UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Investigação da Evasão Estudantil por meio da Mineração de Dados e Aprendizagem de Máquina

Dissertação de Mestrado

Jeferson Andrade de Jesus

São Cristóvão – Sergipe

2024

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Jeferson Andrade de Jesus

Investigação da Evasão Estudantil por meio da Mineração de Dados e Aprendizagem de Máquina

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): Dr. Renê Pereira de Gusmão

São Cristóvão – Sergipe

2024

Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.

Agradecimentos

Acima de tudo, agradeço a Deus por mais esta realização.

Agradeço a toda minha família, mãe, pai, namorada, vô, vó, tias e tios, amigos e ao professor Renê por toda a colaboração, paciência e conhecimentos passados durante o desenvolvimento deste trabalho

É fundamental que o estudante adquira uma compreensão e uma percepção nítida dos valores. Tem de aprender a ter um sentido bem definido do belo e do moralmente bom.
(Einstein)

Resumo

A evasão dos alunos nas escolas e universidades é um problema recorrente na educação, tanto é danoso para o aluno em termos de aprendizagem, como gera prejuízos financeiros para as instituições, sejam públicas ou privadas. Estudos que usam técnicas de Mineração de Dados (MD) e Aprendizagem de Máquina (AM) para investigar problemas na educação estão em ascensão, e a evasão estudantil é um desses problemas, por meio dessas técnicas é possível identificar padrões em indivíduos ou grupos de indivíduos que possam vir a desistir dos estudos. Com essas predições e outras informações exploradas pelas técnicas de MD e AM é possível diminuir a evasão dos alunos nas instituições de ensino, pois, o estudo ajuda a entender melhor o fenômeno e a partir disso verificar ações que ajudam na tomada de decisão e consequentemente na resolução do problema. Esse trabalho tem como objetivo investigar a evasão universitária através de dados do Censo da Educação Superior obtidos no portal do Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP) e analisar a classificação de evasão de alunos do ensino superior utilizando algoritmos clássicos na construção dos modelos de classificação. A investigação mostra que com base no Censo de 2016, 30% dos alunos do ensino superior chegam a desistir do curso ao qual ingressou ou a desistir do ensino superior, também é possível evidenciar que todos os estudos que utiliza classificadores de evasão estudantil como uma maneira de minimizar esse problema não utilizam conjuntos de dados com os mesmos atributos utilizados por outras instituições na construção dos seus modelos, de maneira que não é possível reaproveitar esses modelos para outras instituições. Considerando essas informações foram utilizados os dados do INEP na construção dos modelos de classificação construídos nos experimentos deste trabalho, tendo como melhor resultado para a métrica de acurácia: 0.979, resultado obtido pelo modelo baseado no algoritmo Árvore de Decisão, na modalidade Educação a Distância (EAD) e categoria administrativa pública.

Palavras-chave: Predição, classificação, evasão do estudante, aprendizagem de máquina, mineração de dados.

Abstract

A student dropout is a recurring problem in both schools and universities, posing detrimental effects on students' learning experiences and causing financial losses for institutions, whether public or private. Studies employing data mining (DM) and machine learning (ML) techniques to investigate educational issues are on the rise, and student dropout is one such problem. Through these techniques, it becomes possible to identify patterns in individuals or groups that may be prone to discontinuing their studies. Predictions and insights gained through DM and ML techniques can aid in reducing student dropout rates by providing a better understanding of the phenomenon, thereby informing decision-making and problem resolution. This study aims to investigate university dropout using data from the Higher Education Census obtained from the INEP portal. It analyzes the classification of university student dropout using classical algorithms in building classification models. The investigation reveals that, based on the 2016 Census, 30% of university students end up abandoning their enrolled courses or discontinuing higher education altogether. Additionally, it highlights that existing studies utilizing dropout classifiers lack a standardized set of attributes for constructing their models, making it challenging to apply these models across different institutions. To address these issues, the study employs INEP data in constructing classification models during the experiments. The best result, with an accuracy metric of 0.979, was achieved by the model based on the Decision Tree algorithm in the distance education (Educação a Distância - EAD) modality and public administrative category.

Keywords: Prediction, classification, student dropout, machine learning, data mining.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

1.1 Apresentação Geral

A evasão escolar gera sérias consequências sociais, acadêmicas e econômicas (BAGGI; LOPES, 2011). Trata-se da não concretização do objetivo inicial nos estudos, um fenômeno complexo com diversas razões subjacentes (FRITSCH; ROCHA; VITELLI, 2015). O mapeamento dessas razões é crucial para minimizar o problema, permitindo que o estudante alcance seus objetivos iniciais e, assim, reduzindo impactos negativos nos âmbitos social, acadêmico e econômico.

1.2 Motivação

Com o grande avanço tecnológico presente nos dias de hoje é cada vez mais comum empresas e pessoas buscarem meios tecnológicos facilitadores na resolução dos seus problemas, com esses meios ganha-se tempo, economia financeira, qualidade na solução, e em alguns casos uma melhor tomada de decisão que irá implicar em diversos benefícios.

É notável também o quanto sistemas inteligentes têm evoluído e ajudam na tomada de decisões importantes em qualquer setor. Na educação não é diferente, existem diversos sistemas que contribuem com essa área, sistemas que ajudam o professor a identificar qual metodologia de ensino foi mais eficaz para a aprendizagem dos alunos, sistemas que predizem qual será o desempenho de determinado aluno em determinada disciplina, etc. (PRADEEP; DAS; KIZHEKKETHOTTAM, 2015).

Partindo desse contexto tecnológico e educacional entramos em uma problemática muito crítica na educação superior brasileira, o alto índice de evasão estudantil no ensino superior, com base nos dados do Censo de Educação Superior obtidos no portal do INEP foi possível identificar que cerca de 30% dos alunos do ensino superior acabam por desistir definitivamente de sua

trajetória acadêmica, por diversos motivos. Esse alto índice de evasão acarreta grandes prejuízos para as instituições, sejam elas públicas ou privadas, e quando se trata de uma desistência do ensino superior e não de um curso o problema social é ainda mais grave, o mercado de trabalho está cada vez mais exigente na mesma medida que a demanda por busca de emprego aumenta, logo, não ter atualmente um diploma superior pode acarretar sequelas socioeconômicas para muitos jovens.

1.3 Aprendizagem de Máquina no Contexto da Evasão

A aprendizagem de máquina (AM) representa um campo da inteligência artificial que treina sistemas para extrair padrões, sem serem explicitamente programados para isso, aprendendo com dados e experiências passadas. Na análise da evasão estudantil, a AM oferece a capacidade de identificar comportamentos e fatores de risco, proporcionando uma visão dinâmica e adaptativa do fenômeno.

A utilização de técnicas de Mineração de Dados torna-se essencial nesse contexto, constituindo um conjunto de métodos para explorar grandes volumes de dados, buscando identificar padrões, tendências e correlações. Na investigação da evasão estudantil, a mineração de dados permite a identificação de fatores determinantes, fornecendo *insights* cruciais para a compreensão desse fenômeno complexo (COSTA et al., 2013).

Além disso, a Classificação, como componente vital da aprendizagem de máquina, envolve a categorização de dados em classes distintas. Na análise da evasão, a classificação possibilita a identificação de padrões comportamentais e a previsão de casos potenciais de desistência, contribuindo diretamente para o desenvolvimento de estratégias preventivas (PARDO; NUNES, 2002).

1.4 Justificativa

As hipóteses iniciais foram:

- H1. Verdadeira. Classificadores são capazes de predizer a evasão de alunos com bom desempenho.
- H1. Falsa. Classificadores não são capazes de predizer a evasão de alunos com bom desempenho.
- H2. Verdadeira. Modelos de classificadores podem ser reutilizados para diversas instituições.
- H2. Falsa. Modelos de classificadores não podem ser reutilizados para diversas instituições.

Tendo essa problemática da evasão como base e o objetivo de evidenciar as hipóteses citadas, foi realizado neste presente trabalho um mapeamento sistemático da literatura sobre o tema de evasão estudantil no nível superior. Assim, foi identificado que os modelos de classificação presentes nos experimentos em sua maioria não utilizam bases em comum, ou atributos em comum, para a construção desses modelos, já que muitos experimentos partem de estudos de casos de instituições específicas, com estruturas de dados específicas, logo, cada classificador necessita de dados de entrada com diferentes atributos, o que não viabiliza a utilização desses classificadores por qualquer instituição de ensino superior brasileira.

A busca realizada também mostra que nenhum estudo utiliza os dados do INEP [1] (Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira) para a construção dos modelos. Visando preencher essas lacunas, esse trabalho constrói modelos de classificação a partir dos dados do INEP, que possuem uma série de atributos disponíveis, o que padroniza a entrada dos modelos, já que todas as instituições de ensino superior precisam enviar os dados dos seus alunos para os censos do INEP.

Com modelos preditivos como esses construídos nos experimentos deste trabalho é possível receber como dado de entrada o mesmo registro que qualquer instituição de nível superior encaminha para o INEP durante o censo, facilitando dessa forma análises mais robustas sobre a evasão do ensino superior, pois as instituições estariam com seus dados padronizados tendo como base o mesmo modelo preditivo utilizado.

1.5 Objetivos

1.5.1 Objetivo Geral

O objetivo geral deste trabalho é investigar a evasão e analisar a classificação de evasão de alunos do ensino superior utilizando algoritmos clássicos na construção dos modelos de classificação.

1.5.2 Objetivos Específicos

- Analisar o conjunto de dados do Censo de Educação Superior retirados do INEP, para coleta de informações e estudos futuros;
- Analisar a classificação de evasão de alunos do ensino superior na modalidade EAD (Educação à distância);
- Analisar a classificação de evasão de alunos do ensino superior na modalidade presencial;

[1] https://www.gov.br/inep/pt-br/acesso-a-informacao/dados-abertos/microdados/censo-da-educacao-superior

- Analisar estatísticas descritivas sobre evasão de alunos no ensino superior entre os anos de 2016 e 2017;
- Analisar métricas como acurácia, especificidade, precisão e sensibilidade;
- Comparar o desempenho de classificadores.
- Identificar variáveis importantes para as predições.

1.6 Metodologia

Este trabalho de pesquisa está caracterizado como um estudo que contém objetivos exploratórios e descritivos, com abordagem quantitativa, composto por 3 partes.

A primeira parte é composta por um mapeamento sistemático da literatura, ao qual foi analisado o estado da arte do problema proposto, de maneira que seja possível obter o embasamento teórico necessário para desenvolver as próximas etapas.

A segunda parte consiste em elucidar o contexto da evasão estudantil, e as etapas e técnicas que participam do processo de construção do modelo preditivo para esse problema.

A terceira parte consiste em detalhar toda análise e processo experimental desenvolvido durante o decorrer do trabalho. Trazendo informações estatísticas retiradas dos dados brutos, especificando o processo de implementação como um todo e expondo métricas retornadas pelos modelos preditivos propostos.

1.7 Estrutura do documento

Para facilitar a navegação e melhor entendimento, este documento está estruturado nos seguintes capítulos:

- Capítulo 1 - Introdução: apresenta as definições preliminares da literatura, problemática, argumentações, objetivos e a metodologia aplicada.
- Capítulo 2 - Fundamentação teórica: expõe a contextualização teórica de pontos que serão fundamentais para a construção da dissertação de mestrado com o tema explorado neste trabalho.
- Capítulo 3 - Revisão de Literatura: demonstra todo o mapeamento sistemático feito sobre o tema de evasão estudantil e alguns trabalhos empíricos semelhantes ao que foi realizado de forma experimental neste trabalho.
- Capítulo 4 - Desenvolvimento: detalha os experimentos feitos, explora alguns números observados nos experimentos e também apresenta o cronograma completo de todas as etapas.

- Capítulo 5 - Resultados: apresenta os resultados de métricas variadas e interpreta esses resultados.
- Capítulo 6 - Considerações Finais: Resume o que foi feito nesse trabalho e quais as pretensões futuras.

2

Fundamentação Teórica

Este capítulo apresenta a fundamentação teórica deste trabalho, expondo problemáticas, informações, causas e técnicas que possam auxiliar na resolução dos problemas.

2.1 Evasão Estudantil

Como dito por Filho et al. (2007), a evasão torna-se um desperdício social, acadêmico e econômico para as instituições e para os alunos. Na literatura, é perceptível que o que leva os alunos a evadirem são fatores que podem estar relacionados a fatores institucionais, pessoais e externos, como no trabalho de Nandeshwar, Menzies e Nelson (2011), que segundo eles, para a instituição estudada, o histórico familiar e a situação socioeconômica da família são fundamentais para a permanência do aluno. Por outro lado, para Pradeep, Das e Kizhekkethottam (2015), em sua análise, alguns fatores que contribuem para a evasão escolar são aspectos culturais, dados demográficos, perfil psicológico, processo acadêmico, e também histórico social, familiar e educacional.

Na evasão estudantil existem dois tipos de fatores que podem contribuir com a evasão do aluno: os fatores anteriores ao ingresso do aluno e os fatores posteriores. Fatores anteriores podem ser local de residência (bairro, cidade, estado, etc.), condições socioeconômicas, desempenho acadêmico em anos anteriores, idade, etnia, sexo, etc.; fatores posteriores podem ser notas, interação com os ambientes de aprendizagem e comunicação das instituições, campus e instituição em que estuda, participação em monitoria, participação em atividades extracurriculares, etc. (FRITSCH; ROCHA; VITELLI, 2015).

2.1.1 Problemas causados pela evasão

> Para a instituição, acarreta ociosidade do espaço físico, de professores, de funcionários e de equipamentos, o que, nas instituições públicas, se reflete em desperdícios dos investimentos do governo e, nas particulares, redução no recebimento de mensalidade. Para os estudantes, por sua vez, a evasão pode representar o atraso ou cancelamento de um sonho, perda de oportunidades de trabalho, de crescimento pessoal e de melhoria de renda, entre muitas outras consequências (TONTINI; WALTER, 2014, p. 90).

Tontini e Walter (2014) mostram os principais problemas causados pela evasão estudantil, problemas esses que podem ser minimizados tanto ao prever a possível evasão de um aluno e tentar reter esse aluno, ou ao ter mensuração prévia de quantos alunos podem evadir e assim estudar estratégias que possam minimizar os estragos causados por essas perdas.

2.1.2 Estatísticas sobre evasão no ensino superior

Ao considerarmos as modalidades presencial e EAD do ensino superior, segundo o Censo da Educação Superior de 2016 existiam cerca de 11 milhões de alunos, e 30% desistiram até o ano de 2017, é uma informação trágica, ainda que nessa estatística estejam englobados alunos que mudaram de curso, porém não evadiram do ensino superior, os prejuízos ainda assim são enormes, como já foi exemplificado na seção anterior, a imagem abaixo traz algumas informações a respeito da evasão no ensino superior para o ano de 2016.

Figura 1 – Evasão no Ensino Superior no Ano de 2016

Fonte: (GARGANTINI, 2019)

Para compreender o cenário atual em números absolutos no ensino superior, foi realizado um comparativo com um estudo realizado por Filho et al. (2007), este apresenta informações relevantes sobre o Censo do ensino superior modalidade presencial para o ano de 2005, em contrapartida, esse trabalho apresenta informações sobre o Censo de 2016, as Figuras 2 e 3 apresentam esse comparativo.

Tipo de Instituição de Ensino Superior (IES)		IES		Cursos		Alunos		Razões		
		N	%	N	%	N	%	Cursos/ IES	Alunos/ IES	Alunos/ Curso
Totais		2.165	100,0	20.407	100,0	4.453.156	100,0	9	2.057	218
Públicas	Universidade	90	4,2	5.412	26,5	1.042.816	23,0	60	11587	193
	Centro Universitário	3	0,1	43	0,2	15.757	0,0	14	5.252	366
	Faculdade	138	6,4	736	3,6	133616	3,0	5	968	182
	Total	231	10,7	6.191	30,3	1.192.189	27,0	27	5.161	193
Privadas	Universidade	86	4,0	5.480	26,9	1.426.962	32,0	64	16593	260
	Centro Universitário	111	5,1	2.499	12,2	659.170	15,0	23	5.938	264
	Faculdade	1.737	80,2	6237	30,6	1.174.835	26,0	4	676	188
	Total	1.934	89,3	14.216	69,7	3.260.967	73,0	7	1.544	210

Figura 2 – Cenário do Ensino Superior Presencial do Brasil em 2005

Fonte: (FILHO et al., 2007)

Tipo de Instituição de Ensino Superior (IES)		IES		Cursos		Alunos		Razões		
		N	%	N	%	N	%	Cursos/IES	Alunos/IES	Alunos/Curso
Totais		2.346	100%	32.883	100%	8.815.580	100%	14	3.757	268
Públicas	Universidade	103	4%	8.443	26%	1.883.305	21%	81	18.284	223
	Centro Universitário	4	0%	50	0%	10.185	0%	12	2.546	203
	Faculdade	128	5%	497	2%	136.335	2%	3	1.065	274
	Total	235	10%	8.990	27%	2.029.825	23%	38	8.637	225
Privadas	Universidade	89	4%	7.325	22%	2.483.333	28%	82	27.902	339
	Centro Universitário	156	7%	4.793	15%	1.511.215	17%	30	9.687	315
	Faculdade	1.866	80%	11.775	36%	2.791.207	32%	6	1.495	237
	Total	2.111	90%	23.893	73%	6.785.755	77%	11	3.214	284

Figura 3 – Cenário do Ensino Superior Presencial do Brasil em 2016

Observando as tabelas acima, é possível notar que houve um crescimento geral, de Instituições de Ensino Superior (IES), cursos e alunos, do Censo de 2016, se comparado ao Censo de 2005, tanto no âmbito público como no privado. Entretanto, é perceptível que alguns indicadores apresentaram um crescimento desproporcional, especialmente nas instituições privadas. A razão aluno/IES para faculdades privadas mais que dobrou, o que pode ser considerado um dos fatores internos que contribuem para a evasão estudantil na instituição.

2.2 Aprendizagem de Máquina

A seção de Aprendizagem de Máquina é dedicada à exploração detalhada das técnicas fundamentais aplicadas em nossos experimentos de classificação de evasão estudantil. Ao longo

desta seção, serão apresentadas estratégias e abordagens específicas visando otimizar a precisão e a generalização de nossos modelos preditivos.

Dentre as técnicas discutidas, destacamos a utilização do *Cross Validator k-fold*, uma prática essencial para avaliação robusta de modelos. Este método, ao dividir o conjunto de dados em subconjuntos de treino e teste de maneira iterativa, permite uma análise abrangente da performance do modelo em diferentes configurações, contribuindo para a confiabilidade dos resultados.

Além disso, exploraremos estratégias de balanceamento e desbalanceamento por meio do *Undersampling*, técnica que busca atenuar disparidades entre classes majoritárias e minoritárias. A aplicação criteriosa dessas práticas é crucial para assegurar que nosso modelo seja treinado e avaliado de maneira equitativa, considerando todas as classes envolvidas.

A seção também abordará a condução de testes de hipótese, utilizando, por exemplo, o teste de t-Student, uma ferramenta estatística relevante para avaliar a significância das diferenças observadas entre grupos de dados.

Compreender e aplicar essas técnicas é fundamental para o sucesso de nossos experimentos, uma vez que proporcionam uma base sólida para a análise crítica dos resultados e a validação adequada de nossos modelos. Através dessa abordagem meticulosa em Aprendizagem de Máquina, buscamos não apenas construir modelos preditivos precisos, mas também garantir a confiabilidade e a robustez das conclusões extraídas em nosso estudo sobre evasão estudantil.

2.2.1 Cross-validation

A validação de modelos é uma etapa crucial no processo de construção de sistemas de aprendizado de máquina, visando avaliar a capacidade de generalização do modelo para dados não vistos. Entre as práticas essenciais nesse contexto, destaca-se o k-fold cross-validation, uma técnica robusta para a validação de desempenho de modelos preditivos.

O k-fold cross-validation opera através da divisão do conjunto de dados em k subconjuntos (ou "folds") de tamanhos aproximadamente iguais. O modelo é então treinado k vezes, cada vez utilizando k-1 subconjuntos como dados de treino e o subconjunto restante como dados de teste. Esse processo é repetido k vezes, cada vez utilizando um subconjunto diferente como conjunto de teste.

Ao final das k iterações, os resultados são agregados, proporcionando uma métrica de desempenho mais confiável e robusta. Essa abordagem é particularmente útil quando os conjuntos de dados disponíveis são limitados, permitindo uma melhor utilização das informações disponíveis (ANGUITA et al., 2012).

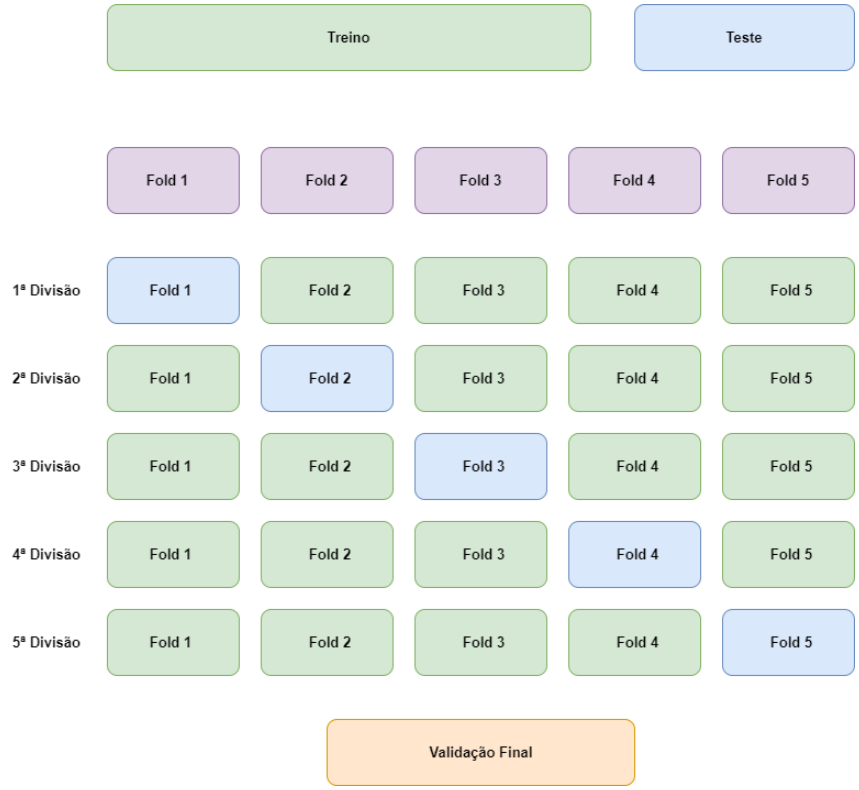

Figura 4 – Cross Validation K-fold

A aplicação do Cross Validator k-fold em nossos experimentos de classificação desempenha um papel crucial, garantindo que a avaliação de nossos modelos seja representativa e generalizável. Essa técnica não apenas oferece uma visão mais abrangente do desempenho do modelo, mas também contribui para a minimização do risco de overfitting, assegurando que o modelo seja capaz de lidar eficientemente com novos dados.

Ao adotar o Cross Validator k-fold como parte integral de nossa metodologia de aprendizado de máquina, buscamos validar e aprimorar a confiabilidade de nossos modelos preditivos, garantindo uma análise mais robusta e representativa no contexto da evasão estudantil (NAGY; MOLONTAY, 2018).

2.2.2 Seleção de Atributos

A Seleção de Atributos é uma prática essencial em projetos de aprendizado de máquina, desempenhando um papel crucial na melhoria do desempenho e da eficiência dos modelos preditivos. Essa técnica refere-se ao processo de escolher um subconjunto relevante de atributos a serem utilizados no treinamento do modelo, descartando aqueles que são redundantes, irrelevantes ou que podem introduzir ruído no processo de classificação.

A decisão cuidadosa sobre quais atributos incluir ou excluir pode resultar em modelos mais simples, interpretáveis e computacionalmente eficientes. Além disso, a seleção de atributos contribui para mitigar possíveis problemas, como a maldição da dimensionalidade, especialmente

quando o conjunto de dados possui um grande número de características (HEGDE; PRAGEETH, 2018).

Ao identificar e eliminar variáveis que possam levar a resultados imprecisos ou prejudicar a generalização do modelo para novos dados, busca-se construir modelos mais eficientes, interpretáveis e adaptáveis.

Essa prática de Seleção de Atributos não apenas aprimora a eficácia dos algoritmos utilizados, mas também proporciona uma compreensão mais profunda da influência de cada variável no processo de classificação. Através dessa técnica, visamos contribuir para a robustez e confiabilidade dos resultados na análise de evasão estudantil.

2.2.3 Undersampling

O balanceamento de conjuntos de dados desempenha um papel crucial em projetos de aprendizado de máquina, especialmente em tarefas de classificação, onde desequilíbrios nas classes podem afetar significativamente o desempenho do modelo. A técnica de *Undersampling* emerge como uma estratégia eficaz para abordar desbalanceamentos, mitigando disparidades entre classes majoritárias e minoritárias.

O *Undersampling* envolve a redução da quantidade de instâncias na classe majoritária, ajustando-a para níveis mais próximos à classe minoritária. Essa abordagem é particularmente útil quando a classe minoritária é de interesse crítico e desejamos evitar que o modelo seja enviesado em favor da classe majoritária (MOHAMMED; RAWASHDEH; ABDULLAH, 2020).

Ao equilibrar cuidadosamente as proporções entre as classes, o *Undersampling* contribui para a melhoria da capacidade preditiva do modelo, assegurando que o mesmo não seja tendencioso em direção à classe majoritária. Essa técnica é fundamental em cenários onde a classe de evasão estudantil pode ser substancialmente menor que a classe de não evasão.

Ao explorar a prática de Balanceamento/Desbalanceamento com *Undersampling*, buscamos garantir que nosso modelo seja treinado e avaliado de maneira justa, considerando adequadamente todas as classes envolvidas. Essa estratégia desempenha um papel essencial na promoção da precisão e confiabilidade dos resultados na análise de evasão estudantil.

2.2.4 Teste de Hipótese t-Student

O Teste de Hipótese t-Student é uma ferramenta estatística valiosa empregada para avaliar a significância das diferenças observadas entre dois grupos de dados. Em nosso contexto de análise de evasão estudantil, a aplicação desse teste oferece uma abordagem estatisticamente fundamentada para validar e interpretar os resultados de nossos experimentos.

A hipótese nula (H_0) no Teste t-Student postula a igualdade das médias entre dois grupos, enquanto a hipótese alternativa (H_1) sugere que existe uma diferença significativa entre eles. A

aplicação desse teste permite-nos determinar se as variações observadas em nossos dados são estatisticamente significativas ou se podem ter ocorrido por acaso.

Esse teste é particularmente relevante em cenários onde desejamos comparar medidas de desempenho, como a acurácia de modelos de predição, entre diferentes abordagens ou conjuntos de dados. A interpretação dos resultados do teste de t-Student fornece uma base sólida para afirmar se as diferenças observadas são estatisticamente significativas, contribuindo para a confiabilidade das conclusões tiradas de nossos experimentos.

Ao incorporar o Teste de Hipótese t-Student em nossa metodologia, buscamos validar de maneira rigorosa as discrepâncias identificadas em nossos dados, garantindo que as conclusões extraídas sejam estatisticamente fundamentadas. Essa abordagem robusta fortalece a validade estatística de nosso estudo sobre evasão estudantil, contribuindo para a solidez e confiabilidade de nossos resultados (OLIVEIRA et al., 2020).

2.3 Mineração de dados educacionais

Segundo Costa et al. (2013), a Mineração de Dados (MD, do Inglês, **Data Mining**, DM), pode ser vista como uma etapa principal de um processo mais amplo de conhecido como descoberta de conhecimento em bases de dados. Ele explica que há uma falta de consenso entre os autores sobre uma definição para o termo Mineração de Dados, dificultando assim uma consolidação de definição única. Contudo, diferente da definição de Mineração de Dados a sua estrutura possui um consenso na comunidade científica, o processo que forma a Mineração de dados pode ser dividido nas seguintes etapas:

1. **Coleta dos Dados**: nesta fase os dados relacionados ao domínio da aplicação final são coletados.

2. **Pré-processamento**: consiste em um conjunto de ações realizadas sobre o conjunto de dados obtidos a partir da etapa anterior, com o objetivo de prepará-los para extração de conhecimento.

3. **Extração de Conhecimento**: utiliza-se alguns algoritmos de aprendizado com o objetivo de extrair conhecimento na forma de regras de associação, relações, segmentação, etc., isso a partir de dados pré-processados.

4. **Avaliação e Interpretação dos Resultados**: nessa etapa os resultados obtidos são analisados, filtrados e selecionados para que o usuário possa ter um melhor entendimento dos dados coletados. Esse entendimento maior pode auxiliar em algum processo de tomada de decisão.

Assim como a MD, a Mineração de Dados Educacionais (MDE) que será explorada neste trabalho tem definições que divergem. Segundo Goldschmidt, Passos e Bezerra (2015), MDE é dita como um conjunto de esforços para a descoberta de conhecimento a partir de

dados de bases educacionais que transforma dados brutos em informações úteis, informações essas que contribuem tanto para a pesquisa em educação como para a prática de processo educacional, podendo ser utilizado para descrição e identificação de características estudantis, via classificação ou clusterização. Já Baker, Isotani e Carvalho (2011), diz que a aplicação da MDE tem como objetivo a descoberta de informações que ajudem na proposta educacional, no melhoramento das condições de infraestrutura escolar, no processo ensino-aprendizagem, na previsão de desempenho dos alunos, além de outros fatores que influenciam a aprendizagem. Visto que MDE pode ser definida de várias formas, mas que todas essas definições possuem um núcleo semelhante, a definição de MDE pode ser resumida em um conjunto de técnicas de MD que serão aplicadas no meio educacional, com o objetivo de gerar informações úteis que possam contribuir para os avanços educacionais.

2.4 Aprendizagem supervisionada

Segundo Monard e Baranauskas (2003), Aprendizado de Máquina (AM) é uma área da Inteligência Artificial cujo objetivo é a construção de sistemas capazes de adquirir conhecimentos de forma automática, e dentro do AM, hierarquicamente, uma das divisões do aprendizado indutivo é o supervisionado, em que o algoritmo de aprendizado já é fornecido, ou seja, na aprendizagem supervisionada o conjunto de dados ao qual se deseja coletar informações já vem previamente rotulado.

2.4.1 Classificação

O objetivo de um algoritmo de indução é construir um classificador que tem como finalidade estimar uma classe para uma nova instância. Esta rotulação em classes discretas é chamada de classificação. A classificação pode ser dividida em dois tipos: multiclasse e binária (PARDO; NUNES, 2002).

2.4.2 Algoritmos de Classificação

Existem vários paradigmas com que a AM trabalha, alguns deles são: neural, genético, estatístico. O paradigma que será explorado neste trabalho é o estatístico, que utiliza métodos estatísticos e probabilísticos com objetivo de encontrar uma boa solução do conceito induzido. Alguns dos algoritmos estatísticos mais conhecidos são: Logistic Regression, Decision Tree, Naive Bayes, SVM (Support Vector Machine) e KNN (K - Nearest Neighbors).

2.4.2.1 Logistic Regression

A regressão logística pode ser considerada uma extensão da regressão linear, pois assim como na regressão linear, ela estuda relações entre variáveis, buscando as variáveis que podem influenciar de alguma maneira em uma variável dependente, sendo que na regressão logística

essa variável dependente deve ser categórica. Enquanto a regressão linear dá uma resposta em valor numérico, a regressão logística dá uma resposta em probabilidade de chances de ocorrer o fato que está sendo trabalhado (BATISTELA; RODRIGUES; BONONI, 2009).

2.4.2.2 Decision Tree

A Decision Tree é um também um algoritmo que segue o paradigma estatístico e que utiliza treinamento supervisionado para construir a árvore, de maneira que cada nó interno pode ser visto como um teste de atributo e as folhas da árvore representam um determinado rótulo. Assim, a árvore irá classificar uma nova instância de acordo com o caminho percorrido na árvore que termina em um nó folha, atribuindo à nova instância o rótulo do mesmo (COSTA et al., 2013).

2.4.2.3 Random Forest

Random Forest (Floresta Aleatória) é um poderoso algoritmo de aprendizado de máquina que pertence à categoria de métodos de ensemble. A ideia principal por trás do Random Forest é construir várias árvores de decisão durante o treinamento e combiná-las para obter uma predição mais robusta e geral. Cada árvore na floresta é treinada em uma amostra aleatória do conjunto de dados, e as previsões individuais são agregadas para produzir uma decisão final (CUTLER; CUTLER; STEVENS, 2012).

2.4.2.4 Support Vector Machine (SVM)

As Máquinas de Vetores de Suporte (SVM) são um conjunto de algoritmos de aprendizado de máquina que são amplamente utilizados tanto em tarefas de classificação quanto de regressão. O objetivo principal do SVM é encontrar um hiperplano ótimo que separe os pontos de dados em classes distintas no espaço de características. Esse hiperplano é aquele que maximiza a margem entre as classes, ou seja, a distância entre os pontos mais próximos de cada classe (NEUMANN; SCHNÖRR; STEIDL, 2005).

2.5 Mapeamento sistemático

Nesta seção, é descrito detalhadamente o processo de Mapeamento Sistemático da Literatura (SLM, Systematic Literature Mapping) (PETERSEN et al., 2008). O Mapeamento Sistemático é uma abordagem metodológica rigorosa para identificar, avaliar e sintetizar estudos relevantes em uma área específica.

2.5.1 Método de Mapeamento Sistemático (SLM)

O método de Mapeamento Sistemático da Literatura (SLM) segue uma estrutura bem definida, compreendendo as seguintes etapas:

2.5.1.1 Questões de Pesquisa

Antes de iniciar o mapeamento, foram formuladas questões de pesquisa específicas para orientar a identificação e a seleção dos estudos. Estas questões foram cuidadosamente elaboradas para abordar aspectos-chave do tema em estudo.

2.5.1.2 Identificação de Estudos

A fase de identificação envolveu a busca sistemática em bases de dados relevantes, utilizando palavras-chave e critérios de inclusão/exclusão predefinidos. A seleção de fontes foi realizada de maneira a abranger estudos que contribuíssem significativamente para o entendimento do problema em questão.

2.5.1.3 Seleção e Avaliação dos Estudos

Os estudos identificados foram submetidos a um processo rigoroso de seleção e avaliação, levando em consideração critérios de qualidade e relevância. Essa etapa visa garantir a inclusão apenas de estudos que atendam aos padrões necessários para contribuir de maneira significativa para o mapeamento.

2.5.1.4 Síntese dos Dados e Apresentação dos Resultados

Após a seleção dos estudos, os dados foram sintetizados e os resultados foram organizados de maneira clara e objetiva. Esta fase proporciona uma visão abrangente do estado atual da literatura no domínio em estudo.

2.5.2 Apresentação do Mapeamento

Os resultados completos do Mapeamento Sistemático da Literatura serão apresentados de forma abrangente no próximo capítulo. Este mapeamento não apenas estabelece um panorama amplo do campo, mas também serve como base para o desenvolvimento do embasamento teórico necessário para este trabalho.

3

Revisão de Literatura

Esse capítulo tratará de apresentar todo o mapeamento sistemático realizado para construção da dissertação e também trabalhos empíricos aos quais seus experimentos se assemelham aos realizados nesse presente trabalho, sintetizando dessa forma no mapeamento sistemático a bibliografia de trabalhos que tem como tema a evasão escolar e dando exemplo de trabalhos empíricos relacionados.

3.1 Investigação da Evasão Estudantil por meio da Mineração de Dados e Aprendizagem de Máquina: Um Mapeamento Sistemático

3.1.1 Introdução

O problema da evasão estudantil tem muitas faces e afeta instituições em diferentes níveis de ensino em todo o mundo. Trata-se de um problema crítico com características especiais, que requer abordagens inovadoras, como a adoção das técnicas de mineração de dados educacionais (*Educational Data Mining* - EDM). Com o uso da EDM é possível ter um panorama dos índices de evasão escolar. No Brasil, as pesquisas sobre a questão da evasão usando EDM ainda estão em seu início. Contudo, se considerarmos o cenário internacional, diversas pesquisas expressivas utilizando EDM têm sido realizadas (MANHÃES; CRUZ; ZIMBRÃO, 2014a).

Com o grande avanço da Inteligência Artificial (IA) na educação, mais especificamente a aplicação da Mineração de Dados (do inglês, *Data Mining* - DM) e o Aprendizado de Máquina (do inglês, *Machine Learning* - ML), é crescente a tentativa das instituições de ensino usarem da quantidade enorme de dados sobre seus alunos para desenvolverem soluções robustas com o objetivo de solucionar alguns problemas, e a evasão de alunos é um dos principais, pois gera muitos prejuízos, econômicos, acadêmicos e sociais (BAGGI; LOPES, 2011). EDM e ML são

recursos relevantes para identificar os motivos da evasão de um aluno. Algoritmos clássicos de aprendizagem de máquina, como Regressão Logística, Árvores de Decisão, Naive Bayes e Máquinas de Vetores de Suporte, são comumente utilizados em tarefas de predição. Com alguns estudos identificando os benefícios desses recursos, alavancaram-se fortemente as pesquisas nesse tema específico. A procura por melhores algoritmos, técnicas e modelos que possam classificar se, futuramente, aquele aluno vai abandonar ou não os estudos. Tendo essa visão prévia, as instituições poderiam planejar a melhor forma de evitar a evasão desse aluno. Essa tomada de decisão também é um problema ao qual a aplicação de IA tem sido investigada, mas que não será tratado neste artigo.

O objetivo deste capítulo é apresentar um processo de mapeamento sistemático sobre Evasão Estudantil por meio da Mineração de Dados e Aprendizagem de Máquina para mapear estudos relevantes sobre a aplicação de DM e ML na classificação de dados em estudos sobre evasão escolar, de forma a caracterizar a evolução dessa área de pesquisa. Assim, este trabalho está organizado da seguinte forma: na seção 3.1.2 são apresentados os trabalhos relacionados. A seção 3.1.3 apresenta a condução desse mapeamento. Na seção 3.1.4 é apresentada a análise dos resultados. Na seção 3.1.5 são apresentadas as ameaças à validade desse estudo. Na seção 3.1.6 são apresentadas as considerações do mapeamento.

3.1.2 Trabalhos Relacionados

Com o objetivo de buscar e identificar estudos primários e selecionar estudos relevantes sobre a aplicação de DM e ML na classificação de dados em estudos sobre evasão escolar, de forma a caracterizar a evolução dessa área de pesquisa, foram detectados revisões e mapeamentos da literatura, mas que não foram classificados como estudos relevantes baseados nas questões de pesquisa e os critérios de inclusão e exclusão que são apresentados nas seções seguintes. No entanto, alguns desses trabalhos se relacionam com a proposta deste estudo, sendo estes apresentados abaixo.

No trabalho realizado por Colpo et al. (2020), é apresentada uma revisão sistemática de estudos que utilizam técnicas de EDM no contexto da previsão de evasão estudantil e que foram publicados no Congresso Brasileiro de Informática na Educação (CBIE), principal evento da área no Brasil. O trabalho ressalta o enfoque em regiões como Sul, Nordeste e Sudeste, notadamente nos estados do Rio Grande do Sul e Rio de Janeiro. As pesquisas variaram em quantidade e tipos de dados. Árvores de decisão e Weka foram amplamente empregadas. O interesse crescente em EDM, especialmente na educação básica e técnica, destaca a necessidade de maior aplicação prática e disponibilização eficaz das previsões.

O segundo trabalho relacionado, realizado por Tamada, Netto e Lima (2019), tem como objetivo principal identificar soluções que utilizem técnicas de ML para reduzir as altas taxas de desistência em Ambientes Virtuais de Aprendizagem (AVA). A pesquisa identificou que as soluções dos estudos utilizam principalmente ML supervisionado para previsão, mais

especificamente os algoritmos de Regressão Logística e Máquina de Vetores de Suporte, com alta precisão em dados de cliques e fóruns. Três sistemas com *Graphical User Interface* (GUI) gerenciam aprendizado. Poucos usam ML não supervisionado para agrupar comportamentos. A precisão dos classificadores varia conforme o ambiente dinâmico de aprendizagem.

Posto isso, o presente estudo diferencia-se dos trabalhos relacionados no escopo da busca, apesar do primeiro trabalho realizar uma busca em todos os níveis de ensino, ele se limita aos trabalhos publicados no CBIE; o segundo trabalho relacionado restringe-se a modalidade de ensino que é Educação à Distância (EAD) e, mais especificamente, nos AVA's. Com isso, este trabalho apresenta um escopo maior de estudos com esse contexto da evasão estudantil.

3.1.3 Condução do Mapeamento

Na realização desse mapeamento foi aplicado o método de mapeamento sistemático da literatura (do inglês, *Systematic Literature Mapping* - SLM) (PETERSEN et al., 2008). O método consiste nas etapas definidas abaixo.

1. "Questões de Pesquisa" são questões que quando respondidas nos ajudam a alcançar os objetivos do mapeamento sistemático.
2. "Identificação de Estudos" que são as palavras-chave, a *string* de busca, os critérios de seleção das fontes de busca, lista das fontes de busca e estratégia de busca.
3. "Seleção e Avaliação dos Estudos" são os critérios de inclusão e de exclusão dos estudos primários.
4. "Síntese dos Dados e Apresentação dos Resultados" é apresentada na seção 3.1.4 (Resultados e Discussões). É nessa seção que estão as informações sobre a estratégia de extração de dados, estratégia de sumarização dos dados e as informações sobre os dados extraídos e sumarizados.

3.1.3.1 Questões de Pesquisa

Foram definidas questões que se mostraram importantes para melhor extração dos dados obtidos, essas questões foram resultado de um processo meticuloso. Isso incluiu revisão ampla da literatura, análise de tendências recentes e exploração das demandas práticas no campo da "Aplicação de DM e ML na classificação de dados em estudos sobre evasão escolar". A seleção considerou a relevância para os objetivos do mapeamento e a capacidade de orientar a coleta precisa e abrangente de dados e dessa forma alcançar os objetivos do mapeamento sistemático. Abaixo, na Tabela 1, estão definidas as questões de pesquisa junto aos seus respectivos dados a serem obtidos:

Tabela 1 – Questões de pesquisa

Questões de pesquisa	**Dados a serem obtidos**
Como pode ser caracterizada a evolução do estado da arte sobre "Aplicação de DM e ML na classificação de dados em estudos sobre evasão escolar" em termos de publicações, autores e grupos de pesquisa?	Para extrair os dados serão analisados os anos das publicações dos artigos, os autores, os países e grupos de pesquisa.
Como contribuir para o amadurecimento dessa área de pesquisa preenchendo as lacunas encontradas?	Para extrair os dados serão analisados os métodos utilizados nas predições, quais fatores influenciam na evasão, informações sobre as bases de dados, metodologias utilizadas na mineração de dados, aprendizagem de máquina e seleção de atributos, e resultados dos experimentos feitos nos artigos.

3.1.3.2 Identificação de Estudos

Para formar as *strings* de busca foram escolhidos assuntos pelo reconhecimento científico e associação com a área de tecnologia da informação e educação, frequentemente apresentadas no contexto desta pesquisa. Após análise e experimentos com palavras-chave que envolvem o tema de pesquisa, as seguintes palavras-chave foram escolhidas: *"PREDICTION"*, *"CLASSIFICATION"*, *"STUDENT DROPOUT"*, *"MACHINE LEARNING"*, *"DATA MINING"* e *"DATA SCIENCE"*. Assim, a *string* de busca que é o conjunto de termos utilizadas para a busca dos estudos, ficou da seguinte forma: *((PREDICTION OR CLASSIFICATION) AND "STUDENT DROPOUT"AND ("MACHINE LEARNING"OR "DATA MINING"OR "DATA SCIENCE"))*, em termos de quantidade e escopo essa *string* de busca abrange um escopo grande, trazendo estudos sobre a aplicação de DM e ML na classificação de dados em estudos sobre evasão escolar em diferentes níveis escolar e categorias de ensino distintas.

Para a realização da busca foram selecionadas as seguintes bases bibliográficas: *ACM Digital Library* (ACM)[1], *IEEE Xplore* (IEEE) [2], *Scopus* [3], *Web of Science* [4] e *ScienceDirect* [5]. As pesquisas dos termos de busca foram realizadas em 30 de outubro de 2022.

3.1.3.3 Seleção e Avaliação dos Estudos

Com o propósito de analisar quais artigos poderão ser utilizados para responder às questões de pesquisa, foram definidos critérios de inclusão e exclusão. Os critérios de inclusão

[1] http://dl.acm.org
[2] http://ieeexplore.ieee.org
[3] https://www.scopus.com
[4] https://www.webofscience.com/
[5] https://www.sciencedirect.com

definidos foram:

1. Estudos que foquem em predição de evasão do aluno por meio da mineração de dados e aprendizagem de máquina.
2. Estudos que apresentem inovação na mineração dos atributos utilizados na predição de evasão do aluno.
3. Estudos que apresentem comparativos entre algoritmos de classificação trabalhando com predição de evasão do aluno.

Foram também selecionados os seguintes critérios de exclusão dos artigos, com a finalidade de retirar estudos não relevantes para o mapeamento:

1. Estudos duplicados.
2. Estudos não disponibilizados na íntegra.

Após a aplicação desses critérios, 71 artigos se mantiveram na pesquisa.

3.1.4 Resultados e Discussões

Nessa seção são apresentados os resultados iniciais da busca, denominados Estudos Primários (EP), Tabela 2; os estudos resultantes após aplicação dos critérios de inclusão e exclusão, denominados Estudos Relevantes (ER), Tabela 3; e os resultados da análise dos estudos relevantes, respondendo às questões de pesquisa definidas na seção 3.1.3.1 deste mapeamento sistemático.

3.1.4.1 Resultados encontrados com a string de busca

A Tabela 2 detalha a distribuição dos estudos em relação às diferentes bases de dados consultadas para o mapeamento sistemático. No total, foram identificados 336 EP nas diferentes bases de dados. Após a aplicação rigorosa dos critérios de inclusão e exclusão estabelecidos, 265 estudos foram rejeitados, enquanto 71 estudos satisfizeram esses critérios e foram classificados como ER. Esse processo de seleção rigorosa garantiu que apenas os estudos mais pertinentes e alinhados aos objetivos deste mapeamento sistemático fossem considerados para análise.

Os números apresentados na tabela revelam variações interessantes entre as diferentes bases de dados. Nota-se que a base Scopus apresentou o maior número de EP (126), porém, após a aplicação dos critérios de seleção, 98 estudos foram rejeitados, resultando em 28 ER. Por outro lado, a base IEEE teve 37 Estudos Primários, dos quais 31 foram rejeitados, restando 6 ER.

Esses resultados preliminares ressaltam a importância da seleção criteriosa de estudos para garantir a qualidade e a relevância das informações analisadas. Os ER selecionados nesta

etapa representam uma base sólida para a análise aprofundada que visa responder às questões de pesquisa delineadas na seção 3.1.3.1 deste mapeamento sistemático.

Tabela 2 – Resultados da Seleção dos ER

Bases	**EP**	**Rejeitados**	**Aceitos**
ACM	50	41	9
IEEE	37	31	6
ScienceDirect	71	64	7
Scopus	126	98	28
Web of Science	52	31	21
Total	**336**	**265**	**71**

3.1.4.2 Estudos resultantes após aplicação de critérios de inclusão e exclusão

A Tabela 3 apresenta a lista completa dos ER após a aplicação dos critérios de inclusão e exclusão aos EP. Cada ER é identificado por um número de referência único (ER1, ER2, ER3, etc.), associado à sua respectiva fonte bibliográfica. A lista de ER abrange uma variedade de fontes, incluindo artigos de conferências, revistas acadêmicas e outros recursos importantes. As referências incluídas nessa tabela fornecem um panorama abrangente dos trabalhos selecionados para análise posterior. Esses estudos representam contribuições significativas para a investigação abordada neste mapeamento sistemático.

Ao analisar a lista, é possível observar a diversidade de fontes e autores envolvidos na área de pesquisa examinada. Os ER foram selecionados com base em sua relevância para os objetivos deste estudo e servirão como base para a análise detalhada das questões de pesquisa propostas. É importante ressaltar que a escolha desses estudos foi guiada pela necessidade de garantir a representatividade e a confiabilidade das informações utilizadas nesta pesquisa. A análise subsequente dos ER permitirá a extração de *insights*, identificação de tendências e a obtenção de respostas substanciais para as questões delineadas.

Essa lista completa de ER é um componente crucial do processo de mapeamento sistemático, fornecendo uma base sólida para a investigação e discussão dos resultados obtidos a partir desses estudos selecionados.

Tabela 3 – Lista de Estudos Relevantes

Id.	Referência
ER1	(KANG; WANG, 2018)
ER2	(WU et al., 2019)
ER3	(MECA et al., 2020)
ER4	(BARANYI; NAGY; MOLONTAY, 2020)
ER5	(CHEN; JOHRI; RANGWALA, 2018)
ER6	(AMERI et al., 2016)
ER7	(MANHÃES; CRUZ; ZIMBRÃO, 2014a)
ER8	(LOTTERING; HANS; LALL, 2020)
ER9	(HEGDE, 2016)
ER10	(NAGY; MOLONTAY, 2018)
ER11	(KUO; PAN; LEI, 2017)
ER12	(FU et al., 2021)
ER13	(MUBARAK; CAO; HEZAM, 2021)
ER14	(CHUNG; LEE, 2019)
ER15	(LYKOURENTZOU et al., 2009)
ER16	(VILORIA; LEZAMA, 2019)
ER17	(SANTANA et al., 2015)
ER18	(SELVAN; NAVADURGA; PRASANNA, 2019)
ER19	(NUANKAEW et al., 2019)
ER20	(ROVIRA; PUERTAS; IGUAL, 2017)
ER21	(RADOVANOVIĆ; DELIBAŠIĆ; SUKNOVIĆ, 2020)
ER22	(DEWAN et al., 2015)
ER23	(MAKSIMOVA; PENTEL; DUNAJEVA, 2020)
ER24	(YAACOB et al., 2020)
ER25	(MOSELEY; MEAD, 2008)
ER26	(COSTA et al., 2020a)
ER27	(QUISHPE-MORALES et al., 2020)
ER28	(MARTINS et al., 2020)
ER29	(SALLAN; BEHAL,)
ER30	(MUBARAK; CAO; ZHANG, 2020)
ER31	(ALBAN; MAURICIO, 2018a)
ER32	(GAMAO; GERARDO, 2019)
ER33	(AGUIRRE; PÉREZ, 2020)
ER34	(SIVAKUMAR; VENKATARAMAN; SELVARAJ, 2016)
ER35	(HUTAGAOL; SUHARJITO, 2019)
ER36	(BONIFRO et al., 2020)
ER37	(TENPIPAT; AKKARAJITSAKUL, 2020)

ER38	(PRADEEP; DAS; KIZHEKKETHOTTAM, 2015)
ER39	(NASEEM et al., 2019)
ER40	(BELLO et al., 2020)
ER41	(SORENSEN, 2019)
ER42	(ŞAHIN, 2021)
ER43	(LOTTERING; HANS; LALL,)
ER44	(PEREIRA; ZAMBRANO, 2017)
ER45	(PEREZ; CASTELLANOS; CORREAL, 2018)
ER46	(PÉREZ et al., 2018)
ER47	(PÉREZ-GUTIÉRREZ, 2020)
ER48	(BURGOS et al., 2018)
ER49	(XING; DU, 2019)
ER50	(LIMSATHITWONG; TIWATTHANONT; YATSUNGNOEN, 2018)
ER51	(BERENS et al., 2018)
ER52	(PANAGIOTAKOPOULOS et al., 2021)
ER53	(FIGUEROA-CAÑAS; S.VINUESA, 2020)
ER54	(HEGDE; PRAGEETH, 2018)
ER55	(PEÑA et al., 2017)
ER56	(COUSSEMENT et al., 2020)
ER57	(TAN; SHAO, 2015)
ER58	(KOTSIANTIS; PIERRAKEAS; PINTELAS, 2003)
ER59	(HEREDIA; AMAYA; BARRIENTOS, 2015)
ER60	(FEI; YEUNG, 2015)
ER61	(LI; CUI; ZHANG, 2022)
ER62	(SU et al., 2022)
ER63	(REVATHY; KAMALAKKANNAN; KAVITHA, 2022)
ER64	(HOSSAIN et al., 2022)
ER65	(NIYOGISUBIZO et al., 2022)
ER66	(GUZMÁN-CASTILLO et al., 2022)
ER67	(NUANMEESRI et al., 2022)
ER68	(MNYAWAMI; MAZIKU; MUSHI, 2022)
ER69	(JARBOU et al., 2022)
ER70	(VEGA et al., 2022)
ER71	(NASEEM; CHAUDHARY; SHARMA, 2022)

3.1.4.3 Como pode ser caracterizada a evolução do estado da arte sobre aplicação de DM e ML na classificação de dados em estudos sobre evasão escolar?

Essa questão tem como finalidade identificar como se deu a evolução da aplicação de DM e ML na classificação de dados em estudos sobre evasão escolar do aluno, e quais as perspectivas de futuro para a área. A Figura 5 apresenta a evolução da quantidade de publicações ao longo dos anos. Não foi estabelecido um critério de busca baseado no ano de publicação, o que resultou em um intervalo de tempo abrangendo as publicações de 2003 a 2022. É notado um crescimento médio de estudos relacionados à área na última década, com destaque para o ano de 2020.

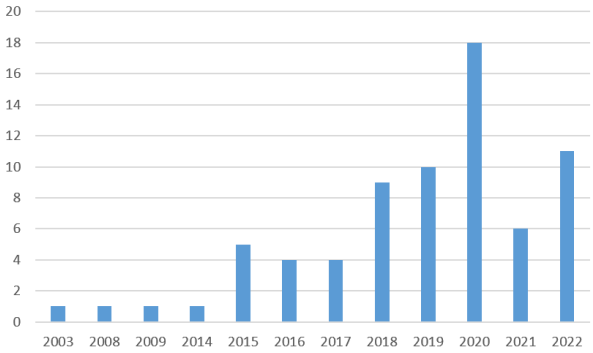

Figura 5 – Anos das Publicações dos Artigos

O gráfico representado pela Figura 6 visa mostrar quais países se destacam na quantidade de artigos publicados sobre a aplicação de DM e ML na classificação de dados em estudos sobre evasão escolar. O país destaque em publicações foi a China, seguida por Estados Unidos e Espanha. Foram encontrados também três estudos brasileiros. Os estados desses estudos foram: Rio de Janeiro (ER7), Rio Grande do Sul (ER26) e Alagoas (ER17), cada um deles com uma publicação.

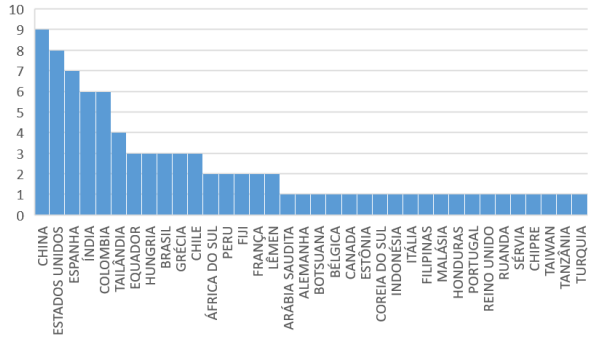

Figura 6 – Países dos artigos

3.1.4.4 Como contribuir para o amadurecimento dessa área de pesquisa preenchendo as lacunas encontradas?

Essa questão busca responder quais os métodos utilizados nas predições, quais fatores influenciam na evasão, informações sobre as bases de dados, metodologias utilizadas na mineração de dados e seleção de atributos, e resultados dos experimentos feitos nos artigos. A Tabela 4 apresenta uma síntese geral dos ER com relação as metodologias de DM, técnicas de seleção de atributos e redução de dimensionalidade, e algoritmos utilizados.

Tabela 4 – Tabela de Síntese Geral dos Dados Extraídos

Id.	**Assunto**	**Estudos**
A1	Metodologia CRISP-DM.	ER3, ER24, ER26, ER39, ER45, ER50 e ER71.
A2	Metodologia KDD-DM.	ER27, ER29, ER33, ER43 e ER59.
A3	Redução de Dimensionalidade.	ER8, ER9, ER43 e ER54.
A4	Seleção de Atributos com técnica especificada.	ER3, ER10, ER28, ER31, ER32, ER33, ER35, ER39, ER50, ER58, ER66, ER67, ER70 e ER71.
A5	Seleção de Atributos com técnica não especificada.	ER2, ER17, ER26, ER30, ER34, ER38, ER40, ER46, ER52, ER53, ER54, ER59, ER63, ER65, ER68 e ER69.
A6	Estudos que utilizam pelo menos uma dessas técnicas/algoritmos: Deep Learning, Adaboost, K Nearest Neighbor, Naive Bayes, Support Vector Machine, Regressão Logística, Redes Neurais, Floresta Aleatória e Árvore de Decisão.	ER1, ER2, ER3, ER4, ER5, ER6, ER7, ER8, ER10, ER11, ER12, ER13, ER14, ER15, ER17, ER18, ER19, ER20, ER21, ER22, ER23, ER24, ER26, ER27, ER28, ER29, ER30, ER31, ER32, ER34, ER35, ER36, ER37, ER38, ER39, ER40, ER41, ER42, ER43, ER44, ER45, ER46, ER47, ER48, ER49, ER50, ER52, ER53, ER54, ER55, ER56, ER57, ER58, ER59, ER60, ER63, ER64, ER65, ER66, ER67, ER68, ER69, ER70 e ER71.
A7	Estudos que utilizam nenhuma das técnicas citadas no A6.	ER9, ER16, ER25, ER33, ER51, ER61 e ER62.

3.1.4.4.1 Metodologia de Mineração de Dados

Apenas 12 estudos especificaram a metodologia de DM utilizada em seus experimentos, sendo 7 deles utilizando CRISP (A1) e 5 KDD (A2).

3.1.4.4.2 Seleção de atributos e/ou redução de dimensionalidade

Dos 71 artigos, 34 explicitaram a aplicação de seleção de atributos ou redução de dimensionalidade em seus experimentos, enquanto 37 não o fizeram. Desses 34, 4 aplicaram

redução de dimensionalidade (A3), enquanto os outros 30 utilizaram seleção de atributos (A4 e A5). As técnicas utilizadas para a seleção de atributos estão apresentadas na Figura 7.

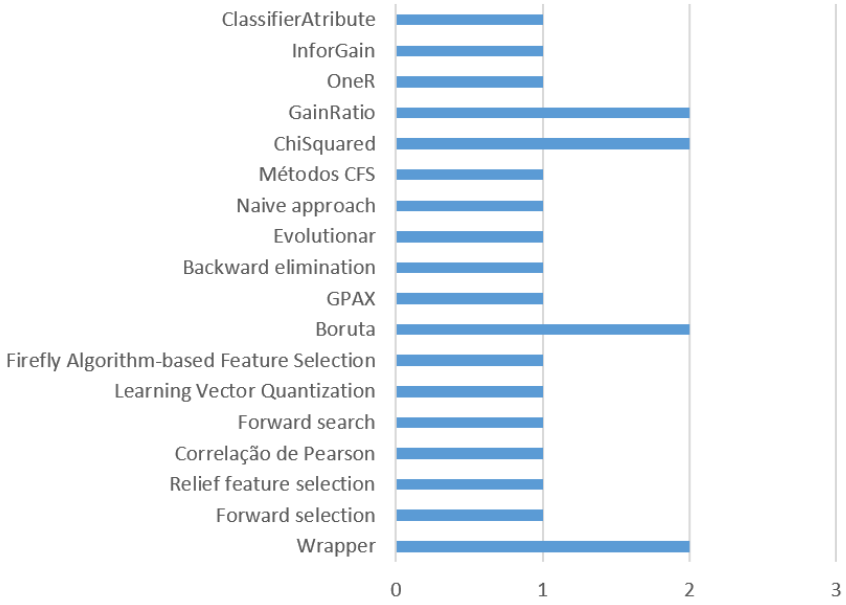

Figura 7 – Técnicas de Seleção de Atributos

3.1.4.4.3 Bases de dados utilizadas nos experimentos

As bases de dados descritas nos artigos são referentes aos anos de 1994 a 2021. A menor amostra utilizada nos experimentos foi de 26 registros, enquanto a maior foi de 220 mil. Os gráficos a seguir destacam a proporção dos níveis de ensino e das modalidades trabalhadas nos experimentos dos estudos relevantes, respectivamente. Percebe-se uma predominância de estudos voltados ao ensino superior e de estudos voltados a modalidade presencial.

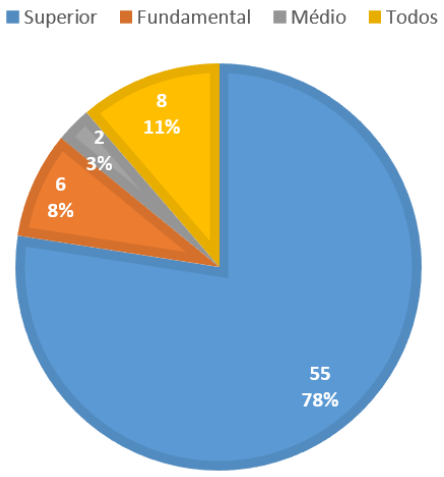

Figura 8 – Proporção dos Níveis de Escolaridade

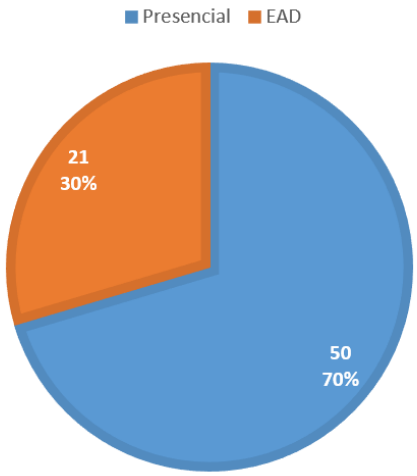

Figura 9 – Proporção das Modalidades de Ensino

Por fim, a Figura 10 apresenta a frequência de utilização de métodos de aprendizagem de máquina nos experimentos dos ER (A6), a categoria "Outros"presente no gráfico aglomera algoritmos e modelos que foram utilizados em 2 estudos, ou menos, como é o caso dos estudos A7.

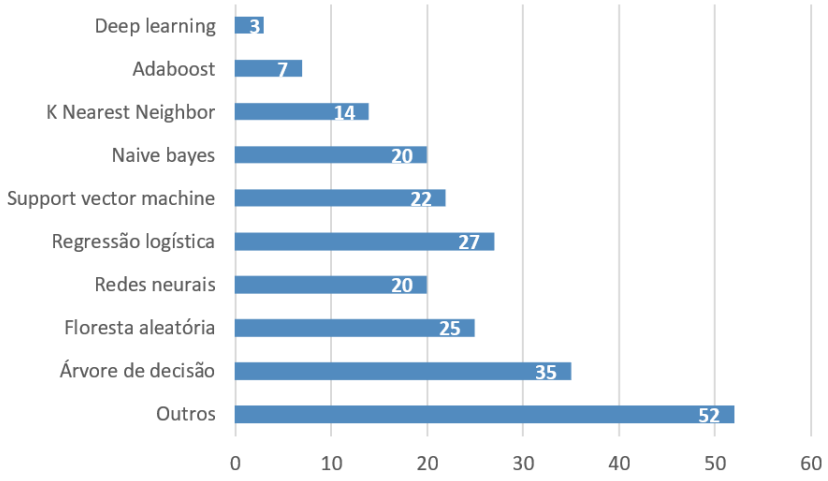

Figura 10 – Frequência dos Algoritmos

3.1.4.5 Síntese Geral e Sumarização dos Trabalhos Selecionados

Nesta etapa serão apresentados breves resumos dos trabalhos selecionados e uma síntese geral das publicações, os resumos estão organizados de acordo com os assuntos já apresentadas na seção 3.1.4.4. Destacando os pontos importantes de cada estudo, em termos de algoritmos, sistemas, em qual nível escolar a classificação foi aplicada, em qual modalidade de ensino, conjunto de dados utilizado, etc.

3.1.4.5.1 Metodologia CRISP-DM

ER3: Essa pesquisa aborda a categorização de desistência de estudantes universitários, dando foco na seleção dos atributos que melhoram essa predição de acordo com cada área acadêmica. O conjunto é formado por 24.894 participantes do ensino superior presencial, foi aplicada a abordagem Wrapper com o algoritmo RPart para seleção de atributos. As variáveis destacadas são adaptadas com base no curso específico. (MECA et al., 2020).

ER24: Esse trabalho trata da classificação de abandono do aluno universitário, especificamente no curso de ciência da computação, os algoritmos utilizados foram árvore de decisão, regressão logística, floresta aleatória, K-vizinho mais próximo e algoritmo de rede neural. O conjunto do estudo é composto por 64 participantes do ensino superior presencial, a variável mais importante é o desempenho em disciplinas específicas. (YAACOB et al., 2020).

ER26: Essa pesquisa aborda a categorização de desistência do estudante universitário, sendo empregados 3 algoritmos, destacando-se a Floresta aleatória, com uma taxa de acurácia de 95,12% e Recall de 91,41%. O estudo englobou 1.516 participantes em nível superior, com ensino presencial. Houve emprego de seleção de atributos, utilizando Regressão Logística, Árvores de Decisão e Floresta Aleatória como algoritmos de classificação. As variáveis destacadas foram a Média do terceiro semestre e a Média dos três primeiros semestres. (COSTA et al., 2020a).

ER39: Esse trabalho aborda a classificação de abandono do aluno universitário, especificamente no curso de Ciências da computação. O algoritmo utilizado é Árvore de decisão, aplica-se também seleção de recurso para identificar quais os melhores atributos para a classificação. A pesquisa incluiu 963 participantes no nível superior, com aulas presenciais. A seleção de atributos foi realizada por meio do algoritmo Boruta. As variáveis em destaque abrangeram GRADE, A1SCORE e QATTEMPT. (NASEEM et al., 2019).

ER45: Essa pesquisa tem como foco a classificação de abandono do aluno universitário, especificamente no curso de engenharia de sistemas. Os algoritmos utilizados nos experimentos foram: Árvore de decisão, Regressão logística e Naive Bayes. Tendo Árvore de decisão como algoritmo que obteve melhor resultado, AUC de 0.94. O estudo envolveu 802 indivíduos no nível superior com aulas presenciais. Não foi realizada seleção de atributos. Não houve destaque específico de variáveis. (PEREZ; CASTELLANOS; CORREAL, 2018).

ER50: Esse trabalho aborda a classificação de abandono do alunos do ensino médio de um instituto de tecnologia da Tailândia, os algoritmos utilizados foram Árvores de decisão e Floresta aleatória, Árvores de decisão obteve os melhores resultados. A pesquisa abrangeu 28.801 participantes em nível superior com ensino presencial. Utilizou-se o critério GPAX para seleção de atributos, as notas nos cursos de idiomas foram identificadas como variáveis de destaque. (LIMSATHITWONG; TIWATTHANONT; YATSUNGNOEN, 2018).

ER71: Nesse trabalho é feita uma análise da evasão no curso de Ciência da Computação, os algoritmos utilizados foram *Random Forest*, *Decision Tree*, *Naive Bayes*, *Logistic Regression* e

K-Nearest Neighbour. No primeiro estágio o algoritmo que obteve o melhor resultado foi o Naive Bayes com uma AUC de 0.6123, nos estágios 2 e 3 o algoritmo que obteve o melhor resultado foi o Logistic Regression com AUC de 0.7523 e 0.8902, respectivamente. O tamanho do conjunto de dados não foi mencionado no artigo, mas trata-se de dados de uma única Instituição de Ensino Superior da região do Pacífico Sul, que é o único campus regional no Pacífico Sul. (NASEEM; CHAUDHARY; SHARMA, 2022).

3.1.4.5.2 Metodologia KDD-DM

ER27: Essa pesquisa aborda a categorização de desistência do estudante universitário, sendo empregados os algoritmos KNN, árvore de decisão, floresta aleatória, SVM e redes neurais. O modelo de redes neurais demonstrou o melhor desempenho, alcançando uma acurácia de 92%. O estudo contou com 26 participantes no nível superior, com aulas presenciais. Não houve seleção de atributos. Não houve destaque específico de variáveis. (QUISHPE-MORALES et al., 2020).

ER29: Nesse estudo, aborda-se a categorização da evasão de estudantes universitários. Como parte da pesquisa, são empregados os algoritmos *Decision Stump*, NDTREE e *Enhanced Machine Learning* (EMLA), os quais são fundamentados em técnicas de aprendizado de máquina. O estudo envolveu 407 participantes no nível superior, com ensino presencial. Não foi realizada seleção de atributos. Não houve destaque específico de variáveis. (SALLAN; BEHAL,).

ER33: Nesse trabalho, trata-se da classificação de abandono dos alunos universitários. O modelo aplicado é a regressão linear e o estudo também enfoca a seleção de recursos. O estudo incluiu 530 indivíduos no ensino superior, com aulas presenciais. A seleção de atributos foi conduzida por meio da Correlação de Pearson. Não houve destaque específico de variáveis. (AGUIRRE; PÉREZ, 2020).

ER43: Nesse estudo, explora-se a categorização do abandono de alunos universitários, utilizando diferentes algoritmos, como Floresta aleatória, máquinas de vetores de suporte, árvores de decisão, Naive Bayes, K-vizinho mais próximo e regressão logística. Destaca-se que o algoritmo de Floresta aleatória apresentou os melhores resultados, com uma acurácia de 94.14%. O estudo abordou o ensino superior em formato presencial, envolvendo 12.293 participantes. O artigo não menciona a aplicação de uma abordagem específica para seleção de atributos. As variáveis importantes englobaram a idade, as disciplinas e o curso dos participantes. (LOTTERING; HANS; LALL,).

ER59: Nesse trabalho, é examinada a classificação do abandono de estudantes universitários, os algoritmos utilizados foram árvores de decisão C4.5 e ID3. O estudo abrangeu 201 participantes no ensino superior com formato presencial. Foi aplicada seleção de atributos, porém detalhes específicos da seleção de atributos não foram especificados. (HEREDIA; AMAYA; BARRIENTOS, 2015).

3.1.4.5.3 Redução de Dimensionalidade

ER8: Esse trabalho trata da classificação do abandono de alunos universitários, explorando a adequação da diminuição dimensional e concentrando os dados significativos encobertos nas informações de alunos com risco de abandono. Embora todos os algoritmos de aprendizado de máquina utilizados nesse trabalho tenham alcançado uma acurácia de mais de 75%, *Support Vetor Machine* e *Naive Bayes* tiveram melhor desempenho ao prever o abandono. O estudo envolveu 4.417 participantes no ensino superior presencial. Para análise, foram utilizados os métodos SVM, Naïve Bayes, Árvore de Decisão, KNN e Floresta Aleatória. Não houve destaque específico de variáveis. (LOTTERING; HANS; LALL, 2020).

ER9: Esse trabalho foca na redução de dimensionalidade dos atributos para melhorar a classificação de abandono de alunos universitários. O estudo envolveu 150 participantes no ensino superior com aulas presenciais. Foi realizada redução dimensional utilizando a técnica PCA (Análise de Componentes Principais). Não houve destaque específico de variáveis. (HEGDE, 2016).

ER43: Nesse estudo, explora-se a categorização do abandono de alunos universitários, utilizando diferentes algoritmos, como Floresta aleatória, máquinas de vetores de suporte, árvores de decisão, Naive Bayes, K-vizinho mais próximo e regressão logística. Destaca-se que o algoritmo de Floresta aleatória apresentou os melhores resultados, com uma acurácia de 94.14%. O estudo abordou o ensino superior em formato presencial, envolvendo 12.293 participantes. O artigo não menciona a aplicação de uma abordagem específica para seleção de atributos. As variáveis importantes englobaram a idade, as disciplinas e o curso dos participantes. (LOTTERING; HANS; LALL,).

ER54: Nesse estudo, aborda-se a classificação de abandono de alunos, tanto em escolas quanto em universidades. O algoritmo utilizado é o *Naive Bayes*, com a linguagem R. O trabalho também foca na seleção de recurso para escolha dos melhores atributos. O estudo envolveu 50 participantes no ensino superior em formato presencial. As variáveis relevantes incluíram fatores acadêmicos, demográficos, psicológicos e de saúde. (HEGDE; PRAGEETH, 2018).

3.1.4.5.4 Seleção de Atributos

Técnica de Seleção de Atributos especificada

ER3: Essa pesquisa aborda a categorização de desistência de estudantes universitários, dando foco na seleção dos atributos que melhoram essa predição de acordo com cada área acadêmica. O conjunto é formado por 24.894 participantes do ensino superior presencial, foi aplicada a abordagem Wrapper com o algoritmo RPart para seleção de atributos. As variáveis destacadas são adaptadas com base no curso específico. (MECA et al., 2020).

ER10: Esse trabalho trata da classificação do abandono de alunos em programas universitários, foram utilizados algoritmos baseados em árvore de decisão, *Naive Bayes*, K-NN, modelos lineares

e aprendizado profundo. Os modelos com melhores resultados foram: *Gradient Boosted Trees* e *Deep Learning*, com uma AUC de 0.808 e 0.811, respectivamente. O estudo abrangeu 15.825 participantes do ensino superior em formato presencial. Não houve destaque específico de variáveis. (NAGY; MOLONTAY, 2018).

ER28: Nesse estudo, é explorada a classificação de abandono do aluno universitário, ele utiliza uma combinação de 3 algoritmos de mineração de dados populares, floresta aleatória, máquinas de vetores de suporte e redes neurais artificiais. O estudo considerou o ensino superior presencial, abrangendo 3.373/3.344 participantes. Foi aplicada a técnica de seleção de atributos forward search. Para o 1º semestre, variáveis destacadas incluíram ects_aprov_s, cod_escola, média_s, sexo, bolseiro_s, entre outras; para o 2º semestre, variáveis como ects_reprov_s, cod_prof_mae, n10_11_acesso, idade, nacionalidade, entre outras, foram relevantes. (MARTINS et al., 2020).

ER31: Nessa pesquisa, explora-se a classificação de abandono do aluno universitário, os algoritmos utilizados foram regressão logística e árvores de decisão. Como resultado, obteve-se que a técnica com maior percentual de acurácia de abandono foi a árvore de decisão com 91.70%. O estudo englobou 1.178 participantes no ensino superior em formato presencial. Houve seleção de atributos utilizando a técnica Relief Feature Selection. Não houve destaque específico de variáveis. (ALBAN; MAURICIO, 2018a).

ER32: Esse trabalho analisa modelos de classificação de abandono do aluno universitário, foram utilizados os algoritmos *Decision Tree* e *Naive Bayes*, o *Decision Tree* obteve melhores resultados. A pesquisa abrangeu 1.862 participantes no ensino superior com formato presencial. Foi aplicada a seleção de atributos baseada no algoritmo Firefly. Não houve destaque específico de variáveis. (GAMAO; GERARDO, 2019).

ER33: Nesse trabalho, trata-se da classificação de abandono dos alunos universitários. O modelo aplicado é a regressão linear e o estudo também enfoca a seleção de recursos. O estudo incluiu 530 indivíduos no ensino superior, com aulas presenciais. A seleção de atributos foi conduzida por meio da Correlação de Pearson. Não houve destaque específico de variáveis. (AGUIRRE; PÉREZ, 2020).

ER35: Essa pesquisa trata da classificação de abandono do aluno universitário, os algoritmos utilizados foram *K-Nearest Neighbor* (KNN), *Naïve Bayes* (NB) e *Decision Tree* (DT). A pesquisa abordou 17.432 participantes no ensino superior presencial. Utilizou a técnica Learning Vector Quantization para seleção de atributos. As variáveis de destaque incluíram frequência, pontuações de tarefas, créditos totais, pontuações UTS, UAS pontuações, GPA, renda dos pais, educação dos pais, sexo e idade dos estudantes. (HUTAGAOL; SUHARJITO, 2019).

ER39: Esse trabalho aborda a classificação de abandono do aluno universitário, especificamente no curso de Ciências da computação. O algoritmo utilizado é Árvore de decisão, aplica-se também seleção de recurso para identificar quais os melhores atributos para a classificação. A pesquisa incluiu 963 participantes no nível superior, com aulas presenciais. A seleção de atributos

foi realizada por meio do algoritmo Boruta. As variáveis em destaque abrangeram GRADE, A1SCORE e QATTEMPT. (NASEEM et al., 2019).

ER50: Esse trabalho aborda a classificação de abandono do alunos do ensino médio de um instituto de tecnologia da Tailândia, os algoritmos utilizados foram Árvores de decisão e Floresta aleatória, Árvores de decisão obteve os melhores resultados. A pesquisa abrangeu 28.801 participantes em nível superior com ensino presencial. Utilizou-se o critério GPAX para seleção de atributos, as notas nos cursos de idiomas foram identificadas como variáveis de destaque. (LIMSATHITWONG; TIWATTHANONT; YATSUNGNOEN, 2018).

ER58: Nesse estudo, aborda-se a classificação de abandono do aluno universitário, os métodos de classificação utilizados incluíram *Naive Bayes*, C4.5, *BackPropagation*, SMO, 3NN e MLE, é implementando também um protótipo de sistema web utilizando o *Naive Bayes*. A pesquisa envolveu 354 participantes no ensino superior na modalidade EaD. A seleção de atributos foi realizada por meio da abordagem Wrapper. Não houve destaque específico de variáveis. (KOTSIANTIS; PIERRAKEAS; PINTELAS, 2003).

ER66: Nesse trabalho foi desenvolvido um sistema que permite o cálculo do risco de evasão por aluno e utiliza um procedimento de geração de alertas para coordenar as intervenções. A plataforma permitiu mensurar o impacto das estratégias de intervenção na permanência dos alunos. O estudo abrangeu 15.805 participantes no ensino superior em formato presencial. Foi empregada a abordagem ingênua (naive approach) para seleção de atributos. As técnicas de classificação utilizadas incluíram o algoritmo *AdaBoost*, *Bayesian GLM*, *Decision Trees*, *LogitBoost*, *Random Forest* (RF) e *Stochastic Gradient Boosting*. Não houve destaque específico de variáveis. (GUZMÁN-CASTILLO et al., 2022).

ER67: Esse estudo criou um modelo combinando seleção de atributos com um método de rede neural perceptron multicamadas. O modelo foi comparado com modelos baseados nos algoritmos de *Logistic regression*, *Decision Tree*, *Random Forest*, *Naive Bayes*, *Support Vector Machine* e *Multilayer Perceptron Neural Network*, e obteve melhores resultados. A pesquisa considerou 1.650 participantes no ensino superior presencial. Foram aplicadas as técnicas de seleção de atributos GR, CS e Métodos CFS. As variáveis destacadas englobaram a média cumulativa de notas (GPA), média cumulativa de notas para assuntos não docentes (GPAnone) e adesão ao grupo de mídia social em assuntos (*Socialclass*). (NUANMEESRI et al., 2022).

ER70: O objetivo nesse trabalho é reduzir a taxa de evasão dos alunos da Faculdade de Engenharia de Sistemas e Informática da Universidade Nacional Mayor de San Marcos (FISI-UNMSM), o modelo de predição utilizado é baseado em árvores de decisão, obteve uma acurácia de 90.34%. A pesquisa abrangeu 1.938 participantes no ensino superior presencial. Foram aplicadas as técnicas de seleção de atributos ChiSquared, OneR, GainRatio, InforGain e ClassifierAttribute. As variáveis destacadas incluíram a média ponderada histórica de notas, a média ponderada de notas do último ciclo e o número de créditos de cursos aprovados. (VEGA et al., 2022).

ER71: Nesse trabalho é feita uma análise da evasão no curso de Ciência da Computação, os algoritmos utilizados foram *Random Forest*, *Decision Tree*, *Naive Bayes*, *Logistic Regression* e *K-Nearest Neighbour*. No primeiro estágio o algoritmo que obteve o melhor resultado foi o Naive Bayes com uma AUC de 0.6123, nos estágios 2 e 3 o algoritmo que obteve o melhor resultado foi o Logistic Regression com AUC de 0.7523 e 0.8902, respectivamente. O tamanho do conjunto de dados não foi mencionado no artigo, mas trata-se de dados de uma única Instituição de Ensino Superior da região do Pacífico Sul, que é o único campus regional no Pacífico Sul. (NASEEM; CHAUDHARY; SHARMA, 2022).

Técnica de Seleção de Atributos não especificada

ER2: Esse trabalho propõe um modelo de rede neural profunda, que é uma combinação de *Convolutional Neural Network, Long Short-Term Memory network e Support Vector Machine*, para ter um melhor desempenho na predição do abandono do aluno nos cursos online abertos massivos (MOOCs). O estudo englobou 120.542 participantes no ensino superior na modalidade EaD. Foi aplicada seleção de atributos, mas os detalhes específicos não foram especificados. As variáveis relevantes foram ID de inscrição, hora, fonte do evento, evento e dispositivo de acesso. (WU et al., 2019).

ER17: Esse trabalho trata da classificação de abandono do aluno universitário em ambiente online, foram utilizados 4 algoritmos, o que se destacou entre eles foi o SVM com acurácia de 92,03%. O estudo considerou 162 participantes no ensino superior na modalidade EaD. Houve seleção de atributos, mas os detalhes específicos não foram mencionados. A variável destacada foi o desempenho nas disciplinas iniciais. (SANTANA et al., 2015).

ER26: Essa pesquisa aborda a categorização de desistência do estudante universitário, sendo empregados 3 algoritmos, destacando-se a Floresta aleatória, com uma taxa de acurácia de 95,12% e *recall* de 91,41%. O estudo englobou 1.516 participantes em nível superior, com ensino presencial. Houve emprego de seleção de atributos, utilizando Regressão Logística, Árvores de Decisão e Floresta Aleatória como algoritmos de classificação. As variáveis destacadas foram a Média do terceiro semestre e a Média dos três primeiros semestres. (COSTA et al., 2020a).

ER30: Esse estudo trata da classificação de abandono de alunos em ambiente online, os modelos utilizados foram Regressão Logística adicionando um termo de regularização e o Modelo de Markov Oculto Input-Output (IOHMM). O estudo abrangeu 32.593 participantes no ensino superior na modalidade EaD. Foi aplicada seleção de atributos, mas os detalhes específicos não foram fornecidos. Não houve destaque específico de variáveis. (MUBARAK; CAO; ZHANG, 2020).

ER34: Esse trabalho trata da classificação de abandono do aluno universitário, o algoritmo utilizado é árvore de decisão aprimorado com base no ID3. A pesquisa englobou 240 participantes do ensino superior em formato presencial. Houve seleção de atributos, porém os detalhes específicos não foram especificados. As variáveis relevantes incluíram problemas familiares,

doença em casa, ambiente do campus, baixa taxa de colação, mudança de objetivo pessoal e problema de ajuste. (SIVAKUMAR; VENKATARAMAN; SELVARAJ, 2016).

ER38: Nessa pesquisa, trata-se da classificação de abandono do aluno em qualquer nível escolar, são utilizadas várias técnicas de classificação, como regras de indução e árvore de decisão. O estudo abordou 670 participantes do ensino médio em formato presencial. Foi aplicada seleção de atributos, embora os detalhes específicos não tenham sido mencionados. Não houve destaque específico de variáveis. (PRADEEP; DAS; KIZHEKKETHOTTAM, 2015).

ER40: Nesse estudo, aborda-se a classificação de abandono de alunos universitários, o algoritmo utilizado é Árvore de decisão. A pesquisa incluiu 206 participantes no ensino superior em formato presencial. Foi aplicada seleção de atributos, embora os detalhes específicos não tenham sido mencionados. As variáveis de destaque incluíram WAG, AI, ASR, SFR, FSE e PF. (BELLO et al., 2020).

ER46: Esse pesquisa explora a classificação de abandono do aluno universitário, onde são comparado dois modelos, um baseado em técnica de regressão logística e o outro modelo baseado em árvores de decisão. A pesquisa envolveu 4.519 participantes no ensino superior em formato presencial. Houve seleção de atributos, embora os detalhes específicos não tenham sido mencionados. Na regressão logística, as variáveis relevantes foram a pontuação de matemática da PSU e as notas do ensino médio (NEM). Na Árvore de Decisão, as variáveis de destaque incluíram NEM, residência do estudante e a pontuação de matemática da PSU. (PÉREZ et al., 2018).

ER52: Esse estudo aborda a classificação de abandono do aluno em MOOCs, são utilizados vários modelos, o melhor desempenho é do modelo *LightGBM*, com acurácia de 96%. O tamanho do conjunto não foi explicitado nesse trabalho. O estudo envolveu níveis de ensino superior, médio e fundamental na modalidade EaD. Houve seleção de atributos, embora os detalhes específicos não tenham sido mencionados. A variável mais relevante foi os dados de interação dos alunos. (PANAGIOTAKOPOULOS et al., 2021).

ER53: Esse trabalho trata da classificação de abandono de alunos universitários, são usados métodos de aprendizagem de máquina baseados em árvore para a predição. O estudo considerou 197 participantes do ensino superior na modalidade EaD. Houve seleção de atributos, embora os detalhes específicos não tenham sido mencionados. As variáveis relevantes incluíram assuntos difíceis de matemática e estatística. (FIGUEROA-CAÑAS; S.VINUESA, 2020).

ER54: Nesse estudo, aborda-se a classificação de abandono de alunos, tanto em escolas quanto em universidades. O algoritmo utilizado é o *Naive Bayes*, com a linguagem R. O trabalho também foca na seleção de recurso para escolha dos melhores atributos. O estudo envolveu 50 participantes no ensino superior em formato presencial. As variáveis relevantes incluíram fatores acadêmicos, demográficos, psicológicos e de saúde. (HEGDE; PRAGEETH, 2018).

ER59: Nesse trabalho, é examinada a classificação do abandono de estudantes universitários,

os algoritmos utilizados foram árvores de decisão C4.5 e ID3. O estudo abrangeu 201 participantes no ensino superior com formato presencial. Foi aplicada seleção de atributos, porém detalhes específicos da seleção de atributos não foram especificados. (HEREDIA; AMAYA; BARRIENTOS, 2015).

ER63: Esse artigo se concentra na descoberta precoce de variáveis de abandono como um avanço pela redução de dimensionalidade usando seleção de atributos e métodos de extração, é utilizada a *Synthetic Minority Oversampling Technique* (SMOTE). O estudo envolveu 1.243 participantes no ensino superior em formato presencial. Foi aplicada seleção de atributos, mas os detalhes específicos não foram mencionados. Não houve destaque específico de variáveis. (REVATHY; KAMALAKKANNAN; KAVITHA, 2022).

ER65: Nesse artigo foi proposto um novo modelo baseado em um híbrido de *Random Forest (RF), Extreme Gradient Boosting (XGBoost), Gradient Boosting (GB) e Feed-forward Neural Networks (FNN)* para prever a evasão de alunos universitários, o modelo se saiu melhor nas métricas de acurácia e AUC, se comparado com modelos clássicos. O estudo abordou o ensino superior em formato presencial, mas o tamanho da amostra não foi especificado. Foi aplicada seleção de atributos, mas os detalhes específicos não foram mencionados. Não houve destaque específico de variáveis. (NIYOGISUBIZO et al., 2022).

ER68: Esse estudo investiga a evasão no ensino médio em alguns países, é também criado um modelo chamado de "AutoML", que foi usado para melhorar a acurácia da previsão, selecionando os hiperparâmetros, recursos e algoritmo ML correspondentes para o conjunto de dados adquirido. O modelo proposto obteve melhores resultados quando comparado a outros modelos. O estudo considerou o ensino fundamental em formato presencial, abrangendo um total de 206.885 participantes. Foi aplicada seleção de atributos, mas os detalhes específicos não foram mencionados. As variáveis relevantes para a análise incluíram notas dos alunos (57%), idade do aluno (18%), distância (7%) e número de crianças (5%). (MNYAWAMI; MAZIKU; MUSHI, 2022).

ER69: Nesse trabalho é realizada uma investigação e criação de modelo de predição para alunos que possuem o transtorno do espectro autista, o modelo de predição desenvolvido é baseado em aprendizagem profunda, usando os algoritmos *Long Short-Term Memory (LSTM) e Multilayer Perceptron (MLP)*. O estudo abordou o ensino fundamental em formato presencial, envolvendo 120 participantes. Foi aplicada seleção de atributos, mas os detalhes específicos não foram mencionados. Não houve destaque específico de variáveis. (JARBOU et al., 2022).

3.1.4.5.5 Estudos que utilizaram métodos clássicos

ER1: Esse trabalho trata da classificação do abandono de alunos universitários, os algoritmos abordados foram, *KNN, Naive Bayes, SVM, Decision Tree e Random Forest*, tiverem resultados bem próximos na maioria das métricas. O estudo abrangeu 1.211 participantes do ensino superior

na modalidade EaD. Não houve seleção de atributos. As variáveis de destaque incluíram etnia, sexo, idade, status de tempo, graduação do estudante, classificação do estudante, GPA cumulativo e GPA de termo. (KANG; WANG, 2018).

ER2: Esse trabalho propõe um modelo de rede neural profunda, que é uma combinação de *Convolutional Neural Network, Long Short-Term Memory network e Support Vector Machine*, para ter um melhor desempenho na predição do abandono do aluno nos cursos online abertos massivos (MOOCs). O estudo englobou 120.542 participantes no ensino superior na modalidade EaD. Foi aplicada seleção de atributos, mas os detalhes específicos não foram especificados. As variáveis relevantes foram ID de inscrição, hora, fonte do evento, evento e dispositivo de acesso. (WU et al., 2019).

ER3: Essa pesquisa aborda a categorização de desistência de estudantes universitários, dando foco na seleção dos atributos que melhoram essa predição de acordo com cada área acadêmica. O conjunto é formado por 24.894 participantes do ensino superior presencial, foi aplicada a abordagem Wrapper com o algoritmo RPart para seleção de atributos. As variáveis destacadas são adaptadas com base no curso específico. (MECA et al., 2020).

ER4: São usados nesse trabalho modelos avançados de aprendizagem de máquina, como redes neurais e árvores aumentadas de gradiente para prever o desempenho do aluno universitário e o abandono do mesmo, que baseia-se nos dados disponíveis no momento da inscrição. O melhor desempenho foi da *Fully Connected Deep Neural Network* (FCNN), com 72.4% de acurácia e 0.771 de AUC. O estudo envolveu 8.319 participantes do ensino superior em formato presencial. Não houve seleção de atributos. Não houve destaque específico de variáveis. (BARANYI; NAGY; MOLONTAY, 2020).

ER5: Esse trabalho trata da classificação de abandono de alunos universitários, dando foco aos cursos de Ciência, Tecnologia, Engenharia e Matemática. Ele compara técnicas de aprendizagem de máquina (AM) com análise de sobrevivência, os algoritmos de AM obtiveram melhores resultados. O estudo considerou 12.293 participantes do ensino superior em formato presencial. Não houve seleção de atributos. As variáveis de destaque foram idade, disciplinas e curso. (CHEN; JOHRI; RANGWALA, 2018).

ER6: São usados nesse estudo os seguintes algoritmos e modelos para predição de abandono na universidade, *Logistic Regression (LR), Adaboost (AB) e Decision tree (DT)* com Cox e TD-Cox. De forma que os melhores resultados foram para *Decision tree (DT)* com Cox e TD-Cox, o Cox com 71,9% de acurácia e 0.751 de AUC, e o TD-Cox com 71,9% e 0.705, no primeiro experimento, e se mantiveram com os melhores resultados no segundo. O estudo envolveu 11.121 participantes do ensino superior em formato presencial. Não houve seleção de atributos. As variáveis relevantes incluíram GPA do ensino médio, informações semestrais como GPA e porcentagem de crédito reprovado, além de atributos financeiros. (AMERI et al., 2016).

ER7: Essa pesquisa explora o abandono de alunos universitários, usando os seguintes algoritmos,

Naive Bayes, Multilayer Perception, Support Vetor Machine e Tabela de Decisão. Tendo o Naive Bayes como algoritmo que obteve os melhores resultados. O estudo abrangeu 1.359 participantes do ensino superior em formato presencial. Não houve seleção de atributos. Não houve destaque específico de variáveis. (MANHÃES; CRUZ; ZIMBRÃO, 2014a).

ER8: Esse trabalho trata da classificação do abandono de alunos universitários, explorando a adequação da diminuição dimensional e concentrando os dados significativos encobertos nas informações de alunos com risco de abandono. Embora todos os algoritmos de aprendizado de máquina utilizados nesse trabalho tenham alcançado uma acurácia de mais de 75%, *Support Vetor Machine* e *Naive Bayes* tiveram melhor desempenho ao prever o abandono. O estudo envolveu 4.417 participantes no ensino superior presencial. Para análise, foram utilizados os métodos SVM, Naïve Bayes, Árvore de Decisão, KNN e Floresta Aleatória. Não houve destaque específico de variáveis. (LOTTERING; HANS; LALL, 2020).

ER10: Esse trabalho trata da classificação do abandono de alunos em programas universitários, foram utilizados algoritmos baseados em árvore de decisão, *Naive Bayes*, K-NN, modelos lineares e aprendizado profundo. Os modelos com melhores resultados foram: *Gradient Boosted Trees* e *Deep Learning*, com uma AUC de 0.808 e 0.811, respectivamente. O estudo abrangeu 15.825 participantes do ensino superior em formato presencial. Não houve destaque específico de variáveis. (NAGY; MOLONTAY, 2018).

ER11: Nessa pesquisa, aborda-se a classificação de alunos universitários, utilizando rede neural profunda. Obtendo uma acurácia de 92,41% com pré-treinamento. O estudo considerou 9.153 participantes do ensino fundamental em formato presencial. Não houve seleção de atributos. Não houve destaque específico de variáveis. (KUO; PAN; LEI, 2017).

ER12: Esse estudo trata da classificação de abandono de alunos em MOOCs, por meio de um modelo denominado CLSA, o modelo usa uma rede neural convolucional. O estudo não especificou o nível de ensino e foi conduzido na modalidade EaD, englobando um total de 79.186 participantes. Não houve seleção de atributos. As variáveis relevantes englobaram características de comportamento dos alunos durante as primeiras 5 semanas. (FU et al., 2021).

ER13: Esse estudo explora da classificação de abandono de alunos em MOOCs, por meio de um modelo denominado CONV-LSTM, o modelo usa redes neurais convolucionais e memória de longo prazo. O estudo considerou o ensino superior na modalidade EaD e envolveu um total de 120.542 participantes. Não houve seleção de atributos. Não houve destaque específico de variáveis (MUBARAK; CAO; HEZAM, 2021).

ER14: Esse trabalho trata da classificação de abandono do aluno do ensino médio, o algoritmo utilizado é Floresta Aleatória aplicado a um big data, com amostra de 165.715 alunos. Não houve seleção de atributos. As variáveis relevantes incluíram ausência, atraso, tempo de atividade autorregulada, tempo de desenvolvimento de carreira e saída não autorizada. (CHUNG; LEE, 2019).

ER15: Esse modelo trata da classificação de abandono de alunos universitários, são usadas as seguintes técnicas de aprendizagem de máquina, redes neurais *feed-forward*, *support vector machine* e conjuntos probabilísticos simplificados Fuzzy ARTMAP. Esses algoritmos são combinados, a fim de obter uma única classificação baseada nessa combinação. O estudo considerou o ensino superior na modalidade EaD, com um total de 193 participantes. Não houve seleção de atributos. Não houve destaque específico de variáveis. (LYKOURENTZOU et al., 2009).

ER17: Esse trabalho trata da classificação de abandono do aluno universitário em ambiente online, foram utilizados 4 algoritmos, o que se destacou entre eles foi o SVM como acurácia de 92,03%. O estudo considerou 162 participantes no ensino superior na modalidade EaD. Houve seleção de atributos, mas os detalhes específicos não foram mencionados. A variável destacada foi o desempenho nas disciplinas iniciais. (SANTANA et al., 2015).

ER18: Esse estudo explora a classificação de abandono de alunos no ensino fundamental e médio, o algoritmo utilizado foi árvores de decisão. O estudo abordou o ensino fundamental em formato presencial, englobando 220 participantes. Não houve seleção de atributos. Não houve destaque específico de variáveis. (SELVAN; NAVADURGA; PRASANNA, 2019).

ER19: Nessa pesquisa, trata-se da classificação de abandono do aluno universitário, especificamente de um curso de computação empresarial, o algoritmo utilizado foi Árvores de decisão. O estudo considerou 1.888 participantes do ensino superior em formato presencial. Não houve seleção de atributos. As variáveis de destaque incluíram curso e período. (NUANKAEW et al., 2019).

ER20: Nesse estudo, explora-se a classificação de abandono de alunos universitários, é utilizado um sistema que utiliza várias técnicas de aprendizagem de máquina, esse sistema foi feito com intuito de auxiliar os tutores na Universidade de Barcelona. O estudo abrangeu 4.434 participantes do ensino superior em formato presencial. Não houve seleção de atributos. Não houve destaque específico de variáveis. (ROVIRA; PUERTAS; IGUAL, 2017).

ER21: Essa pesquisa trata da classificação de abandono do aluno universitário em ambiente online, foi utilizado regressão logística lasso e ridge para criar um modelo de previsão de abandono. O estudo envolveu 32.593 participantes do ensino superior na modalidade EaD. Não houve seleção de atributos. Não houve destaque específico de variáveis. (RADOVANOVIĆ; DELIBAŠIĆ; SUKNOVIĆ, 2020).

ER22: Nessa pesquisa, trata-se da classificação de abandono do aluno universitário, ele utiliza uma combinação de algoritmos para aprimorar a classificação. O estudo considerou 200 participantes do ensino superior na modalidade EaD. Não houve seleção de atributos. Não houve destaque específico de variáveis. (DEWAN et al., 2015).

ER23: Esse trabalho explora a classificação de abandono do aluno universitário, especificamente do curso de ciência da computação, os algoritmos utilizados foram *Naive Bayes*, árvores de

decisão, Regressão Logística, Máquinas de Vetores de Suporte e Redes Neurais. O estudo considerou 367 participantes do ensino superior em formato presencial. Não houve seleção de atributos. As variáveis de destaque foram ECTS do primeiro semestre (ECTS_1_sem) e SGPA do primeiro semestre (SGPA_1_sem). (MAKSIMOVA; PENTEL; DUNAJEVA, 2020).

ER24: Esse trabalho trata da classificação de abandono do aluno universitário, especificamente no curso de ciência da computação, os algoritmos utilizados foram Árvore de decisão, regressão logística, floresta aleatória, K-vizinho mais próximo e algoritmo de rede neural. O conjunto do estudo é composto por 64 participantes do ensino superior presencial, a variável mais importante é o desempenho em disciplinas específicas. (YAACOB et al., 2020).

ER26: Essa pesquisa aborda a categorização de desistência do estudante universitário, sendo empregados 3 algoritmos, destacando-se a Floresta aleatória, com uma taxa de acurácia de 95,12% e *recall* de 91,41%. O estudo englobou 1.516 participantes em nível superior, com ensino presencial. Houve emprego de seleção de atributos, utilizando Regressão Logística, Árvores de Decisão e Floresta Aleatória como algoritmos de classificação. As variáveis destacadas foram a Média do terceiro semestre e a Média dos três primeiros semestres. (COSTA et al., 2020a).

ER27: Essa pesquisa aborda a categorização de desistência do estudante universitário, sendo empregados os algoritmos KNN, árvore de decisão, floresta aleatória, SVM e redes neurais. O modelo de redes neurais demonstrou o melhor desempenho, alcançando uma acurácia de 92%. O estudo contou com 26 participantes no nível superior, com aulas presenciais. Não houve seleção de atributos. Não houve destaque específico de variáveis. (QUISHPE-MORALES et al., 2020).

ER28: Nesse estudo, é explorada a classificação de abandono do aluno universitário, ele utiliza uma combinação de 3 algoritmos de mineração de dados populares, floresta aleatória, máquinas de vetores de suporte e redes neurais artificiais. O estudo considerou o ensino superior presencial, abrangendo 3.373/3.344 participantes. Foi aplicada a técnica de seleção de atributos forward search. Para o 1º semestre, variáveis destacadas incluíram ects_aprov_s, cod_escola, média_s, sexo, bolseiro_s, entre outras; para o 2º semestre, variáveis como ects_reprov_s, cod_prof_mae, n10_11_acesso, idade, nacionalidade, entre outras, foram relevantes. (MARTINS et al., 2020).

ER29: Nesse estudo, aborda-se a categorização da evasão de estudantes universitários. Como parte da pesquisa, são empregados os algoritmos *Decision Stump*, NDTREE e *Enhanced Machine Learning* (EMLA), os quais são fundamentados em técnicas de aprendizado de máquina. O estudo envolveu 407 participantes no nível superior, com ensino presencial. Não foi realizada seleção de atributos. Não houve destaque específico de variáveis. (SALLAN; BEHAL,).

ER30: Esse estudo trata da classificação de abandono de alunos em ambiente online, os modelos utilizados foram Regressão Logística adicionando um termo de regularização e o Modelo de Markov Oculto Input-Output (IOHMM). O estudo abrangeu 32.593 participantes no ensino superior na modalidade EaD. Foi aplicada seleção de atributos, mas os detalhes específicos não foram fornecidos. Não houve destaque específico de variáveis. (MUBARAK; CAO; ZHANG,

2020).

ER31: Nessa pesquisa, explora-se a classificação de abandono do aluno universitário, os algoritmos utilizados foram regressão logística e árvores de decisão. Como resultado, obteve-se que a técnica com maior percentual de acurácia de abandono foi a árvore de decisão com 91.70%. O estudo englobou 1.178 participantes no ensino superior em formato presencial. Houve seleção de atributos utilizando a técnica Relief Feature Selection. Não houve destaque específico de variáveis. (ALBAN; MAURICIO, 2018a).

ER32: Esse trabalho analisa modelos de classificação de abandono do aluno universitário, foram utilizados os algoritmos *Decision Tree* e *Naive Bayes*, o *Decision Tree* obteve melhores resultados. A pesquisa abrangeu 1.862 participantes no ensino superior com formato presencial. Foi aplicada a seleção de atributos baseada no algoritmo Firefly. Não houve destaque específico de variáveis. (GAMAO; GERARDO, 2019).

ER34: Esse trabalho trata da classificação de abandono do aluno universitário, o algoritmo utilizado é árvore de decisão aprimorado com base no ID3. A pesquisa englobou 240 participantes do ensino superior em formato presencial. Houve seleção de atributos, porém os detalhes específicos não foram especificados. As variáveis relevantes incluíram problemas familiares, doença em casa, ambiente do campus, baixa taxa de colação, mudança de objetivo pessoal e problema de ajuste. (SIVAKUMAR; VENKATARAMAN; SELVARAJ, 2016).

ER35: Essa pesquisa trata da classificação de abandono do aluno universitário, os algoritmos utilizados foram *K-Nearest Neighbor* (KNN), *Naïve Bayes* (NB) e *Decision Tree* (DT). A pesquisa abordou 17.432 participantes no ensino superior presencial. Utilizou a técnica Learning Vector Quantization para seleção de atributos. As variáveis de destaque incluíram frequência, pontuações de tarefas, créditos totais, pontuações UTS, UAS pontuações, GPA, renda dos pais, educação dos pais, sexo e idade dos estudantes. (HUTAGAOL; SUHARJITO, 2019).

ER36: Nesse estudo, trata-se da classificação de abandono de alunos no ensino base, foram utilizados vários algoritmos, apesar de tradicional o SVM mostrou bons resultados comparado com modelos mais complexos. O conjunto é formado por um total de 15.000 participantes. Foi aplicada seleção de atributos, mas os detalhes específicos não foram mencionados. Não houve destaque específico de variáveis. (BONIFRO et al., 2020).

ER37: Esse trabalho trata da classificação do abandono de alunos universitários. Foram desenvolvidos 3 classificadores com base em uma árvore de decisão, floresta aleatória e classificação de aumento de gradiente. Os resultados mostram que a acurácia da previsão de aumento de gradiente, árvore de decisão e modelos de floresta aleatória são 93%, 92% e 92%, respectivamente. O trabalho também aponta alguns atributos relevantes na classificação, ano letivo, GPA do ensino médio e canais de admissão. O estudo envolveu 13.714 participantes do ensino superior em formato presencial. Não houve seleção de atributos. (TENPIPAT; AKKARAJITSAKUL, 2020).

ER38: Nessa pesquisa, trata-se da classificação de abandono do aluno em qualquer nível escolar,

são utilizadas várias técnicas de classificação, como regras de indução e árvore de decisão. O estudo abordou 670 participantes do ensino médio em formato presencial. Foi aplicada seleção de atributos, embora os detalhes específicos não tenham sido mencionados. Não houve destaque específico de variáveis. (PRADEEP; DAS; KIZHEKKETHOTTAM, 2015).

ER39: Esse trabalho aborda a classificação de abandono do aluno universitário, especificamente no curso de Ciências da computação. O algoritmo utilizado é Árvore de decisão, aplica-se também seleção de recurso para identificar quais os melhores atributos para a classificação. A pesquisa incluiu 963 participantes no nível superior, com aulas presenciais. A seleção de atributos foi realizada por meio do algoritmo Boruta. As variáveis em destaque abrangeram GRADE, A1SCORE e QATTEMPT. (NASEEM et al., 2019).

ER40: Nesse estudo, aborda-se a classificação de abandono de alunos universitários, o algoritmo utilizado é Árvore de decisão. A pesquisa incluiu 206 participantes no ensino superior em formato presencial. Foi aplicada seleção de atributos, embora os detalhes específicos não tenham sido mencionados. As variáveis de destaque incluíram WAG, AI, ASR, SFR, FSE e PF. (BELLO et al., 2020).

ER41: Esse trabalho trata da classificação de abandono do aluno em qualquer nível escolar, ele utiliza técnicas de previsão, com foco em métodos de classificação baseados em árvores e SVM. O estudo abrangeu 220.685 participantes da modalidade presencial. Não houve seleção de atributos. As variáveis de destaque foram ausências geralmente altas e pontuações baixas em matemática. (SORENSEN, 2019).

ER42: Esse estudo trata da classificação de abandono de alunos em MCCOs, o estudo utiliza um sistema de inferência neuro-difuso adaptativo (ANFIS), que é comparado com algoritmos clássicos, como árvore de decisão, regressão logística, svm, etc. Os resultados mostram que a ANFIS obteve o melhor desempenho com relação aos algoritmos. O estudo abrangeu os níveis de ensino superior, médio e fundamental na modalidade EaD, com um total de 120.542 participantes. Não houve seleção de atributos. Não houve destaque específico de variáveis. (ŞAHIN, 2021).

ER43: Nesse estudo, explora-se a categorização do abandono de alunos universitários, utilizando diferentes algoritmos, como Floresta aleatória, máquinas de vetores de suporte, árvores de decisão, Naive Bayes, K-vizinho mais próximo e regressão logística. Destaca-se que o algoritmo de Floresta aleatória apresentou os melhores resultados, com uma acurácia de 94.14%. O estudo abordou o ensino superior em formato presencial, envolvendo 12.293 participantes. O artigo não menciona a aplicação de uma abordagem específica para seleção de atributos. As variáveis importantes englobaram a idade, as disciplinas e o curso dos participantes. (LOTTERING; HANS; LALL,).

ER44: Esse trabalho trata da classificação de abandono do aluno universitário, o modelo de classificador é baseado em árvores de decisão. O estudo considerou 6.870 participantes da modalidade presencial. Não houve seleção de atributos. As variáveis de destaque foram média

baixa nas notas e o número de cursos perdidos nos semestres iniciais do programa. (PEREIRA; ZAMBRANO, 2017).

ER45: Essa pesquisa tem como foco a classificação de abandono do aluno universitário, especificamente no curso de engenharia de sistemas. Os algoritmos utilizados nos experimentos foram: Árvore de decisão, Regressão logística e Naive Bayes. Tendo Árvore de decisão como algoritmo que obteve melhor resultado, AUC de 0.94. O estudo envolveu 802 indivíduos no nível superior com aulas presenciais. Não foi realizada seleção de atributos. Não houve destaque específico de variáveis. (PEREZ; CASTELLANOS; CORREAL, 2018).

ER46: Esse pesquisa explora a classificação de abandono do aluno universitário, onde são comparado dois modelos, um baseado em técnica de regressão logística e o outro modelo baseado em árvores de decisão. A pesquisa envolveu 4.519 participantes no ensino superior em formato presencial. Houve seleção de atributos, embora os detalhes específicos não tenham sido mencionados. Na regressão logística, as variáveis relevantes foram a pontuação de matemática da PSU e as notas do ensino médio (NEM). Na Árvore de Decisão, as variáveis de destaque incluíram NEM, residência do estudante e a pontuação de matemática da PSU. (PÉREZ et al., 2018).

ER47: Esse estudo trata da classificação de abandono do aluno universitário, mais especificamente dos cursos de engenharia. Os algoritmos utilizados foram Árvores de decisão, regressão logística e Naive Bayes. O estudo abrangeu 762 participantes da modalidade presencial. Não houve seleção de atributos. A variável de destaque foi o Std GPA (GPA padronizado). (PÉREZ-GUTIÉRREZ, 2020).

ER48: Esse trabalho trata da classificação de abandono do aluno em ambiente online, modelos de regressão logística são usados para a classificação. O estudo considerou o ensino superior, envolvendo um total de 104 participantes. Não houve seleção de atributos. A variável de destaque foram as notas nas semanas cruciais do curso. (BURGOS et al., 2018).

ER49: Essa pesquisa aborda a classificação de abandono do aluno em MOOCs, é estudado modelos de aprendizado profundo, já que segundo o estudo essa abordagem constrói modelos de previsão de abandono mais precisos. O estudo abrangeu os níveis de ensino superior, médio e fundamental, tendo como conjunto um total de 3.617 participantes. Não houve seleção de atributos. Não houve destaque específico de variáveis. (XING; DU, 2019).

ER50: Esse trabalho aborda a classificação de abandono do alunos do ensino médio de um instituto de tecnologia da Tailândia, os algoritmos utilizados foram Árvores de decisão e Floresta aleatória, Árvores de decisão obteve os melhores resultados. A pesquisa abrangeu 28.801 participantes em nível superior com ensino presencial. Utilizou-se o critério GPAX para seleção de atributos, as notas nos cursos de idiomas foram identificadas como variáveis de destaque. (LIMSATHITWONG; TIWATTHANONT; YATSUNGNOEN, 2018).

ER52: Esse estudo aborda a classificação de abandono do aluno em MOOCs, são utilizados

vários modelos, o melhor desempenho é do modelo *LightGBM*, com acurácia de 96%. O tamanho do conjunto não foi explicitado nesse trabalho. O estudo envolveu níveis de ensino superior, médio e fundamental na modalidade EaD. Houve seleção de atributos, embora os detalhes específicos não tenham sido mencionados. A variável mais relevante foi os dados de interação dos alunos. (PANAGIOTAKOPOULOS et al., 2021).

ER53: Esse trabalho trata da classificação de abandono de alunos universitários, são usados métodos de aprendizagem de máquina baseados em árvore para a predição. O estudo considerou 197 participantes do ensino superior na modalidade EaD. Houve seleção de atributos, embora os detalhes específicos não tenham sido mencionados. As variáveis relevantes incluíram assuntos difíceis de matemática e estatística. (FIGUEROA-CAÑAS; S.VINUESA, 2020).

ER54: Nesse estudo, aborda-se a classificação de abandono de alunos, tanto em escolas quanto em universidades. O algoritmo utilizado é o *Naive Bayes*, com a linguagem R. O trabalho também foca na seleção de recurso para escolha dos melhores atributos. O estudo envolveu 50 participantes no ensino superior em formato presencial. As variáveis relevantes incluíram fatores acadêmicos, demográficos, psicológicos e de saúde. (HEGDE; PRAGEETH, 2018).

ER55: Essa pesquisa trata da classificação de abandono do aluno em ambiente online, são usados modelos de regressão para a classificação. O estudo abordou 104 participantes do ensino superior na modalidade EaD. Não houve seleção de atributos. As variáveis de destaque foram as notas nas semanas cruciais do semestre. (PEÑA et al., 2017).

ER56: Nessa pesquisa, explora-se a classificação de abandono do aluno no ambiente online, ele apresenta um *benchmark* de algoritmo recentemente proposto, algoritmo de modelo de folha (LLM), comparando-o com outros 8 algoritmos, o LLM tem os melhores resultados se comparado aos outros algoritmos. O estudo envolveu níveis de ensino superior, médio e fundamental na modalidade EaD, com um total de 10.554 participantes. Não houve seleção de atributos. A variável de destaque foi acadêmico noivado. (COUSSEMENT et al., 2020).

ER57: Esse trabalho trata da classificação de abandono do aluno em ambiente online, os modelos de previsão foram desenvolvidos usando *Artificial Neural Network (ANN), Decision Tree (DT) e Bayesian Networks (BNs)*, o que obteve melhor desempenho foi DT. O estudo considerou 62.375 participantes no ensino superior na modalidade EaD. Não houve seleção de atributos. Não houve destaque específico de variáveis. (TAN; SHAO, 2015).

ER58: Nesse estudo, aborda-se a classificação de abandono do aluno universitário, os métodos de classificação utilizados incluíram *Naive Bayes*, C4.5, *BackPropagation*, SMO, 3NN e MLE, é implementando também um protótipo de sistema web utilizando o *Naive Bayes*. A pesquisa envolveu 354 participantes no ensino superior na modalidade EaD. A seleção de atributos foi realizada por meio da abordagem Wrapper. Não houve destaque específico de variáveis. (KOTSIANTIS; PIERRAKEAS; PINTELAS, 2003).

ER59: Nesse trabalho, é examinada a classificação do abandono de estudantes universitários,

os algoritmos utilizados foram árvores de decisão C4.5 e ID3. O estudo abrangeu 201 participantes no ensino superior com formato presencial. Foi aplicada seleção de atributos, porém detalhes específicos da seleção de atributos não foram especificados. (HEREDIA; AMAYA; BARRIENTOS, 2015).

ER60: Esse trabalho trata da classificação de abandono do aluno em ambiente online, foi utilizado um modelo mais robustos nesse estudo, modelo de rede neural recorrente (RNN) com células de memória de curto prazo longa (LSTM). O estudo abrangeu níveis de ensino superior, médio e fundamental na modalidade EaD, com um total de 39.877/27.629 participantes. Não houve seleção de atributos. Não houve destaque específico de variáveis. (FEI; YEUNG, 2015).

ER63: Esse artigo se concentra na descoberta precoce de variáveis de abandono como um avanço pela redução de dimensionalidade usando seleção de atributos e métodos de extração, é utilizada a *Synthetic Minority Oversampling Technique* (SMOTE). O estudo envolveu 1.243 participantes no ensino superior em formato presencial. Foi aplicada seleção de atributos, mas os detalhes específicos não foram mencionados. Não houve destaque específico de variáveis. (REVATHY; KAMALAKKANNAN; KAVITHA, 2022).

ER64: Esse trabalho realiza uma investigação da evasão de estudantes universitários, nessa investigação é considerado o contexto da pandemia do COVID-19 e quais fatores da pandemia influenciaram na evasão. Para a construção dos modelos de predição foi utilizado algoritmos populares, como SVM, regressão logística, floresta aleatória, árvore de decisão, etc. O estudo considerou a modalidade EaD, sem especificar o tamanho da amostra. Não houve seleção de atributos. As variáveis mais relevantes incluíram Número de Semestres com Reprovação, Motivo de Reprovação em Semestres, Tipo de Universidade, Impacto da Covid, Estudante de Engenharia, Média CGPA na Disciplina 'C', Área de Residência, Gênero, Disponibilidade de Internet e Renda Familiar. (HOSSAIN et al., 2022).

ER65: Nesse artigo foi proposto um novo modelo baseado em um híbrido de *Random Forest (RF), Extreme Gradient Boosting (XGBoost), Gradient Boosting (GB) e Feed-forward Neural Networks (FNN)* para prever a evasão de alunos universitários, o modelo se saiu melhor nas métricas de acurácia e AUC, se comparado com modelos clássicos. O estudo abordou o ensino superior em formato presencial, mas o tamanho da amostra não foi especificado. Foi aplicada seleção de atributos, mas os detalhes específicos não foram mencionados. Não houve destaque específico de variáveis. (NIYOGISUBIZO et al., 2022).

ER66: Nesse trabalho foi desenvolvido um sistema que permite o cálculo do risco de evasão por aluno e utiliza um procedimento de geração de alertas para coordenar as intervenções. A plataforma permitiu mensurar o impacto das estratégias de intervenção na permanência dos alunos. O estudo abrangeu 15.805 participantes no ensino superior em formato presencial. Foi empregada a abordagem ingênua (naive approach) para seleção de atributos. As técnicas de classificação utilizadas incluíram o algoritmo *AdaBoost*, *Bayesian GLM*, *Decision Trees*, *LogitBoost*, *Random Forest* (RF) e *Stochastic Gradient Boosting*. Não houve destaque específico

de variáveis. (GUZMÁN-CASTILLO et al., 2022).

ER67: Esse estudo criou um modelo combinando seleção de atributos com um método de rede neural perceptron multicamadas. O modelo foi comparado com modelos baseados nos algoritmos de *Logistic regression*, *Decision Tree*, *Random Forest*, *Naive Bayes*, *Support Vector Machine* e *Multilayer Perceptron Neural Network*, e obteve melhores resultados. A pesquisa considerou 1.650 participantes no ensino superior presencial. Foram aplicadas as técnicas de seleção de atributos GR, CS e Métodos CFS. As variáveis destacadas englobaram a média cumulativa de notas (GPA), média cumulativa de notas para assuntos não docentes (GPAnone) e adesão ao grupo de mídia social em assuntos (*Socialclass*). (NUANMEESRI et al., 2022).

ER68: Esse estudo investiga a evasão no ensino médio em alguns países, é também criado um modelo chamado de "AutoML", que foi usado para melhorar a acurácia da previsão, selecionando os hiperparâmetros, recursos e algoritmo ML correspondentes para o conjunto de dados adquirido. O modelo proposto obteve melhores resultados quando comparado a outros modelos. O estudo considerou o ensino fundamental em formato presencial, abrangendo um total de 206.885 participantes. Foi aplicada seleção de atributos, mas os detalhes específicos não foram mencionados. As variáveis relevantes para a análise incluíram notas dos alunos (57%), idade do aluno (18%), distância (7%) e número de crianças (5%). (MNYAWAMI; MAZIKU; MUSHI, 2022).

ER69: Nesse trabalho é realizada uma investigação e criação de modelo de predição para alunos que possuem o transtorno do espectro autista, o modelo de predição desenvolvido é baseado em aprendizagem profunda, usando os algoritmos *Long Short-Term Memory (LSTM) e Multilayer Perceptron (MLP)*. O estudo abordou o ensino fundamental em formato presencial, envolvendo 120 participantes. Foi aplicada seleção de atributos, mas os detalhes específicos não foram mencionados. Não houve destaque específico de variáveis. (JARBOU et al., 2022).

ER70: O objetivo nesse trabalho é reduzir a taxa de evasão dos alunos da Faculdade de Engenharia de Sistemas e Informática da Universidade Nacional Mayor de San Marcos (FISI-UNMSM), o modelo de predição utilizado é baseado em árvores de decisão, obteve uma acurácia de 90.34%. A pesquisa abrangeu 1.938 participantes no ensino superior presencial. Foram aplicadas as técnicas de seleção de atributos ChiSquared, OneR, GainRatio, InforGain e ClassifierAttribute. As variáveis destacadas incluíram a média ponderada histórica de notas, a média ponderada de notas do último ciclo e o número de créditos de cursos aprovados. (VEGA et al., 2022).

ER71: Nesse trabalho é feita uma análise da evasão no curso de Ciência da Computação, os algoritmos utilizados foram *Random Forest*, *Decision Tree*, *Naive Bayes*, *Logistic Regression* e *K-Nearest Neighbour*. No primeiro estágio o algoritmo que obteve o melhor resultado foi o Naive Bayes com uma AUC de 0.6123, nos estágios 2 e 3 o algoritmo que obteve o melhor resultado foi o Logistic Regression com AUC de 0.7523 e 0.8902, respectivamente. O tamanho do conjunto de dados não foi mencionado no artigo, mas trata-se de dados de uma única Instituição de Ensino Superior da região do Pacífico Sul, que é o único campus regional no Pacífico Sul. (NASEEM;

CHAUDHARY; SHARMA, 2022).

3.1.4.5.6 Estudos que não utilizaram métodos clássicos

ER9: Esse trabalho foca na redução de dimensionalidade dos atributos para melhorar a classificação de abandono de alunos universitários. O estudo envolveu 150 participantes no ensino superior com aulas presenciais. Foi realizada redução dimensional utilizando a técnica PCA (Análise de Componentes Principais). Não houve destaque específico de variáveis. (HEGDE, 2016).

ER16: Esse estudo trata da classificação de abandono de alunos de todos níveis de ensino, utilizando Modelos de Equações Estruturais de Mistura (MSEM). O estudo envolveu 12.548 participantes da modalidade presencial. Não houve seleção de atributos. As variáveis de destaque foram saúde do aluno, frequência e relações interpessoais. (VILORIA; LEZAMA, 2019).

ER25: Nesse estudo, trata-se da classificação de abandono de alunos universitários, mais especificamente do curso de enfermagem. Este estudo relata o uso do pacote Answer Tree da SPSS para esse propósito. O estudo considerou 528 participantes em modalidade presencial. Não houve seleção de atributos. Não houve destaque específico de variáveis. (MOSELEY; MEAD, 2008).

ER33: Nesse trabalho, trata-se da classificação de abandono dos alunos universitários. O modelo aplicado é a regressão linear e o estudo também enfoca a seleção de recursos. O estudo incluiu 530 indivíduos no ensino superior, com aulas presenciais. A seleção de atributos foi conduzida por meio da Correlação de Pearson. Não houve destaque específico de variáveis. (AGUIRRE; PÉREZ, 2020).

ER51: Essa pesquisa explora a classificação de abandono de aluno universitário, ele utiliza em seu benchmark um sistema chamado *FragSte*, um sistema de detecção precoce para universidades alemãs, o FragSte aplica métodos de aprendizagem de máquina. O estudo abrangeu 29.500 participantes em modalidade presencial. Não houve seleção de atributos. Não houve destaque específico de variáveis. (BERENS et al., 2018).

ER61: É apresentado nesse estudo um novo modelo que usa vários módulos Inception conectados por meio de uma rede residual. Operações paralelas são executadas nos dados usando a operação MaxPooling e três comprimentos diferentes de convolução dentro do módulo inicial. É feito um comparativo com outros modelos e o modelo proposto se saiu significativamente melhor. O estudo abordou os níveis de ensino superior, médio e fundamental na modalidade EaD, sem especificar o tamanho da amostra. Não houve seleção de atributos. Não houve destaque específico de variáveis. (LI; CUI; ZHANG, 2022).

ER62: Esse trabalho tem como objetivo melhorar um sistema de alerta de evasão, nele é incorporado fatores relacionados ao ambiente escolar e ao aprendizado socioemocional, diferenciando-se de outros sistemas de evasão que dependem exclusivamente dos fatores: frequência, comportamento e curso. O estudo considerou o ensino fundamental em formato presencial, com um total

de 62.000 e 48.900 participantes em duas situações distintas. Não houve seleção de atributos. O algoritmo de classificação utilizado foi Regressão Linear. As variáveis de destaque foram Competência Linguística Cultural, Relacionamentos, Segurança Emocional e Segurança Física. (SU et al., 2022).

É notório o uso majoritário de algoritmos baseados em árvores de decisão na grande maioria dos experimentos, além deles mostrarem excelentes resultados em várias métricas de análise de modelos de classificação, como acurácia, curva ROC, entre outras, se comparados com algoritmos clássicos, também é notado que a modalidade de ensino presencial e alunos no nível universitário foram os mais utilizados nas predições, ainda assim foram encontrados muitos trabalhos que realizam a predição em modalidade EaD, principalmente quando se trata de MOOCs (do inglês, *Massive Open Online Course*).

Desempenho durante determinado tempo de estudo, disciplina e curso, são as variáveis mais destacadas pelos modelos de classificação. Todos os estudos tiveram uma abordagem empírica, e tratando-se dos resultados dos modelos, mais de 30% dos ER tiveram em seus resultados acurácia superiores a 95%, o estudo com menor acurácia obteve 61.68%.

3.1.5 Ameaças à Validade

Mesmo o mapeamento sistemático tendo sido realizado de forma cautelosa, ainda é possível elencar algumas possíveis limitações ou ameaças que afetem a validade dos resultados, sendo elas: (i) a decisão sobre quais estudos incluir ou excluir, pode em algum momento ter sido tendenciosa, mesmo que de forma não intencional e, portando, uma ameaça. De forma a minimizá-la, os processos de seleção foram realizados e revisados em par; (ii) a *string* de busca pode não incluir todos os termos relacionados ao tema de pesquisa. No entanto, foram realizados testes pilotos de maneira a ajustar e refinar a *string* apresentada; e (iii) este mapeamento sistemático focou totalmente em estudo empíricos, descartando estudos como revisões ou mapeamentos sistemáticos, aumentando a ameaça de validade externa. Ainda assim esse mapeamento abrangeu o escopo ao investigar a evasão escolar em todos os níveis de ensino e modalidades de ensino, trazendo dessa forma mais detalhes sobre os experimentos de datasets variados (MORAIS et al., 2021).

3.1.6 Considerações do Mapeamento

Esse mapeamento teve como objetivo mapear sistematicamente os artigos do estado da arte sobre a aplicação de DM e ML na classificação de dados em estudos sobre evasão escolar.

Para realização desse mapeamento sistemático foram utilizadas as bases ACM, IEEE, Scopus, Web of Science e ScienceDirect, citadas anteriormente. Logo depois, foram filtrados os estudos primários, através dos critérios de inclusão e exclusão, tendo como resultado os ER, que foram utilizados para responder às questões propostas. Dessa forma, percebe-se que a quantidade

de publicações sobre aplicação de DM e ML na classificação de dados em estudos sobre evasão escolar é crescente, e que surgem cada vez mais modelos, algoritmos, sistemas que ajudam na evolução da área de pesquisa. Além disso, percebeu-se um uso maior de algoritmos baseados em árvores de decisão nos experimentos, e que também a modalidade de ensino presencial e alunos no nível universitário foram os mais abordados nos trabalhos.

Portanto, é perceptível os avanços de algoritmos, softwares, sistemas e outros, que tem como objetivo melhorar o entendimento sobre o fenômeno da evasão escolar, tanto na avaliação de métodos de AM em termos de acurácia, curva ROC, como também na identificação de variáveis relevantes para prever a evasão, ajudando assim na gestão das instituições de ensino, consequentemente reduzindo prejuízos para as mesmas.

3.2 Trabalhos Empíricos

3.2.1 WAVE: An Architecture for Predicting Dropout in Undergraduate Courses Using EDM

O trabalho "WAVE: An Architecture for Predicting Dropout in Undergraduate Courses Using EDM", esse artigo mostra resultados do experimento inicial usando dados do mundo real sobre três cursos de graduação em engenharia de uma das maiores universidades públicas do Brasil, a UFRJ. Ele trata do abandono de alunos universitários, usando os seguintes algoritmos: Naive Bayes, Multilayer Perception, Support Vetor Machine e Tabela de Decisao. Tendo o Naive Bayes como algoritmo que obteve os melhores resultados (MANHÃES; CRUZ; ZIMBRÃO, 2014b).

3.2.2 Prediction analysis of student dropout in a Computer Science course using Educational Data Mining

O segundo trabalho experimental relacionado é o "Prediction analysis of student dropout in a Computer Science course using Educational Data Mining", esse trabalho apresenta um modelo que pode predizer o risco de evasão do aluno a partir de dados dos três primeiros semestres frequentados por alunos de graduação em Ciência da Computação (N = 1516) da Universidade Federal de Pelotas. Esse trabalho trata da classificação de abandono do aluno universitário, são utilizados 3 algoritmos, o que se destaca entre eles é o Floresta aleatória, com acurácia de 95,12% e Recall de 91,41% (COSTA et al., 2020b).

3.2.3 Prediction of university dropout through technological factors: A case study in Ecuador

E por último o "Prediction of university dropout through technological factors: A case study in Ecuador", o objetivo desse artigo e analisar a influência de fatores relacionados ao uso inadequado de internet, mídias sociais e tecnologia na evasão universitária e avaliar por meio de técnicas de mineração de dados se esses fatores podem prever a evasão universitária. Esse trabalho trata da classificação de evasão do aluno universitário, os algoritmos utilizados foram regressão logística e árvores de decisão. Como resultado, obteve-se que a técnica com maior percentual de acurácia de evasão foi a árvore de decisão, com 91,70% (ALBAN; MAURICIO, 2018b).

4

Desenvolvimento

As etapas desse trabalho seguem a metodologia Cross-Industry Standard Process for Data Mining (CRISP-DM) (SHEARER, 2000). Segundo a metodologia CRISP-DM, o processo de MD deve conter seis fases: (1) Entendimento do negócio, (2) Entendimento dos Dados, (3) Preparação dos Dados, (4) Modelagem, (5) Avaliação e (6) Implantação.

4.1 Fase 1 - Entendimento do Negócio

Na fase inicial do CRISP-DM, concentramo-nos em estabelecer um entendimento claro do negócio, alinhando os objetivos do trabalho com as demandas, desafios e requisitos essenciais relacionados à evasão estudantil no ensino superior. Para este trabalho, o foco principal é investigar a evasão estudantil no ensino superior e a aplicação de técnicas de Mineração de Dados e Aprendizagem de Máquina na classificação da evasão.

- **Objetivos Norteadores**:
  - Identificar Necessidades da Área: Explorar e compreender as demandas, desafios e requisitos essenciais relacionados à evasão estudantil no ensino superior. Este processo visa alinhar os objetivos do trabalho com as exigências inerentes ao campo de estudo, abordando as complexidades associadas à investigação da evasão por meio de técnicas de Mineração de Dados e Aprendizagem de Máquina.
  - Definir Metas Específicas: Estabelecer metas claras e mensuráveis para o projeto, visando a criação de modelos preditivos eficazes na previsão de evasão estudantil e que possam ser utilizados por diferentes instituições de ensino.
  - Mapear Impactos Potenciais: Avaliar o impacto que a previsão de evasão pode ter no ambiente educacional superior, considerando tanto aspectos operacionais quanto estratégicos.

Ao definir esses objetivos norteadores nesta fase, buscamos uma compreensão mais aprofundada das nuances do ambiente educacional superior, alinhando-se de maneira mais precisa com os objetivos específicos de investigar a evasão estudantil e a classificação de evasão.

4.2 Fase 2 - Entendimento dos Dados

O Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (Inep), vinculado ao Ministério da Educação (MEC), realiza pesquisas e avaliações nos vários níveis de Ensino e disponibiliza diversas bases de dados em formato de Dados Abertos [1]. Neste trabalho, utilizamos as bases de dados referentes aos Censos da Educação Superior de 2016 e 2017.

Essas bases contêm informações valiosas sobre instituições, cursos e alunos do ensino superior. É importante ressaltar que as informações sobre instituições, cursos e alunos estão armazenadas em tabelas distintas, disponibilizadas no formato CSV.

4.3 Fase 3 - Preparação dos Dados

As bases de dados utilizadas tanto da modalidade EAD como da presencial foram do INEP, referentes ao Censo de Educação Superior do ano de 2016. As etapas de preparação dos dados foram as seguintes: Mineração dos atributos; Balanceamento do *dataset*; Divisão do *dataset* em *dataset* de treino e *dataset* de teste.

4.3.1 Etapa 1 - Mineração dos atributos:

Os *datasets* originais possuem 119 atributos, após passar por um processo de seleção de atributos, foram descartados atributos redundantes: atributos que possuem um código de identificação e uma descrição como outro atributo, esses atributos de descrição foram excluídos; atributos que são códigos de identificação formados por letras e números, precisam ser tratados futuramente, portanto foram excluídos desses experimentos iniciais para melhor desempenho dos algoritmos, após essa seleção restou um total de 96 atributos, o atributo DESISTENTE foi criado, mas não é considerado para a construção dos modelos.

4.3.2 Etapa 2 - Balanceamento do *dataset*:

Os conjuntos de dados iniciais revelaram desequilíbrios notáveis nas modalidades específicas de ensino à distância (EAD) e presencial, tanto na esfera pública quanto na privada. Em resposta a essa disparidade, aplicamos a técnica de *Undersampling* para ajustar essas discrepâncias e criar subconjuntos balanceados para cada modalidade e categoria administrativa.

[1] https://www.gov.br/inep/pt-br/acesso-a-informacao/dados-abertos/microdados/censo-da-educacao-superior

Por exemplo, na modalidade EAD e categoria administrativa pública, inicialmente, tínhamos 20.761 alunos desistentes e 155.377 alunos não desistentes. Após o *Undersampling*, obtivemos uma base ajustada com 20.611 registros de alunos não desistentes e 20.761 registros de alunos desistentes, proporcionando um conjunto de dados mais equilibrado com 41.372 alunos.

Esse processo foi repetido em todas as modalidades e categorias administrativas, fornecendo subconjuntos balanceados para cada modalidade e categoria administrativa. Essa abordagem refinada proporciona uma base sólida para análises subsequentes, garantindo a confiabilidade e a representatividade dos resultados obtidos neste estudo científico.

4.3.3 Etapa 3 - Divisão do *dataset* em *dataset* de treino e *dataset* de teste.

Conforme a literatura recomenda, o *dataset* foi dividido utilizando a metodologia Holdout, reservando 70% dos dados para treino e 30% para teste (HAN; KAMBER; PEI, 2012). A divisão foi realizada por meio da função *RandomSplit* da biblioteca pySpark.

4.4 Fase 4 - Modelagem

A fase de modelagem do processo CRISP-DM desempenha um papel crucial na transformação dos dados preparados em modelos preditivos. Para a modelagem e desenvolvimento do classificador, essencial para atingir os objetivos deste trabalho, foram formuladas as seguintes questões norteadoras:

- 1. Qual modelo de classificação demonstra o melhor desempenho ao prever a evasão estudantil no ensino superior?
- 2. Quais variáveis se destacam como as mais importantes na predição da evasão no ensino superior?
- 3. Existem diferenças significativas na evasão entre escolas públicas e privadas no ensino superior?
- 4. Há variações notáveis na evasão no ensino superior entre modalidade presencial e EAD?

Nesta etapa, foram escolhidos quatro algoritmos distintos para a construção de modelos de classificação: Regressão Logística (LR), Árvore de Decisão (DT), Floresta Aleatória (RF) e Máquina de Vetores de Suporte (SVM). A seleção desses algoritmos foi baseada na sua adequação para lidar com o problema específico em questão e na diversidade que cada um oferece em termos de abordagem e comportamento preditivo.

Para a implementação eficaz desses modelos, optou-se por utilizar a linguagem de programação Python e a biblioteca PySpark, uma ferramenta poderosa para processamento e

análise de dados em larga escala. A escolha do PySpark proporcionou não apenas uma execução eficiente dos algoritmos, mas também a capacidade de escalabilidade para conjuntos de dados substanciais. Os códigos encontram-se no apêndice deste trabalho.

4.4.1 Cross-validation

Um componente fundamental integrado à construção desses modelos foi a aplicação do método de validação cruzada (*Cross-validation*). A validação cruzada é uma técnica essencial para avaliar o desempenho dos modelos em diferentes conjuntos de dados, fornecendo uma estimativa mais robusta de sua capacidade de generalização. Cada algoritmo foi submetido a um processo detalhado de validação cruzada utilizando o algoritmo K-Fold Cross-validation.

É crucial esclarecer que, neste contexto, o Cross-validation não foi utilizado para o ajuste de hiperparâmetros, pois o PySpark já incorpora funcionalidades específicas para otimização desses parâmetros. Em vez disso, o Cross-validation foi empregado para capturar uma média de acurácia, fornecendo uma métrica consolidada do desempenho do modelo. Essa média de acurácia foi instrumental para a avaliação da performance global dos modelos gerados.

Para a análise crítica do desempenho, adotamos uma abordagem de particionamento da base de dados em 70% para treino e 30% para teste. Dessa forma, o modelo foi validado utilizando esses dados de teste, possibilitando uma avaliação mais robusta de sua eficácia em condições mais próximas às da aplicação prática. Essa estratégia permitiu, portanto, uma comparação direta entre a acurácia média proporcionada pelo Cross-validation e o desempenho real do modelo nos dados de teste, possibilitando a conclusão sobre a sua capacidade de se sobressair à média em cenários reais.

Em resumo, a construção dos modelos de classificação incorporou não apenas a escolha criteriosa de algoritmos, mas também a implementação de técnicas avançadas, para assegurar a robustez dos modelos desenvolvidos. O próximo passo crucial será a avaliação rigorosa desses modelos para determinar sua eficácia e utilidade na solução do problema em questão.

5
Resultados

Neste capítulo, são expostos estatísticas descritivas sobre a evasão de alunos em 2016 e os resultados provenientes dos modelos desenvolvidos nos experimentos. Os algoritmos *Logistic Regression* (LR), *Decision Tree* (DT), *Random Forest* (RF), e *Support Vector Machine* (SVM) foram minuciosamente analisados e comparados entre si. As métricas empregadas para essa avaliação incluem verdadeiro positivo (VP), falso positivo (FP), verdadeiro negativo (VN), falso negativo (FN), precisão, sensibilidade (Recall) e acurácia. Essa análise abrangente busca fornecer uma compreensão detalhada do desempenho de cada algoritmo no contexto específico dos experimentos realizados.

5.1 Evasão de Alunos Antes e Após Balanceamento do *Dataset*

Na modalidade EAD e categoria administrativa pública, observamos 20.761 alunos desistentes e 155.377 alunos não desistentes, totalizando 176.138 alunos nessa modalidade. Após a aplicação da técnica de *Undersampling*, a base de dados foi ajustada, resultando em 20.611 registros de alunos não desistentes e 20.761 registros de alunos desistentes, totalizando 41.372 alunos.

Na modalidade EAD e categoria administrativa privada, encontramos 418.333 alunos desistentes e 1.755.842 alunos não desistentes, totalizando 2.174.175 alunos. Após o balanceamento, a base de dados foi reconfigurada, apresentando 417.727 registros de alunos não desistentes e 418.333 registros de alunos desistentes, totalizando 836.060 alunos.

No âmbito presencial e categoria administrativa pública, foram registrados 139.277 alunos desistentes e 2.173.877 alunos não desistentes, totalizando 2.313.154 alunos. Após o balanceamento, a base foi ajustada para 139.250 registros de alunos não desistentes e 139.277 registros de alunos desistentes, totalizando 278.527 alunos.

Por fim, na modalidade presencial e categoria administrativa privada, verificamos 780.063

alunos desistentes e 6.005.692 alunos não desistentes, totalizando 6.785.755 alunos. Após o processo de balanceamento, a base foi reorganizada, apresentando 780.419 registros de alunos não desistentes e 780.063 registros de alunos desistentes, totalizando 1.560.482 alunos.

5.2 Métricas e Resultados brutos

Nas Tabelas 5, 6, 7, 8 estão sendo apresentados os resultados dos modelos construídos de acordo com cada modalidade de ensino e as categorias administrativas. Os melhores resultados para acurácia estão representados na cor azul.

Tabela 5 – Resultados dos Modelos de Classificação para Modalidade EAD Privada

Algoritmo	VP	VN	FP	FN	Acurácia	Precisão	Recall	Especificidade
LR	0.999	0.928	0.071	0.000	**0.964**	0.933	0.999	0.928
DT	0.998	0.930	0.069	0.001	**0.965**	0.935	0.998	0.930
RF	0.973	0.932	0.067	0.026	**0.952**	0.935	0.973	0.932
SVM	0.999	0.859	0.140	0.000	**0.929**	0.877	0.999	0.859

Tabela 6 – Resultados dos Modelos de Classificação para Modalidade EAD Pública

Algoritmo	VP	VN	FP	FN	Acurácia	Precisão	Recall	Especificidade
LR	0.954	0.938	0.061	0.045	**0.946**	0.939	0.954	0.938
DT	0.999	0.959	0.040	0.000	**0.979**	0.961	0.999	0.959
RF	0.990	0.959	0.040	0.009	**0.975**	0.961	0.990	0.959
SVM	0.917	0.906	0.093	0.082	**0.912**	0.907	0.917	0.906

Tabela 7 – Resultados dos Modelos de Classificação para Modalidade Presencial Privada

Algoritmo	VP	VN	FP	FN	Acurácia	Precisão	Recall	Especificidade
LR	0.992	0.913	0.086	0.007	**0.952**	0.919	0.992	0.913
DT	0.992	0.917	0.082	0.007	**0.954**	0.923	0.992	0.917
RF	0.951	0.922	0.077	0.048	**0.936**	0.924	0.951	0.922
SVM	0.992	0.841	0.158	0.007	**0.917**	0.862	0.992	0.841

Tabela 8 – Resultados dos Modelos de Classificação para Modalidade Presencial Pública

Algoritmo	VP	VN	FP	FN	Acurácia	Precisão	Recall	Especificidade
LR	0.983	0.931	0.068	0.016	**0.957**	0.935	0.983	0.931
DT	0.986	0.941	0.058	0.013	**0.963**	0.943	0.986	0.941
RF	0.975	0.938	0.061	0.024	**0.957**	0.941	0.975	0.938
SVM	0.984	0.889	0.110	0.015	**0.936**	0.899	0.984	0.889

5.3 Análise Estatística dos Resultados

A fase de avaliação dos modelos desenvolvidos desempenha um papel crucial na determinação de sua eficácia e na comparação de seus desempenhos. Dentro desse contexto,

foram realizados testes de hipótese, utilizando o teste t de Student pareado, para comparar as acurácias obtidas pelos modelos LR, DT, RF e SVM. Importante ressaltar que esse procedimento foi repetido em dez iterações, visando assegurar robustez nas conclusões. Essa abordagem foi aplicada de forma par a par para todos os algoritmos, proporcionando uma análise comparativa detalhada do desempenho relativo de cada modelo no âmbito do problema em estudo.

5.3.1 Teste de Hipótese: Comparação de Modelos

Além da comparação específica entre os modelos *Random Forest* (RF) e Máquina de Vetores de Suporte (SVM), o teste de hipótese foi aplicado de forma análoga para comparar todas as combinações de modelos desenvolvidos, mas usamos esse par de modelos para elucidar melhor os testes realizados. Este procedimento proporcionou uma avaliação abrangente da eficácia relativa de cada modelo no contexto do problema investigado.

- **Parâmetros do Teste:**
  - Amostras Independentes: Os conjuntos de dados utilizados para treinar e avaliar os modelos *Random Forest* e SVM são independentes, garantindo que as observações em um conjunto não influenciem as do outro.
  - Métrica de Avaliação: A métrica escolhida para avaliação de desempenho foi a acurácia. A acurácia é uma medida abrangente que expressa a proporção de predições corretas em relação ao total de observações.
  - Número de Iterações: O teste de hipótese foi repetido dez vezes, utilizando diferentes conjuntos de treino e teste a cada iteração, a fim de fornecer uma análise mais abrangente e robusta.
  - Teste Estatístico Utilizado: O teste t de Student pareado foi empregado para determinar se existe uma diferença significativa nas acurácias entre os modelos RF e SVM. Esse teste é apropriado para comparar as médias de duas amostras relacionadas.
  - Nível de Significância: O nível de significância foi fixado em 0.05. Este valor é comumente utilizado para determinar se a diferença observada é estatisticamente significativa.
- **Resultados obtidos:**
  - Observações RF: As acurácias obtidas em cada uma das dez iterações foram registradas como observações para o modelo *Random Forest*.
  - Observações SVM: De maneira análoga, as acurácias correspondentes ao modelo de Máquina de Vetores de Suporte foram observadas em cada uma das dez iterações.
- **Teste Estatístico:**

  - Interpretação dos Resultados:
    - Hipótese Nula (H0): "Não há diferença significativa no desempenho entre os modelos (ACURÁCIA)."
    - Hipótese Alternativa (H1): "Há diferença significativa no desempenho entre os modelos (ACURÁCIA)."
  - Conclusão do Teste: Com base no valor-p obtido, se o valor-p for menor que Alpha, rejeitamos a hipótese nula em favor da hipótese alternativa.

- **Considerações Finais sobre o teste:**

  A análise estatística comparativa entre RF e SVM contribuiu significativamente para a compreensão da eficácia relativa desses modelos no contexto específico do problema em estudo. Essa metodologia de avaliação rigorosa e abrangente foi aplicada de maneira consistente a todos os modelos desenvolvidos, incluindo Regressão Logística, Árvore de Decisão, Floresta Aleatória e SVM. Cada iteração do teste de hipótese abordou a comparação entre dois modelos distintos, proporcionando uma visão detalhada das diferenças em termos de acurácia.

  É relevante ressaltar que, em todos os casos, a hipótese nula foi rejeitada. Em outras palavras, foi identificada uma diferença estatisticamente significativa no desempenho entre os modelos, conforme expresso pela métrica de acurácia. A hipótese nula, que afirmava "Não há diferença significativa no desempenho entre os modelos (ACURÁCIA)", foi descartada em todas as comparações realizadas.

  Esses resultados são cruciais para orientar a seleção final do modelo a ser empregado na solução prática do problema. A rejeição consistente da hipótese nula sugere que, em relação à acurácia, os modelos avaliados apresentaram desempenhos distintos e que há uma base estatística para essa diferenciação. Essa conclusão robusta, fundamentada em evidências estatísticas, fornece uma base sólida e confiável para a tomada de decisão na escolha do modelo mais adequado para a resolução do problema em questão.

5.4 Interpretação dos Resultados

5.4.1 Tendências

Na análise dos resultados dos modelos de classificação para diferentes modalidades de ensino e categorias administrativas, observamos distintos padrões e tendências.

5.4.1.1 Modalidade EAD Privada

Os resultados para a modalidade EAD privada indicam desempenho notável em todos os modelos. A Regressão Logística (LR) atingiu uma acurácia de 96.4%, com elevada precisão de

93.3%, excelente recall de 99.9%, e notável equilíbrio entre precisão e recall. A *Decision Tree* (DT) obteve uma acurácia ligeiramente superior a 96.5%, com destaque para precisão, recall, e especificidade elevados. O *Random Forest* (RF) apresentou consistência com uma acurácia de 95.2%, mantendo notáveis valores de precisão, recall e especificidade. Por fim, a *Support Vector Machine* (SVM) teve uma acurácia de 92.9%, evidenciando um desafio relativo ao balanceamento de precisão e recall, mas com considerável especificidade de 85.9%.

5.4.1.2 Modalidade EAD Pública

Os modelos para a modalidade EAD pública também apresentaram resultados sólidos. Destaca-se a *Decision Tree* (DT) com uma acurácia impressionante de 97.9%, evidenciando maior destaque para precisão e recall. O *Random Forest* (RF) obteve uma acurácia elevada de 97.5%, com desempenho consistente em precisão e recall. A Regressão Logística (LR) demonstrou uma acurácia sólida de 94.6%, equilibrando precisão, recall e especificidade. A *Support Vector Machine* (SVM) teve uma acurácia de 91.2%, evidenciando um equilíbrio entre precisão e recall, com uma importante especificidade de 90.6%.

5.4.1.3 Modalidade Presencial Privada

Analogamente às modalidades EAD, os modelos para a modalidade presencial privada também apresentaram desempenho consistente. A Regressão Logística (LR) obteve uma acurácia robusta de 95.2%, com destaque para recall e especificidade. A *Decision Tree* (DT) manteve uma acurácia sólida de 95.4%, com desempenho consistente em precisão, recall e especificidade. O *Random Forest* (RF) alcançou uma acurácia de 93.6%, mantendo um bom equilíbrio entre precisão e recall, com destaque para especificidade. A *Support Vector Machine* (SVM) teve uma acurácia de 91.7%, destacando um compromisso entre precisão e recall, com um notável recall de 99.2%.

5.4.1.4 Modalidade Presencial Pública

Assim como nas modalidades EAD, os modelos para a modalidade presencial pública também apresentaram resultados consistentes. A Regressão Logística (LR) alcançou uma acurácia elevada de 95.7%, com desempenho sólido em precisão, recall e especificidade. A *Decision Tree* (DT) obteve uma acurácia destacada de 96.3%, com maior ênfase em precisão e recall. O *Random Forest* (RF) apresentou uma acurácia consistente de 95.7%, mantendo equilíbrio entre precisão, recall e especificidade. A *Support Vector Machine* (SVM) teve uma acurácia de 93.6%, evidenciando um compromisso entre precisão e recall, com boa especificidade de 88.9%.

5.4.2 Discussão Geral dos Resultados

Ao analisar os resultados obtidos, destaca-se que a DT demonstrou consistentemente a melhor acurácia em todas as modalidades de ensino, evidenciando sua eficácia em diferentes

contextos. Esta constatação sugere que a árvore de decisão é um modelo robusto e confiável para a tarefa de classificação nas condições investigadas.

Por outro lado, o SVM apresentou a menor acurácia em todas as modalidades, indicando que, em comparação com os outros modelos, ele teve um desempenho inferior na tarefa de classificação. Esta observação é crucial para a compreensão das limitações do SVM no contexto específico do problema em estudo.

A significância dessas diferenças foi confirmada por meio de testes de hipótese estatística. O teste t de Student pareado foi aplicado para comparar as acurácias entre os modelos em cada modalidade. Os resultados estatísticos proporcionaram evidências confiáveis para rejeitar a hipótese nula de que não há diferença significativa no desempenho entre esses modelos. Assim, podemos afirmar com segurança que as disparidades nas acurácias são estatisticamente significativas.

Essa análise reforça a importância da escolha do modelo na solução prática do problema. A consistência da DT em superar os demais modelos em acurácia sugere que este pode ser o modelo preferencial para a aplicação específica considerada. Por outro lado, as limitações do SVM, evidenciadas estatisticamente, fornecem insights valiosos para o refinamento da escolha do modelo, considerando as características únicas do problema em questão.

Esses resultados são cruciais para uma abordagem informada na seleção do modelo mais adequado, proporcionando uma base sólida e confiável para a tomada de decisão na resolução do problema.

5.5 Importância dos Atributos

A compreensão da contribuição individual de cada atributo em um modelo preditivo é crucial para interpretar seu funcionamento e identificar os fatores mais significativos na tomada de decisão. Nesta seção, exploraremos a importância dos atributos nos modelos desenvolvidos, destacando visualmente as influências de cada variável por meio de gráficos de barras horizontais.

Cada gráfico apresentado reflete as importâncias ou coeficientes atribuídos a diferentes atributos do modelo, revelando a magnitude e a direção da influência de cada variável no resultado final. A análise foi direcionada às 10 variáveis que mais impactam positiva ou negativamente na previsão da evasão estudantil. Para algoritmos que utilizam coeficientes, é possível observar uma distinção clara: as barras verdes indicam as variáveis que impactam positivamente, ou seja, influenciam na desistência do aluno, enquanto as barras vermelhas representam as variáveis que impactam negativamente, ou seja, influenciam na não desistência do aluno.

Vale ressaltar que quando nos referimos a "impacto positivo" e "impacto negativo", não estamos sugerindo que a desistência do aluno seja algo positivo. Na verdade, essa categorização é estabelecida em relação à label de desistência, em que o valor igual a 1 é considerado "positivo".

Dessa forma, as variáveis que impactam positivamente estão associadas à influência na ocorrência da desistência, enquanto as que impactam negativamente estão relacionadas à não-desistência.

A seguir, serão apresentados os gráficos correspondentes a diferentes modelos, apresentando graficamente as variáveis mais relevantes em cada caso, e destacando o atributo mais importante de cada modelo.

5.5.1 EAD Privada

Figura 11 – Modelo LR - EAD Privada

Variável mais importante: **IN_CONCLUINTE**, essa variável informa se o aluno está no último período para conclusão do curso.

Figura 12 – Modelo DT - EAD Privada

Variável mais importante: **CO_ALUNO_SITUACAO**, esse atributo identifica a situação de vínculo do aluno no curso: Cursando, Matrícula trancada, Desvinculado do curso, Transferido para outro curso da mesma IES, Formado ou Falecido.

Figura 13 – Modelo RF - EAD Privada

Variável mais importante: **CO_ALUNO_SITUACAO**, esse atributo identifica a situação de vínculo do aluno no curso: Cursando, Matrícula trancada, Desvinculado do curso, Transferido para outro curso da mesma IES, Formado ou Falecido.

Figura 14 – Modelo SVM - EAD Privada

Variável mais importante: **IN_CONCLUINTE**, essa variável informa se o aluno está no último período para conclusão do curso.

5.5.2 EAD Pública

Figura 15 – Modelo LR - EAD Pública

Variável mais importante: **IN_CONCLUINTE**, essa variável informa se o aluno está no último período para conclusão do curso.

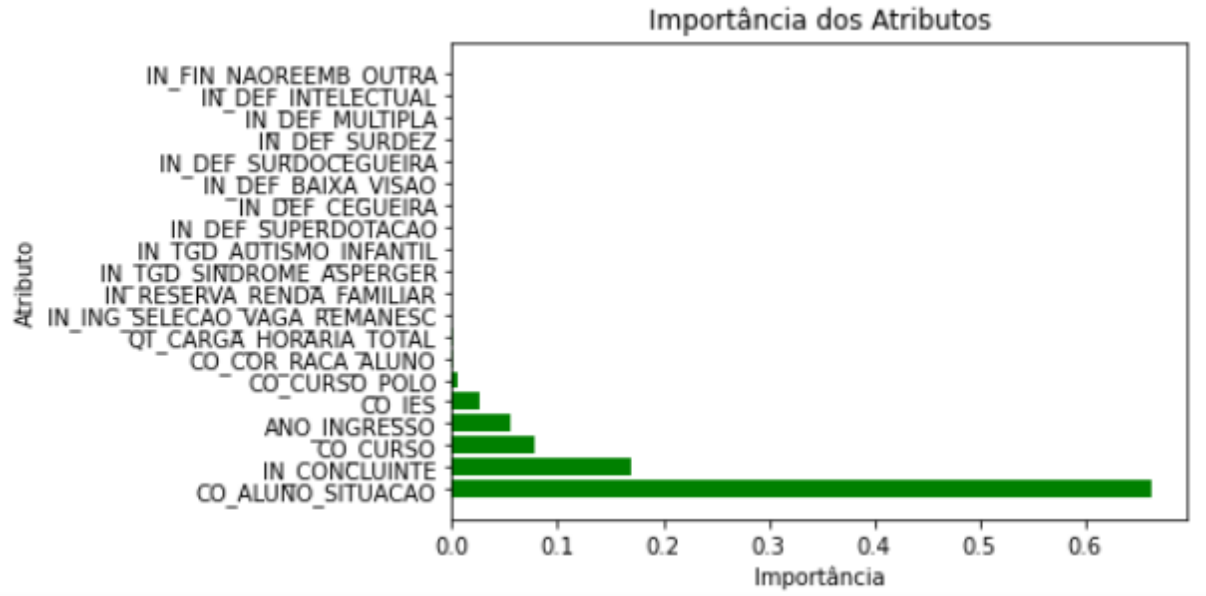

Figura 16 – Modelo DT - EAD Pública

Variável mais importante: **CO_ALUNO_SITUACAO**, esse atributo identifica a situação de vínculo do aluno no curso: Cursando, Matrícula trancada, Desvinculado do curso, Transferido para outro curso da mesma IES, Formado ou Falecido.

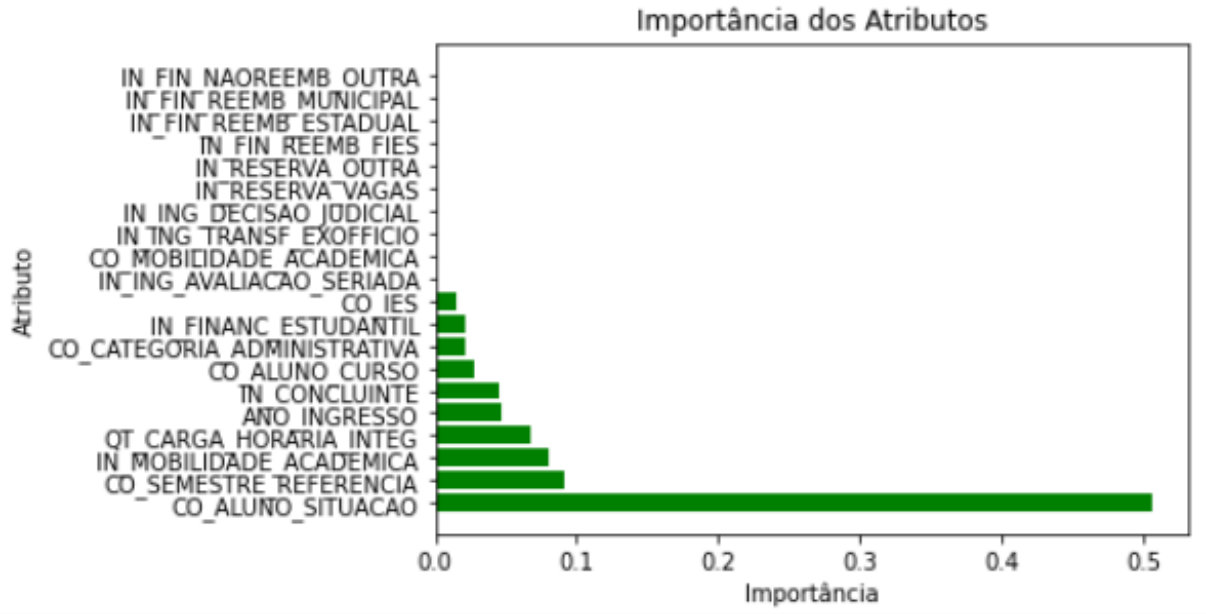

Figura 17 – Modelo RF - EAD Pública

Variável mais importante: **CO_ALUNO_SITUACAO**, esse atributo identifica a situação de vínculo do aluno no curso: Cursando, Matrícula trancada, Desvinculado do curso, Transferido para outro curso da mesma IES, Formado ou Falecido.

Figura 18 – Modelo SVM - EAD Pública

Variável mais importante: **IN_CONCLUINTE**, essa variável informa se o aluno está no último período para conclusão do curso.

5.5.3 Presencial Privada

Figura 19 – Modelo LR - Presencial Privada

Variável mais importante: **IN_CONCLUINTE**, essa variável informa se o aluno está no último período para conclusão do curso.

Figura 20 – Modelo DT - Presencial Privada

Variável mais importante: **CO_ALUNO_SITUACAO**, esse atributo identifica a situação de vínculo do aluno no curso: Cursando, Matrícula trancada, Desvinculado do curso, Transferido para outro curso da mesma IES, Formado ou Falecido.

Figura 21 – Modelo RF - Presencial Privada

Variável mais importante: **CO_ALUNO_SITUACAO**, esse atributo identifica a situação de vínculo do aluno no curso: Cursando, Matrícula trancada, Desvinculado do curso, Transferido para outro curso da mesma IES, Formado ou Falecido.

Figura 22 – Modelo SVM - Presencial Privada

Variável mais importante: **IN_CONCLUINTE**, essa variável informa se o aluno está no último período para conclusão do curso.

5.5.4 Presencial Pública

Figura 23 – Modelo LR - Presencial Pública

Variável mais importante: **IN_CONCLUINTE**, essa variável informa se o aluno está no último período para conclusão do curso.

Figura 24 – Modelo DT - Presencial Pública

Variável mais importante: **CO_ALUNO_SITUACAO**, esse atributo identifica a situação de vínculo do aluno no curso: Cursando, Matrícula trancada, Desvinculado do curso, Transferido para outro curso da mesma IES, Formado ou Falecido.

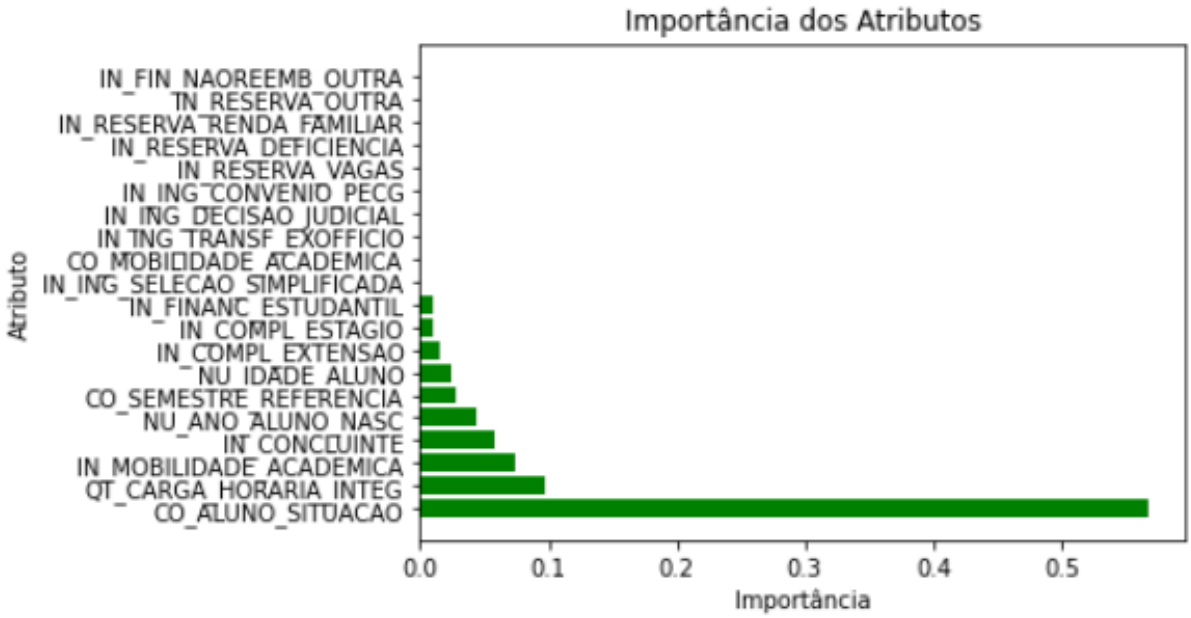

Figura 25 – Modelo RF - Presencial Pública

Variável mais importante: **CO_ALUNO_SITUACAO**, esse atributo identifica a situação de vínculo do aluno no curso: Cursando, Matrícula trancada, Desvinculado do curso, Transferido para outro curso da mesma IES, Formado ou Falecido.

Figura 26 – Modelo SVM - Presencial Pública

Variável mais importante: **IN_CONCLUINTE**, essa variável informa se o aluno está no último período para conclusão do curso.

5.5.5 Síntese da Importância dos Atributos

Numa síntese abrangente, independentemente da modalidade (EAD ou presencial) e das categorias administrativas (pública ou privada), os modelos LR, DT, RF e SVM revelaram um padrão consistente em relação às variáveis mais influentes na previsão da evasão estudantil.

Para LR e SVM, a variável mais crucial em todos os casos foi **IN_CONCLUINTE**, indicando se o aluno está no último período para a conclusão do curso. Essa variável demonstrou ser um fator significativo, destacando-se como um indicativo forte para a avaliação do risco de evasão, tanto em ambientes EAD quanto presenciais, e em instituições públicas e privadas.

Já para RF e DT, a variável preponderante em todas as situações foi **CO_ALUNO_SI -TUACAO**, que identifica a situação de vínculo do aluno no curso. Seja cursando, com matrícula trancada, desvinculado do curso, transferido para outro curso da mesma instituição, formado ou falecido, essa variável se mostrou essencial para compreender e antecipar a evasão estudantil.

A identificação do aluno como concluinte é significativa porque indica que ele está em uma fase crítica do curso, onde poderia enfrentar desafios específicos relacionados à conclusão dos estudos, como a pressão para finalizar o trabalho de conclusão, lidar com a ansiedade em relação ao ingresso no mercado de trabalho, entre outros. Por outro lado, a situação de vínculo do aluno abrange diversas categorias, como cursando, com matrícula trancada, desvinculado do curso, transferido para outro curso da mesma instituição, formado ou falecido. Conhecer essas diferentes situações e comparar com históricos de outros alunos para identificar padrões de

evasão em diferentes situações de vínculo, é significativamente importante na predição da evasão do aluno.

Portanto, saber se o aluno está no último período para a conclusão do curso ou qual é sua situação de vínculo é crucial para antecipar possíveis casos de evasão e implementar estratégias de retenção e suporte adequadas, visando melhorar a experiência do aluno e aumentar as taxas de conclusão do curso.

Esses resultados consistentes, independentemente das especificidades da modalidade de ensino e da categoria administrativa da instituição, destacam a importância dessas variáveis-chave na tomada de decisões relacionadas à evasão estudantil, fornecendo uma base sólida para estratégias de retenção e suporte ao estudante.

Destaca-se que, embora estas variáveis tenham sido identificadas como as mais importantes, conforme evidenciado pelos gráficos, há diversas outras variáveis igualmente relevantes para a análise da evasão estudantil. O destaque para **IN_CONCLUINTE** e **CO_ALUNO_SITUACAO** é devido à sua posição proeminente no contexto geral, mas a compreensão abrangente da evasão requer consideração cuidadosa de múltiplos fatores que contribuem para a complexidade desse fenômeno.

5.6 Comparativo de Evasão entre as Modalidades e Categorias Administrativas

A análise comparativa de evasão entre as modalidades EAD e Presencial, considerando também as categorias administrativas de instituições públicas e privadas, oferece insights cruciais sobre os padrões de desistência no ensino superior. Segue na Tabela 9 informações levantadas a partir do censo de 2016 sobre evasão para cada caso.

Tabela 9 – Comparativo de Evasão

Modalidade	Instituição	Total de Alunos	Desistentes	Proporção de Evasão (%)
EAD	Pública	176.138	20.761	11,8
	Privada	2.174.175	418.333	19,3
Presencial	Pública	2.313.154	139.277	6,0
	Privada	6.785.755	780.063	11,5

5.6.1 Análise Comparativa: Entre Instituições de Ensino Superior Públicas e Privadas

Observa-se uma tendência clara de maior evasão em instituições privadas de ensino superior. Ao examinar as proporções, verifica-se que a evasão nessas instituições é significativamente mais alta do que em instituições públicas, tanto na modalidade presencial quanto na EAD.

Especificamente, na modalidade EAD, a proporção de alunos desistentes em instituições privadas (19,3%) supera substancialmente a proporção em instituições públicas (11,8%), indicando uma disparidade marcante.

5.6.2 Análise Comparativa: Entre Modalidades Presencial e EAD

A análise das proporções revela que a modalidade presencial, quer seja em instituições públicas ou privadas, apresenta cifras notavelmente menores de alunos desistentes em comparação com a modalidade EAD. No contexto das instituições públicas, a evasão na modalidade presencial (6,0%) é consideravelmente inferior à evasão na modalidade EAD (11,8%). Da mesma forma, nas instituições privadas, a proporção de alunos desistentes na modalidade presencial (11,5%) é substancialmente menor do que na modalidade EAD (19,3%).

Esses dados quantitativos evidenciam diferenças expressivas na evasão entre instituições públicas e privadas de ensino superior, bem como variações consideráveis entre as modalidades presencial e EAD. Essa análise numérica ressalta a importância de considerar esses fatores ao implementar estratégias de retenção e apoio aos estudantes, proporcionando uma visão mais precisa e fundamentada na formulação de políticas educacionais.

5.7 Respostas às Hipóteses Iniciais

Esta seção apresenta as conclusões e descobertas resultantes da avaliação de duas hipóteses centrais no contexto da predição de evasão estudantil no ensino superior, citadas na seção 1.4.

A primeira hipótese (H1) indaga sobre a possibilidade de construir classificadores eficientes capazes de predizer a evasão de alunos com desempenho significativo.

A segunda hipótese (H2) direciona nosso olhar para a extensibilidade desses modelos.

Ao abordar essas hipóteses, buscamos não apenas responder a perguntas específicas, mas também contribuir para a compreensão mais ampla da utilidade e potencial desses modelos na previsão da evasão estudantil, impulsionando assim estratégias eficazes de retenção e aprimoramento na esfera do ensino superior.

5.7.1 H1. Classificadores são capazes de predizer a evasão de alunos com bom desempenho.

A análise dos resultados obtidos revelou uma resposta positiva à primeira hipótese. A construção dos classificadores demonstrou consistência e eficácia, uma vez que, em todas as modalidades e categorias administrativas avaliadas, alguns modelos alcançaram uma acurácia superior a 95%. Essa elevada assertividade sugere que o modelo desenvolvido possui uma

capacidade significativa de predizer a evasão de alunos, indicando seu potencial como uma ferramenta robusta para identificar padrões e fatores associados à evasão no contexto do ensino superior.

5.7.2 H2. Modelos de classificadores podem ser reutilizados para diversas instituições.

A segunda hipótese também recebeu uma resposta positiva com base na metodologia empregada. O modelo de classificação foi construído utilizando dados provenientes do censo de educação superior, proporcionando uma base padronizada e abrangente que reflete a realidade das instituições de ensino superior brasileiras. Essa abordagem permite a generalização e reutilização do modelo em diferentes instituições, uma vez que o uso de dados do censo estabelece uma base comum entre as entidades educacionais. Assim, os resultados obtidos indicam que o modelo pode ser aplicado de forma consistente em diversas instituições, contribuindo para uma abordagem mais ampla e abrangente na previsão da evasão estudantil.

6

Considerações Finais

O presente trabalho representou uma exploração abrangente no âmbito da evasão estudantil, dividido em duas frentes distintas: a análise detalhada de informações relacionadas ao tema e a implementação de modelos de classificação para predição desse fenômeno complexo. O mapeamento sistemático, que serviu como ponto de partida, ofereceu uma visão aprofundada do estado atual da pesquisa nesse campo crucial da educação.

Na abordagem quantitativa e qualitativa adotada, identificamos lacunas significativas, especialmente na falta de padronização dos atributos considerados em estudos, o que ressalta a necessidade de uma abordagem mais uniforme para tornar as análises mais generalizáveis e aplicáveis em diversos contextos educacionais.

No âmbito da implementação de modelos de classificação, exploramos algoritmos como Regressão Logística, Decision Tree, Random Forest e Support Vector Machine. Os resultados obtidos, minuciosamente apresentados e discutidos no capítulo de resultados, lançaram luz sobre o desempenho variado desses modelos em diferentes modalidades de ensino. A aplicação de testes de hipótese proporcionou uma análise estatística robusta, evidenciando diferenças entre os modelos e contribuindo para uma compreensão mais profunda dos resultados. Destaca-se que a Decision Tree (DT) demonstrou consistentemente a mais alta taxa de acurácia em todos os casos, reforçando sua eficácia na previsão da evasão.

As variáveis **IN_CONCLUINTE** e **CO_ALUNO_SITUACAO** emergiram como as mais importantes em todos os casos, evidenciando a influência crucial desses fatores na predição da evasão estudantil. Ressalta-se que **IN_CONCLUINTE** foi mais relevante nos modelos de Regressão Logística e Support Vector Machine (SVM), enquanto **CO_ALUNO_SITUACAO** se destacou nos modelos de Random Forest (RF) e Decision Tree (DT).

As diferenças notáveis nas taxas de evasão entre modalidades e categorias administrativas sublinham a importância de abordagens específicas ao contexto na formulação de estratégias de retenção.

O objetivo inicial de investigar a evasão e analisar a classificação de evasão utilizando algoritmos clássicos foi satisfatoriamente alcançado. Contudo, reconhecemos que há muito a ser explorado, especialmente em níveis de ensino fundamental e médio. A transformação deste projeto em uma dissertação completa representa um compromisso renovado com a resolução contínua do desafiador problema da evasão estudantil. Ao olhar para o futuro, seria promissor realizar uma investigação de novas técnicas e algoritmos avançados para a análise de evasão estudantil. Considerar, também, a possibilidade de realizar experimentos utilizando dados mais recentes e abrangentes, como os censos educacionais mais recentes. Além disso, destacamos a importância de nossa pesquisa na orientação de estratégias educacionais mais informadas e eficazes, visando aprimorar a retenção de alunos e, consequentemente, o sucesso educacional.

Embora tenha alcançado os objetivos propostos, este trabalho enfrentou algumas limitações. Por exemplo, nos Censos anteriores a 2016, o INEP não mantém a mesma chave primária do aluno para o ano subsequente, dificultando análises longitudinais. Além disso, a partir de 2018, devido à implementação da Lei Geral de Proteção de Dados Pessoais (LGPD), houve uma redução nos atributos disponibilizados nos Censos, o que inviabilizou a investigação para outros anos. Essas limitações ressaltam a necessidade de considerar as mudanças nas políticas de proteção de dados ao realizar pesquisas ao longo do tempo, destacando os desafios enfrentados devido às restrições impostas pela LGPD.

Referências

AGUIRRE, C. E.; PÉREZ, J. C. Predictive data analysis techniques applied to dropping out of university studies. In: IEEE. *2020 XLVI Latin American Computing Conference (CLEI)*. [S.l.], 2020. p. 512–521. Citado 4 vezes nas páginas 34, 41, 43 e 58.

ALBAN, M. S.; MAURICIO, D. Prediction of university dropout through technological factors: a case study in ecuador. *Revista Espacios*, v. 39, n. 52, 2018. Citado 3 vezes nas páginas 34, 43 e 52.

ALBAN, M. S.; MAURICIO, D. Prediction of university dropout through technological factors: a case study in ecuador. *Revista Espacios*, v. 39, n. 52, 2018. Citado na página 61.

AMERI, S. et al. Survival analysis based framework for early prediction of student dropouts. In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. [S.l.: s.n.], 2016. p. 903–912. Citado 2 vezes nas páginas 34 e 48.

ANGUITA, D. et al. The'k'in k-fold cross validation. In: *ESANN*. [S.l.: s.n.], 2012. p. 441–446. Citado na página 21.

BAGGI, C. A. D. S.; LOPES, D. A. Evasão e avaliação institucional no ensino superior: uma discussão bibliográfica. *Avaliação: Revista da Avaliação da Educação Superior (Campinas)*, SciELO Brasil, v. 16, p. 355–374, 2011. Citado 2 vezes nas páginas 13 e 28.

BAKER, R.; ISOTANI, S.; CARVALHO, A. Mineraçao de dados educacionais: Oportunidades para o brasil. *Revista Brasileira de Informática na Educação*, v. 19, n. 02, p. 03, 2011. Citado na página 25.

BARANYI, M.; NAGY, M.; MOLONTAY, R. Interpretable deep learning for university dropout prediction. In: *Proceedings of the 21st Annual Conference on Information Technology Education*. [S.l.: s.n.], 2020. p. 13–19. Citado 2 vezes nas páginas 34 e 48.

BATISTELA, G. C.; RODRIGUES, S. A.; BONONI, J. T. C. M. Estudo sobre a evasão escolar usando regressão logística: análise dos alunos do curso de administração da fundação educacional de ituverava. *Tekhne e Logos*, v. 1, n. 1, p. 103–120, 2009. Citado na página 26.

BELLO, F. A. et al. Using machine learning methods to identify significant variables for the prediction of first-year informatics engineering students dropout. In: IEEE. *2020 39th International Conference of the Chilean Computer Science Society (SCCC)*. [S.l.], 2020. p. 1–5. Citado 3 vezes nas páginas 35, 46 e 53.

BERENS, J. et al. Early detection of students at risk–predicting student dropouts using administrative student data and machine learning methods. CESifo Working Paper, 2018. Citado 2 vezes nas páginas 35 e 58.

BONIFRO, F. D. et al. Student dropout prediction. In: SPRINGER. *International Conference on Artificial Intelligence in Education*. [S.l.], 2020. p. 129–140. Citado 2 vezes nas páginas 34 e 52.

BURGOS, C. et al. Data mining for modeling students' performance: A tutoring action plan to prevent academic dropout. *Computers & Electrical Engineering*, Elsevier, v. 66, p. 541–556, 2018. Citado 2 vezes nas páginas 35 e 54.

CHEN, Y.; JOHRI, A.; RANGWALA, H. Running out of stem: a comparative study across stem majors of college students at-risk of dropping out early. In: *Proceedings of the 8th international conference on learning analytics and knowledge*. [S.l.: s.n.], 2018. p. 270–279. Citado 2 vezes nas páginas 34 e 48.

CHUNG, J. Y.; LEE, S. Dropout early warning systems for high school students using machine learning. *Children and Youth Services Review*, Elsevier, v. 96, p. 346–353, 2019. Citado 2 vezes nas páginas 34 e 49.

COLPO, M. P. et al. Mineraçao de dados educacionais na previsao de evasao: uma rsl sob a perspectiva do congresso brasileiro de informática na educaçao. In: SBC. *Anais do XXXI Simpósio Brasileiro de Informática na Educação*. [S.l.], 2020. p. 1102–1111. Citado na página 29.

COSTA, A. G. et al. Prediction analysis of student dropout in a computer science course using educational data mining. In: IEEE. *2020 XV Conferencia Latinoamericana de Tecnologias de Aprendizaje (LACLO)*. [S.l.], 2020. p. 1–6. Citado 4 vezes nas páginas 34, 40, 45 e 51.

COSTA, A. G. et al. Prediction analysis of student dropout in a computer science course using educational data mining. In: IEEE. *2020 XV Conferencia Latinoamericana de Tecnologias de Aprendizaje (LACLO)*. [S.l.], 2020. p. 1–6. Citado na página 60.

COSTA, E. et al. Mineração de dados educacionais: conceitos, técnicas, ferramentas e aplicações. *Jornada de Atualização em Informática na Educação*, v. 1, n. 1, p. 1–29, 2013. Citado 3 vezes nas páginas 14, 24 e 26.

COUSSEMENT, K. et al. Predicting student dropout in subscription-based online learning environments: The beneficial impact of the logit leaf model. *Decision Support Systems*, Elsevier, v. 135, p. 113325, 2020. Citado 2 vezes nas páginas 35 e 55.

CUTLER, A.; CUTLER, D. R.; STEVENS, J. R. Random forests. *Ensemble machine learning: Methods and applications*, Springer, p. 157–175, 2012. Citado na página 26.

DEWAN, M. A. A. et al. Predicting dropout-prone students in e-learning education system. In: IEEE. *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*. [S.l.], 2015. p. 1735–1740. Citado 2 vezes nas páginas 34 e 50.

FEI, M.; YEUNG, D.-Y. Temporal models for predicting student dropout in massive open online courses. In: IEEE. *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*. [S.l.], 2015. p. 256–263. Citado 2 vezes nas páginas 35 e 56.

FIGUEROA-CAÑAS, J.; S.VINUESA, T. Early prediction of dropout and final exam performance in an online statistics course. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, IEEE, v. 15, n. 2, p. 86–94, 2020. Citado 3 vezes nas páginas 35, 46 e 55.

FILHO, R. L. L. S. et al. A evasão no ensino superior brasileiro. *Cadernos de pesquisa*, SciELO Brasil, v. 37, p. 641–659, 2007. Citado 2 vezes nas páginas 18 e 20.

FRITSCH, R.; ROCHA, C. S. da; VITELLI, R. F. A evasão nos cursos de graduação em uma instituição de ensino superior privada. *Revista Educação em Questão*, v. 52, n. 38, p. 81–108, 2015. Citado 2 vezes nas páginas 13 e 18.

FU, Q. et al. Clsa: A novel deep learning model for mooc dropout prediction. *Computers & Electrical Engineering*, Elsevier, v. 94, p. 107315, 2021. Citado 2 vezes nas páginas 34 e 49.

GAMAO, A.; GERARDO, B. Prediction-based model for student dropouts using modified mutated firefly algorithm. v. 8, n. 6, p. 3461–3469, 2019. Citado 3 vezes nas páginas 34, 43 e 52.

GARGANTINI, T. *Evasão nas Universidades: A Frustração de Trilhões de Dólares*. 2019. Urlhttps://www.expoensino.com.br/post/evasão-nas-universidades-a-frustração-de-trilhões-de-dólares. Citado na página 19.

GOLDSCHMIDT, R.; PASSOS, E.; BEZERRA, E. *Data Mining*. [S.l.]: Elsevier Brasil, 2015. Citado na página 24.

GUZMÁN-CASTILLO, S. et al. Implementation of a predictive information system for university dropout prevention. *Procedia Computer Science*, Elsevier, v. 198, p. 566–571, 2022. Citado 3 vezes nas páginas 35, 44 e 57.

HAN, J.; KAMBER, M.; PEI, J. Data mining concepts and techniques third edition. *University of Illinois at Urbana-Champaign Micheline Kamber Jian Pei Simon Fraser University*, 2012. Citado na página 64.

HEGDE, V. Dimensionality reduction technique for developing undergraduate student dropout model using principal component analysis through r package. In: IEEE. *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*. [S.l.], 2016. p. 1–6. Citado 3 vezes nas páginas 34, 42 e 58.

HEGDE, V.; PRAGEETH, P. Higher education student dropout prediction and analysis through educational data mining. In: IEEE. *2018 2nd International Conference on Inventive Systems and Control (ICISC)*. [S.l.], 2018. p. 694–699. Citado 5 vezes nas páginas 23, 35, 42, 46 e 55.

HEREDIA, D.; AMAYA, Y.; BARRIENTOS, E. Student dropout predictive model using data mining techniques. *IEEE Latin America Transactions*, IEEE, v. 13, n. 9, p. 3127–3134, 2015. Citado 4 vezes nas páginas 35, 41, 47 e 56.

HOSSAIN, M. et al. Predictive analysis on university dropout rate of bangladesh in covid-19. In: IEEE. *2022 International Conference on Innovations in Science, Engineering and Technology (ICISET)*. [S.l.], 2022. p. 439–444. Citado 2 vezes nas páginas 35 e 56.

HUTAGAOL, N.; SUHARJITO. Predictive modelling of student dropout using ensemble classifier method in higher education. ASTES Publishers, v. 4, n. 4, p. 206–211, 2019. Citado 3 vezes nas páginas 34, 43 e 52.

JARBOU, M. et al. Deep learning-based school attendance prediction for autistic students. *Scientific Reports*, Nature Publishing Group, v. 12, n. 1, p. 1–11, 2022. Citado 3 vezes nas páginas 35, 47 e 57.

KANG, K.; WANG, S. Analyze and predict student dropout from online programs. In: *Proceedings of the 2nd International Conference on Compute and Data Analysis*. [S.l.: s.n.], 2018. p. 6–12. Citado 2 vezes nas páginas 34 e 48.

KOTSIANTIS, S. B.; PIERRAKEAS, C.; PINTELAS, P. E. Preventing student dropout in distance learning using machine learning techniques. In: SPRINGER. *International conference on knowledge-based and intelligent information and engineering systems*. [S.l.], 2003. p. 267–274. Citado 3 vezes nas páginas 35, 44 e 55.

KUO, J. Y.; PAN, C. W.; LEI, B. Using stacked denoising autoencoder for the student dropout prediction. In: IEEE. *2017 IEEE International Symposium on Multimedia (ISM)*. [S.l.], 2017. p. 483–488. Citado 2 vezes nas páginas 34 e 49.

LI, Y.; CUI, X.; ZHANG, Z. Dropout rate prediction for mooc based on inceptiontime model. In: *Proceedings of the 7th International Conference on Distance Education and Learning*. [S.l.: s.n.], 2022. p. 54–59. Citado 2 vezes nas páginas 35 e 58.

LIMSATHITWONG, K.; TIWATTHANONT, K.; YATSUNGNOEN, T. Dropout prediction system to reduce discontinue study rate of information technology students. In: IEEE. *2018 5th International Conference on Business and Industrial Research (ICBIR)*. [S.l.], 2018. p. 110–114. Citado 4 vezes nas páginas 35, 40, 44 e 54.

LOTTERING, R.; HANS, R.; LALL, M. A machine learning approach to identifying students at risk of dropout: A case study. Citado 4 vezes nas páginas 35, 41, 42 e 53.

LOTTERING, R.; HANS, R.; LALL, M. A model for the identification of students at risk of dropout at a university of technology. In: IEEE. *2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)*. [S.l.], 2020. p. 1–8. Citado 3 vezes nas páginas 34, 42 e 49.

LYKOURENTZOU, I. et al. Dropout prediction in e-learning courses through the combination of machine learning techniques. *Computers & Education*, Elsevier, v. 53, n. 3, p. 950–965, 2009. Citado 2 vezes nas páginas 34 e 50.

MAKSIMOVA, N.; PENTEL, A.; DUNAJEVA, O. Predicting first-year computer science students drop-out with machine learning methods: A case study. In: SPRINGER. *International Conference on Interactive Collaborative Learning*. [S.l.], 2020. p. 719–726. Citado 2 vezes nas páginas 34 e 51.

MANHÃES, L. M. B.; CRUZ, S. M. S. da; ZIMBRÃO, G. Wave: an architecture for predicting dropout in undergraduate courses using edm. In: *Proceedings of the 29th annual acm symposium on applied computing*. [S.l.: s.n.], 2014. p. 243–247. Citado 3 vezes nas páginas 28, 34 e 49.

MANHÃES, L. M. B.; CRUZ, S. M. S. da; ZIMBRÃO, G. Wave: an architecture for predicting dropout in undergraduate courses using edm. In: *Proceedings of the 29th annual acm symposium on applied computing*. [S.l.: s.n.], 2014. p. 243–247. Citado na página 60.

MARTINS, M. P. et al. Previsão do abandono académico numa instituição de ensino superior com recurso a data mining. 2020. Citado 3 vezes nas páginas 34, 43 e 51.

MECA, I. et al. Early warning methodology for dropping out of university degrees. In: *Eighth International Conference on Technological Ecosystems for Enhancing Multiculturality*. [S.l.: s.n.], 2020. p. 245–249. Citado 4 vezes nas páginas 34, 40, 42 e 48.

MNYAWAMI, Y. N.; MAZIKU, H. H.; MUSHI, J. C. Enhanced model for predicting student dropouts in developing countries using automated machine learning approach: A case of tanzanian's secondary schools. *Applied Artificial Intelligence*, Taylor & Francis, v. 36, n. 1, p. 2071406, 2022. Citado 3 vezes nas páginas 35, 47 e 57.

MOHAMMED, R.; RAWASHDEH, J.; ABDULLAH, M. Machine learning with oversampling and undersampling techniques: overview study and experimental results. In: IEEE. *2020 11th international conference on information and communication systems (ICICS)*. [S.l.], 2020. p. 243–248. Citado na página 23.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, Manole, v. 1, n. 1, p. 32, 2003. Citado na página 25.

MORAIS, F. L. de et al. Modelos de regressao aplicados na previsao da evasao escolar do ensino básico: uma revisao sistemática da literatura. *Anais do XXXII Simpósio Brasileiro de Informática na Educação*, SBC, p. 168–178, 2021. Citado na página 59.

MOSELEY, L. G.; MEAD, D. M. Predicting who will drop out of nursing courses: a machine learning exercise. *Nurse education today*, Elsevier, v. 28, n. 4, p. 469–475, 2008. Citado 2 vezes nas páginas 34 e 58.

MUBARAK, A. A.; CAO, H.; HEZAM, I. M. Deep analytic model for student dropout prediction in massive open online courses. *Computers & Electrical Engineering*, Elsevier, v. 93, p. 107271, 2021. Citado 2 vezes nas páginas 34 e 49.

MUBARAK, A. A.; CAO, H.; ZHANG, W. Prediction of students' early dropout based on their interaction logs in online learning environment. *Interactive Learning Environments*, Taylor & Francis, p. 1–20, 2020. Citado 3 vezes nas páginas 34, 45 e 52.

NAGY, M.; MOLONTAY, R. Predicting dropout in higher education based on secondary school performance. In: IEEE. *2018 IEEE 22nd international conference on intelligent engineering systems (INES)*. [S.l.], 2018. p. 000389–000394. Citado 4 vezes nas páginas 22, 34, 43 e 49.

NANDESHWAR, A.; MENZIES, T.; NELSON, A. Learning patterns of university student retention. *Expert Systems with Applications*, Elsevier, v. 38, n. 12, p. 14984–14996, 2011. Citado na página 18.

NASEEM, M.; CHAUDHARY, K.; SHARMA, B. Predicting freshmen attrition in computing science using data mining. *Education and Information Technologies*, Springer, p. 1–31, 2022. Citado 4 vezes nas páginas 35, 41, 45 e 58.

NASEEM, M. et al. Using ensemble decision tree model to predict student dropout in computing science. In: IEEE. *2019 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*. [S.l.], 2019. p. 1–8. Citado 4 vezes nas páginas 35, 40, 44 e 53.

NEUMANN, J.; SCHNÖRR, C.; STEIDL, G. Combined svm-based feature selection and classification. *Machine learning*, Springer, v. 61, p. 129–150, 2005. Citado na página 26.

NIYOGISUBIZO, J. et al. Predicting student's dropout in university classes using two-layer ensemble machine learning approach: A novel stacked generalization. *Computers and Education: Artificial Intelligence*, Elsevier, v. 3, p. 100066, 2022. Citado 3 vezes nas páginas 35, 47 e 56.

NUANKAEW, P. et al. Analysis dropout situation of business computer students at university of phayao. In: SPRINGER. *International Conference on Interactive Collaborative Learning*. [S.l.], 2019. p. 419–432. Citado 2 vezes nas páginas 34 e 50.

NUANMEESRI, S. et al. Improving dropout forecasting during the covid-19 pandemic through feature selection and multilayer perceptron neural network. *International Journal of Information and Education Technology*, v. 12, n. 9, 2022. Citado 3 vezes nas páginas 35, 44 e 57.

OLIVEIRA, J. L. et al. Undergraduate students' effectiveness in an institution with high dropout index. In: IEEE. *2020 IEEE Frontiers in Education Conference (FIE)*. [S.l.], 2020. p. 1–7. Citado na página 24.

PANAGIOTAKOPOULOS, T. et al. Early dropout prediction in moocs through supervised learning and hyperparameter optimization. *Electronics*, Multidisciplinary Digital Publishing Institute, v. 10, n. 14, p. 1701, 2021. Citado 3 vezes nas páginas 35, 46 e 55.

PARDO, T. A. S.; NUNES, M. d. G. V. Aprendizado bayesiano aplicado ao processamento de línguas naturais. *Série de Relatórios Técnicos do Instituto de Ciências Matemáticas e de Computação-ICMC, Universidade de São Paulo*, v. 180, 2002. Citado 2 vezes nas páginas 14 e 25.

PEÑA, D. de la et al. Mining activity grades to model students' performance. In: IEEE. *2017 International Conference on Engineering & MIS (ICEMIS)*. [S.l.], 2017. p. 1–6. Citado 2 vezes nas páginas 35 e 55.

PEREIRA, R. T.; ZAMBRANO, J. C. Application of decision trees for detection of student dropout profiles. In: IEEE. *2017 16th IEEE international conference on machine learning and applications (ICMLA)*. [S.l.], 2017. p. 528–531. Citado 2 vezes nas páginas 35 e 54.

PÉREZ, A. et al. Comparative analysis of prediction techniques to determine student dropout: Logistic regression vs decision trees. In: IEEE. *2018 37th International Conference of the Chilean Computer Science Society (SCCC)*. [S.l.], 2018. p. 1–8. Citado 3 vezes nas páginas 35, 46 e 54.

PEREZ, B.; CASTELLANOS, C.; CORREAL, D. Applying data mining techniques to predict student dropout: a case study. In: IEEE. *2018 IEEE 1st colombian conference on applications in computational intelligence (colcaci)*. [S.l.], 2018. p. 1–6. Citado 3 vezes nas páginas 35, 40 e 54.

PÉREZ-GUTIÉRREZ, B. R. Comparación de técnicas de mineria de datos para identificar indicios de deserción estudiantil, a partir del desempeño académico. *Revista UIS Ingenierías*, v. 19, n. 1, p. 193–204, 2020. Citado 2 vezes nas páginas 35 e 54.

PETERSEN, K. et al. Systematic mapping studies in software engineering. In: *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*. [S.l.: s.n.], 2008. p. 1–10. Citado 2 vezes nas páginas 26 e 30.

PRADEEP, A.; DAS, S.; KIZHEKKETHOTTAM, J. J. Students dropout factor prediction using edm techniques. In: IEEE. *2015 International Conference on Soft-Computing and Networks Security (ICSNS)*. [S.l.], 2015. p. 1–7. Citado 5 vezes nas páginas 13, 18, 35, 46 e 53.

QUISHPE-MORALES, S. et al. Modelo de predicción de la deserción universitaria mediante analítica de datos: Estrategia para la sustentabilidad. *Revista Ibérica de Sistemas e Tecnologias de Informação*, Associação Ibérica de Sistemas e Tecnologias de Informacao, n. E35, p. 38–47, 2020. Citado 3 vezes nas páginas 34, 41 e 51.

RADOVANOVIĆ, S.; DELIBAŠIĆ, B.; SUKNOVIĆ, M. Predicting dropout in online learning environments. *Computer Science and Information Systems*, n. 00, p. 53–53, 2020. Citado 2 vezes nas páginas 34 e 50.

REVATHY, M.; KAMALAKKANNAN, S.; KAVITHA, P. Machine learning based prediction of dropout students from the education university using smote. In: IEEE. *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*. [S.l.], 2022. p. 1750–1758. Citado 3 vezes nas páginas 35, 47 e 56.

ROVIRA, S.; PUERTAS, E.; IGUAL, L. Data-driven system to predict academic grades and dropout. *PLoS one*, Public Library of Science San Francisco, CA USA, v. 12, p. e0171207, 2017. Citado 2 vezes nas páginas 34 e 50.

ŞAHIN, M. A comparative analysis of dropout prediction in massive open online courses. *Arabian Journal for Science and Engineering*, Springer, v. 46, n. 2, p. 1845–1861, 2021. Citado 2 vezes nas páginas 35 e 53.

SALLAN, G.; BEHAL, S. Prediction of student dropout using enhanced machine learning algorithm. Citado 3 vezes nas páginas 34, 41 e 51.

SANTANA, M. A. et al. A predictive model for identifying students with dropout profiles in online courses. In: *EDM (Workshops)*. [S.l.: s.n.], 2015. Citado 3 vezes nas páginas 34, 45 e 50.

SELVAN, M.; NAVADURGA, N.; PRASANNA, N. An efficient model for predicting student dropout using data mining and machine learning techniques. In: . [S.l.: s.n.], 2019. v. 8, p. 750–752. Citado 2 vezes nas páginas 34 e 50.

SHEARER, C. The crisp-dm model: the new blueprint for data mining. *Journal of data warehousing*, THE DATA WAREHOUSE INSTITUTE, v. 5, n. 4, p. 13–22, 2000. Citado na página 62.

SIVAKUMAR, S.; VENKATARAMAN, S.; SELVARAJ, R. Predictive modeling of student dropout indicators in educational data mining using improved decision tree. *Indian Journal of Science and Technology*, Indian Society for Education and Environment, v. 9, n. 4, p. 1–5, 2016. Citado 3 vezes nas páginas 34, 46 e 52.

SORENSEN, L. C. "big data" in educational administration: An application for predicting school dropout risk. *Educational Administration Quarterly*, SAGE Publications Sage CA: Los Angeles, CA, v. 55, n. 3, p. 404–446, 2019. Citado 2 vezes nas páginas 35 e 53.

SU, M. et al. Re-envisioning a k-12 early warning system with school climate factors. In: *Proceedings of the Ninth ACM Conference on Learning@ Scale*. [S.l.: s.n.], 2022. p. 405–408. Citado 2 vezes nas páginas 35 e 59.

TAMADA, M. M.; NETTO, J. F. de M.; LIMA, D. P. R. de. Predicting and reducing dropout in virtual learning using machine learning techniques: A systematic review. In: IEEE. *2019 IEEE Frontiers in Education Conference (FIE)*. [S.l.], 2019. p. 1–9. Citado na página 29.

TAN, M.; SHAO, P. Prediction of student dropout in e-learning program through the use of machine learning method. *International journal of emerging technologies in learning*, v. 10, n. 1, 2015. Citado 2 vezes nas páginas 35 e 55.

TENPIPAT, W.; AKKARAJITSAKUL, K. Student dropout prediction: A kmutt case study. In: IEEE. *2020 1st International Conference on Big Data Analytics and Practices (IBDAP)*. [S.l.], 2020. p. 1–5. Citado 2 vezes nas páginas 34 e 52.

TONTINI, G.; WALTER, S. A. Pode-se identificar a propensão e reduzir a evasão de alunos?: ações estratégicas e resultados táticos para instituições de ensino superior. *Avaliação: Revista da Avaliação da Educação Superior (Campinas)*, SciELO Brasil, v. 19, p. 89–110, 2014. Citado na página 19.

VEGA, H. et al. Intelligent system to predict university students dropout. *International Journal of Online & Biomedical Engineering*, v. 18, n. 7, 2022. Citado 3 vezes nas páginas 35, 44 e 57.

VILORIA, A.; LEZAMA, O. B. P. Mixture structural equation models for classifying university student dropout in latin america. *Procedia Computer Science*, Elsevier, v. 160, p. 629–634, 2019. Citado 2 vezes nas páginas 34 e 58.

WU, N. et al. Clms-net: dropout prediction in moocs with deep learning. In: *Proceedings of the ACM Turing Celebration Conference-China*. [S.l.: s.n.], 2019. p. 1–6. Citado 3 vezes nas páginas 34, 45 e 48.

XING, W.; DU, D. Dropout prediction in moocs: Using deep learning for personalized intervention. *Journal of Educational Computing Research*, SAGE Publications Sage CA: Los Angeles, CA, v. 57, n. 3, p. 547–570, 2019. Citado 2 vezes nas páginas 35 e 54.

YAACOB, W. W. et al. Predicting student drop-out in higher institution using data mining techniques. In: IOP PUBLISHING. *Journal of Physics: Conference Series*. [S.l.], 2020. v. 1496, n. 1, p. 012005. Citado 3 vezes nas páginas 34, 40 e 51.

Apêndices

APÊNDICE A – Levantamento de Estatísticas

Listing A.1 – Quantidade Alunos Presencial

```
#!/usr/bin/env python
# coding: utf-8

# In[2]:


#Verificando a Quantidade

basePresencial2016 =
    open('presencial2016ColunaDesistenteAjustada.csv')
print(len(basePresencial2016.readlines()))


# In[3]:


#Formatando a base

import time
from tqdm import tqdm
inicio = time.time()

basePresencial2016 =
    open('presencial2016ColunaDesistenteAjustada.csv')
baseInicial = list(map(lambda x: x.strip('\n').split('|'),
    tqdm(basePresencial2016.readlines())))


fim = time.time()
print(fim - inicio)
print(baseInicial[0:5])


# In[6]:
```

```
32
33
34 #Quantidades Alunos em Instituicoes Publicas
35
36 alunosPublicas = []
37 for aluno in baseInicial:
38     if(aluno[2] in ('1','2','3')):
39         alunosPublicas.append(aluno)
40
41 alunosUniversidadesPublicas = []
42 for aluno in alunosPublicas:
43     if(aluno[4] == '1'):
44         alunosUniversidadesPublicas.append(aluno)
45
46 alunosCentrosUniversitariosPublicos = []
47 for aluno in alunosPublicas:
48     if(aluno[4] == '2'):
49         alunosCentrosUniversitariosPublicos.append(aluno)
50
51 alunosFaculdadesPublicas = []
52 for aluno in alunosPublicas:
53     if(aluno[4] == '3'):
54         alunosFaculdadesPublicas.append(aluno)
55
56 soma = len(alunosUniversidadesPublicas)
57        +len(alunosCentrosUniversitariosPublicos)
58        +len(alunosFaculdadesPublicas)
59 print('Quantidade Alunos em Publicas: ', soma)
60 print('Quantidade Alunos em Universidades Publicas: ',
      len(alunosUniversidadesPublicas))
61 print('Quantidade Alunos em Centros Universitarios Publicos: ',
      len(alunosCentrosUniversitariosPublicos))
62 print('Quantidade Alunos em Faculdades Publicas: ',
      len(alunosFaculdadesPublicas))
63
64
65 # In[5]:
66
67
68 #Quantidades Alunos em Instituicoes Privadas
69
70 alunosPrivadas = []
```

```
for aluno in baseInicial:
    if(aluno[2] in ('4','5')):
        alunosPrivadas.append(aluno)

alunosUniversidadesPrivadas = []
for aluno in alunosPrivadas:
    if(aluno[4] == '1'):
        alunosUniversidadesPrivadas.append(aluno)

alunosCentrosUniversitariosPrivados = []
for aluno in alunosPrivadas:
    if(aluno[4] == '2'):
        alunosCentrosUniversitariosPrivados.append(aluno)

alunosFaculdadesPrivadas = []
for aluno in alunosPrivadas:
    if(aluno[4] == '3'):
        alunosFaculdadesPrivadas.append(aluno)

print('Quantidade Alunos em Privadas: ', len(alunosPrivadas))
print('Quantidade Alunos em Universidades Privadas: ',
   len(alunosUniversidadesPrivadas))
print('Quantidade Alunos em Centros Universitarios Privados: ',
   len(alunosCentrosUniversitariosPrivados))
print('Quantidade Alunos em Faculdades Privadas: ',
   len(alunosFaculdadesPrivadas))
```

Listing A.2 – Quantidade Cursos Presencial

```
#!/usr/bin/env python
# coding: utf-8

# In[2]:


#Formatando a base

import time
from tqdm import tqdm
inicio = time.time()

abrir = open('DM_CURSO_2016.csv')
```

```
14 baseInicial = list(map(lambda x: x.strip('\n').split('|'),
       tqdm(abrir.readlines())))
15
16
17 fim = time.time()
18 print(fim - inicio)
19
20
21 # In[7]:
22
23
24 #Quantidades Cursos em Instituicoes Publicas
25
26 cursosPublicas = []
27 for curso in baseInicial:
28     if(curso[2] in ('1','2','3')):
29         cursosPublicas.append(curso)
30
31 cursosUniversidadesPublicas = []
32 for curso in cursosPublicas:
33     if(curso[4] == '1'):
34         cursosUniversidadesPublicas.append(curso)
35
36 cursosCentrosUniversitariosPublicos = []
37 for curso in cursosPublicas:
38     if(curso[4] == '2'):
39         cursosCentrosUniversitariosPublicos.append(curso)
40
41 cursosFaculdadesPublicas = []
42 for curso in cursosPublicas:
43     if(curso[4] == '3'):
44         cursosFaculdadesPublicas.append(curso)
45
46 soma = len(cursosUniversidadesPublicas)
47        +len(cursosCentrosUniversitariosPublicos)
48        +len(cursosFaculdadesPublicas)
49 print('Quantidade cursos em Publicas: ', soma)
50 print('Quantidade cursos em Universidades Publicas: ',
       len(cursosUniversidadesPublicas))
51 print('Quantidade cursos em Centros Universitarios Publicos: ',
       len(cursosCentrosUniversitariosPublicos))
52 print('Quantidade cursos em Faculdades Publicas: ',
```

```
    len(cursosFaculdadesPublicas))
53
54
55 # In[5]:
56
57
58 #Quantidades Cursos em Instituicoes Privadas
59
60 cursosPrivadas = []
61 for curso in baseInicial:
62     if(curso[2] in ('4','5')):
63         cursosPrivadas.append(curso)
64
65 cursosUniversidadesPrivadas = []
66 for curso in cursosPrivadas:
67     if(curso[4] == '1'):
68         cursosUniversidadesPrivadas.append(curso)
69
70 cursosCentrosUniversitariosPrivados = []
71 for curso in cursosPrivadas:
72     if(curso[4] == '2'):
73         cursosCentrosUniversitariosPrivados.append(curso)
74
75 cursosFaculdadesPrivadas = []
76 for curso in cursosPrivadas:
77     if(curso[4] == '3'):
78         cursosFaculdadesPrivadas.append(curso)
79
80 print('Quantidade cursos em Privadas: ', len(cursosPrivadas))
81 print('Quantidade cursos em Universidades Privadas: ',
    len(cursosUniversidadesPrivadas))
82 print('Quantidade cursos em Centros Universitarios Privados: ',
    len(cursosCentrosUniversitariosPrivados))
83 print('Quantidade cursos em Faculdades Privadas: ',
    len(cursosFaculdadesPrivadas))
```

Listing A.3 – Quantidade Instituicoes Presencial

```
1 #!/usr/bin/env python
2 # coding: utf-8
3
4 # In[13]:
5
```

```
6
7  #Formatando a base
8
9  import time
10 from tqdm import tqdm
11 inicio = time.time()
12
13 abrir = open('DM_IES_2016.csv')
14 baseInicial = list(map(lambda x: x.strip('\n').split('|'),
       tqdm(abrir.readlines())))
15
16
17 fim = time.time()
18 print(fim - inicio)
19
20
21 # In[26]:
22
23
24 #Quantidades Instituicoes Publicas
25
26 iesPublicas = []
27 for ies in baseInicial:
28     if(ies[5] in ('1','2','3')):
29         iesPublicas.append(ies)
30
31 iesUniversidadesPublicas = []
32 for ies in iesPublicas:
33     if(ies[7] == '1'):
34         iesUniversidadesPublicas.append(ies)
35
36 iesCentrosUniversitariosPublicos = []
37 for ies in iesPublicas:
38     if(ies[7] == '2'):
39         iesCentrosUniversitariosPublicos.append(ies)
40
41 iesFaculdadesPublicas = []
42 for ies in iesPublicas:
43     if(ies[7] == '3'):
44         iesFaculdadesPublicas.append(ies)
45
46 soma = len(iesUniversidadesPublicas)
```

```
47          +len(iesCentrosUniversitariosPublicos)
48          +len(iesFaculdadesPublicas);
49  print('Quantidade Publicas: ', soma)
50  print('Quantidade Universidades Publicas: ',
        len(iesUniversidadesPublicas))
51  print('Quantidade Centros Universitarios Publicos: ',
        len(iesCentrosUniversitariosPublicos))
52  print('Quantidade Faculdades Publicas: ', len(iesFaculdadesPublicas))
53
54
55  # In[22]:
56
57
58  #Quantidades Instituicoes Privadas
59
60  iesPrivadas = []
61  for ies in baseInicial:
62      if(ies[5] in ('4','5')):
63          iesPrivadas.append(ies)
64
65  iesUniversidadesPrivadas = []
66  for ies in iesPrivadas:
67      if(ies[7] == '1'):
68          iesUniversidadesPrivadas.append(ies)
69
70  iesCentrosUniversitariosPrivados = []
71  for ies in iesPrivadas:
72      if(ies[7] == '2'):
73          iesCentrosUniversitariosPrivados.append(ies)
74
75  iesFaculdadesPrivadas = []
76  for ies in iesPrivadas:
77      if(ies[7] == '3'):
78          iesFaculdadesPrivadas.append(ies)
79
80  print('Quantidade Privadas: ', len(iesPrivadas))
81  print('Quantidade Universidades Privadas: ',
        len(iesUniversidadesPrivadas))
82  print('Quantidade Centros Universitarios Privados: ',
        len(iesCentrosUniversitariosPrivados))
83  print('Quantidade Faculdades Privadas: ', len(iesFaculdadesPrivadas))
```

APÊNDICE B – Mineracao das Bases

Listing B.1 – Divisao Presencial e EAD 2016

```
#!/usr/bin/env python
# coding: utf-8

# In[2]:


import time
from tqdm import tqdm
inicio = time.time()

abrir = open('DM_ALUNO_2016.csv')
baseInicial = list(map(lambda x: x.strip('\n').split('|'),
    tqdm(abrir.readlines())))

novaBase = []
for linha in tqdm(range(len(baseInicial))):
    if(baseInicial[linha][13] != '2'):
        novaBase.append(baseInicial[linha])

fim = time.time()
print(fim-inicio)


# In[3]:


#Gravando o presencial ---------------
import csv

file = open('presencial2016.csv', 'a+', newline ='\n')
with file:
    write = csv.writer(file, delimiter='|')
    write.writerows(novaBase)


# In[4]:
```

```
36
37
38 abrir = open('presencial2016.csv')
39 print(len(abrir.readlines()))
40 #--------------------------------------
41
42
43 # In[5]:
44
45
46 #Gravando o EAD ---------------------------
47 novaBaseEAD = []
48 for linha in tqdm(range(len(baseInicial))):
49     if(baseInicial[linha][13] != '1'):
50         novaBaseEAD.append(baseInicial[linha])
51
52
53 # In[6]:
54
55
56 import csv
57
58 file = open('ead2016.csv', 'a+', newline ='\n')
59 with file:
60     write = csv.writer(file, delimiter='|')
61     write.writerows(novaBaseEAD)
62
63
64 # In[7]:
65
66
67 abrir = open('ead2016.csv')
68 print(len(abrir.readlines()))
```

Listing B.2 – Divisao Presencial e EAD 2017

```
1 #!/usr/bin/env python
2 # coding: utf-8
3
4 # In[3]:
5
6
7 import time
```

```
8  from tqdm import tqdm
9  inicio = time.time()
10
11 abrir = open('DM_ALUNO_2017.csv')
12 baseInicial = list(map(lambda x: x.strip('\n').split('|'),
       tqdm(abrir.readlines())))
13
14 novaBase = []
15 for linha in tqdm(range(len(baseInicial))):
16     if(baseInicial[linha][8] != '2'):
17         novaBase.append(baseInicial[linha])
18
19 fim = time.time()
20 print(fim-inicio)
21
22
23 # In[4]:
24
25
26 #Gravando o presencial ---------------
27 import csv
28
29 file = open('presencial2017.csv', 'a+', newline ='\n')
30 with file:
31     write = csv.writer(file, delimiter='|')
32     write.writerows(novaBase)
33
34
35 # In[5]:
36
37
38 abrir = open('presencial2017.csv')
39 print(len(abrir.readlines()))
40 #--------------------------------------
41
42
43 # In[6]:
44
45
46 #Gravando o EAD ---------------------------
47 novaBaseEAD = []
48 for linha in tqdm(range(len(baseInicial))):
```

```
49      if(baseInicial[linha][8] != '1'):
50          novaBaseEAD.append(baseInicial[linha])
51
52
53  # In[7]:
54
55
56  import csv
57
58  file = open('ead2017.csv', 'a+', newline ='\n')
59  with file:
60      write = csv.writer(file, delimiter='|')
61      write.writerows(novaBaseEAD)
62
63
64  # In[8]:
65
66
67  abrir = open('ead2017.csv')
68  print(len(abrir.readlines()))
```

Listing B.3 – Divisao EAD Publico e Privado

```
1   #!/usr/bin/env python
2   # coding: utf-8
3
4   # In[1]:
5
6
7   #Dividindo a base do EAD em publica e privada
8
9   import time
10  from tqdm import tqdm
11  inicio = time.time()
12
13  abrir = open('ead2016.csv')
14  baseInicial = list(map(lambda x: x.strip('\n').split('|'),
        tqdm(abrir.readlines())))
15
16  novaBase = []
17  for linha in tqdm(range(len(baseInicial))):
18      if(baseInicial[linha][2] == '4' or baseInicial[linha][2] == '5'
           or baseInicial[linha][2] == 'CO_CATEGORIA_ADMINISTRATIVA'):
```

```
19          novaBase.append(baseInicial[linha])
20
21  fim = time.time()
22  print(fim-inicio)
23
24
25  # In[2]:
26
27
28  #Gravando o ead privada
29
30  import csv
31
32  file = open('ead2016privada.csv', 'a+', newline ='\n')
33  with file:
34      write = csv.writer(file, delimiter='|')
35      write.writerows(novaBase)
36
37
38  # In[3]:
39
40
41  abrir = open('ead2016privada.csv')
42  print(len(abrir.readlines()))
43  #--------------------------------------
44
45
46  # In[4]:
47
48
49  #Gravando o ead publica
50
51  novaBaseEAD = []
52  for linha in tqdm(range(len(baseInicial))):
53      if(baseInicial[linha][2] == '1' or baseInicial[linha][2] == '2'
           or baseInicial[linha][2] == '3'
54         or baseInicial[linha][2] == '7' or baseInicial[linha][2] ==
              'CO_CATEGORIA_ADMINISTRATIVA'):
55          novaBaseEAD.append(baseInicial[linha])
56
57
58  # In[5]:
```

```
59
60
61 import csv
62
63 file = open('ead2016publica.csv', 'a+', newline ='\n')
64 with file:
65     write = csv.writer(file, delimiter='|')
66     write.writerows(novaBaseEAD)
67
68
69 # In[6]:
70
71
72 abrir = open('ead2016publica.csv')
73 print(len(abrir.readlines()))
```

Listing B.4 – Divisao Presencial Publico e Privado

```
1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # In[1]:
5
6
7  #Dividindo a base do Presencial em publica e privada
8
9  import time
10 from tqdm import tqdm
11 inicio = time.time()
12
13 abrir = open('presencial2016.csv')
14 baseInicial = list(map(lambda x: x.strip('\n').split('|'),
       tqdm(abrir.readlines())))
15
16 novaBase = []
17 for linha in tqdm(range(len(baseInicial))):
18     if(baseInicial[linha][2] == '4' or baseInicial[linha][2] == '5'
          or baseInicial[linha][2] == 'CO_CATEGORIA_ADMINISTRATIVA'):
19         novaBase.append(baseInicial[linha])
20
21 fim = time.time()
22 print(fim-inicio)
23
```

```
24
25 # In[2]:
26
27
28 #Gravando o presencial privada
29
30 import csv
31
32 file = open('presencial2016privada.csv', 'a+', newline ='\n')
33 with file:
34     write = csv.writer(file, delimiter='|')
35     write.writerows(novaBase)
36
37
38 # In[3]:
39
40
41 abrir = open('presencial2016privada.csv')
42 print(len(abrir.readlines()))
43 #---------------------------------------
44
45
46 # In[4]:
47
48
49 #Gravando o presencial publica
50
51 novaBasePresencial = []
52 for linha in tqdm(range(len(baseInicial))):
53     if(baseInicial[linha][2] == '1' or baseInicial[linha][2] == '2'
        or baseInicial[linha][2] == '3'
54        or baseInicial[linha][2] == '7' or baseInicial[linha][2] ==
           'CO_CATEGORIA_ADMINISTRATIVA'):
55         novaBasePresencial.append(baseInicial[linha])
56
57
58 # In[5]:
59
60
61 import csv
62
63 file = open('presencial2016publica.csv', 'a+', newline ='\n')
```

```
with file:
    write = csv.writer(file, delimiter='|')
    write.writerows(novaBasePresencial)


# In[6]:


abrir = open('presencial2016publica.csv')
print(len(abrir.readlines()))
```

Listing B.5 – Identificando alunos desistentes EAD Privada

```
#!/usr/bin/env python
# coding: utf-8

# In[2]:


import time
from tqdm import tqdm
inicio = time.time()


#Base de inicial ------------------------------------------------
abrir = open('ead2016privada.csv')
baseInicial = list(map(lambda x: x.strip('\n').split('|'),
   tqdm(abrir.readlines())))


#Base de busca ------------------------------------------------
abrir2 = open('ead2017.csv')
baseBusca = list(map(lambda x: x.strip('\n').split('|'),
   tqdm(abrir2.readlines())))
baseBusca = sorted(baseBusca, key = lambda x: x[14])


fim = time.time()
print(fim - inicio)


print('------------------------------------------------------------')
print('------------------------------------------------------------')
```

```
29
30
31 # In[3]:
32
33
34 from bisect import bisect_left
35
36 def BinarySearch(a, x):
37     i = bisect_left(a, x)
38     if i != len(a) and a[i] == x:
39         return i
40     else:
41         return -1
42
43
44 keys = [r[14] for r in tqdm(baseBusca)]
45
46
47 # In[4]:
48
49
50 #Descobrir qual e a linha dos titulos
    ----------------------------------------------------
51 inicio2 = time.time()
52
53 n = 0
54 for linha in tqdm(range(len(baseInicial))):
55     n = n+1
56     for item in range(len(baseInicial[linha])):
57         if(baseInicial[linha][item] == 'CO_ALUNO'):
58             print(baseInicial[linha])
59             print(linha)
60             break
61
62 fim2 = time.time()
63 print(fim2 - inicio2)
64
65
66 # In[5]:
67
68
69 print(baseInicial[0])
```

```
70
71
72  # In[6]:
73
74
75  #Setando o campo de
       DESISTENTE----------------------------------------------------
76  #Sao alunos que realmente desistiram dos estudos e nao que mudaram
       de curso, os concluintes nao sao considerados,
77  #e nem formados.
78  inicio2 = time.time()
79  baseNova = []
80
81  baseInicial[0].append('DESISTENTE')
82  baseNova.append(baseInicial[0])
83
84  del(baseInicial[0])
85  for linha in tqdm(range(len(baseInicial))):
86      if(BinarySearch(keys,baseInicial[linha][27]) == -1 and
           baseInicial[linha][115] == '0' and baseInicial[linha][55] !=
           '6'):
87          baseInicial[linha].append(1)
88          baseNova.append(baseInicial[linha])
89      else:
90          baseInicial[linha].append(0)
91          baseNova.append(baseInicial[linha])
92
93
94  fim2 = time.time()
95  print(fim2 - inicio2)
96
97
98  # In[7]:
99
100
101 #Gravando csv ----------------------------------------------------
102 import csv
103
104 file = open('ead2016ColunaDesistentePrivada.csv', 'a+', newline
       ='\n')
105 with file:
106     write = csv.writer(file, delimiter='|')
```

```
107     write.writerows(baseNova)
108
109
110 # In[8]:
111
112
113 abrir = open('ead2016ColunaDesistente.csv')
114 print(len(abrir.readlines()))
115
116 abrir = open('ead2016ColunaDesistentePrivada.csv')
117 print(len(abrir.readlines()))
```

Listing B.6 – Identificando alunos desistentes EAD Publica

```
1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # In[1]:
5
6
7  import time
8  from tqdm import tqdm
9  inicio = time.time()
10
11
12 #Base de inicial ------------------------------------------------
13 abrir = open('ead2016publica.csv')
14 baseInicial = list(map(lambda x: x.strip('\n').split('|'),
       tqdm(abrir.readlines())))
15
16
17 #Base de busca -------------------------------------------------
18 abrir2 = open('ead2017.csv')
19 baseBusca = list(map(lambda x: x.strip('\n').split('|'),
       tqdm(abrir2.readlines())))
20 baseBusca = sorted(baseBusca, key = lambda x: x[14])
21
22
23 fim = time.time()
24 print(fim - inicio)
25
26
27 print('-------------------------------------------------------------')
```

```
28 print('------------------------------------------------------------')
29
30
31 # In[2]:
32
33
34 from bisect import bisect_left
35
36 def BinarySearch(a, x):
37     i = bisect_left(a, x)
38     if i != len(a) and a[i] == x:
39         return i
40     else:
41         return -1
42
43
44 keys = [r[14] for r in tqdm(baseBusca)]
45
46
47 # In[3]:
48
49
50 #Descobrir qual e a linha dos titulos
    ----------------------------------------------------
51 inicio2 = time.time()
52
53 n = 0
54 for linha in tqdm(range(len(baseInicial))):
55     n = n+1
56     for item in range(len(baseInicial[linha])):
57         if(baseInicial[linha][item] == 'CO_ALUNO'):
58             print(baseInicial[linha])
59             print(linha)
60             break
61
62 fim2 = time.time()
63 print(fim2 - inicio2)
64
65
66 # In[4]:
67
68
```

```
69 print(baseInicial[0])
70
71
72 # In[5]:
73
74
75 #Setando o campo de
      DESISTENTE----------------------------------------------------
76 #Sao alunos que realmente desistiram dos estudos e nao que mudaram
      de curso, os concluintes nao sao considerados,
77 #e nem formados.
78 inicio2 = time.time()
79 baseNova = []
80
81 baseInicial[0].append('DESISTENTE')
82 baseNova.append(baseInicial[0])
83
84 del(baseInicial[0])
85 for linha in tqdm(range(len(baseInicial))):
86     if(BinarySearch(keys,baseInicial[linha][27]) == -1 and
          baseInicial[linha][115] == '0' and baseInicial[linha][55] !=
          '6'):
87         baseInicial[linha].append(1)
88         baseNova.append(baseInicial[linha])
89     else:
90         baseInicial[linha].append(0)
91         baseNova.append(baseInicial[linha])
92
93
94 fim2 = time.time()
95 print(fim2 - inicio2)
96
97
98 # In[6]:
99
100
101 #Gravando csv ----------------------------------------------------
102 import csv
103
104 file = open('ead2016ColunaDesistentePublica.csv', 'a+', newline
      ='\n')
105 with file:
```

```
106     write = csv.writer(file, delimiter='|')
107     write.writerows(baseNova)
108
109
110 # In[7]:
111
112
113 abrir = open('ead2016ColunaDesistente.csv')
114 print(len(abrir.readlines()))
115
116 abrir = open('ead2016ColunaDesistentePublica.csv')
117 print(len(abrir.readlines()))
```

Listing B.7 – Identificando alunos desistentes Presencial Privada

```
1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # In[1]:
5
6
7  import time
8  from tqdm import tqdm
9  inicio = time.time()
10
11
12 #Base de inicial ------------------------------------------------
13 abrir = open('presencial2016privada.csv')
14 baseInicial = list(map(lambda x: x.strip('\n').split('|'),
       tqdm(abrir.readlines())))
15
16
17 #Base de busca -------------------------------------------------
18 abrir2 = open('presencial2017.csv')
19 baseBusca = list(map(lambda x: x.strip('\n').split('|'),
       tqdm(abrir2.readlines())))
20 baseBusca = sorted(baseBusca, key = lambda x: x[14])
21
22
23 fim = time.time()
24 print(fim - inicio)
25
26
```

```
27 print('------------------------------------------------------------')
28 print('------------------------------------------------------------')
29
30
31 # In[2]:
32
33
34 from bisect import bisect_left
35
36 def BinarySearch(a, x):
37     i = bisect_left(a, x)
38     if i != len(a) and a[i] == x:
39         return i
40     else:
41         return -1
42
43
44 keys = [r[14] for r in tqdm(baseBusca)]
45
46
47 # In[3]:
48
49
50 #Descobrir qual e a linha dos titulos
    ----------------------------------------------------
51 inicio2 = time.time()
52
53 n = 0
54 for linha in tqdm(range(len(baseInicial))):
55     n = n+1
56     for item in range(len(baseInicial[linha])):
57         if(baseInicial[linha][item] == 'CO_ALUNO'):
58             print(baseInicial[linha])
59             print(linha)
60             break
61
62 fim2 = time.time()
63 print(fim2 - inicio2)
64
65
66 # In[4]:
67
```

```
68
69 print(baseInicial[0])
70
71
72 # In[5]:
73
74
75 #Setando o campo de
      DESISTENTE----------------------------------------
76 #ao alunos que realmente desistiram dos estudos e nao que mudaram de
      curso, os concluintes nao sao considerados,
77 #e nem formados.
78 inicio2 = time.time()
79 baseNova = []
80
81 baseInicial[0].append('DESISTENTE')
82 baseNova.append(baseInicial[0])
83
84 del(baseInicial[0])
85 for linha in tqdm(range(len(baseInicial))):
86     if(BinarySearch(keys,baseInicial[linha][27]) == -1 and
          baseInicial[linha][115] == '0' and baseInicial[linha][55] !=
          '6'):
87         baseInicial[linha].append(1)
88         baseNova.append(baseInicial[linha])
89     else:
90         baseInicial[linha].append(0)
91         baseNova.append(baseInicial[linha])
92
93
94 fim2 = time.time()
95 print(fim2 - inicio2)
96
97
98 # In[6]:
99
100
101 #Gravando csv ----------------------------------------------------
102 import csv
103
104 file = open('presencial2016ColunaDesistentePrivada.csv', 'a+',
      newline ='\n')
```

```
with file:
    write = csv.writer(file, delimiter='|')
    write.writerows(baseNova)


# In[7]:


abrir = open('presencial2016ColunaDesistente.csv')
print(len(abrir.readlines()))

abrir = open('presencial2016ColunaDesistentePrivada.csv')
print(len(abrir.readlines()))
```

Listing B.8 – Identificando alunos desistentes Presencial Publica

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import time
from tqdm import tqdm
inicio = time.time()


#Base de inicial ------------------------------------------------
abrir = open('presencial2016publica.csv')
baseInicial = list(map(lambda x: x.strip('\n').split('|'),
   tqdm(abrir.readlines())))


#Base de busca ------------------------------------------------
abrir2 = open('presencial2017.csv')
baseBusca = list(map(lambda x: x.strip('\n').split('|'),
   tqdm(abrir2.readlines())))
baseBusca = sorted(baseBusca, key = lambda x: x[14])


fim = time.time()
print(fim - inicio)

```

```
26
27 print('------------------------------------------------------------')
28 print('------------------------------------------------------------')
29
30
31 # In[2]:
32
33
34 from bisect import bisect_left
35
36 def BinarySearch(a, x):
37     i = bisect_left(a, x)
38     if i != len(a) and a[i] == x:
39         return i
40     else:
41         return -1
42
43
44 keys = [r[14] for r in tqdm(baseBusca)]
45
46
47 # In[3]:
48
49
50 #Descobrir qual e a linha dos titulos
    ----------------------------------------------------
51 inicio2 = time.time()
52
53 n = 0
54 for linha in tqdm(range(len(baseInicial))):
55     n = n+1
56     for item in range(len(baseInicial[linha])):
57         if(baseInicial[linha][item] == 'CO_ALUNO'):
58             print(baseInicial[linha])
59             print(linha)
60             break
61
62 fim2 = time.time()
63 print(fim2 - inicio2)
64
65
66 # In[4]:
```

```
67
68
69  print(baseInicial[0])
70
71
72  # In[5]:
73
74
75  #Setando o campo de
       DESISTENTE----------------------------------------------------
76  #Sao alunos que realmente desistiram dos estudos e nao que mudaram
       de curso, os concluintes nao sao considerados,
77  #e nem formados.
78  inicio2 = time.time()
79  baseNova = []
80
81  baseInicial[0].append('DESISTENTE')
82  baseNova.append(baseInicial[0])
83
84  del(baseInicial[0])
85  for linha in tqdm(range(len(baseInicial))):
86      if(BinarySearch(keys,baseInicial[linha][27]) == -1 and
           baseInicial[linha][115] == '0' and baseInicial[linha][55] !=
           '6'):
87          baseInicial[linha].append(1)
88          baseNova.append(baseInicial[linha])
89      else:
90          baseInicial[linha].append(0)
91          baseNova.append(baseInicial[linha])
92
93
94  fim2 = time.time()
95  print(fim2 - inicio2)
96
97
98  # In[6]:
99
100
101 #Gravando csv ----------------------------------------------------
102 import csv
103
104 file = open('presencial2016ColunaDesistentePublica.csv', 'a+',
```

```
    newline ='\n')
with file:
    write = csv.writer(file, delimiter='|')
    write.writerows(baseNova)


# In[7]:


abrir = open('presencial2016ColunaDesistente.csv')
print(len(abrir.readlines()))

abrir = open('presencial2016ColunaDesistentePublica.csv')
print(len(abrir.readlines()))
```

APÊNDICE C – Preparo e Execução dos Modelos de Classificação

Listing C.1 – Classificador EAD privada

```
1
2  #!/usr/bin/env python
3  # coding: utf-8
4
5  # # Classificadores de Evasao Estudantil
6
7  # ## Configurando pyspark:
8
9  # In[1]:
10
11
12 #!pip install pyspark
13
14
15 # In[2]:
16
17
18 from pyspark.sql import SparkSession
19
20
21 # In[3]:
22
23
24 #Define onde sera criado a sessao do spark, nesse caso sera local
25 spark =
       SparkSession.builder.master('local[*]').appName("Classificacao
       com Spark").getOrCreate()
26
27
28 # In[4]:
29
30
31 spark
32
```

```
33
34 # ## Carregando os dados do csv:
35
36 # In[5]:
37
38
39 dados = spark.read.csv('ead2016ColunaDesistentePrivada.csv',sep='|',
40 header=True, inferSchema=True)
41
42
43 # In[6]:
44
45
46 dados.count()
47
48
49 # In[7]:
50
51
52 #Quantidade de Desistentes e Nao-Desistentes
53 dados.groupBy('DESISTENTE').count().show()
54
55
56 # ## Selecionando e tratando atributos:
57
58 # In[8]:
59
60
61 dados = dados.drop(*(
62 # 'CO_IES',
63   'NO_IES', #--Campo inutil
64 # 'CO_CATEGORIA_ADMINISTRATIVA',
65   'DS_CATEGORIA_ADMINISTRATIVA', #--Campo inutil
66 # 'CO_ORGANIZACAO_ACADEMICA',
67   'DS_ORGANIZACAO_ACADEMICA', #--Campo inutil
68 # 'CO_CURSO',
69   'NO_CURSO', #--Campo inutil
70 # 'CO_CURSO_POLO',
71 # 'CO_TURNO_ALUNO',
72   'DS_TURNO_ALUNO', #--Campo inutil
73 # 'CO_GRAU_ACADEMICO',
74   'DS_GRAU_ACADEMICO', #--Campo inutil
```

```
75  # 'CO_MODALIDADE_ENSINO',
76    'DS_MODALIDADE_ENSINO', #--Campo inutil
77  # 'CO_NIVEL_ACADEMICO',
78    'DS_NIVEL_ACADEMICO', #--Campo inutil
79    'CO_OCDE', #--Nao tratado
80    'NO_OCDE', #--Campo inutil
81  # 'CO_OCDE_AREA_GERAL',
82    'NO_OCDE_AREA_GERAL', #--Campo inutil
83  # 'CO_OCDE_AREA_ESPECIFICA',
84    'NO_OCDE_AREA_ESPECIFICA', #--Campo inutil
85  # 'CO_OCDE_AREA_DETALHADA',
86    'NO_OCDE_AREA_DETALHADA', #--Campo inutil
87  # 'CO_ALUNO_CURSO',
88    'CO_ALUNO_CURSO_ORIGEM', #--Nao tratado
89    'CO_ALUNO', #--Campo inutil
90  # 'CO_COR_RACA_ALUNO',
91    'DS_COR_RACA_ALUNO', #--Campo inutil
92  # 'IN_SEXO_ALUNO',
93    'DS_SEXO_ALUNO', #--Campo inutil
94  # 'NU_ANO_ALUNO_NASC',
95  # 'NU_MES_ALUNO_NASC',
96  # 'NU_DIA_ALUNO_NASC',
97  # 'NU_IDADE_ALUNO',
98  # 'CO_NACIONALIDADE_ALUNO',
99    'DS_NACIONALIDADE_ALUNO', #--Campo inutil
100 # 'CO_PAIS_ORIGEM_ALUNO',
101   'CO_UF_NASCIMENTO', #--Nao tratado
102   'CO_MUNICIPIO_NASCIMENTO', #--Nao tratado
103 # 'IN_ALUNO_DEF_TGD_SUPER',
104 # 'IN_DEF_AUDITIVA',
105 # 'IN_DEF_FISICA',
106 # 'IN_DEF_INTELECTUAL',
107 # 'IN_DEF_MULTIPLA',
108 # 'IN_DEF_SURDEZ',
109 # 'IN_DEF_SURDOCEGUEIRA',
110 # 'IN_DEF_BAIXA_VISAO',
111 # 'IN_DEF_CEGUEIRA',
112 # 'IN_DEF_SUPERDOTACAO',
113 # 'IN_TGD_AUTISMO_INFANTIL',
114 # 'IN_TGD_SINDROME_ASPERGER',
115 # 'IN_TGD_SINDROME_RETT',
116 # 'IN_TGD_TRANSTOR_DESINTEGRATIVO',
```

```
117 # 'CO_ALUNO_SITUACAO',
118   'DS_ALUNO_SITUACAO', #--Campo inutil
119 # 'QT_CARGA_HORARIA_TOTAL',
120 # 'QT_CARGA_HORARIA_INTEG',
121   'DT_INGRESSO_CURSO', #--Nao tratado
122 # 'IN_ING_VESTIBULAR',
123 # 'IN_ING_ENEM',
124 # 'IN_ING_AVALIACAO_SERIADA',
125 # 'IN_ING_SELECAO_SIMPLIFICADA',
126 # 'IN_ING_SELECAO_VAGA_REMANESC',
127 # 'IN_ING_SELECAO_VAGA_PROG_ESPEC',
128 # 'IN_ING_TRANSF_EXOFFICIO',
129 # 'IN_ING_DECISAO_JUDICIAL',
130 # 'IN_ING_CONVENIO_PECG',
131 # 'IN_RESERVA_VAGAS',
132 # 'IN_RESERVA_ETNICO',
133 # 'IN_RESERVA_DEFICIENCIA',
134 # 'IN_RESERVA_ENSINO_PUBLICO',
135 # 'IN_RESERVA_RENDA_FAMILIAR',
136 # 'IN_RESERVA_OUTRA',
137 # 'IN_FINANC_ESTUDANTIL',
138 # 'IN_FIN_REEMB_FIES',
139 # 'IN_FIN_REEMB_ESTADUAL',
140 # 'IN_FIN_REEMB_MUNICIPAL',
141 # 'IN_FIN_REEMB_PROG_IES',
142 # 'IN_FIN_REEMB_ENT_EXTERNA',
143 # 'IN_FIN_REEMB_OUTRA',
144 # 'IN_FIN_NAOREEMB_PROUNI_INTEGR',
145 # 'IN_FIN_NAOREEMB_PROUNI_PARCIAL',
146 # 'IN_FIN_NAOREEMB_ESTADUAL',
147 # 'IN_FIN_NAOREEMB_MUNICIPAL',
148 # 'IN_FIN_NAOREEMB_PROG_IES',
149 # 'IN_FIN_NAOREEMB_ENT_EXTERNA',
150 # 'IN_FIN_NAOREEMB_OUTRA',
151 # 'IN_APOIO_SOCIAL',
152 # 'IN_APOIO_ALIMENTACAO',
153 # 'IN_APOIO_BOLSA_PERMANENCIA',
154 # 'IN_APOIO_BOLSA_TRABALHO',
155 # 'IN_APOIO_MATERIAL_DIDATICO',
156 # 'IN_APOIO_MORADIA',
157 # 'IN_APOIO_TRANSPORTE',
158 # 'IN_ATIVIDADE_EXTRACURRICULAR',
```

```
159 # 'IN_COMPL_ESTAGIO',
160 # 'IN_COMPL_EXTENSAO',
161 # 'IN_COMPL_MONITORIA',
162 # 'IN_COMPL_PESQUISA',
163 # 'IN_BOLSA_ESTAGIO',
164 # 'IN_BOLSA_EXTENSAO',
165 # 'IN_BOLSA_MONITORIA',
166 # 'IN_BOLSA_PESQUISA',
167 # 'CO_TIPO_ESCOLA_ENS_MEDIO',
168 # 'IN_ALUNO_PARFOR',
169  'CO_SEMESTRE_CONCLUSAO', #--Nao tratado
170 # 'CO_SEMESTRE_REFERENCIA',
171 # 'IN_MOBILIDADE_ACADEMICA',
172 # 'CO_MOBILIDADE_ACADEMICA',
173 # 'CO_MOBILIDADE_ACADEMICA_INTERN',
174   'CO_IES_DESTINO', #--Nao tratado
175   'CO_PAIS_DESTINO', #--Nao tratado
176   'IN_MATRICULA', #--Campo inutil
177 # 'IN_CONCLUINTE',
178 # 'IN_INGRESSO_TOTAL',
179 # 'IN_INGRESSO_VAGA_NOVA',
180 # 'ANO_INGRESSO',
181 # 'DESISTENTE'
182                     ))
183
184
185 # In[9]:
186
187
188 dados.printSchema()
189
190
191 # In[10]:
192
193
194 #dados.select('IN_DEF_INTELECTUAL').where('IN_DEF_INTELECTUAL is
       null').show()
195 #dados.groupBy('IN_DEF_INTELECTUAL').count().show()
196
197
198 # ## Formatando Atributos para processamento
199
```

```
# In[11]:


from pyspark.sql import functions as f

colunasParaCorrigir0 = [
'CO_TURNO_ALUNO'
]

todasColunas0 = [f.when(f.col(c) == 1, 1).when(f.col(c) == 2, 2).
                 when(f.col(c) == 3, 3).when(f.col(c) == 4, 4).
                 otherwise(5).alias(c)
                for c in colunasParaCorrigir0]


# In[12]:


for coluna in dados.columns:
    if coluna not in colunasParaCorrigir0:
        todasColunas0.insert(0, coluna)


#todasColunas0

dados = dados.select(todasColunas0)
#dataset.groupBy('IN_DEF_INTELECTUAL').count().show()


# In[13]:


from pyspark.sql import functions as f

colunasParaCorrigir1 = [
 #'CO_SEMESTRE_CONCLUSAO',
 'CO_SEMESTRE_REFERENCIA',
 'CO_MOBILIDADE_ACADEMICA',
 'CO_MOBILIDADE_ACADEMICA_INTERN'
]

```

```
241 todasColunas1 = [f.when(f.col(c) == 1, 1).when(f.col(c) == 2 ,
    2).otherwise(3).alias(c)
242                 for c in colunasParaCorrigir1]
243
244
245 # In[14]:
246
247
248 for coluna in dados.columns:
249     if coluna not in colunasParaCorrigir1:
250         todasColunas1.insert(0, coluna)
251
252
253 #todasColunas1
254
255 dados = dados.select(todasColunas1)
256 #dataset.groupBy('IN_DEF_INTELECTUAL').count().show()
257
258
259 # In[15]:
260
261
262 from pyspark.sql import functions as f
263
264 colunasParaCorrigir = [
265  'IN_ALUNO_DEF_TGD_SUPER',
266  'IN_DEF_AUDITIVA',
267  'IN_DEF_FISICA',
268  'IN_DEF_INTELECTUAL',
269  'IN_DEF_MULTIPLA',
270  'IN_DEF_SURDEZ',
271  'IN_DEF_SURDOCEGUEIRA',
272  'IN_DEF_BAIXA_VISAO',
273  'IN_DEF_CEGUEIRA',
274  'IN_DEF_SUPERDOTACAO',
275  'IN_TGD_AUTISMO_INFANTIL',
276  'IN_TGD_SINDROME_ASPERGER',
277  'IN_TGD_SINDROME_RETT',
278  'IN_TGD_TRANSTOR_DESINTEGRATIVO',
279
280  'IN_ING_VESTIBULAR',
281  'IN_ING_ENEM',
```

```
282  'IN_ING_AVALIACAO_SERIADA',
283  'IN_ING_SELECAO_SIMPLIFICADA',
284  'IN_ING_SELECAO_VAGA_REMANESC',
285  'IN_ING_SELECAO_VAGA_PROG_ESPEC',
286  'IN_ING_TRANSF_EXOFFICIO',
287  'IN_ING_DECISAO_JUDICIAL',
288  'IN_ING_CONVENIO_PECG',
289  'IN_RESERVA_VAGAS',
290  'IN_RESERVA_ETNICO',
291  'IN_RESERVA_DEFICIENCIA',
292  'IN_RESERVA_ENSINO_PUBLICO',
293  'IN_RESERVA_RENDA_FAMILIAR',
294  'IN_RESERVA_OUTRA',
295  'IN_FINANC_ESTUDANTIL',
296  'IN_FIN_REEMB_FIES',
297  'IN_FIN_REEMB_ESTADUAL',
298  'IN_FIN_REEMB_MUNICIPAL',
299  'IN_FIN_REEMB_PROG_IES',
300  'IN_FIN_REEMB_ENT_EXTERNA',
301  'IN_FIN_NAOREEMB_PROUNI_INTEGR',
302  'IN_FIN_NAOREEMB_PROUNI_PARCIAL',
303  'IN_FIN_NAOREEMB_ESTADUAL',
304  'IN_FIN_NAOREEMB_MUNICIPAL',
305  'IN_FIN_NAOREEMB_PROG_IES',
306  'IN_FIN_NAOREEMB_ENT_EXTERNA',
307  'IN_APOIO_SOCIAL',
308  'IN_APOIO_ALIMENTACAO',
309  'IN_APOIO_BOLSA_PERMANENCIA',
310  'IN_APOIO_BOLSA_TRABALHO',
311  'IN_APOIO_MATERIAL_DIDATICO',
312  'IN_APOIO_MORADIA',
313  'IN_APOIO_TRANSPORTE',
314  'IN_ATIVIDADE_EXTRACURRICULAR',
315  'IN_COMPL_ESTAGIO',
316  'IN_COMPL_EXTENSAO',
317  'IN_COMPL_MONITORIA',
318  'IN_COMPL_PESQUISA',
319  'IN_BOLSA_ESTAGIO',
320  'IN_BOLSA_EXTENSAO',
321  'IN_BOLSA_MONITORIA',
322  'IN_BOLSA_PESQUISA',
323  'CO_TIPO_ESCOLA_ENS_MEDIO',
```

```
 'IN_ALUNO_PARFOR',

 'IN_MOBILIDADE_ACADEMICA',

 #'IN_MATRICULA',
 'IN_CONCLUINTE',
 'IN_INGRESSO_TOTAL',
 'IN_INGRESSO_VAGA_NOVA',

 'IN_FIN_REEMB_OUTRA',
 'IN_FIN_NAOREEMB_OUTRA'

]

todasColunas = [f.when(f.col(c) == 0, 0).when(f.col(c) == 1 , 1).otherwise(2).alias(c)
                for c in colunasParaCorrigir]


# In[16]:


for coluna in dados.columns:
    if coluna not in colunasParaCorrigir:
        todasColunas.insert(0, coluna)


#todasColunas
dataset = dados.select(todasColunas)
#dataset.groupBy('IN_DEF_INTELECTUAL').count().show()


# ## Balanceamento dos dados:

# In[17]:


dados_not_desistente = dataset.select(todasColunas).where('DESISTENTE = 0')
print('dados nao desistente: '+str(dados_not_desistente.count()))

```

```
363 dados_not_desistente =
       dataset.select(todasColunas).where('DESISTENTE =
       0').sample(withReplacement=True, fraction=0.2382, seed=101)
364 print('dados nao desistente fracionado:
       '+str(dados_not_desistente.count()))
365
366 dados_desistente = dataset.select(todasColunas).where('DESISTENTE =
       1')
367 print('dados desistente: '+str(dados_desistente.count()))
368
369
370 # In[18]:
371
372
373 ######################### UNIR AS BASES APOS BALANCEAMENTO
374 dados = dados_not_desistente.unionAll(dados_desistente)
375
376 dados.count()
377 ########################
378
379
380 # ## Informacoes de correlacao dos atributos:
381
382 # In[19]:
383
384
385 #!pip install plotly
386 #dataset.count()
387
388 #So executar daqui para baixo
389 import plotly.express as px
390 dados_plot = dados
391 fig = px.imshow(dados_plot.toPandas().corr(), text_auto=True)
392 fig.show()
393
394
395 # In[20]:
396
397
398 import pandas as pd
399 pd.set_option('display.max_columns',  None)
400 pd.set_option('display.max_rows',  None)
```

```
401
402 dados_plot.toPandas().corr()
403
404
405 # In[21]:
406
407
408 import pandas as pd
409 import plotly.express as px
410
411 # Configuracao para exibir todas as colunas e linhas
412 pd.set_option('display.max_columns', None)
413 pd.set_option('display.max_rows', None)
414
415 # Calcula a matriz de correlacao
416 correlation_matrix = dados_plot.toPandas().corr()
417
418 # Obtem os 10 pares de atributos com a maior correlacao, excluindo a
    correlacao consigo mesmo
419 top_correlations = (correlation_matrix
420                     .unstack()
421                     .sort_values(ascending=False)
422                     .drop_duplicates()
423                     .head(16)  # Adiciona 1 para incluir a primeira
                           entrada (correlacao consigo mesmo)
424                     .tail(15))  # Exclui a correlacao consigo mesmo
425
426 # Exibe os pares de atributos
427 print("Top 10 pares de atributos com maior correlacao (excluindo a
    correlacao consigo mesmo):")
428 print(top_correlations)
429
430 ##########################
431 # Filtra a matriz de correlacao apenas para os 10 pares de atributos
432 filtered_correlation_matrix =
    correlation_matrix.loc[top_correlations.index.get_level_values(0),
    top_correlations.index.get_level_values(1)]
433
434 # Cria o grafico de heatmap usando Plotly para os 10 pares de
    atributos
435 fig = px.imshow(filtered_correlation_matrix, text_auto=True)
436 fig.show()
```

```
437
438
439 # ## Definindo qual atributo e o label
440
441 # In[19]:
442
443
444 dados = dados.withColumnRenamed('DESISTENTE','label')
445
446
447 # In[20]:
448
449
450 x = dados.columns
451 x.remove('label')
452 len(x)
453
454
455 # ## Preparando vetor de entrada para a construcao dos modelos:
456
457 # In[21]:
458
459
460 from pyspark.ml.feature import VectorAssembler
461
462
463 # In[22]:
464
465
466 assembler = VectorAssembler(inputCols=x, outputCol='features')
467
468
469 # In[23]:
470
471
472 dados_preparados = assembler.setHandleInvalid("skip").
473 transform(dados).select('features','label')
474
475
476 # In[27]:
477
478
```

```
dados_preparados.show()


# ## Dividindo o dataset em Treino e Teste:

# In[24]:


SEED = 101


# In[25]:


treino, teste = dados_preparados.randomSplit([0.7, 0.3], seed=SEED)


# In[27]:


treino.show()


# In[28]:


treino.count()


# In[29]:


teste.count()


# # LogisticRegression

# In[26]:


from pyspark.ml.classification import LogisticRegression

```

```
521
522 # In[27]:
523
524
525 lr = LogisticRegression()
526
527
528 # In[35]:
529
530
531 from pyspark.ml import Pipeline
532
533 # Criar um pipeline, para que o CrossValidator utilize as mesmas
       etapas de pre-processamento com os mesmos dados utilizados
534 # para a geracao no modelo no pyspark
535 pipeline = Pipeline(stages=[assembler, lr])
536
537
538 # In[36]:
539
540
541 #Esse CrossValidator nao e para ajuste de hiperparametros, pois o
       pyspark ja faz isso,
542 # esse CrossValidator para captura de uma media de Acuracia dada por
       ele para saber se
543 # o modelo gerado no pyspark se sobressai sobre a media. Nos
       particionamos nossa base em
544 # 70% de treino e 30% teste de maneira que o modelo sera validado
       com esses dados de teste.
545 from sklearn.model_selection import cross_val_score
546 from sklearn.pipeline import Pipeline
547 from sklearn.preprocessing import StandardScaler
548 from sklearn.linear_model import LogisticRegression
549 from pyspark.ml.feature import VectorAssembler
550
551 # Converter DataFrame PySpark para um DataFrame Pandas
552 df_pandas = treino.toPandas()
553
554 # Extrair features e labels do DataFrame Pandas
555 X = df_pandas['features'].tolist()
556 y = df_pandas['label'].tolist()
557
```

```
# Criar um pipeline com um modelo de regressao logistica e um scaler
pipeline = Pipeline([
    ('scaler', StandardScaler()),  # Scaler direto, sem o
       VectorAssembler
    ('lr', LogisticRegression(max_iter=1000))
])

# Realizar a validacao cruzada usando Scikit-Learn
cross_val_result = cross_val_score(pipeline, X, y, cv=10)  # Voce
   pode ajustar o numero de folds conforme necessario

# Imprimir os resultados da validacao cruzada
print("Resultados da validacao cruzada:", cross_val_result)
print("Acuracia media:", cross_val_result.mean())


# In[28]:


modelo_lr = lr.fit(treino)


# In[29]:


import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Exibe os coeficientes atribuidos a cada atributo no modelo
coefficients = modelo_lr.coefficients

# Monta o dicionario, x e a lista de atributos que foi gerada
   algumas linhas acima
coefficients_dict = dict(zip(x, coefficients))

coefficients_df = pd.DataFrame({'Attribute': x, 'Coefficient':
   coefficients})
print(coefficients_df)

# ordene os atributos por ordem decrescente de importancia
```

```
595 coefficients_df.sort_values(by="Coefficient", ascending=False,
       inplace=True)
596
597 # divida as variaveis em positivas e negativas
598 positive_coefficients =
       coefficients_df[coefficients_df["Coefficient"] > 0].head(10)
599 negative_coefficients =
       coefficients_df[coefficients_df["Coefficient"] < 0].tail(10)
600
601 # combine as variaveis positivas e negativas
602 display_coefficients = pd.concat([positive_coefficients,
       negative_coefficients])
603
604 # plote o grafico de barras horizontais
605 # Em red sao as variaveis que influenciam no resultado negativo, ou
       seja, influenciam em o aluno nao desistir
606 # Em red green as variaveis que influenciam no resultado positivo,
       ou seja, influenciam em o aluno desistir
607 colors = np.array(["green"]*10 + ["red"]*10)
608 plt.barh(y=display_coefficients["Attribute"],
       width=display_coefficients["Coefficient"], height=0.8,
       color=colors)
609
610 # defina o titulo do grafico e os rotulos dos eixos
611 plt.title("Importancia dos Atributos")
612 plt.xlabel("Coeficiente")
613 plt.ylabel("Atributo")
614
615 # exiba o grafico
616 plt.show()
617
618
619 # In[30]:
620
621
622 previsoes_lr_teste = modelo_lr.transform(teste)
623
624
625 # In[40]:
626
627
628 previsoes_lr_teste.show()
```

```
629
630
631 # In[31]:
632
633
634 resumo_lr_treino = modelo_lr.summary
635
636
637 # In[32]:
638
639
640 print("Acuracia: %f" % resumo_lr_treino.accuracy)
641 print("Precisao: %f" % resumo_lr_treino.precisionByLabel[1])
642 print("Recall (Sensibilidade): %f"%
       resumo_lr_treino.recallByLabel[1])
643 print("F1: %f"% resumo_lr_treino.fMeasureByLabel()[1])
644
645
646 # ## Trazendo as metricas dos dados de teste usados de entrada para
       o modelo:
647
648 # In[33]:
649
650
651 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
652
653
654 # In[34]:
655
656
657 evaluator = MulticlassClassificationEvaluator()
658
659
660 # In[35]:
661
662
663 evaluator.evaluate(previsoes_lr_teste, {evaluator.metricName:
       'accuracy'})
664
665
666 # In[36]:
667
```

```
668
669 tp = evaluator.evaluate(previsoes_lr_teste, {evaluator.metricName:
       'truePositiveRateByLabel', evaluator.metricLabel: 1.0})
670 tn = evaluator.evaluate(previsoes_lr_teste, {evaluator.metricName:
       'truePositiveRateByLabel', evaluator.metricLabel: 0.0})
671 fp = evaluator.evaluate(previsoes_lr_teste, {evaluator.metricName:
       'falsePositiveRateByLabel', evaluator.metricLabel: 1.0})
672 fn = evaluator.evaluate(previsoes_lr_teste, {evaluator.metricName:
       'falsePositiveRateByLabel', evaluator.metricLabel: 0.0})
673
674
675 # In[37]:
676
677
678 print("VP: %f"% tp)
679 print("VN: %f"% tn)
680 print("FP: %f"% fp)
681 print("FN: %f"% fn)
682
683
684 # In[38]:
685
686
687 print('\033[1m'+'                          Predito\n'+'\033[0m')
688 print('      TP: ' + str(tp) +' '+ str(fp))
689 print('\033[1m'+'Real'+'\033[0m')
690 print('      TN: ' + str(fn) +' '+ str(tn))
691
692
693 # In[39]:
694
695
696 accuracy = (tp+tn)/(tp+tn+fp+fn)
697 print("Acuracia: %f" % accuracy)
698
699 precisao = (tp)/(tp+fp)
700 print("Precisao: %f" % precisao)
701
702 #Metrica que avalia a capacidade de classificar corretamente
703 #resultados classificados como positivos
704 sensibilidade = (tp)/(tp+fn)
705 print("Sensibilidade (Recall): %f" % sensibilidade)
```

```
706
707 #Metrica que avalia a capacidade de classificar corretamente
708 #resultados classificados como negativos
709 especificidade = (tn)/(tn+fp)
710 print("Especificidade: %f" % especificidade)
711
712
713 # # Decision Tree
714
715 # In[40]:
716
717
718 from pyspark.ml.classification import DecisionTreeClassifier
719
720
721 # In[41]:
722
723
724 dt = DecisionTreeClassifier(seed=SEED)
725
726
727 # In[52]:
728
729
730 from pyspark.ml import Pipeline
731
732 # Criar um pipeline, para que o CrossValidator utilize as mesmas
       etapas de pre-processamento com os mesmos dados utilizados
733 # para a geracao no modelo no pyspark
734 pipeline = Pipeline(stages=[assembler, dt])
735
736
737 # In[53]:
738
739
740 #Esse CrossValidator nao e para ajuste de hiperparametros, pois o
       pyspark ja faz isso,
741 # esse CrossValidator para captura de uma media de Acuracia dada por
       ele para saber se
742 # o modelo gerado no pyspark se sobressai sobre a media. Nos
       particionamos nossa base em
```

```
743 # 70% de treino e 30% teste de maneira que o modelo sera validado
        com esses dados de teste.
744 from sklearn.model_selection import cross_val_score
745 from sklearn.pipeline import Pipeline
746 from sklearn.preprocessing import StandardScaler
747 from sklearn.tree import DecisionTreeClassifier
748 from pyspark.ml.feature import VectorAssembler
749
750 # Converter DataFrame PySpark para um DataFrame Pandas
751 df_pandas = treino.toPandas()
752
753 # Extrair features e labels do DataFrame Pandas
754 X = df_pandas['features'].tolist()
755 y = df_pandas['label'].tolist()
756
757 # Criar um pipeline com uma arvore de decisao e um scaler
758 pipeline = Pipeline([
759     ('scaler', StandardScaler()),  # Scaler direto, sem o
            VectorAssembler
760     ('dt', DecisionTreeClassifier())
761 ])
762
763 # Realizar a validacao cruzada usando Scikit-Learn
764 cross_val_result = cross_val_score(pipeline, X, y, cv=10)  # Voce
        pode ajustar o numero de folds conforme necessario
765
766 # Imprimir os resultados da validacao cruzada
767 print("Resultados da validacao cruzada:", cross_val_result)
768 print("Acuracia media:", cross_val_result.mean())
769
770
771 # In[42]:
772
773
774 modelo_dt = dt.fit(treino)
775
776
777 # In[43]:
778
779
780 import pandas as pd
781 import matplotlib.pyplot as plt
```

```
782 import numpy as np
783
784 # Exibe os coeficientes atribuidos a cada atributo no modelo
785 importances = modelo_dt.featureImportances
786
787 # Monta o dicionario, x e a lista de atributos que foi gerada
    algumas linhas acima
788 importances_dict = dict(zip(x, importances))
789
790 importances_df = pd.DataFrame({'Attribute': x, 'Importance':
    importances})
791 print(importances_df)
792
793 # ordene os atributos por ordem decrescente de importancia
794 importances_df.sort_values(by="Importance", ascending=False,
    inplace=True)
795
796 # divida as variaveis em positivas e negativas
797 positive_coefficients = importances_df.head(10)
798 negative_coefficients = importances_df.tail(10)
799
800 # combine as variaveis positivas e negativas
801 display_coefficients = pd.concat([positive_coefficients,
    negative_coefficients])
802
803 # plote o grafico de barras horizontais
804 colors = ["green"] * 10 + ["red"] * 10
805 plt.barh(y=display_coefficients["Attribute"],
    width=display_coefficients["Importance"], height=0.8,
    color=colors)
806
807 # defina o titulo do grafico e os rotulos dos eixos
808 plt.title("Importancia dos Atributos")
809 plt.xlabel("Importancia")
810 plt.ylabel("Atributo")
811
812 # exiba o grafico
813 plt.show()
814
815
816 # In[44]:
817
```

```
818
819 previsoes_dt_teste = modelo_dt.transform(teste)
820
821
822 # In[57]:
823
824
825 previsoes_dt_teste.show()
826
827
828 # In[58]:
829
830
831 #resumo_dt_treino = modelo_dt.summary
832
833
834 # # Trazendo as metricas dos dados de teste usados de entrada para o
       modelo:
835
836 # In[45]:
837
838
839 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
840
841
842 # In[46]:
843
844
845 evaluator = MulticlassClassificationEvaluator()
846
847
848 # In[47]:
849
850
851 evaluator.evaluate(previsoes_dt_teste, {evaluator.metricName:
       'accuracy'})
852
853
854 # In[48]:
855
856
```

```
857 tp = evaluator.evaluate(previsoes_dt_teste, {evaluator.metricName:
       'truePositiveRateByLabel', evaluator.metricLabel: 1.0})
858 tn = evaluator.evaluate(previsoes_dt_teste, {evaluator.metricName:
       'truePositiveRateByLabel', evaluator.metricLabel: 0.0})
859 fp = evaluator.evaluate(previsoes_dt_teste, {evaluator.metricName:
       'falsePositiveRateByLabel', evaluator.metricLabel: 1.0})
860 fn = evaluator.evaluate(previsoes_dt_teste, {evaluator.metricName:
       'falsePositiveRateByLabel', evaluator.metricLabel: 0.0})
861
862
863 # In[49]:
864
865
866 print("VP: %f"% tp)
867 print("VN: %f"% tn)
868 print("FP: %f"% fp)
869 print("FN: %f"% fn)
870
871
872 # In[50]:
873
874
875 print('\033[1m'+'           Predito\n'+'\033[0m')
876 print('      TP: ' + str(tp) +' '+ str(fp))
877 print('\033[1m'+'Real'+'\033[0m')
878 print('      TN: ' + str(fn) +' '+ str(tn))
879
880
881 # In[51]:
882
883
884 accuracy = (tp+tn)/(tp+tn+fp+fn)
885 print("Acuracia: %f" % accuracy)
886
887 precisao = (tp)/(tp+fp)
888 print("Precisao: %f" % precisao)
889
890 #Metrica que avalia a capacidade de classificar corretamente
891 #resultados classificados como positivos
892 sensibilidade = (tp)/(tp+fn)
893 print("Sensibilidade (Recall): %f" % sensibilidade)
894
```

```
895 #Metrica que avalia a capacidade de classificar corretamente
896 #resultados classificados como negativos
897 especificidade = (tn)/(tn+fp)
898 print("Especificidade: %f" % especificidade)
899
900
901 # # Random Forest
902
903 # In[52]:
904
905
906 from pyspark.ml.classification import RandomForestClassifier
907
908
909 # In[53]:
910
911
912 rf = RandomForestClassifier(seed=SEED)
913
914
915 # In[68]:
916
917
918 from pyspark.ml import Pipeline
919
920 # Criar um pipeline, para que o CrossValidator utilize as mesmas
       etapas de pre-processamento com os mesmos dados utilizados
921 # para a geracao no modelo no pyspark
922
923 pipeline = Pipeline(stages=[assembler, rf])
924
925
926 # In[69]:
927
928
929 #Esse CrossValidator nao e para ajuste de hiperparametros, pois o
       pyspark ja faz isso,
930 # esse CrossValidator para captura de uma media de Acuracia dada por
       ele para saber se
931 # o modelo gerado no pyspark se sobressai sobre a media. Nos
       particionamos nossa base em
```

```
932 # 70% de treino e 30% teste de maneira que o modelo sera validado
    com esses dados de teste.
933 from sklearn.model_selection import cross_val_score
934 from sklearn.pipeline import Pipeline
935 from sklearn.preprocessing import StandardScaler
936 from sklearn.ensemble import RandomForestClassifier  # Importar o
    RandomForestClassifier do scikit-learn
937 from pyspark.ml.feature import VectorAssembler
938
939 # Converter DataFrame PySpark para um DataFrame Pandas
940 df_pandas = treino.toPandas()
941
942 # Extrair features e labels do DataFrame Pandas
943 X = df_pandas['features'].tolist()
944 y = df_pandas['label'].tolist()
945
946 # Criar um pipeline com um modelo de Random Forest e um scaler
947 pipeline = Pipeline([
948     ('scaler', StandardScaler()),  # Scaler direto, sem o
        VectorAssembler
949     ('rf', RandomForestClassifier(n_estimators=100,
        random_state=42))  # Numero de arvores pode ser ajustado
        conforme necessario
950 ])
951
952 # Realizar a validacao cruzada usando Scikit-Learn
953 cross_val_result = cross_val_score(pipeline, X, y, cv=10)  # Voce
    pode ajustar o numero de folds conforme necessario
954
955 # Imprimir os resultados da validacao cruzada
956 print("Resultados da validacao cruzada:", cross_val_result)
957 print("Acuracia media:", cross_val_result.mean())
958
959
960 # In[54]:
961
962
963 modelo_rf = rf.fit(treino)
964
965
966 # In[55]:
967
```

```
968
969 import pandas as pd
970 import matplotlib.pyplot as plt
971 import numpy as np
972
973 # Exibe os coeficientes atribuidos a cada atributo no modelo
974 importances = modelo_rf.featureImportances
975
976 # Monta o dicionario, x e a lista de atributos que foi gerada
    algumas linhas acima
977 importances_dict = dict(zip(x, importances))
978
979 importances_rf = pd.DataFrame({'Attribute': x, 'Importance':
    importances})
980 print(importances_rf)
981
982 # ordene os atributos por ordem decrescente de importancia
983 importances_rf.sort_values(by="Importance", ascending=False,
    inplace=True)
984
985 # divida as variaveis em positivas e negativas
986 positive_coefficients = importances_rf.head(10)
987 negative_coefficients = importances_rf.tail(10)
988
989 # combine as variaveis positivas e negativas
990 display_coefficients = pd.concat([positive_coefficients,
    negative_coefficients])
991
992 # plote o grafico de barras horizontais
993 colors = ["green"] * 10 + ["red"] * 10
994 plt.barh(y=display_coefficients["Attribute"],
    width=display_coefficients["Importance"], height=0.8,
    color=colors)
995
996 # defina o titulo do grafico e os rotulos dos eixos
997 plt.title("Importancia dos Atributos")
998 plt.xlabel("Importancia")
999 plt.ylabel("Atributo")
1000
1001 # exiba o grafico
1002 plt.show()
1003
```

```
# In[56]:


previsoes_rf_teste = modelo_rf.transform(teste)


# In[73]:


previsoes_rf_teste.show()


# In[74]:


#resumo_rf_treino = modelo_rf.summary


# In[57]:


from pyspark.ml.evaluation import MulticlassClassificationEvaluator


# In[58]:


evaluator = MulticlassClassificationEvaluator()


# In[59]:


evaluator.evaluate(previsoes_rf_teste, {evaluator.metricName:
    'accuracy'})


# In[60]:


```

```
tp = evaluator.evaluate(previsoes_rf_teste, {evaluator.metricName:
    'truePositiveRateByLabel', evaluator.metricLabel: 1.0})
tn = evaluator.evaluate(previsoes_rf_teste, {evaluator.metricName:
    'truePositiveRateByLabel', evaluator.metricLabel: 0.0})
fp = evaluator.evaluate(previsoes_rf_teste, {evaluator.metricName:
    'falsePositiveRateByLabel', evaluator.metricLabel: 1.0})
fn = evaluator.evaluate(previsoes_rf_teste, {evaluator.metricName:
    'falsePositiveRateByLabel', evaluator.metricLabel: 0.0})


# In[61]:


print("VP: %f"% tp)
print("VN: %f"% tn)
print("FP: %f"% fp)
print("FN: %f"% fn)


# In[62]:


print('\033[1m'+'           Predito\n'+'\033[0m')
print('      TP: ' + str(tp) +' '+ str(fp))
print('\033[1m'+'Real'+'\033[0m')
print('      TN: ' + str(fn) +' '+ str(tn))


# In[63]:


accuracy = (tp+tn)/(tp+tn+fp+fn)
print("Acuracia: %f" % accuracy)

precisao = (tp)/(tp+fp)
print("Precisao: %f" % precisao)

#Metrica que avalia a capacidade de classificar corretamente
#resultados classificados como positivos
sensibilidade = (tp)/(tp+fn)
print("Sensibilidade (Recall): %f" % sensibilidade)

```

```
1082 #Metrica que avalia a capacidade de classificar corretamente
1083 #resultados classificados como negativos
1084 especificidade = (tn)/(tn+fp)
1085 print("Especificidade: %f" % especificidade)
1086
1087
1088 # # SVM
1089
1090 # In[64]:
1091
1092
1093 from pyspark.ml.classification import LinearSVC
1094
1095
1096 # In[65]:
1097
1098
1099 svm = LinearSVC(maxIter=10, regParam=0.1)
1100
1101
1102 # In[28]:
1103
1104
1105 from pyspark.ml import Pipeline
1106
1107 # Criar um pipeline, para que o CrossValidator utilize as mesmas
        etapas de pre-processamento com os mesmos dados utilizados
1108 # para a geracao no modelo no pyspark
1109
1110 pipeline = Pipeline(stages=[assembler, svm])
1111
1112
1113 # In[ ]:
1114
1115
1116 #Esse CrossValidator nao e para ajuste de hiperparametros, pois o
        pyspark ja faz isso,
1117 # esse CrossValidator para captura de uma media de Acuracia dada por
        ele para saber se
1118 # o modelo gerado no pyspark se sobressai sobre a media. Nos
        particionamos nossa base em
```

```
1119 # 70% de treino e 30% teste de maneira que o modelo sera validado
        com esses dados de teste.
1120 from sklearn.model_selection import cross_val_score
1121 from sklearn.pipeline import Pipeline
1122 from sklearn.preprocessing import StandardScaler
1123 from sklearn.svm import SVC  # Importar o Support Vector Classifier
        (SVC) do scikit-learn
1124 from pyspark.ml.feature import VectorAssembler
1125
1126 # Converter DataFrame PySpark para um DataFrame Pandas
1127 df_pandas = treino.toPandas()
1128
1129 # Extrair features e labels do DataFrame Pandas
1130 X = df_pandas['features'].tolist()
1131 y = df_pandas['label'].tolist()
1132
1133 # Criar um pipeline com um modelo de SVM e um scaler
1134 pipeline = Pipeline([
1135     ('scaler', StandardScaler()),
1136     ('svm', SVC(kernel='linear', C=1.0))  # Kernel e outros
           parametros podem ser ajustados conforme necessario
1137 ])
1138
1139 # Realizar a validacao cruzada usando Scikit-Learn
1140 cross_val_result = cross_val_score(pipeline, X, y, cv=10)
1141
1142 # Imprimir os resultados da validacao cruzada
1143 print("Resultados da validacao cruzada:", cross_val_result)
1144 print("Acuracia media:", cross_val_result.mean())
1145
1146
1147 # In[66]:
1148
1149
1150 modelo_svm = svm.fit(treino)
1151
1152
1153 # In[67]:
1154
1155
1156 import pandas as pd
1157 import matplotlib.pyplot as plt
```

```
1158 import numpy as np
1159
1160 # Exibe os coeficientes atribuidos a cada atributo no modelo
1161 coefficients = modelo_svm.coefficients
1162
1163 # Monta o dicionario, x e a lista de atributos que foi gerada
        algumas linhas acima
1164 coefficients_dict = dict(zip(x, coefficients))
1165
1166 coefficients_df = pd.DataFrame({'Attribute': x, 'Coefficient':
        coefficients})
1167 print(coefficients_df)
1168
1169 # ordene os atributos por ordem decrescente de importancia
1170 coefficients_df.sort_values(by="Coefficient", ascending=False,
        inplace=True)
1171
1172 # divida as variaveis em positivas e negativas
1173 positive_coefficients =
        coefficients_df[coefficients_df["Coefficient"] > 0].head(10)
1174 negative_coefficients =
        coefficients_df[coefficients_df["Coefficient"] < 0].tail(10)
1175
1176 # combine as variaveis positivas e negativas
1177 display_coefficients = pd.concat([positive_coefficients,
        negative_coefficients])
1178
1179 # plote o grafico de barras horizontais
1180 # Em red sao as variaveis que influenciam no resultado negativo, ou
        seja, influenciam em o aluno nao desistir
1181 # Em red green as variaveis que influenciam no resultado positivo,
        ou seja, influenciam em o aluno desistir
1182 colors = np.array(["green"]*10 + ["red"]*10)
1183 plt.barh(y=display_coefficients["Attribute"],
        width=display_coefficients["Coefficient"], height=0.8,
        color=colors)
1184
1185 # defina o titulo do grafico e os rotulos dos eixos
1186 plt.title("Importancia dos Atributos")
1187 plt.xlabel("Coeficiente")
1188 plt.ylabel("Atributo")
1189
```

```
# exiba o grafico
plt.show()


# In[68]:


previsoes_svm_teste = modelo_svm.transform(teste)


# In[69]:


previsoes_svm_teste.show()


# In[ ]:


#resumo_rf_treino = modelo_rf.summary


# In[70]:


from pyspark.ml.evaluation import MulticlassClassificationEvaluator


# In[71]:


evaluator = MulticlassClassificationEvaluator()


# In[72]:


evaluator.evaluate(previsoes_svm_teste, {evaluator.metricName:
    'accuracy'})


# In[73]:
```

```
1231
1232
1233 tp = evaluator.evaluate(previsoes_svm_teste, {evaluator.metricName:
        'truePositiveRateByLabel', evaluator.metricLabel: 1.0})
1234 tn = evaluator.evaluate(previsoes_svm_teste, {evaluator.metricName:
        'truePositiveRateByLabel', evaluator.metricLabel: 0.0})
1235 fp = evaluator.evaluate(previsoes_svm_teste, {evaluator.metricName:
        'falsePositiveRateByLabel', evaluator.metricLabel: 1.0})
1236 fn = evaluator.evaluate(previsoes_svm_teste, {evaluator.metricName:
        'falsePositiveRateByLabel', evaluator.metricLabel: 0.0})
1237
1238
1239 # In[74]:
1240
1241
1242 print("VP: %f"% tp)
1243 print("VN: %f"% tn)
1244 print("FP: %f"% fp)
1245 print("FN: %f"% fn)
1246
1247
1248 # In[75]:
1249
1250
1251 print('\033[1m'+'           Predito\n'+'\033[0m')
1252 print('      TP: ' + str(tp) +' '+ str(fp))
1253 print('\033[1m'+'Real'+'\033[0m')
1254 print('      TN: ' + str(fn) +' '+ str(tn))
1255
1256
1257 # In[76]:
1258
1259
1260 accuracy = (tp+tn)/(tp+tn+fp+fn)
1261 print("Acuracia: %f" % accuracy)
1262
1263 precisao = (tp)/(tp+fp)
1264 print("Precisao: %f" % precisao)
1265
1266 #Metrica que avalia a capacidade de classificar corretamente
1267 #resultados classificados como positivos
1268 sensibilidade = (tp)/(tp+fn)
```

```
1269 print("Sensibilidade (Recall): %f" % sensibilidade)
1270
1271 #Metrica que avalia a capacidade de classificar corretamente
1272 #resultados classificados como negativos
1273 especificidade = (tn)/(tn+fp)
1274 print("Especificidade: %f" % especificidade)
1275
1276
1277 # ## Teste de Hipotese T-student
1278
1279 # In[77]:
1280
1281
1282 from itertools import combinations
1283 from scipy.stats import ttest_rel
1284
1285 algoritmos = ['lr', 'dt', 'rf', 'svm']
1286
1287 # Suponha que voce tenha dados de treino e teste definidos como
       treino e teste
1288
1289 for alg1, alg2 in combinations(algoritmos, 2):
1290     observacoes_alg1_acc = []
1291     observacoes_alg2_acc = []
1292
1293     for i in range(10):  # Altere o numero de iteracoes conforme
           necessario
1294         # Treinamento e avaliacao do algoritmo 1
1295         modelo_alg1 = eval(alg1).fit(treino) # Substitua None pelo
               codigo apropriado para treinar alg1
1296         previsoes_alg1_teste = modelo_alg1.transform(teste) #
               Substitua None pelo codigo apropriado para obter
               previsoes do teste
1297         acc_alg1 = evaluator.evaluate(previsoes_alg1_teste,
               {evaluator.metricName: 'accuracy'})  # Substitua None
               pelo codigo apropriado para calcular a acuracia
1298         observacoes_alg1_acc.append(acc_alg1)
1299
1300         # Treinamento e avaliacao do algoritmo 2
1301         modelo_alg2 = eval(alg2).fit(treino)  # Substitua None pelo
               codigo apropriado para treinar alg2
```

```
1302         previsoes_alg2_teste = modelo_alg2.transform(teste)  #
                Substitua None pelo codigo apropriado para obter
                previsoes do teste
1303         acc_alg2 = evaluator.evaluate(previsoes_alg2_teste,
                {evaluator.metricName: 'accuracy'})  # Substitua None
                pelo codigo apropriado para calcular a acuracia
1304         observacoes_alg2_acc.append(acc_alg2)
1305
1306     # Realiza o teste t de Student
1307     t_statistic, p_value = ttest_rel(observacoes_alg1_acc,
            observacoes_alg2_acc)
1308
1309     # Compara o valor-p com o nivel de significancia alfa para
            aceitar ou rejeitar a hipotese nula
1310     alpha = 0.05
1311     h0 = f"Nao ha diferenca significativa no desempenho entre os
            modelos {alg1.upper()} e {alg2.upper()} (ACURACIA)"
1312     h1 = f"Ha diferenca significativa no desempenho entre os modelos
            {alg1.upper()} e {alg2.upper()} (ACURACIA)"
1313
1314     if p_value <= alpha:
1315         print(f"Rejeitamos a hipotese nula: {h0}, com um valor-p de
                {p_value:.3f}.")
1316         #print(f"Ha evidencia estatistica de que {h1}.")
1317     else:
1318         print(f"Aceitamos a hipotese nula: {h0}, com um valor-p de
                {p_value:.3f}.")
```

Listing C.2 – Classificador EAD publica

```
1
2  #!/usr/bin/env python
3  # coding: utf-8
4
5  # # Classificadores de Evasao Estudantil
6
7  # ## Configurando pyspark:
8
9  # In[18]:
10
11
12 #!pip install pyspark
13
```

```
14
15 # In[19]:
16
17
18 from pyspark.sql import SparkSession
19
20
21 # In[20]:
22
23
24 #Define onde sera criado a sessao do spark, nesse caso sera local
25 spark =
      SparkSession.builder.master('local[*]').appName("Classificacao
      com Spark").getOrCreate()
26
27
28 # In[21]:
29
30
31 spark
32
33
34 # ## Carregando os dados do csv:
35
36 # In[22]:
37
38
39 dados = spark.read.csv('ead2016ColunaDesistentePublica.csv',sep='|',
40 header=True, inferSchema=True)
41
42
43 # In[23]:
44
45
46 dados.count()
47
48
49 # In[24]:
50
51
52 #Quantidade de Desistentes e Nao-Desistentes
53 dados.groupBy('DESISTENTE').count().show()
```

```
54
55
56 # ## Selecionando e tratando atributos:
57
58 # In[25]:
59
60
61 dados = dados.drop(*(
62 # 'CO_IES',
63   'NO_IES', #--Campo inutil
64 # 'CO_CATEGORIA_ADMINISTRATIVA',
65   'DS_CATEGORIA_ADMINISTRATIVA', #--Campo inutil
66 # 'CO_ORGANIZACAO_ACADEMICA',
67   'DS_ORGANIZACAO_ACADEMICA', #--Campo inutil
68 # 'CO_CURSO',
69   'NO_CURSO', #--Campo inutil
70 # 'CO_CURSO_POLO',
71 # 'CO_TURNO_ALUNO',
72   'DS_TURNO_ALUNO', #--Campo inutil
73 # 'CO_GRAU_ACADEMICO',
74   'DS_GRAU_ACADEMICO', #--Campo inutil
75 # 'CO_MODALIDADE_ENSINO',
76   'DS_MODALIDADE_ENSINO', #--Campo inutil
77 # 'CO_NIVEL_ACADEMICO',
78   'DS_NIVEL_ACADEMICO', #--Campo inutil
79   'CO_OCDE', #--Nao tratado
80   'NO_OCDE', #--Campo inutil
81 # 'CO_OCDE_AREA_GERAL',
82   'NO_OCDE_AREA_GERAL', #--Campo inutil
83 # 'CO_OCDE_AREA_ESPECIFICA',
84   'NO_OCDE_AREA_ESPECIFICA', #--Campo inutil
85 # 'CO_OCDE_AREA_DETALHADA',
86   'NO_OCDE_AREA_DETALHADA', #--Campo inutil
87 # 'CO_ALUNO_CURSO',
88   'CO_ALUNO_CURSO_ORIGEM', #--Nao tratado
89   'CO_ALUNO', #--Campo inutil
90 # 'CO_COR_RACA_ALUNO',
91   'DS_COR_RACA_ALUNO', #--Campo inutil
92 # 'IN_SEXO_ALUNO',
93   'DS_SEXO_ALUNO', #--Campo inutil
94 # 'NU_ANO_ALUNO_NASC',
95 # 'NU_MES_ALUNO_NASC',
```

```
96  # 'NU_DIA_ALUNO_NASC',
97  # 'NU_IDADE_ALUNO',
98  # 'CO_NACIONALIDADE_ALUNO',
99    'DS_NACIONALIDADE_ALUNO', #--Campo inutil
100 # 'CO_PAIS_ORIGEM_ALUNO',
101   'CO_UF_NASCIMENTO', #--Nao tratado
102   'CO_MUNICIPIO_NASCIMENTO', #--Nao tratado
103 # 'IN_ALUNO_DEF_TGD_SUPER',
104 # 'IN_DEF_AUDITIVA',
105 # 'IN_DEF_FISICA',
106 # 'IN_DEF_INTELECTUAL',
107 # 'IN_DEF_MULTIPLA',
108 # 'IN_DEF_SURDEZ',
109 # 'IN_DEF_SURDOCEGUEIRA',
110 # 'IN_DEF_BAIXA_VISAO',
111 # 'IN_DEF_CEGUEIRA',
112 # 'IN_DEF_SUPERDOTACAO',
113 # 'IN_TGD_AUTISMO_INFANTIL',
114 # 'IN_TGD_SINDROME_ASPERGER',
115 # 'IN_TGD_SINDROME_RETT',
116 # 'IN_TGD_TRANSTOR_DESINTEGRATIVO',
117 # 'CO_ALUNO_SITUACAO',
118   'DS_ALUNO_SITUACAO', #--Campo inutil
119 # 'QT_CARGA_HORARIA_TOTAL',
120 # 'QT_CARGA_HORARIA_INTEG',
121   'DT_INGRESSO_CURSO', #--Nao tratado
122 # 'IN_ING_VESTIBULAR',
123 # 'IN_ING_ENEM',
124 # 'IN_ING_AVALIACAO_SERIADA',
125 # 'IN_ING_SELECAO_SIMPLIFICADA',
126 # 'IN_ING_SELECAO_VAGA_REMANESC',
127 # 'IN_ING_SELECAO_VAGA_PROG_ESPEC',
128 # 'IN_ING_TRANSF_EXOFFICIO',
129 # 'IN_ING_DECISAO_JUDICIAL',
130 # 'IN_ING_CONVENIO_PECG',
131 # 'IN_RESERVA_VAGAS',
132 # 'IN_RESERVA_ETNICO',
133 # 'IN_RESERVA_DEFICIENCIA',
134 # 'IN_RESERVA_ENSINO_PUBLICO',
135 # 'IN_RESERVA_RENDA_FAMILIAR',
136 # 'IN_RESERVA_OUTRA',
137 # 'IN_FINANC_ESTUDANTIL',
```

```
138 # 'IN_FIN_REEMB_FIES',
139 # 'IN_FIN_REEMB_ESTADUAL',
140 # 'IN_FIN_REEMB_MUNICIPAL',
141 # 'IN_FIN_REEMB_PROG_IES',
142 # 'IN_FIN_REEMB_ENT_EXTERNA',
143 # 'IN_FIN_REEMB_OUTRA',
144 # 'IN_FIN_NAOREEMB_PROUNI_INTEGR',
145 # 'IN_FIN_NAOREEMB_PROUNI_PARCIAL',
146 # 'IN_FIN_NAOREEMB_ESTADUAL',
147 # 'IN_FIN_NAOREEMB_MUNICIPAL',
148 # 'IN_FIN_NAOREEMB_PROG_IES',
149 # 'IN_FIN_NAOREEMB_ENT_EXTERNA',
150 # 'IN_FIN_NAOREEMB_OUTRA',
151 # 'IN_APOIO_SOCIAL',
152 # 'IN_APOIO_ALIMENTACAO',
153 # 'IN_APOIO_BOLSA_PERMANENCIA',
154 # 'IN_APOIO_BOLSA_TRABALHO',
155 # 'IN_APOIO_MATERIAL_DIDATICO',
156 # 'IN_APOIO_MORADIA',
157 # 'IN_APOIO_TRANSPORTE',
158 # 'IN_ATIVIDADE_EXTRACURRICULAR',
159 # 'IN_COMPL_ESTAGIO',
160 # 'IN_COMPL_EXTENSAO',
161 # 'IN_COMPL_MONITORIA',
162 # 'IN_COMPL_PESQUISA',
163 # 'IN_BOLSA_ESTAGIO',
164 # 'IN_BOLSA_EXTENSAO',
165 # 'IN_BOLSA_MONITORIA',
166 # 'IN_BOLSA_PESQUISA',
167 # 'CO_TIPO_ESCOLA_ENS_MEDIO',
168 # 'IN_ALUNO_PARFOR',
169  'CO_SEMESTRE_CONCLUSAO', #--Nao tratado
170 # 'CO_SEMESTRE_REFERENCIA',
171 # 'IN_MOBILIDADE_ACADEMICA',
172 # 'CO_MOBILIDADE_ACADEMICA',
173 # 'CO_MOBILIDADE_ACADEMICA_INTERN',
174   'CO_IES_DESTINO', #--Nao tratado
175   'CO_PAIS_DESTINO', #--Nao tratado
176   'IN_MATRICULA', #--Campo inutil
177 # 'IN_CONCLUINTE',
178 # 'IN_INGRESSO_TOTAL',
179 # 'IN_INGRESSO_VAGA_NOVA',
```

```
# 'ANO_INGRESSO',
# 'DESISTENTE'
                    ))


# In[26]:


dados.printSchema()


# In[27]:


#dados.select('IN_DEF_INTELECTUAL').where('IN_DEF_INTELECTUAL is
   null').show()
#dados.groupBy('IN_DEF_INTELECTUAL').count().show()


# ## Formatando Atributos para processamento

# In[28]:


from pyspark.sql import functions as f

colunasParaCorrigir0 = [
'CO_TURNO_ALUNO'
]

todasColunas0 = [f.when(f.col(c) == 1, 1).when(f.col(c) == 2, 2).
                 when(f.col(c) == 3, 3).when(f.col(c) == 4, 4).
                 otherwise(5).alias(c)
                for c in colunasParaCorrigir0]


# In[29]:


for coluna in dados.columns:
    if coluna not in colunasParaCorrigir0:
        todasColunas0.insert(0, coluna)
```

```
221
222
223 #todasColunas0
224
225 dados = dados.select(todasColunas0)
226 #dataset.groupBy('IN_DEF_INTELECTUAL').count().show()
227
228
229 # In[30]:
230
231
232 from pyspark.sql import functions as f
233
234 colunasParaCorrigir1 = [
235  #'CO_SEMESTRE_CONCLUSAO',
236  'CO_SEMESTRE_REFERENCIA',
237  'CO_MOBILIDADE_ACADEMICA',
238  'CO_MOBILIDADE_ACADEMICA_INTERN'
239 ]
240
241 todasColunas1 = [f.when(f.col(c) == 1, 1).when(f.col(c) == 2 ,
       2).otherwise(3).alias(c)
242                 for c in colunasParaCorrigir1]
243
244
245 # In[31]:
246
247
248 for coluna in dados.columns:
249     if coluna not in colunasParaCorrigir1:
250         todasColunas1.insert(0, coluna)
251
252
253 #todasColunas1
254
255 dados = dados.select(todasColunas1)
256 #dataset.groupBy('IN_DEF_INTELECTUAL').count().show()
257
258
259 # In[32]:
260
261
```

```
262 from pyspark.sql import functions as f
263
264 colunasParaCorrigir = [
265  'IN_ALUNO_DEF_TGD_SUPER',
266  'IN_DEF_AUDITIVA',
267  'IN_DEF_FISICA',
268  'IN_DEF_INTELECTUAL',
269  'IN_DEF_MULTIPLA',
270  'IN_DEF_SURDEZ',
271  'IN_DEF_SURDOCEGUEIRA',
272  'IN_DEF_BAIXA_VISAO',
273  'IN_DEF_CEGUEIRA',
274  'IN_DEF_SUPERDOTACAO',
275  'IN_TGD_AUTISMO_INFANTIL',
276  'IN_TGD_SINDROME_ASPERGER',
277  'IN_TGD_SINDROME_RETT',
278  'IN_TGD_TRANSTOR_DESINTEGRATIVO',
279
280  'IN_ING_VESTIBULAR',
281  'IN_ING_ENEM',
282  'IN_ING_AVALIACAO_SERIADA',
283  'IN_ING_SELECAO_SIMPLIFICADA',
284  'IN_ING_SELECAO_VAGA_REMANESC',
285  'IN_ING_SELECAO_VAGA_PROG_ESPEC',
286  'IN_ING_TRANSF_EXOFFICIO',
287  'IN_ING_DECISAO_JUDICIAL',
288  'IN_ING_CONVENIO_PECG',
289  'IN_RESERVA_VAGAS',
290  'IN_RESERVA_ETNICO',
291  'IN_RESERVA_DEFICIENCIA',
292  'IN_RESERVA_ENSINO_PUBLICO',
293  'IN_RESERVA_RENDA_FAMILIAR',
294  'IN_RESERVA_OUTRA',
295  'IN_FINANC_ESTUDANTIL',
296  'IN_FIN_REEMB_FIES',
297  'IN_FIN_REEMB_ESTADUAL',
298  'IN_FIN_REEMB_MUNICIPAL',
299  'IN_FIN_REEMB_PROG_IES',
300  'IN_FIN_REEMB_ENT_EXTERNA',
301  'IN_FIN_NAOREEMB_PROUNI_INTEGR',
302  'IN_FIN_NAOREEMB_PROUNI_PARCIAL',
303  'IN_FIN_NAOREEMB_ESTADUAL',
```

```
304  'IN_FIN_NAOREEMB_MUNICIPAL',
305  'IN_FIN_NAOREEMB_PROG_IES',
306  'IN_FIN_NAOREEMB_ENT_EXTERNA',
307  'IN_APOIO_SOCIAL',
308  'IN_APOIO_ALIMENTACAO',
309  'IN_APOIO_BOLSA_PERMANENCIA',
310  'IN_APOIO_BOLSA_TRABALHO',
311  'IN_APOIO_MATERIAL_DIDATICO',
312  'IN_APOIO_MORADIA',
313  'IN_APOIO_TRANSPORTE',
314  'IN_ATIVIDADE_EXTRACURRICULAR',
315  'IN_COMPL_ESTAGIO',
316  'IN_COMPL_EXTENSAO',
317  'IN_COMPL_MONITORIA',
318  'IN_COMPL_PESQUISA',
319  'IN_BOLSA_ESTAGIO',
320  'IN_BOLSA_EXTENSAO',
321  'IN_BOLSA_MONITORIA',
322  'IN_BOLSA_PESQUISA',
323  'CO_TIPO_ESCOLA_ENS_MEDIO',
324  'IN_ALUNO_PARFOR',
325
326  'IN_MOBILIDADE_ACADEMICA',
327
328  #'IN_MATRICULA',
329  'IN_CONCLUINTE',
330  'IN_INGRESSO_TOTAL',
331  'IN_INGRESSO_VAGA_NOVA',
332
333  'IN_FIN_REEMB_OUTRA',
334  'IN_FIN_NAOREEMB_OUTRA'
335
336 ]
337
338 todasColunas = [f.when(f.col(c) == 0, 0).when(f.col(c) == 1 ,
       1).otherwise(2).alias(c)
339                 for c in colunasParaCorrigir]
340
341
342 # In[33]:
343
344
```

```
345 for coluna in dados.columns:
346     if coluna not in colunasParaCorrigir:
347         todasColunas.insert(0, coluna)
348
349
350 #todasColunas
351 dataset = dados.select(todasColunas)
352 #dataset.groupBy('IN_DEF_INTELECTUAL').count().show()
353
354
355 # ## Balanceamento dos dados:
356
357 # In[34]:
358
359
360 dados_not_desistente =
       dataset.select(todasColunas).where('DESISTENTE = 0')
361 print('dados nao desistente: '+str(dados_not_desistente.count()))
362
363 dados_not_desistente =
       dataset.select(todasColunas).where('DESISTENTE =
       0').sample(withReplacement=True, fraction=0.1336, seed=101)
364 print('dados nao desistente fracionado:
       '+str(dados_not_desistente.count()))
365
366 dados_desistente = dataset.select(todasColunas).where('DESISTENTE =
       1')
367 print('dados desistente: '+str(dados_desistente.count()))
368
369
370 # In[35]:
371
372
373 ######################### UNIR AS BASES APOS BALANCEAMENTO
374 dados = dados_not_desistente.unionAll(dados_desistente)
375
376 dados.count()
377 ########################
378
379
380 # ## Informacoes de correlacao dos atributos:
381
```

```
382 # In[36]:
383
384
385 #So executar daqui para baixo
386 import plotly.express as px
387 dados_plot = dados
388 fig = px.imshow(dados_plot.toPandas().corr(), text_auto=True)
389 fig.show()
390
391
392 # In[37]:
393
394
395 import pandas as pd
396 pd.set_option('display.max_columns',  None)
397 pd.set_option('display.max_rows',  None)
398
399 dados_plot.toPandas().corr()
400
401
402 # In[38]:
403
404
405 import pandas as pd
406 import plotly.express as px
407
408 # Configuracao para exibir todas as colunas e linhas
409 pd.set_option('display.max_columns', None)
410 pd.set_option('display.max_rows', None)
411
412 # Calcula a matriz de correlacao
413 correlation_matrix = dados_plot.toPandas().corr()
414
415 # Obtem os 10 pares de atributos com a maior correlacao, excluindo a
       correlacao consigo mesmo
416 top_correlations = (correlation_matrix
417                     .unstack()
418                     .sort_values(ascending=False)
419                     .drop_duplicates()
420                     .head(16)  # Adiciona 1 para incluir a primeira
                           entrada (correlacao consigo mesmo)
421                     .tail(15))  # Exclui a correlacao consigo mesmo
```

```
422
423 # Exibe os pares de atributos
424 print("Top 10 pares de atributos com maior correlacao (excluindo a
       correlacao consigo mesmo):")
425 print(top_correlations)
426
427 ##########################
428 # Filtra a matriz de correlacao apenas para os 10 pares de atributos
429 filtered_correlation_matrix =
       correlation_matrix.loc[top_correlations.index.get_level_values(0),
       top_correlations.index.get_level_values(1)]
430
431 # Cria o grafico de heatmap usando Plotly para os 10 pares de
       atributos
432 fig = px.imshow(filtered_correlation_matrix, text_auto=True)
433 fig.show()
434
435
436 # ## Definindo qual atributo e o label
437
438 # In[39]:
439
440
441 dados = dados.withColumnRenamed('DESISTENTE','label')
442
443
444 # In[40]:
445
446
447 x = dados.columns
448 x.remove('label')
449 len(x)
450
451
452 # ## Preparando vetor de entrada para a construcao dos modelos:
453
454 # In[41]:
455
456
457 from pyspark.ml.feature import VectorAssembler
458
459
```

```
# In[42]:


assembler = VectorAssembler(inputCols=x, outputCol='features')


# In[43]:


dados_preparados = assembler.setHandleInvalid("skip").
transform(dados).select('features','label')


# In[44]:


dados_preparados.show()


# ## Dividindo o dataset em Treino e Teste:

# In[45]:


SEED = 101


# In[46]:


treino, teste = dados_preparados.randomSplit([0.7, 0.3], seed=SEED)


# In[47]:


treino.show()


# In[48]:


```

```
502 treino.count()
503
504
505 # In[49]:
506
507
508 teste.count()
509
510
511 # # LogisticRegression
512
513 # In[50]:
514
515
516 from pyspark.ml.classification import LogisticRegression
517
518
519 # In[51]:
520
521
522 lr = LogisticRegression()
523
524
525 # In[52]:
526
527
528 from pyspark.ml import Pipeline
529
530 # Criar um pipeline, para que o CrossValidator utilize as mesmas
       etapas de pre-processamento com os mesmos dados utilizados
531 # para a geracao no modelo no pyspark
532 pipeline = Pipeline(stages=[assembler, lr])
533
534
535 # In[53]:
536
537
538 #Esse CrossValidator nao e para ajuste de hiperparametros, pois o
       pyspark ja faz isso,
539 # esse CrossValidator para captura de uma media de Acuracia dada por
       ele para saber se
540 # o modelo gerado no pyspark se sobressai sobre a media. Nos
```

```
    particionamos nossa base em
# 70% de treino e 30% teste de maneira que o modelo sera validado
    com esses dados de teste.
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from pyspark.ml.feature import VectorAssembler

# Converter DataFrame PySpark para um DataFrame Pandas
df_pandas = treino.toPandas()

# Extrair features e labels do DataFrame Pandas
X = df_pandas['features'].tolist()
y = df_pandas['label'].tolist()

# Criar um pipeline com um modelo de regressao logistica e um scaler
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('lr', LogisticRegression(max_iter=1000))
])

# Realizar a validacao cruzada usando Scikit-Learn
cross_val_result = cross_val_score(pipeline, X, y, cv=10)

# Imprimir os resultados da validacao cruzada
print("Resultados da validacao cruzada:", cross_val_result)
print("Acuracia media:", cross_val_result.mean())


# In[54]:


modelo_lr = lr.fit(treino)


# In[55]:


import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

```
581
582 # Exibe os coeficientes atribuidos a cada atributo no modelo
583 coefficients = modelo_lr.coefficients
584
585 # Monta o dicionario, x e a lista de atributos que foi gerada
       algumas linhas acima
586 coefficients_dict = dict(zip(x, coefficients))
587
588 coefficients_df = pd.DataFrame({'Attribute': x, 'Coefficient':
       coefficients})
589 print(coefficients_df)
590
591 # ordena os atributos por ordem decrescente de importancia
592 coefficients_df.sort_values(by="Coefficient", ascending=False,
       inplace=True)
593
594 # divide as variaveis em positivas e negativas
595 positive_coefficients =
       coefficients_df[coefficients_df["Coefficient"] > 0].head(10)
596 negative_coefficients =
       coefficients_df[coefficients_df["Coefficient"] < 0].tail(10)
597
598 # combina as variaveis positivas e negativas
599 display_coefficients = pd.concat([positive_coefficients,
       negative_coefficients])
600
601 # plota o grafico de barras horizontais
602 # Em red sao as variaveis que influenciam no resultado negativo, ou
       seja, influenciam em o aluno nao desistir
603 # Em red green as variaveis que influenciam no resultado positivo,
       ou seja, influenciam em o aluno desistir
604 colors = np.array(["green"]*10 + ["red"]*10)
605 plt.barh(y=display_coefficients["Attribute"],
       width=display_coefficients["Coefficient"], height=0.8,
       color=colors)
606
607 # definindo o titulo do grafico e os rotulos dos eixos
608 plt.title("Importancia dos Atributos")
609 plt.xlabel("Coeficiente")
610 plt.ylabel("Atributo")
611
612 # exibe o grafico
```

```
plt.show()


# In[56]:


previsoes_lr_teste = modelo_lr.transform(teste)


# In[57]:


previsoes_lr_teste.show()


# In[58]:


# A funcao summary fornece estatisticas e informacoes sobre
# o modelo treinado com base nos dados de treinamento
# a previsao com os dados de teste e feita linhas abaixo
resumo_lr_treino = modelo_lr.summary


# In[59]:


print("Acuracia: %f" % resumo_lr_treino.accuracy)
print("Precisao: %f" % resumo_lr_treino.precisionByLabel[1])
print("Recall (Sensibilidade): %f"%
   resumo_lr_treino.recallByLabel[1])
print("F1: %f"% resumo_lr_treino.fMeasureByLabel()[1])


# ## Trazendo as metricas dos dados de teste usados de entrada para
   o modelo:

# In[60]:


from pyspark.ml.evaluation import MulticlassClassificationEvaluator

```

```
653
654 # In[61]:
655
656
657 evaluator = MulticlassClassificationEvaluator()
658
659
660 # In[62]:
661
662
663 evaluator.evaluate(previsoes_lr_teste, {evaluator.metricName:
       'accuracy'})
664
665
666 # In[63]:
667
668
669 tp = evaluator.evaluate(previsoes_lr_teste, {evaluator.metricName:
       'truePositiveRateByLabel', evaluator.metricLabel: 1.0})
670 tn = evaluator.evaluate(previsoes_lr_teste, {evaluator.metricName:
       'truePositiveRateByLabel', evaluator.metricLabel: 0.0})
671 fp = evaluator.evaluate(previsoes_lr_teste, {evaluator.metricName:
       'falsePositiveRateByLabel', evaluator.metricLabel: 1.0})
672 fn = evaluator.evaluate(previsoes_lr_teste, {evaluator.metricName:
       'falsePositiveRateByLabel', evaluator.metricLabel: 0.0})
673
674
675 # In[64]:
676
677
678 print("VP: %f"% tp)
679 print("VN: %f"% tn)
680 print("FP: %f"% fp)
681 print("FN: %f"% fn)
682
683
684 # In[65]:
685
686
687 print('\033[1m'+'                          Predito\n'+'\033[0m')
688 print('      TP: ' + str(tp) +' '+ str(fp))
689 print('\033[1m'+'Real'+'\033[0m')
```

```
690 print('      TN: ' + str(fn) +' '+ str(tn))
691
692
693 # In[66]:
694
695
696 accuracy = (tp+tn)/(tp+tn+fp+fn)
697 print("Acuracia: %f" % accuracy)
698
699 precisao = (tp)/(tp+fp)
700 print("Precisao: %f" % precisao)
701
702 #Metrica que avalia a capacidade de classificar corretamente
703 #resultados classificados como positivos
704 sensibilidade = (tp)/(tp+fn)
705 print("Sensibilidade (Recall): %f" % sensibilidade)
706
707 #Metrica que avalia a capacidade de classificar corretamente
708 #resultados classificados como negativos
709 especificidade = (tn)/(tn+fp)
710 print("Especificidade: %f" % especificidade)
711
712
713 # # Decision Tree
714
715 # In[67]:
716
717
718 from pyspark.ml.classification import DecisionTreeClassifier
719
720
721 # In[68]:
722
723
724 dt = DecisionTreeClassifier(seed=SEED)
725
726
727 # In[69]:
728
729
730 from pyspark.ml import Pipeline
731
```

```
732 # Criar um pipeline, para que o CrossValidator utilize as mesmas
        etapas de pre-processamento com os mesmos dados utilizados
733 # para a geracao no modelo no pyspark
734 pipeline = Pipeline(stages=[assembler, dt])
735
736
737 # In[70]:
738
739
740 #Esse CrossValidator nao e para ajuste de hiperparametros, pois o
        pyspark ja faz isso,
741 # esse CrossValidator para captura de uma media de Acuracia dada por
        ele para saber se
742 # o modelo gerado no pyspark se sobressai sobre a media. Nos
        particionamos nossa base em
743 # 70% de treino e 30% teste de maneira que o modelo sera validado
        com esses dados de teste.
744 from sklearn.model_selection import cross_val_score
745 from sklearn.pipeline import Pipeline
746 from sklearn.preprocessing import StandardScaler
747 from sklearn.tree import DecisionTreeClassifier
748 from pyspark.ml.feature import VectorAssembler
749
750 # Converter DataFrame PySpark para um DataFrame Pandas
751 df_pandas = treino.toPandas()
752
753 # Extrair features e labels do DataFrame Pandas
754 X = df_pandas['features'].tolist()
755 y = df_pandas['label'].tolist()
756
757 # Criar um pipeline com uma arvore de decisao e um scaler
758 pipeline = Pipeline([
759     ('scaler', StandardScaler()),
760     ('dt', DecisionTreeClassifier())
761 ])
762
763 # Realizar a validacao cruzada usando Scikit-Learn
764 cross_val_result = cross_val_score(pipeline, X, y, cv=10)
765
766 # Imprimir os resultados da validacao cruzada
767 print("Resultados da validacao cruzada:", cross_val_result)
768 print("Acuracia media:", cross_val_result.mean())
```

```
769
770
771 # In[71]:
772
773
774 modelo_dt = dt.fit(treino)
775
776
777 # In[72]:
778
779
780 import pandas as pd
781 import matplotlib.pyplot as plt
782 import numpy as np
783
784 # Exibe os coeficientes atribuidos a cada atributo no modelo
785 importances = modelo_dt.featureImportances
786
787 # Monta o dicionario, x e a lista de atributos que foi gerada
       algumas linhas acima
788 importances_dict = dict(zip(x, importances))
789
790 importances_df = pd.DataFrame({'Attribute': x, 'Importance':
       importances})
791 print(importances_df)
792
793 # ordena os atributos por ordem decrescente de importancia
794 importances_df.sort_values(by="Importance", ascending=False,
       inplace=True)
795
796 # divide as variaveis em positivas e negativas
797 positive_coefficients = importances_df.head(10)
798 negative_coefficients = importances_df.tail(10)
799
800 # combina as variaveis positivas e negativas
801 display_coefficients = pd.concat([positive_coefficients,
       negative_coefficients])
802
803 # plota o grafico de barras horizontais
804 colors = ["green"] * 10 + ["red"] * 10
805 plt.barh(y=display_coefficients["Attribute"],
       width=display_coefficients["Importance"], height=0.8,
```

```
    color=colors)
806
807 # definindo o titulo do grafico e os rotulos dos eixos
808 plt.title("Importancia dos Atributos")
809 plt.xlabel("Importancia")
810 plt.ylabel("Atributo")
811
812 # exibe o grafico
813 plt.show()
814
815
816 # In[73]:
817
818
819 previsoes_dt_teste = modelo_dt.transform(teste)
820
821
822 # In[74]:
823
824
825 previsoes_dt_teste.show()
826
827
828 # # Trazendo as metricas dos dados de teste usados de entrada para o
    modelo:
829
830 # In[75]:
831
832
833 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
834
835
836 # In[76]:
837
838
839 evaluator = MulticlassClassificationEvaluator()
840
841
842 # In[77]:
843
844
845 evaluator.evaluate(previsoes_dt_teste, {evaluator.metricName:
```

```
    'accuracy'})
846
847
848 # In[78]:
849
850
851 tp = evaluator.evaluate(previsoes_dt_teste, {evaluator.metricName:
    'truePositiveRateByLabel', evaluator.metricLabel: 1.0})
852 tn = evaluator.evaluate(previsoes_dt_teste, {evaluator.metricName:
    'truePositiveRateByLabel', evaluator.metricLabel: 0.0})
853 fp = evaluator.evaluate(previsoes_dt_teste, {evaluator.metricName:
    'falsePositiveRateByLabel', evaluator.metricLabel: 1.0})
854 fn = evaluator.evaluate(previsoes_dt_teste, {evaluator.metricName:
    'falsePositiveRateByLabel', evaluator.metricLabel: 0.0})
855
856
857 # In[79]:
858
859
860 print("VP: %f"% tp)
861 print("VN: %f"% tn)
862 print("FP: %f"% fp)
863 print("FN: %f"% fn)
864
865
866 # In[80]:
867
868
869 print('\033[1m'+'           Predito\n'+'\033[0m')
870 print('      TP: ' + str(tp) +' '+ str(fp))
871 print('\033[1m'+'Real'+'\033[0m')
872 print('      TN: ' + str(fn) +' '+ str(tn))
873
874
875 # In[81]:
876
877
878 accuracy = (tp+tn)/(tp+tn+fp+fn)
879 print("Acuracia: %f" % accuracy)
880
881 precisao = (tp)/(tp+fp)
882 print("Precisao: %f" % precisao)
```

```
883
884 #Metrica que avalia a capacidade de classificar corretamente
885 #resultados classificados como positivos
886 sensibilidade = (tp)/(tp+fn)
887 print("Sensibilidade (Recall): %f" % sensibilidade)
888
889 #Metrica que avalia a capacidade de classificar corretamente
890 #resultados classificados como negativos
891 especificidade = (tn)/(tn+fp)
892 print("Especificidade: %f" % especificidade)
893
894
895 # # Random Forest
896
897 # In[82]:
898
899
900 from pyspark.ml.classification import RandomForestClassifier
901
902
903 # In[83]:
904
905
906 rf = RandomForestClassifier(seed=SEED)
907
908
909 # In[84]:
910
911
912 from pyspark.ml import Pipeline
913
914 # Criar um pipeline, para que o CrossValidator utilize as mesmas
       etapas de pre-processamento com os mesmos dados utilizados
915 # para a geracao no modelo no pyspark
916
917 pipeline = Pipeline(stages=[assembler, rf])
918
919
920 # In[85]:
921
922
923 #Esse CrossValidator nao e para ajuste de hiperparametros, pois o
```

```
    pyspark ja faz isso,
924 # esse CrossValidator para captura de uma media de Acuracia dada por
    ele para saber se
925 # o modelo gerado no pyspark se sobressai sobre a media. Nos
    particionamos nossa base em
926 # 70% de treino e 30% teste de maneira que o modelo sera validado
    com esses dados de teste.
927 from sklearn.model_selection import cross_val_score
928 from sklearn.pipeline import Pipeline
929 from sklearn.preprocessing import StandardScaler
930 from sklearn.ensemble import RandomForestClassifier  # Importar o
    RandomForestClassifier do scikit-learn
931 from pyspark.ml.feature import VectorAssembler
932
933 # Converter DataFrame PySpark para um DataFrame Pandas
934 df_pandas = treino.toPandas()
935
936 # Extrair features e labels do DataFrame Pandas
937 X = df_pandas['features'].tolist()
938 y = df_pandas['label'].tolist()
939
940 # Criar um pipeline com um modelo de Random Forest e um scaler
941 pipeline = Pipeline([
942     ('scaler', StandardScaler()),
943     ('rf', RandomForestClassifier(n_estimators=100,
        random_state=42))  # Numero de arvores pode ser ajustado
        conforme necessario
944 ])
945
946 # Realizar a validacao cruzada usando Scikit-Learn
947 cross_val_result = cross_val_score(pipeline, X, y, cv=10)
948
949 # Imprimir os resultados da validacao cruzada
950 print("Resultados da validacao cruzada:", cross_val_result)
951 print("Acuracia media:", cross_val_result.mean())
952
953
954 # In[86]:
955
956
957 modelo_rf = rf.fit(treino)
958
```

```
959
960 # In[87]:
961
962
963 import pandas as pd
964 import matplotlib.pyplot as plt
965 import numpy as np
966
967 # Exibe os coeficientes atribuidos a cada atributo no modelo
968 importances = modelo_rf.featureImportances
969
970 # Monta o dicionario, x e a lista de atributos que foi gerada
       algumas linhas acima
971 importances_dict = dict(zip(x, importances))
972
973 importances_rf = pd.DataFrame({'Attribute': x, 'Importance':
       importances})
974 print(importances_rf)
975
976 # ordena os atributos por ordem decrescente de importancia
977 importances_rf.sort_values(by="Importance", ascending=False,
       inplace=True)
978
979 # divide as variaveis em positivas e negativas
980 positive_coefficients = importances_rf.head(10)
981 negative_coefficients = importances_rf.tail(10)
982
983 # combina as variaveis positivas e negativas
984 display_coefficients = pd.concat([positive_coefficients,
       negative_coefficients])
985
986 # plota o grafico de barras horizontais
987 colors = ["green"] * 10 + ["red"] * 10
988 plt.barh(y=display_coefficients["Attribute"],
       width=display_coefficients["Importance"], height=0.8,
       color=colors)
989
990 # definindo o titulo do grafico e os rotulos dos eixos
991 plt.title("Importancia dos Atributos")
992 plt.xlabel("Importancia")
993 plt.ylabel("Atributo")
994
```

```
995 # exibe o grafico
996 plt.show()
997
998
999 # In[88]:
1000
1001
1002 previsoes_rf_teste = modelo_rf.transform(teste)
1003
1004
1005 # In[89]:
1006
1007
1008 previsoes_rf_teste.show()
1009
1010
1011 # In[90]:
1012
1013
1014 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
1015
1016
1017 # In[91]:
1018
1019
1020 evaluator = MulticlassClassificationEvaluator()
1021
1022
1023 # In[92]:
1024
1025
1026 evaluator.evaluate(previsoes_rf_teste, {evaluator.metricName:
        'accuracy'})
1027
1028
1029 # In[93]:
1030
1031
1032 tp = evaluator.evaluate(previsoes_rf_teste, {evaluator.metricName:
        'truePositiveRateByLabel', evaluator.metricLabel: 1.0})
1033 tn = evaluator.evaluate(previsoes_rf_teste, {evaluator.metricName:
        'truePositiveRateByLabel', evaluator.metricLabel: 0.0})
```

```
fp = evaluator.evaluate(previsoes_rf_teste, {evaluator.metricName:
    'falsePositiveRateByLabel', evaluator.metricLabel: 1.0})
fn = evaluator.evaluate(previsoes_rf_teste, {evaluator.metricName:
    'falsePositiveRateByLabel', evaluator.metricLabel: 0.0})


# In[94]:


print("VP: %f"% tp)
print("VN: %f"% tn)
print("FP: %f"% fp)
print("FN: %f"% fn)


# In[95]:


print('\033[1m'+'           Predito\n'+'\033[0m')
print('      TP: ' + str(tp) +' '+ str(fp))
print('\033[1m'+'Real'+'\033[0m')
print('      TN: ' + str(fn) +' '+ str(tn))


# In[96]:


accuracy = (tp+tn)/(tp+tn+fp+fn)
print("Acuracia: %f" % accuracy)

precisao = (tp)/(tp+fp)
print("Precisao: %f" % precisao)

#Metrica que avalia a capacidade de classificar corretamente
#resultados classificados como positivos
sensibilidade = (tp)/(tp+fn)
print("Sensibilidade (Recall): %f" % sensibilidade)

#Metrica que avalia a capacidade de classificar corretamente
#resultados classificados como negativos
especificidade = (tn)/(tn+fp)
print("Especificidade: %f" % especificidade)
```

```
1074
1075
1076 # # SVM
1077
1078 # In[97]:
1079
1080
1081 from pyspark.ml.classification import LinearSVC
1082
1083
1084 # In[98]:
1085
1086
1087 svm = LinearSVC(maxIter=10, regParam=0.1)
1088
1089
1090 # In[99]:
1091
1092
1093 from pyspark.ml import Pipeline
1094
1095 # Criar um pipeline, para que o CrossValidator utilize as mesmas
       etapas de pre-processamento com os mesmos dados utilizados
1096 # para a geracao no modelo no pyspark
1097
1098 pipeline = Pipeline(stages=[assembler, svm])
1099
1100
1101 # In[100]:
1102
1103
1104 #Esse CrossValidator nao e para ajuste de hiperparametros, pois o
       pyspark ja faz isso,
1105 # esse CrossValidator para captura de uma media de Acuracia dada por
       ele para saber se
1106 # o modelo gerado no pyspark se sobressai sobre a media. Nos
       particionamos nossa base em
1107 # 70% de treino e 30% teste de maneira que o modelo sera validado
       com esses dados de teste.
1108 from sklearn.model_selection import cross_val_score
1109 from sklearn.pipeline import Pipeline
1110 from sklearn.preprocessing import StandardScaler
```

```
1111 from sklearn.svm import SVC  # Importar o Support Vector Classifier
        (SVC) do scikit-learn
1112 from pyspark.ml.feature import VectorAssembler
1113
1114 # Converter DataFrame PySpark para um DataFrame Pandas
1115 df_pandas = treino.toPandas()
1116
1117 # Extrair features e labels do DataFrame Pandas
1118 X = df_pandas['features'].tolist()
1119 y = df_pandas['label'].tolist()
1120
1121 # Criar um pipeline com um modelo de SVM e um scaler
1122 pipeline = Pipeline([
1123     ('scaler', StandardScaler()),
1124     ('svm', SVC(kernel='linear', C=1.0))  # Kernel e outros
           parametros podem ser ajustados conforme necessario
1125 ])
1126
1127 # Realizar a validacao cruzada usando Scikit-Learn
1128 cross_val_result = cross_val_score(pipeline, X, y, cv=10)
1129
1130 # Imprimir os resultados da validacao cruzada
1131 print("Resultados da validacao cruzada:", cross_val_result)
1132 print("Acuracia media:", cross_val_result.mean())
1133
1134
1135 # In[101]:
1136
1137
1138 modelo_svm = svm.fit(treino)
1139
1140
1141 # In[102]:
1142
1143
1144 import pandas as pd
1145 import matplotlib.pyplot as plt
1146 import numpy as np
1147
1148 # Exibe os coeficientes atribuidos a cada atributo no modelo
1149 coefficients = modelo_svm.coefficients
1150
```

```
1151 # Monta o dicionario, x e a lista de atributos que foi gerada
        algumas linhas acima
1152 coefficients_dict = dict(zip(x, coefficients))
1153
1154 coefficients_df = pd.DataFrame({'Attribute': x, 'Coefficient':
        coefficients})
1155 print(coefficients_df)
1156
1157 # ordena os atributos por ordem decrescente de importancia
1158 coefficients_df.sort_values(by="Coefficient", ascending=False,
        inplace=True)
1159
1160 # divide as variaveis em positivas e negativas
1161 positive_coefficients =
        coefficients_df[coefficients_df["Coefficient"] > 0].head(10)
1162 negative_coefficients =
        coefficients_df[coefficients_df["Coefficient"] < 0].tail(10)
1163
1164 # combina as variaveis positivas e negativas
1165 display_coefficients = pd.concat([positive_coefficients,
        negative_coefficients])
1166
1167 # plota o grafico de barras horizontais
1168 # Em red sao as variaveis que influenciam no resultado negativo, ou
        seja, influenciam em o aluno nao desistir
1169 # Em red green as variaveis que influenciam no resultado positivo,
        ou seja, influenciam em o aluno desistir
1170 colors = np.array(["green"]*10 + ["red"]*10)
1171 plt.barh(y=display_coefficients["Attribute"],
        width=display_coefficients["Coefficient"], height=0.8,
        color=colors)
1172
1173 # definindo o titulo do grafico e os rotulos dos eixos
1174 plt.title("Importancia dos Atributos")
1175 plt.xlabel("Coeficiente")
1176 plt.ylabel("Atributo")
1177
1178 # exibe o grafico
1179 plt.show()
1180
1181
1182 # In[103]:
```

```
1183
1184
1185 previsoes_svm_teste = modelo_svm.transform(teste)
1186
1187
1188 # In[104]:
1189
1190
1191 previsoes_svm_teste.show()
1192
1193
1194 # In[105]:
1195
1196
1197 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
1198
1199
1200 # In[106]:
1201
1202
1203 evaluator = MulticlassClassificationEvaluator()
1204
1205
1206 # In[107]:
1207
1208
1209 evaluator.evaluate(previsoes_svm_teste, {evaluator.metricName:
       'accuracy'})
1210
1211
1212 # In[108]:
1213
1214
1215 tp = evaluator.evaluate(previsoes_svm_teste, {evaluator.metricName:
       'truePositiveRateByLabel', evaluator.metricLabel: 1.0})
1216 tn = evaluator.evaluate(previsoes_svm_teste, {evaluator.metricName:
       'truePositiveRateByLabel', evaluator.metricLabel: 0.0})
1217 fp = evaluator.evaluate(previsoes_svm_teste, {evaluator.metricName:
       'falsePositiveRateByLabel', evaluator.metricLabel: 1.0})
1218 fn = evaluator.evaluate(previsoes_svm_teste, {evaluator.metricName:
       'falsePositiveRateByLabel', evaluator.metricLabel: 0.0})
1219
```

```
1220
1221 # In[109]:
1222
1223
1224 print("VP: %f"% tp)
1225 print("VN: %f"% tn)
1226 print("FP: %f"% fp)
1227 print("FN: %f"% fn)
1228
1229
1230 # In[110]:
1231
1232
1233 print('\033[1m'+'           Predito\n'+'\033[0m')
1234 print('      TP: ' + str(tp) +' '+ str(fp))
1235 print('\033[1m'+'Real'+'\033[0m')
1236 print('      TN: ' + str(fn) +' '+ str(tn))
1237
1238
1239 # In[111]:
1240
1241
1242 accuracy = (tp+tn)/(tp+tn+fp+fn)
1243 print("Acuracia: %f" % accuracy)
1244
1245 precisao = (tp)/(tp+fp)
1246 print("Precisao: %f" % precisao)
1247
1248 #Metrica que avalia a capacidade de classificar corretamente
1249 #resultados classificados como positivos
1250 sensibilidade = (tp)/(tp+fn)
1251 print("Sensibilidade (Recall): %f" % sensibilidade)
1252
1253 #Metrica que avalia a capacidade de classificar corretamente
1254 #resultados classificados como negativos
1255 especificidade = (tn)/(tn+fp)
1256 print("Especificidade: %f" % especificidade)
1257
1258
1259 # ## Teste de Hipotese T-student
1260
1261 # In[112]:
```

```
1262
1263
1264 from itertools import combinations
1265 from scipy.stats import ttest_rel
1266
1267 algoritmos = ['lr', 'dt', 'rf', 'svm']
1268
1269 for alg1, alg2 in combinations(algoritmos, 2):
1270     observacoes_alg1_acc = []
1271     observacoes_alg2_acc = []
1272
1273     for i in range(10):
1274         # Treinamento e avaliacao do algoritmo 1
1275         modelo_alg1 = eval(alg1).fit(treino)
1276         previsoes_alg1_teste = modelo_alg1.transform(teste)
1277         acc_alg1 = evaluator.evaluate(previsoes_alg1_teste,
                {evaluator.metricName: 'accuracy'})
1278         observacoes_alg1_acc.append(acc_alg1)
1279
1280         # Treinamento e avaliacao do algoritmo 2
1281         modelo_alg2 = eval(alg2).fit(treino)
1282         previsoes_alg2_teste = modelo_alg2.transform(teste)
1283         acc_alg2 = evaluator.evaluate(previsoes_alg2_teste,
                {evaluator.metricName: 'accuracy'})
1284         observacoes_alg2_acc.append(acc_alg2)
1285
1286     # Realiza o teste t de Student
1287     t_statistic, p_value = ttest_rel(observacoes_alg1_acc,
            observacoes_alg2_acc)
1288
1289     # Compara o valor-p com o nivel de significancia alfa para
            aceitar ou rejeitar a hipotese nula
1290     alpha = 0.05
1291     h0 = f"Nao ha diferenca significativa no desempenho entre os
            modelos {alg1.upper()} e {alg2.upper()} (ACURACIA)"
1292     h1 = f"Ha diferenca significativa no desempenho entre os modelos
            {alg1.upper()} e {alg2.upper()} (ACURACIA)"
1293
1294     if p_value <= alpha:
1295         print(f"Rejeitamos a hipotese nula: {h0}, com um valor-p de
                {p_value:.3f}.")
1296         #print(f"Ha evidencia estatistica de que {h1}.")
```

```
1297     else:
1298         print(f"Aceitamos a hipotese nula: {h0}, com um valor-p de
             {p_value:.3f}.")
```

Listing C.3 – Classificador Presencial privada

```
1
2  #!/usr/bin/env python
3  # coding: utf-8
4
5  # # Classificadores de Evasao Estudantil
6
7  # ## Configurando pyspark:
8
9  # In[1]:
10
11
12 #!pip install pyspark
13
14
15 # In[2]:
16
17
18 from pyspark.sql import SparkSession
19
20
21 # In[3]:
22
23
24 #Define onde sera criado a sessao do spark, nesse caso sera local
25 spark =
       SparkSession.builder.master('local[*]').appName("Classificacao
       com Spark").getOrCreate()
26
27
28 # In[4]:
29
30
31 spark
32
33
34 # ## Carregando os dados do csv:
35
```

```
# In[5]:


dados = spark.read.csv('presencial2016ColunaDesistentePrivada.csv',
sep='|',header=True, inferSchema=True)


# In[6]:


dados.count()


# In[7]:


#Quantidade de Desistentes e Nao-Desistentes
dados.groupBy('DESISTENTE').count().show()


# ## Selecionando e tratando atributos:

# In[8]:


dados = dados.drop(*(
# 'CO_IES',
  'NO_IES', #--Campo inutil
# 'CO_CATEGORIA_ADMINISTRATIVA',
  'DS_CATEGORIA_ADMINISTRATIVA', #--Campo inutil
# 'CO_ORGANIZACAO_ACADEMICA',
  'DS_ORGANIZACAO_ACADEMICA', #--Campo inutil
# 'CO_CURSO',
  'NO_CURSO', #--Campo inutil
  'CO_CURSO_POLO', #--Campo inutil (presencial)
# 'CO_TURNO_ALUNO',
  'DS_TURNO_ALUNO', #--Campo inutil
# 'CO_GRAU_ACADEMICO',
  'DS_GRAU_ACADEMICO', #--Campo inutil
# 'CO_MODALIDADE_ENSINO',
  'DS_MODALIDADE_ENSINO', #--Campo inutil
# 'CO_NIVEL_ACADEMICO',
```

```
78    'DS_NIVEL_ACADEMICO', #--Campo inutil
79    'CO_OCDE', #--Nao tratado
80    'NO_OCDE', #--Campo inutil
81  # 'CO_OCDE_AREA_GERAL',
82    'NO_OCDE_AREA_GERAL', #--Campo inutil
83  # 'CO_OCDE_AREA_ESPECIFICA',
84    'NO_OCDE_AREA_ESPECIFICA', #--Campo inutil
85  # 'CO_OCDE_AREA_DETALHADA',
86    'NO_OCDE_AREA_DETALHADA', #--Campo inutil
87  # 'CO_ALUNO_CURSO',
88    'CO_ALUNO_CURSO_ORIGEM', #--Nao tratado
89    'CO_ALUNO', #--Campo inutil
90  # 'CO_COR_RACA_ALUNO',
91    'DS_COR_RACA_ALUNO', #--Campo inutil
92  # 'IN_SEXO_ALUNO',
93    'DS_SEXO_ALUNO', #--Campo inutil
94  # 'NU_ANO_ALUNO_NASC',
95  # 'NU_MES_ALUNO_NASC',
96  # 'NU_DIA_ALUNO_NASC',
97  # 'NU_IDADE_ALUNO',
98  # 'CO_NACIONALIDADE_ALUNO',
99    'DS_NACIONALIDADE_ALUNO', #--Campo inutil
100 # 'CO_PAIS_ORIGEM_ALUNO',
101   'CO_UF_NASCIMENTO', #--Nao tratado
102   'CO_MUNICIPIO_NASCIMENTO', #--Nao tratado
103 # 'IN_ALUNO_DEF_TGD_SUPER',
104 # 'IN_DEF_AUDITIVA',
105 # 'IN_DEF_FISICA',
106 # 'IN_DEF_INTELECTUAL',
107 # 'IN_DEF_MULTIPLA',
108 # 'IN_DEF_SURDEZ',
109 # 'IN_DEF_SURDOCEGUEIRA',
110 # 'IN_DEF_BAIXA_VISAO',
111 # 'IN_DEF_CEGUEIRA',
112 # 'IN_DEF_SUPERDOTACAO',
113 # 'IN_TGD_AUTISMO_INFANTIL',
114 # 'IN_TGD_SINDROME_ASPERGER',
115 # 'IN_TGD_SINDROME_RETT',
116 # 'IN_TGD_TRANSTOR_DESINTEGRATIVO',
117 # 'CO_ALUNO_SITUACAO',
118   'DS_ALUNO_SITUACAO', #--Campo inutil
119 # 'QT_CARGA_HORARIA_TOTAL',
```

```
120 # 'QT_CARGA_HORARIA_INTEG',
121   'DT_INGRESSO_CURSO', #--Nao tratado
122 # 'IN_ING_VESTIBULAR',
123 # 'IN_ING_ENEM',
124 # 'IN_ING_AVALIACAO_SERIADA',
125 # 'IN_ING_SELECAO_SIMPLIFICADA',
126 # 'IN_ING_SELECAO_VAGA_REMANESC',
127 # 'IN_ING_SELECAO_VAGA_PROG_ESPEC',
128 # 'IN_ING_TRANSF_EXOFFICIO',
129 # 'IN_ING_DECISAO_JUDICIAL',
130 # 'IN_ING_CONVENIO_PECG',
131 # 'IN_RESERVA_VAGAS',
132 # 'IN_RESERVA_ETNICO',
133 # 'IN_RESERVA_DEFICIENCIA',
134 # 'IN_RESERVA_ENSINO_PUBLICO',
135 # 'IN_RESERVA_RENDA_FAMILIAR',
136 # 'IN_RESERVA_OUTRA',
137 # 'IN_FINANC_ESTUDANTIL',
138 # 'IN_FIN_REEMB_FIES',
139 # 'IN_FIN_REEMB_ESTADUAL',
140 # 'IN_FIN_REEMB_MUNICIPAL',
141 # 'IN_FIN_REEMB_PROG_IES',
142 # 'IN_FIN_REEMB_ENT_EXTERNA',
143 # 'IN_FIN_REEMB_OUTRA',
144 # 'IN_FIN_NAOREEMB_PROUNI_INTEGR',
145 # 'IN_FIN_NAOREEMB_PROUNI_PARCIAL',
146 # 'IN_FIN_NAOREEMB_ESTADUAL',
147 # 'IN_FIN_NAOREEMB_MUNICIPAL',
148 # 'IN_FIN_NAOREEMB_PROG_IES',
149 # 'IN_FIN_NAOREEMB_ENT_EXTERNA',
150 # 'IN_FIN_NAOREEMB_OUTRA',
151 # 'IN_APOIO_SOCIAL',
152 # 'IN_APOIO_ALIMENTACAO',
153 # 'IN_APOIO_BOLSA_PERMANENCIA',
154 # 'IN_APOIO_BOLSA_TRABALHO',
155 # 'IN_APOIO_MATERIAL_DIDATICO',
156 # 'IN_APOIO_MORADIA',
157 # 'IN_APOIO_TRANSPORTE',
158 # 'IN_ATIVIDADE_EXTRACURRICULAR',
159 # 'IN_COMPL_ESTAGIO',
160 # 'IN_COMPL_EXTENSAO',
161 # 'IN_COMPL_MONITORIA',
```

```
162 # 'IN_COMPL_PESQUISA',
163 # 'IN_BOLSA_ESTAGIO',
164 # 'IN_BOLSA_EXTENSAO',
165 # 'IN_BOLSA_MONITORIA',
166 # 'IN_BOLSA_PESQUISA',
167 # 'CO_TIPO_ESCOLA_ENS_MEDIO',
168 # 'IN_ALUNO_PARFOR',
169  'CO_SEMESTRE_CONCLUSAO',
170 # 'CO_SEMESTRE_REFERENCIA',
171 # 'IN_MOBILIDADE_ACADEMICA',
172 # 'CO_MOBILIDADE_ACADEMICA',
173 # 'CO_MOBILIDADE_ACADEMICA_INTERN',
174   'CO_IES_DESTINO', #--Nao tratado
175   'CO_PAIS_DESTINO', #--Nao tratado
176   'IN_MATRICULA',
177 # 'IN_CONCLUINTE',
178 # 'IN_INGRESSO_TOTAL',
179 # 'IN_INGRESSO_VAGA_NOVA',
180 # 'ANO_INGRESSO',
181 # 'DESISTENTE'
182                     ))
183
184
185 # In[9]:
186
187
188 dados.printSchema()
189
190
191 # In[10]:
192
193
194 #dados.select('IN_DEF_INTELECTUAL').where('IN_DEF_INTELECTUAL is
       null').show()
195 #dados.groupBy('IN_DEF_INTELECTUAL').count().show()
196
197
198 # ## Formatando Atributos para processamento
199
200 # In[11]:
201
202
```

```
203 from pyspark.sql import functions as f
204
205 colunasParaCorrigir0 = [
206 'CO_TURNO_ALUNO'
207 ]
208
209 todasColunas0 = [f.when(f.col(c) == 1, 1).when(f.col(c) == 2, 2).
210                  when(f.col(c) == 3, 3).when(f.col(c) == 4, 4).
211                  otherwise(5).alias(c)
212                 for c in colunasParaCorrigir0]
213
214
215 # In[12]:
216
217
218 for coluna in dados.columns:
219     if coluna not in colunasParaCorrigir0:
220         todasColunas0.insert(0, coluna)
221
222
223 #todasColunas0
224
225 dados = dados.select(todasColunas0)
226 #dataset.groupBy('IN_DEF_INTELECTUAL').count().show()
227
228
229 # In[13]:
230
231
232 from pyspark.sql import functions as f
233
234 colunasParaCorrigir1 = [
235  #'CO_SEMESTRE_CONCLUSAO',
236  'CO_SEMESTRE_REFERENCIA',
237  'CO_MOBILIDADE_ACADEMICA',
238  'CO_MOBILIDADE_ACADEMICA_INTERN'
239 ]
240
241 todasColunas1 = [f.when(f.col(c) == 1, 1).when(f.col(c) == 2 ,
    2).otherwise(3).alias(c)
242                 for c in colunasParaCorrigir1]
243
```

```
244
245 # In[14]:
246
247
248 for coluna in dados.columns:
249     if coluna not in colunasParaCorrigir1:
250         todasColunas1.insert(0, coluna)
251
252
253 #todasColunas1
254
255 dados = dados.select(todasColunas1)
256 #dataset.groupBy('IN_DEF_INTELECTUAL').count().show()
257
258
259 # In[15]:
260
261
262 from pyspark.sql import functions as f
263
264 colunasParaCorrigir = [
265  'IN_ALUNO_DEF_TGD_SUPER',
266  'IN_DEF_AUDITIVA',
267  'IN_DEF_FISICA',
268  'IN_DEF_INTELECTUAL',
269  'IN_DEF_MULTIPLA',
270  'IN_DEF_SURDEZ',
271  'IN_DEF_SURDOCEGUEIRA',
272  'IN_DEF_BAIXA_VISAO',
273  'IN_DEF_CEGUEIRA',
274  'IN_DEF_SUPERDOTACAO',
275  'IN_TGD_AUTISMO_INFANTIL',
276  'IN_TGD_SINDROME_ASPERGER',
277  'IN_TGD_SINDROME_RETT',
278  'IN_TGD_TRANSTOR_DESINTEGRATIVO',
279
280  'IN_ING_VESTIBULAR',
281  'IN_ING_ENEM',
282  'IN_ING_AVALIACAO_SERIADA',
283  'IN_ING_SELECAO_SIMPLIFICADA',
284  'IN_ING_SELECAO_VAGA_REMANESC',
285  'IN_ING_SELECAO_VAGA_PROG_ESPEC',
```

```
286  'IN_ING_TRANSF_EXOFFICIO',
287  'IN_ING_DECISAO_JUDICIAL',
288  'IN_ING_CONVENIO_PECG',
289  'IN_RESERVA_VAGAS',
290  'IN_RESERVA_ETNICO',
291  'IN_RESERVA_DEFICIENCIA',
292  'IN_RESERVA_ENSINO_PUBLICO',
293  'IN_RESERVA_RENDA_FAMILIAR',
294  'IN_RESERVA_OUTRA',
295  'IN_FINANC_ESTUDANTIL',
296  'IN_FIN_REEMB_FIES',
297  'IN_FIN_REEMB_ESTADUAL',
298  'IN_FIN_REEMB_MUNICIPAL',
299  'IN_FIN_REEMB_PROG_IES',
300  'IN_FIN_REEMB_ENT_EXTERNA',
301  'IN_FIN_NAOREEMB_PROUNI_INTEGR',
302  'IN_FIN_NAOREEMB_PROUNI_PARCIAL',
303  'IN_FIN_NAOREEMB_ESTADUAL',
304  'IN_FIN_NAOREEMB_MUNICIPAL',
305  'IN_FIN_NAOREEMB_PROG_IES',
306  'IN_FIN_NAOREEMB_ENT_EXTERNA',
307  'IN_APOIO_SOCIAL',
308  'IN_APOIO_ALIMENTACAO',
309  'IN_APOIO_BOLSA_PERMANENCIA',
310  'IN_APOIO_BOLSA_TRABALHO',
311  'IN_APOIO_MATERIAL_DIDATICO',
312  'IN_APOIO_MORADIA',
313  'IN_APOIO_TRANSPORTE',
314  'IN_ATIVIDADE_EXTRACURRICULAR',
315  'IN_COMPL_ESTAGIO',
316  'IN_COMPL_EXTENSAO',
317  'IN_COMPL_MONITORIA',
318  'IN_COMPL_PESQUISA',
319  'IN_BOLSA_ESTAGIO',
320  'IN_BOLSA_EXTENSAO',
321  'IN_BOLSA_MONITORIA',
322  'IN_BOLSA_PESQUISA',
323  'CO_TIPO_ESCOLA_ENS_MEDIO',
324  'IN_ALUNO_PARFOR',
325
326  'IN_MOBILIDADE_ACADEMICA',
327
```

```
 #'IN_MATRICULA',
 'IN_CONCLUINTE',
 'IN_INGRESSO_TOTAL',
 'IN_INGRESSO_VAGA_NOVA',

 'IN_FIN_REEMB_OUTRA',
 'IN_FIN_NAOREEMB_OUTRA'

]

todasColunas = [f.when(f.col(c) == 0, 0).when(f.col(c) == 1 , 1).otherwise(2).alias(c)
                for c in colunasParaCorrigir]


# In[16]:


for coluna in dados.columns:
    if coluna not in colunasParaCorrigir:
        todasColunas.insert(0, coluna)


#todasColunas
dataset = dados.select(todasColunas)
#dataset.groupBy('IN_DEF_INTELECTUAL').count().show()


# ## Balanceamento dos dados:

# In[17]:


dados_not_desistente = dataset.select(todasColunas).where('DESISTENTE = 0')
print('dados nao desistente: '+str(dados_not_desistente.count()))

dados_not_desistente = dataset.select(todasColunas).where('DESISTENTE = 0').sample(withReplacement=True, fraction=0.1298, seed=101)
print('dados nao desistente fracionado: '+str(dados_not_desistente.count()))
```

```
365
366 dados_desistente = dataset.select(todasColunas).where('DESISTENTE =
       1')
367 print('dados desistente: '+str(dados_desistente.count()))
368
369
370 # In[18]:
371
372
373 ######################### UNIR AS BASES APOS BALANCEAMENTO
374 dados = dados_not_desistente.unionAll(dados_desistente)
375
376 dados.count()
377 ########################
378
379
380 # ## Informacoes de correlacao dos atributos:
381
382 # In[ ]:
383
384
385 #So executar daqui para baixo
386 import plotly.express as px
387 dados_plot = dados
388 fig = px.imshow(dados.toPandas().corr(), text_auto=True)
389 fig.show()
390
391
392 # In[ ]:
393
394
395 import pandas as pd
396 pd.set_option('display.max_columns',  None)
397 pd.set_option('display.max_rows',  None)
398
399 dados_plot.toPandas().corr()
400
401
402 # In[ ]:
403
404
405 import pandas as pd
```

```
406 import plotly.express as px
407
408 # Configuracao para exibir todas as colunas e linhas
409 pd.set_option('display.max_columns', None)
410 pd.set_option('display.max_rows', None)
411
412 # Calcula a matriz de correlacao
413 correlation_matrix = dados_plot.toPandas().corr()
414
415 # Obtem os 10 pares de atributos com a maior correlacao, excluindo a
    correlacao consigo mesmo
416 top_correlations = (correlation_matrix
417                     .unstack()
418                     .sort_values(ascending=False)
419                     .drop_duplicates()
420                     .head(16)  # Adiciona 1 para incluir a primeira
                           entrada (correlacao consigo mesmo)
421                     .tail(15))  # Exclui a correlacao consigo mesmo
422
423 # Exibe os pares de atributos
424 print("Top 10 pares de atributos com maior correlacao (excluindo a
    correlacao consigo mesmo):")
425 print(top_correlations)
426
427 ############################
428 # Filtra a matriz de correlacao apenas para os 10 pares de atributos
429 filtered_correlation_matrix =
    correlation_matrix.loc[top_correlations.index.get_level_values(0),
    top_correlations.index.get_level_values(1)]
430
431 # Cria o grafico de heatmap usando Plotly para os 10 pares de
    atributos
432 fig = px.imshow(filtered_correlation_matrix, text_auto=True)
433 fig.show()
434
435
436 # ## Definindo qual atributo e o label
437
438 # In[24]:
439
440
441 dados = dados.withColumnRenamed('DESISTENTE','label')
```

```
442
443
444 # In[25]:
445
446
447 x = dados.columns
448 x.remove('label')
449 len(x)
450
451
452 # ## Preparando vetor de entrada para a construcao dos modelos:
453
454 # In[26]:
455
456
457 from pyspark.ml.feature import VectorAssembler
458
459
460 # In[27]:
461
462
463 assembler = VectorAssembler(inputCols=x, outputCol='features')
464
465
466 # In[28]:
467
468
469 dados_preparados = assembler.setHandleInvalid("skip").
470 transform(dados).select('features','label')
471
472
473 # In[29]:
474
475
476 dados_preparados.show()
477
478
479 # ## Dividindo o dataset em Treino e Teste:
480
481 # In[30]:
482
483
```

```
SEED = 101


# In[31]:


treino, teste = dados_preparados.randomSplit([0.7, 0.3], seed=SEED)


# In[32]:


treino.show()


# In[33]:


treino.count()


# In[34]:


teste.count()


# # LogisticRegression

# In[35]:


from pyspark.ml.classification import LogisticRegression


# In[36]:


lr = LogisticRegression()


# In[ ]:
```

```
526
527
528 from pyspark.ml import Pipeline
529
530 # Criar um pipeline, para que o CrossValidator utilize as mesmas
      etapas de pre-processamento com os mesmos dados utilizados
531 # para a geracao no modelo no pyspark
532 pipeline = Pipeline(stages=[assembler, lr])
533
534
535 # In[34]:
536
537
538 #Esse CrossValidator nao e para ajuste de hiperparametros, pois o
      pyspark ja faz isso,
539 # esse CrossValidator para captura de uma media de Acuracia dada por
      ele para saber se
540 # o modelo gerado no pyspark se sobressai sobre a media. Nos
      particionamos nossa base em
541 # 70% de treino e 30% teste de maneira que o modelo sera validado
      com esses dados de teste.
542 from sklearn.model_selection import cross_val_score
543 from sklearn.pipeline import Pipeline
544 from sklearn.preprocessing import StandardScaler
545 from sklearn.linear_model import LogisticRegression
546 from pyspark.ml.feature import VectorAssembler
547
548 # Converter DataFrame PySpark para um DataFrame Pandas
549 df_pandas = treino.toPandas()
550
551 # Extrair features e labels do DataFrame Pandas
552 X = df_pandas['features'].tolist()
553 y = df_pandas['label'].tolist()
554
555 # Criar um pipeline com um modelo de regressao logistica e um scaler
556 pipeline = Pipeline([
557     ('scaler', StandardScaler()),
558     ('lr', LogisticRegression(max_iter=1000))
559 ])
560
561 # Realizar a validacao cruzada usando Scikit-Learn
562 cross_val_result = cross_val_score(pipeline, X, y, cv=10)
```

```
563
564 # Imprimir os resultados da validacao cruzada
565 print("Resultados da validacao cruzada:", cross_val_result)
566 print("Acuracia media:", cross_val_result.mean())
567
568
569 # In[37]:
570
571
572 modelo_lr = lr.fit(treino)
573
574
575 # In[38]:
576
577
578 import pandas as pd
579 import matplotlib.pyplot as plt
580 import numpy as np
581
582 # Exibe os coeficientes atribuidos a cada atributo no modelo
583 coefficients = modelo_lr.coefficients
584
585 # Monta o dicionario, x e a lista de atributos que foi gerada
    algumas linhas acima
586 coefficients_dict = dict(zip(x, coefficients))
587
588 coefficients_df = pd.DataFrame({'Attribute': x, 'Coefficient':
    coefficients})
589 print(coefficients_df)
590
591 # ordena os atributos por ordem decrescente de importancia
592 coefficients_df.sort_values(by="Coefficient", ascending=False,
    inplace=True)
593
594 # divide as variaveis em positivas e negativas
595 positive_coefficients =
    coefficients_df[coefficients_df["Coefficient"] > 0].head(10)
596 negative_coefficients =
    coefficients_df[coefficients_df["Coefficient"] < 0].tail(10)
597
598 # combina as variaveis positivas e negativas
599 display_coefficients = pd.concat([positive_coefficients,
```

```
    negative_coefficients])
600
601 # plota o grafico de barras horizontais
602 # Em red sao as variaveis que influenciam no resultado negativo, ou
    seja, influenciam em o aluno nao desistir
603 # Em red green as variaveis que influenciam no resultado positivo,
    ou seja, influenciam em o aluno desistir
604 colors = np.array(["green"]*10 + ["red"]*10)
605 plt.barh(y=display_coefficients["Attribute"],
    width=display_coefficients["Coefficient"], height=0.8,
    color=colors)
606
607 # definindo o titulo do grafico e os rotulos dos eixos
608 plt.title("Importancia dos Atributos")
609 plt.xlabel("Coeficiente")
610 plt.ylabel("Atributo")
611
612 # exibe o grafico
613 plt.show()
614
615
616 # In[39]:
617
618
619 previsoes_lr_teste = modelo_lr.transform(teste)
620
621
622 # In[40]:
623
624
625 previsoes_lr_teste.show()
626
627
628 # In[41]:
629
630
631 # A funcao summary fornece estatisticas e informacoes sobre
632 # o modelo treinado com base nos dados de treinamento
633 # a previsao com os dados de teste e feita linhas abaixo
634 resumo_lr_treino = modelo_lr.summary
635
636
```

```
637 # In[42]:
638
639
640 print("Acuracia: %f" % resumo_lr_treino.accuracy)
641 print("Precisao: %f" % resumo_lr_treino.precisionByLabel[1])
642 print("Recall (Sensibilidade): %f"%
       resumo_lr_treino.recallByLabel[1])
643 print("F1: %f"% resumo_lr_treino.fMeasureByLabel()[1])
644
645
646 # ## Trazendo as metricas dos dados de teste usados de entrada para
       o modelo:
647
648 # In[43]:
649
650
651 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
652
653
654 # In[44]:
655
656
657 evaluator = MulticlassClassificationEvaluator()
658
659
660 # In[45]:
661
662
663 evaluator.evaluate(previsoes_lr_teste, {evaluator.metricName:
       'accuracy'})
664
665
666 # In[46]:
667
668
669 tp = evaluator.evaluate(previsoes_lr_teste, {evaluator.metricName:
       'truePositiveRateByLabel', evaluator.metricLabel: 1.0})
670 tn = evaluator.evaluate(previsoes_lr_teste, {evaluator.metricName:
       'truePositiveRateByLabel', evaluator.metricLabel: 0.0})
671 fp = evaluator.evaluate(previsoes_lr_teste, {evaluator.metricName:
       'falsePositiveRateByLabel', evaluator.metricLabel: 1.0})
672 fn = evaluator.evaluate(previsoes_lr_teste, {evaluator.metricName:
```

```
    'falsePositiveRateByLabel', evaluator.metricLabel: 0.0})


# In[47]:


print("VP: %f"% tp)
print("VN: %f"% tn)
print("FP: %f"% fp)
print("FN: %f"% fn)


# In[48]:


print('\033[1m'+'                          Predito\n'+'\033[0m')
print('      TP: ' + str(tp) +' '+ str(fp))
print('\033[1m'+'Real'+'\033[0m')
print('      TN: ' + str(fn) +' '+ str(tn))


# In[49]:


accuracy = (tp+tn)/(tp+tn+fp+fn)
print("Acuracia: %f" % accuracy)

precisao = (tp)/(tp+fp)
print("Precisao: %f" % precisao)

#Metrica que avalia a capacidade de classificar corretamente
#resultados classificados como positivos
sensibilidade = (tp)/(tp+fn)
print("Sensibilidade (Recall): %f" % sensibilidade)

#Metrica que avalia a capacidade de classificar corretamente
#resultados classificados como negativos
especificidade = (tn)/(tn+fp)
print("Especificidade: %f" % especificidade)


# # Decision Tree
```

```
714
715 # In[50]:
716
717
718 from pyspark.ml.classification import DecisionTreeClassifier
719
720
721 # In[51]:
722
723
724 dt = DecisionTreeClassifier(seed=SEED)
725
726
727 # In[ ]:
728
729
730 from pyspark.ml import Pipeline
731
732 # Criar um pipeline, para que o CrossValidator utilize as mesmas
       etapas de pre-processamento com os mesmos dados utilizados
733 # para a geracao no modelo no pyspark
734 pipeline = Pipeline(stages=[assembler, dt])
735
736
737 # In[ ]:
738
739
740 #Esse CrossValidator nao e para ajuste de hiperparametros, pois o
       pyspark ja faz isso,
741 # esse CrossValidator para captura de uma media de Acuracia dada por
       ele para saber se
742 # o modelo gerado no pyspark se sobressai sobre a media. Nos
       particionamos nossa base em
743 # 70% de treino e 30% teste de maneira que o modelo sera validado
       com esses dados de teste.
744 from sklearn.model_selection import cross_val_score
745 from sklearn.pipeline import Pipeline
746 from sklearn.preprocessing import StandardScaler
747 from sklearn.tree import DecisionTreeClassifier
748 from pyspark.ml.feature import VectorAssembler
749
750 # Converter DataFrame PySpark para um DataFrame Pandas
```

```
751 df_pandas = treino.toPandas()
752
753 # Extrair features e labels do DataFrame Pandas
754 X = df_pandas['features'].tolist()
755 y = df_pandas['label'].tolist()
756
757 # Criar um pipeline com uma arvore de decisao e um scaler
758 pipeline = Pipeline([
759     ('scaler', StandardScaler()),
760     ('dt', DecisionTreeClassifier())
761 ])
762
763 # Realizar a validacao cruzada usando Scikit-Learn
764 cross_val_result = cross_val_score(pipeline, X, y, cv=10)
765
766 # Imprimir os resultados da validacao cruzada
767 print("Resultados da validacao cruzada:", cross_val_result)
768 print("Acuracia media:", cross_val_result.mean())
769
770
771 # In[52]:
772
773
774 modelo_dt = dt.fit(treino)
775
776
777 # In[53]:
778
779
780 import pandas as pd
781 import matplotlib.pyplot as plt
782 import numpy as np
783
784 # Exibe os coeficientes atribuidos a cada atributo no modelo
785 importances = modelo_dt.featureImportances
786
787 # Monta o dicionario, x e a lista de atributos que foi gerada
       algumas linhas acima
788 importances_dict = dict(zip(x, importances))
789
790 importances_df = pd.DataFrame({'Attribute': x, 'Importance':
       importances})
```

```
791 print(importances_df)
792
793 # ordena os atributos por ordem decrescente de importancia
794 importances_df.sort_values(by="Importance", ascending=False,
       inplace=True)
795
796 # divide as variaveis em positivas e negativas
797 positive_coefficients = importances_df.head(10)
798 negative_coefficients = importances_df.tail(10)
799
800 # combina as variaveis positivas e negativas
801 display_coefficients = pd.concat([positive_coefficients,
       negative_coefficients])
802
803 # plota o grafico de barras horizontais
804 colors = ["green"] * 10 + ["red"] * 10
805 plt.barh(y=display_coefficients["Attribute"],
       width=display_coefficients["Importance"], height=0.8,
       color=colors)
806
807 # definindo o titulo do grafico e os rotulos dos eixos
808 plt.title("Importancia dos Atributos")
809 plt.xlabel("Importancia")
810 plt.ylabel("Atributo")
811
812 # exibe o grafico
813 plt.show()
814
815
816 # In[54]:
817
818
819 previsoes_dt_teste = modelo_dt.transform(teste)
820
821
822 # In[55]:
823
824
825 previsoes_dt_teste.show()
826
827
828 # # Trazendo as metricas dos dados de teste usados de entrada para o
```

```
    modelo:
829
830 # In[56]:
831
832
833 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
834
835
836 # In[57]:
837
838
839 evaluator = MulticlassClassificationEvaluator()
840
841
842 # In[58]:
843
844
845 evaluator.evaluate(previsoes_dt_teste, {evaluator.metricName:
    'accuracy'})
846
847
848 # In[59]:
849
850
851 tp = evaluator.evaluate(previsoes_dt_teste, {evaluator.metricName:
    'truePositiveRateByLabel', evaluator.metricLabel: 1.0})
852 tn = evaluator.evaluate(previsoes_dt_teste, {evaluator.metricName:
    'truePositiveRateByLabel', evaluator.metricLabel: 0.0})
853 fp = evaluator.evaluate(previsoes_dt_teste, {evaluator.metricName:
    'falsePositiveRateByLabel', evaluator.metricLabel: 1.0})
854 fn = evaluator.evaluate(previsoes_dt_teste, {evaluator.metricName:
    'falsePositiveRateByLabel', evaluator.metricLabel: 0.0})
855
856
857 # In[60]:
858
859
860 print("VP: %f"% tp)
861 print("VN: %f"% tn)
862 print("FP: %f"% fp)
863 print("FN: %f"% fn)
864
```

```
865
866 # In[61]:
867
868
869 print('\033[1m'+'           Predito\n'+'\033[0m')
870 print('      TP: ' + str(tp) +' '+ str(fp))
871 print('\033[1m'+'Real'+'\033[0m')
872 print('      TN: ' + str(fn) +' '+ str(tn))
873
874
875 # In[62]:
876
877
878 accuracy = (tp+tn)/(tp+tn+fp+fn)
879 print("Acuracia: %f" % accuracy)
880
881 precisao = (tp)/(tp+fp)
882 print("Precisao: %f" % precisao)
883
884 #Metrica que avalia a capacidade de classificar corretamente
885 #resultados classificados como positivos
886 sensibilidade = (tp)/(tp+fn)
887 print("Sensibilidade (Recall): %f" % sensibilidade)
888
889 #Metrica que avalia a capacidade de classificar corretamente
890 #resultados classificados como negativos
891 especificidade = (tn)/(tn+fp)
892 print("Especificidade: %f" % especificidade)
893
894
895 # # Random Forest
896
897 # In[63]:
898
899
900 from pyspark.ml.classification import RandomForestClassifier
901
902
903 # In[64]:
904
905
906 rf = RandomForestClassifier(seed=SEED)
```

```
907
908
909 # In[ ]:
910
911
912 from pyspark.ml import Pipeline
913
914 # Criar um pipeline, para que o CrossValidator utilize as mesmas
       etapas de pre-processamento com os mesmos dados utilizados
915 # para a geracao no modelo no pyspark
916
917 pipeline = Pipeline(stages=[assembler, rf])
918
919
920 # In[ ]:
921
922
923 #Esse CrossValidator nao e para ajuste de hiperparametros, pois o
       pyspark ja faz isso,
924 # esse CrossValidator para captura de uma media de Acuracia dada por
       ele para saber se
925 # o modelo gerado no pyspark se sobressai sobre a media. Nos
       particionamos nossa base em
926 # 70% de treino e 30% teste de maneira que o modelo sera validado
       com esses dados de teste.
927 from sklearn.model_selection import cross_val_score
928 from sklearn.pipeline import Pipeline
929 from sklearn.preprocessing import StandardScaler
930 from sklearn.ensemble import RandomForestClassifier  # Importar o
       RandomForestClassifier do scikit-learn
931 from pyspark.ml.feature import VectorAssembler
932
933 # Converter DataFrame PySpark para um DataFrame Pandas
934 df_pandas = treino.toPandas()
935
936 # Extrair features e labels do DataFrame Pandas
937 X = df_pandas['features'].tolist()
938 y = df_pandas['label'].tolist()
939
940 # Criar um pipeline com um modelo de Random Forest e um scaler
941 pipeline = Pipeline([
942     ('scaler', StandardScaler()),
```

```
943     ('rf', RandomForestClassifier(n_estimators=100,
            random_state=42))  # Numero de arvores pode ser ajustado
            conforme necessario
944 ])
945
946 # Realizar a validacao cruzada usando Scikit-Learn
947 cross_val_result = cross_val_score(pipeline, X, y, cv=10)
948
949 # Imprimir os resultados da validacao cruzada
950 print("Resultados da validacao cruzada:", cross_val_result)
951 print("Acuracia media:", cross_val_result.mean())
952
953
954 # In[65]:
955
956
957 modelo_rf = rf.fit(treino)
958
959
960 # In[66]:
961
962
963 import pandas as pd
964 import matplotlib.pyplot as plt
965 import numpy as np
966
967 # Exibe os coeficientes atribuidos a cada atributo no modelo
968 importances = modelo_rf.featureImportances
969
970 # Monta o dicionario, x e a lista de atributos que foi gerada
       algumas linhas acima
971 importances_dict = dict(zip(x, importances))
972
973 importances_rf = pd.DataFrame({'Attribute': x, 'Importance':
       importances})
974 print(importances_rf)
975
976 # ordena os atributos por ordem decrescente de importancia
977 importances_rf.sort_values(by="Importance", ascending=False,
       inplace=True)
978
979 # divide as variaveis em positivas e negativas
```

```
980 positive_coefficients = importances_rf.head(10)
981 negative_coefficients = importances_rf.tail(10)
982
983 # combina as variaveis positivas e negativas
984 display_coefficients = pd.concat([positive_coefficients,
       negative_coefficients])
985
986 # plota o grafico de barras horizontais
987 colors = ["green"] * 10 + ["red"] * 10
988 plt.barh(y=display_coefficients["Attribute"],
       width=display_coefficients["Importance"], height=0.8,
       color=colors)
989
990 # definindo o titulo do grafico e os rotulos dos eixos
991 plt.title("Importancia dos Atributos")
992 plt.xlabel("Importancia")
993 plt.ylabel("Atributo")
994
995 # exibe o grafico
996 plt.show()
997
998
999 # In[67]:
1000
1001
1002 previsoes_rf_teste = modelo_rf.transform(teste)
1003
1004
1005 # In[68]:
1006
1007
1008 previsoes_rf_teste.show()
1009
1010
1011 # In[69]:
1012
1013
1014 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
1015
1016
1017 # In[70]:
1018
```

```
1019
1020 evaluator = MulticlassClassificationEvaluator()
1021
1022
1023 # In[71]:
1024
1025
1026 evaluator.evaluate(previsoes_rf_teste, {evaluator.metricName:
        'accuracy'})
1027
1028
1029 # In[72]:
1030
1031
1032 tp = evaluator.evaluate(previsoes_rf_teste, {evaluator.metricName:
        'truePositiveRateByLabel', evaluator.metricLabel: 1.0})
1033 tn = evaluator.evaluate(previsoes_rf_teste, {evaluator.metricName:
        'truePositiveRateByLabel', evaluator.metricLabel: 0.0})
1034 fp = evaluator.evaluate(previsoes_rf_teste, {evaluator.metricName:
        'falsePositiveRateByLabel', evaluator.metricLabel: 1.0})
1035 fn = evaluator.evaluate(previsoes_rf_teste, {evaluator.metricName:
        'falsePositiveRateByLabel', evaluator.metricLabel: 0.0})
1036
1037
1038 # In[73]:
1039
1040
1041 print("VP: %f"% tp)
1042 print("VN: %f"% tn)
1043 print("FP: %f"% fp)
1044 print("FN: %f"% fn)
1045
1046
1047 # In[74]:
1048
1049
1050 print('\033[1m'+'           Predito\n'+'\033[0m')
1051 print('      TP: ' + str(tp) +' '+ str(fp))
1052 print('\033[1m'+'Real'+'\033[0m')
1053 print('      TN: ' + str(fn) +' '+ str(tn))
1054
1055
```

```
# In[75]:


accuracy = (tp+tn)/(tp+tn+fp+fn)
print("Acuracia: %f" % accuracy)

precisao = (tp)/(tp+fp)
print("Precisao: %f" % precisao)

#Metrica que avalia a capacidade de classificar corretamente
#resultados classificados como positivos
sensibilidade = (tp)/(tp+fn)
print("Sensibilidade (Recall): %f" % sensibilidade)

#Metrica que avalia a capacidade de classificar corretamente
#resultados classificados como negativos
especificidade = (tn)/(tn+fp)
print("Especificidade: %f" % especificidade)


# # SVM

# In[76]:


from pyspark.ml.classification import LinearSVC


# In[77]:


svm = LinearSVC(maxIter=10, regParam=0.1)


# In[ ]:


from pyspark.ml import Pipeline

# Criar um pipeline, para que o CrossValidator utilize as mesmas
   etapas de pre-processamento com os mesmos dados utilizados
# para a geracao no modelo no pyspark
```

```
1097
1098 pipeline = Pipeline(stages=[assembler, svm])
1099
1100
1101 # In[ ]:
1102
1103
1104 #Esse CrossValidator nao e para ajuste de hiperparametros, pois o
       pyspark ja faz isso,
1105 # esse CrossValidator para captura de uma media de Acuracia dada por
       ele para saber se
1106 # o modelo gerado no pyspark se sobressai sobre a media. Nos
       particionamos nossa base em
1107 # 70% de treino e 30% teste de maneira que o modelo sera validado
       com esses dados de teste.
1108 from sklearn.model_selection import cross_val_score
1109 from sklearn.pipeline import Pipeline
1110 from sklearn.preprocessing import StandardScaler
1111 from sklearn.svm import SVC  # Importar o Support Vector Classifier
       (SVC) do scikit-learn
1112 from pyspark.ml.feature import VectorAssembler
1113
1114 # Converter DataFrame PySpark para um DataFrame Pandas
1115 df_pandas = treino.toPandas()
1116
1117 # Extrair features e labels do DataFrame Pandas
1118 X = df_pandas['features'].tolist()
1119 y = df_pandas['label'].tolist()
1120
1121 # Criar um pipeline com um modelo de SVM e um scaler
1122 pipeline = Pipeline([
1123     ('scaler', StandardScaler()),
1124     ('svm', SVC(kernel='linear', C=1.0))  # Kernel e outros
          parametros podem ser ajustados conforme necessario
1125 ])
1126
1127 # Realizar a validacao cruzada usando Scikit-Learn
1128 cross_val_result = cross_val_score(pipeline, X, y, cv=10)
1129
1130 # Imprimir os resultados da validacao cruzada
1131 print("Resultados da validacao cruzada:", cross_val_result)
1132 print("Acuracia media:", cross_val_result.mean())
```

```
1133
1134
1135 # In[78]:
1136
1137
1138 modelo_svm = svm.fit(treino)
1139
1140
1141 # In[79]:
1142
1143
1144 import pandas as pd
1145 import matplotlib.pyplot as plt
1146 import numpy as np
1147
1148 # Exibe os coeficientes atribuidos a cada atributo no modelo
1149 coefficients = modelo_svm.coefficients
1150
1151 # Monta o dicionario, x e a lista de atributos que foi gerada
       algumas linhas acima
1152 coefficients_dict = dict(zip(x, coefficients))
1153
1154 coefficients_df = pd.DataFrame({'Attribute': x, 'Coefficient':
       coefficients})
1155 print(coefficients_df)
1156
1157 # ordena os atributos por ordem decrescente de importancia
1158 coefficients_df.sort_values(by="Coefficient", ascending=False,
       inplace=True)
1159
1160 # divide as variaveis em positivas e negativas
1161 positive_coefficients =
       coefficients_df[coefficients_df["Coefficient"] > 0].head(10)
1162 negative_coefficients =
       coefficients_df[coefficients_df["Coefficient"] < 0].tail(10)
1163
1164 # combina as variaveis positivas e negativas
1165 display_coefficients = pd.concat([positive_coefficients,
       negative_coefficients])
1166
1167 # plota o grafico de barras horizontais
1168 # Em red sao as variaveis que influenciam no resultado negativo, ou
```

```
    seja, influenciam em o aluno nao desistir
# Em red green as variaveis que influenciam no resultado positivo,
    ou seja, influenciam em o aluno desistir
colors = np.array(["green"]*10 + ["red"]*10)
plt.barh(y=display_coefficients["Attribute"],
    width=display_coefficients["Coefficient"], height=0.8,
    color=colors)

# definindo o titulo do grafico e os rotulos dos eixos
plt.title("Importancia dos Atributos")
plt.xlabel("Coeficiente")
plt.ylabel("Atributo")

# exibe o grafico
plt.show()


# In[80]:


previsoes_svm_teste = modelo_svm.transform(teste)


# In[81]:


previsoes_svm_teste.show()


# In[82]:


from pyspark.ml.evaluation import MulticlassClassificationEvaluator


# In[83]:


evaluator = MulticlassClassificationEvaluator()


# In[84]:
```

```
1207
1208
1209 evaluator.evaluate(previsoes_svm_teste, {evaluator.metricName:
        'accuracy'})
1210
1211
1212 # In[85]:
1213
1214
1215 tp = evaluator.evaluate(previsoes_svm_teste, {evaluator.metricName:
        'truePositiveRateByLabel', evaluator.metricLabel: 1.0})
1216 tn = evaluator.evaluate(previsoes_svm_teste, {evaluator.metricName:
        'truePositiveRateByLabel', evaluator.metricLabel: 0.0})
1217 fp = evaluator.evaluate(previsoes_svm_teste, {evaluator.metricName:
        'falsePositiveRateByLabel', evaluator.metricLabel: 1.0})
1218 fn = evaluator.evaluate(previsoes_svm_teste, {evaluator.metricName:
        'falsePositiveRateByLabel', evaluator.metricLabel: 0.0})
1219
1220
1221 # In[86]:
1222
1223
1224 print("VP: %f"% tp)
1225 print("VN: %f"% tn)
1226 print("FP: %f"% fp)
1227 print("FN: %f"% fn)
1228
1229
1230 # In[87]:
1231
1232
1233 print('\033[1m'+'           Predito\n'+'\033[0m')
1234 print('      TP: ' + str(tp) +' '+ str(fp))
1235 print('\033[1m'+'Real'+'\033[0m')
1236 print('      TN: ' + str(fn) +' '+ str(tn))
1237
1238
1239 # In[88]:
1240
1241
1242 accuracy = (tp+tn)/(tp+tn+fp+fn)
1243 print("Acuracia: %f" % accuracy)
```

```
1244
1245 precisao = (tp)/(tp+fp)
1246 print("Precisao: %f" % precisao)
1247
1248 #Metrica que avalia a capacidade de classificar corretamente
1249 #resultados classificados como positivos
1250 sensibilidade = (tp)/(tp+fn)
1251 print("Sensibilidade (Recall): %f" % sensibilidade)
1252
1253 #Metrica que avalia a capacidade de classificar corretamente
1254 #resultados classificados como negativos
1255 especificidade = (tn)/(tn+fp)
1256 print("Especificidade: %f" % especificidade)
1257
1258
1259 # ## Teste de Hipotese T-student
1260
1261 # In[ ]:
1262
1263
1264 from itertools import combinations
1265 from scipy.stats import ttest_rel
1266
1267 algoritmos = ['lr', 'dt', 'rf', 'svm']
1268
1269 for alg1, alg2 in combinations(algoritmos, 2):
1270     observacoes_alg1_acc = []
1271     observacoes_alg2_acc = []
1272
1273     for i in range(10):
1274         # Treinamento e avaliacao do algoritmo 1
1275         modelo_alg1 = eval(alg1).fit(treino)
1276         previsoes_alg1_teste = modelo_alg1.transform(teste)
1277         acc_alg1 = evaluator.evaluate(previsoes_alg1_teste,
                {evaluator.metricName: 'accuracy'})
1278         observacoes_alg1_acc.append(acc_alg1)
1279
1280         # Treinamento e avaliacao do algoritmo 2
1281         modelo_alg2 = eval(alg2).fit(treino)
1282         previsoes_alg2_teste = modelo_alg2.transform(teste)
1283         acc_alg2 = evaluator.evaluate(previsoes_alg2_teste,
                {evaluator.metricName: 'accuracy'})
```

```
1284         observacoes_alg2_acc.append(acc_alg2)
1285
1286     # Realiza o teste t de Student
1287     t_statistic, p_value = ttest_rel(observacoes_alg1_acc,
            observacoes_alg2_acc)
1288
1289     # Compara o valor-p com o nivel de significancia alfa para
            aceitar ou rejeitar a hipotese nula
1290     alpha = 0.05
1291     h0 = f"Nao ha diferenca significativa no desempenho entre os
            modelos {alg1.upper()} e {alg2.upper()} (ACURACIA)"
1292     h1 = f"Ha diferenca significativa no desempenho entre os modelos
            {alg1.upper()} e {alg2.upper()} (ACURACIA)"
1293
1294     if p_value <= alpha:
1295         print(f"Rejeitamos a hipotese nula: {h0}, com um valor-p de
                {p_value:.3f}.")
1296         #print(f"Ha evidencia estatistica de que {h1}.")
1297     else:
1298         print(f"Aceitamos a hipotese nula: {h0}, com um valor-p de
                {p_value:.3f}.")
```

Listing C.4 – Classificador Presencial publica

```
1
2  #!/usr/bin/env python
3  # coding: utf-8
4
5  # # Classificadores de Evasao Estudantil
6
7  # ## Configurando pyspark:
8
9  # In[1]:
10
11
12 #!pip install pyspark
13
14
15 # In[2]:
16
17
18 from pyspark.sql import SparkSession
19
```

```
20
21 # In[3]:
22
23
24 #Define onde sera criado a sessao do spark, nesse caso sera local
25 spark =
      SparkSession.builder.master('local[*]').appName("Classificacao
      com Spark").getOrCreate()
26
27
28 # In[4]:
29
30
31 spark
32
33
34 # ## Carregando os dados do csv:
35
36 # In[5]:
37
38
39 dados = spark.read.csv('presencial2016ColunaDesistentePublica.csv',
40 sep='|',header=True, inferSchema=True)
41
42
43 # In[6]:
44
45
46 dados.count()
47
48
49 # In[7]:
50
51
52 #Quantidade de Desistentes e Nao-Desistentes
53 dados.groupBy('DESISTENTE').count().show()
54
55
56 # ## Selecionando e tratando atributos:
57
58 # In[8]:
59
```

```
60
61 dados = dados.drop(*(
62 # 'CO_IES',
63   'NO_IES', #--Campo inutil
64 # 'CO_CATEGORIA_ADMINISTRATIVA',
65   'DS_CATEGORIA_ADMINISTRATIVA', #--Campo inutil
66 # 'CO_ORGANIZACAO_ACADEMICA',
67   'DS_ORGANIZACAO_ACADEMICA', #--Campo inutil
68 # 'CO_CURSO',
69   'NO_CURSO', #--Campo inutil
70   'CO_CURSO_POLO', #--Campo inutil (presencial)
71 # 'CO_TURNO_ALUNO',
72   'DS_TURNO_ALUNO', #--Campo inutil
73 # 'CO_GRAU_ACADEMICO',
74   'DS_GRAU_ACADEMICO', #--Campo inutil
75 # 'CO_MODALIDADE_ENSINO',
76   'DS_MODALIDADE_ENSINO', #--Campo inutil
77 # 'CO_NIVEL_ACADEMICO',
78   'DS_NIVEL_ACADEMICO', #--Campo inutil
79   'CO_OCDE', #--Nao tratado
80   'NO_OCDE', #--Campo inutil
81 # 'CO_OCDE_AREA_GERAL',
82   'NO_OCDE_AREA_GERAL', #--Campo inutil
83 # 'CO_OCDE_AREA_ESPECIFICA',
84   'NO_OCDE_AREA_ESPECIFICA', #--Campo inutil
85 # 'CO_OCDE_AREA_DETALHADA',
86   'NO_OCDE_AREA_DETALHADA', #--Campo inutil
87 # 'CO_ALUNO_CURSO',
88   'CO_ALUNO_CURSO_ORIGEM', #--Nao tratado
89   'CO_ALUNO', #--Campo inutil
90 # 'CO_COR_RACA_ALUNO',
91   'DS_COR_RACA_ALUNO', #--Campo inutil
92 # 'IN_SEXO_ALUNO',
93   'DS_SEXO_ALUNO', #--Campo inutil
94 # 'NU_ANO_ALUNO_NASC',
95 # 'NU_MES_ALUNO_NASC',
96 # 'NU_DIA_ALUNO_NASC',
97 # 'NU_IDADE_ALUNO',
98 # 'CO_NACIONALIDADE_ALUNO',
99   'DS_NACIONALIDADE_ALUNO', #--Campo inutil
100 # 'CO_PAIS_ORIGEM_ALUNO',
101   'CO_UF_NASCIMENTO', #--Nao tratado
```

```
102   'CO_MUNICIPIO_NASCIMENTO', #--Nao tratado
103 # 'IN_ALUNO_DEF_TGD_SUPER',
104 # 'IN_DEF_AUDITIVA',
105 # 'IN_DEF_FISICA',
106 # 'IN_DEF_INTELECTUAL',
107 # 'IN_DEF_MULTIPLA',
108 # 'IN_DEF_SURDEZ',
109 # 'IN_DEF_SURDOCEGUEIRA',
110 # 'IN_DEF_BAIXA_VISAO',
111 # 'IN_DEF_CEGUEIRA',
112 # 'IN_DEF_SUPERDOTACAO',
113 # 'IN_TGD_AUTISMO_INFANTIL',
114 # 'IN_TGD_SINDROME_ASPERGER',
115 # 'IN_TGD_SINDROME_RETT',
116 # 'IN_TGD_TRANSTOR_DESINTEGRATIVO',
117 # 'CO_ALUNO_SITUACAO',
118   'DS_ALUNO_SITUACAO', #--Campo inutil
119 # 'QT_CARGA_HORARIA_TOTAL',
120 # 'QT_CARGA_HORARIA_INTEG',
121   'DT_INGRESSO_CURSO', #--Nao tratado
122 # 'IN_ING_VESTIBULAR',
123 # 'IN_ING_ENEM',
124 # 'IN_ING_AVALIACAO_SERIADA',
125 # 'IN_ING_SELECAO_SIMPLIFICADA',
126 # 'IN_ING_SELECAO_VAGA_REMANESC',
127 # 'IN_ING_SELECAO_VAGA_PROG_ESPEC',
128 # 'IN_ING_TRANSF_EXOFFICIO',
129 # 'IN_ING_DECISAO_JUDICIAL',
130 # 'IN_ING_CONVENIO_PECG',
131 # 'IN_RESERVA_VAGAS',
132 # 'IN_RESERVA_ETNICO',
133 # 'IN_RESERVA_DEFICIENCIA',
134 # 'IN_RESERVA_ENSINO_PUBLICO',
135 # 'IN_RESERVA_RENDA_FAMILIAR',
136 # 'IN_RESERVA_OUTRA',
137 # 'IN_FINANC_ESTUDANTIL',
138 # 'IN_FIN_REEMB_FIES',
139 # 'IN_FIN_REEMB_ESTADUAL',
140 # 'IN_FIN_REEMB_MUNICIPAL',
141 # 'IN_FIN_REEMB_PROG_IES',
142 # 'IN_FIN_REEMB_ENT_EXTERNA',
143 # 'IN_FIN_REEMB_OUTRA',
```

```
144 # 'IN_FIN_NAOREEMB_PROUNI_INTEGR',
145 # 'IN_FIN_NAOREEMB_PROUNI_PARCIAL',
146 # 'IN_FIN_NAOREEMB_ESTADUAL',
147 # 'IN_FIN_NAOREEMB_MUNICIPAL',
148 # 'IN_FIN_NAOREEMB_PROG_IES',
149 # 'IN_FIN_NAOREEMB_ENT_EXTERNA',
150 # 'IN_FIN_NAOREEMB_OUTRA',
151 # 'IN_APOIO_SOCIAL',
152 # 'IN_APOIO_ALIMENTACAO',
153 # 'IN_APOIO_BOLSA_PERMANENCIA',
154 # 'IN_APOIO_BOLSA_TRABALHO',
155 # 'IN_APOIO_MATERIAL_DIDATICO',
156 # 'IN_APOIO_MORADIA',
157 # 'IN_APOIO_TRANSPORTE',
158 # 'IN_ATIVIDADE_EXTRACURRICULAR',
159 # 'IN_COMPL_ESTAGIO',
160 # 'IN_COMPL_EXTENSAO',
161 # 'IN_COMPL_MONITORIA',
162 # 'IN_COMPL_PESQUISA',
163 # 'IN_BOLSA_ESTAGIO',
164 # 'IN_BOLSA_EXTENSAO',
165 # 'IN_BOLSA_MONITORIA',
166 # 'IN_BOLSA_PESQUISA',
167 # 'CO_TIPO_ESCOLA_ENS_MEDIO',
168 # 'IN_ALUNO_PARFOR',
169  'CO_SEMESTRE_CONCLUSAO',
170 # 'CO_SEMESTRE_REFERENCIA',
171 # 'IN_MOBILIDADE_ACADEMICA',
172 # 'CO_MOBILIDADE_ACADEMICA',
173 # 'CO_MOBILIDADE_ACADEMICA_INTERN',
174   'CO_IES_DESTINO', #--Nao tratado
175   'CO_PAIS_DESTINO', #--Nao tratado
176   'IN_MATRICULA',
177 # 'IN_CONCLUINTE',
178 # 'IN_INGRESSO_TOTAL',
179 # 'IN_INGRESSO_VAGA_NOVA',
180 # 'ANO_INGRESSO',
181 # 'DESISTENTE'
182                     ))
183
184
185 # In[9]:
```

```
186
187
188 dados.printSchema()
189
190
191 # In[10]:
192
193
194 #dados.select('IN_DEF_INTELECTUAL').where('IN_DEF_INTELECTUAL is
       null').show()
195 #dados.groupBy('IN_DEF_INTELECTUAL').count().show()
196
197
198 # ## Formatando Atributos para processamento
199
200 # In[11]:
201
202
203 from pyspark.sql import functions as f
204
205 colunasParaCorrigir0 = [
206 'CO_TURNO_ALUNO'
207 ]
208
209 todasColunas0 = [f.when(f.col(c) == 1, 1).when(f.col(c) == 2, 2).
210                  when(f.col(c) == 3, 3).when(f.col(c) == 4, 4).
211                  otherwise(5).alias(c)
212                 for c in colunasParaCorrigir0]
213
214
215 # In[12]:
216
217
218 for coluna in dados.columns:
219     if coluna not in colunasParaCorrigir0:
220         todasColunas0.insert(0, coluna)
221
222
223 #todasColunas0
224
225 dados = dados.select(todasColunas0)
226 #dataset.groupBy('IN_DEF_INTELECTUAL').count().show()
```

```
227
228
229 # In[13]:
230
231
232 from pyspark.sql import functions as f
233
234 colunasParaCorrigir1 = [
235  #'CO_SEMESTRE_CONCLUSAO',
236  'CO_SEMESTRE_REFERENCIA',
237  'CO_MOBILIDADE_ACADEMICA',
238  'CO_MOBILIDADE_ACADEMICA_INTERN'
239 ]
240
241 todasColunas1 = [f.when(f.col(c) == 1, 1).when(f.col(c) == 2 ,
       2).otherwise(3).alias(c)
242                  for c in colunasParaCorrigir1]
243
244
245 # In[14]:
246
247
248 for coluna in dados.columns:
249     if coluna not in colunasParaCorrigir1:
250         todasColunas1.insert(0, coluna)
251
252
253 #todasColunas1
254
255 dados = dados.select(todasColunas1)
256 #dataset.groupBy('IN_DEF_INTELECTUAL').count().show()
257
258
259 # In[15]:
260
261
262 from pyspark.sql import functions as f
263
264 colunasParaCorrigir = [
265  'IN_ALUNO_DEF_TGD_SUPER',
266  'IN_DEF_AUDITIVA',
267  'IN_DEF_FISICA',
```

```
268  'IN_DEF_INTELECTUAL',
269  'IN_DEF_MULTIPLA',
270  'IN_DEF_SURDEZ',
271  'IN_DEF_SURDOCEGUEIRA',
272  'IN_DEF_BAIXA_VISAO',
273  'IN_DEF_CEGUEIRA',
274  'IN_DEF_SUPERDOTACAO',
275  'IN_TGD_AUTISMO_INFANTIL',
276  'IN_TGD_SINDROME_ASPERGER',
277  'IN_TGD_SINDROME_RETT',
278  'IN_TGD_TRANSTOR_DESINTEGRATIVO',
279
280  'IN_ING_VESTIBULAR',
281  'IN_ING_ENEM',
282  'IN_ING_AVALIACAO_SERIADA',
283  'IN_ING_SELECAO_SIMPLIFICADA',
284  'IN_ING_SELECAO_VAGA_REMANESC',
285  'IN_ING_SELECAO_VAGA_PROG_ESPEC',
286  'IN_ING_TRANSF_EXOFFICIO',
287  'IN_ING_DECISAO_JUDICIAL',
288  'IN_ING_CONVENIO_PECG',
289  'IN_RESERVA_VAGAS',
290  'IN_RESERVA_ETNICO',
291  'IN_RESERVA_DEFICIENCIA',
292  'IN_RESERVA_ENSINO_PUBLICO',
293  'IN_RESERVA_RENDA_FAMILIAR',
294  'IN_RESERVA_OUTRA',
295  'IN_FINANC_ESTUDANTIL',
296  'IN_FIN_REEMB_FIES',
297  'IN_FIN_REEMB_ESTADUAL',
298  'IN_FIN_REEMB_MUNICIPAL',
299  'IN_FIN_REEMB_PROG_IES',
300  'IN_FIN_REEMB_ENT_EXTERNA',
301  'IN_FIN_NAOREEMB_PROUNI_INTEGR',
302  'IN_FIN_NAOREEMB_PROUNI_PARCIAL',
303  'IN_FIN_NAOREEMB_ESTADUAL',
304  'IN_FIN_NAOREEMB_MUNICIPAL',
305  'IN_FIN_NAOREEMB_PROG_IES',
306  'IN_FIN_NAOREEMB_ENT_EXTERNA',
307  'IN_APOIO_SOCIAL',
308  'IN_APOIO_ALIMENTACAO',
309  'IN_APOIO_BOLSA_PERMANENCIA',
```

```
 'IN_APOIO_BOLSA_TRABALHO',
 'IN_APOIO_MATERIAL_DIDATICO',
 'IN_APOIO_MORADIA',
 'IN_APOIO_TRANSPORTE',
 'IN_ATIVIDADE_EXTRACURRICULAR',
 'IN_COMPL_ESTAGIO',
 'IN_COMPL_EXTENSAO',
 'IN_COMPL_MONITORIA',
 'IN_COMPL_PESQUISA',
 'IN_BOLSA_ESTAGIO',
 'IN_BOLSA_EXTENSAO',
 'IN_BOLSA_MONITORIA',
 'IN_BOLSA_PESQUISA',
 'CO_TIPO_ESCOLA_ENS_MEDIO',
 'IN_ALUNO_PARFOR',

 'IN_MOBILIDADE_ACADEMICA',

 #'IN_MATRICULA',
 'IN_CONCLUINTE',
 'IN_INGRESSO_TOTAL',
 'IN_INGRESSO_VAGA_NOVA',

 'IN_FIN_REEMB_OUTRA',
 'IN_FIN_NAOREEMB_OUTRA'

]

todasColunas = [f.when(f.col(c) == 0, 0).when(f.col(c) == 1 ,
   1).otherwise(2).alias(c)
                for c in colunasParaCorrigir]


# In[16]:


for coluna in dados.columns:
    if coluna not in colunasParaCorrigir:
        todasColunas.insert(0, coluna)


#todasColunas
```

```
351 dataset = dados.select(todasColunas)
352 #dataset.groupBy('IN_DEF_INTELECTUAL').count().show()
353
354
355 # ## Balanceamento dos dados:
356
357 # In[17]:
358
359
360 dados_not_desistente =
       dataset.select(todasColunas).where('DESISTENTE = 0')
361 print('dados nao desistente: '+str(dados_not_desistente.count()))
362
363 dados_not_desistente =
       dataset.select(todasColunas).where('DESISTENTE =
       0').sample(withReplacement=True, fraction=0.064, seed=101)
364 print('dados nao desistente fracionado:
       '+str(dados_not_desistente.count()))
365
366 dados_desistente = dataset.select(todasColunas).where('DESISTENTE =
       1')
367 print('dados desistente: '+str(dados_desistente.count()))
368
369
370 # In[18]:
371
372
373 ######################### UNIR AS BASES APOS BALANCEAMENTO
374 dados = dados_not_desistente.unionAll(dados_desistente)
375
376 dados.count()
377 ########################
378
379
380 # ## Informacoes de correlacao dos atributos:
381
382 # In[19]:
383
384
385 #So executar daqui para baixo
386 import plotly.express as px
387 dados_plot = dados
```

```
388 fig = px.imshow(dados_plot.toPandas().corr(), text_auto=True)
389 fig.show()
390
391
392 # In[20]:
393
394
395 import pandas as pd
396 pd.set_option('display.max_columns',  None)
397 pd.set_option('display.max_rows',  None)
398
399 dados_plot.toPandas().corr()
400
401
402 # In[21]:
403
404
405 import pandas as pd
406 import plotly.express as px
407
408 # Configuracao para exibir todas as colunas e linhas
409 pd.set_option('display.max_columns', None)
410 pd.set_option('display.max_rows', None)
411
412 # Calcula a matriz de correlacao
413 correlation_matrix = dados_plot.toPandas().corr()
414
415 # Obtem os 10 pares de atributos com a maior correlacao, excluindo a
    correlacao consigo mesmo
416 top_correlations = (correlation_matrix
417                     .unstack()
418                     .sort_values(ascending=False)
419                     .drop_duplicates()
420                     .head(16)  # Adiciona 1 para incluir a primeira
                           entrada (correlacao consigo mesmo)
421                     .tail(15))  # Exclui a correlacao consigo mesmo
422
423 # Exibe os pares de atributos
424 print("Top 10 pares de atributos com maior correlacao (excluindo a
    correlacao consigo mesmo):")
425 print(top_correlations)
426
```

```
427 ############################
428 # Filtra a matriz de correlacao apenas para os 10 pares de atributos
429 filtered_correlation_matrix =
       correlation_matrix.loc[top_correlations.index.get_level_values(0),
       top_correlations.index.get_level_values(1)]
430
431 # Cria o grafico de heatmap usando Plotly para os 10 pares de
       atributos
432 fig = px.imshow(filtered_correlation_matrix, text_auto=True)
433 fig.show()
434
435
436 # ## Definindo qual atributo e o label
437
438 # In[19]:
439
440
441 dados = dados.withColumnRenamed('DESISTENTE','label')
442
443
444 # In[20]:
445
446
447 x = dados.columns
448 x.remove('label')
449 len(x)
450
451
452 # ## Preparando vetor de entrada para a construcao dos modelos:
453
454 # In[21]:
455
456
457 from pyspark.ml.feature import VectorAssembler
458
459
460 # In[22]:
461
462
463 assembler = VectorAssembler(inputCols=x, outputCol='features')
464
465
```

```
# In[23]:


dados_preparados = assembler.setHandleInvalid("skip").
transform(dados).select('features','label')


# In[27]:


dados_preparados.show()


# ## Dividindo o dataset em Treino e Teste:

# In[24]:


SEED = 101


# In[25]:


treino, teste = dados_preparados.randomSplit([0.7, 0.3], seed=SEED)


# In[30]:


treino.show()


# In[31]:


treino.count()


# In[32]:


```

```
508 teste.count()
509
510
511 # # LogisticRegression
512
513 # In[40]:
514
515
516 from pyspark.ml.classification import LogisticRegression
517
518
519 # In[41]:
520
521
522 lr = LogisticRegression()
523
524
525 # In[35]:
526
527
528 from pyspark.ml import Pipeline
529
530 # Criar um pipeline, para que o CrossValidator utilize as mesmas
       etapas de pre-processamento com os mesmos dados utilizados
531 # para a geracao no modelo no pyspark
532 pipeline = Pipeline(stages=[assembler, lr])
533
534
535 # In[36]:
536
537
538 #Esse CrossValidator nao e para ajuste de hiperparametros, pois o
       pyspark ja faz isso,
539 # esse CrossValidator para captura de uma media de Acuracia dada por
       ele para saber se
540 # o modelo gerado no pyspark se sobressai sobre a media. Nos
       particionamos nossa base em
541 # 70% de treino e 30% teste de maneira que o modelo sera validado
       com esses dados de teste.
542 from sklearn.model_selection import cross_val_score
543 from sklearn.pipeline import Pipeline
544 from sklearn.preprocessing import StandardScaler
```

```
545 from sklearn.linear_model import LogisticRegression
546 from pyspark.ml.feature import VectorAssembler
547
548 # Converter DataFrame PySpark para um DataFrame Pandas
549 df_pandas = treino.toPandas()
550
551 # Extrair features e labels do DataFrame Pandas
552 X = df_pandas['features'].tolist()
553 y = df_pandas['label'].tolist()
554
555 # Criar um pipeline com um modelo de regressao logistica e um scaler
556 pipeline = Pipeline([
557     ('scaler', StandardScaler()),
558     ('lr', LogisticRegression(max_iter=1000))
559 ])
560
561 # Realizar a validacao cruzada usando Scikit-Learn
562 cross_val_result = cross_val_score(pipeline, X, y, cv=10)
563
564 # Imprimir os resultados da validacao cruzada
565 print("Resultados da validacao cruzada:", cross_val_result)
566 print("Acuracia media:", cross_val_result.mean())
567
568
569 # In[37]:
570
571
572 modelo_lr = lr.fit(treino)
573
574
575 # In[38]:
576
577
578 import pandas as pd
579 import matplotlib.pyplot as plt
580 import numpy as np
581
582 # Exibe os coeficientes atribuidos a cada atributo no modelo
583 coefficients = modelo_lr.coefficients
584
585 # Monta o dicionario, x e a lista de atributos que foi gerada
       algumas linhas acima
```

```
586 coefficients_dict = dict(zip(x, coefficients))
587
588 coefficients_df = pd.DataFrame({'Attribute': x, 'Coefficient':
        coefficients})
589 print(coefficients_df)
590
591 # ordena os atributos por ordem decrescente de importancia
592 coefficients_df.sort_values(by="Coefficient", ascending=False,
        inplace=True)
593
594 # divide as variaveis em positivas e negativas
595 positive_coefficients =
        coefficients_df[coefficients_df["Coefficient"] > 0].head(10)
596 negative_coefficients =
        coefficients_df[coefficients_df["Coefficient"] < 0].tail(10)
597
598 # combina as variaveis positivas e negativas
599 display_coefficients = pd.concat([positive_coefficients,
        negative_coefficients])
600
601 # plota o grafico de barras horizontais
602 # Em red sao as variaveis que influenciam no resultado negativo, ou
        seja, influenciam em o aluno nao desistir
603 # Em red green as variaveis que influenciam no resultado positivo,
        ou seja, influenciam em o aluno desistir
604 colors = np.array(["green"]*10 + ["red"]*10)
605 plt.barh(y=display_coefficients["Attribute"],
        width=display_coefficients["Coefficient"], height=0.8,
        color=colors)
606
607 # definindo o titulo do grafico e os rotulos dos eixos
608 plt.title("Importancia dos Atributos")
609 plt.xlabel("Coeficiente")
610 plt.ylabel("Atributo")
611
612 # exibe o grafico
613 plt.show()
614
615
616 # In[39]:
617
618
```

```
619 previsoes_lr_teste = modelo_lr.transform(teste)
620
621
622 # In[40]:
623
624
625 previsoes_lr_teste.show()
626
627
628 # In[41]:
629
630
631 # A funcao summary fornece estatisticas e informacoes sobre
632 # o modelo treinado com base nos dados de treinamento
633 # a previsao com os dados de teste e feita linhas abaixo
634 resumo_lr_treino = modelo_lr.summary
635
636
637 # In[42]:
638
639
640 print("Acuracia: %f" % resumo_lr_treino.accuracy)
641 print("Precisao: %f" % resumo_lr_treino.precisionByLabel[1])
642 print("Recall (Sensibilidade): %f"%
       resumo_lr_treino.recallByLabel[1])
643 print("F1: %f"% resumo_lr_treino.fMeasureByLabel()[1])
644
645
646 # ## Trazendo as metricas dos dados de teste usados de entrada para
       o modelo:
647
648 # In[43]:
649
650
651 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
652
653
654 # In[44]:
655
656
657 evaluator = MulticlassClassificationEvaluator()
658
```

```
659
660 # In[45]:
661
662
663 evaluator.evaluate(previsoes_lr_teste, {evaluator.metricName:
        'accuracy'})
664
665
666 # In[46]:
667
668
669 tp = evaluator.evaluate(previsoes_lr_teste, {evaluator.metricName:
        'truePositiveRateByLabel', evaluator.metricLabel: 1.0})
670 tn = evaluator.evaluate(previsoes_lr_teste, {evaluator.metricName:
        'truePositiveRateByLabel', evaluator.metricLabel: 0.0})
671 fp = evaluator.evaluate(previsoes_lr_teste, {evaluator.metricName:
        'falsePositiveRateByLabel', evaluator.metricLabel: 1.0})
672 fn = evaluator.evaluate(previsoes_lr_teste, {evaluator.metricName:
        'falsePositiveRateByLabel', evaluator.metricLabel: 0.0})
673
674
675 # In[47]:
676
677
678 print("VP: %f"% tp)
679 print("VN: %f"% tn)
680 print("FP: %f"% fp)
681 print("FN: %f"% fn)
682
683
684 # In[48]:
685
686
687 print('\033[1m'+'                          Predito\n'+'\033[0m')
688 print('      TP: ' + str(tp) +' '+ str(fp))
689 print('\033[1m'+'Real'+'\033[0m')
690 print('      TN: ' + str(fn) +' '+ str(tn))
691
692
693 # In[49]:
694
695
```

```
696 accuracy = (tp+tn)/(tp+tn+fp+fn)
697 print("Acuracia: %f" % accuracy)
698
699 precisao = (tp)/(tp+fp)
700 print("Precisao: %f" % precisao)
701
702 #Metrica que avalia a capacidade de classificar corretamente
703 #resultados classificados como positivos
704 sensibilidade = (tp)/(tp+fn)
705 print("Sensibilidade (Recall): %f" % sensibilidade)
706
707 #Metrica que avalia a capacidade de classificar corretamente
708 #resultados classificados como negativos
709 especificidade = (tn)/(tn+fp)
710 print("Especificidade: %f" % especificidade)
711
712
713 # # Decision Tree
714
715 # In[42]:
716
717
718 from pyspark.ml.classification import DecisionTreeClassifier
719
720
721 # In[43]:
722
723
724 dt = DecisionTreeClassifier(seed=SEED)
725
726
727 # In[52]:
728
729
730 from pyspark.ml import Pipeline
731
732 # Criar um pipeline, para que o CrossValidator utilize as mesmas
       etapas de pre-processamento com os mesmos dados utilizados
733 # para a geracao no modelo no pyspark
734 pipeline = Pipeline(stages=[assembler, dt])
735
736
```

```
737 # In[53]:
738
739
740 #Esse CrossValidator nao e para ajuste de hiperparametros, pois o
       pyspark ja faz isso,
741 # esse CrossValidator para captura de uma media de Acuracia dada por
       ele para saber se
742 # o modelo gerado no pyspark se sobressai sobre a media. Nos
       particionamos nossa base em
743 # 70% de treino e 30% teste de maneira que o modelo sera validado
       com esses dados de teste.
744 from sklearn.model_selection import cross_val_score
745 from sklearn.pipeline import Pipeline
746 from sklearn.preprocessing import StandardScaler
747 from sklearn.tree import DecisionTreeClassifier
748 from pyspark.ml.feature import VectorAssembler
749
750 # Converter DataFrame PySpark para um DataFrame Pandas
751 df_pandas = treino.toPandas()
752
753 # Extrair features e labels do DataFrame Pandas
754 X = df_pandas['features'].tolist()
755 y = df_pandas['label'].tolist()
756
757 # Criar um pipeline com uma arvore de decisao e um scaler
758 pipeline = Pipeline([
759     ('scaler', StandardScaler()),
760     ('dt', DecisionTreeClassifier())
761 ])
762
763 # Realizar a validacao cruzada usando Scikit-Learn
764 cross_val_result = cross_val_score(pipeline, X, y, cv=10)
765
766 # Imprimir os resultados da validacao cruzada
767 print("Resultados da validacao cruzada:", cross_val_result)
768 print("Acuracia media:", cross_val_result.mean())
769
770
771 # In[54]:
772
773
774 modelo_dt = dt.fit(treino)
```

```
775
776
777 # In[55]:
778
779
780 import pandas as pd
781 import matplotlib.pyplot as plt
782 import numpy as np
783
784 # Exibe os coeficientes atribuidos a cada atributo no modelo
785 importances = modelo_dt.featureImportances
786
787 # Monta o dicionario, x e a lista de atributos que foi gerada
       algumas linhas acima
788 importances_dict = dict(zip(x, importances))
789
790 importances_df = pd.DataFrame({'Attribute': x, 'Importance':
       importances})
791 print(importances_df)
792
793 # ordena os atributos por ordem decrescente de importancia
794 importances_df.sort_values(by="Importance", ascending=False,
       inplace=True)
795
796 # divide as variaveis em positivas e negativas
797 positive_coefficients = importances_df.head(10)
798 negative_coefficients = importances_df.tail(10)
799
800 # combina as variaveis positivas e negativas
801 display_coefficients = pd.concat([positive_coefficients,
       negative_coefficients])
802
803 # plota o grafico de barras horizontais
804 colors = ["green"] * 10 + ["red"] * 10
805 plt.barh(y=display_coefficients["Attribute"],
       width=display_coefficients["Importance"], height=0.8,
       color=colors)
806
807 # definindo o titulo do grafico e os rotulos dos eixos
808 plt.title("Importancia dos Atributos")
809 plt.xlabel("Importancia")
810 plt.ylabel("Atributo")
```

```
811
812 # exibe o grafico
813 plt.show()
814
815
816 # In[56]:
817
818
819 previsoes_dt_teste = modelo_dt.transform(teste)
820
821
822 # In[57]:
823
824
825 previsoes_dt_teste.show()
826
827
828 # # Trazendo as metricas dos dados de teste usados de entrada para o
       modelo:
829
830 # In[58]:
831
832
833 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
834
835
836 # In[59]:
837
838
839 evaluator = MulticlassClassificationEvaluator()
840
841
842 # In[60]:
843
844
845 evaluator.evaluate(previsoes_dt_teste, {evaluator.metricName:
       'accuracy'})
846
847
848 # In[61]:
849
850
```

```
851 tp = evaluator.evaluate(previsoes_dt_teste, {evaluator.metricName:
    'truePositiveRateByLabel', evaluator.metricLabel: 1.0})
852 tn = evaluator.evaluate(previsoes_dt_teste, {evaluator.metricName:
    'truePositiveRateByLabel', evaluator.metricLabel: 0.0})
853 fp = evaluator.evaluate(previsoes_dt_teste, {evaluator.metricName:
    'falsePositiveRateByLabel', evaluator.metricLabel: 1.0})
854 fn = evaluator.evaluate(previsoes_dt_teste, {evaluator.metricName:
    'falsePositiveRateByLabel', evaluator.metricLabel: 0.0})
855
856
857 # In[62]:
858
859
860 print("VP: %f"% tp)
861 print("VN: %f"% tn)
862 print("FP: %f"% fp)
863 print("FN: %f"% fn)
864
865
866 # In[63]:
867
868
869 print('\033[1m'+'           Predito\n'+'\033[0m')
870 print('      TP: ' + str(tp) +' '+ str(fp))
871 print('\033[1m'+'Real'+'\033[0m')
872 print('      TN: ' + str(fn) +' '+ str(tn))
873
874
875 # In[64]:
876
877
878 accuracy = (tp+tn)/(tp+tn+fp+fn)
879 print("Acuracia: %f" % accuracy)
880
881 precisao = (tp)/(tp+fp)
882 print("Precisao: %f" % precisao)
883
884 #Metrica que avalia a capacidade de classificar corretamente
885 #resultados classificados como positivos
886 sensibilidade = (tp)/(tp+fn)
887 print("Sensibilidade (Recall): %f" % sensibilidade)
888
```

```
889 #Metrica que avalia a capacidade de classificar corretamente
890 #resultados classificados como negativos
891 especificidade = (tn)/(tn+fp)
892 print("Especificidade: %f" % especificidade)
893
894
895 # # Random Forest
896
897 # In[44]:
898
899
900 from pyspark.ml.classification import RandomForestClassifier
901
902
903 # In[45]:
904
905
906 rf = RandomForestClassifier(seed=SEED)
907
908
909 # In[67]:
910
911
912 from pyspark.ml import Pipeline
913
914 # Criar um pipeline, para que o CrossValidator utilize as mesmas
       etapas de pre-processamento com os mesmos dados utilizados
915 # para a geracao no modelo no pyspark
916
917 pipeline = Pipeline(stages=[assembler, rf])
918
919
920 # In[68]:
921
922
923 #Esse CrossValidator nao e para ajuste de hiperparametros, pois o
       pyspark ja faz isso,
924 # esse CrossValidator para captura de uma media de Acuracia dada por
       ele para saber se
925 # o modelo gerado no pyspark se sobressai sobre a media. Nos
       particionamos nossa base em
926 # 70% de treino e 30% teste de maneira que o modelo sera validado
```

```
   com esses dados de teste.
927 from sklearn.model_selection import cross_val_score
928 from sklearn.pipeline import Pipeline
929 from sklearn.preprocessing import StandardScaler
930 from sklearn.ensemble import RandomForestClassifier  # Importar o
   RandomForestClassifier do scikit-learn
931 from pyspark.ml.feature import VectorAssembler
932
933 # Converter DataFrame PySpark para um DataFrame Pandas
934 df_pandas = treino.toPandas()
935
936 # Extrair features e labels do DataFrame Pandas
937 X = df_pandas['features'].tolist()
938 y = df_pandas['label'].tolist()
939
940 # Criar um pipeline com um modelo de Random Forest e um scaler
941 pipeline = Pipeline([
942     ('scaler', StandardScaler()),
943     ('rf', RandomForestClassifier(n_estimators=100,
        random_state=42))  # Numero de arvores pode ser ajustado
        conforme necessario
944 ])
945
946 # Realizar a validacao cruzada usando Scikit-Learn
947 cross_val_result = cross_val_score(pipeline, X, y, cv=10)
948
949 # Imprimir os resultados da validacao cruzada
950 print("Resultados da validacao cruzada:", cross_val_result)
951 print("Acuracia media:", cross_val_result.mean())
952
953
954 # In[69]:
955
956
957 modelo_rf = rf.fit(treino)
958
959
960 # In[70]:
961
962
963 import pandas as pd
964 import matplotlib.pyplot as plt
```

```
965 import numpy as np
966
967 # Exibe os coeficientes atribuidos a cada atributo no modelo
968 importances = modelo_rf.featureImportances
969
970 # Monta o dicionario, x e a lista de atributos que foi gerada
       algumas linhas acima
971 importances_dict = dict(zip(x, importances))
972
973 importances_rf = pd.DataFrame({'Attribute': x, 'Importance':
       importances})
974 print(importances_rf)
975
976 # ordena os atributos por ordem decrescente de importancia
977 importances_rf.sort_values(by="Importance", ascending=False,
       inplace=True)
978
979 # divide as variaveis em positivas e negativas
980 positive_coefficients = importances_rf.head(10)
981 negative_coefficients = importances_rf.tail(10)
982
983 # combina as variaveis positivas e negativas
984 display_coefficients = pd.concat([positive_coefficients,
       negative_coefficients])
985
986 # plota o grafico de barras horizontais
987 colors = ["green"] * 10 + ["red"] * 10
988 plt.barh(y=display_coefficients["Attribute"],
       width=display_coefficients["Importance"], height=0.8,
       color=colors)
989
990 # definindo o titulo do grafico e os rotulos dos eixos
991 plt.title("Importancia dos Atributos")
992 plt.xlabel("Importancia")
993 plt.ylabel("Atributo")
994
995 # exibe o grafico
996 plt.show()
997
998
999 # In[71]:
1000
```

```
1001
1002 previsoes_rf_teste = modelo_rf.transform(teste)
1003
1004
1005 # In[72]:
1006
1007
1008 previsoes_rf_teste.show()
1009
1010
1011 # In[73]:
1012
1013
1014 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
1015
1016
1017 # In[74]:
1018
1019
1020 evaluator = MulticlassClassificationEvaluator()
1021
1022
1023 # In[75]:
1024
1025
1026 evaluator.evaluate(previsoes_rf_teste, {evaluator.metricName:
        'accuracy'})
1027
1028
1029 # In[76]:
1030
1031
1032 tp = evaluator.evaluate(previsoes_rf_teste, {evaluator.metricName:
        'truePositiveRateByLabel', evaluator.metricLabel: 1.0})
1033 tn = evaluator.evaluate(previsoes_rf_teste, {evaluator.metricName:
        'truePositiveRateByLabel', evaluator.metricLabel: 0.0})
1034 fp = evaluator.evaluate(previsoes_rf_teste, {evaluator.metricName:
        'falsePositiveRateByLabel', evaluator.metricLabel: 1.0})
1035 fn = evaluator.evaluate(previsoes_rf_teste, {evaluator.metricName:
        'falsePositiveRateByLabel', evaluator.metricLabel: 0.0})
1036
1037
```

```
# In[77]:


print("VP: %f"% tp)
print("VN: %f"% tn)
print("FP: %f"% fp)
print("FN: %f"% fn)


# In[78]:


print('\033[1m'+'           Predito\n'+'\033[0m')
print('      TP: ' + str(tp) +' '+ str(fp))
print('\033[1m'+'Real'+'\033[0m')
print('      TN: ' + str(fn) +' '+ str(tn))


# In[79]:


accuracy = (tp+tn)/(tp+tn+fp+fn)
print("Acuracia: %f" % accuracy)

precisao = (tp)/(tp+fp)
print("Precisao: %f" % precisao)

#Metrica que avalia a capacidade de classificar corretamente
#resultados classificados como positivos
sensibilidade = (tp)/(tp+fn)
print("Sensibilidade (Recall): %f" % sensibilidade)

#Metrica que avalia a capacidade de classificar corretamente
#resultados classificados como negativos
especificidade = (tn)/(tn+fp)
print("Especificidade: %f" % especificidade)


# # SVM

# In[26]:

```

```
1080
1081 from pyspark.ml.classification import LinearSVC
1082
1083
1084 # In[27]:
1085
1086
1087 svm = LinearSVC(maxIter=10, regParam=0.1)
1088
1089
1090 # In[82]:
1091
1092
1093 from pyspark.ml import Pipeline
1094
1095 # Criar um pipeline, para que o CrossValidator utilize as mesmas
       etapas de pre-processamento com os mesmos dados utilizados
1096 # para a geracao no modelo no pyspark
1097
1098 pipeline = Pipeline(stages=[assembler, svm])
1099
1100
1101 # In[ ]:
1102
1103
1104 #Esse CrossValidator nao e para ajuste de hiperparametros, pois o
       pyspark ja faz isso,
1105 # esse CrossValidator para captura de uma media de Acuracia dada por
       ele para saber se
1106 # o modelo gerado no pyspark se sobressai sobre a media. Nos
       particionamos nossa base em
1107 # 70% de treino e 30% teste de maneira que o modelo sera validado
       com esses dados de teste.
1108 from sklearn.model_selection import cross_val_score
1109 from sklearn.pipeline import Pipeline
1110 from sklearn.preprocessing import StandardScaler
1111 from sklearn.svm import SVC  # Importar o Support Vector Classifier
       (SVC) do scikit-learn
1112 from pyspark.ml.feature import VectorAssembler
1113
1114 # Converter DataFrame PySpark para um DataFrame Pandas
1115 df_pandas = treino.toPandas()
```

```
1116
1117 # Extrair features e labels do DataFrame Pandas
1118 X = df_pandas['features'].tolist()
1119 y = df_pandas['label'].tolist()
1120
1121 # Criar um pipeline com um modelo de SVM e um scaler
1122 pipeline = Pipeline([
1123     ('scaler', StandardScaler()),
1124     ('svm', SVC(kernel='linear', C=1.0))  # Kernel e outros
        parametros podem ser ajustados conforme necessario
1125 ])
1126
1127 # Realizar a validacao cruzada usando Scikit-Learn
1128 cross_val_result = cross_val_score(pipeline, X, y, cv=10)
1129
1130 # Imprimir os resultados da validacao cruzada
1131 print("Resultados da validacao cruzada:", cross_val_result)
1132 print("Acuracia media:", cross_val_result.mean())
1133
1134
1135 # In[28]:
1136
1137
1138 modelo_svm = svm.fit(treino)
1139
1140
1141 # In[29]:
1142
1143
1144 import pandas as pd
1145 import matplotlib.pyplot as plt
1146 import numpy as np
1147
1148 # Exibe os coeficientes atribuidos a cada atributo no modelo
1149 coefficients = modelo_svm.coefficients
1150
1151 # Monta o dicionario, x e a lista de atributos que foi gerada
    algumas linhas acima
1152 coefficients_dict = dict(zip(x, coefficients))
1153
1154 coefficients_df = pd.DataFrame({'Attribute': x, 'Coefficient':
    coefficients})
```

```
1155 print(coefficients_df)
1156
1157 # ordena os atributos por ordem decrescente de importancia
1158 coefficients_df.sort_values(by="Coefficient", ascending=False,
        inplace=True)
1159
1160 # divide as variaveis em positivas e negativas
1161 positive_coefficients =
        coefficients_df[coefficients_df["Coefficient"] > 0].head(10)
1162 negative_coefficients =
        coefficients_df[coefficients_df["Coefficient"] < 0].tail(10)
1163
1164 # combina as variaveis positivas e negativas
1165 display_coefficients = pd.concat([positive_coefficients,
        negative_coefficients])
1166
1167 # plota o grafico de barras horizontais
1168 # Em red sao as variaveis que influenciam no resultado negativo, ou
        seja, influenciam em o aluno nao desistir
1169 # Em red green as variaveis que influenciam no resultado positivo,
        ou seja, influenciam em o aluno desistir
1170 colors = np.array(["green"]*10 + ["red"]*10)
1171 plt.barh(y=display_coefficients["Attribute"],
        width=display_coefficients["Coefficient"], height=0.8,
        color=colors)
1172
1173 # definindo o titulo do grafico e os rotulos dos eixos
1174 plt.title("Importancia dos Atributos")
1175 plt.xlabel("Coeficiente")
1176 plt.ylabel("Atributo")
1177
1178 # exibe o grafico
1179 plt.show()
1180
1181
1182 # In[30]:
1183
1184
1185 previsoes_svm_teste = modelo_svm.transform(teste)
1186
1187
1188 # In[31]:
```

```
1189
1190
1191 previsoes_svm_teste.show()
1192
1193
1194 # In[32]:
1195
1196
1197 from pyspark.ml.evaluation import MulticlassClassificationEvaluator
1198
1199
1200 # In[33]:
1201
1202
1203 evaluator = MulticlassClassificationEvaluator()
1204
1205
1206 # In[34]:
1207
1208
1209 evaluator.evaluate(previsoes_svm_teste, {evaluator.metricName:
       'accuracy'})
1210
1211
1212 # In[35]:
1213
1214
1215 tp = evaluator.evaluate(previsoes_svm_teste, {evaluator.metricName:
       'truePositiveRateByLabel', evaluator.metricLabel: 1.0})
1216 tn = evaluator.evaluate(previsoes_svm_teste, {evaluator.metricName:
       'truePositiveRateByLabel', evaluator.metricLabel: 0.0})
1217 fp = evaluator.evaluate(previsoes_svm_teste, {evaluator.metricName:
       'falsePositiveRateByLabel', evaluator.metricLabel: 1.0})
1218 fn = evaluator.evaluate(previsoes_svm_teste, {evaluator.metricName:
       'falsePositiveRateByLabel', evaluator.metricLabel: 0.0})
1219
1220
1221 # In[36]:
1222
1223
1224 print("VP: %f"% tp)
1225 print("VN: %f"% tn)
```

```
print("FP: %f"% fp)
print("FN: %f"% fn)


# In[37]:


print('\033[1m'+'           Predito\n'+'\033[0m')
print('      TP: ' + str(tp) +' '+ str(fp))
print('\033[1m'+'Real'+'\033[0m')
print('      TN: ' + str(fn) +' '+ str(tn))


# In[38]:


accuracy = (tp+tn)/(tp+tn+fp+fn)
print("Acuracia: %f" % accuracy)

precisao = (tp)/(tp+fp)
print("Precisao: %f" % precisao)

#Metrica que avalia a capacidade de classificar corretamente
#resultados classificados como positivos
sensibilidade = (tp)/(tp+fn)
print("Sensibilidade (Recall): %f" % sensibilidade)

#Metrica que avalia a capacidade de classificar corretamente
#resultados classificados como negativos
especificidade = (tn)/(tn+fp)
print("Especificidade: %f" % especificidade)


# ## Teste de Hipotese T-student

# In[46]:


from itertools import combinations
from scipy.stats import ttest_rel

algoritmos = ['lr', 'dt', 'rf', 'svm']
```

```
1268
1269 for alg1, alg2 in combinations(algoritmos, 2):
1270     observacoes_alg1_acc = []
1271     observacoes_alg2_acc = []
1272
1273     for i in range(10):
1274         # Treinamento e avaliacao do algoritmo 1
1275         modelo_alg1 = eval(alg1).fit(treino)
1276         previsoes_alg1_teste = modelo_alg1.transform(teste)
1277         acc_alg1 = evaluator.evaluate(previsoes_alg1_teste,
                {evaluator.metricName: 'accuracy'})
1278         observacoes_alg1_acc.append(acc_alg1)
1279
1280         # Treinamento e avaliacao do algoritmo 2
1281         modelo_alg2 = eval(alg2).fit(treino)
1282         previsoes_alg2_teste = modelo_alg2.transform(teste)
1283         acc_alg2 = evaluator.evaluate(previsoes_alg2_teste,
                {evaluator.metricName: 'accuracy'})
1284         observacoes_alg2_acc.append(acc_alg2)
1285
1286     # Realiza o teste t de Student
1287     t_statistic, p_value = ttest_rel(observacoes_alg1_acc,
            observacoes_alg2_acc)
1288
1289     # Compara o valor-p com o nivel de significancia alfa para
            aceitar ou rejeitar a hipotese nula
1290     alpha = 0.05
1291     h0 = f"Nao ha diferenca significativa no desempenho entre os
            modelos {alg1.upper()} e {alg2.upper()} (ACURACIA)"
1292     h1 = f"Ha diferenca significativa no desempenho entre os modelos
            {alg1.upper()} e {alg2.upper()} (ACURACIA)"
1293
1294     if p_value <= alpha:
1295         print(f"Rejeitamos a hipotese nula: {h0}, com um valor-p de
                {p_value:.3f}.")
1296         #print(f"Ha evidencia estatistica de que {h1}.")
1297     else:
1298         print(f"Aceitamos a hipotese nula: {h0}, com um valor-p de
                {p_value:.3f}.")
```

Ata da Sessão Solene de Defesa da Dissertação do Curso de Mestrado em Ciência da Computação-UFS.
Candidato: JEFERSON ANDRADE DE JESUS

Em 28 dias do mês de fevereiro do ano de dois mil e vinte quatro, com início às 09h, realizou-se na Sala de Seminários do PROCC da Universidade Federal de Sergipe, na Cidade Universitária Prof. José Aloísio de Campos, a Sessão Pública de Defesa de Dissertação de Mestrado do candidato **Jeferson Andrade de Jesus**, que desenvolveu o trabalho intitulado: "***Investigação da Evasão Estudantil por meio da Mineração de Dados e Aprendizagem de Máquina***", sob a orientação do Prof. Dr. Renê Pereira de Gusmão. A Sessão foi presidida pelo Prof. Dr. Renê Pereira de Gusmão (PROCC/UFS), que após a apresentação da dissertação passou a palavra aos outros membros da Banca Examinadora, Prof. Dr. Bruno Almeida Pimentel (UFAL) e, em seguida, o Prof. Dr. Gilton José Ferreira da Silva (Procc/UFS). Após as discussões, a Banca Examinadora reuniu-se e considerou o mestrando (a) APROVADO. Atendidas as exigências da Instrução Normativa 05/2019/PROCC, do Regimento Interno do PROCC (Resolução 67/2014/CONEPE), e da Resolução no 04/2021/CONEPE que regulamentam a Apresentação e Defesa de Dissertação, e nada mais havendo a tratar, a Banca Examinadora elaborou esta Ata que será assinada pelos seus membros e pelo mestrando.

Cidade Universitária "*Prof. José Aloísio de Campos",* 28 de fevereiro de 2024.

Prof. Dr. Renê Pereira de Gusmão
(PROCC/UFS)
Presidente

Prof. Dr. Gilton José Ferreira da Silva
(PROCC/UFS)
Examinador Interno

Prof. Dr. Bruno Almeida Pimentel
(UFAL)
Examinador Externo

Jeferson Andrade de Jesus
Candidato

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL
UNIVERSIDADE FEDERAL DE SERGIPE

Jesus, Jeferson Andrade de
J58i Investigação da evasão estudantil por meio da mineração de dados e aprendizagem de máquina / Jeferson Andrade de Jesus ; orientador Renê Pereira de Gusmão. - São Cristóvão, 2024.
256 f. : il.

Dissertação (mestrado em Ciência da Computação) – Universidade Federal de Sergipe, 2024.

1. Evasão universitária. 2. Mineração de dados (Computação). 3. Aprendizado do computador. I. Gusmão, Renê Pereira de orient. II. Título.

CDU 004.8