(0.89, 0.95)
0	KNN	(0.84, 0.9)	(0.79, 0.87)	(0.72, 0.83)	(0.83, 0.89)	(0.88, 0.96)
1	KNN	(0.84, 0.9)	(0.84, 0.9)	(0.91, 0.97)	(0.83, 0.89)	(0.78, 0.87)
0	DT	(0.83, 0.89)	(0.76, 0.85)	(0.73, 0.84)	(0.84, 0.89)	(0.79, 0.89)
1	DT	(0.83, 0.89)	(0.87, 0.91)	(0.89, 0.94)	(0.84, 0.89)	(0.84, 0.9)
0	MLP	(0.87, 0.93)	(0.84, 0.9)	(0.81, 0.91)	(0.88, 0.93)	(0.81, 0.95)
1	MLP	(0.87, 0.93)	(0.9, 0.95)	(0.9, 0.98)	(0.88, 0.93)	(0.88, 0.96)

Fonte: Próprio Autor.

Tabela 46 – Intervalo de confiança de $LSTM_{7NB}$

Modelo	target	accuracy e std	precision e std	recall e std	f1score e std
LSTM	0	(0.71, 0.94)	(0.63, 0.92)	(0.74, 0.97)	(0.68, 0.93)
LSTM	1	(0.71, 0.94)	(0.79, 0.98)	(0.67, 0.97)	(0.73, 0.96)

Fonte: Próprio Autor.

Tabela 47 – Melhor de 7P

Target/Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0 SVM_{7NB}	(0.87, 0.92)	(0.82, 0.89)	(0.83, 0.94)	(0.87, 0.92)	(0.8, 0.89)
1 SVM_{7NB}	(0.87, 0.92)	(0.89, 0.93)	(0.86, 0.93)	(0.87, 0.92)	(0.91, 0.96)
0 $LR_{7BUnder}$	(0.86, 0.91)	(0.82, 0.89)	(0.88, 0.95)	(0.88, 0.92)	(0.77, 0.87)
1 $LR_{7BUnder}$	(0.86, 0.91)	(0.89, 0.94)	(0.86, 0.92)	(0.88, 0.92)	(0.92, 0.97)
0 MLP_{7BOver}	(0.87, 0.93)	(0.84, 0.9)	(0.81, 0.91)	**(0.88, 0.93)**	(0.81, 0.95)
1 MLP_{7BOver}	(0.87, 0.93)	(0.9, 0.95)	(0.9, 0.98)	(0.88, 0.93)	(0.88, 0.96)

Fonte: Próprio Autor.

5.1.1.11 Conjunto de dados 8P -Treino e Validação

Dados de alunos que cursaram pelo menos o **oitavo período**: 78 alunos, 56 formandos e 22 evadidos.

Dados de Treinamento e validação Não Balanceados -NB: 62 alunos, 44 concluintes e 14 desistentes;

Dados de Treinamento Não Balanceados : 49 alunos, 37 concluintes e 12 desistentes; e

Dados de validação Não Balanceados : 13 alunos, 11 concluintes e 2 desistentes;

Tabela 48 – RL, SVM, KNN e DT aplicados nos dados de validação Não balanceados

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.76, 0.96)	(0.85, 0.95)	(0.63, 1.05)	(0.56, 0.95)	(0.59, 0.99)
1	LR	(0.76, 0.96)	(0.85, 0.95)	(0.83, 0.95)	(0.93, 1.0)	(0.89, 0.96)
0	SVM	(0.88, 0.98)	(0.87, 0.97)	(0.71, 0.98)	(0.82, 1.01)	(0.77, 0.95)
1	SVM	(0.88, 0.98)	(0.87, 0.97)	(0.92, 1.01)	(0.88, 0.99)	(0.91, 0.98)
0	KNN	(0.81, 0.99)	(0.86, 0.98)	(0.74, 0.98)	(0.7, 1.01)	(0.71, 0.96)
1	KNN	(0.81, 0.99)	(0.86, 0.98)	(0.91, 1.0)	(0.88, 1.0)	(0.9, 0.99)
0	DT	(0.82, 0.93)	(0.85, 0.94)	(0.72, 0.96)	(0.77, 0.93)	(0.76, 0.91)
1	DT	(0.82, 0.93)	(0.85, 0.94)	(0.86, 0.97)	(0.81, 0.99)	(0.84, 0.97)
0	MLP	(0.88, 0.96)	(0.83, 0.93)	(0.79, 0.92)	(0.88, 0.96)	(0.86, 0.98)
1	MLP	(0.88, 0.96)	(0.9, 0.97)	(0.91, 0.99)	(0.88, 0.96)	(0.89, 0.96)

Fonte: Próprio Autor.

Dados de Treinamento Balanceados -Under : 24 alunos, 12 concluintes e 12 desistentes

Dados de validação Não Balanceados : 13 alunos, 11 concluintes e 2 desistentes;

Tabela 49 – RL, SVM, KNN e DT aplicados nos dados de validação balanceados Under

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.86, 1.0)	(0.91, 1.0)	(0.82, 1.04)	(0.75, 1.02)	(0.79, 1.0)
1	LR	(0.86, 1.0)	(0.91, 1.0)	(0.91, 1.01)	(0.95, 1.01)	(0.94, 1.0)
0	SVM	(0.85, 1.0)	(0.87, 0.97)	(0.77, 0.98)	(0.75, 1.07)	(0.73, 0.98)
1	SVM	(0.85, 1.0)	(0.87, 0.97)	(0.9, 1.02)	(0.88, 0.98)	(0.91, 0.98)
0	KNN	(0.89, 0.98	(0.89, 0.97)	(0.74, 0.96)	(0.9, 1.02)	(0.82, 0.96)
1	KNN	(0.89, 0.98)	(0.89, 0.97)	(0.94, 1.01)	(0.85, 0.97)	(0.9, 0.98)
0	DT	(0.77, 0.91)	(0.78, 0.9)	(0.51, 0.8)	(0.71, 0.96)	(0.61, 0.83)
1	DT	(0.77, 0.91)	(0.78, 0.9)	(0.91, 0.98)	(0.76, 0.92)	(0.83, 0.93)
0	MLP	(0.9, 0.97)	(0.82, 0.93)	(0.75, 0.95)	(0.92, 0.97)	(0.8, 1.0)
1	MLP	(0.9, 0.97)	(0.94, 0.97)	(0.95, 1.0)	(0.92, 0.97)	(0.9, 0.97)

Fonte: Próprio Autor.

Dados de Treinamento Balanceados -Over : 74 alunos, 37 concluintes e 37 desistentes

Dados de validação Não Balanceados : 13 alunos, 11 concluintes e 2 desistentes;

Tabela 50 – RL, SVM, KNN e DT aplicados nos dados de validação balanceados Over

Target	Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0	LR	(0.8, 0.97)	(0.86, 0.96)	(0.74, 0.99)	(0.67, 1.0)	(0.71, 0.95)
1	LR	(0.8, 0.97)	(0.86, 0.96)	(0.89, 1.0)	(0.88, 0.98)	(0.9, 0.96)
0	SVM	(0.93, 1.0)	(0.92, 1.0)	(0.78, 1.0)	(0.94, 1.02)	(0.85, 1.0)
1	SVM	(0.93, 1.0)	(0.92, 1.0)	(0.96, 1.02)	(0.92, 1.0)	(0.94, 1.0)
0	KNN	(0.81, 0.97)	(0.81, 0.96)	(0.58, 0.93)	(0.8, 1.02)	(0.67, 0.94)
1	KNN	(0.81, 0.97)	(0.81, 0.96)	(0.92, 1.01)	(0.76, 0.97)	(0.84, 0.97)
0	DT	(0.78, 0.93)	(0.79, 0.95)	(0.74, 0.98)	(0.74, 0.93)	(0.74, 0.92)
1	DT	(0.78, 0.93)	(0.79, 0.95)	(0.83, 0.95)	(0.77, 0.99)	(0.8, 0.96)
0	MLP	(0.93, 0.97)	(0.86, 0.95)	(0.77, 0.93)	(0.91, 0.96)	(0.96, 1.0)
1	MLP	(0.93, 0.97)	(0.93, 0.97)	(0.98, 1.0)	(0.91, 0.96)	(0.87, 0.95)

Fonte: Próprio Autor.

Tabela 51 – Intervalo de confiança de $LSTM_{8NB}$

Modelo	target	accuracy e std	precision e std	recall e std	f1score e std
LSTM	0	(0.80, 0.92)	(0.7, 0.96)	(0.59, 0.88)	(0.68, 0.83)
LSTM	1	(0.80, 0.92)	(0.87, 0.97)	(0.82, 1.0)	(0.84, 0.95)

Fonte: Próprio Autor.

Tabela 52 – Melhor de 8P

Modelo	rocauc e std	accuracy e std	precision e std	recall e std	f1score e std
0 MLP_{8NB}	(0.88, 0.96)	(0.83, 0.93)	(0.79, 0.92)	(0.88, 0.96)	(0.86, 0.98)
1 MLP_{8NB}	(0.88, 0.96)	(0.9, 0.97)	(0.91, 0.99)	(0.88, 0.96)	(0.89, 0.96)
0 MLP_{8Under}	(0.9, 0.97)	(0.82, 0.93)	(0.75, 0.95)	(0.92, 0.97)	(0.8, 1.0)
1 MLP_{8Under}	(0.9, 0.97)	(0.94, 0.97)	(0.95, 1.0)	(0.92, 0.97)	(0.9, 0.97)
0 SVM_{8BOver}	(0.93, 1.0)	(0.92, 1.0)	(0.78, 1.0)	**(0.94, 1.0)**	(0.85, 1.0)
1 SVM_{8BOver}	(0.93, 1.0)	(0.92, 1.0)	(0.96, 1.02)	(0.92, 1.0)	(0.94, 1.0)

Fonte: Próprio Autor.

5.1.2 Comparação dos Resultados dos Modelos nos treinos e testes

Como os dados dos primeiros períodos e dos últimos são desbalanceados, optamos por avaliar também os métodos de subamostragem e sobreamostragem para balancear a distribuição de classes nos dados de treinamento. Utilizamos a biblioteca "imblearn under-sampling"para gerar subamostra e sobreamostra aleatórias ("RandomUnderSampler"). No caso de subamostragem, foi gerada subamostra aleatória da classe de maior frequência e foi mantida a classe de menor frequência. No caso de sobreamostragem, foi gerada uma amostra replicando aleatoriamente os registros da classe de menor frequência ("RandomOverSampler").

Para separar os dados de treino e teste, Hold-out ou Validação Cruzada, separa os dados de forma aleatória, então com exceção dos dados utilizados para treino e teste do algoritmo LSTM, o qual usamos uma ordem cronológica dos dados, as demais dados utilizados para treino e teste dos modelos "Logistic-Regression","Support-Vector-Machin", "KNN","DecisionTreeClassifier"e Rede neural MLP, foram separados de forma aleatória e para não perder informações cronológicas as informações de vários semestres foram transformadas numa média. Por exemplo, o conjunto de dados com os alunos que cursaram pelo menos n semestres (períodos), se for feito separação aleatória, pode ser que tenhamos dados de $n-k$ períodos no conjunto de dados de teste e k períodos no conjunto de dados de treinamento deste mesmo aluno, então para evitar isso, achamos melhor calcular a média dos dados de cada atributo de cada aluno, com isso cada aluno ficou com representado por somente uma linha de dados e não n linhas. Veja o exemplo para conjunto de dados 6P.

Os conjuntos de dados i-semanas para os algoritmos "LSTM", teve a inserção na linha $i+1$ o TARGET. Vejamos o exemplo, seja dado o conjunto de dados de 6 semestres, então $7k,\ k \in \{1, 2, \cdots, N\}$ linha foi inserido o TARGET do aluno em todas as colunas de atributos. Veja a figura abaixo.

Figura 51 – Conjuntos de dados 6P

Aluno	Período	QtDisAprovadasSemestre	QtDisReprovadasSemestre	QtDisReprovadasSemestrePorNota	QtDisReprovadasSemestrePorFalta	PorcMediaFaltasSemestre	NotaMediaSemestre	QtDscisplinasSemestre	ChDisciplinasSemestre	TARGET
Aluno 01	1	8	0	0	0	0,06	8,45	8	390	0
Aluno 01	2	2	3	2	1	0,16	5,14	5	390	0
Aluno 01	3	3	1	1	0	0,14	7	4	180	0
Aluno 01	4	4	1	1	0	0,16	7,18	5	360	0
Aluno 01	5	6	1	1	0	0,09	7,64	7	405	0
Aluno 01	6	8	0	0	0	0,02	9	8	465	0
Aluno	Período	QtDisAprovadasSemestre	QtDisReprovadasSemestre	QtDisReprovadasSemestrePorNota	QtDisReprovadasSemestrePorFalta	PorcMediaFaltasSemestre	NotaMediaSemestre	QtDscisplinasSemestre	ChDisciplinasSemestre	TARGET
Aluno 01	1	5,1666667	1	0,833333	0,166667	0,105	7,401667	6,166667	365	0

Fonte: Próprio Autor

Figura 52 – Conjuntos de dados 6P- LSTM

Aluno	Período	QtDisAprovadasSemestre	QtDisReprovadasSemestre	QtDisReprovadasSemestrePorNota	QtDisReprovadasSemestrePorFalta	PorcMediaFaltasSemestre	NotaMediaSemestre	QtDscisplinasSemestre	ChDisciplinasSemestre	**TARGET**
Aluno 01	1	8	0	0	0	0,06	8,45	8	390	**0**
Aluno 01	2	2	3	2	1	0,16	5,14	5	390	**0**
Aluno 01	3	3	1	1	0	0,14	7	4	180	**0**
Aluno 01	4	4	1	1	0	0,16	7,18	5	360	**0**
Aluno 01	5	6	1	1	0	0,09	7,64	7	405	**0**
Aluno 01	6	8	0	0	0	0,02	9	8	465	**0**
Aluno	Período	QtDisAprovadasSemestre	QtDisReprovadasSemestre	QtDisReprovadasSemestrePorNota	QtDisReprovadasSemestrePorFalta	PorcMediaFaltasSemestre	NotaMediaSemestre	QtDscisplinasSemestre	ChDisciplinasSemestre	
Aluno 01	1	8	0	0	0	0,06	8,45	8	390	
Aluno 01	2	2	3	2	1	0,16	5,14	5	390	
Aluno 01	3	3	1	1	0	0,14	7	4	180	
Aluno 01	4	4	1	1	0	0,16	7,18	5	360	
Aluno 01	5	6	1	1	0	0,09	7,64	7	405	
Aluno 01	6	8	0	0	0	0,02	9	8	465	
TARGET	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	

Fonte: Próprio Autor

5.1.3 Melhor modelo para cada dataset conforme a métrica f1-score e intervalo de confiança

Embora a validação cruzada, possa ajudar a ter uma estimativa da performance do modelo. Decidimos escolher o melhor algoritmo olhando os resultados dos algortimos que foram aplicados no conjunto de dados de testes.

Resultados de Treino: Eles mostram o desempenho do modelo nos dados com os quais ele foi treinado. No entanto, esses resultados podem ser enganosos, porque o modelo pode se ajustar muito bem a esses dados (inclusive superajustar).

Resultados de Teste: Estes resultados refletem o desempenho do modelo em dados que ele nunca viu antes, sendo mais representativos da sua capacidade de generalizar para

novos exemplos. Testar em dados novos é o melhor indicador de como o modelo vai se comportar em um ambiente real.

Aplicamos 10 testes utilizando a proporção 75% para dados de treino e 25% para dados teste. Com base nos resultados dos modelos aplicados nos conjuntos de dados para **testes**, tivemos os seguintes resultados com intervelos de confiança de 95%.

Tabela 53 – Melhores Modelos

Target Modelo	rocauc e std	accuracy	precision	recall	f1score
0 SVM_{1NB}	(0.5, 0.51)	(0.82, 0.87)	(0.82, 0.87)	**(1.0, 1.0)**	(0.9, 0.93)
1 SVM_{1NB}	(0.5, 0.51)	(0.82, 0.87)	(0.0, 0.31)	(0.0, 0.02)	(0.0, 0.04)
0 LR_{2NB}	(0.76, 0.81)	(0.89, 0.92)	(0.88, 0.92)	**(0.84, 0.87)**	(0.9, 0.93)
1 LR_{2NB}	(0.76, 0.81)	(0.62, 0.71)	(0.66, 0.76)	(0.84, 0.87)	(0.6, 0.71)
0 MLP_{3BOver}	(0.86, 0.94)	(0.87, 0.94)	(0.91, 0.98)	**(0.86, 0.93)**	(0.85, 0.92)
1 MLP_{3BOver}	(0.86, 0.94)	(0.82, 0.9)	(0.77, 0.87)	(0.86, 0.93)	(0.85, 0.97)
0 $SVM_{4BUnder}$	(0.86, 0.91)	(0.87, 0.92)	(0.9, 0.94)	**(0.86, 0.91)**	(0.85, 0.91)
1 $SVM_{4BUnder}$	(0.86, 0.91)	(0.83, 0.89)	(0.8, 0.87)	(0.86, 0.91)	(0.87, 0.92)
0 MLP_{5BOver}	(0.89, 0.95)	(0.88, 0.95)	(0.85, 0.95)	**(0.88, 0.95)**	(0.9, 0.97)
1 MLP_{5BOver}	(0.89, 0.95)	(0.9, 0.94)	(0.91, 0.98)	(0.88, 0.95)	(0.86, 0.94)
0 $LR_{6BUnder}$	(0.77, 0.82)	(0.84, 0.9)	(0.81, 0.9)	**(0.88, 0.92)**	(0.85, 0.94)
1 $LR_{6BUnder}$	(0.88, 0.93)	(0.89, 0.94)	(0.91, 0.96)	(0.88, 0.92)	(0.88, 0.94)
0 MLP_{7BOver}	(0.87, 0.93)	(0.84, 0.9)	(0.81, 0.91)	**(0.88, 0.93)**	(0.81, 0.95)
1 MLP_{7BOver}	(0.87, 0.93)	(0.9, 0.95)	(0.9, 0.98)	(0.88, 0.93)	(0.88, 0.96)
0 SVM_{8BOver}	(0.93, 1.0)	(0.92, 1.0)	(0.78, 1.0)	**(0.94, 1.0)**	(0.85, 1.0)
1 SVM_{8BOver}	(0.93, 1.0)	(0.92, 1.0)	(0.96, 1.02)	(0.92, 1.0)	(0.94, 1.0)

Fonte: Próprio Autor.

5.1.4 Hiperparâmetros

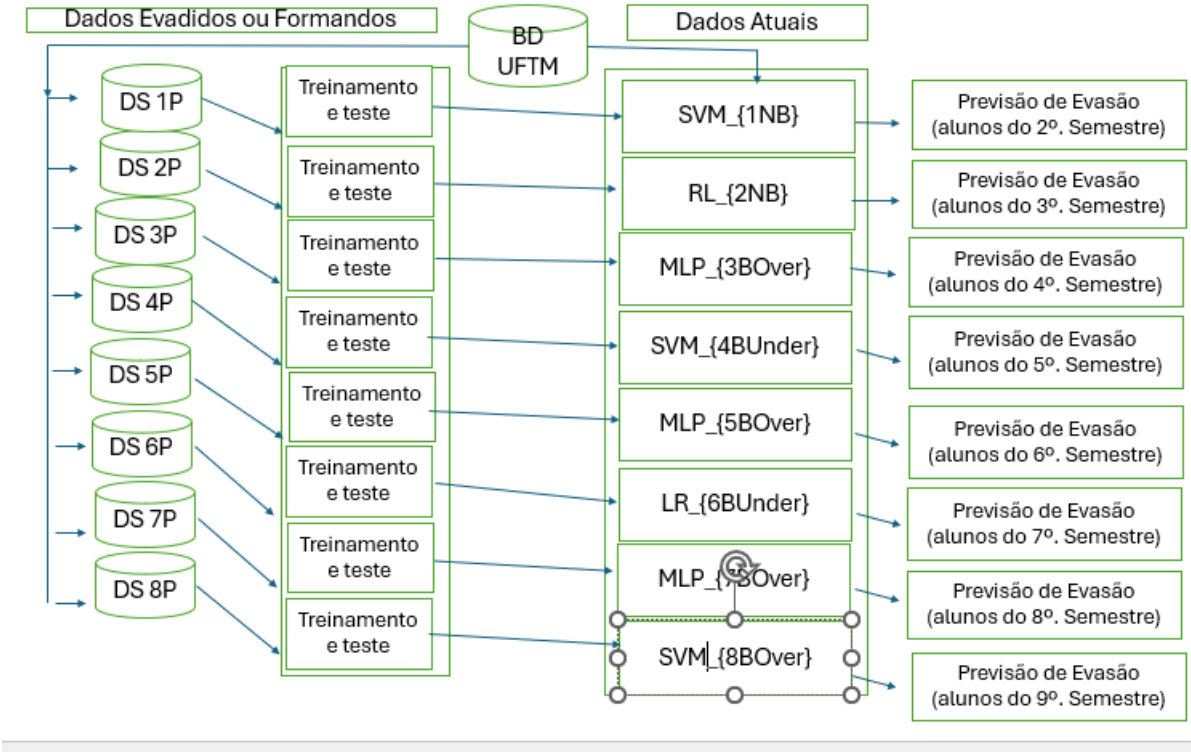
A maioria dos algoritmos de aprendizado de máquina possuem hiperparâmetros, configurações que pode-se usar para controlar o comportamento do algoritmo. Em outras palavras os hiperparâmetros são as variáveis que determinam a estrutura do modelo. Na fase anterior, treinamos os modelos sem analisar quais parâmetros seriam melhores para os modelos, então aplicaremos duas técnicas que buscam otimizá-los que resultará uma melhor acurácia em seu modelo.

5.1.4.1 Ajuste de hiperparâmetros

Abordaremos duas técnicas para melhorar os parâmetros dos modelos:

1. Grid search
2. Random search

Figura 53 – Sistema de Previsão de Evasões com 8 modelos



Fonte: Proprio Autor

5.1.4.2 Grid search

Essa técnica irá testar todas as combinações possíveis dos hiperparâmetros, exaustivamente. Basicamente, irá fornecer alguns valores de input e testar todas as combinações plotando em um plano cartesiano (por isso o nome de grid). Em seguida, selecionará os hiperparâmetros que obtiveram o menor erro.

```
from sklearn.model_selection import GridSearchCV
```

5.1.4.2.1 Modelo SVM_{1NB}

```
parameters = {
'gamma':[1,0.75,0.5,0.3,0.2,0.1,0.01,0.001,0.0001],
'kernel':['linear','rbf','poly'],
'C': [0.001,0.01,0.1,1,5,10,50,100],}
 model =SVC()}
# Criar a m trica de scoring para recall do r tulo 0
scorer = make_scorer(recall_score, pos_label=0)
grid_search = GridSearchCV(model, parameters,cv=5,scoring=scorer)
grid_search.fit(X_treino, y_treino)
```

```
#GRID
print(grid_search.best_params_)
print(grid_search.score(X_treino, y_treino))
print(grid_search.score(X_teste, y_teste))
{'C': 0.001, 'gamma': 1, 'kernel': 'linear'}
1.0
1.0
```

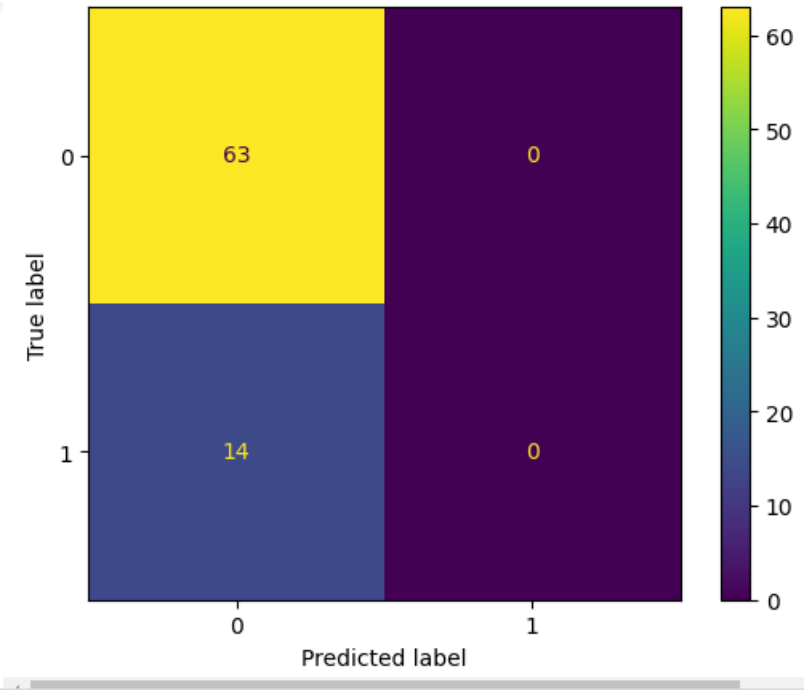
```
model =SVC(C=0.001,kernel='linear',gamma=1.0)
```

Tabela 54 – SVM_{1NB} tunado

Modelo Teste	Target	accuracy	precision	recall	f1score
SVM_{1NB}	0	(0.82, 0.87)	(0.82, 0.87)	**(1.0, 1.0)**	(0.9, 0.93)
SVM_{1NB}	1	(0.82, 0.87)	(0.0, 0.31)	(0.0, 0.02)	(0.0, 0.04)
SVM_{1NB} tunado	0	0.882	0.882	1.000	0.938
SVM_{1NB} tunado	1	0.882	0.000	0.000	0.000

Fonte: Próprio Autor.

Figura 54 – Modelo $MLP_{1BUnder}$ aplicado nos dados de teste



Fonte: Produzido pelo autor

5.1.4.2.2 Modelo LR_{2NB}

Seja dado os seguintes parâmetros

```
parameters = {'penalty' : ['l1', 'l2', 'elasticnet', 'none'],
              'C' : np.logspace(-4, 4, 20),
              'solver' : ['lbfgs','newton-cg','liblinear','sag','
                 saga'],
              'max_iter' : [100, 1000,2500, 5000]
              }
model =LogisticRegression()
```

```
#GRID
print(grid_search.best_params_)
print(grid_search.score(X_treino, y_treino))
print(grid_search.score(X_teste, y_teste))
{'C': 0.0001, 'max_iter': 100, 'penalty': 'l1', 'solver': '
   liblinear'}
1.0
1.0
```

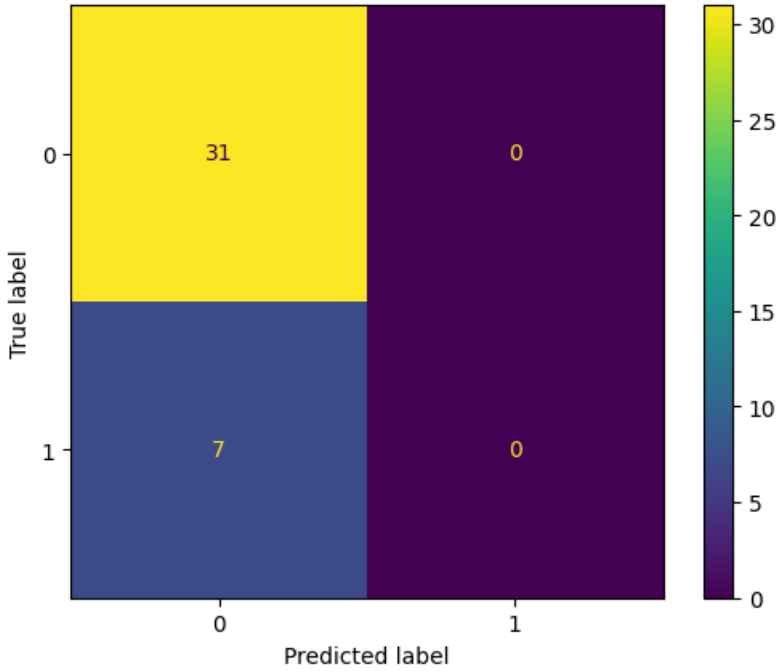
```
model = LogisticRegression(C=0.0001, max_iter=100,
 penalty='l1', solver='liblinear')
```

Tabela 55 – LR_{2NB} tunado

Modelo Teste	Target	accuracy	precision	recall	f1score
LR_{2NB}	0	(0.89, 0.92)	(0.88, 0.92)	**(0.84, 0.87)**	(0.9, 0.93)
LR_{2NB}	1	(0.62, 0.71)	(0.66, 0.76)	(0.84, 0.87)	(0.6, 0.71)
LR_{2NB} tunado	0	0.816	0.816	1.000	0.899
LR_{2NB} tunado	1	0.816	0.000	0.000	0.000

Fonte: Próprio Autor.

Figura 55 – Modelo RL_{2NB} tunado aplicado nos dados de teste



Fonte: Produzido pelo autor

5.1.4.2.3 Modelo MLP_{3BOver}

```
parameters = {'gamma':[1,0.75,0.5,0.3,0.2,0.1,0.01,0.001,0.0001],
              'kernel':['linear','rbf','poly'],
              'C': [0.001,0.01,0.1,1,5,10,50,100]}
model =SVC()
grid_search = GridSearchCV(model, parameters,cv=5,scoring='f1')
grid_search.fit(X_treino, y_treino)
```

```
#GRID
print(grid_search.best_params_)
print(grid_search.score(X_treino, y_treino))
print(grid_search.score(X_teste, y_teste))
Melhores par metros: {'activation': 'tanh',
'batch_size': 1, 'hidden_layer_sizes': (10, 5),
 'learning_rate': 'constant', 'max_iter': 75,
'solver': 'adam'}
Melhor recall para label=0 (valida  o cruzada): nan
```

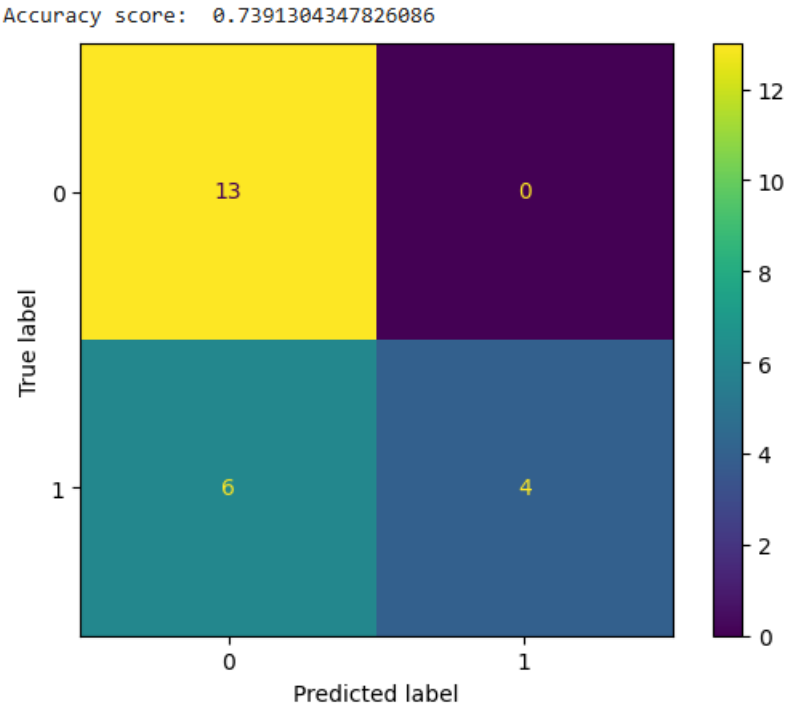
```
model =SVC(C=1.0,kernel='poly',gamma=0.2)
```

Tabela 56 – SVM_{3BOver} tunado

Modelo Teste	Target	accuracy	precision	recall	f1score
0 MLP_{3BOver}	0	(0.87, 0.94)	(0.91, 0.98)	**(0.86, 0.93)**	(0.85, 0.92)
1 MLP_{3BOver}	1	(0.82, 0.9)	(0.77, 0.87)	(0.86, 0.93)	(0.85, 0.97)
0 MLP_{3BOver} tunado	0	0.789	0.870	0.800	0.833
1 MLP_{3BOver} tunado	1	0.789	0.667	0.769	0.714

Fonte: Próprio Autor.

Figura 56 – Modelo $SVM_{4BUnder}$ aplicado nos dados de teste



Fonte: Produzido pelo autor

5.1.4.2.4 Modelo $SVM_{4BUnder}$

```
parameters = {'gamma':[1,0.75,0.5,0.3,0.2,0.1,0.01,0.001,0.0001],
              'kernel':['linear','rbf','poly'],
              'C': [0.001,0.01,0.1,1,5,10,50,100]}
model =SVC()
grid_search = GridSearchCV(model, parameters,cv=5,scoring='f1')
grid_search.fit(X_treino, y_treino)
```

```
#GRID
print(grid_search.best_params_)
print(grid_search.score(X_treino, y_treino))
print(grid_search.score(X_teste, y_teste))
{'C': 1, 'gamma': 0.2, 'kernel': 'poly'}
0.9473684210526315
1.0
```

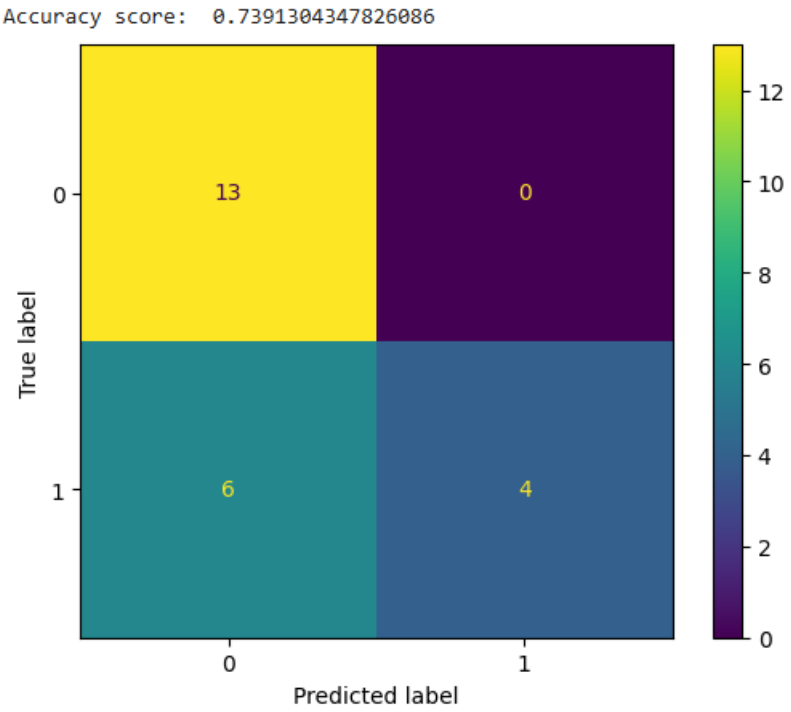
```
model =SVC(C=1.0,kernel='poly',gamma=0.2)
```

Tabela 57 – $SVM_{4BUnder}$ tunado

Modelo Teste	Target	accuracy	precision	recall	f1score
$SVM_{4BUnder}$	0	(0.87, 0.92)	(0.9, 0.94)	**(0.86, 0.91)**	(0.85, 0.91)
$SVM_{4BUnder}$	1	(0.83, 0.89)	(0.8, 0.87)	(0.86, 0.91)	(0.87, 0.92)
$SVM_{4BUnder}$ tunado	0	0.739	0.684	1.000	0.812
$SVM_{4BUnder}$ tunado	1	0.739	1.000	0.400	0.571

Fonte: Próprio Autor.

Figura 57 – Modelo $SVM_{4BUnder}$ aplicado nos dados de teste



Fonte: Produzido pelo autor

5.1.4.2.5 Modelo MLP_{5BOver}

```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV, KFold
from sklearn.metrics import accuracy_score

mlp = MLPClassifier(max_iter=200, random_state=1)

param_grid = {
    'hidden_layer_sizes': [(10,5),(100), (100,50)],
    'activation': ['tanh', 'relu','softmax','sigmoid'],
    'solver': ['adam', 'sgd'],
    'batch_size':[1,2],
    'learning_rate': ['constant', 'adaptive'],
    'max_iter':[75,100,125] # Changed 'epochs' to 'max_iter'}
```

```
#Grid Search
# Criar a m trica de scoring para recall do r tulo 0
scorer = make_scorer(recall_score, pos_label=0)
grid_search = GridSearchCV(mlp, param_grid, cv=5, scoring=scorer)
grid_search.fit(X_treino, y_treino)
best_params = grid_search.best_params_
```

```
print("Melhores hiperpar metros:", best_params)
y_pred01 = grid_search.predict(X_treino)
y_pred02 = grid_search.predict(X_val)
accuracy01 = accuracy_score(y_treino, y_pred01)
accuracy02 = accuracy_score(y_val, y_pred02)
print("Acur cia no conjunto de treino:", accuracy01)
print("Acur cia no conjunto de teste:", accuracy02)
Melhores hiperpar metros: {'activation': 'tanh',
'batch_size': 1, 'hidden_layer_sizes': (10, 5),
'learning_rate': 'constant', 'max_iter': 75,
'solver': 'adam'}
Acur cia no conjunto de treino: 0.925
Acur cia no conjunto de teste: 0.9
```

MLP_{5NB} tunado

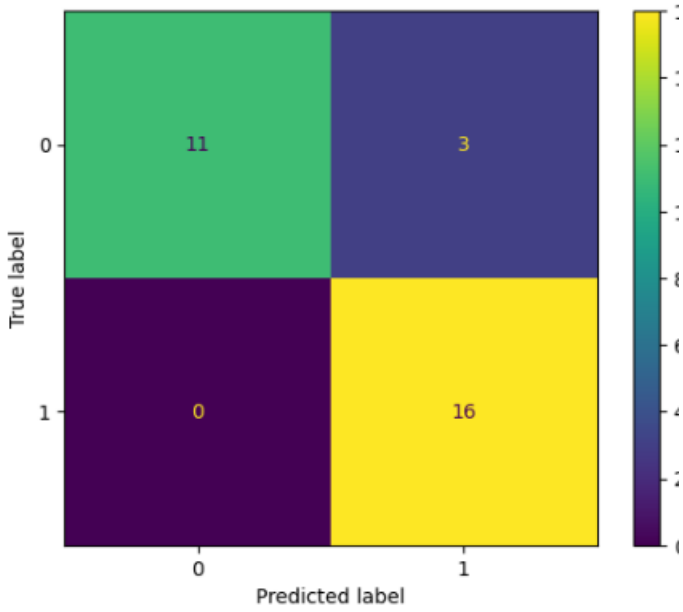
```
model = Sequential()
model.add(Dense(10, input_dim=n_features,kernel_initializer='normal
    ',
activation='tanh'))
#model.add(Dense(5, kernel_initializer='normal', activation='tanh')
    )
model.add(Dense(2, kernel_initializer='normal',activation='softmax'
    ))
model.summary()

model.compile(optimizer=Adam(), loss='categorical_crossentropy',
              metrics=['accuracy'])
historico = model.fit(X_treino, y_treino, batch_size=1, epochs= 75,
            validation_data = (X_teste,y_teste), verbose=1)
```

Tabela 58 – O modelo MLP_{5BOver} tunado

Modelo	Target	accuracy	precision	recall	f1score
MLP_{5BOver}	0	(0.88, 0.95)	(0.85, 0.95)	**(0.88, 0.95)**	(0.9, 0.97)
MLP_{5BOver}	1	(0.9, 0.94)	(0.91, 0.98)	(0.88, 0.95)	(0.86, 0.94)
MLP_{5BOver} tunado	0	0.917	1.000	0.818	0.900
MLP_{5BOver} tunado	1	0.917	0.867	1.000	0.929

Fonte: Próprio Autor.

Figura 58 – Modelo MLP_{5NB} aplicado nos dados de teste



Fonte: Produzido pelo autor

5.1.4.2.6 Modelo $LR_{6BUnder}$

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import ParameterGrid
from sklearn.model_selection import RandomizedSearchCV
parameters = {
'penalty' : ['l1', 'l2', 'elasticnet', 'none'],
'C' : np.logspace(-4, 4, 20),
'solver' : ['lbfgs','newton-cg','liblinear','sag','saga'],
'max_iter' : [100, 1000,2500, 5000],
              }
model =LogisticRegression()
grid_search = GridSearchCV(model, parameters,cv=5,scoring='f1')
grid_search.fit(X_treino, y_treino)
```

```
#GRID
print(grid_search.best_params_)
print(grid_search.score(X_treino, y_treino))
print(grid_search.score(X_teste, y_teste))
{'C': 0.0001, 'max_iter': 100, 'penalty': 'l1',
'solver': 'liblinear'}
1.0
1.0
```

$LR_{6BUnder}$ tunado

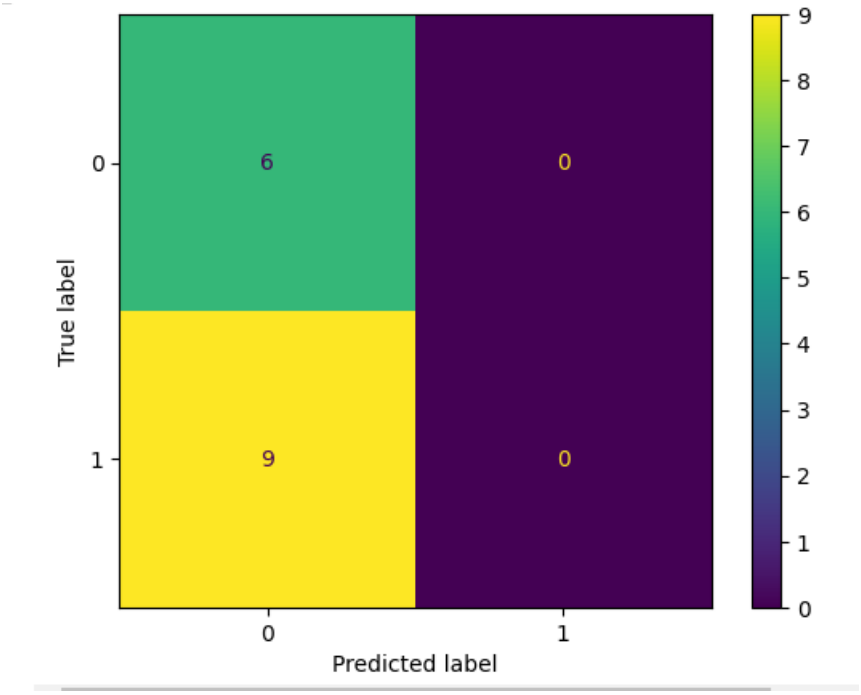
```
model =SVC(C=1.0,kernel='rbf',gamma=0.3)
```

Tabela 59 – O modelo $LR_{6BUnder}$ tunado

Modelo	Target	accuracy	precision	recall	f1score
$LR_{6BUnder}$	0	(0.84, 0.9)	(0.81, 0.9)	**(0.88, 0.92)**	(0.85, 0.94)
$LR_{6BUnder}$	1	(0.89, 0.94)	(0.91, 0.96)	(0.88, 0.92)	(0.88, 0.94)
$LR_{6BUnder}$ tunado	0	0.400	0.400	1.000	0.571
$LR_{6BUnder}$ tunado	1	0.400	0.000	0.000	0.000

Fonte: Próprio Autor.

Figura 59 – Modelo SVM_{6BOver} aplicado nos dados de teste



Fonte: Produzido pelo autor

5.1.4.2.7 Modelo MLP_{7BOver}

```
mlp = MLPClassifier(max_iter=200, random_state=1)

param_grid = {
    'hidden_layer_sizes': [(10,5),(100), (100,50)],
    'activation': ['tanh', 'relu','softmax','sigmoid'],
    'solver': ['adam', 'sgd'],
    'batch_size':[1,2],
    'learning_rate': ['constant', 'adaptive'],
    'max_iter':[75,100,125]
    }
```

```
#Grid Search
# Criar a m trica de scoring para recall do r tulo 0
scorer = make_scorer(recall_score, pos_label=0)
grid_search = GridSearchCV(mlp, param_grid, cv=2, scoring=scorer)
grid_search.fit(X_treino, y_treino)
best_params = grid_search.best_params_
print("Melhores hiperpar metros:", best_params)
y_pred01 = grid_search.predict(X_treino)
y_pred02 = grid_search.predict(X_val)
accuracy01 = accuracy_score(y_treino, y_pred01)
accuracy02 = accuracy_score(y_val, y_pred02)
Melhores hiperpar metros: {'activation': 'tanh', 'batch_size': 1,
'hidden_layer_sizes': (10, 5), 'learning_rate': 'constant',
'max_iter': 75, 'solver': 'adam'}
Acur cia no conjunto de treino: 0.9230769230769231
Acur cia no conjunto de teste: 0.8666666666666667
```

MLP_{7BOver} tunado

```
model = Sequential()
model.add(Dense(10, input_dim=n_features,kernel_initializer='normal
    ',
activation='tanh'))
model.add(Dense(5, kernel_initializer='normal', activation='tanh'))
model.add(Dense(2, kernel_initializer='normal',activation='softmax'
    ))
model.summary()

#model.compile(optimizer=SGD(learning_rate=lr_schedule),
loss='categorical_crossentropy', metrics=['accuracy'])
```

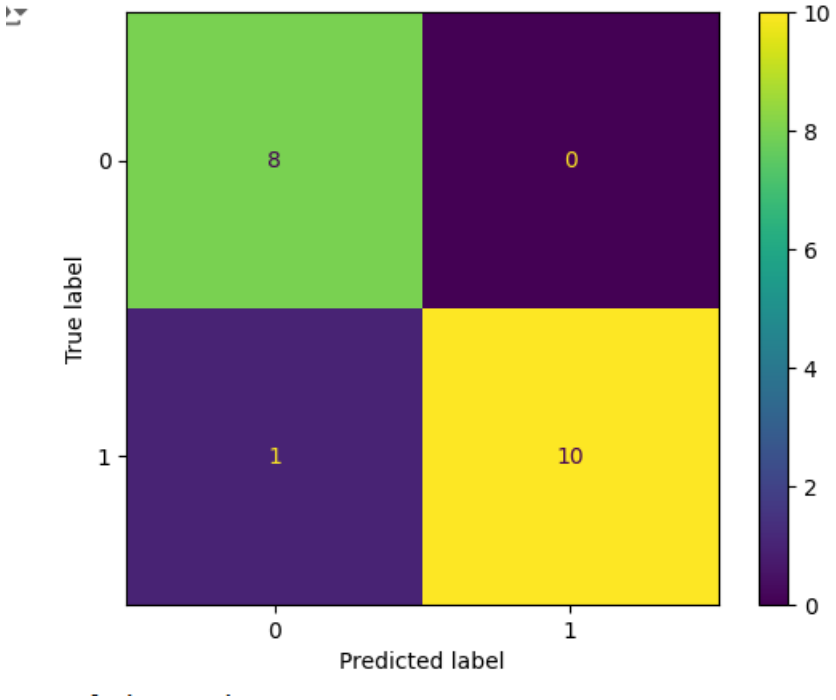
```
model.compile(optimizer=Adam(), loss='categorical_crossentropy',
metrics=['accuracy'])
historico = model.fit(X_treino, y_treino, batch_size=1, epochs=75,
 validation_data = (X_teste,y_teste), verbose=1)
```

Tabela 60 – MLP_{7BOver} tunado

Modelo	Target	accuracy	precision	recall	f1score
MLP_{7BOver}	0	(0.84, 0.9)	(0.81, 0.91)	**(0.88, 0.93)**	(0.81, 0.95)
MLP_{7BOver}	1	(0.9, 0.95)	(0.9, 0.98)	(0.88, 0.93)	(0.88, 0.96)
MLP_{7BOver} tunado	0	0.947	0.889	1.000	0.941
MLP_{7BOver} tunado	1	0.947	1.000	0.909	0.952

Fonte: Próprio Autor.

Figura 60 – Modelo SVM_{7NB} aplicado nos dados de teste



Fonte: Produzido pelo autor

5.1.4.2.8 Modelo SVM_{8BOver}

```
parameters = {
'gamma':[1,0.75,0.5,0.3,0.2,0.1,0.01,0.001,0.0001],
'kernel':['linear','rbf','poly'],
'C': [0.001,0.01,0.1,1,5,10,50,100],
                    }
 model =SVC()
# Criar a m trica de scoring para recall do r tulo 0
scorer = make_scorer(recall_score, pos_label=0)
grid_search = GridSearchCV(model, parameters,cv=5,scoring=scorer)
grid_search.fit(X_treino, y_treino)
```

```
#GRID
print(grid_search.best_params_)
print(grid_search.score(X_treino, y_treino))
print(grid_search.score(X_teste, y_teste))
{'C': 0.1, 'gamma': 1, 'kernel': 'rbf'}
1.0
1.0
```

SVM_{8BOver} tunado

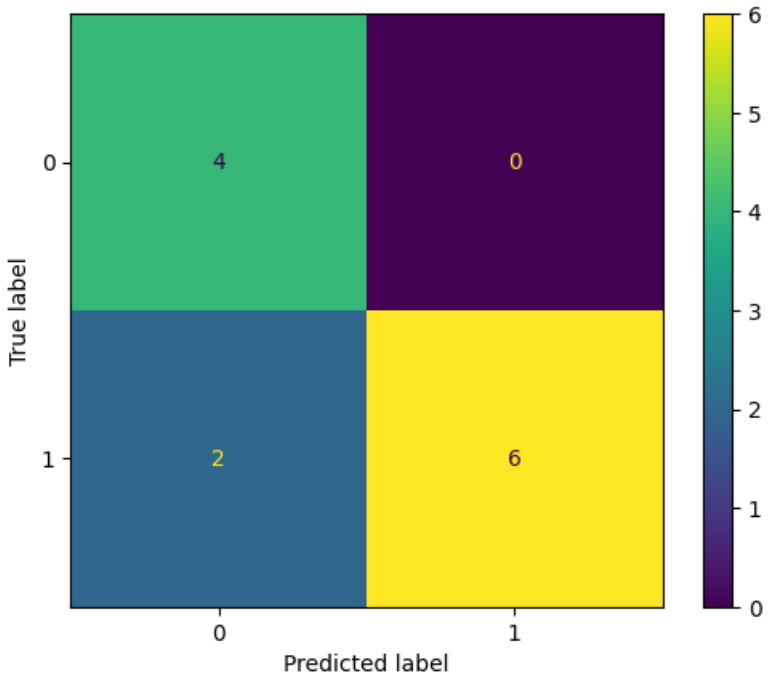
```
model =SVC(C=0.1,kernel='rbf',gamma=1.0)
```

Tabela 61 – O modelo MLP_{8BOver} tunado

Modelo	Target	accuracy	precision	recall	f1score
SVM_{8BOver}	0	(0.92, 1.0)	(0.78, 1.0)	(0.94, 1.0)	(0.85, 1.0)
SVM_{8BOver}	1	(0.92, 1.0)	(0.96, 1.02)	(0.92, 1.0)	(0.94, 1.0)
SVM_{8BOver} tunado	0	0.833	0.667	1.000	0.800
SVM_{8BOver} tunado	1	0.833	1.000	0.750	0.857

Fonte: Próprio Autor.

Figura 61 – Modelo $MLP_{8BUnder}$ aplicado nos dados de teste



Fonte: Produzido pelo autor

5.1.4.3 Random search

Essa técnica acaba suprindo o problema com muitas combinações. Isso se deve ao fato de testar combinações aleatórias e os melhores resultados funciona como um guia para a escolha dos próximos hiperparâmetros. Contudo, percebemos claramente que isso poderá levar em sua maioria, para o mínimo local e não para o mínimo global.

```
from sklearn.model_selection import RandomizedSearchCV
```

Random search

```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV, KFold,
    RandomizedSearchCV
from sklearn.metrics import accuracy_score
```

5.1.4.3.1 Modelo $MLP_{1BUnder}$

```
parameters = {
'gamma':[1,0.75,0.5,0.3,0.2,0.1,0.01,0.001,0.0001],
'kernel':['linear','rbf','poly'],
'C': [0.001,0.01,0.1,1,5,10,50,100],
                    }
model =SVC()
```

```
# Criar a m trica de scoring para recall do r tulo 0
scorer = make_scorer(recall_score, pos_label=0)
random_search = RandomizedSearchCV(model, parameters,cv=5,scoring=
    scorer)
random_search .fit(X_treino, y_treino)
```

```
#RANDOM
print(random_search .best_params_)
print(random_search .score(X_treino, y_treino))
print(random_search .score(X_teste, y_teste))
{'kernel': 'linear', 'gamma': 0.001, 'C': 0.01}
1.0
1.0
```

SVM_{1NB} tunado

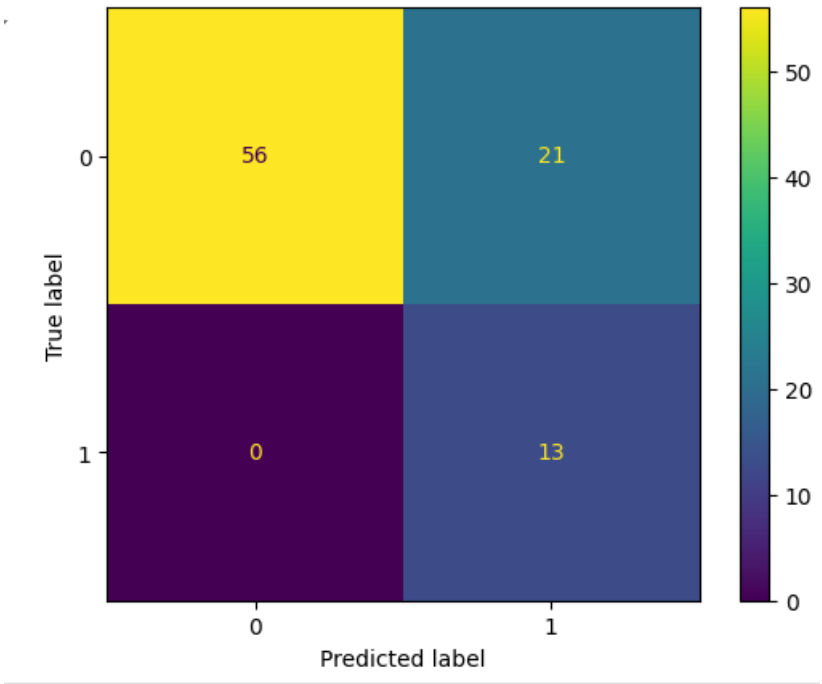
```
model =SVC(C=0.01,kernel='linear',gamma=0.001)
```

Tabela 62 – SVM_{1NB} tunado

Modelo Teste	Target	accuracy	precision	recall	f1score
SVM_{1NB}	0	(0.82, 0.87)	(0.82, 0.87)	**(1.0, 1.0)**	(0.9, 0.93)
SVM_{1NB}	1	(0.82, 0.87)	(0.0, 0.31)	(0.0, 0.02)	(0.0, 0.04)
SVM_{1NB} tunado	0	0.882	0.882	1.000	0.938
SVM_{1NB} tunado	1	0.882	0.000	0.000	0.000

Fonte: Próprio Autor.

Figura 62 – Modelo $MLP_{1BUnder}$ aplicado nos dados de teste



Fonte: Produzido pelo autor

5.1.4.3.2 Modelo LR_{2NB}

Seja dado os seguintes parâmetros

```
# RANDOM SEARCH
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import ParameterGrid
from sklearn.model_selection import RandomizedSearchCV

parameters = {'gamma':[1,0.75,0.5,0.3,0.2,0.1,0.01,0.001,0.0001],
              'kernel':['linear','rbf','poly'],
              'C': [0.001,0.01,0.1,1,5,10,50,100]}
model =SVC()
Rand_grid_search = RandomizedSearchCV(model, parameters,cv=5,
   scoring='f1')
Rand_grid_search.fit(X_treino, y_treino)
```

```
#RANDOM
print(Rand_grid_search.best_params_)
print(Rand_grid_search.score(X_treino, y_treino))
print(Rand_grid_search.score(X_teste, y_teste))
{'solver': 'sag', 'penalty': 'l2', 'max_iter': 5000,
'C': 0.012742749857031334}
1.0
```

```
1.0
```

```
model = LogisticRegression(C=0.23357214690901212,
        max_iter=2500, penalty='l1', solver='saga')
```

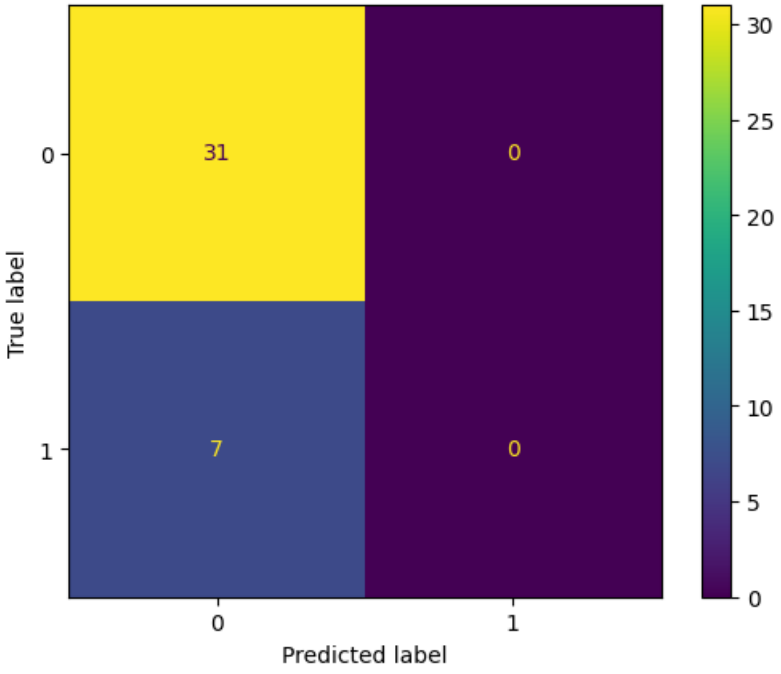
```
model = LogisticRegression(C=0.0001, max_iter=100,
 penalty='l1', solver='liblinear')
```

Tabela 63 – LR_{2NB} tunado

Modelo Teste	Target	accuracy	precision	recall	f1score
LR_{2NB}	0	(0.89, 0.92)	(0.88, 0.92)	**(0.84, 0.87)**	(0.9, 0.93)
LR_{2NB}	1	(0.62, 0.71)	(0.66, 0.76)	(0.84, 0.87)	(0.6, 0.71)
LR_{2NB} tunado	0	0.816	0.816	1.000	0.899
LR_{2NB} tunado	1	0.816	0.000	0.000	0.000

Fonte: Próprio Autor.

Figura 63 – Modelo RL_{2NB} tunado aplicado nos dados de teste



Fonte: Produzido pelo autor

5.1.4.3.3 Modelo SVM_{3BOver}

```
# RANDOM SEARCH
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import ParameterGrid
from sklearn.model_selection import RandomizedSearchCV

parameters = {'gamma':[1,0.75,0.5,0.3,0.2,0.1,0.01,0.001,0.0001],
              'kernel':['linear','rbf','poly'],
              'C': [0.001,0.01,0.1,1,5,10,50,100]}
model =SVC()
Rand_grid_search = RandomizedSearchCV(model, parameters,cv=5,
                   scoring='f1', random_state=0)
Rand_grid_search.fit(X_treino, y_treino)
```

```
#RANDOM
print(Rand_grid_search.best_params_)
print(Rand_grid_search.score(X_treino, y_treino))
print(Rand_grid_search.score(X_teste, y_teste))
{'kernel': 'linear', 'gamma': 0.01, 'C': 100}
0.8421052631578947
0.9230769230769231
```

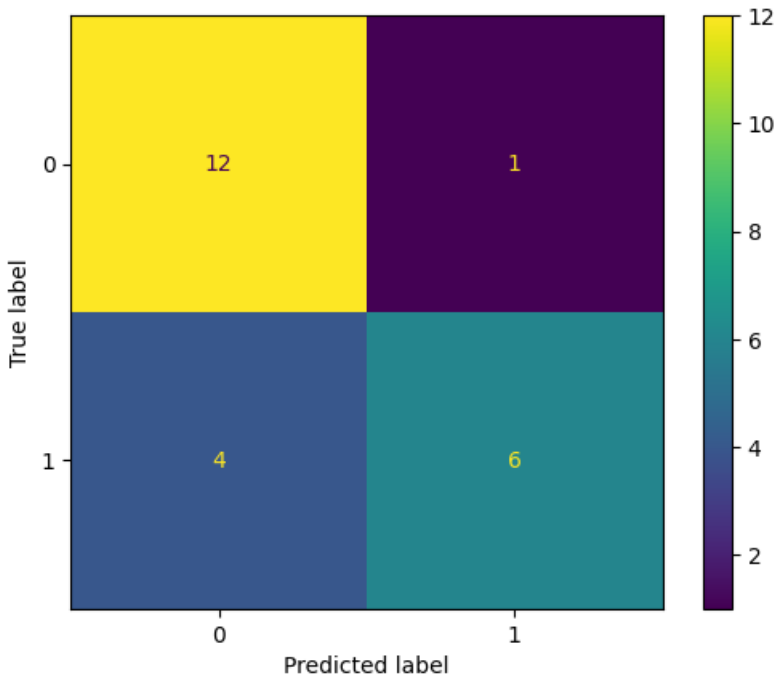
```
model =SVC(C=100.0,kernel='linear',gamma=0.01)
```

Tabela 64 – $SVM_{4BUnder}$ tunado

Modelo Teste	Target	accuracy	precision	recall	f1score
$SVM_{4BUnder}$	0	(0.87, 0.92)	(0.9, 0.94)	(0.86, 0.91)	(0.85, 0.91)
$SVM_{4BUnder}$	1	(0.83, 0.89)	(0.8, 0.87)	(0.86, 0.91)	(0.87, 0.92)
$SVM_{4BUnder}$ tunado	0	0.739	0.750	0.923	0.828
$SVM_{4BUnder}$ tunado	1	0.739	0.857	0.600	0.706

Fonte: Próprio Autor.

Figura 64 – Modelo $SVM_{4BUnder}$ aplicado nos dados de teste



Fonte: Produzido pelo autor

5.1.4.3.4 Modelo $SVM_{4BUnder}$

```
# RANDOM SEARCH
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import ParameterGrid
from sklearn.model_selection import RandomizedSearchCV

parameters = {'gamma':[1,0.75,0.5,0.3,0.2,0.1,0.01,0.001,0.0001],
              'kernel':['linear','rbf','poly'],
              'C': [0.001,0.01,0.1,1,5,10,50,100]}
model =SVC()
Rand_grid_search = RandomizedSearchCV(model, parameters,cv=5,
                   scoring='f1', random_state=0)
Rand_grid_search.fit(X_treino, y_treino)
```

```
#RANDOM
print(Rand_grid_search.best_params_)
print(Rand_grid_search.score(X_treino, y_treino))
print(Rand_grid_search.score(X_teste, y_teste))
{'kernel': 'linear', 'gamma': 0.01, 'C': 100}
0.8421052631578947
0.9230769230769231
```

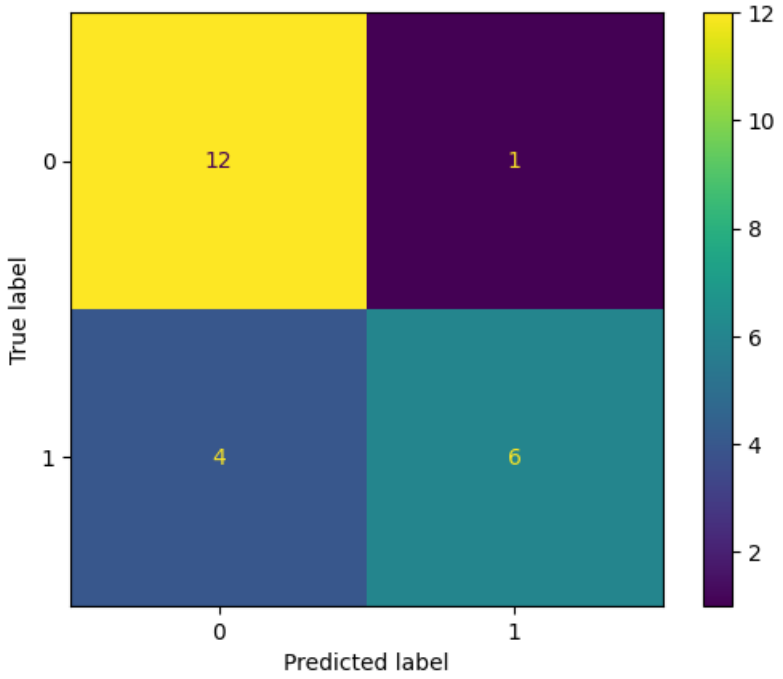
```
model =SVC(C=100.0,kernel='linear',gamma=0.01)
```

Tabela 65 – $SVM_{4BUnder}$ tunado

Modelo Teste	Target	accuracy	precision	recall	f1score
$SVM_{4BUnder}$	0	(0.87, 0.92)	(0.9, 0.94)	(0.86, 0.91)	(0.85, 0.91)
$SVM_{4BUnder}$	1	(0.83, 0.89)	(0.8, 0.87)	(0.86, 0.91)	(0.87, 0.92)
$SVM_{4BUnder}$ tunado	0	0.739	0.750	0.923	0.828
$SVM_{4BUnder}$ tunado	1	0.739	0.857	0.600	0.706

Fonte: Próprio Autor.

Figura 65 – Modelo $SVM_{4BUnder}$ aplicado nos dados de teste



Fonte: Produzido pelo autor

5.1.4.3.5 Modelo MLP_{5BOver}

```
mlp = MLPClassifier(max_iter=200, random_state=1)
param_grid = {
    'hidden_layer_sizes': [(10,5),(100), (100,50)], #neuronios
    'activation': ['tanh', 'relu','softmax','sigmoid'],
    'solver': ['adam', 'sgd'],
    'batch_size':[1,2],
    'learning_rate': ['constant', 'adaptive'],
    'max_iter':[75,100,125] # Changed 'epochs' to 'max_iter'}
```

```
rand_search =RandomizedSearchCV(mlp, param_grid, cv=2, scoring='
   accuracy')
rand_search.fit(X_treino, y_treino)
best_params = grid_search.best_params_
print("Melhores hiperpar metros:", best_params)
y_pred01 = rand_search.predict(X_treino)
y_pred02 = rand_search.predict(X_teste)
accuracy01 = accuracy_score(y_treino, y_pred01)
accuracy02 = accuracy_score(y_teste, y_pred02)
print("Acur cia no conjunto de treino:", accuracy01)
print("Acur cia no conjunto de teste:", accuracy02)
Melhores hiperpar metros: {'solver': 'sgd', 'max_iter': 125,
'learning_rate': 'constant', 'hidden_layer_sizes': (100, 50),
 'batch_size': 2, 'activation': 'tanh'}
Acur cia no conjunto de treino: 0.925
Acur cia no conjunto de teste: 0.85
```

MLP_{5NB} tunado

```
model = Sequential()
model.add(Dense(100, input_dim=n_features,kernel_initializer='
   normal',
 activation='tanh'))
model.add(Dense(50, kernel_initializer='normal', activation='tanh')
   )
model.add(Dense(2, kernel_initializer='normal',activation='softmax'
   ))
model.summary()

model.compile(optimizer=SGD(learning_rate=lr_schedule),
           loss='categorical_crossentropy', metrics=['accuracy'])
```

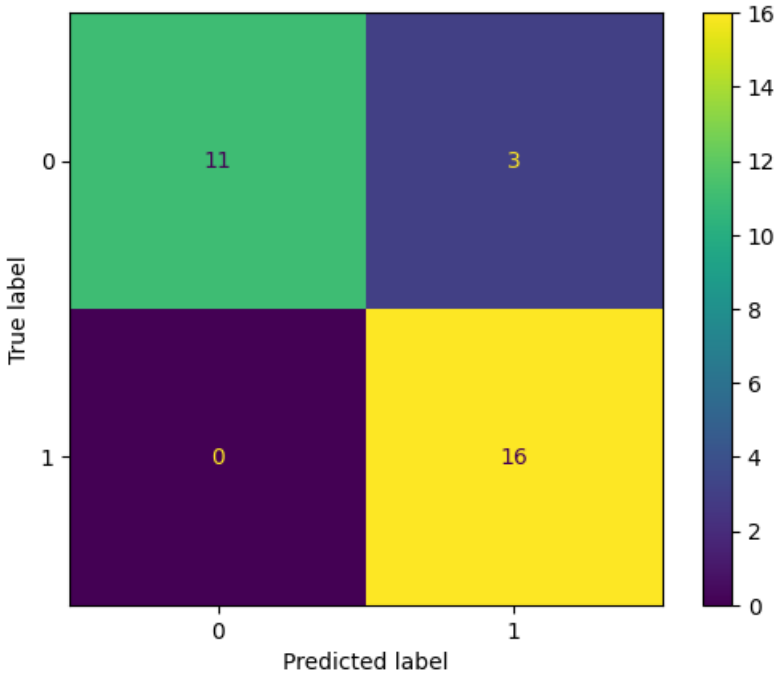
```
historico = model.fit(X_treino, y_treino, batch_size=2, epochs=
    125,
              validation_data = (X_teste,y_teste), verbose=1)
```

Tabela 66 – O modelo MLP_{5BOver} tunado

Modelo	Target	accuracy	precision	recall	f1score
MLP_{5BOver}	0	(0.88, 0.95)	(0.85, 0.95)	**(0.88, 0.95)**	(0.9, 0.97)
MLP_{5BOver}	1	(0.9, 0.94)	(0.91, 0.98)	(0.88, 0.95)	(0.86, 0.94)
MLP_{5BOver} tunado	0	0.917	1.000	0.818	0.900
MLP_{5BOver} tunado	1	0.917	0.867	1.000	0.929

Fonte: Próprio Autor.

Figura 66 – Modelo MLP_{5NB} aplicado nos dados de teste



Fonte: Produzido pelo autor

5.1.4.3.6 Modelo SVM_{6BOver}

```
# RANDOM SEARCH
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import ParameterGrid
from sklearn.model_selection import RandomizedSearchCV

parameters = {
'penalty' : ['l1', 'l2', 'elasticnet', 'none'],
'C' : np.logspace(-4, 4, 20),
 'solver' : ['lbfgs','newton-cg','liblinear','sag','saga'],
'max_iter' : [100, 1000,2500, 5000],
               }
model =LogisticRegression()
Rand_grid_search = RandomizedSearchCV(model, parameters,
cv=5,scoring='f1')
Rand_grid_search.fit(X_treino, y_treino)
```

```
#RANDOM
print(Rand_grid_search.best_params_)
print(Rand_grid_search.score(X_treino, y_treino))
print(Rand_grid_search.score(X_teste, y_teste))
{'solver': 'saga', 'penalty': 'l1', 'max_iter': 5000, 'C': 10000.0}
0.7692307692307693
0.8333333333333334
```

MLP_{6BOver} tunado

```
model = LogisticRegression(C= 10000.0, max_iter=5000, penalty='l1',
solver='saga')
```

$LR_{6BUnder}$ tunado

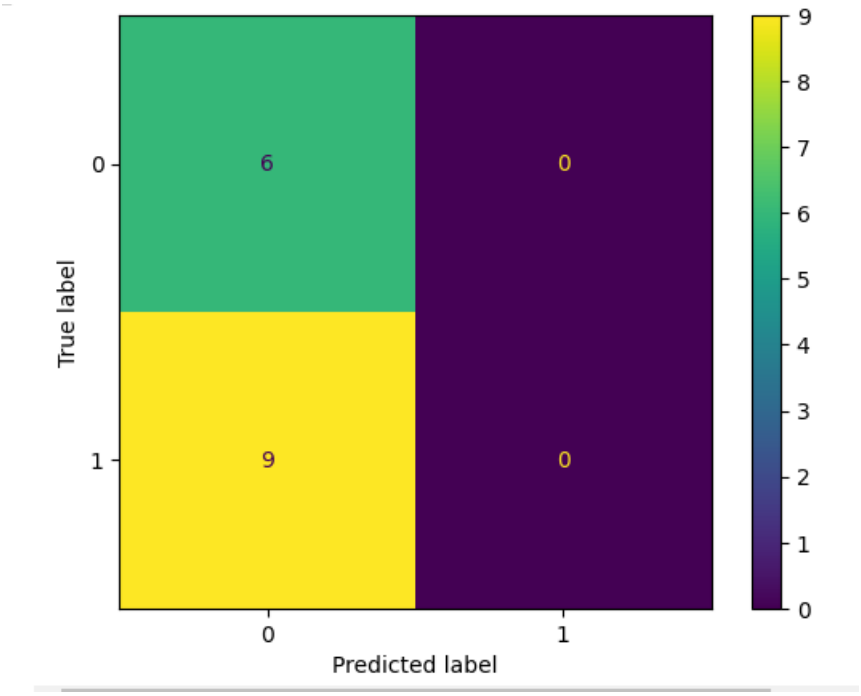
```
model =SVC(C=1.0,kernel='rbf',gamma=0.3)
```

Tabela 67 – O modelo $LR_{6BUnder}$ tunado

Modelo	Target	accuracy	precision	recall	f1score
$LR_{6BUnder}$	0	(0.84, 0.9)	(0.81, 0.9)	**(0.88, 0.92)**	(0.85, 0.94)
$LR_{6BUnder}$	1	(0.89, 0.94)	(0.91, 0.96)	(0.88, 0.92)	(0.88, 0.94)
$LR_{6BUnder}$ tunado	0	0.933	1.000	0.833	0.909
$LR_{6BUnder}$ tunado	1	0.933	0.900	1.000	0.947

Fonte: Próprio Autor.

Figura 67 – Modelo SVM_{6BOver} aplicado nos dados de teste



Fonte: Produzido pelo autor

5.1.4.3.7 Modelo MLP_{7BOver}

```
mlp = MLPClassifier(max_iter=200, random_state=1)

param_grid = {
    'hidden_layer_sizes': [(10,5),(100), (100,50)],
    'activation': ['tanh', 'relu','softmax'],
    'solver': ['adam', 'sgd'],
    'batch_size':[1,2],
    'learning_rate': ['constant', 'adaptive'],
    'max_iter':[75,100,125]
    }
```

```
#Random Search
# Criar a m trica de scoring para recall do r tulo 0
scorer = make_scorer(recall_score, pos_label=0)
rand_search =RandomizedSearchCV(mlp, param_grid,n_iter=20,
cv=5, scoring=scorer, n_jobs=-1)

rand_search.fit(X_treino, y_treino)


best_params = rand_search.best_params_
```

```
print("Melhores hiperpar metros:", best_params)


y_pred01 = rand_search.predict(X_treino)
y_pred02 = rand_search.predict(X_val)
accuracy01 = accuracy_score(y_treino, y_pred01)
accuracy02 = accuracy_score(y_val, y_pred02)
print("Acur cia no conjunto de treino:", accuracy01)
print("Acur cia no conjunto de teste:", accuracy02)
Melhores hiperpar metros: {'solver': 'sgd', 'max_iter': 75,
'learning_rate': 'adaptive', 'hidden_layer_sizes': (10, 5),
 'batch_size': 2, 'activation': 'relu'}
Acur cia no conjunto de treino: 0.9615384615384616
Acur cia no conjunto de teste: 0.9333333333333333
```

MLP_{7BOver} tunado

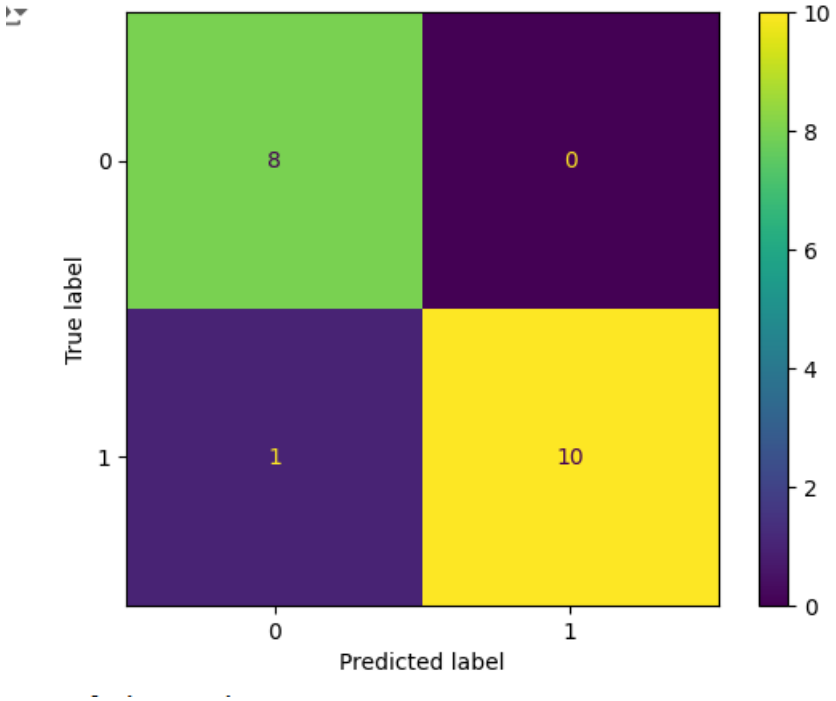
```
model = Sequential()
model.add(Dense(10, input_dim=n_features,kernel_initializer='normal
    ',
 activation='relu'))
model.add(Dense(5, kernel_initializer='normal', activation='relu'))
model.add(Dense(2, kernel_initializer='normal',activation='softmax'
    ))
model.summary()

model.compile(optimizer=SGD(learning_rate=lr_schedule),
loss='categorical_crossentropy', metrics=['accuracy'])
#model.compile(optimizer=Adam(), loss='categorical_crossentropy',
metrics=['accuracy'])
historico = model.fit(X_treino, y_treino, batch_size=2, epochs= 75,
validation_data = (X_teste,y_teste), verbose=1)
```

Tabela 68 – MLP_{7BOver} tunado

Modelo	Target	accuracy	precision	recall	f1score
MLP_{7BOver}	0	(0.84, 0.9)	(0.81, 0.91)	(0.88, 0.93)	(0.81, 0.95)
MLP_{7BOver}	1	(0.9, 0.95)	(0.9, 0.98)	(0.88, 0.93)	(0.88, 0.96)
MLP_{7BOver} tunado	0	0.947	0.889	1.000	0.941
MLP_{7BOver} tunado	1	0.947	1.000	0.909	0.952

Fonte: Próprio Autor.

Figura 68 – Modelo SVM_{7NB} aplicado nos dados de teste



Fonte: Produzido pelo autor

5.1.4.3.8 Modelo SVM_{8BOver}

```
parameters = {
'gamma':[1,0.75,0.5,0.3,0.2,0.1,0.01,0.001,0.0001],
'kernel':['linear','rbf','poly'],
'C': [0.001,0.01,0.1,1,5,10,50,100],
                    }
model =SVC()
# Criar a m trica de scoring para recall do r tulo 0
scorer = make_scorer(recall_score, pos_label=0)
#Rand_grid_search = RandomizedSearchCV(model, parameters,cv=5,
scoring='f1',
random_state=0)
random_search = RandomizedSearchCV(model, parameters,cv=5,
scoring=scorer)
random_search .fit(X_treino, y_treino)
```

```
#RANDOM
print(random_search .best_params_)
print(random_search .score(X_treino, y_treino))
print(random_search .score(X_teste, y_teste))
{'kernel': 'poly', 'gamma': 1, 'C': 100}
1.0
```

```
0.75
```

SVM_{8BOver} tunado

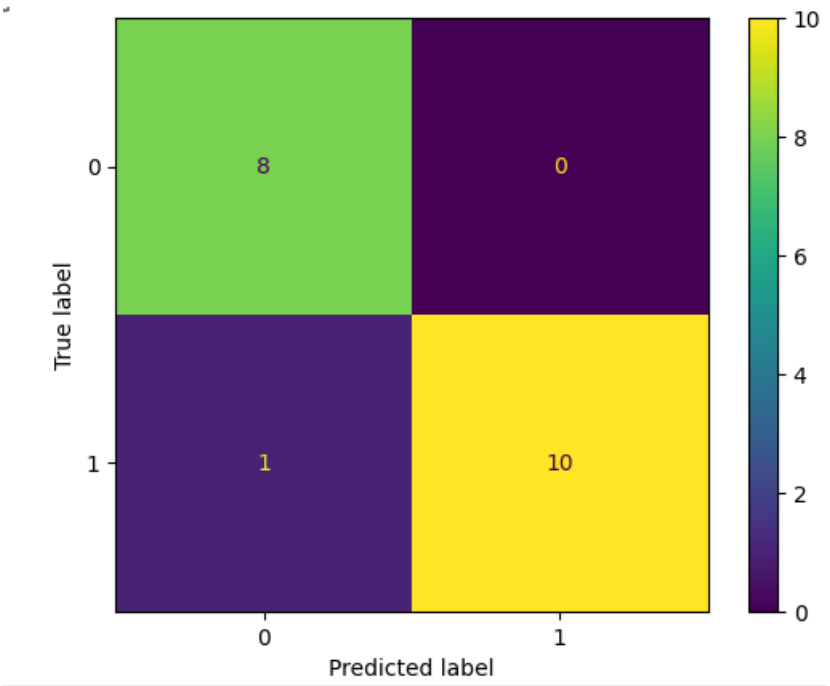
```
model =SVC(C=100.0,kernel='poly',gamma=1.0)
```

Tabela 69 – O modelo MLP_{8BOver} tunado

Modelo	Target	accuracy	precision	recall	f1score
SVM_{8BOver}	0	(0.92, 1.0)	(0.78, 1.0)	(0.94, 1.0)	(0.85, 1.0)
SVM_{8BOver}	1	(0.92, 1.0)	(0.96, 1.0)	(0.92, 1.0)	(0.94, 1.0)
SVM_{8BOver} tunado	0	0.833	0.750	0.750	0.750
SVM_{8BOver} tunado	1	0.833	0.875	0.875	0.875

Fonte: Próprio Autor.

Figura 69 – Modelo $MLP_{8BUnder}$ aplicado nos dados de teste



Fonte: Produzido pelo autor

5.1.4.4 Optuna

Optuna é uma ferramenta de código aberto para framework de otimização de hiperparâmetros para automatizar a busca de hiperparâmetros. Pode ser usada com qualquer framework de aprendizado de máquina ou aprendizado profundo.

```
!pip install optuna
import optuna
```

5.1.4.4.1 Optuna para LSTM

5.1.4.4.2 Modelo $LSTM_{3NB}$

Modelo LSTM que foi aplicado nos treinos e testes.

```
# design network modelo 01
model = Sequential()
model.add(LSTM(10, input_shape=(X_train.shape[1], X_train.shape[2])
    ,
          return_sequences=True))
model.add(LSTM(5, return_sequences=False))
model.add(Dense(8)) #sa da da LSTM
model.compile(loss='mean_absolute_error', optimizer='Adam', metrics
    =['acc'])
model.summary()
historico = model.fit(X_train, Y_train, validation_data = (X_test,
    Y_test),
            epochs=100, batch_size=1, verbose=1)
```

Segue o processo de tunagem usando Optuna

```
$LSTM_{1NB}$
Melhores hiperpar metros: {'lstm_units': 48, 'lstm_units1': 31,
 'dropout_rate': 0.15006387196072826,
'dropout_rate1': 0.21328237705194245,
'learning_rate': 0.0016218386065991902, 'epochs': 50,
'batch_size': 2}
Melhor acur cia de valida  o: 0.699999988079071
```

```
$LSTM_{2NB}$
Melhores hiperpar metros: {'lstm_units': 62,
 'lstm_units1': 29,
'dropout_rate': 0.16413531730174516,
'dropout_rate1': 0.3326032213504911,
```

```
'learning_rate': 0.0004003714072614367, 'epochs': 100,
 'batch_size': 1}
Melhor acur cia de valida  o: 0.8670212626457214
```

```
$LSTM_{3NB}$
Melhores hiperpar metros: {'lstm_units': 86, 'lstm_units1': 22,
'dropout_rate': 0.4139160259065219,
'dropout_rate1': 0.3053881285937692,
'learning_rate': 0.03639619671369007,
'epochs': 100, 'batch_size': 1}
Melhor acur cia de valida  o: 0.8642857074737549
```

```
$LSTM_{4NB}$
Melhores hiperpar metros: {'lstm_units': 62, 'lstm_units1': 20,
'dropout_rate': 0.12426759864242043,
'dropout_rate1': 0.38076932644218753,
'learning_rate': 0.0025079967180776158, 'epochs': 100,
'batch_size': 2}
Melhor acur cia de valida  o: 0.8288288116455078
```

```
$LSTM_{5NB}$
Melhores hiperpar metros: {'lstm_units': 87, 'lstm_units1': 43,
 'dropout_rate': 0.42557965431632516,
'dropout_rate1': 0.4181484549677744,
'learning_rate': 0.014968487898805607, 'epochs': 100,
'batch_size': 1}
Melhor acur cia de valida  o: 0.90625
```

```
$LSTM_{6NB}$
n= 6 d= 0
Melhores hiperpar metros: {'lstm_units': 18, 'lstm_units1': 28,
 'dropout_rate': 0.4048991799266112,
'dropout_rate1': 0.19402623487863468,
'learning_rate': 0.028029106983601317, 'epochs': 100, 'batch_size':
    1}
Melhor acur cia de valida  o: 0.7875000238418579
```

```
$LSTM_{7NB}$
n= 7 d= 0
Melhores hiperpar metros: {'lstm_units': 44, 'lstm_units1': 12,
 'dropout_rate': 0.4335649719263832,
'dropout_rate1': 0.2750996837445917,
```

```
 'learning_rate': 0.0016994245800195861, 'epochs': 100,
'batch_size': 2}
Melhor acur cia de valida  o: 0.7121211886405945
```

```
$LSTM_{8NB}$
n= 8 d= 0
Melhores hiperpar metros: {'lstm_units': 71, 'lstm_units1': 13,
 'dropout_rate': 0.34016510045264275,
'dropout_rate1': 0.18032706272790955,
'learning_rate': 0.016591863288515155, 'epochs': 50,
 'batch_size': 1}
Melhor acur cia de valida  o: 0.7931034564971924
```

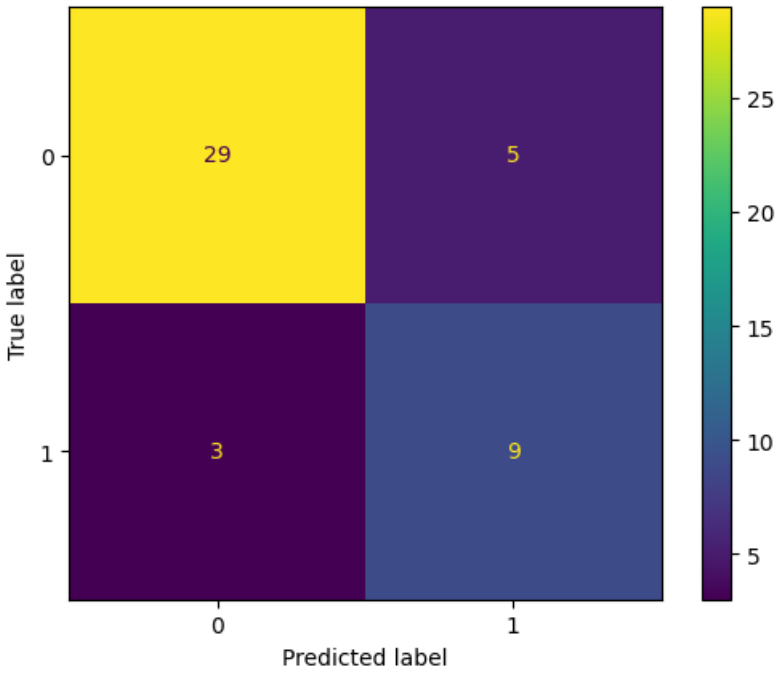
```
# Construir o modelo final com os melhores hiperpar metros
best_model = build_model(study.best_trial)

# Treinar o modelo final
best_model.fit(X_train, Y_train, validation_data=(X_test, Y_test),
epochs=100, batch_size=1, verbose=1)

# Avaliar o modelo final no conjunto de teste
loss, accuracy = best_model.evaluate(X_test, Y_test)
print(f"Acur cia no conjunto de teste: {accuracy}")
```

Tabela 70 – O modelo $LSTM_{3NB}$ tunado

Modelo	Target	accuracy	precision	recall	f1score
$LSTM_{1NB}$	0	(0.65, 0.92)	(0.12, 1.0)	(0.13, 1.0)	(0.12, 1.0)
$LSTM_{1NB}$	1	(0.65, 0.92)	(0.08, 0.74)	(0.0, 0.89)	(0.01, 0.72)
$LSTM_{1NB}$ tunado	0	0.855	0.855	1.0	0.9222
$LSTM_{1NB}$ tunado	1	0.855	0	0	0
$LSTM_{2NB}$	0	(0.86, 0.9)	(0.95, 0.96)	(0.86, 0.90)	(0.91, 092)
$LSTM_{2NB}$	1	(0.86, 0.9)	(0.62, 0.80)	(0.86, 0.90)	(0.73, 0.84)
$LSTM_{2NB}$ tunado	0	0.794	0.880	0.863	0.871
$LSTM_{2NB}$ tunado	1	0.794	0.462	0.500	0.480
$LSTM_{3NB}$	0	(0.81, 0.90)	(0.88, 0.90)	(0.86, 0.95)	(0.87, 0.93)
$LSTM_{3NB}$	1	(0.81, 0.90)	(0.68, 0.86)	(0.63, 0.79)	(0.68, 0.82)
$LSTM_{3NB}$ tunado	0	0.826	0.906	0.853	0.879
$LSTM_{3NB}$ tunado	1	0.826	0.643	0.750	0.692
$LSTM_{4NB}$	0	(0.76, 0.9)	(0.81, 0.93)	(0.77, 0.91)	(0.78, 0.91)
$LSTM_{4NB}$	1	(0.76, 0.9)	(0.70, 0.83)	(0.76, 0.88)	(0.74, 0.86)
$LSTM_{4NB}$ tunado	0	0.865	0.913	0.875	0.894
$LSTM_{4NB}$ tunado	1	0.865	0.786	0.846	0.815
$LSTM_{5NB}$	0	(0.80, 0.89)	(0.75, 0.9)	(0.80, 0.89)	(0.81, 0.90)
$LSTM_{5NB}$	1	(0.80, 0.89)	(0.75, 0.9)	(0.80, 0.91)	(0.81, 0.90)
$LSTM_{5NB}$ tunado	0	0.875	0.909	0.833	0.870
$LSTM_{5NB}$ tunado	1	0.875	0.846	0.917	0.880
$LSTM_{6NB}$	0	(0.83, 0.98)	(0.81, 1.0)	(0.73, 0.99)	(0.78, 0.98)
$LSTM_{6NB}$	1	(0.83, 0.98)	(0.83, 0.98)	(0.87, 0.99)	(0.86, 0.97)
$LSTM_{6NB}$tunado	0	1.000	1.000	1.000	1.000
$LSTM_{6NB}$tunado	1	1.000	1.000	1.000	1.000
$LSTM_{7NB}$	0	(0.71, 0.94)	(0.63, 0.92)	(0.74, 0.97)	(0.68, 0.93)
$LSTM_{7NB}$	1	(0.71, 0.94)	(0.79, 0.98)	(0.67, 0.97)	(0.73, 0.96)
$LSTM_{7NB}$tunado	0	0.964	1.000	0.900	0.947
$LSTM_{7NB}$tunado	1	0.964	0.947	1.000	0.973
$LSTM_{8NB}$	0	(0.80, 0.92)	(0.7, 0.96)	(0.59, 0.88)	(0.68, 0.83)
$LSTM_{8NB}$	1	(0.80, 0.92)	(0.87, 0.97)	(0.82, 1.0)	(0.84, 0.95)
$LSTM_{8NB}$tunado	0	0.950	1.000	0.800	0.889
$LSTM_{8NB}$tunado	1	0.950	0.938	1.000	0.968

Fonte: Próprio Autor.

Figura 70 – Modelo $MLP_{8BUnder}$ aplicado nos dados de teste



Fonte: Produzido pelo autor

6 RECURSO DIDÁTICO E TECNOLÓGICO COM GEOGEBRA E PYTHON

Os aplicativos do Geogebra estão no link abaixo Aplicativo K-Nearest Neighbors (KNN-3) Link para o aplicativo no Geogebra
Árvore de Decisão - Decision Tree Link para o aplicativo no Geogebra

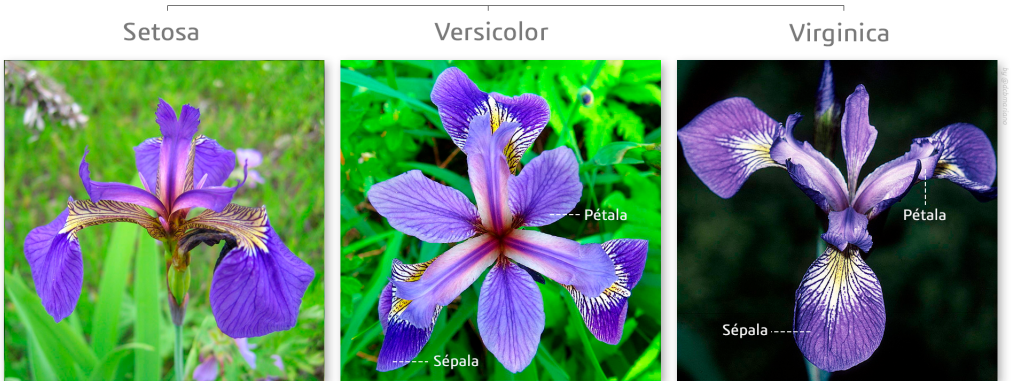
6.1 PROBLEMA: CLASSIFICAR O CONJUNTO DE DADOS DE FLOR DE ÍRIS

O conjunto de dados da flor de íris é um dos conjuntos de dados mais famosos na área de aprendizado de máquina e estatística. Ele foi introduzido por Fisher em 1936. Este conjunto de dados é amplamente utilizado como um exemplo introdutório para várias técnicas de classificação e agrupamento de dados. O conjunto de dados da flor de íris consiste em 150 amostras de flores de íris. Cada amostra pertence a uma de três espécies de íris: Setosa, Versicolor ou Virginica. Para cada amostra, são registradas quatro características:

1. Comprimento do sépala (em centímetros)
2. Largura do sépala (em centímetros)
3. Comprimento da pétala (em centímetros)
4. Largura da pétala (em centímetros)

A imagem a seguir mostra as espécies da flor de íris.

Figura 71 – Espécies de Flor de Íris



Fonte: Imagem retirada de [datacamp], disponível em: https://www.datacamp.com/tutorial/machine-learning-in-r, acesso em: 20 de maio de 2024.

O objetivo primário do conjunto de dados da flor de íris é demonstrar a capacidade de algoritmos de aprendizado de máquina em classificar corretamente as espécies de flores de íris com base em suas características morfológicas. Este conjunto de dados é frequentemente usado para fins educacionais e para avaliar algoritmos de classificação. O conjunto de dados da flor de íris está disponível em várias bibliotecas de software e pacotes de linguagens de programação populares, isso facilita o acesso e a utilização por parte de pesquisadores e estudantes. Uma plataforma em que podemos encontrar esse dados é o Kaggle.

O conjunto de dados da flor de íris serve como um ponto de partida ideal para iniciantes em aprendizado de máquina, pois é pequeno, facilmente compreensível e oferece uma oportunidade para aplicar uma variedade de algoritmos de classificação. Sua disponibilidade e familiaridade o tornam uma ferramenta valiosa para experimentação e aprendizado de algoritmos de predição.

6.2 CONCEITOS MATEMÁTICOS PARA RESOLUÇÃO DO PROBLEMA

Para resolver o problema de classificação utilizamos alguns conceitos matemáticos importantes, tanto no algoritmo KNN, quanto no algoritmo árvore de decisão, a seguir está listados alguns conteúdos abordaddos:

Via KNN

1. Localizar pontos no plano cartesiano
2. Distâncias Euclidianas e distâncias não euclidianas-Minkowski (Manhattan para n=1)

3. Conjuntos/Listas
4. Máximos e mínimos
5. Ordem crescente e decrescente

Via árvore de decisão

1. desigualdades
2. índice Gini
3. árvore de decisão (fluxograma)
4. partição de conjuntos

6.2.1 Principais conteúdos abordados nas atividades

Nessa seção serão apresentados alguns dos principais conteúdos que serão abordados nas atividades, como localização de pontos, distância entre pontos e desigualdades, que são conteúdos importantes e abordados na grade curricular da educação básica.

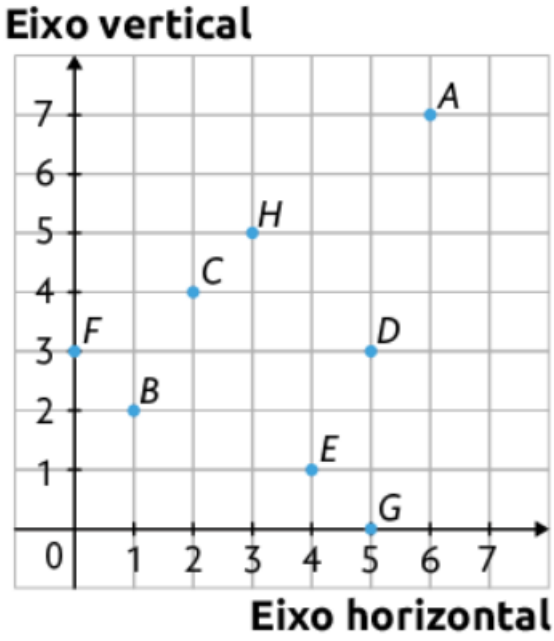
6.2.2 Localização de pontos no plano cartesiano

Esse conteúdo será trabalhado nas duas atividades, mesmo sendo um conteúdo previsto para o ensino fundamental, a localização de pontos é um conteúdo importante para que os alunos desenvolvam outros conteúdos relacionados ao plano cartesiano.

Localização de pontos:

O plano cartesiano, introduzido por René Descartes no século XVII, é uma ferramenta fundamental em matemática, utilizada para representar pontos, linhas e figuras geométricas em duas dimensões. Ele é composto por dois eixos perpendiculares: o eixo das abscissas (eixo X) que é o eixo horizontal, e o eixo das ordenadas (eixo Y) que é o eixo vertical.

Figura 72 – Pontos no plano cartesiano



Fonte: Livro: SuperAção Matemática - 7º Ano TEIXEIRA

Cada ponto no plano cartesiano é representado por um par ordenado (x, y), onde: x indica a posição horizontal do ponto em relação à origem, e y indica a posição vertical do ponto em relação à origem.

Podemos observar o plano cartesiano apresentado anteriormente, o ponto A por exemplo possui posição horizontal x (abscissa) igual a 6 e posição vertical y (ordenada) igual a 7, logo sua localização será representada pelo par ordenado $(6, 7)$. Já o ponto C, possui abscissa igual a 2 e ordenada igual a 4, assim sua localização será $(2, 4)$. Podemos também observar que no ponto F a abscissa será zero e a ordenada 3, sendo assim sua localização será $(0, 3)$. Logo a localização de cada ponto será:

$A(6, 7); B(1, 2); C(2, 4); D(5, 3); E(4, 1); F(0, 3); G(5, 0); H(3, 5)$

Compreender o plano cartesiano e a localização de pontos é essencial para diversas áreas do conhecimento, facilitando a visualização e a análise de informações de maneira clara e precisa.

6.2.3 Distância entre pontos

O conteúdo distância entre pontos será trabalhado na atividade sobre o algoritmo KNN. De acordo com as habilidades estabelecidas pela BNCC, esse conteúdo deve ser trabalho

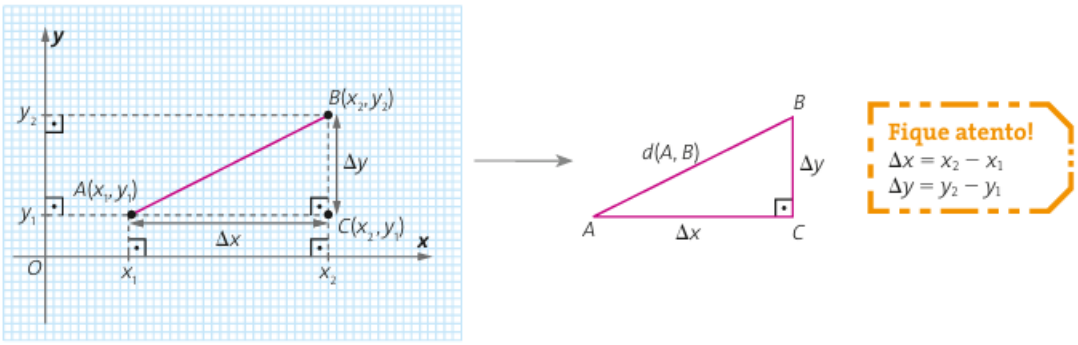
ao longo dos três anos do ensino médio, mas receberá um foco maior no 3° ano do ensino médio.

6.2.3.1 Distância Euclidiana

Sejam dois pontos A e B, a distância entre eles, que será indicada por $d(A, B)$ será a medida do segmento de reta que terá extremidades A e B.

Podemos determinar uma fórmula que indica a distância entre A e B, quaisquer que sejam $A(x_1, y_1)$ e $B(x_2, y_2)$. O triângulo ABC é retângulo em C, logo podemos usar o teorema de Pitágoras, como mostra a figura abaixo:

Figura 73 – Distância entre pontos



Fonte: Livro: Matemática: Contexto e Aplicações - Volume 3 DANTE

Aplicando o Teorema de Pitágoras temos que,

$$[d(A, B)]^2 = (\Delta_x)^2 + (\Delta_y)^2$$

$$d(A, B) = \sqrt{(\Delta_x)^2 + (\Delta_y)^2}$$

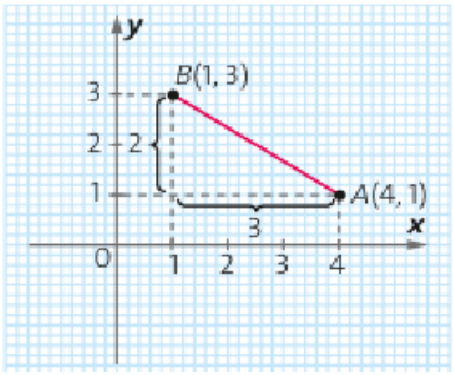
$$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Concluímos, então, que a distância entre dois pontos A e B quaisquer do plano, tal que $A(x_1, y_1)$ e $B(x_2, y_2)$, é dada por:

$$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Tomamos como o exemplo os pontos $A(4, 1)$ e $B(1, 3)$, como apresenta a imagem a seguir:

Figura 74 – Distância entre pontos A e B



Fonte: Livro: Matemática: Contexto e Aplicações - Volume 3 DANTE

Logo a distância entre esses dois pontos será,

$$d(A,B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$d(A,B) = \sqrt{(1-4)^2 + (3-1)^2}$$

$$d(A,B) = \sqrt{(-3)^2 + (2)^2}$$

$$d(A,B) = \sqrt{9+4}$$

$$d(A,B) = \sqrt{13}$$

6.2.3.2 Distância de Manhattan

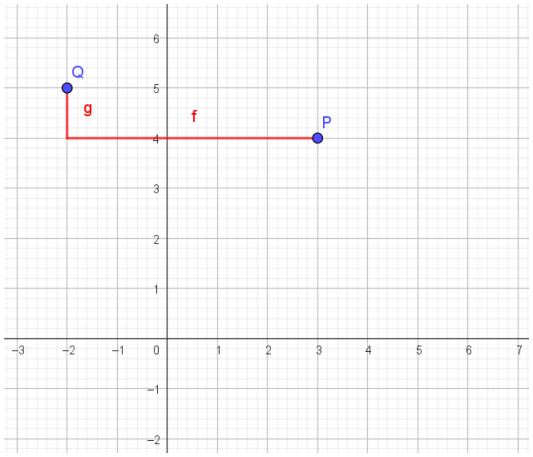
A Distância de Manhattan, também conhecida como Distância do Táxi, é uma métrica utilizada para medir a distância entre dois pontos em um espaço cartesiano. Ao contrário da Distância Euclidiana, que mede a distância mais curta (em linha reta) entre dois pontos, a Distância de Manhattan considera apenas movimentos horizontais e verticais. segundo SILVA; SANTOS a distância de Manhattan foi utilizada para resolver uma situação que envolve a distância que um carro deve realizar, em um percurso entre dois pontos da cidade. Considerando que o carro deve andar somente em ruas virando as esquinas. A Distância de Manhattan entre dois pontos $P_1(x_1, y_1)$ e $P_2(x_2, y_2)$ em um plano cartesiano é dada pela soma das diferenças absolutas das coordenadas correspondentes: $d_Manhattan(P_1, P_2) = |(x_1 - x_2)| + |(y_1 - y_2)|$

Exemplo

Vamos determinar a distância de Manhattan entre os pontos $P(3,4)$ e $Q(-2,5)$. $d_Manhattan(P,Q) = |(3-(-2))| + |(4-5)| = |5| + |(-1)| = 5 + 1 = 6$

Portanto a distância percorrida nesse exemplo será de 6 unidades. Podemos ver no plano cartesiano a seguir a representação dessa distância em destaque:

Figura 75 – Distância Manhattan



Fonte: figura produzida pelo autor.

De modo geral a Distância de Manhattan é uma métrica essencial para medir a distância entre dois pontos em um espaço onde o movimento é restrito a direções ortogonais, como ruas em um percurso urbano. Essa métrica é especialmente relevante em situações onde os deslocamentos não podem ser feitos em linha reta, tornando-a mais prática e intuitiva em certos contextos, como planejamento urbano, robótica, e em algumas situações nos algoritmos de aprendizado de máquina.

6.2.4 Inequação

No caso da inequação, esse conteúdo será abordado principalmente na atividade sobre o algoritmo Árvore de Decisão. Na Base Nacional Comum Curricular (BNCC) do Ensino Médio, o conteúdo de desigualdades é trabalhado principalmente nos últimos anos, que correspondem ao 2º e 3º anos do Ensino Médio, além disso esse conteúdo também é trabalhado no 8° e 9° do ensino fundamental. De acordo com a BNCC, na área de Matemática, os estudantes devem desenvolver habilidades relacionadas à compreensão e resolução de desigualdades.

Inequação Uma inequação é uma expressão matemática que indica que duas quantidades não são iguais, estabelecendo uma relação de ordem entre elas. As inequações utilizam símbolos de desigualdade para mostrar a relação entre as quantidades.

Tipos de inequações:

1. *Inequação Simples:*

- Menor que ($<$): Indica que uma quantidade é menor que outra. Como por exemplo $x < 5$.
- Maior que ($>$): Indica que uma quantidade é maior que outra. Como por exemplo $x > 8$.
- Menor ou igual a ($\leq$): Indica que uma quantidade é menor ou igual a outra. Como por exemplo $x \leq 12$.
- Maior ou igual a ($\geq$): Indica que uma quantidade é maior ou igual a outra. Como por exemplo $x \geq 7$.

2. *Inequação o composta:* São combinações de duas ou mais inequações simples, nesse caso teremos um intervalo de soluções. Veja os exemplos:

   - $1 < x < 5$
   - $-3 \leq x < 4$
   - $8 \leq x \leq 20$

3. *Inequação linear:* Inequações que envolvem variáveis de grau 1, no geral uma inequação de 1° grau são inequações que apresentam a seguinte forma:

   $ax + b > c$ $\quad ax + b < c$ $\quad ax + b \geq c$ $\quad ax + b \leq c$ $\quad$ com $a, b, c \in \mathbb{R}$

   Vejamos os exemplos a seguir:

   - $2x + 3 > 8$
   - $3x - 8 \leq 7$
   - $5x - 6 \geq 9$
   - $10x + 12 < 20$

4. *Inequação quadrática:* Inequações que envolvem variáveis de grau 2, no geral as inequações de 2° grau apresentam as seguintes formas:

   $ax^2 + bx + c > d$ $\quad ax^2 + bx + c < d$ $\quad ax^2 + bx + c \geq d$ $\quad ax^2 + bx + c \leq d$ com $a, b, c, d \in \mathbb{R}$

   Veja os exemplos a seguir:

   - $x^2 + 3x - 2 > 12$
   - $3x^2 - 6 - 1 \leq 9$
   - $2x^2 - 5 \geq -6$
   - $-4x^2 + 4x < 8$

5. *Inequação racional:* As inequações polinomiais, são quaisquer inequações em que um dos termos pode ser escrito como o quociente de dois polinômios, de modo geral temos as seguintes formas de inequações racionais:

   $\frac{p}{q} > c$ $\frac{p}{q} < c$ $\frac{p}{q} \geq c$ $\frac{p}{q} \leq c$ com $c \in \mathbb{R}$, e sendo p e q polinômios.

   A seguir temos alguns exemplos:

   - $\frac{x+2}{x-1} > 3$
   - $\frac{x^2-1}{x+5} \leq 4$
   - $\frac{x^2+x}{x^2+1} < -2$

Resolução de inequações

A resolução de inequações segue passos semelhantes à resolução de equações, com algumas particularidades devido à natureza da desigualdade. Devemos lembrar que ao multiplicar ou dividir ambos os lados por um número negativo, inverte-se o sinal da desigualdade. Além disso é muito importante fazer a análise do sinal corretamente. Vejamos como resolver uma inequação linear e uma inequação quadrática.

Resolução de Inequação de 1° Grau

Para resolver uma inequação de 1° grau devemos seguir dois passos:

1- isolar a incógnita;

2- Analisar o sinal.

Vamos ver como resolver a inequação $2x - 3 > 1$. Isolando a incógnita teremos,

$2x - 3 > 1$

$2x - 3 + \mathbf{3} > 1 + \mathbf{3}$

$2x > 4$

$\frac{2x}{\mathbf{2}} > \frac{4}{\mathbf{2}}$

$x > 2$

Neste caso representamos o conjunto solução com "‘bolinha aberta"em 2, pois o mesmo não está inserido nas soluções da inequação.

Vejamos agora como resolver a inequação $-3x + 3 \leq x + 11$.

$-3x + 3 \leq x + 11$

$-3x + 3 \mathbf{-3} \leq x + 11 \mathbf{-3}$

$-3x \leq x + 8$

$-3x \mathbf{-x} \leq x + 8 \mathbf{-x}$

$-4x \leq 8$

Neste caso como o coeficiente da incógnita é negativo, devemos multiplicar a inequação por -1, isso faz com que todos os sinais da inequação mudem de forma equivalente, inclusive a desigualdade.

$-4x \leq 8$**.(-1)**

$4x \geq -8$

$\frac{4x}{4} \geq \frac{-8}{4}$

$x \geq -2$

Observe que neste caso temos 'bolinha fechada"' em x=-2, pois o -2 está incluído nas soluções desta inequação.

Resolução de Inequação de 2° Grau

Resolver uma inequação de 2° grau envolve determinar os valores de x que satisfazem a desigualdade dada. para isso devemos seguir os seguintes passos:

1- Escrever a inequação em sua forma padrão $ax^2 + bx + c > 0$. (ou$<, \geq, \leq$)

2- Determinar os zeros da função associada à inequação.

3- Analisar o sinal da função associando a desigualdade com o gráfico da função.

Vamos determinar a solução da inequação $x^2 + 2x - 3 < 0$.

Primeiramente determinaremos os zeros da função $y = x^2 + 2x - 3$, logo teremos,

$x^2 + 2x - 3 = 0$

$\Delta = 2^2 - 4.1.(-3)$

$\Delta = 4 + 12$

$\Delta = 16$

Logo temos que

$x = \frac{-2 \pm \sqrt{16}}{2.1}$

$x = \frac{-2 \pm 4}{2}$

desta forma temos que

$x_1 = \frac{-2+4}{2}$

$x_1 = \frac{2}{2}$

$x_1 = 1$

e

$x_2 = \frac{-2-4}{2}$

$x_2 = \frac{-6}{2}$

$x_2 = -3$

Analisando o sinal dessa função temos a seguinte parábola:

Como a inequação é $x^2 + 2x - 3 < 0$, teremos como solução todos os valores de x em que $y < 0$, logo a solução dessa inequação será,

$S = (x \in \mathbb{R} / -3 < x < 1)$

Vejamos agora como resolver a inequação $-x^2 + 3x + 4 \leq 0$.

Determinando os zeros da função $y = -x^2 + 3x + 4$ associada a inequação teremos,
$-x^2 + 3x + 4 = 0$
$\Delta = 3^2 - 4.(-1).4$
$\Delta = 9 + 16$
$\Delta = 25$
Logo pela fórmula resolutiva,
$x = \frac{-3 \pm \sqrt{25}}{2.(-1)}$
$x = \frac{-3 \pm 5}{-2}$
Assim teremos que,
$x_1 = \frac{-3+5}{-2}$
$x_1 = \frac{2}{-2}$
$x_1 = -1$
e
$x_2 = \frac{-3-5}{-2}$
$x_2 = \frac{-8}{-2}$
$x_2 = 4$
Analisando o sinal dessa função teremos a seguinte parábola:
Como a inequação é $-x^2 + 3x + 4 \leq 0$, teremos como solução todos os valores de x em que $y \leq 0$, logo a solução dessa inequação será,
$S = (x \in \mathbb{R} / x \leq -1 ou x \geq 4)$

6.2.5 O índice GINI

O índice Gini é amplamente conhecido como uma medida de desigualdade econômica, mas também desempenha um papel fundamental em algoritmos de aprendizado de máquina, especificamente na construção de árvores de decisão. Neste contexto, o índice Gini é utilizado como um critério de seleção para dividir os nós da árvore de decisão, ajudando a determinar a "pureza"dos nós. Em algoritmos de árvores de decisão, o índice Gini é utilizado como um critério de impureza. A ideia é selecionar as divisões que aumentem ao máximo a "pureza"dos nós resultantes. Quanto mais puro for um nó, melhor ele representa uma única classe. Segundo ONODA; EBECKEN para descobrir a melhor quebra para um nó são avaliadas todas as quebras dos atributos, sendo escolhido para dividir o nó o atributo que contém o ponto de quebra com o menor valor de índice Gini.

Para um conjunto de dados, o índice Gini é calculado como:
$Gini\ (D) = 1 - \sum_{i=1}^{c} {p_i}^2$
Onde:

- D é o conjunto de dados
- C é o número de classes.

- p_i é a frequência relativa de cada classe em cada nó.

O índice Gini determina valores entre 0 e 1, devemos analisar a divisão no seguinte parâmetro, quando este índice é igual a zero, o nó é totalmente puro, e quando ele for igual a um, o nó será totalmente impuro, ou seja, quanto mais próximo de zero mais puro será o nó da árvore de decisão.

Para determinar o índice devemos calcular da seguinte maneira:

1. Calcular o índice Gini do conjunto de dados inteiro antes de qualquer divisão.
2. Testar todas as possíveis divisões dos dados em dois subgrupos com base em um atributo e seu valor e para cada possível divisão, calcular o índice Gini dos dois subgrupos resultantes.
3. Calcular a diferença na impureza entre o conjunto de dados original e os subgrupos resultantes. para isso utilizamos a fórmula a seguir:

$\Delta Gini = Gini(D) - (\frac{N_1}{N} \times Gini(D_1) + \frac{N_2}{N} \times Gini(D_2))$
Onde:

- Gini(D) é o índice Gini do conjunto de dados original.
- $Gini(D_1)$ e $Gini(D_2)$ são os índices Gini dos dois subgrupos após a divisão.
- N_1 e N_2 são os tamanhos dos dois subgrupos.
- N é o tamanho total do conjunto de dados.

Vejamos a seguir um exemplo para melhorar o entendimento sobre o cálculo do índice Gini.

Suponha um conjunto de dados com a seguinte distribuição de classes:

Classe A: 40% = 0,4
Classe B: 60% = 0,6
Logo o índice Gini para o conjunto de dados D será,

$Gini(D) = 1 - [(0.4)^2 + (0,6)^2]$
$Gini(D) = 1 - (0.16 + 0.36)$
$Gini(D) = 1 - 0.52$
$Gini(D) = 0.48$

Agora vamos considerar que em uma determinada divisão, obtemos os seguintes subgrupos:

$Subgrupo D_1 : 70\% Classe A, 30\% Classe B.$
$Subgrupo D_2 : 10\% Classe A, 90\% Classe B.$
Calcularemos então o índice Gini para cada subgrupo,

$Gini(D_1) = 1 - [(0.7)^2 + (0,3)^2]$
$Gini(D_1) = 1 - [0,49 + 0,09]$
$Gini(D_1) = 1 - 0,58$
$Gini(D_1) = 0,42$

$Gini(D_2) = 1 - [(0.1)^2 + (0,9)^2]$
$Gini(D_2) = 1 - [0,01 + 0,81]$
$Gini(D_2) = 1 - 0,82$
$Gini(D_2) = 0,18$

Agora calcularemos a diferença na impureza entre o conjunto de dados original (D) e os subgrupos (D_1 e D_2).

$\Delta Gini = Gini(D) - (\frac{N_1}{N} \times Gini(D_1) + \frac{N_2}{N} \times Gini(D_2))$
$\Delta Gini = 0,48 - (\frac{N_1}{N} \times 0,42 + \frac{N_2}{N} \times 0,18)$

Se $N_1 e N_2$ forem, por exemplo, iguais (ou seja, cada subgrupo contém metade dos dados originais), teremos,

$\Delta Gini = 0,48 - (0,5 \times 0,42 + 0,5 \times 0,18)$
$\Delta Gini = 0,48 - (0,21 + 0,09)$
$\Delta Gini = 0,48 - 0,3$
$\Delta Gini = 0,18$

Assim, esta divisão resultaria em uma redução de impureza de 0.18, tornando-se uma possível escolha para a divisão.

No contexto de árvores de decisão, o índice Gini serve como um critério para avaliar a qualidade das divisões dos nós, buscando melhorar a pureza dos nós resultantes. Ao selecionar a divisão que minimiza a impureza, o índice Gini ajuda a construir uma árvore de decisão eficiente e eficaz para a classificação dos dados.

6.3 RECURSO DIDÁTICO E TECNOLÓGICO COM GEOGEBRA

O Geogebra é um software de matemática dinâmica que combina geometria, álgebra, cálculo, planilhas, gráficos e estatísticas em um único ambiente interativo. Ele permite que

os usuários explorem e visualizem conceitos matemáticos de forma dinâmica, manipulando objetos geométricos e representações algébricas em tempo real. O Geogebra é amplamente utilizado por estudantes, professores e profissionais da matemática em todo o mundo para criar demonstrações, resolver problemas, investigar conjecturas e visualizar relações matemáticas complexas. Além disso, sua interface intuitiva e recursos poderosos o tornam uma ferramenta valiosa para o ensino e aprendizado de matemática em todos os níveis de escolaridade.

Buscando aproximar os alunos do ensino médio de algumas técnicas de aprendizagem de máquina, foi desenvolvidos atividades com o auxílio do Geogebra para facilitar a compreensão dos alunos em relação ao funcionamento dos algoritmos de predição KNN e Árvore de decisão. A seguir serão apresentadas as atividades para serem trabalhadas com os alunos e como devem ser conduzidas.

6.3.1 Algoritmo KNN

Antes de abordarmos a atividade é necessário lembrarmos algumas características importantes sobre o algoritmo KNN. O algoritmo KNN é um método de aprendizado supervisionado utilizado para classificação e regressão. Ele opera de forma bastante simples:

Treinamento: Primeiro, o algoritmo armazena todos os pontos de dados e suas classes associadas (no caso de classificação) ou valores (no caso de regressão).

Distância: Quando é feita uma previsão para um novo ponto de dados, o algoritmo calcula a distância entre esse ponto e todos os pontos de dados do conjunto de treinamento. A métrica de distância mais comumente usada é a distância euclidiana, mas outras métricas também podem ser usadas, dependendo do contexto.

Vizinhos mais próximos: O algoritmo então seleciona os k pontos de dados mais próximos ao novo ponto, onde k é um parâmetro especificado pelo usuário.

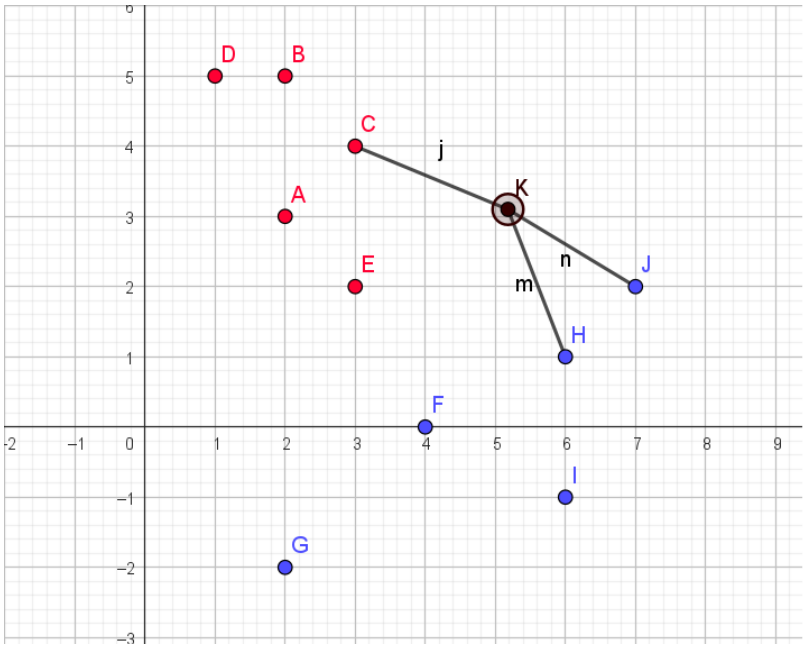
Classificação ou Regressão: Para problemas de classificação, o algoritmo atribui a classe mais comum entre os k vizinhos mais próximos ao novo ponto como a classe prevista para esse ponto. No caso de regressão, ele calcula a média dos valores alvo dos k vizinhos mais próximos e usa esse valor como a previsão.

O KNN é um algoritmo de aprendizado de máquina não paramétrico, o que significa que não faz suposições explícitas sobre a forma funcional dos dados. Ele simplesmente memoriza os dados de treinamento e faz previsões com base na similaridade entre novos pontos de dados e os pontos de dados já conhecidos.

De modo geral o algoritmo KNN classifica um dado de acordo com a similaridade com os n vizinhos mais próximos, por exemplo na imagem a seguir, temos a ilustração do

processo de classificação.

Figura 76 – Classificação de um ponto K utilizando o algoritmo KNN.



Fonte: Imagem produzida pelo autor.

Nesse exemplo os pontos foram separados em dois conjuntos de cinco pontos, Vermelho (A, B, C, D e E) e Azul (F, G, H, I e J), o ponto K é o ponto que devemos classificar, nesse caso o algoritmo KNN foi utilizado para classificar o ponto K com base na similaridade com os três pontos mais próximos. Como é possível observar para essa quantidade de vizinhos mais próximos podemos classificar o ponto K como pertencente ao conjunto Azul, pois dois dos pontos mais próximos, H e G, pertencem ao conjunto azul, e apenas um ponto entre os mais próximos, ponto C, pertence ao conjunto Vermelho.

Vemos aqui a importância da distância entre pontos, assim como apresenta MURPHY em seu livro, ele mostra que a distância euclidiana é frequentemente usada, mas que outras métricas de distância, como a distância de Manhattan e a distância de Minkowski, podem ser utilizadas dependendo do problema e dos dados.

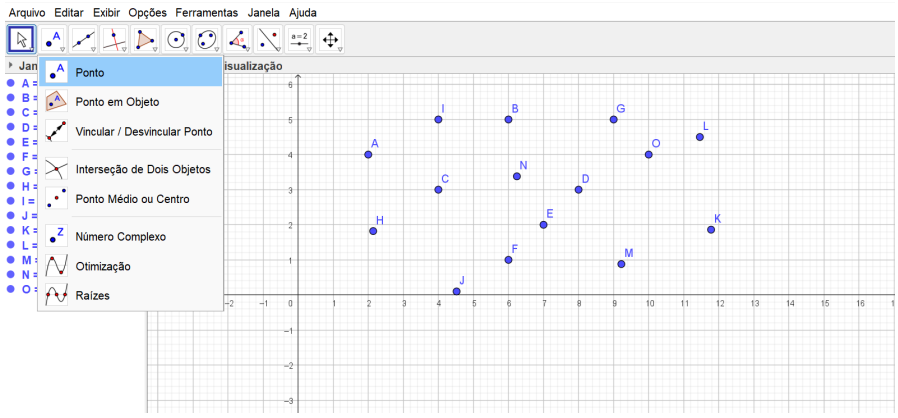
A atividade proposta a seguir, apresenta uma abordagem para duas métricas, a distância euclidiana e a distância de Manhattan.

6.3.1.1 Atividade complementar no GeoGebra, utilizando distância Euclidiana

Essa atividade complementar será utilizada para o compreender alguns conceitos básicos sobre o algoritmo de aprendizado de máquina KNN, além disso essa atividade será importante para os alunos manipularem e aprederem alguns comandos do Geogebra.

Inicialmente os alunos deverão criar 15 pontos no Geogebra. O professor deve orientar os alunos a selecionarem a opção "Ponto"no Geogebra, criando assim 15 pontos da maneira que preferirem.

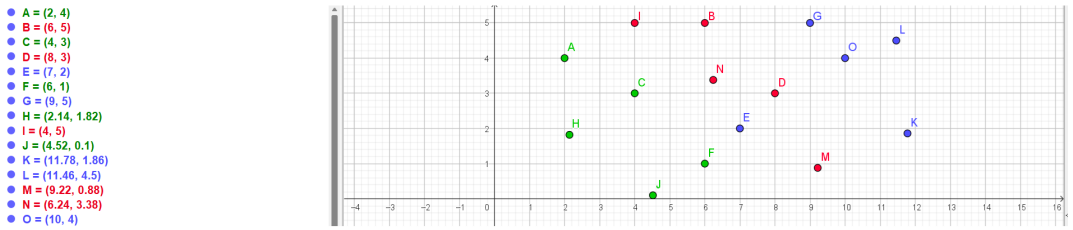
Figura 77 – Criando pontos no Geogebra



Fonte: Imagem produzida pelo o autor.

Quando os alunos determinam pontos no Geogebra o próprio programa gera os pares ordenados desses pontos, nesse momento os alunos deverão anotar no caderno os pares ordenados de cada ponto e compará-los com os pares ordenados gerados pelo Geogebra.

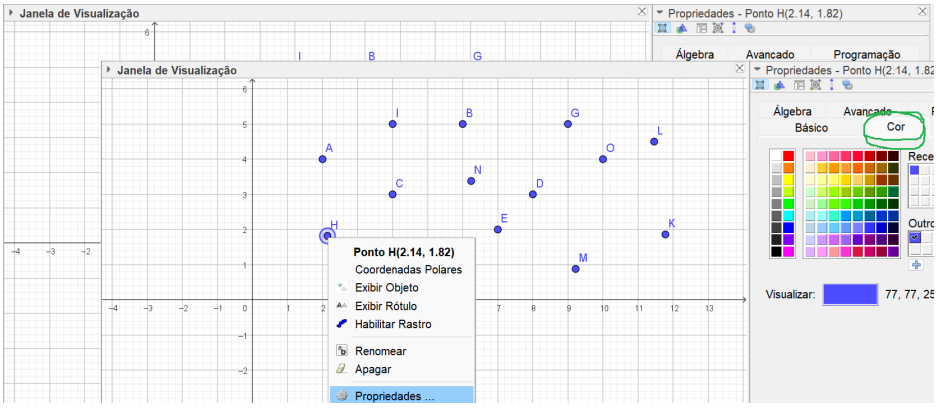
Figura 78 – Pares Ordenados no Geogebra



Fonte: Imagem produzida pelo o autor.

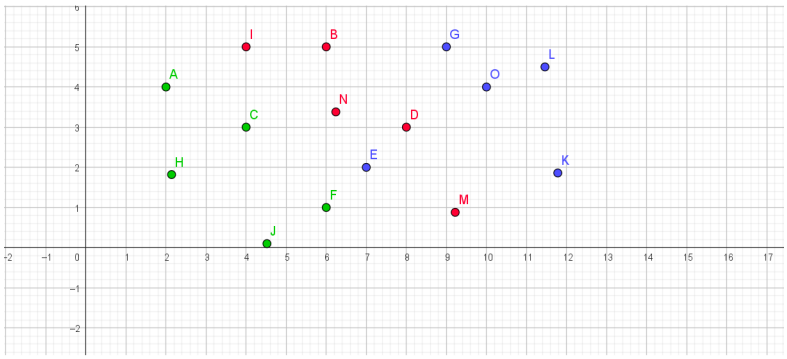
Agora os alunos deverão separar esses 15 pontos em três grupos de 5 pontos, separados por cores diferentes, os próprios alunos podem escolher quais pontos formarão o conjunto de pontos azul, verde e vermelho. Para mudar a cor dos pontos os alunos devem clicar no ponto com o botão direito do mouse e selecionar a opção "Propriedades"nessa janela o aluno deverá selecionar o item "cor"e separar cinco pontos com cor azul, cinco vermelhos e cinco verdes. Como mostra o exemplo a seguir:

Figura 79 – Selecionando a cor dos pontos



Fonte: Imagem produzida pelo o autor.

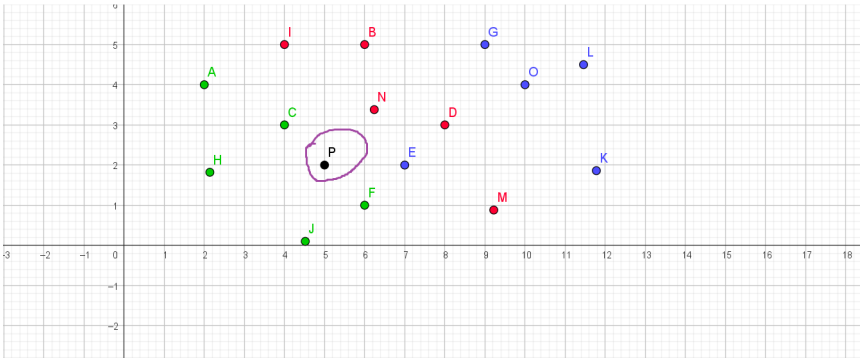
Figura 80 – Conjuntos de pontos selecionados por cores



Fonte: Imagem produzida pelo o autor.

Após a criação dos conjuntos de pontos, os alunos deverão criar um ponto P que será o ponto que faremos a classificação de acordo com o algoritmo KNN. Novamente os alunos deverão selecionar a opção "Ponto"e criar um ponto no local que desejar. Para que não haja confusão é interessante que nesse primeiro momento o ponto P esteja de cor diferente dos outros, posteriormente iremos programar para que o ponto fique com cor de acordo com sua classificação.

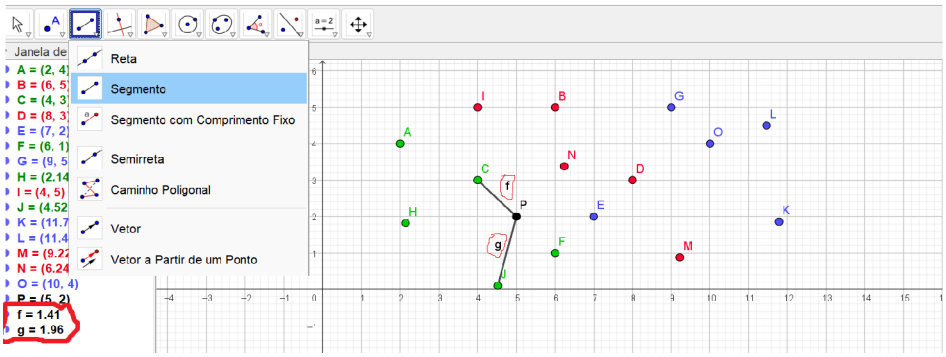
Figura 81 – Ponto P



Fonte: Imagem produzida pelo o autor.

Como algoritmo KNN classifica os pontos de acordo com as distâncias mínimas os alunos deverão determinar as distâncias entre o ponto P e os demais pontos, utilizaremos as três distância mais próximas ($k = 3$) para classificar o ponto, algumas dessas distâncias serão primeiramente calculadas pelos alunos no caderno, depois no Geogebra podemos calcular todas as distância. Iremos calcular as distâncias utilizando os segmentos de retas entre os pontos e o ponto P, os alunos deverão determinar todos os quinze segmentos entre os pontos e o ponto P. Os alunos primeiro criarão segmentos de retas entre dois pontos, desta forma o Geogebra calculará a distância entre os pontos, os alunos deverão selecionar a janela "Retas"' e nessa janela selecionar o item "Segmento", feito isso basta selecionar o ponto P e um dos pontos dos conjuntos de pontos criados, assim o segmento de reta será criado entre esses pontos e na janela de Álgebra aparecerá a distância entre esses pontos, como mostra a figura a seguir:

Figura 82 – Distância entre os pontos utilizando o segmento, $d(P, C) = f$ e $d(P, J) = g$

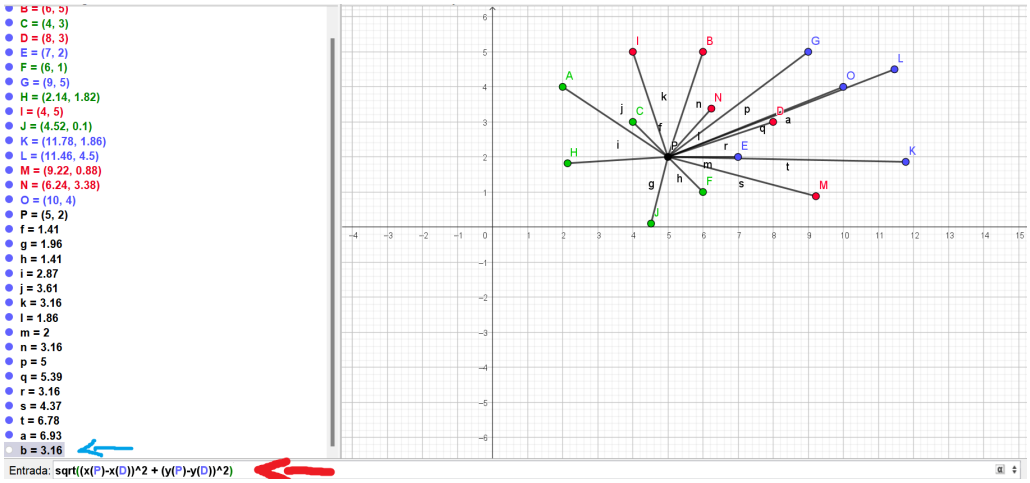


Fonte: Imagem produzida pelo o autor.

Podemos também determinar a distância entre pontos utilizando a fórmula de distância Euclidiana, no Geogebra o aluno deve digitar na barra de entrada a distância entre o ponto

P e um dos pontos dos conjuntos de ponto, deta forma eles verão que o Geogebra também calcula a distância utilizando a fórmula da distância. No exemplo a seguir foi determinado a distância entre o ponto P e o ponto D, que teve medida representa por b no Geogebra, assim na barra de entrada deve ser digitado $sqrt((x(P) - x(D))^2 + (y(P) - y(D))^2)$, lembrando que o comando "$sqrt$"determina a raiz quadrada.

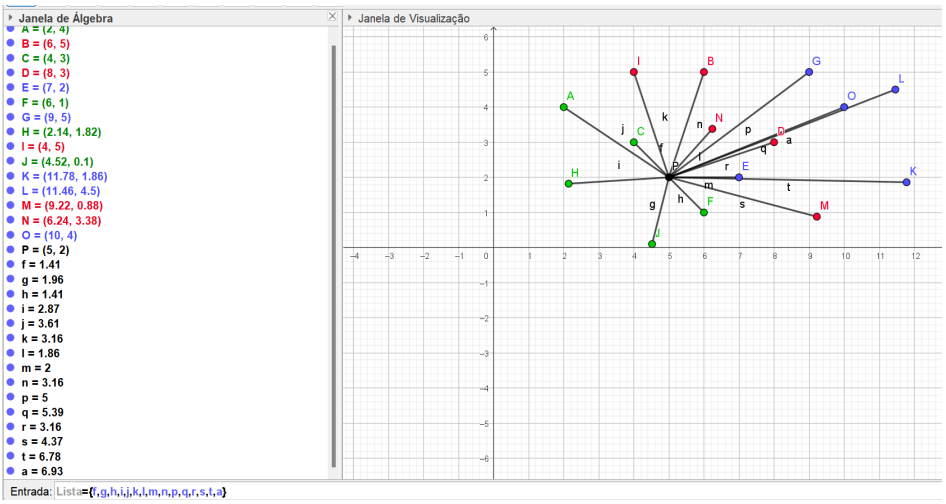
Figura 83 – Distância entre os pontos utilizando a fórmula euclidiana, $d(P, d) = b$



Fonte: Imagem produzida pelo o autor.

Após calcular todas as quinze distâncias os alunos devem determinar quais são as três distâncias mínimas, eles podem fazer essa análise no caderno. Feito isso no Geogebra os alunos também podem determinar as distâncias mínimas, para isso eles devem inicialmente criar uma lista com todos as distâncias encontradas na etapa anterior. para criar a lista os alunos devem digitar na barra de entrada $Lista = \{f, g, h, i, j, k, l, m, n, p, q, r, s, t, a\}$, sendo $\{f, g, h, i, j, k, l, m, n, p, q, r, s, t, a\}$ as distâncias determinadas anteriormente.

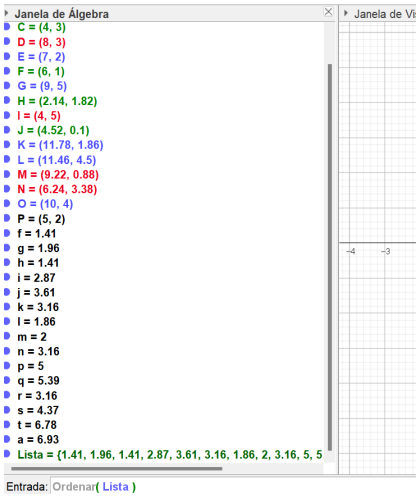
Figura 84 – Criando lista da distâncias



Fonte: Imagem produzida pelo o autor.

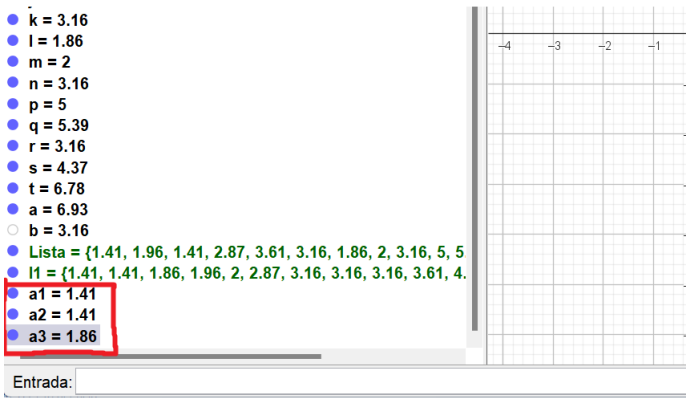
Para melhor análise os alunos podem ainda ordenar a lista, colocando as distâncias em ordem crescente, para isso devem digitar o comando "Ordenar(Lista)". Assim os alunos podem selecionar as menores distâncias. No Geogebra os alunos poderão selecionar as menores distâncias selecionando as três primeiras medidas da lista que no exemplo foi denominada "$l1$"que foi a lista criada quando ordenamos a lista das distâncias, para isso os alunos devem digitar na barra de entrada do Geogebra os comandos, $a1 = l1(1)$, $a2 = l1(2)$, $a3 = l1(3)$. Desta forma selecionaremos as três distâncias mínimas, como mostra a imagem.

Figura 85 – Criando lista ordenada das distâncias (l1)



Fonte: Imagem produzida pelo o autor.

Figura 86 – Distâncias mínimas, $a1, a2, a3$

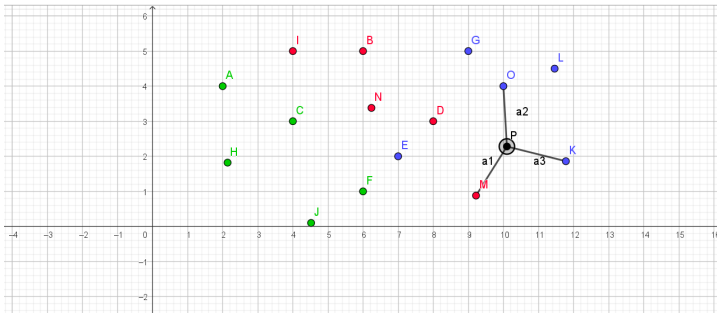


Fonte: Imagem produzida pelo o autor.

Desta forma os alunos devem esconder todas as distâncias deixando apenas as distâncias mínimas, e ao movimentar o ponto P, sempre aparecerão as três menores distâncias. Assim

os alunos poderão classificar os pontos conforme o conjunto de cores que possui pontos mais próximos, de acordo com o algoritmo KNN.

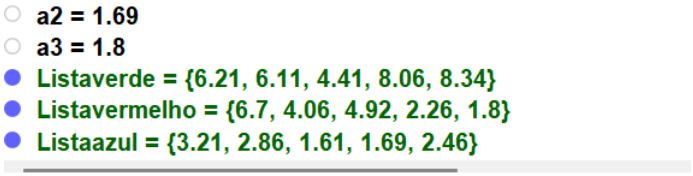
Figura 87 – Distâncias mínimas entre os conjuntos de pontos e o ponto P



Fonte: Imagem produzida pelo o autor.

Para melhor visualização do ponto P, podemos programar o Geogebra para que o ponto fique com a cor de acordo com a sua classificação. Para isso vamos construir inicialmente três listas, uma lista com as distâncias $\{f, g, h, i, j\}$, que são as distâncias entre os pontos verdes e o ponto P, para essa lista os alunos devem digitar na barra de entrada "$Listaverde = \{f, g, h, i, j\}$", outra lista com as distâncias $\{k, l, n, r, s\}$ entre os pontos vermelhos e o ponto P, para essa lista os alunos deverão digitar o comando "$Listavermelha = \{k, l, n, r, s\}$", e para as distâncias $\{m, p, q, t, a\}$, entre os pontos azuis e o ponto P, deverá ser digitado na barra de entrada o comando "$Listaazul\ \{m, p, q, t, a\}$".

Figura 88 – Listas das distâncias de cada conjunto de pontos

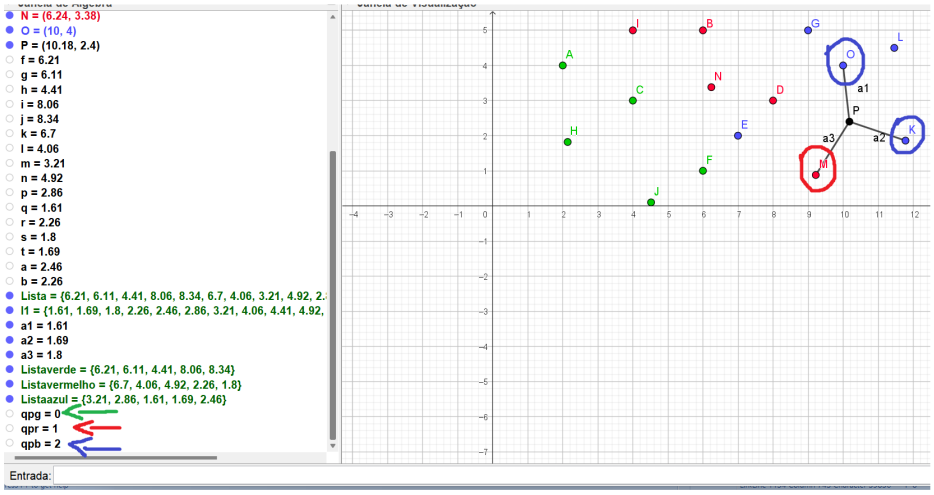


Fonte: Imagem produzida pelo o autor.

Após a criação das listas de cada conjunto de cores os alunos criarão no Geogebra um comando que determina a quantidade de pontos de cada cor que as distâncias mínimas (a1, a2, a3) representam. Para representar a quantidade de pontos verdes utilizaremos qpg, os alunos deverão digitar o comando $qpr = ContarSe(x \leq a3, Listaverde)$, para a quantidade de pontos vermelhos, utilizaremos o comando $qpr = ContarSe(x \leq a3, Listavermelho)$, e para a a quantidade de pontos azuis utilizaremos qpb, os alunos deverão digitar na barra de entrada $qpb = ContarSe(x \leq a3, Listaazul)$. Esse comando faz com que o Geogebra

conte quantas distâncias de cada lista, são menores ou iguais a $a3$, desta forma o Geogebra contará apenas as três distâncias mínimas.

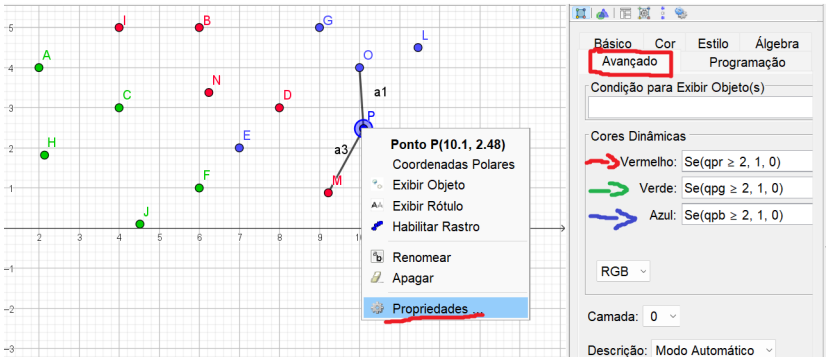
Figura 89 – Quantidade de pontos verdes, vermelho e azuis (qpg, qpr, qpb), de acordo as menores distâncias.



Fonte: Imagem produzida pelo o autor.

Para determinar a cor do ponto P de acordo com a sua classificação, os alunos devem clicar no ponto P com o lado direito do mouse e selecionar a opção propriedades,depois devem selecionar a janela "Avançado", nessa janela teremos a opção de "cores dinâmicas"onde os alunos deverão digitar os seguintes comando: Em Vermelho devem digitar $Se(qpr \geq 2, 1, 0)$; Em Verde devem digitar $Se(qpg \geq 2, 1, 0)$; e em Azul devem digitar o comando $Se(qpb \geq 2, 1, 0)$. Esse comandos definem que o ponto será da cor vermelha, verde ou azul, se a quantidade pontos mais próximos for igual ou maior que dois. A imagem a seguir apresenta essa programação.

Figura 90 – Determinando a cor do ponto P de acordo com a sua classificação



Fonte: Imagem produzida pelo o autor.

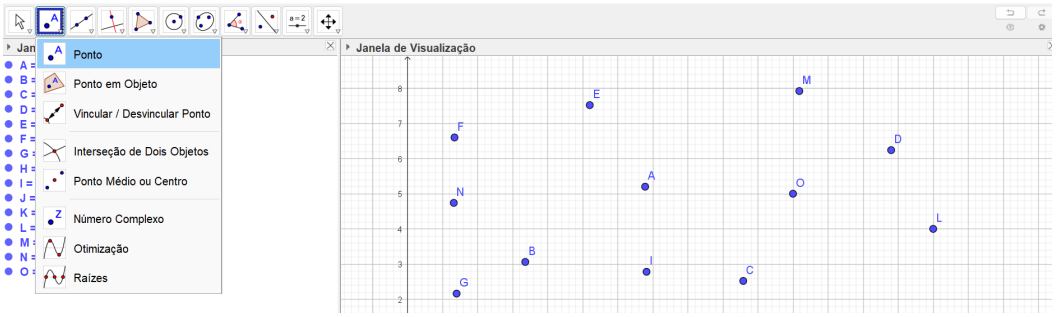
Desta forma os alunos poderão classificar o ponto P de acordo com o algoritmo KNN com $k = 3$, ou seja utilizando as três distâncias euclidianas mínimas.

6.3.1.2 Atividade complementar no GeoGebra, utilizando distância de Manhattan

Assim como a atividade anterior essa atividade complementar será utilizada para o compreender alguns conceitos básicos sobre o algoritmo de aprendizado de máquina KNN, utilizando a distância de Manhattan, assim como também será importante para que os alunos aprendam alguns comandos do Geogebra.

Inicialmente os alunos deverão criar quinze pontos no Geogebra. Para isso os alunos devem selecionar a opção "Ponto"no Geogebra, criando os quinze pontos como preferirem.

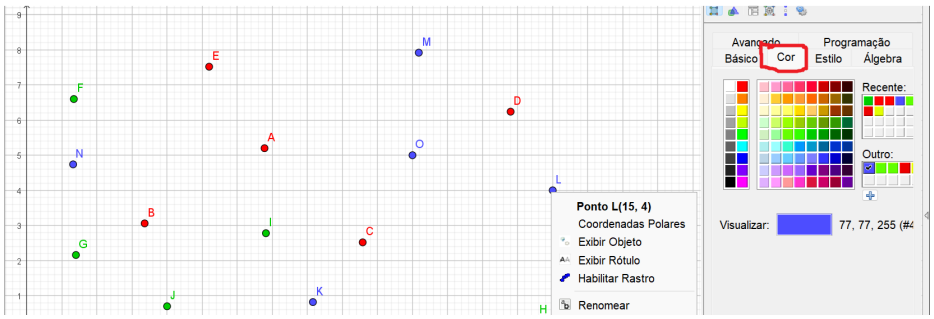
Figura 91 – Criando pontos no Geogebra



Fonte: Imagem produzida pelo o autor.

Agora os alunos deverão separar esses quinze pontos em três grupos de cinco pontos, sendo os grupos de pontos separados pelas cores azul, verde e vermelho. Os alunos podem escolher quais pontos formarão o conjunto de pontos azul, verde e vermelho. Para mudar a cor dos pontos os alunos devem clicar no ponto com o botão direito do mouse e selecionar a opção "Propriedades"nessa janela o aluno deverá selecionar o item "cor"e separar cinco pontos com cor azul, cinco vermelhos e cinco verdes, assim como foi feito na atividade anterior.

Figura 92 – Selecionando a cor dos pontos

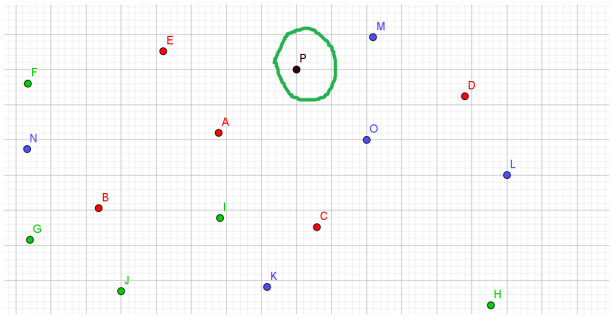


Fonte: Imagem produzida pelo o autor.

Escolhidos os conjuntos de pontos separados por cores, os alunos deverão criar um ponto P que será o ponto que faremos a classificação de acordo com o algoritmo KNN.

Novamente os alunos deverão selecionar a opção "Ponto"e criar o ponto P na localização que desejarem.

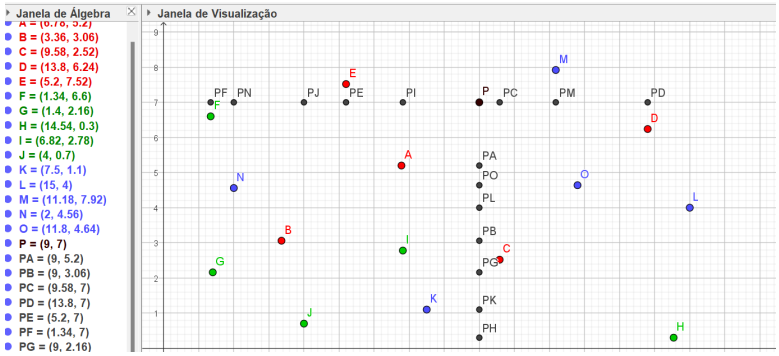
Figura 93 – Ponto P



Fonte: Imagem produzida pelo o autor.

Para determinar a distância Manhattan devemos inicialmente criar pontos auxiliares, esse ponto auxiliar deverá ter a abscissa do ponto P e a ordenada dos pontos de cada conjunto de pontos, ou vice e versa, para melhor visualização podemos misturar essa escolha de abscissa r ordenada, ora a abscissa será do ponto P, ora será de um ponto dos conjuntos. Para determinar por exemplo o ponto auxiliar entre o ponto P e o ponto A, os alunos deverão digitar na barra de entreda do Geogebra o comando $PA = (x(P), y(A))$, desta forma os alunos deverão determinar todos os pontos auxiliares.

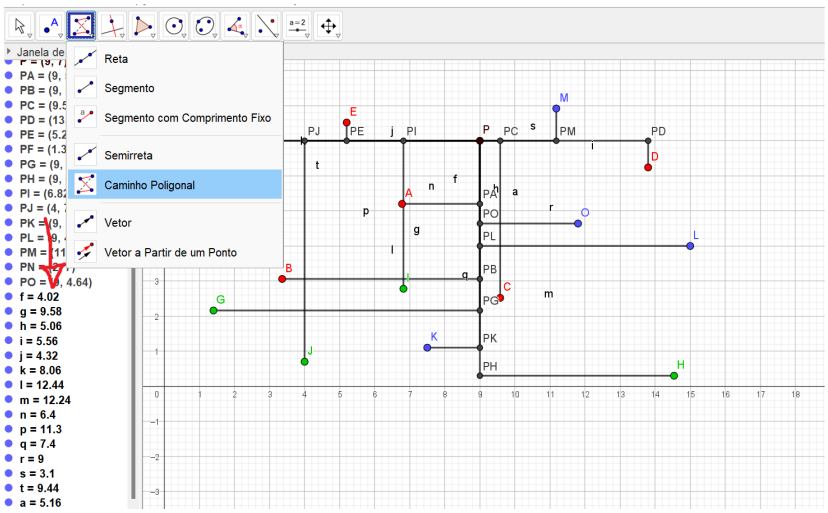
Figura 94 – Criação dos pontos auxiliares



Fonte: Imagem produzida pelo o autor.

Após criarem os pontos auxiliares os alunos criarão um caminho do ponto P a cada um dos pontos de cada conjunto de cor, para isso eles devem clicar na janela "retas"e selecionar a opção "caminho poligonal"e selecionar os pontos P, um ponto auxiliar e um ponto do conjunto de cores correspondente, desta forma criaremos esse caminho, e o próprio Geogebra determinará essa distância entre o ponto P e o ponto do conjunto de cores, no exemplo essas distâncias foram representadas por $\{f, g, h, i, j, k, l, m, n, p, q, r, s, t, a\}$.

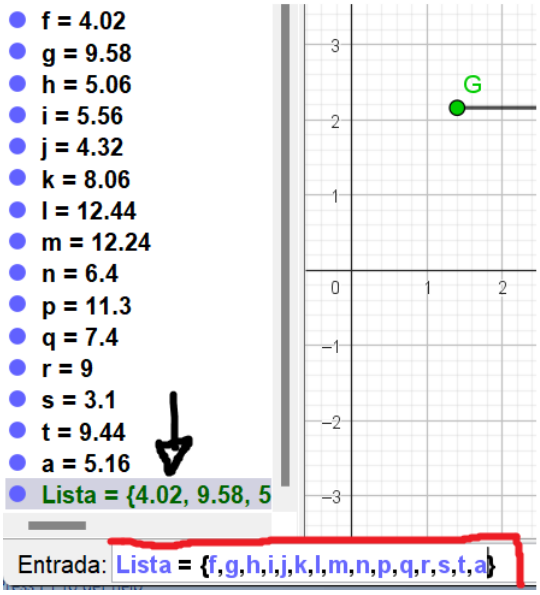
Figura 95 – Construção dos caminhos poligonais, e determinação das distâncias de Manhattan $\{f, g, h, i, j, k, l, m, n, p, q, r, s, t, a\}$



Fonte: Imagem produzida pelo o autor.

Após calcular todas as quinze distâncias os alunos devem determinar quais são as três distâncias mínimas, eles podem fazer essa análise no caderno. Feito isso no Geogebra os alunos também podem determinar as distâncias mínimas, para isso eles devem inicialmente criar uma lista com todos as distâncias encontradas na etapa anterior. para criar a lista os alunos devem digitar na barra de entrada $Lista = \{f, g, h, i, j, k, l, m, n, p, q, r, s, t, a\}$, sendo $\{f, g, h, i, j, k, l, m, n, p, q, r, s, t, a\}$ as distâncias determinadas anteriormente.

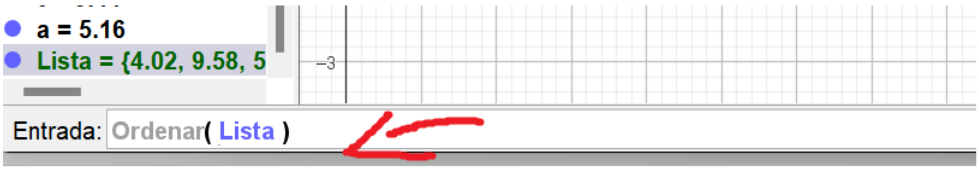
Figura 96 – Criando lista da distâncias de Manhattan



Fonte: Imagem produzida pelo o autor.

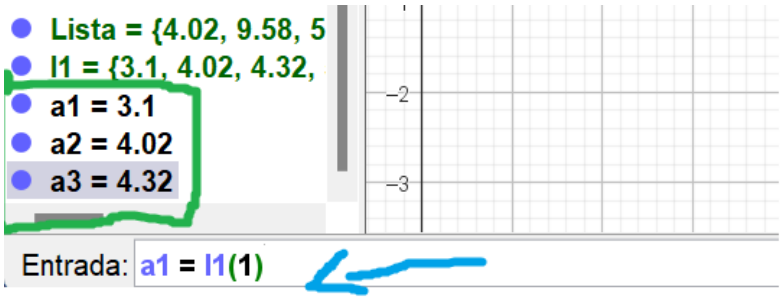
Para determinar as distâncias mínimas os alunos devem ordenar a lista, colocando as distâncias em ordem crescente, para isso devem digitar o comando "Ordenar(Lista)"'. Assim os alunos criarão uma lista "'l1"no Geogebra, no qual os alunos poderão selecionar as três primeiras medidas dessa lista que foi a lista, para isso os alunos devem digitar na barra de entrada do Geogebra os comandos, $a1 = l1(1)$, $a2 = l1(2)$, $a3 = l1(3)$. Desta forma selecionaremos as três distâncias mínimas, como mostra a imagem.

Figura 97 – Criando lista ordenada das distâncias de Manhattan (l1)



Fonte: Imagem produzida pelo o autor.

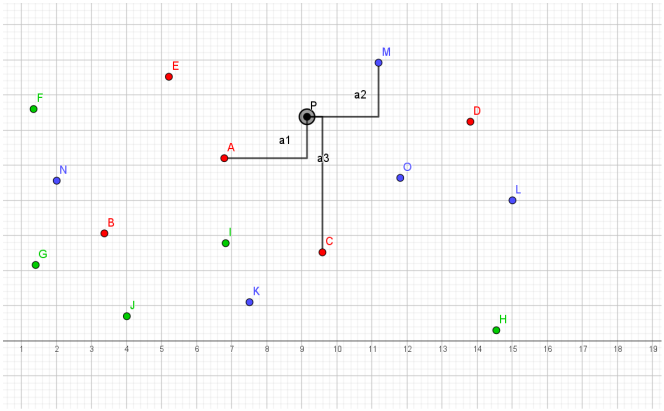
Figura 98 – Distâncias mínimas, $a1, a2, a3$



Fonte: Imagem produzida pelo o autor.

Feito isso os alunos devem esconder todas as distâncias deixando apenas as distâncias mínimas, e ao movimentar o ponto P, sempre aparecerão as três menores distâncias. Assim os alunos poderão classificar os pontos conforme o conjunto de cores que possui pontos mais próximos, de acordo com o algoritmo KNN.

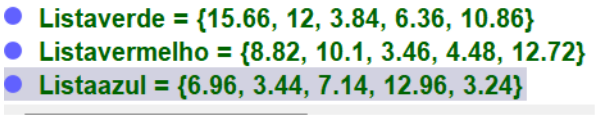
Figura 99 – Distâncias mínimas (Manhattan) entre os conjuntos de pontos e o ponto P



Fonte: Imagem produzida pelo o autor.

Para melhor visualização do ponto P, podemos programar o Geogebra para que o ponto fique com a cor de acordo com a sua classificação. Para isso vamos construir inicialmente três listas, uma lista com as distâncias $\{k, l, m, n, p\}$, que são as distâncias entre os pontos verdes e o ponto P, para essa lista os alunos devem digitar na barra de entrada "$Listaverde = \{k, l, m, n, p\}$", outra lista com as distâncias $\{f, g, h, i, j\}$ entre os pontos vermelhos e o ponto P, para essa lista os alunos deverão digitar o comando "$Listavermelha = \{f, g, h, i, j\}$", e para as distâncias $\{q, r, s, t, a\}$, entre os pontos azuis e o ponto P, deverá ser digitado na barra de entrada o comando "$Listaazul\ \{q, r, s, t, a\}$".

Figura 100 – Listas das distâncias de cada conjunto de pontos

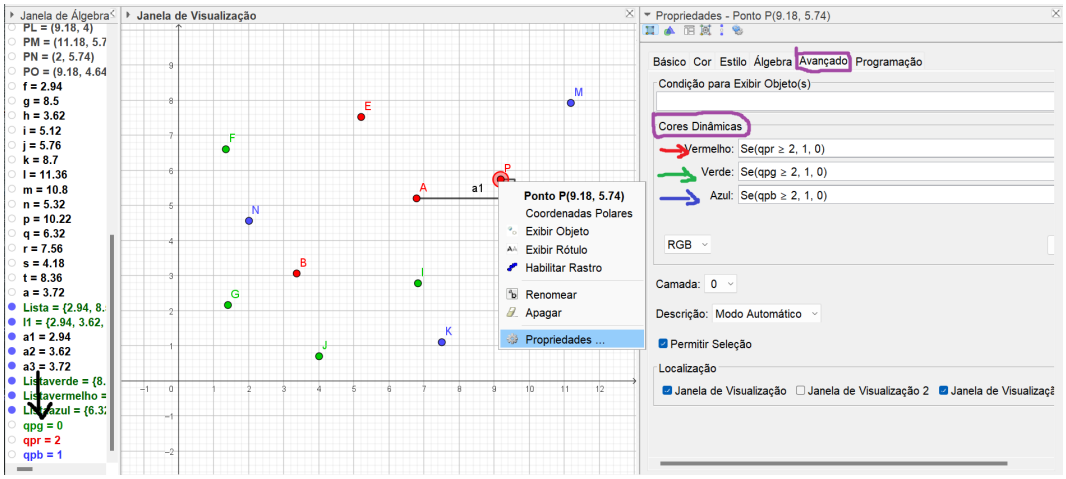


Fonte: Imagem produzida pelo o autor.

Após a criação das listas de cada conjunto de cores os alunos criarão no Geogebra um comando que determina a quantidade de pontos de cada cor que as distâncias mínimas (a1, a2, a3) representam. Para representar a quantidade de pontos verdes utilizaremos qpg, os alunos deverão digitar o comando $qpr = ContarSe(x \leq a3, Listaverde)$, para a quantidade de pontos vermelhos, utilizaremos o comando $qpr = ContarSe(x \leq a3, Listavermelho)$, e para a a quantidade de pontos azuis utilizaremos qpb, os alunos deverão digitar na barra de entrada $qpb = ContarSe(x \leq a3, Listaazul)$. Esse comando faz com que o Geogebra conte quantas distâncias de cada lista, são menores ou iguais a $a3$, desta forma o Geogebra contará apenas as três distâncias mínimas.

Para determinar a cor do ponto P de acordo com a sua classificação, os alunos devem clicar no ponto P com o lado direito do mouse e selecionar a opção propriedades,depois devem selecionar a janela "Avançado", nessa janela teremos a opção de "cores dinâmicas"onde os alunos deverão digitar os seguintes comando: Em Vermelho devem digitar $Se(qpr \geq 2, 1, 0)$; Em Verde devem digitar $Se(qpg \geq 2, 1, 0)$; e em Azul devem digitar o comando $Se(qpb \geq 2, 1, 0)$. Esse comandos definem que o ponto será da cor vermelha, verde ou azul, se a quantidade pontos mais próximos for igual ou maior que dois. A imagem a seguir apresenta essa programação.

Figura 101 – Determinando a cor do ponto P de acordo com a sua classificação



Fonte: Imagem produzida pelo o autor.

A partir dessas configurações os alunos poderão mover o ponto P e classificá-lo de acordo com o algoritmo KNN, utilizando a distância de Manhattan.

6.3.1.3 Atividade 1: Algoritmo KNN para a classificação de pontos

Como já foi ressaltado nesse trabalho de acordo com as Diretrizes Curriculares Nacionais para o Ensino Médio (DCNEM), no currículo do ensino médio pode ser feito um aprofundamento de conhecimentos estruturantes para aplicação de diferentes conceitos matemáticos na área de inteligência artificial, programação entre outras. Visando esse aprofundamento a atividade tem como objetivo compreender a estrutura e comportamento do algoritmo de predição KNN, além disso conceitos básicos do currículo comum do ensino médio como localização de pontos no plano cartesiano e distância entre pontos, também serão trabalhados nessa atividade.

Como existem algumas ferramentas do Geogebra e alguns conceitos sobre aprendizado de máquina, que não são de conhecimento dos alunos, algumas partes da atividade deverão ser feitas pelo professor, sendo a atividade apresentada aos alunos já com esses conceitos preparados, deixando para que os alunos desenvolvam, durante a atividade, apenas conceitos

pertinentes de acordo com o currículo comum, e conceitos que sejam do conhecimento dos alunos. A seguir segue a atividade sobre o algoritmo KNN.

Conteúdo: A atividade sobre o KNN abordará conceitos como localização de pontos e distância entre pontos Euclidiana e de Manhattan.

Objetivo: Classificar pontos (tipos de flor de íris) de acordo com o seu grupo. Compreender, analisar e interpretar o algoritmo KNN de maneira simples.

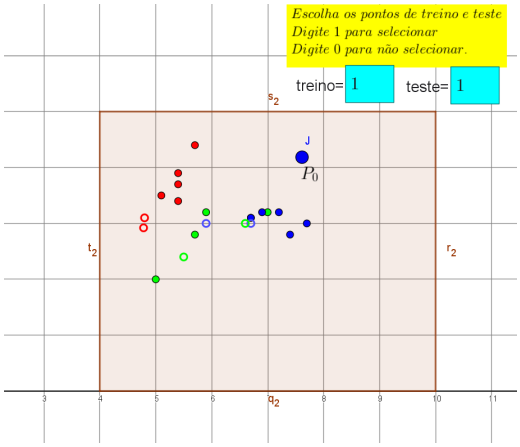
Duração: Sugestão de no mínimo três aulas de 50 minutos, portanto 2 hora e 30 minutos.

Desenvolvimento: Antes de começar a atividade no aplicativo Geogebra, o professor deve explicar alguns conceitos básicos sobre o aplicativo. Além disso, os conteúdos sobre localização de pontos e distância entre pontos devem ser revisados com os alunos, nesse capitulo anteriormente foi apresentado uma breve revisão de alguns desses conteúdos. Após essa preparação uma introdução sobre o que é a classificação de pontos deve ser feita com os alunos, explicando qual o objetivo da predição de acordo com um conjunto de dados, e mostrando a importância da predição de dados e do aprendizado de máquinas para os dias atuais. Para realizar a atividade os alunos poderão faze-la em uma sala de informática, com computadores individuais ou em grupos, ou na falta de uma sala de informática poderá ser feita em sala de aula sendo projetado para todos visualizarem.

Para começar utilizaremos o problema da flor de iris apresentado anteriormente como a problematização dessa atividade, dos 150 dado de flor de íris utilizaremos apenas 21 pontos, lembrando também que trabalharemos com apenas duas medidas, o comprimento e largura da Sépala, transformando os dados em pares ordenados. A partir disso o professor deverá preparar três conjuntos de pontos, que representarão cada tipo de flor de íris, eles serão separados em três cores, azul (Virginica), verde (Setosa) e vermelho(Versicolor). Cada conjunto será formado por sete pontos que serão escolhidos de acordo com as características de cada tipo de flor de íris encontrado no conjunto de dados íris, no qual o professor deverá escolher cinco pontos para representar o conjunto de treino e dois pontos para representar o conjunto de teste. Vale lembrar que no processo de predição no Python esses pontos normalmente são escolhidos de maneira aleatória utilizando a função *train test split*, porém no Geogebra esse procedimento não é possível, porém ressalto que o objetivo da atividade é fazer os alunos compreender como funciona o algoritmo KNN, sendo assim esta escolha não afetará nesse processo. Escolhidos os pontos, para facilitar a realização da atividade o professor pode selecionar uma lista para o treino e outra para o teste e criar um controle

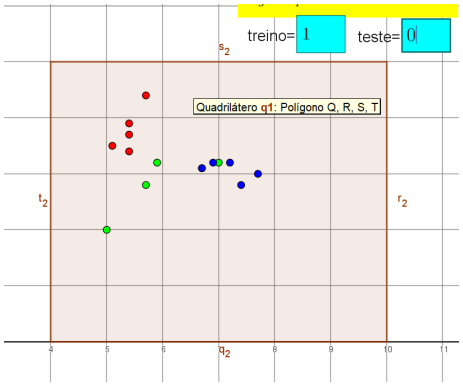
de seleção, sendo 1 para selecionar o conjunto de ponto e 0 para não selecionar, além disso também deve ser criado um ponto móvel P_0, esse ponto será o ponto utilizado para fazer a classificação pelo algoritmo KNN (k vizinhos mais próximos). Feito essa preparação obteremos os seguinte conjuntos de pontos.

Figura 102 – Conjuntos de pontos e o ponto P_0



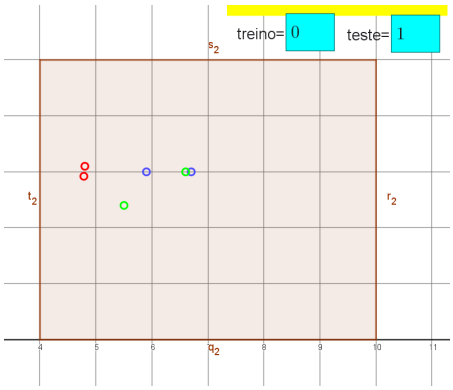
Fonte: Imagem produzida pelo o autor.

Figura 103 – Conjuntos de pontos treino



Fonte: Imagem produzida pelo o autor.

Figura 104 – Conjuntos de pontos teste

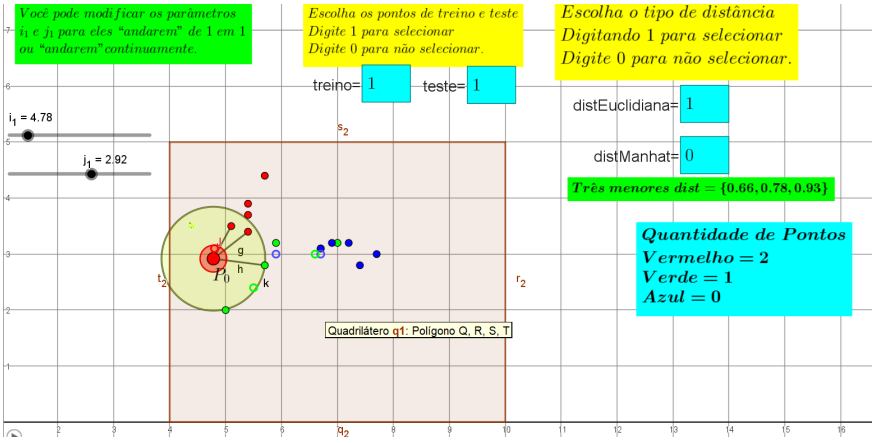


Fonte: Imagem produzida pelo o autor.

Nessa etapa inicial os alunos deverão identificar cada ponto do conjunto de treino a partir do seu par ordenado, fazendo sua localização, é importante também que os alunos anotem todas as localizações e identifique os conjuntos que cada ponto pertence. Também é interessante que o aluno movimente e determine algumas localizações do ponto móvel para visualizar algumas possibilidades. Os pontos do conjunto de teste devem estar escondidos para os alunos nesse momento da atividade.

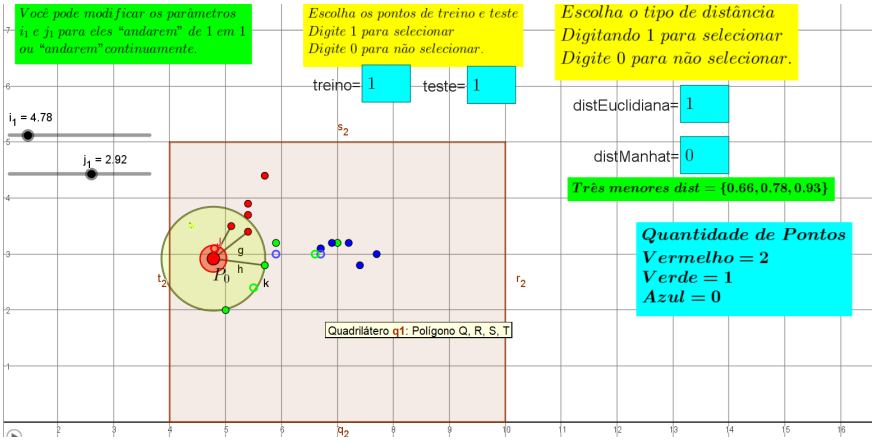
Após a localização dos pontos o conjunto de pontos teste será apresentado e iniciaremos a análise dos k-vizinhos mais próximos de acordo com as distâncias mínimas entre os pontos treino e os pontos teste, utilizando o ponto P_0 como classificador. Lembrando que nessa atividade podemos utilizar a distância euclidiana e a distância de Manhattan, criando um comparativo entre as duas métricas. Para a distância euclidiana devemos determinar os segmentos de retas entre os pontos e o ponto P_0, neste caso a distância entre dois pontos será determinada pela hipotenusa, utilizando a fórmula $d(A,B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$. Já na distância de Manhattan a distância será determinada pelos catetos, ou seja, o caminho será feito pelas "esquinas", neste caso utilizamos a fórmula $d(A,B) = |(x_A - x_B)| + |(y_A - y_B)|$. No caso dessa atividade o professor deve escolher qual a quantidade k de vizinhos mais próximos utilizar, seria interessante utilizar valores k diferentes para que os alunos analisem qual a melhor opção para predição desses pontos.

Figura 105 – Classificação KNN, utilizando a distância Euclidiana



Fonte: Imagem produzida pelo o autor.

Figura 106 – Classificação KNN, utilizando a distância Manhattan



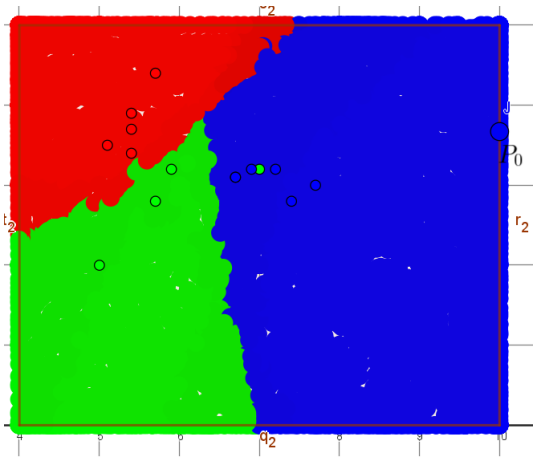
Fonte: Imagem produzida pelo o autor.

Nesse momento da atividade, os alunos devem utilizar o ponto P_0, posicionando o ponto classificador sobre cada ponto do conjunto de teste, determinando para cada ponto teste, cada uma das 15 distâncias entre os pontos do conjunto treino e o ponto P_0. Nesse momento sugiro que o professor permita que os alunos utilizem a calculadora para auxiliar no calculo das distâncias, isso ajudará a agilizar esse processo, é importante novamente pedir que os alunos anotem todas as distâncias analisadas no caderno. Devem ser calculados as distâncias Euclidianas e de Manhattan. Feito o cálculo da distância o professor deve explicar o conceito do algoritmo KNN, deixando claro a questão da classificação de acordo com os k vizinhos mais próximos. Após explicar os alunos deverão analisar as três distâncias (no caso de k=3) mais próximas de cada ponto teste, deverão a partir disso classificar o ponto P_0 de acordo com o algoritmo KNN, classificando dessa forma os pontos que representam uma flor de íris como Virgínica, Versicolor ou Setosa.

No Geogebra o professor pode programar para que apareçam apenas os segmentos mais próximos de acordo com o tipo de distância escolhido, para que o aluno possa ver de forma mais concreta as classificações para o ponto P_0, como foi mostrado nas figuras 93 e 94, além disso é interessante que o professor programe para deixar o rastro do ponto P_0 com a cor de acordo com a sua classificação. Essa etapa deve ser repetida com os alunos algumas vezes, mudando a localização do ponto P_0 e o classificando de acordo com o algoritmo.

Para finalizar a atividade o professor poderá utilizar o rastro do ponto P_0, deixando marcado e mostrando aos alunos a classificação de vários outros pontos. Uma sugestão é que o professor preencha todo o espaço de classificação no plano cartesiano, o professor pode delimitar esse espaço, no caso do exemplo apresentado o plano cartesiano foi limitado nos intervalos $4 \leq x \leq 10$ e $0 \leq y \leq 5$ no Geogebra, assim com o rastro do ponto P_0 será criado uma espécie de mapa de classificação dos conjuntos de pontos. Assim como ilustra a imagem a seguir.

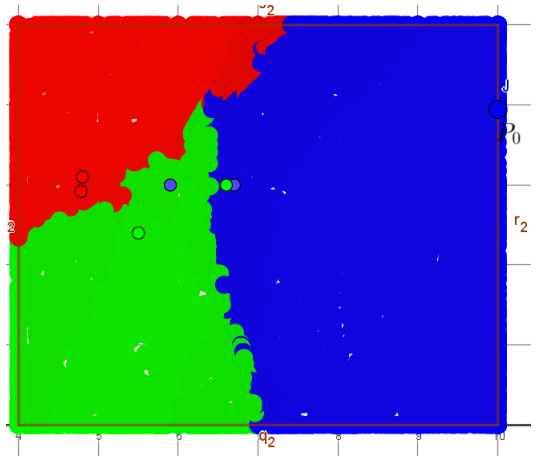
Figura 107 – Mapa de classificação utilizando a distância Euclidiana



Fonte: Imagem produzida pelo o autor.

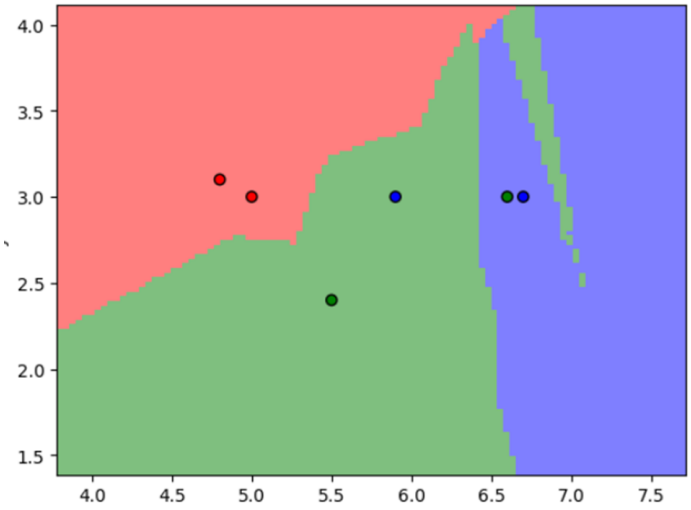
Após esses procedimentos os pontos do conjunto de teste podem ser apresentados sozinhos, nesse momento o aluno terá uma ideia de como foi a eficácia desse modelo de predição, vendo a assertividade de acordo com o mapa de classificação criado e a localização dos pontos teste, sendo possível observar se o modelo criado pelo mapa previu os tipos de flores de íris que estavam "‘escondidos"no conjunto de teste. Além disso o professor pode mostrar no Python, no caso dessa atividade foi utilizado o Google Colaboratory (GOOGLE), que é uma plataforma online e gratuita, é interessante mostrar os resultados no Python e fazer uma comparação com os resultado obtidos na atividade do Geogebra.

Figura 108 – Localização dos pontos teste no mapa de classificação pelo Geogebra



Fonte: Imagem produzida pelo o autor.

Figura 109 – Localização dos pontos teste no mapa de classificação pelo Google Colab



Fonte: Imagem produzida pelo o autor.

De acordo com a localização desses pontos teste os alunos poderão observar que o modelo acertou quatro das seis flores de íris, representadas pelos pontos de teste, é interessante pedir que os alunos calculem a porcentagem de acerto desse conjunto de pontos teste. No caso do exemplo que apresentamos teremos aproximadamente 66,7% de acurácia.

Após a realização da atividade o professor poderá falar sobre a importância do processo de predição e utilização de inteligência artificial, para o cenário atual e futuro, além disso é importante também ouvir os alunos e ter um *feedback* da atividade. Com essa atividade é esperado que os alunos tenham uma compreensão melhor sobre como funciona o algoritmo de predição KNN, além de também trabalhar a distância entre pontos e localização de pontos que está previsto no currículo escolar dos alunos do ensino médio.

6.3.2 Algoritmo Árvore de decisão

Para MURPHY as árvores de decisão é um método de aprendizado supervisionado que é intuitivo e facilmente interpretável. Elas são usadas tanto para problemas de classificação quanto de regressão. O livro explora como essas árvores fazem previsões dividindo os dados em subconjuntos com base em características relevantes. Assim temos que o algoritmo de árvore de decisão é uma técnica de aprendizado de máquina supervisionado usada para modelar e prever resultados a partir de dados. Ele é popular devido à sua capacidade de lidar com conjuntos de dados complexos e variáveis, enquanto é relativamente fácil de entender e interpretar.

O algoritmo inicia com todo o conjunto de dados representado como uma única árvore e, em seguida, divide iterativamente os dados em subconjuntos menores, com base nas características dos dados, para fazer previsões precisas. A árvore de decisão seleciona a

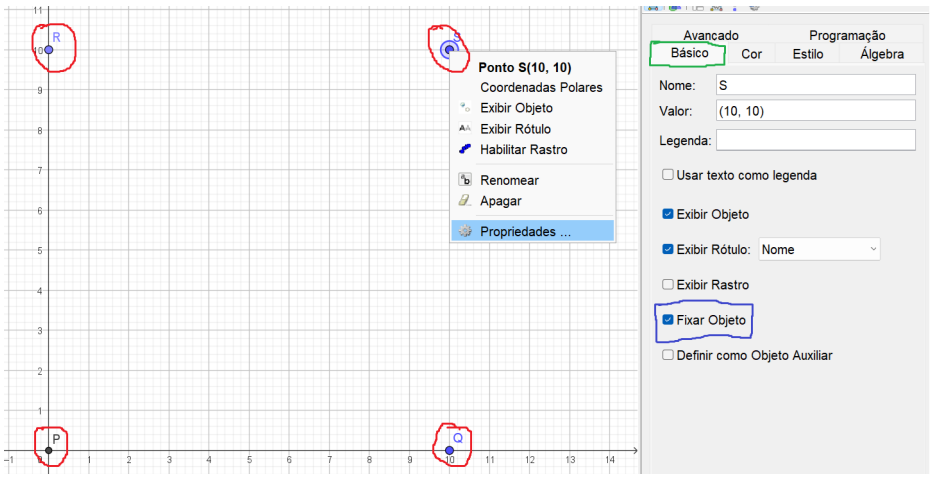
melhor característica (ou atributo) para dividir os dados em cada etapa. Isso é feito com base em critérios como ganho de informação, índice de Gini ou entropia. À medida que a árvore é construída, cada nó representa uma decisão ou uma divisão nos dados. Os nós são divididos em filhos com base nas condições das características, continuando o processo até que os nós finais (folhas) sejam alcançados. O processo de divisão continua até que uma determinada condição de parada seja atendida, como a profundidade máxima da árvore, um número mínimo de amostras em um nó, ou quando não há mais ganho de informação significativo com a divisão.

6.3.2.1 Atividade complementar utilizando árvore de decisão

Essa atividade complementar será utilizada para o compreender alguns conceitos básicos sobre o algoritmo de aprendizado de máquina Árvore de decisões, além disso essa atividade será importante para os alunos manipularem e aprederem alguns comandos do Geogebra.

Inicialmente começaremos limitando o nosso espaço de localização, no Geogebra os alunos devem selecionar o item "Ponto", e selecionar quatro pontos, esses pontos devem estar localizados no plano cartesiano em (0,0), (10,0), (0,10) e (10,10). Os alunos deverão também fixar esses quatro pontos, para isso ele devem clicar com o lado direito do mouse e selecionar propriedades, nessa janela devem selecionar a opção "Básico", onde deverão marcar a caixa "Fixar objeto", como apresenta a figura a seguir:

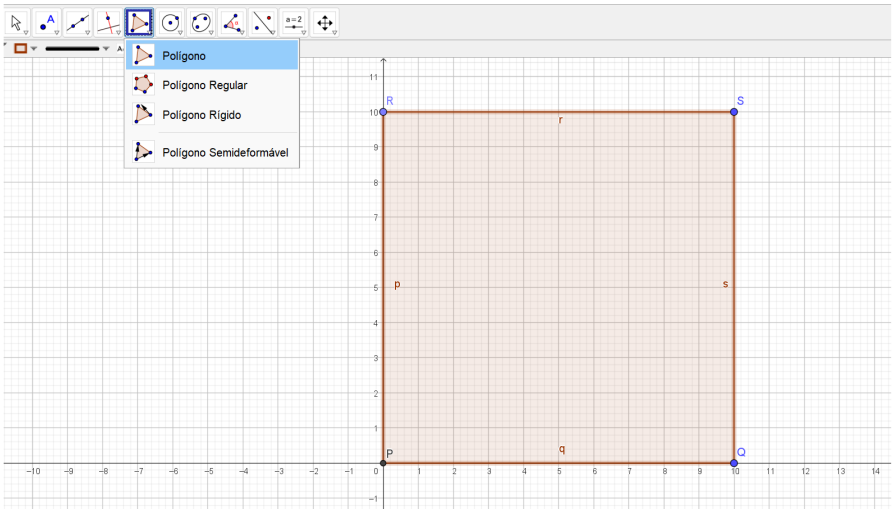
Figura 110 – localização dos pontos P, Q, R e S fixados



Fonte: Imagem produzida pelo o autor.

Após criar esses pontos, os alunos devem construir um polígono que passa por esses quatro pontos, formando assim um quadrado, que será nossa área de localização. Para isso os alunos devem selecionar a opção "Polígono"e clicar nos quatro pontos, criando assim o polígono, como mostra a imagem a seguir:

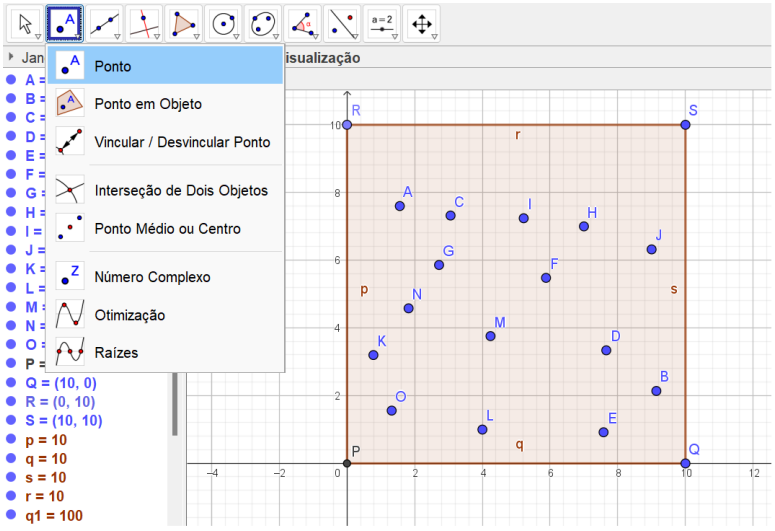
Figura 111 – Criando a área de localização, Polígono PQRS.



Fonte: Imagem produzida pelo o autor.

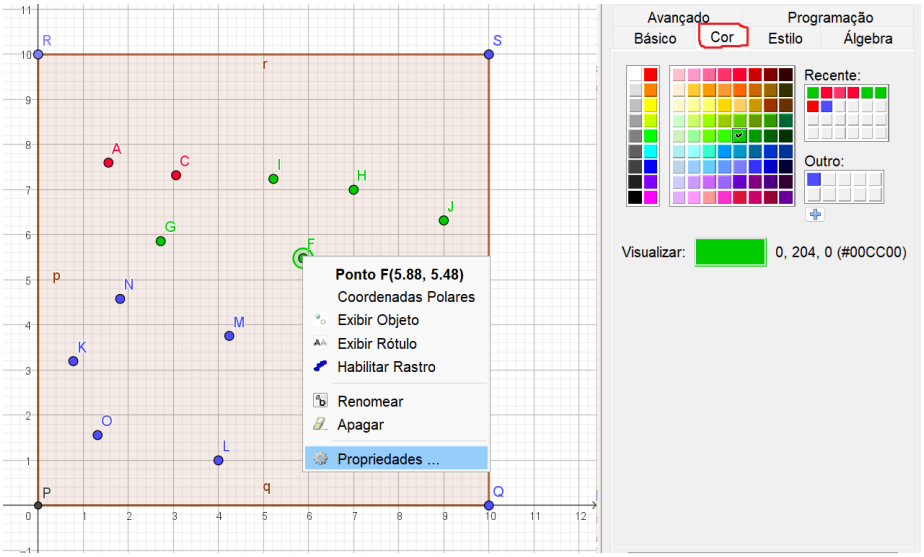
Após a criação da área de localização, os alunos devem selecionar quinze pontos da maneira que preferirem, dentro da área delimitada anteriormente. Para colocar esses pontos no Geogebra os alunos alunos devem selecionar a opção "Ponto"e plotar os pontos no plano cartesiano. Além disso os alunos deverão separar os pontos em três conjuntos de pontos, vermelho, verde e azul, assim alunos deverão mudar a cor de cada ponto, formando assim os conjuntos de pontos, para isso eles devem clicar com o lado direito do mouse e selecionar a opção "propriedades", na janela que abrir devem selecionar a opção "cor"e escolher a cor que preferir.

Figura 112 – Localização dos pontos.



Fonte: Imagem produzida pelo o autor.

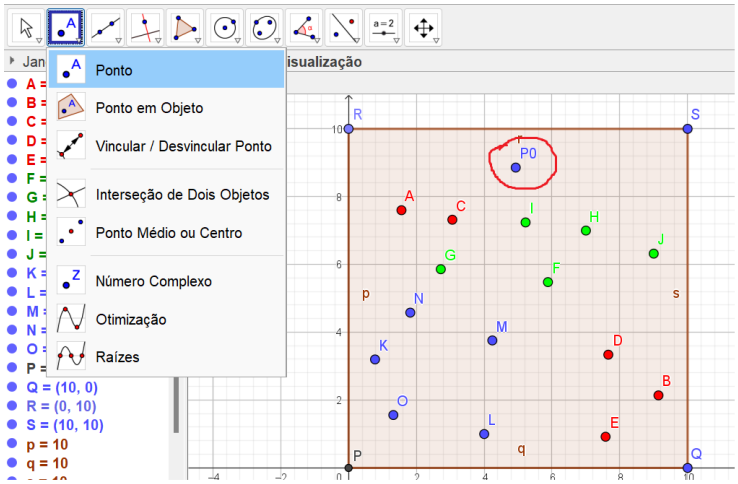
Figura 113 – Definição dos conjuntos de pontos.



Fonte: Imagem produzida pelo o autor.

Escolhido os conjuntos de pontos os alunos deverão criar um ponto P_0, que será o ponto que utilizaremos para classificar. Novamente os alunos devem selecionar na janela a opção "Ponto", como mostra a figura.

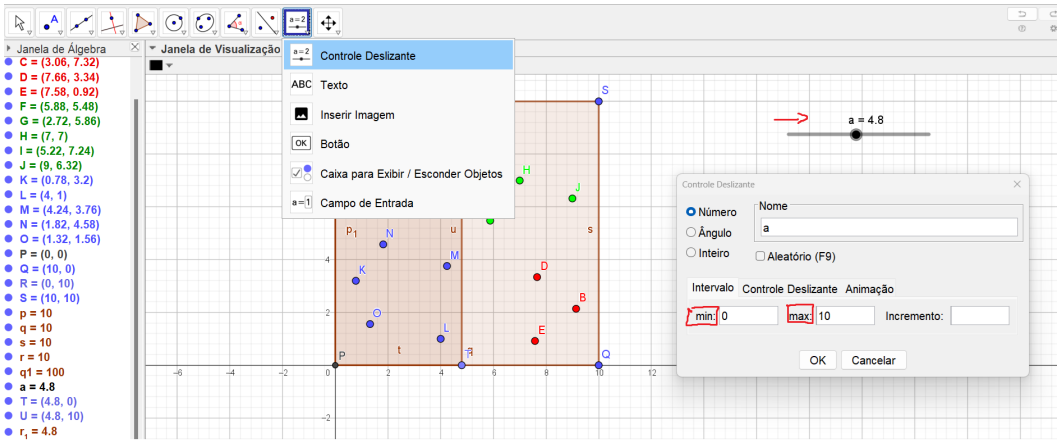
Figura 114 – Ponto P_0.



Fonte: Imagem produzida pelo o autor.

Agora os alunos deverão criar as divisões da região, para simular o algoritmo árvore de decisões, para isso os alunos deverão primeiro criar um controle deslizante, variando de 0 a 10, que é o intervalo da nossa região. Para isso basta que os alunos selecionem a janela "controle deslizante"no Geogebra e cliquem no plano cartesiano ao abrir a janela do controle deslizantes no intervalo deve-se digitar 0 em "min:"e 10 em "max:".

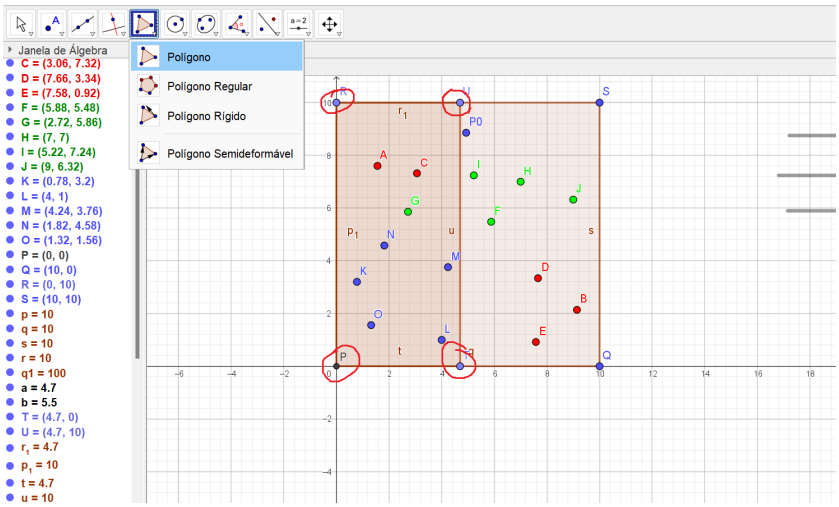
Figura 115 – Controle deslizante.



Fonte: Imagem produzida pelo o autor.

Agora criaremos a primeira divisão em relação ao eixo x, para isso construiremos um polígono na parte esquerda da nossa região de localização, os alunos deverão primeiro criar dois pontos U e T quaisquer, clicando em "Ponto", e selecionando no plano cartesiano, posteriormente na barra de entrada deverão digitar o comando "$U = (a, 10)$ e $T = (a, 0)$". A partir desses pontos os alunos devem construir o polígono PRUT, os alunos devem selecionar a janela "Polígono"e selecionar esses quatro pontos P, R, T e U. Neste caso o controle deslizante **a** determinará a divisão no eixo x.

Figura 116 – Divisão da região em relação ao eixo X

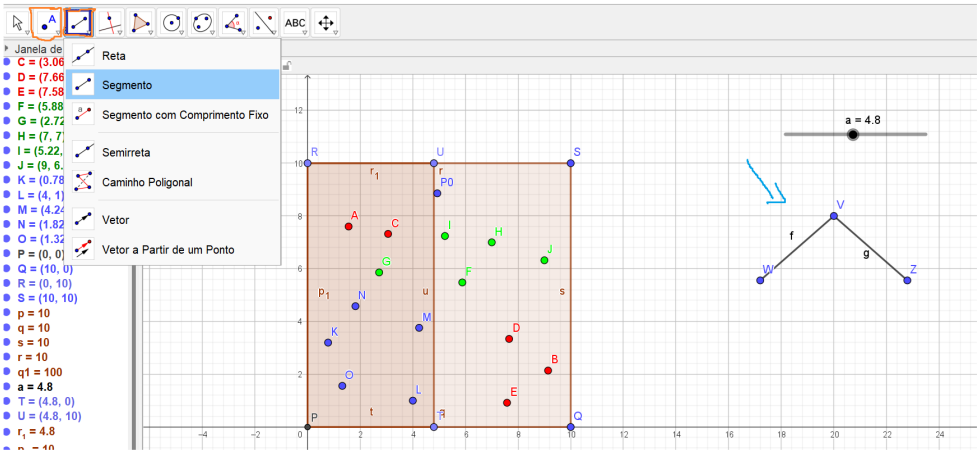


Fonte: Imagem produzida pelo o autor.

Nessa parte da atividade, é importante que a cada divisão feita o aluno faça uma análise, no caderno, das desigualdades criando as árvores de decisão e indicando a quantidade de pontos em cada região. Esse processo pode também ser feito no Geogebra, para organizar nossa árvore de decisão os alunos podem criar dois segmentos de retas para indicar as

ramificações das árvores de decisões, os alunos deverão criar três pontos e construir o segmento entre esses pontos, como já foi mostrado anteriormente.

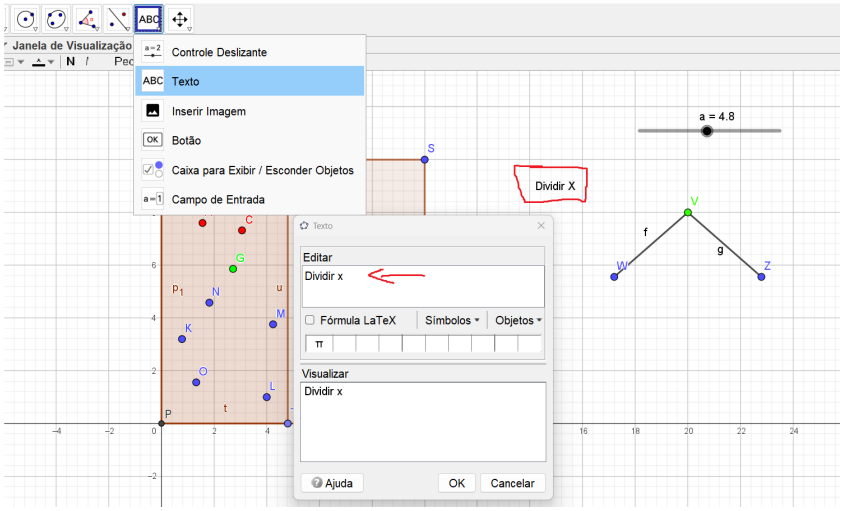
Figura 117 – Construção dos segmentos VW e VZ, para ramificação da árvore de decisão



Fonte: Imagem produzida pelo o autor.

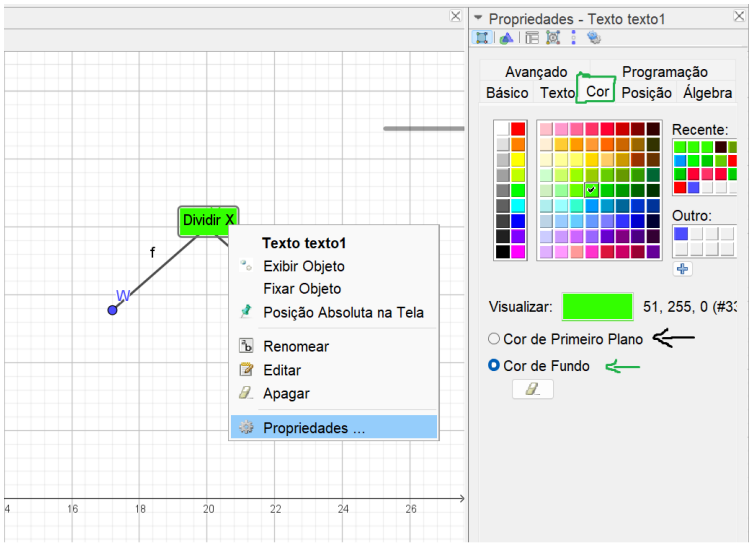
Após a construção dos segmentos, os alunos deverão criar um texto indicando a divisão em x, para isso os alunos podem criar um texto clicando na mesma janela do controle deslizante e selecionando a opção "Texto"e clicar no plano cartesiano, na janela que abrirá o aluno pode escrever *DividirX*, desta forma será criado o texto, o aluno deve movimentar o texto colocando sobre o ponto V. Além disso o aluno pode mudar a cor do texto, clicando com o botão direito do mouse sobre o texto, o aluno deve selecionar a opção "Propriedades", na janela que se abre selecionar a opção "Cor"nessa janela os alunos poderão mudar a cor da letra em "Cor de primeiro plano"e podem mudar a cor do fundo em "Cor de fundo", os alunos devem personalizar esse texto como preferirem.

Figura 118 – Construção do texto (Dividir X)



Fonte: Imagem produzida pelo o autor.

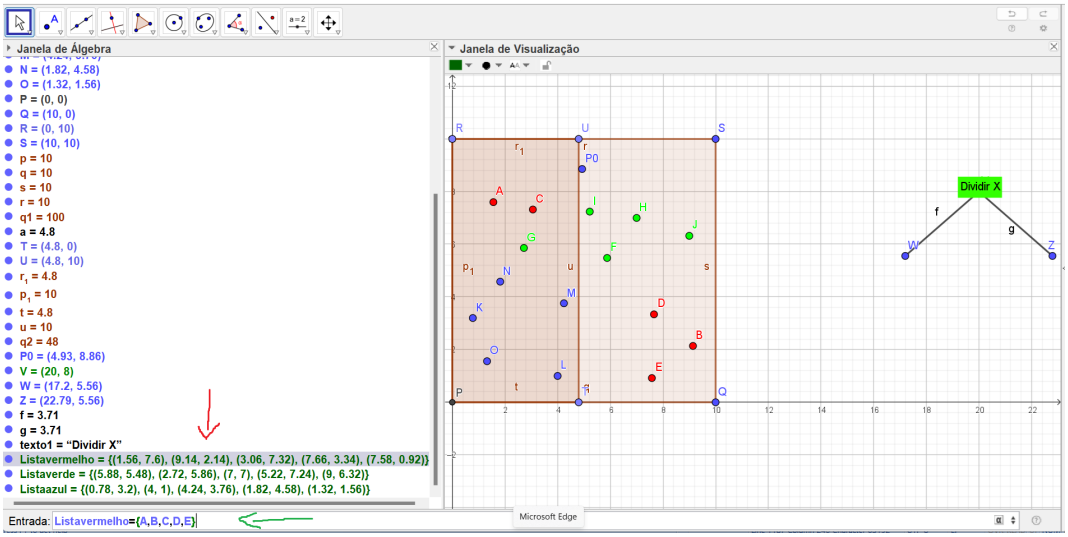
Figura 119 – Personalização do texto



Fonte: Imagem produzida pelo o autor.

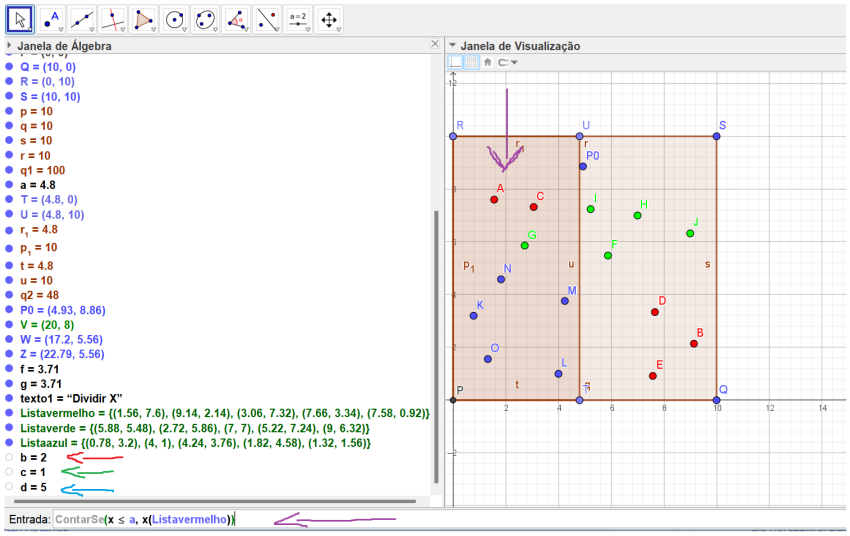
Após essa construção os alunos devem criar três listas de pontos a Listavermelho contendo os pontos vermelhos, a Listaverde contendo os pontos verdes e a Listaazul contendo os pontos azuis. Então no Geogebra os alunos devem digitar na barra de entrada os comandos "$Listavermelho = \{A, B, C, D, E\}$", "$Listaverde = \{F, G, H, I, J\}$"e "$Listaazul = \{K, L, M, N, O\}$". Além disso os alunos devem digitar na barra de entrada os comandos "$ContarSe(x \leq a, x(Listavermelho))$", "$ContarSe(x \leq a, x(Listaverde))$"e "$ContarSe(x \leq a, x(Listaazul))$", desta forma o Geogebra determinará a quantidade de pontos vermelhos, de pontos verdes e de pontos azuis que a região do lado esquerdo possui.

Figura 120 – Construção das listas dos conjuntos de pontos



Fonte: Imagem produzida pelo o autor.

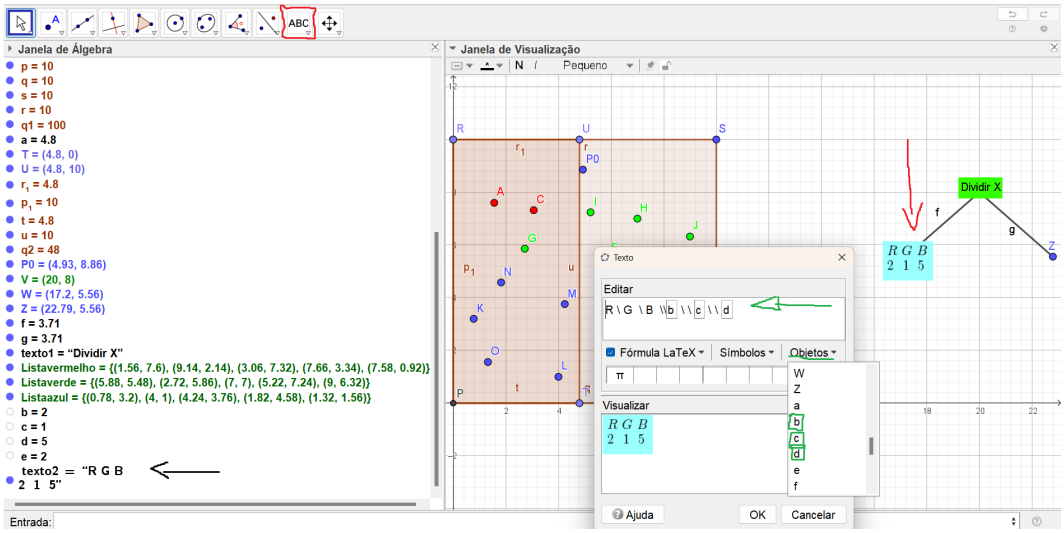
Figura 121 – Determinando a quantidade de pontos de cada conjunto na região RUTP



Fonte: Imagem produzida pelo o autor.

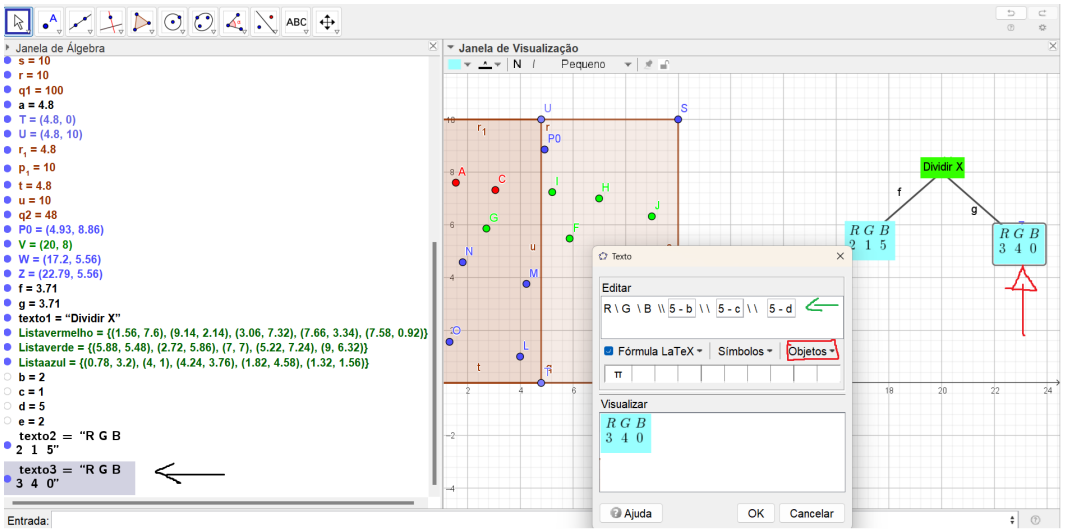
Agora iremos construir as árvores de decisão, os alunos deverão selecionar a janela texto e na janela de comando devem digitar "RGB"e selecionar os objetos b, c e d, desta forma o Geogebra determinará a árvore de decisão da região formada ao lado esquerdo e devem colocar o texto sobre o ponto W, para determinar a árvore de decisão da outra região, os alunos devem novamente selecionar a janela texto, e na janela de comando devem digitar "RGB"e selecionar os objetos $5-b$, $5-c$ e $5-d$ e colocar o texto sobre o ponto Z, como mostra a figura a seguir.

Figura 122 – Criando a árvore de decisão da Região RUTP



Fonte: Imagem produzida pelo o autor.

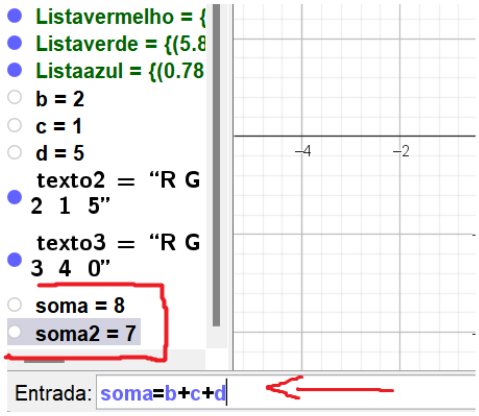
Figura 123 – Criando a árvore de decisão da Região UTSQ



Fonte: Imagem produzida pelo o autor.

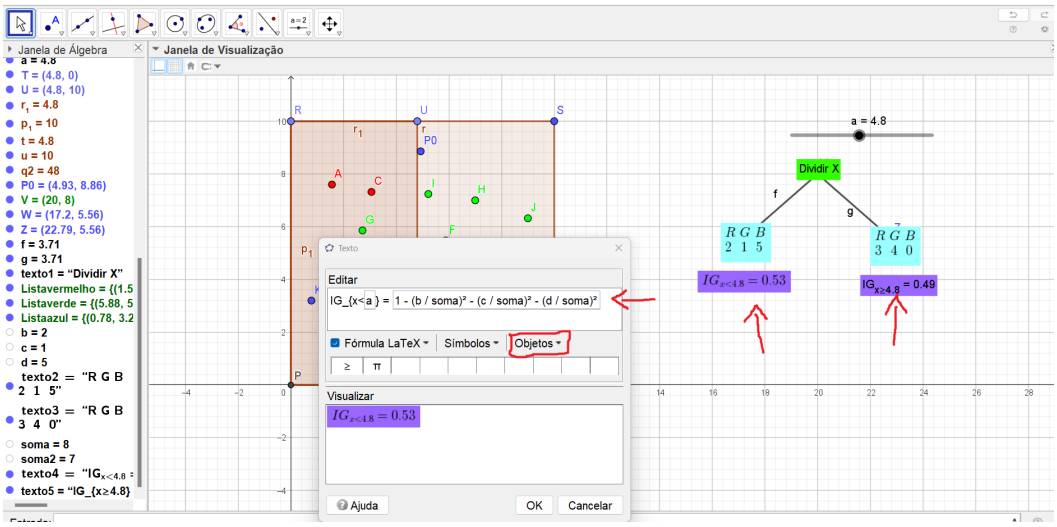
Para determinar a melhor divisão de região no algoritmo árvore de decisão podemos utilizar o índice Gini, quanto menor o índice Gini melhor será a divisão da região. Para melhor entendimento do índice Gini os alunos devem realizar alguns cálculos do índice no caderno. No Geogebra os alunos deverão inicialmente determinar a quantidade de pontos de cada região, para isso os alunos devem digitar na barra de entrada o comando "$soma = b + c + d$", e "$soma2 = 15 - soma$". Agora determinaremos o índice Gini de cada região, para região com o intervalo $\{0 \leq x \leq a\}$ e $\{a \leq x \leq 10\}$, para a região à esquerda os alunos devem selecionar a janela texto e digitar o seguinte comando "$IG\{x < objetoa\} = objeto1 - (b/soma)^2 - (c/soma)^2 - (d/soma)^2$", para a região à direita os alunos devem selecionar a janela texto e digitar o seguinte comando "$IG\{x \geq objetoa\} = objeto1 - ((5-b)/soma2)^2 - ((5-c)/soma2)^2 - ((5-d)/soma2)^2$".

Figura 124 – Soma dos pontos de cada região



Fonte: Imagem produzida pelo o autor.

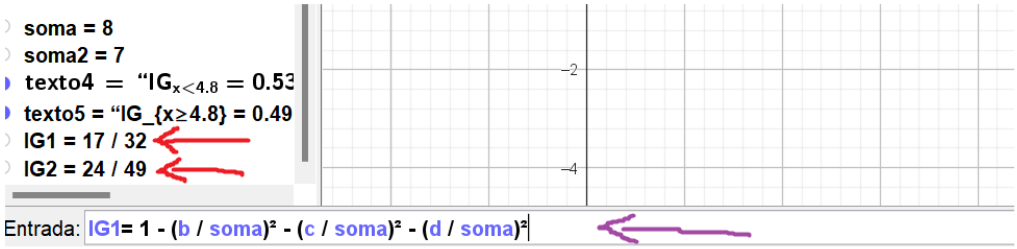
Figura 125 – Construção texto índice Gini



Fonte: Imagem produzida pelo o autor.

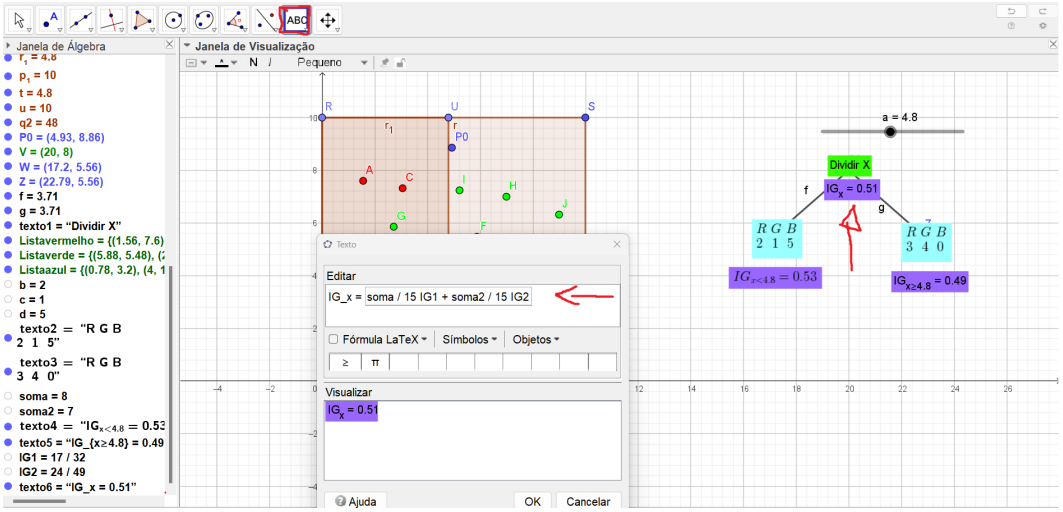
Para finalizar os alunos deverão construir no Geogebra o índice Gini geral, para isso eles deverão determinar o valor dos índices de cada região na janela de álgebra, para isso eles deverão digitar na janela de entrada os comandos $IG1 = 1 - (b/soma)^2 - (c/soma)^2 - (d/soma)^2$ e $IG2 = 1 - ((5-b)/soma2)^2 - ((5-c)/soma2)^2 - ((5-d) \div soma2)^2$. Feito isso os alunos deverão criar um novo texto com o seguinte comando "$IG_x = objetosoma/15IG1 + soma2/15IG2$", desta forma obteremos o índice Gini, como mostra a imagem a seguir.

Figura 126 – Calculando o índice Gini no Geogebra



Fonte: Imagem produzida pelo o autor.

Figura 127 – Construção do índice Gini Geral



Fonte: Imagem produzida pelo o autor.

Feito isso os alunos podem mover o ponto P_0 e classificá-lo de acordo com a região onde estiver, se a região possuir mais pontos vermelhos o classificamos como vermelho, se tiver mais pontos verdes o classificamos como verde e se tiver mais pontos azuis o classificaremos como azul. Se o professor desejar fazer mais divisões de região basta continuar seguindo esses passos em novas divisões de regiões.

6.3.2.2 Atividade 2: Utilizando o Geogebra para classificar pontos de acordo com o algoritmo árvore de decisões

Para a realização dessa atividade o professor deverá dar uma introdução sobre o algoritmo Árvore de decisões, mostrando características importantes como foi apresentado no inicio dessa seção. Assim como foi feito na atividade 1, alguns conceitos e ferramentas do Geogebra devem ser explicados e algumas partes mais complexas da atividade deverão ser feitas pelo professor, sendo apenas apresentado aos alunos, deixando para que os alunos desenvolvam durante a atividade apenas conceitos pertinentes e que sejam de seus conhecimentos. A seguir será apresentada a atividade sobre árvore de decisão.

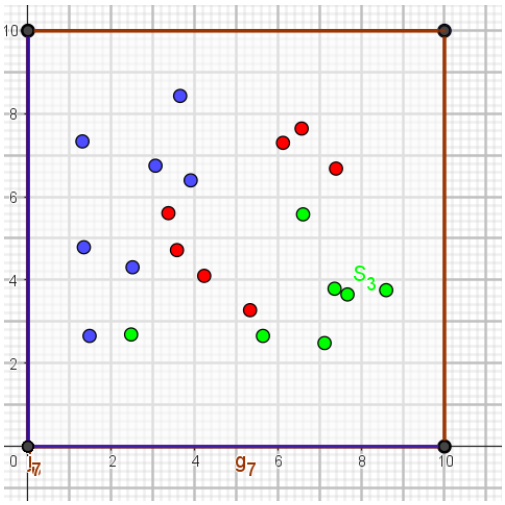
Conteúdo: A atividade sobre árvore de decisão abordará conceitos como localização de pontos e inequação.

Objetivo: Classificar os conjuntos de pontos, localizar pontos no plano cartesiano, trabalhar o conceito de desigualdades, e compreender o algoritmo árvore de decisões.

Duração: Sugestão de no mínimo três aulas de 50 minutos, portanto 2 hora e 30 minutos.

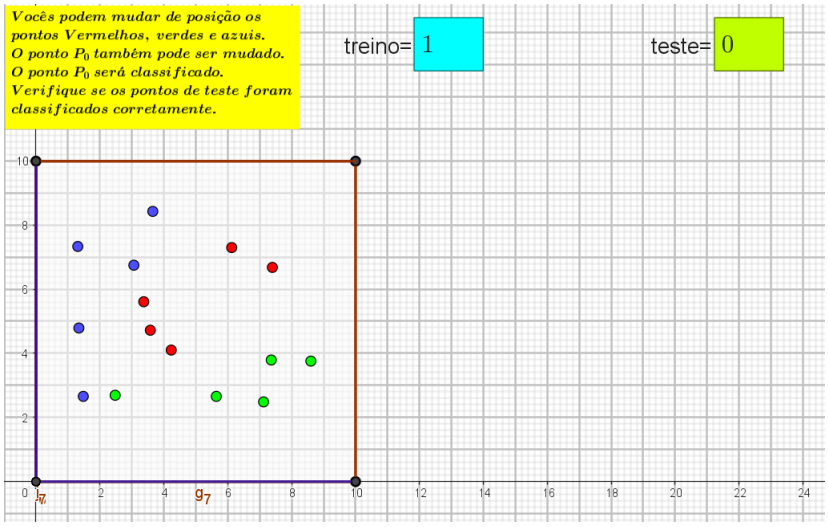
Desenvolvimento: Antes de iniciar a atividade, os conteúdos sobre localização de pontos e desigualdade já deverão terem sido trabalhados com os alunos, como já foi destacado anteriormente nesse capítulo foi apresentado uma breve explicação de alguns desses conteúdos. A atividade iniciará com os pontos e a região plana limitada, já construídos pelo professor, esses pontos serão separados por grupos de cores, sendo sete pontos azuis, sete pontos verdes e sete pontos vermelhos, sendo cinco pontos de cada cor separados para o conjunto de pontos de treino e dois pontos de cada cor para o conjunto de teste, além de um ponto móvel P_0 que será o ponto classificador, montando o modelo de classificação a partir dele. A localização desses pontos deve ser determinados pelo professor da maneira que achar conveniente, assim como as regiões planas que deverão ser limitadas, no exemplo realizado a região foi limitada no eixo das abscissas no intervalo $0 \leq x \leq 10$ e no eixo das ordenadas no intervalo $0 \leq y \leq 10$, essa região plana será utilizada para criar as regiões para a divisão de pontos para o algoritmo árvore de decisões.

Figura 128 – Conjuntos de pontos



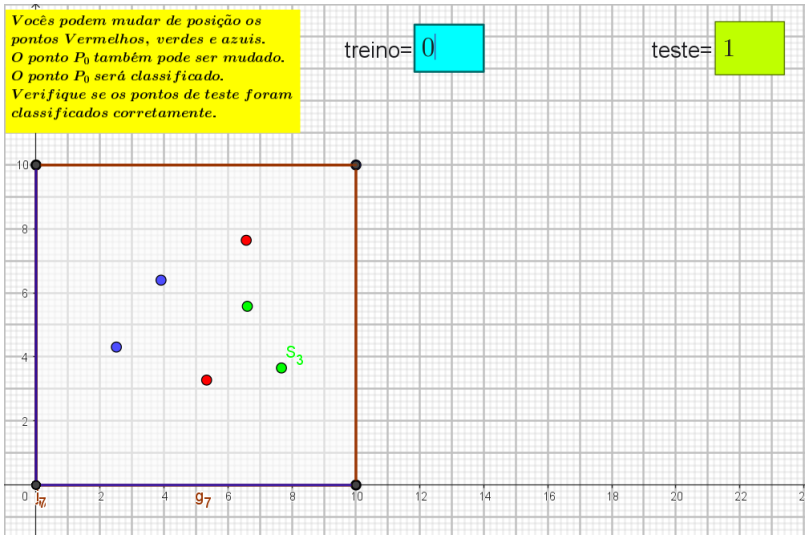
Fonte: Imagem produzida pelo o autor.

Figura 129 – Conjuntos de pontos treino



Fonte: Imagem produzida pelo o autor.

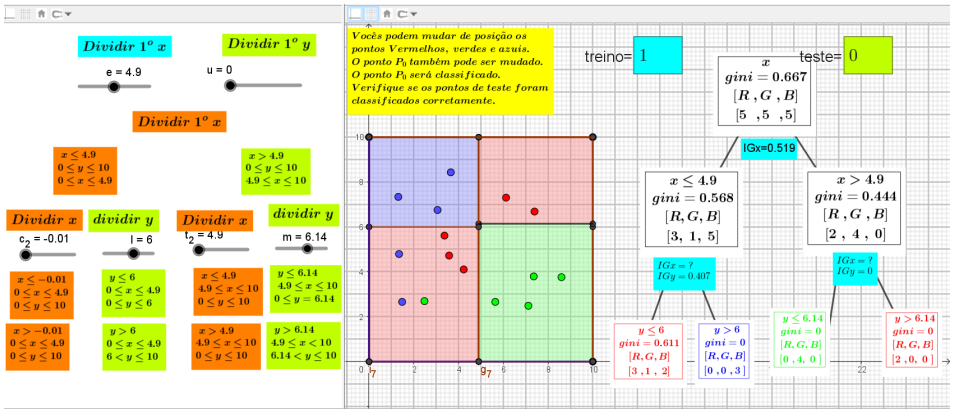
Figura 130 – Conjuntos de pontos teste



Fonte: Imagem produzida pelo o autor.

Após apresentar os pontos, os alunos deverão localizar os pontos de treino no plano cartesiano, é importante que esse processo seja anotado no caderno, e que os alunos identifiquem a qual grupo e cores cada par ordenado pertence. Após a localização o professor deverá mostrar as divisões dos grupos de pontos em relação ao eixo x e y. Nessa etapa da atividade os alunos deverão manipular o Geogebra encontrando diferentes divisões de pontos analisando as desigualdades que aparecerão e a quantidade de pontos de cada grupo em cada lado da desigualdade, é interessante que o aluno faça no caderno as árvores de decisão, obtidas em cada divisão, comparando as arvores de divisão criadas com as árvores de decisão apresentadas no Geogebra.

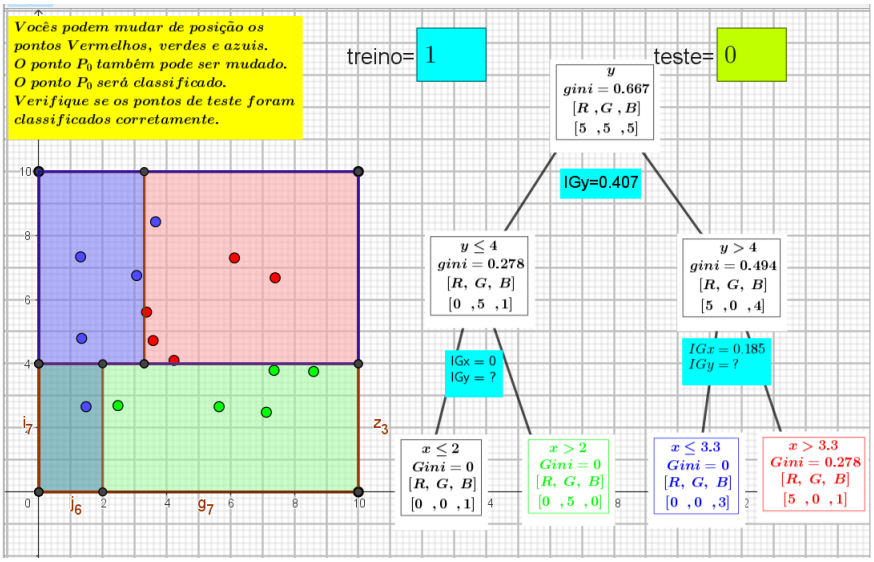
Figura 131 – Análise da divisão de regiões e as árvores de decisão.



Fonte: Imagem produzida pelo autor.

Para obtermos um melhor resultado no processo de divisão de regiões podemos utilizar o índice Gini, que é uma medida que determina a pureza da divisão das árvores de decisões, ajudando dessa forma a encontrar a melhor divisão possível. O cálculo do índice Gini deve ser feito pelos os próprios alunos, para auxiliar nesse cálculo o professor deve explicar como é realizado fazendo alguns cálculos junto aos alunos, na seção 6.6 temos uma breve explicação desse cálculo. Nesta etapa os alunos deverão explorar o Geogebra fazendo divisões de regiões e analisando o índice Gini. Após a análise das árvores de decisão e da divisão das regiões, o professor deverá explicar que a melhor divisão é a que utiliza menos ramificações de forma que os grupos de pontos tenham a melhor separação possível, além disso o professor deve explicar que quanto menor o índice Gini melhor será a divisão, dito isso os alunos deverão fazer a separação dos grupos de pontos treino da melhor maneira possível analisando essas características. Ainda para facilitar a análise o professor deve programar para que as regiões fiquem coloridas de acordo com a classificação do algoritmo, o professor deve mostrar que a região classificará o ponto de acordo com o número de pontos treino, se tiver mais pontos vermelhos na região, então a região ficará vermelha, ou seja, o modelo identificará qualquer ponto daquela região como um ponto do conjunto vermelho, o mesmo vale para os pontos dos outros conjuntos de cores.

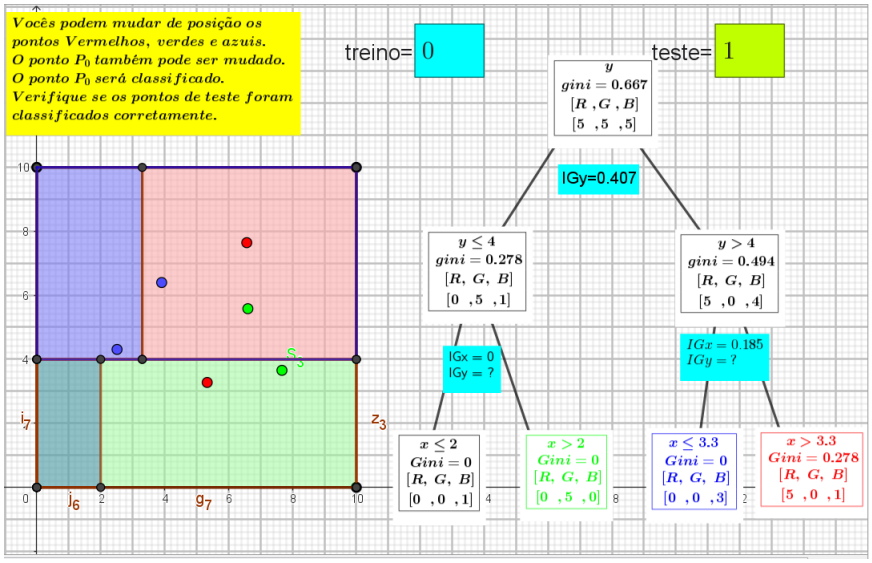
Figura 132 – Divisão de pontos utilizando o índice Gini



Fonte: Imagem produzida pelo autor.

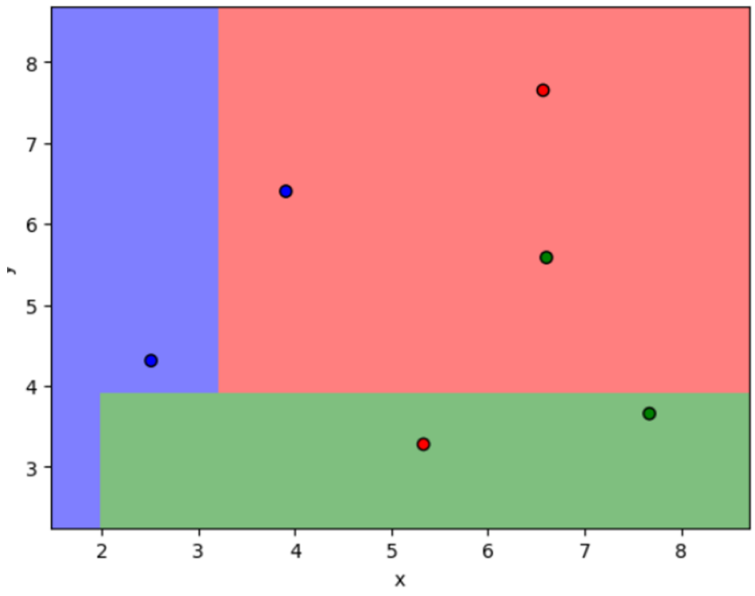
Nesse processo de análise da divisão das regiões é importante o professor pedir para que os alunos identifiquem os intervalos de cada região, anotando-os no caderno, e indicando cada região de classificação de pontos. Após esse processo de divisão de pontos o professor deverá mostrar a localização dos pontos teste, pedindo para que os alunos identifique a localização desses pontos, classificando-os a partir das regiões de classificação determinadas, analisando se o modelo de classificação criado está condizentes com a classificação dos pontos teste. Desta forma os alunos poderão analisar a eficácia do modelo criado, no caso do exemplo apresentado podemos ver que obtemos 50% de acurácia, acertando três pontos de um total de seis. Assim como na atividade anterior, também é interessante que o professor mostre o processo de classificação desses pontos no Python, utilizando o algoritmo árvore de decisão, desta forma os alunos poderão ter uma experiência de como esse processo de análise e predição de dados ocorre, e pode ser feito uma comparação entre os resultados obtido na atividade com o Geogebra e o processamento realizado no Google Colab.

Figura 133 – Apresentação dos pontos teste no Geogebra



Fonte: Imagem produzida pelo autor.

Figura 134 – Apresentação dos pontos teste no Google Colab



Fonte: Imagem produzida pelo autor.

Com o desenvolvimento dessa atividade além de os alunos desenvolverem conteúdos previstos pela BNCC, como localização de pontos e análise de desigualdades, também dará proximidade com procedimentos de aprendizado de máquinas, mais especificamente o desenvolvimento do algoritmo árvore de decisão. A atividade não apresenta conceitos complexos que o algoritmo possui, porém o apresenta de maneira simplificada, possibilitando aos alunos desenvolvam o entendimento de como tal algoritmo se comporta.

6.4 RECURSO DIDÁTICO E TECNOLÓGICO COM PYTHON

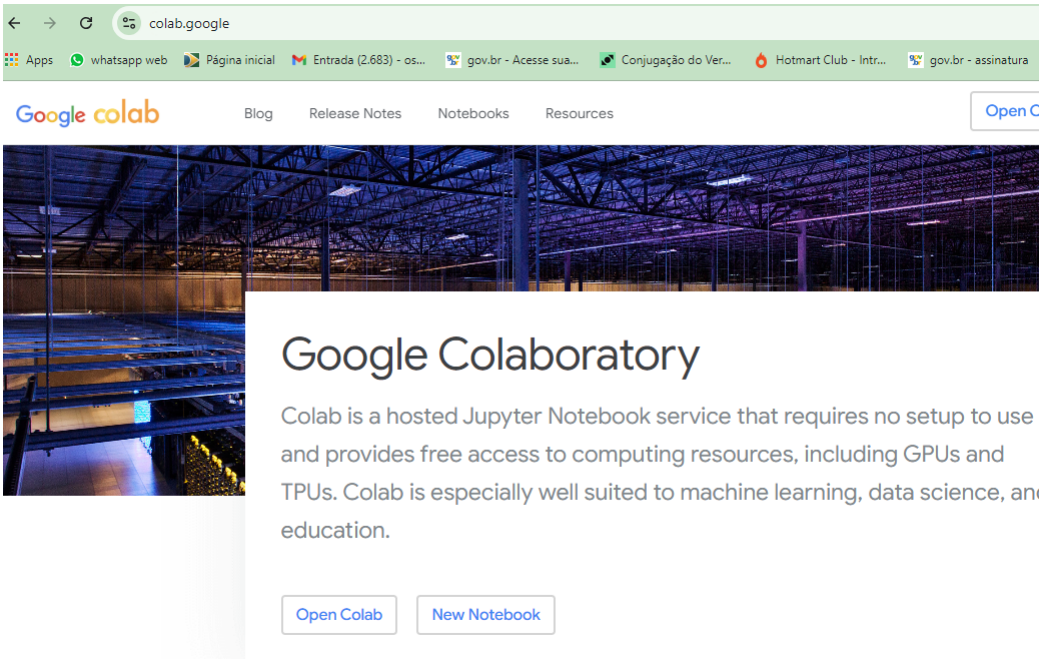
O algoritmo de KNN e Árvores de Decisão para o exemplo feito no Geogebra na seção anterior, foram feitas no Colab Python, veja: Link para o Google Colab

Segue alguns detalhes da programação python do algoritmo Knn e DT.

6.4.1 Iniciando no Python: instalação ou uso online

Ao invés de instalar o python Anaconda Jupyter (https://jupyter.org/) instalando no próprio computador que pode ser encontrado no site oficial https://www.python.org/, sugerimos utilizar o Google Colab (https://colab.research.google.com/), uma forma simples de programar online com o servidor da Google. A ferramenta pode ser utilizada em dispositivo móvel, todavia existem algumas limitações. Além disso, o tempo de processamento é limitado pelos servidores da Google e exige conexão permanente com a internet.

Figura 135 – Google Colab



Fonte: Google Colaboratory

6.4.2 Importando e instalando bibliotecas

Existem milhares de bibliotecas, das quais usaremos principalmente a Scikit-Learn (!pip install scikit-learn) e as bibliotecas NumPy (http://numpy.org), Matplotlib (http://matplotlib.org). A importação das bibliotecas deve ser explicitamente declarada com import, pois os recursos não são carregados por padrão. Por exemplo, o comando import numpy as np importa a biblioteca numpy com o apelido de np. Isso é especialmente útil para biblioteca de nome

extenso, já que para chamar funções da biblioteca numpy, apenas escreva np. A mudança de nome é promovida pelo as, representando o advérbio como.

```
!pip install scikit-learn

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn import tree
from sklearn.model_selection import cross_val_score
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.inspection import DecisionBoundaryDisplay
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_validate
#scoring
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, precision_score,
      recall_score,average_precision_score, roc_auc_score,
      precision_recall_curve, roc_curve, auc,f1_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix,
   ConfusionMatrixDisplay
```

6.4.3 Lendo dados

```
import pandas as pd

# Carregar o arquivo Excel
#df = pd.read_excel('/content/dadosGeogebra.xlsx')

# Salvar como CSV
```

```
#df.to_csv('dadoscores21pontos.csv', index=False)

# Carregar o arquivo como txt
df = pd.read_csv('/content/dadosGeogKNNDT.txt', sep='\t')
```

Vamos importar dados do Geogebra em formato txt ou formado excel ou csv.

Tabela 71 – Conjunto de dados para treino: 17 pontos, 7 vermenlhos, 7 verdes e 7.

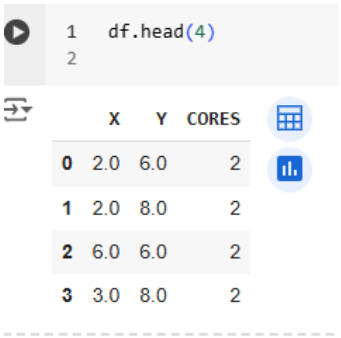
	X	Y	CORES
0	2.0	6.0	azul
1	2.0	8.0	azul
2	6.0	6.0	azul
3	3.0	8.0	azul
4	1.0	7.0	azul
5	7.0	7.0	vermelho
6	8.0	8.0	vermelho
7	2.0	4.0	vermelho
8	3.0	7.0	vermelho
9	6.0	8.0	vermelho
10	8.0	4.0	verde
11	7.0	6.0	verde
12	5.0	3.0	verde
13	6.0	1.0	verde
14	6.0	4.0	verde
15	4.0	6.0	azul
16	2.0	4.0	azul
17	7.0	3.0	verde
18	6.0	6.0	verde
19	4.0	5.0	vermelho
20	7.0	7.0	vermelho

Fonte: Próprio Autor.

rotulando os dados

```
# rotulando as cores em 0,1,2
type_dict = {'vermelho':0,'verde':1,'azul':2}
df['CORES'] = df['CORES'].map(type_dict)
```

Figura 136 – Google Colab



Fonte: Próprio Autor

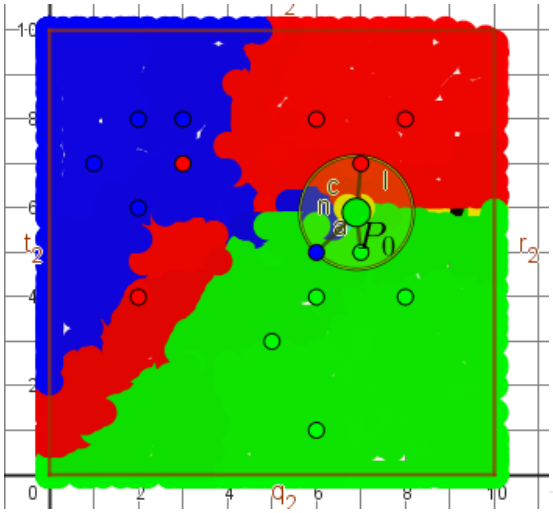
escolhendo os 15 dados de treino e 6 de teste

```
# Separa dado de treino e teste utilizando os dadaset df1 e df2
    acima
X = df[['X','Y']]
y = df[['CORES']]
X_treino = df1[['X','Y']]
y_treino = df1[['CORES']]
X_teste = df2[['X','Y']]
y_teste = df2[['CORES']]
```

6.4.3.1 KNN

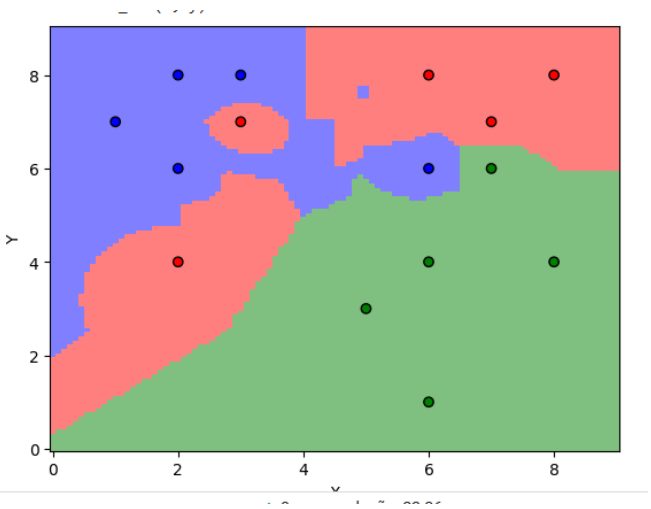
```
classificador = KNeighborsClassifier(n_neighbors = 3)
classificador.fit(X_treino,y_treino.values.ravel())

scores_dt = cross_val_score(classificador,X_treino,y_treino.values.
    ravel(),
              scoring='accuracy', cv=3)
print(scores_dt.mean())
0.67
```

Figura 137 – KNN no Geogebra



Fonte: Próprio Autor

Figura 138 – KNN no Python



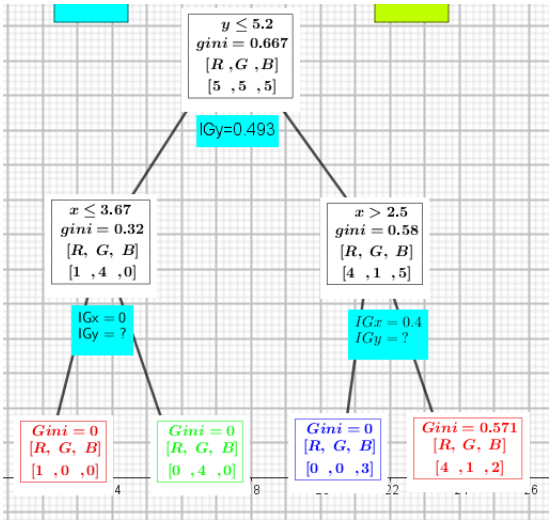
Fonte: Próprio Autor

6.4.3.2 Árvore de Decisão

```
clf = DecisionTreeClassifier(criterion='gini', max_depth=2)
# Train the classifier on the training data
clf.fit(X_treino, y_treino)
accuracy = metrics.accuracy_score(y_treino, y_pred)
print(f"Accuracy: {accuracy}")
0.8
```

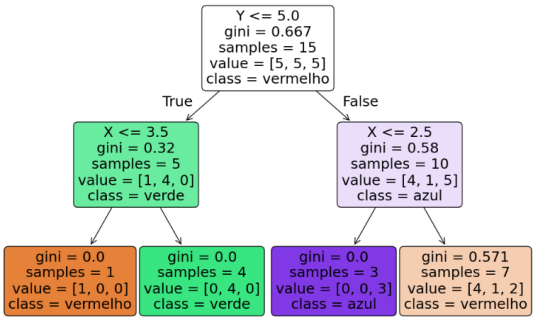
```
# Visualize the decision tree
plt.figure(figsize=(12, 8))
tree.plot_tree(clf, feature_names=feature_names, class_names=
    class_names, filled=True, rounded=True)
plt.show()
```

Figura 139 – Árvore de Decisão no Geogebra



Fonte: Próprio Autor

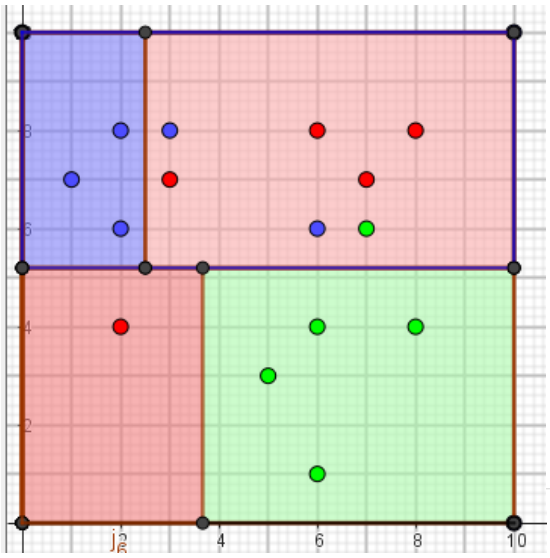
Figura 140 – Árvores de Decisão no Python



Fonte: Próprio Autor

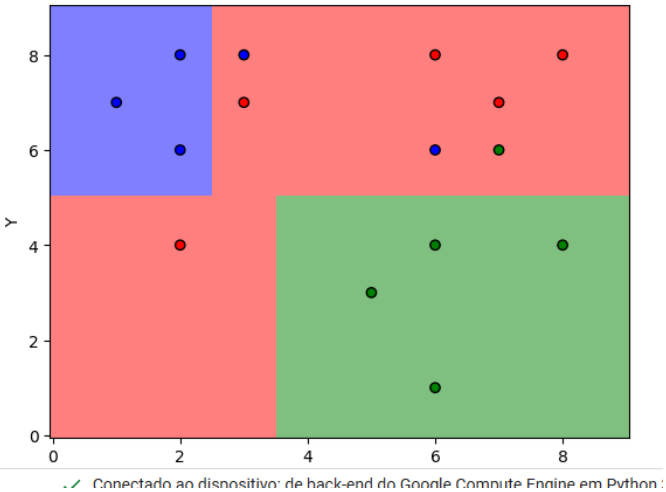
```
#    uniform     : uniform weights. All points in each neighborhood
   are weighted equally.
from matplotlib.colors import ListedColormap

classifier = DecisionTreeClassifier(max_depth=2).fit(X_treino,
   y_treino)

#Create color maps
cmap_light = ListedColormap(["red", "green", "blue"])
cmap_bold = ListedColormap(["red", "green", "blue"])

disp = DecisionBoundaryDisplay.from_estimator(
    classifier,
    X_treino,
    response_method="predict",
    plot_method="pcolormesh",
    xlabel=feature_names[0],
    ylabel=feature_names[1],
    shading="auto",
    cmap=cmap_light,
    alpha=0.5,
)
disp.ax_.scatter(X_treino.iloc[:, 0], X_treino.iloc[:, 1],
            c=y_treino.values.ravel(),cmap=cmap_bold, edgecolor="k")

plt.show()
```

Figura 141 – Mapa de Cores DT no Geogebra



Fonte: Próprio Autor

Figura 142 – Mapa de Cores DT no Python



Fonte: Próprio Autor

7 RESULTADOS

Neste capítulo apresentaremos os resultados dos capítulos 5 e 6.

7.1 RESULTADOS: PREVISÃO DE EVASÃO NO CURSO LICENCIATURA EM MATEMÁTICA DA UFTM

Nesta seção são apresentados os resultados para os 8 modelos de previsão de desistência do curso descritos no capítulo 5 e 4.

Os atributos escolhidos foram os que nos trazem informações variantes com o tempo, isto é, eles carregam informações temporais, exceto do TARGET. Os atributos Idade e Quantidade de Trancamentos sejam variantes com o tempo, não foram escolhidos, pois no caso do atributo Idade, no trabalho (SOUZA *et al.*, 2021), foi detectado que idade não seria relevante para a evasão dos alunos do curso de licenciatura em Matemática da UFTM. No caso do atributo quantidade de disciplinas canceladas por semestre ou quantidade de trancamento do curso, embora seja uma atributo importante para prever evasão, conforme os trabalhos (SOUZA *et al.*, 2021) e Ramon Nóbrega, fizemos a opção de não escolher este atributo, pois da forma que nos foi fornecido os dados, quando o aluno faz o trancamento Geral do curso não aparece a quantidade de disciplinas canceladas e acreditamos que isso poderia distorcer este atributo quantidade de disciplinas canceladas por semestre.

Verificamos a Correlação dos Dados e como a variável resposta(label ou Target) é dicotômica (binária), utiliza-se a correlação Ponto-Bisserial. Todas as variáveis (atributos) selecionados tem uma correlação 46, então optamos de deixar todos os atributos.

Todos os atributos foram verificados quanto a incidência de valores nulos, não foi encontrado nenhum registro com valores nulos. Os outliers, uma abordagem comum é simplesmente removê-los do conjunto de dados, descartando os pontos problemáticos antes das análises, porém como temos poucos dados fizemos a opção de deichá-los nos dados e aplicarmos transformações. Embora a suavização dos outliers foi melhor aplicando QuantileTransformer, não necessariamente obtemos um resultado melhor do que aplicar o StanderScaler.

Para os algoritmos: Regressão Logística, Máquina de Vetores de Suporte, k-Vizinhos Mais Próximos-KNN, Árvores de Decisão e Multilayer Perceptron-MLP aplicamos a técnica Holdout no conjunto de dados e também fizemos o balanceamento dos dados. Já para o LSTM a divisão dos dados foi feita através de uma função de criação de dados de treino e teste, esta separação não aleatória. Também não conseguimos balancear dos dados, pois os dados utilizados para LSTM seguem uma ordem temporal.

Neste trabalho avaliamos 6 algoritmos de aprendizado de máquina para detectar desistência (evasão) e conclusão do curso, são eles: Regressão Logística, Máquina de Vetores de Suporte, k-Vizinhos Mais Próximos-KNN, Árvores de Decisão, Multilayer Perceptron-MLP e LSTM num total de 24 datasets diferentes, gerando 144 modelos.

Os 24 conjunto de dados diferentes, vem da divisão do dataset principal em 8 datasets $(df1, df2, \cdots, df8)$, onde dfk indica o conjunto de dados dos alunos que cursaram pelo menos até o k-ésimo semestre e destes 8 datasets obtivemos 24 conjuntos de dados (8 desbalanceados, 8 balanceados pela técnica subamonstragem e 8 balanceados pela técnica sobreamostragem).

Para Regressão Logística, Máquina de Vetores de Suporte, k-Vizinhos Mais Próximos-KNN eÁrvores de Decisão, aplicamos 30 testes utilizando a proporção 85% para dados de treino e validação e 15% e para dados teste. Dos 85% para dados de treino e validação, separamos 80% para dados de treino e validação e 20%, dando uma porcentagem dos dados originais de 68% para dados de treino, 17% para validação e 15% e para dados teste.

Para Multilayer Perceptron-MLP e aplicamos 5 testes utilizando a proporção 80% para dados de treino e validação e 20% e para dados teste. Dos 80% para dados de treino e validação, separamos 80% para dados de treino e validação e 20%, dando uma porcentagem dos dados originais de 64% para dados de treino, 16% para validação e 20% e para dados teste.

Para LSTM aplicamos 3 testes utilizando mais ou menos uma proporção 80% para dados de treino e validação e 20% e para dados teste. Dos 80% para dados de treino e validação, separamos 80% para dados de treino e validação e 20%, dando uma porcetagem dos dados originais de 64% para dados de treino, 16% para validação e 20% e para dados teste.

Para avaliar os modelos utilizamos as métricas de desempenho: Acurácia, Precisão, Recall, f1-score, todas descritas por 2.3, 2.5, 2.7 e 2.8 no capítulo 4, olhando o intervalo de confiança dado pela média e desvio padrão. Para obter o intervalo de confiança da média e desvio padrão tivemos que aplicar 30 testes sem o $random\_state$ na técnica de separação de dados Holdout ($train\_test\_split$) para Regressão Logística, Máquina de Vetores de Suporte, k-Vizinhos Mais Próximos-KNN, Árvores de Decisão e 5 testes para Multilayer Perceptron-MLP e no caso do LSTM fizemos três separações da dados manualmente e aplicamos a média e desvio padrão para três dados.

Comparamos somente modelos de algoritmos aplicados no dataset dfk com k período, dentre estes modelos aplicados, não é possível dizer que na média um modelo é melhor do que o outro, embora tenhamos os intervalos de confiança com 95%, teríamos que fazer os testes da estatísticos da média, então para escolher o "'melhor"' modelo, olhamos para o intervalo de confiança e aqueles que tiveram sobreposição, escolhemos aquele que está mais a direita, isto é, entre (a, b) e $(a + \epsilon, b + \delta)$, escolhemos $(a + \epsilon, b + \delta)$. Dentre a métrica acurracy, precision, recall e f1-score para as classes 0 e 1, olhamos o intervalo de confiança

da recall para classe 0.

A escolha da métrica **recall** para o **rotulo 0** ($recall_0 = \frac{VN}{VN+FP}$), vem do fato que temos datasets dfk são desbalanceados nos primeiros períodos $k \in \{1, 2, 3\}$ e nos períodos finais $k \in \{6, 7, 8\}$, neste caso, melhor pegar recall separado para cada classe e no caso nosso o interesse é me desistência(evasão) conforme (JUNIOR, 2023), portanto pegamos o recall da classe 0.

Recall para classe 0 (desitência/evasão) é uma medida que mostra o quanto de alunos evadidos nos dados de testes estão previstas corretamente. Ela é definida como a razão entre o número total de alunos desistentes/evadidos previstos corretamente dividido pelo número total de desistentes/evadidos tem nos dados de testes.

$$recall_0 = \frac{desistentes\ previstos\ corretamente}{total\ de\ alunos\ desistentes}$$

7.1.1 Regressão Logística, Máquina de Vetores de Suporte, k-Vizinhos Mais Próximos-KNN, Árvores de Decisão e Multilayer Perceptron-MLP

Comparando os algoritmos Regressão Logística, Máquina de Vetores de Suporte, k-Vizinhos Mais Próximos-KNN, Árvores de Decisão e Multilayer Perceptron-MLP temos os seguintes melhores modelos.

Obs: Não comparamos os algoritmos LR, SVM, KNN, DT e MLP com LSTM, pois no LSTM não é fácil separar e não conseguimos separar dados de treino, validação e teste pela técnica aleatoriamente pela técnica Holdout no conjunto de dados. A separação de os dataset dfk foi manualmente. Ver 4.

Tabela 72 – Modelos com melhor recall

Modelo	Target	accuracy	precision	**recall**	f1score
SVM_{1NB} tunado Grid	0	$88,2\%$	$88,2\%$	**100%**	$93,8\%$
LR_{2NB} tunado Grid	0	$81,6\%$	$81,6\%$	**100%**	$89,9\%$
0 MLP_{3BOver} tunado	0	$78,9\%$	87%	**80%**	$83,3\%$
$SVM_{4BUnder}$ tunado Grid	0	$73,9\%$	$68,4\%$	**100%**	$81,2\%$
MLP_{5BOver} tunado Grid	0	$95,8\%$	100%	**90,9%**	$95,2\%$
$LR_{6BUnder}$ tunado Grid	0	40%	40%	**100%**	$57,1\%$
MLP_{7BOver} tunado Grid	0	$94,7\%$	$88,9\%$	**100%**	$94,1\%$
SVM_{8BOver} tunado Grid	0	$83,3\%$	$66,7\%$	**100%**	80%

Fonte: Próprio Autor.

Previsão utilizando dados do 1° semestre para prever quais alunos irão desistir nos próximos semestre. O modelo SVM_{1NB} previu corretamente os 63 desistentes e nenhuma previsão errônea de desistentes, isto é, previu o desistente como concluinte (FP). O conjunto

de teste 77 alunos: 63 desistentes e 14 concluintes, como descrito na figura da matriz de confusão deste modelo 54.

Previsão utilizando dados do 1° e 2° para prever quais alunos irão desistir/evadir nos próximos semestre. O modelo LR_{2NB} previu corretamente os 31 desistentes/evadidos e nenhuma previsão errônea de desistentes, isto é, previu o desistente como concluinte (FP). O conjunto de teste tem 38 alunos: 31 desistentes e 7 concluintes, como descrito na figura da matriz de confusão deste modelo 55.

Previsão utilizando dados do 1°,2° e 3° semestre para prever quais alunos irão desistir/evadir nos próximos semestre. O modelo SVM_{3NB} previu corretamente os 20 desistentes/evadidos e 5 previsão errônea de desistentes, isto é, previu o 5 desistentes como concluintes (FP). O conjunto de teste 38 alunos: 25 desistentes e 13 concluintes, como descrito na figura da matriz de confusão deste modelo 70.

Previsão utilizando dados dos 1°, 2°, 3° e 4° semestres para prever quais alunos irão desistir/evadir nos próximos semestre. O modelo $SVM_{4BUnder}$ previu corretamente os 12 desistentes/evadidos 1 previsão errônea de desistentes, isto é, previu o 5 desistentes como concluintes (FP). O conjunto de teste tem 23 alunos: 13 desistentes e 10 concluintes, como descrito na figura da matriz de confusão deste modelo 65.

Previsão utilizando dados dos 1°, 2°, 3°, 4° e 5° semestres para prever quais alunos irão desistir/evadir nos próximos semestre. O modelo MLP_{5BOver} previu corretamente os 11 desistentes (VN) e previu erroneamente 3 desistentes, isto é, previu como concluinte (FP). O conjunto de teste tem 30 alunos: 14 desistentes e 16 concluintes, como descrito na figura da matriz de confusão deste modelo 66.

Previsão utilizando dados dos 1°, 2°, 3°, 4°, 5° e 6° semestres para prever quais alunos irão desistir/evadir nos próximos semestre. O modelo $LR_{6BUnder}$ previu corretamente os 6 desistentes/evadidos e nenhuma previsão errônea de desistentes, isto é, previu o desistente como concluinte (FP). O conjunto de teste tem 15 alunos: 6 desistentes e 9 concluintes, como descrito na figura da matriz de confusão deste modelo 67.

Previsão utilizando dados dos 1°, 2°, 3°, 4°, 5°, 6° e 7° semestres para prever quais alunos irão desistir/evadir nos próximos semestre. O modelo MLP_{7BOver} previu corretamente os 8 desistentes/evadidos e nenhuma previsão errônea de desistentes, isto é, previu o desistente como concluinte (FP). O conjunto de teste tem 19 alunos: 8 desistentes e 11 concluintes, como descrito na figura da matriz de confusão deste modelo 68.

Previsão utilizando dados dos 1°, 2°, 3°, 4°, 5° e 6°, 7° e 8° semestres para prever quais alunos irão desistir/evadir nos próximos semestre. O modelo SVM_{8BOver} previu corretamente os 4 desistentes/evadidos e nenhuma previsão errônea de desistentes, isto é, previu o desistente como concluinte (FP). O conjunto de teste tem 12 alunos: 4 desistentes e 8 concluintes, como descrito na figura da matriz de confusão deste modelo 70.

7.2 RESULTADOS: RECUSRSO DIDÁTICO E TECNOLÓGICO COM GEOGEBRA E PYTHON

Nesta seção são apresentados os resultados do aplicativo KNN Geogebra e DT Geogebra expostos no capítulo 6.

Aplicações de Inteligência Artificial (IA) usando Aprendizado de Máquina (AM) está se tornando cada vez mais comum no nosso cotidiano. Isso despertou em nós a necessidade de desenvolver conhecimento de aprendizado de máquina na Educação Básica, dentro da lógica do Pensamento Computacional. Neste sentido, este trabalho apresenta o desenvolvimento de um material didático para ser aplicado em laboratórios de computadores das escolar por meio do software Geogebra e Google Colab. Foi desenvolvido dois aplicativos do Geogebra que simulam até um certo nível os algoritmos de aprendizado de máquina k-vizinhos mais próximos- KNN com k=3 e Árvore de Decisão até a profundidade 2, eles foram descritos no capítulo 6.

O problema de classificação apresentado no capítulo 6 foi classificar a flor de íris, porém os 21 pontos retirados do conjunto de 150 potnso da flor de iris estão muitos próximos, então para uma compreensão mais didática apresentamos um problema similar que é classificar os 21 pontos (7 vermelhos, 7 verdes e 7 azuis) conforme suas cores. .

Qual seria a vantagem de utilizar o aplicativo do KNN Geogebra e DT Geogebra ao invés de utilizar o Python diretamente?

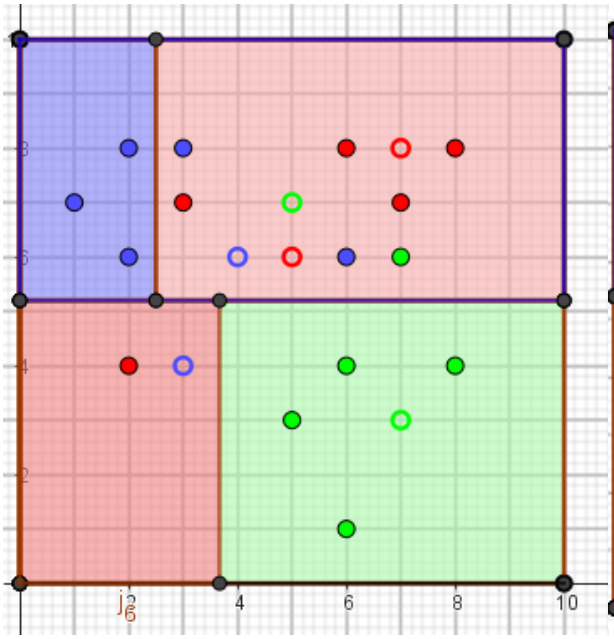
1. Mudar os pontos do conjunto de dados simulando outros situações
2. Verificar visualmente e dinamicamente como o KNN classifica os pontos utilizando k distâncias minimas.
3. Verificar visualmente e dinamicamente como a árvore de decisão é construída.

7.2.1 Mudar os pontos do conjunto de dados simulando outros situações

O fato do Geogebra ser um software dinâmico, o aluno pode mexer nos pontos do conjunto de dados de treinamento, simulando vários conjuntos de treinamento, como se fosse dados de treino obtidos randomicamente como no python através da técnica Holdout (testsplit).

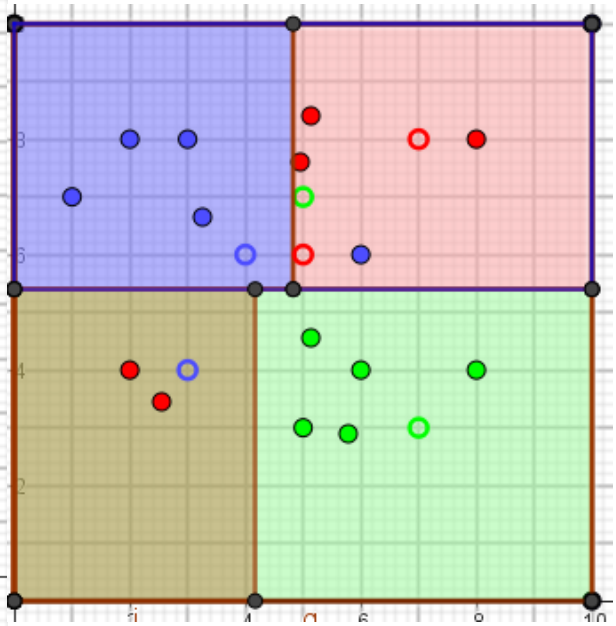
Na figura 143 temos 50% de acerto com o conjunto de pontos dados e na Na figura 144 temos 90% de acerto com o novo conjunto de pontos dados.

Figura 143 – Árvore de Decisão no Geogebra



Fonte: Próprio Autor

Figura 144 – Árvores de Decisão no Geogebra



Fonte: Próprio Autor

7.2.2 Verificar visualmente e dinamicamente como o KNN classifica os pontos utilizando k distâncias minimas

Quando construímos os aplicativo do KNN Geogebra, os primeiros passos são inserir um ponto P e calcular as distâncias entre o ponto P e todos os pontos de treinamento. Criando segmentos do ponto P até os 15 pontos, criaremos uma lista1 de 15 seguimentos com 15 medidas de comprimento e ordenando a lista com o comando "'Ordenar(lista1)"' você pode escolher as três primeiras medidas que serão os três segmentos com menores distâncias de P outros pontos, o Geogebra automaticamente habilita estes segmentos. Você deve desabilitar exibir objeto dos 15 segmentos. Com isso o Geogebra irá mostrar sempre os três segmentos com menores medidas, e ligará o ponto P aos pontos com as três menores distâncias. Como os 15 pontos tem 5 vermelhos, 5 verdes e 5 azuis, você poderá colorir o ponto P com a cor predominante das três cores dos pontos ligados pelos três menores segmentos até ele.

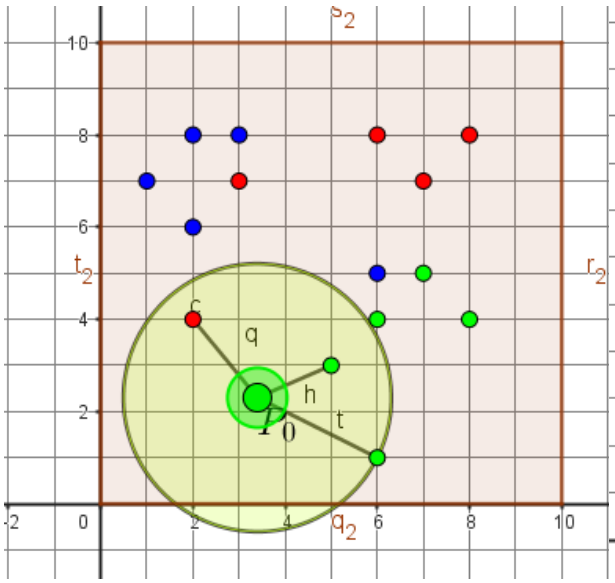
Veja a figura

7.2.3 Verificar visualmente e dinamicamente como a árvore de decisão é construída

A árvore de decisão é um tipo de diagrama te ajuda a entender melhor suas escolhas e os resultados. Visualizar sua construção pode ajudar no entendimento e na resolução do problema de classificação.

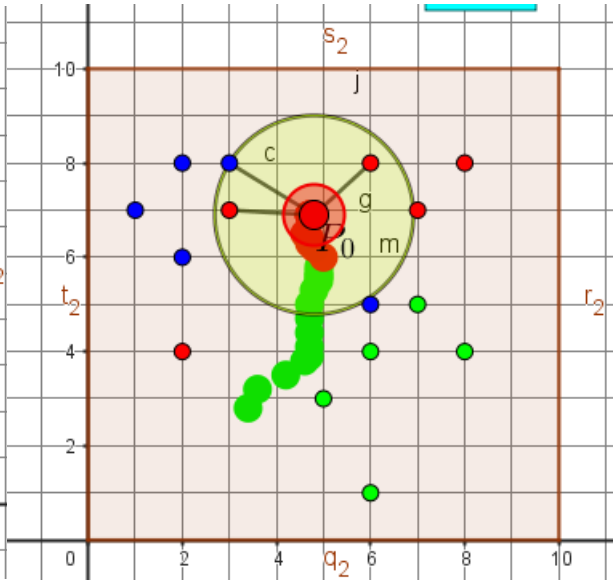
Na figura 147 tem a árvore com um nó raiz e dois nó folhas,

Figura 145 – KNN no Geogebra



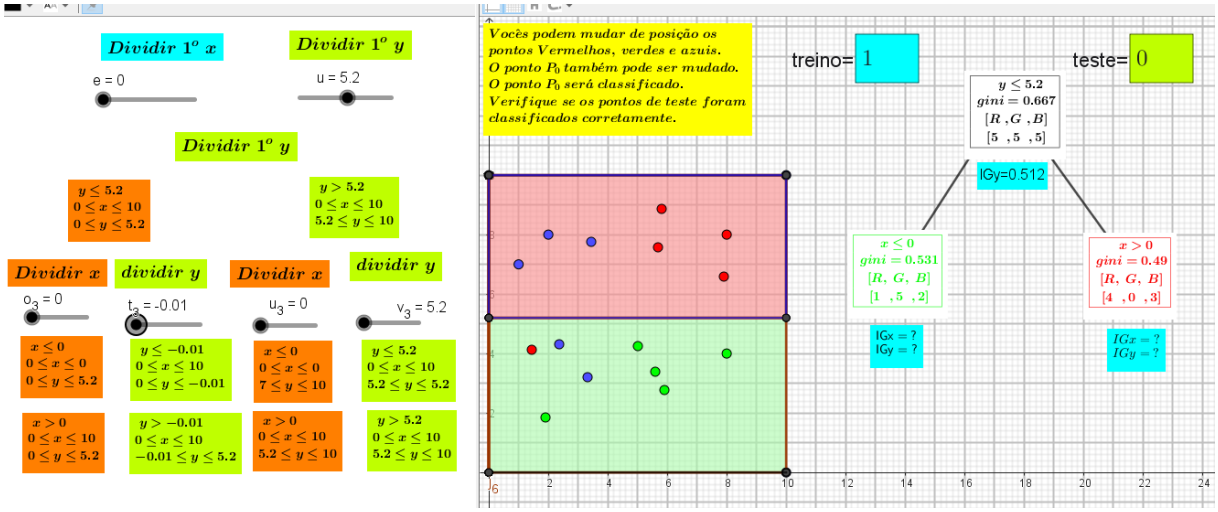
Fonte: Próprio Autor

Figura 146 – KNN no Geogebra



Fonte: Próprio Autor

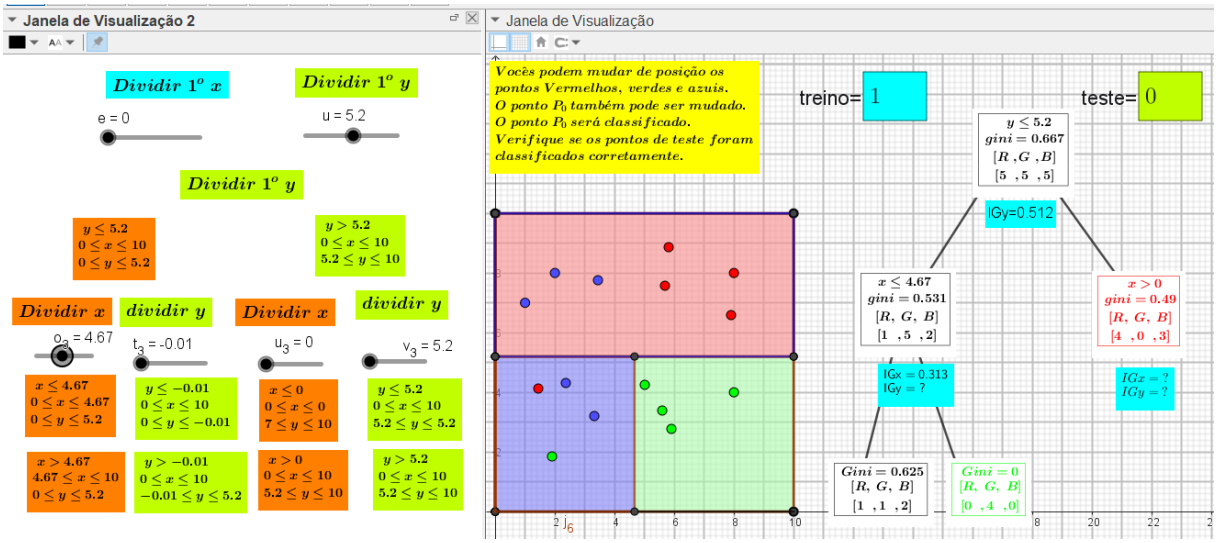
Figura 147 – Árvore de Decisão no Geogebra



Fonte: Próprio Autor

já na figura 148 temos uma árvore de decisão com um nó raiz, um nó de decisão e dois nó folhas.

Figura 148 – Árvores de Decisão no Geogebra



Fonte: Próprio Autor

8 CONSIDERAÇÕES FINAIS

Esta dissertação teve como principal objetivo criar um recurso didático tecnológico utilizando o software Geogebra para o ensino de conceitos de aprendizado de máquina no ensino médio e ao mesmo tempo analisar a evasão acadêmica no curso de Licenciatura em Matemática da UFTM, utilizando algoritmos LR, SVM, KNN, DT, MLP e LSTM para a previsão dessa evasão.

Sobre as atividades com o Geogebra, foi possível introduzir um recurso didático por meio de aplicativos no Geogebra, permitindo que os estudantes visualizem e compreendam de maneira prática a lógica por trás dos algoritmos de classificação KNN e Árvore de Decisões. Esse recurso, voltado para o ensino médio, pode ser eficaz na transmissão de conceitos complexos, como a localização de pontos no plano cartesiano, distância entre pontos e análise de desigualdades, além de apresentar aos alunos a ideia de aprendizado de máquina para a classificação de dados. Entre as principais contribuições deste estudo, destaca-se a integração de uma metodologia tecnológica no ensino da matemática, algo que pode promover o aprendizado de habilidades computacionais e matemáticas de forma mais acessível. Infelizmente as atividades não foram implementadas em sala de aula, o que impossibilitou analisar os desafios e dificuldades dos alunos, para poder aprimora-la.

Sobre evasão do curso de licenciatura em Matemática, foram desenvolvidos modelos preditivos baseados em algoritmos, como kNN, MLP (Redes Neurais), Árvore de Decisão, Regressão logística, SVM e LSTM. Foi analisado os dados dos alunos semestre a semestre, analisando o histórico completo do aluno do ingresso ao período atual com base em alguns atributos, para cada semestre utilizado o modelo de predição que apresentou melhores resultados conforme a métrica recall para classe 0. Foi observado que a evasão se dá mais nos primeiros períodos, porém ela pode ocorrer mas tardiamente também, no quinto, sexto e sétimo e até oitavo períodos.

Na etapa de treino, os melhores modelos foram escolhidos após aplicados nos dados de treinamento não balanceados e balanceado para cada dataset $df\,k$ de k semestres prevendo a desistência (ou evasão). As informações de cada aluno, isto é, as informações semestrais, foram transformados em uma média nos dadaset utilizados para todos algoritmos, exceto para o LSTM, pois o LSTM precisa de informação temporal. Os resultados aplicados em dados de treinamento e validação foram apresentados em forma de intervalo de confiança com 95% de precisão. Destes escolhemos os melhores olhando o intervalo mais a direita e não as médias e desvio padrão, pois para dizer que uma média é melhor do que outra, precisaríamos de testes estatísticos, então optamos pelo intervalo de confiança. O desvio padrão e média foram obtidos em treinamento e validação. Os modelos foram tunados e por fim o melhor modelo nos dados de teste. Os resultados foram de acordo com a métrica

recall para a classe 0 (desistentes).

Tabela 73 – Modelos com melhor recall

Modelo	Target	**recall**
SVM_{1NB} tunado Grid	0	**100%**
LR_{2NB} tunado Grid	0	**100%**
0 MLP_{3BOver} tunado	0	**80%**
$SVM_{4BUnder}$ tunado Grid	0	**100%**
MLP_{5BOver} tunado Grid	0	**90,9%**
$LR_{6BUnder}$ tunado Grid	0	**100%**
MLP_{7BOver} tunado Grid	0	**100%**
SVM_{8BOver} tunado Grid	0	**100%**

Fonte: Próprio Autor.

Observa que modelos apresentados se mostraram muito eficazes na predição de evasão, sendo assim, esse trabalho contribui com dados importantes sobre o comportamento dos estudantes no curso de Licenciatura em Matemática da UFTM, evidenciando padrões de evasão nos primeiros e últimos períodos do curso, esses dados podem contribuir para a tomada de decisão da instituição com os alunos com potencial de evasão. Por outro lado, algumas limitações foram observadas, como a necessidade de uma base de dados mais extensa e diversificada para aprimorar os modelos de previsão.

Para trabalhos futuros, espero poder ampliar o estudo para outros cursos e instituições, além de um aprofundamento nas técnicas de aprendizado de máquina. Assim como aplicar as atividades em sala de aula, buscando aprimorar essas ferramentas didáticas baseadas em aprendizado de máquina, buscando sempre aprimorar o processo de ensino-aprendizagem dos alunos por meio da tecnologia.

Referências

AMORIM, N. **Teste de Normalidade com Python**. <https://dev4lab.github.io/posts/teste-normalidade/>.

ASKINADZE, A.; CONRAD, S. Predicting student dropout in higher education based on previous exam results. **Proceedings of The 12th International Conference on Educational Data Mining**, p. 500–503, 2019.

BALDUíNO, V. A. P. **Análise de ativos financeiros por meio de Rede Neural Recorrente do tipo LSTM**. TCC, 2023.

BARBOSA, G. M. **Métodos de Aprendizado de Máquina a Evasão Escolar**. Dissertação de Mestrado, 2022.

BATISTA, I. **Testes de correlação**. <https://ivanildo-batista13.medium.com/testes-de-correla%C3%A7%C3%A3o-3cb0a37e0f2>.

BOUCINHA, R. M. **Aprendizagem do Pensamento Computacional e Desenvolvimento do Raciocínio**. Tese de Doutorado, 2017.

BRANDãO, D. A. Inteligência artificial e aprendizado de máquina: da teoria ao algoritmo pronto no ensino médio. 2024.

BRASIL. **Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (Inep): Resumo Técnico do Censo da Educação Superior 2008.** [S.l.]: Brasília-DF, 2009.

_____. **Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (Inep): Resumo Técnico: Censo da Educação Superior 2010-2023.** [S.l.]: Brasília-DF, 2023.

COUTINHO, B. **Aprenda a criar seu primeiro algoritmo de classificação com SVM.** Https://medium.com/turing-talks/turing-talks-12-classifica

DANTE, L. R. **Matemática: Contexto e Aplicações - Volume 3**. São Paulo: Editora Ática, 2016. ISBN 9788508163045.

FERRAZ, T. **Testes de Normalidade usando Python**. <https://thalesferraz.medium.com/testes-de-normalidade-usando-python-1abddbc9311f>.

FILHO, F. W. B. H.; VINUTO, T. S.; LEAL, B. C. Uma análise de modelos de séries temporais para predição de evasão discente: Estudo de caso do ifce. VII Congresso nacional de Educação, Conedu, 2021.

FILHO, R. L. L. S.; LOBO, M. B. d. C. M. **Esclarecimentos Metodológicos sobre os cálculos de Evasão**. Instituto Lobo. Acesso em: 24 de janeiro 2024. Disponível em: <https://www.institutolobo.org.br/core/uploads/artigos/art_078.pdf>.

FILHO, R. L. L. S. *et al.* **A Evasão no Ensino Superior Brasileiro.** Cadernos de Pesquisa, v. 37, n. 132, p. 641-659, 2007. Acesso em: 10 dez. 2019. Disponível em: <https://www.scielo.br/pdf/cp/v37n132/a0737132.pdf>.

FISHER, R. A. The use of multiple measurements in taxonomic problems. **Annals of eugenics**, Wiley Online Library, v. 7, n. 2, p. 179–188, 1936.

FORPLAD. **Fórum de Pró-Reitores de Planejamento e Administração, Comissão de Planejamento e Avaliação dos Instituições Federais de Ensino Superio**. Ouro Preto- MG: [s.n.], 2015. Acesso em: 24 de janeiro 2024. Disponível em: <https://www.uff.br/sites/default/files/indicadores_do_forplad.pdf>.

FáVERO, L. P.; BELFIORE, P. **Manual de Análise de Dados**. [S.l.]: Rio de Janeiro: Elsevier, 2017.

GOOGLE. **Colaboratory: A research tool for machine learning and data science**. 2023. <https://colab.research.google.com/>. Accessed: 2024-09-20.

JUNIOR, I. R. **Compreendendo a Matriz de Confusão**. 2023. Publicado em: 20/09/2023. Disponível em: <https://medium.com/@ingoreichertjr/interpretando-a-matriz-de-confus%C3%A3o-cecc6b3303dd>.

JUNIOR, J. R. F. **Redes Neurais Recorrentes-LSTM**. 2019. Acesso em: 17/01/2024. Disponível em: <https://medium.com/@web2ajax/redes-neurais-recorrentes-lstm-b90b720dc3f6>.

Kaggle. **Iris Flower Dataset**. 2024. Accessed: 2024-09-20. Disponível em: <https://www.kaggle.com/datasets/arshid/iris-flower-dataset/data>.

KALEFF, A. M.; NASCIMENTO, R. S. d. Atividades introdutórias às geometrias não-euclidianas: o exemplo da geometria do táxi. **BOLETIM GEPEM**, v. 44, 2004.

KNIGHT, S.; WISE A, F.; CHEN, B. Time for change: Whylearning analytics needs temporal analysis. jounal of Learning Analytics, 2021.

KOLI, S. **Decision Trees: A Complete Introduction With Examples**. Https://medium.com/MrBam44/decision-trees-91f61a42c724.

LEIVAS, J. C. P.; PORTELLA, H. P. d.; SOUZA, H. Geometrias não-euclidianas: uma investigação na escola básica no brasil com utilização do geogebra. **Revista Thema**, v. 14, 2017.

LORENA, A. C.; CARVALHO, A. C. P. L. F. de. Uma introdução às support vector machines. v. 14, p. 43–67, Dec. 2007. Disponível em: <https://seer.ufrgs.br/index.php/rita/article/view/rita_v14_n2_p43-67>.

LUZ, E. C. d. **O Uso do Google Earth e do GeoGebra no Ensino da Geometria do Táxi e da Geometria Euclidiana: uma abordagem com alunos da 3ª série do ensino médio da rede pública de Teresina-PI**. Dissertação de Mestrado, 2023.

MAHZOON, M. J. *et al.* **A sequende data model for analysing temporal patterns of student data**. Journal of Learning Analytics, 2018.

MARTINS, C. *et al.* Modelos de previsão de evasão tardia na graduação de uma universidade pública. In: **Anais do VIII Congresso sobre Tecnologias na Educação**. Porto Alegre, RS, Brasil: SBC, 2023. p. 41–50. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/ctrle/article/view/25782>.

MELO, A. L. **Uso da Técnica de Mineração de Dados como uma Ferramenta de Gestão da Evasão no Ensino Superior**. Dissertação (Mestrado) — Mestrado Profissional em Inovação Tecnológica, Universidade Federal do Triângulo Mineiro-UFTM, Uberaba, MG, 2019.

MOLENAAR, I. Advances in temporal analysis in learning and instruction. **Frontline Learning Research**, v. 2, n. 4, p. 15–24, Dec. 2014. Disponível em: <https://journals.sfu.ca/flr/index.php/journal/article/view/118>.

MURPHY, K. P. **Machine Learning: A Probabilistic Perspective**. Cambridge, MA: MIT Press, 2012. ISBN 9780262018029.

OLIVEIRA, R. d. S.; MEDEIROS, F. P. A. d. Modelo de predição de evasão escolar com base em dados de autoavaliação de cursos de graduação. **Revista Brasileira de Informática na Educação**, v. 32, p. 1–21, jan. 2024. Disponível em: <https://journals-sol.sbc.org.br/index.php/rbie/article/view/3542>.

ONODA, M.; EBECKEN, N. F. Implementação em java de um algoritmo de árvore de decisão acoplado a um sgbd relacional. In: **SBBD**. [S.l.: s.n.], 2001. p. 55–64.

PANTALONI, V. **Algorithme des k plus proches voisins - Iris KNN.** 2023. Accessed: 2024-01-06. Disponível em: <https://www.geogebra.org/m/ujfsvhtn>.

PETRICS, G. **The knn Supervised Machine Learning Algorithm**. 2020. Accessed: 2024-01-06. Disponível em: <https://www.geogebra.org/m/tneq8bsv>.

_____. **The Recursive Partitioning Supervised Machine Learning Algorithm**. 2020. Accessed: 2024-01-06. Disponível em: <https://www.geogebra.org/m/fqqk2zt3>.

RODRIGUES, E. H. A. **Análise de Caracterização Quantitativa e Predição da Evasão Escolar nos Cursos da Área de Computação do ICEA por meio de Técnicas de Data Science**. TCC, 2022. Disponível em: <https://monografias.ufop.br/bitstream/35400000/4414/6/MONOGRAFIA_AnaliseCaracterizacaoQuantitativa.pdf>.

SANTOS, C. H.; MARTINS, S.; PLASTINO, A. É possível prever evasão com base apenas no desempenho acadêmico? In: **Anais do XXXII Simpósio Brasileiro de Informática na Educação**. Porto Alegre, RS, Brasil: SBC, 2021. p. 792–802. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/sbie/article/view/18107>.

SILVA, A. R. d. **Uma visão geral sobre machine learning – Classificação**. Https://statplace.com.br/blog/uma-visao-geral-sobre-machine-learning/.

SILVA, F. d. **Reamostragem em modelos preditivos: separação treino e teste**. 2023. Publicado em: 31/07/2023. Disponível em: <https://analisemacro.com.br/econometria-e-machine-learning/reamostragem-em-modelos-preditivos-separacao-treino-e-teste/>.

SILVA, J.; SANTOS, M. Análise de distâncias em geometria: Distância euclidiana e distância de manhattan. **Revista Brasileira de Matemática Aplicada e Computacional**, v. 42, p. 45–60, 2020.

SOLUTIONS, s. p. d. c. M. **Explainable Artificial Intelligence (XAI). Desafios na interpretabilidade do modelo**. <https://www.managementsolutions.com/sites/default/files/minisite/static/22959b0f-b3da-47c8-9d5c-80ec3216552b/iax/pdf/explainable-artificial-intelligence-pt.pdf>.

SOUZA, F. G. d. C. **Previsão de evasão e retenção escolar no ensino médio profissional: uma abordagem baseada em redes neurais artificiais**. Dissertação de Mestrado — Instituto Federal do Sertão de Pernambuco, Salgueiro, Pernambuco-PE, 2021.

SOUZA, W. D. D. *et al.* Evasão em cursos de licenciatura de ciências exatas e naturais. Revista Triângulo-UFTM, 2021. Disponível em: <https://seer.uftm.edu.br/revistaeletronica/index.php/revistatriangulo/article/view/5461>.

TAVENARD, R. **An introduction to dynamic time warping**. [S.l.]: Dostupné z URL:< https://rtavenar. github. io/blog/fig/dtw_vs_euc. svg, 2022.

TEIXEIRA, L. A. (Ed.). **SuperAção Matemática - 7º Ano**. São Paulo: Editora Moderna, 2022. ISBN 9788516136321.

VIANA, F. S. V.; SANTANA, A. M.; RABêLO, R. d. A. L. Avaliação de classificadores para predição de evasão no ensino superior utilizando janela semestral. In: **Anais do XXXIII Simpósio Brasileiro de Informática na Educação (SBIE 2022)**. [S.l.]: XI Congresso Brasileiro de Informática na Educação, 2022.

VIEIRA, R. d. S. G. **Predição da evasão acadêmica aplicando análise temporal**. Universidade Federal de Goiás, 2021.