

Dissertação apresentada à Pró-Reitoria de Pós-Graduação do Instituto Tecnológico de Aeronáutica, como parte dos requisitos para obtenção do título de Mestre em Ciências no Programa de Pós-Graduação em Física, Área de Física Nuclear.

Gabriel Coelho Teles de Moura

**USO DE *MACHINE LEARNING* PARA
CLASSIFICAÇÃO DE EQUAÇÕES DE ESTADO DE
ESTRELAS DE NÊUTRONS**

Dissertação aprovada em sua versão final pelos a baixo assinados:



Prof. Dr. César Henrique Lenzi

Orientador

Documento assinado digitalmente
gov.br NADJA SIMAO MAGALHAES
Data: 06/02/2024 11:15:06-0300
Verifique em <https://validar.iti.gov.br>

Prof^a. Dr^a. Nadja Simão Magalhães

Coorientadora

Prof^a. Dr^a. Emília Villani

Pró-Reitora de Pós-Graduação

Campo Montenegro
São José dos Campos, SP - Brasil
2023

Dados Internacionais de Catalogação-na-Publicação (CIP)
Divisão de Informação e Documentação

Moura, Gabriel Coelho Teles de
Uso de *Machine Learning* para Classificação de Equações de Estado de Estrelas de Nêutrons /
Gabriel Coelho Teles de Moura.
São José dos Campos, 2023.
102f.

Dissertação de Mestrado – Curso de Física. Área de Física Nuclear – Instituto Tecnológico de Aeronáutica, 2023. Orientador: Prof. Dr. César Henrique Lenzi. Coorientadora: Prof^a. Dr^a. Nadjá Simão Magalhães.

1. Machine learning. 2. Estrela de neutron. 3. Equação de estado. I. Instituto Tecnológico de Aeronáutica. II. Título.

REFERÊNCIA BIBLIOGRÁFICA

MOURA, Gabriel Coelho Teles de. **Uso de *Machine Learning* para Classificação de Equações de Estado de Estrelas de Nêutrons**. 2023. 102f. Dissertação de Mestrado – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSÃO DE DIREITOS

NOME DO AUTOR: Gabriel Coelho Teles de Moura

TÍTULO DO TRABALHO: Uso de *Machine Learning* para Classificação de Equações de Estado de Estrelas de Nêutrons.

TIPO DO TRABALHO/ANO: Dissertação / 2023

É concedida ao Instituto Tecnológico de Aeronáutica permissão para reproduzir cópias desta dissertação e para emprestar ou vender cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte desta dissertação pode ser reproduzida sem a autorização do autor.

Gabriel Coelho Teles de Moura
Av. Mario Lopes Leão 952, Ap. 912-1
04.754-010 – São Paulo–SP

USO DE *MACHINE LEARNING* PARA CLASSIFICAÇÃO DE EQUAÇÕES DE ESTADO DE ESTRELAS DE NÊUTRONS

Gabriel Coelho Teles de Moura

Composição da Banca Examinadora:

Prof. Dr.	Pedro José Pompeia	Presidente	-	ITA
Prof. Dr.	César Henrique Lenzi	Orientador	-	ITA
Prof ^a . Dr ^a .	Nadja Simão Magalhães	Coorientadora	-	UNIFESP
Prof. Dr.	Rafael Augusto Couceiro Correa	Membro interno	-	ITA
Dr.	Rodolfo Valentim da Costa Lima	Membro externo	-	UNIFESP

Aos amigos da Graduação e Pós-Graduação do ITA aos meus pais e irmãos, mas principalmente ao meu marido Lucas por sempre me motivar a seguir em frente e nunca desistir.

Agradecimentos

Gostaria de agradecer a toda minha família que acompanha a minha jornada desde o início, mas principalmente aos meus pais por terem apoiado minhas escolhas.

Ao Lucas Pereira de Almeida, por sempre estar ao meu lado, seja na alegria e ou na dor, sempre me dando suporte e me impulsionando a sempre voar mais alto.

Aos meus amigos do ITA, pelos momentos de alegria e descontração.

Aos professores César e Nadja por terem topado seguir nesta aventura comigo, e por todo apoio e conhecimento compartilhado durante o desenvolvimento deste trabalho.

À CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pelo apoio financeiro concedido por meio desta bolsa de mestrado. Sua contribuição foi fundamental para a realização deste estudo e meu desenvolvimento acadêmico.

E por fim à Power of Data e ao Bruno Jardim pela confiança e todo o suporte técnico. Esta pesquisa não teria sido possível sem seu auxílio, e por este motivo gostaria de expressar minha mais profunda gratidão.

"We can only see a short distance ahead, but we can see plenty there that needs to be done."

— ALAN TURING

Resumo

Esta dissertação concentra-se na avaliação da eficácia de quatro algoritmos de classificação (*Gradient Boosting*, *Extreme Gradient Boosting*, *Light Gradient Boosting* e *Support Vector Machines*) na categorização de equações de estado de estrelas de nêutrons. O objetivo primordial é realizar essa categorização com base nas curvas massa-raio que coincidem com regiões observacionais específicas, nomeadamente as do NICER, LIGO, bem como aquelas que simultaneamente abrangem ambas as regiões. Como resultado deste estudo, foram desenvolvidos 12 modelos de classificação. Observou-se que o melhor modelo para a região LIGO foi o LGBM, enquanto para a região NICER e em ambas as regiões simultaneamente, o *gradient boosting* demonstrou ser a escolha mais eficaz. Esses modelos proporcionaram uma análise objetiva e imparcial das capacidades desses algoritmos no contexto altamente especializado das estrelas de nêutrons.

Palavras-chave: estrelas de nêutrons, equações de estado, aprendizado de máquina, classificação, curvas massa-raio, astrofísica.

Abstract

This dissertation focuses on assessing the effectiveness of four classification algorithms (Gradient Boosting, Extreme Gradient Boosting, Light Gradient Boosting, and Support Vector Machines) in categorizing equations of state for neutron stars. The primary objective is to perform this categorization based on mass-radius curves that align with specific observational regions, namely those of NICER, LIGO, as well as those simultaneously encompassing both regions. As a result of this study, 12 classification models were developed. It was observed that the best model for the LIGO region was the LGBM, while for the NICER region and in both regions simultaneously, the gradient boosting proved to be the most effective choice. These models provided an objective and unbiased analysis of the capabilities of these algorithms in the highly specialized context of neutron stars.

Keywords: neutron stars, equations of state, machine learning, classification, mass-radius curves, astrophysics.

Lista de Figuras

FIGURA 3.1 – Evolução da inteligência artificial.	25
FIGURA 3.2 – Aprendizagem Não Supervisionada	26
FIGURA 3.3 – Aprendizagem Supervisionada.	27
FIGURA 3.4 – Regressão Linear Simples e Múltipla.	28
FIGURA 3.5 – Diferentes tipos de classificação.	28
FIGURA 3.6 – <i>Label Encoding</i> e <i>One-Hot Encoding</i>	30
FIGURA 3.7 – Exemplos de normalização dos dados.	30
FIGURA 3.8 – Balanceamento de Classe	31
FIGURA 3.9 – Regressão Logística	33
FIGURA 3.10 – Árvore de decisão	34
FIGURA 3.11 – Comparação entre <i>Random Forest</i> e <i>Extra Trees</i>	35
FIGURA 3.12 – <i>Gradient Boosting</i>	36
FIGURA 3.13 – <i>Support Vector Machine</i>	38
FIGURA 3.14 – <i>Cross Validation</i>	39
FIGURA 3.15 – <i>Confusion Matrix</i>	40
FIGURA 3.16 – Curva ROC	43
FIGURA 3.17 – <i>Calibration Curve</i>	44
FIGURA 4.1 – Equação de Estado (esquerda) e sua respectiva curva M-R (direita)	49
FIGURA 4.2 – 10 mil Equações de Estado (esquerda) e Curvas M-R (direita) gera- das para o treinamento dos modelos	51
FIGURA 4.3 – Exemplo do processo de <i>Labeling</i>	52
FIGURA 4.4 – Equações de Estado e Curvas M-R com <i>label</i> 1 (laranja) e <i>label</i> 0 (cinza).	53

FIGURA 4.5 – Divisão dos dados entre treino e teste.	54
FIGURA 5.1 – <i>Confusion Matrix</i> do modelo <i>Gradient Boosting</i> - LIGO	57
FIGURA 5.2 – <i>Confusion Matrix</i> do modelo LGBM - LIGO	58
FIGURA 5.3 – <i>Confusion Matrix</i> do modelo XGBoost - LIGO	58
FIGURA 5.4 – <i>Confusion Matrix</i> do modelo SVM - LIGO	58
FIGURA 5.5 – Curva ROC do modelo <i>Gradient Boosting</i>	62
FIGURA 5.6 – Curva ROC do modelo LGBM	62
FIGURA 5.7 – Curva ROC do modelo XGBoost	63
FIGURA 5.8 – Curva ROC do modelo SVM	63
FIGURA 5.9 – Curva de calibração do modelo <i>Gradient Boosting</i>	65
FIGURA 5.10 – Curva de calibração do modelo LGBM	65
FIGURA 5.11 – Curva de calibração do modelo XGBoost	65
FIGURA 5.12 – Curva de calibração do modelo SVM	66
FIGURA 5.13 – Performance geral do modelo LGBM para o caso LIGO	67
FIGURA 5.14 – <i>Confusion Matrix</i> do modelo <i>Gradient Boosting</i> - NICER	67
FIGURA 5.15 – <i>Confusion Matrix</i> do modelo LGBM - NICER	68
FIGURA 5.16 – <i>Confusion Matrix</i> do modelo XGBoost - NICER	68
FIGURA 5.17 – <i>Confusion Matrix</i> do modelo SVM - NICER	68
FIGURA 5.18 – Curva ROC do modelo <i>Gradient Boosting</i> - NICER	71
FIGURA 5.19 – Curva ROC do modelo LGBM - NICER	72
FIGURA 5.20 – Curva ROC do modelo XGBoost - NICER	72
FIGURA 5.21 – Curva ROC do modelo SVM - NICER	73
FIGURA 5.22 – Curva de calibração do modelo <i>Gradient Boosting</i> - NICER	74
FIGURA 5.23 – Curva de calibração do modelo LGBM - NICER	74
FIGURA 5.24 – Curva de calibração do modelo XGBoost - NICER	75
FIGURA 5.25 – Curva de calibração do modelo SVM - NICER	75
FIGURA 5.26 – Performance geral do modelo <i>Gradient Boosting</i> para o caso NICER	76
FIGURA 5.27 – <i>Confusion Matrix</i> do modelo <i>Gradient Boosting</i> - LIGO e NICER	76
FIGURA 5.28 – <i>Confusion Matrix</i> do modelo LGBM - LIGO e NICER	77

FIGURA 5.29 – <i>Confusion Matrix</i> do modelo XGBoost - LIGO e NICER	77
FIGURA 5.30 – <i>Confusion Matrix</i> do modelo SVM - LIGO e NICER	77
FIGURA 5.31 – Curva ROC do modelo <i>Gradient Boosting</i> - LIGO e NICER	80
FIGURA 5.32 – Curva ROC do modelo LGBM - LIGO e NICER	80
FIGURA 5.33 – Curva ROC do modelo XGBoost - LIGO e NICER	81
FIGURA 5.34 – Curva ROC do modelo SVM - LIGO e NICER	81
FIGURA 5.35 – Curva de calibração do modelo <i>Gradient Boosting</i> - LIGO e NICER	82
FIGURA 5.36 – Curva de calibração do modelo LGBM - LIGO e NICER	83
FIGURA 5.37 – Curva de calibração do modelo XGBoost - LIGO e NICER	83
FIGURA 5.38 – Curva de calibração do modelo SVM - LIGO e NICER	83
FIGURA 5.39 – Performance geral do modelo <i>Gradient Boosting</i> para o caso LIGO e NICER	84

Lista de Tabelas

TABELA 4.1 – Faixas de densidade de energia	50
TABELA 4.2 – Forma que as equações são apresentadas aos modelos	53
TABELA 5.1 – <i>Cross Validation</i> do modelo <i>Gradient Boosting</i> - LIGO	59
TABELA 5.2 – <i>Cross Validation</i> do modelo LGBM - LIGO	59
TABELA 5.3 – <i>Cross Validation</i> do modelo XGBoost - LIGO	60
TABELA 5.4 – <i>Cross Validation</i> do modelo SVM - LIGO	60
TABELA 5.5 – Comparação da média para os modelos da região LIGO	60
TABELA 5.6 – Comparação do desvio padrão para os modelos da região LIGO	61
TABELA 5.7 – Comparação dos modelos com os dados de teste - LIGO	64
TABELA 5.8 – <i>Cross Validation</i> do modelo <i>Gradient Boosting</i> - NICER	69
TABELA 5.9 – <i>Cross Validation</i> do modelo LGBM - NICER	69
TABELA 5.10 – <i>Cross Validation</i> do modelo XGBoost - NICER	70
TABELA 5.11 – <i>Cross Validation</i> do modelo SVM - NICER	70
TABELA 5.12 – Comparação da média para os modelos da região NICER	70
TABELA 5.13 – Comparação do desvio padrão para os modelos da região NICER	70
TABELA 5.14 – Comparação dos modelos com os dados de teste - NICER	73
TABELA 5.15 – <i>Cross Validation</i> do modelo <i>Gradient Boosting</i> - LIGO e NICER	78
TABELA 5.16 – <i>Cross Validation</i> do modelo LGBM - LIGO e NICER	78
TABELA 5.17 – <i>Cross Validation</i> do modelo XGBoost - LIGO e NICER	78
TABELA 5.18 – <i>Cross Validation</i> do modelo SVM - LIGO e NICER	79
TABELA 5.19 – Comparação da média para os modelos das regiões LIGO e NICER	79

TABELA 5.20 –Comparação do desvio padrão para os modelos das regiões LIGO e NICER	79
TABELA 5.21 –Comparação dos modelos com os dados de teste - LIGO e NICER .	82

Lista de Abreviaturas e Siglas

AUC	<i>Area Under Curve</i>
CM	<i>Confusion Matrix</i>
FPR	<i>False Positive Rate</i>
FN	<i>False Negative</i>
FP	<i>False Positive</i>
GB	<i>Gradient Boosting</i>
KS	<i>Kolmogorov-Smirnov</i>
LGBM	<i>Light Gradient Boosting</i>
LIGO	<i>Laser Interferometer Gravitational-Wave Observatory</i>
MCC	<i>Matthews's Correlation Coefficient</i>
M-R	Massa-Raio
NICER	<i>The Neutron Star Interior Composition Explorer</i>
RBF	<i>Radial Basis Function</i>
ROC	<i>Receiver Operating Characteristic</i>
SMOTE	<i>Synthetic Minority Over-Sampling Technique</i>
SVM	<i>Support Vector Machine</i>
TOV	Tolman-Oppenheimer-Volkoff
TP	<i>True Positive</i>
TN	<i>True Negative</i>
TPR	<i>True Positive Rate</i>
XGB	<i>Extreme Gradient Boosting</i>

Lista de Símbolos

M_{\odot}	Massa solar
c	Velocidade da luz
G	Constante gravitacional universal
ϵ	Densidade de energia
$T_{\mu\nu}$	Tensor energia-momento
M	Massa total
γ	Índice adiabático
k	Constante de proporcionalidade
c_s	Velocidade do som

Sumário

1	INTRODUÇÃO	18
2	ESTRELAS DE NÊUTRONS	20
2.1	Equação de Tolman-Oppenheimer-Volkoff	21
2.2	Equação de Estado	23
3	APRENDIZADO DE MÁQUINA	25
3.1	Tipos de aprendizagem	26
3.1.1	Aprendizagem Não Supervisionada	26
3.1.2	Aprendizagem Supervisionada	27
3.2	Tratamento dos Dados	29
3.2.1	<i>Encoding</i>	29
3.2.2	Normalização	30
3.2.3	Balanceamento de Classe	31
3.3	Algoritmos de Classificação	32
3.3.1	Regressão Logística	32
3.3.2	Árvores de Decisão	34
3.3.3	<i>Extra Trees</i> e <i>Random Forest</i>	35
3.3.4	<i>Gradient Boosting</i>	35
3.3.5	<i>Support Vector Machines</i>	37
3.4	Métricas de Avaliação dos Modelos	38
3.4.1	<i>Cross Validation</i>	39
3.4.2	<i>Confusion Matrix</i>	40
3.4.3	Acurácia	41

3.4.4	<i>Precision</i>	41
3.4.5	<i>Recall</i>	41
3.4.6	<i>F1-Score</i>	42
3.4.7	Curva ROC	42
3.4.8	<i>Calibration Curve</i>	44
3.4.9	Gini	45
3.4.10	<i>Kappa</i>	45
3.4.11	MCC	46
3.4.12	KS	46
4	METODOLOGIA	47
4.1	Geração dos Dados Sintéticos	48
4.2	Etiquetagem e Preparação dos Dados	51
4.3	Divisão dos Dados	54
4.4	Treinamento dos Modelos	55
4.5	Avaliação dos Modelos	55
5	RESULTADOS	57
5.1	Modelos para a região LIGO	57
5.2	Modelos para a região NICER	67
5.3	Modelos para as regiões LIGO e NICER	76
6	CONCLUSÃO	85
	REFERÊNCIAS	87
	APÊNDICE A – CÓDIGO PARA GERAÇÃO DAS EQUAÇÕES DE ESTADO E OBTENÇÃO DAS CURVAS M-R	92
	APÊNDICE B – CÓDIGO PARA VERIFICAÇÕES DAS CURVAS M-R E ORGANIZAÇÃO DOS DADOS	96

1 Introdução

A interseção fascinante entre a astrofísica e a ciência de dados representa um território que oferece uma promessa imensa e instigante: desvendar os mistérios do cosmos por meio de uma abordagem inovadora e analítica, que incorpora as melhores práticas de duas disciplinas aparentemente distintas. A busca pela compreensão das estrelas de nêutrons, essas entidades astronômicas extremamente compactas e densas, é uma jornada que há décadas desafia os limites do conhecimento e leva a humanidade a explorar os recantos mais profundos do universo.

A trajetória histórica da astrofísica trilhou um caminho que passa desde as observações primordiais realizadas com telescópios rudimentares até os modernos observatórios espaciais, que oferecem vistas sem precedentes do cosmos. No entanto, as estrelas de nêutrons, apesar de serem uma das maravilhas mais intrigantes do universo, continuam a desafiar os limites da compreensão humana. Elas surgem como os destroços colapsados de estrelas massivas, nascidas em supernovas espetaculares, e o que resta após a explosão é uma esfera incrivelmente pequena, porém extraordinariamente densa, composta principalmente por nêutrons. As propriedades físicas dessas estrelas de nêutrons são verdadeiramente extremas, praticamente uma massa equivalente à do Sol, mas comprimida em um diâmetro de apenas alguns quilômetros. A densidade nesse núcleo estelar é inimaginável, rivalizando com a de um núcleo atômico. Os campos magnéticos que emanam dessas estrelas são intensos o suficiente para distorcer a própria estrutura do espaço-tempo, criando ambientes onde as leis da física adotam nuances únicas.

Em paralelo a essa busca constante por compreender as estrelas de nêutrons, a evolução exponencial da ciência de dados emergiu como um poderoso aliado. A revolução digital, marcada por avanços na capacidade de processamento de dados, trouxe consigo uma profusão de ferramentas e técnicas capazes de extrair *insights* valiosos dos vastos conjuntos de dados gerados pelo universo. Os observatórios espaciais modernos, como o Telescópio Espacial James Webb e o Observatório de Raios-X Chandra, produzem volumes enormes de dados astronômicos que capturam eventos cósmicos desde a explosão de supernovas até a fusão de buracos negros. Essas observações detalhadas não apenas expandiram o conhecimento sobre o cosmos, mas também apresentaram desafios computacionais significativos na análise desses dados. Os métodos de aprendizado de máquina, em particular,

surgiram como uma promissora ferramenta nessa análise de dados astronômicos. Ao utilizar algoritmos de aprendizado de máquina, é possível decifrar padrões complexos e revelar correlações sutis em meio a uma profusão de informações. Esses métodos têm a capacidade de automatizar a detecção de eventos astronômicos, a classificação de objetos celestes e a previsão de comportamentos estelares. Eles não apenas aceleram a análise de dados, mas também permitem que os astrônomos explorem aspectos do universo que anteriormente eram difíceis de discernir. E é neste contexto que esta dissertação se propõe a unir a expertise da astrofísica de estrelas de nêutrons com a sofisticação da ciência de dados. Por meio da aplicação de modelos de aprendizado de máquina na classificação de equações de estado dessas estrelas com base em dados observacionais, essa pesquisa almeja não apenas expandir no conhecimento sobre essas entidades estelares notáveis, mas também exemplificar como a fusão desses dois campos aparentemente distintos pode resultar em descobertas poderosas e transformadoras.

O foco central desta pesquisa reside na aplicação de algoritmos avançados de classificação para investigar as equações de estado que governam a formação de curvas de massa-raio de estrelas de nêutrons. Especificamente, o objetivo é determinar quais equações de estado são consistentes com as regiões observacionais estabelecidas por observatórios como o NICER e o LIGO. Essas regiões observacionais são cruciais, pois representam as fronteiras do conhecimento atual, onde as estrelas de nêutrons reais podem ser encontradas. Ao aplicar métodos de aprendizado de máquina, esta pesquisa busca classificar com precisão as equações de estado com base nestes dados observacionais, permitindo uma compreensão mais sólida das propriedades físicas dessas estrelas extraordinárias. Para tal, o Capítulo 2 dedica-se à apresentação dos principais conceitos relacionados às estrelas de nêutrons, incluindo as equações de estado e a equação de Tolman-Oppenheimer-Volkoff. O Capítulo 3, por sua vez, concentra-se na exposição dos conceitos relativos à aprendizagem de máquina, abrangendo os diferentes tipos de aprendizado, abordagens de pré-processamento de dados e algoritmos de classificação. O Capítulo 4 abrange detalhadamente a metodologia adotada neste estudo, descrevendo os procedimentos e técnicas empregados na geração de dados sintéticos para o treinamento dos modelos e a preparação dos dados. Finalmente, no Capítulo 5, são apresentados e discutidos os resultados obtidos ao longo da pesquisa.

2 Estrelas de Nêutrons

A descoberta do nêutron como uma partícula elementar em 1932 garantiu a Sir James Chadwick o Nobel de Física em 1935 (CHADWICK, 1932), mas antes disso, em 1933, o nêutron já estaria envolvido em uma nova especulação. Walter Baade e Fritz Zwicky estavam a procura de uma possível explicação para a criação das supernovas, explosões altamente luminosas capazes de ofuscar até mesmo a luz de galáxias inteiras. Sua proposta era de que estas explosões eram originadas no estágio final da vida de uma estrela, e que a energia liberada por esta estrela era o que alimentava as supernovas (BAADE; ZWICKY, 1933). O que restava após este processo seria uma estrela formada quase em sua totalidade por nêutrons.

As estrelas de nêutrons são formadas durante o colapso de estrelas massivas, e são um dos poucos finais possíveis para a evolução estelar (sendo as demais possibilidades a formação de uma anã branca ou um buraco negro (SHAPIRO; TEUKOLSKY, 1983)). Quando já não há mais combustível e a pressão gerada pelas reações nucleares no núcleo dessa estrela (que geralmente tem pelo menos 8 vezes a massa do Sol) não é mais suficiente para compensar a força de sua própria gravidade, então ela entra em colapso. A estrela explode formando uma supernova, e o que sobra é apenas um núcleo formado pelos nêutrons originados da junção de prótons e elétrons devido a altíssima força gravitacional. Tipicamente uma estrela de nêutrons tem a massa em torno de $1.34 M_{\odot}$ e $2.35 M_{\odot}$ com raio entre 10 km e 20 km (RILEY *et al.*, 2019; RILEY *et al.*, 2021; MILLER *et al.*, 2019; MILLER *et al.*, 2021; ROMANI *et al.*, 2022), e densidade entre $8 \times 10^{13} \text{ g/cm}^3$ e $2 \times 10^{15} \text{ g/cm}^3$, próxima à densidade do núcleo atômico.

Algumas das propriedades das estrelas de nêutrons, como sua massa e seu raio, são obtidas resolvendo-se equações de equilíbrio hidrostático dentro do contexto relativístico. Este capítulo tem como objetivo discutir estas equações, bem como definir a equação de estado dentro do contexto da relatividade geral. Para tal, serão usadas unidades naturais, em que $G = c = 1$.

2.1 Equação de Tolman-Oppenheimer-Volkoff

É necessário resolver as equações de Einstein no interior da estrela de nêutrons para obter suas propriedades de equilíbrio, como massa e raio. O ponto de partida é considerar a estrela como esfericamente simétrica e estática, ou seja, é possível determinar um sistema de coordenadas onde a métrica é invariante sob rotação, além de ser também invariante sobre inversões temporais (não depende explicitamente da variável temporal). O elemento de linha da métrica é então da forma

$$ds^2 = -e^{\nu(r)} dt^2 + e^{\lambda(r)} dr^2 + r^2(d\theta^2 + \sin^2 \theta d\phi^2), \quad (2.1)$$

onde $\nu(r)$ e $\lambda(r)$ são funções a serem determinadas levando em consideração as simetrias em questão.

O próximo passo é considerar a matéria da estrela como sendo um fluido perfeito, livre de viscosidade, sem fluxo de energia, com densidade total ϵ , e com apenas uma pressão isotrópica P . Seu tensor energia-momento pode então ser escrito da forma

$$T_{\mu\nu} = \epsilon U_\mu U_\nu + P(g_{\mu\nu} + U_\mu U_\nu), \quad (2.2)$$

onde U_μ representa a quadri-velocidade do fluido, indicando que ϵ e P devem ser medidas no seu referencial de repouso (GLEDENNING, 2000). Se tratando de uma estrela estática, espera-se que a quadri-velocidade tenha apenas a componente temporal, e impondo a normalização

$$U^\mu U_\mu = -1 \quad (2.3)$$

obtemos $U_\mu = (e^{\nu/2}, 0, 0, 0)$, de modo que o tensor energia-momento tenha forma diagonal

$$[T_{\mu\nu}] = \begin{bmatrix} e^{\nu(r)}\epsilon & 0 & 0 & 0 \\ 0 & e^{\lambda(r)}P & 0 & 0 \\ 0 & 0 & r^2P & 0 \\ 0 & 0 & 0 & r^2 \sin^2 \theta P \end{bmatrix}. \quad (2.4)$$

Na região externa da estrela, onde não há matéria, a solução é a usual métrica de Schwarzschild (SCHWARZSCHILD, 1916)

$$ds^2 = -\left(1 - \frac{2M}{r}\right) dt^2 + \left(1 - \frac{2M}{r}\right)^{-1} dr^2 + r^2 (d\theta^2 + \sin^2 \theta d\phi^2), \quad (2.5)$$

onde M é a massa total da estrela, sendo

$$M = 4\pi \int_0^R \epsilon(r) r^2 dr. \quad (2.6)$$

Voltando a atenção para as equações de Einstein, primeiramente para componente G_{tt}

$$G_{tt} = \frac{e^{-\lambda(r)+\nu(r)}(-1 + e^{\lambda(r)} + r\partial_r\lambda(r))}{r^2} = 8\pi T_{tt} = 8\pi e^{\nu(r)}\epsilon$$

$$e^{-\lambda(r)}(-1 + e^{\lambda(r)} + r\partial_r\lambda(r)) = 8\pi\epsilon r^2, \quad (2.7)$$

e identificando

$$e^{-\lambda(r)} = 1 - \frac{2M(r)}{r}, \quad (2.8)$$

$$\frac{dM(r)}{dr} = 4\pi\epsilon r^2, \quad (2.9)$$

é possível visualizar que a 2.7 é uma generalização do caso Schwarzschild.

Para a componente G_{rr} obtém-se

$$G_{rr} = \frac{1 - e^{\lambda(r)} + r\partial_r\nu(r)}{r^2} = 8\pi T_{rr} = 8\pi e^{\lambda(r)}P$$

$$\frac{d\nu(r)}{dr} = 2\frac{M(r) + 4\pi r^3 P}{r(r - 2M(r))}, \quad (2.10)$$

que diferente de $\lambda(r)$ não é uma generalização do caso Schwarzschild.

Partindo da conservação do tensor energia-momento (CARROLL, 2004), é possível encontrar uma equação de campo para P , utilizando do fato de que há apenas a componente $\nu = r$ como não trivial, o que resulta em

$$\frac{dP}{dr} = -\frac{(\epsilon + P)}{2} \frac{d\nu(r)}{dr}$$

$$\frac{dP}{dr} = -\frac{(\epsilon + P)(M(r) + 4\pi r^3 P)}{r^2} \left(1 - \frac{2M(r)}{r}\right)^{-1}. \quad (2.11)$$

A 2.11, juntamente com 2.9, são conhecidas como equações de Tolman-Oppenheimer-Volkov (OPPENHEIMER; VOLKOFF, 1916), que descrevem a massa e a pressão de objetos esféricamente simétricos e estáticos de material isotrópico em equilíbrio gravitacional. É importante notar que, no limite newtoniano, $P \ll \epsilon$, $Pr^3 \ll M(r)$ e $2M(r) \ll r$, de modo que para estrelas não relativísticas recuperamos as expressões newtonianas

$$\frac{dP}{dr} = -\frac{\epsilon M(r)}{r^2}, \quad \frac{dM(r)}{dr} = 4\pi r^2 \epsilon, \quad \frac{d\nu(r)}{dr} = \frac{M(r)}{r^2}. \quad (2.12)$$

As equações de Tolman-Oppenheimer-Volkov (TOV) devem ser integradas utilizando as condições de contorno $P(0) = P_c$ e $M(0) = 0$, onde P_c e $M(0)$ são a pressão e a massa da estrela em $r = 0$, até que $P(r = R) = 0$, sendo que $r = R$ define a superfície da estrela

onde a pressão é nula, mantendo assim a continuidade com a solução no seu exterior

$$e^{\nu(R)} = e^{-\lambda(R)} = 1 - \frac{2M}{R}. \quad (2.13)$$

Deste modo, para cada valor de densidade central é possível determinar a massa M e o raio R de uma estrela de nêutrons, desde que se tenha uma equação de estado, ou seja, uma equação que relacione a pressão com a densidade de energia e temperatura ($P = P(\epsilon, T)$), e que determina o comportamento termodinâmico do fluido.

2.2 Equação de Estado

Para que uma estrela seja estável é necessário que sua própria gravidade seja compensada pela pressão oriunda das reações nucleares em seu interior. À medida que seu combustível vai se esgotando, a pressão interna não é mais suficiente para equilibrar sua força gravitacional, de forma que seu interior é comprimido cada vez mais. Com o aumento da densidade da estrela, há o surgimento de pressões de origem não térmica (HAENSEL *et al.*, 2007).

Quando a densidade alcança cerca de $10^8 - 10^{11}$ kg/m³ e a região disponível para os elétrons se torna cada vez menor, e de acordo com o princípio da incerteza de Heisenberg há um aumento no momento (e conseqüentemente na energia), mas sendo férmions os elétrons também devem obedecer ao princípio de exclusão de Pauli, isto é, dois elétrons só podem ocupar o mesmo nível de energia caso tenham spins opostos. É esta pressão estritamente quântica dos elétrons altamente degenerados que sustenta a gravidade da estrela, que nestas condições é dita uma anã branca caso sua massa seja próxima a do Sol. Caso a estrela seja massiva o bastante, sua gravidade é capaz de combinar os elétrons e prótons em seu interior, formando nêutrons que, também sendo férmions, geram uma pressão similar àquela dos elétrons (HAENSEL *et al.*, 2007).

É intuitivo calcular a equação de estado para um gás de férmions livres e não interagentes. Neste caso a temperatura tem pouca relevância na equação de estado, já que no caso degenerado a energia de Fermi supera a energia média das partículas, e somado ao fato de que estrelas de nêutrons mais velhas possuem uma baixa temperatura graças à emissão de neutrinos, então é possível aproximar $T = 0$ K (SABBATA; GASPERINI, 1985).

Inicialmente considera-se que todos os elétrons estão confinados em uma caixa de volume V . Devido ao princípio da incerteza, tem-se uma incerteza no momento da ordem $\delta p = hV^{-1/3}$, sendo h a constante de Planck. O número de estados ocupados no espaço

de momento, já considerando que se tratam de férmions de spin 1/2, é

$$dN = \frac{8\pi p^2 V}{h^3} dp, \quad (2.14)$$

ou seja, o número máximo de férmions que podem ocupar um mesmo estado com momento p . Integrando a 2.14 de um momento $p = 0$ até $p = p_F$, dito momento de Fermi, obtém-se

$$\frac{N}{V} = \int_0^{p_F} \frac{8\pi p^2 dp}{h^3} = \frac{8\pi p_F^3}{3h^3}. \quad (2.15)$$

Sabendo que a expressão relativística para a energia do férmion é $E = (p^2 + m^2)^{1/2}$, sendo m a massa do elétron ou do nêutron, pode-se obter a densidade de energia total do gás de férmions

$$\epsilon = \frac{E}{V} = \int_0^{p_F} \frac{8\pi p^2 dp}{h^3} (p^2 + m^2)^{1/2}. \quad (2.16)$$

Para obter a pressão basta utilizar a primeira lei da termodinâmica com o número de partículas fixo e sem dependência da temperatura,

$$P = -\frac{d\epsilon}{dV} = -V \frac{8\pi p_F^2}{h^3} (m^2 + p_F^2)^{1/2} \frac{dp_F}{dV} - \epsilon = \left(\frac{8\pi}{3h^3} \right) p_F^3 (m^2 + p_F^2)^{1/2} - \epsilon, \quad (2.17)$$

e como para um gás altamente relativístico $p_F \gg m$, então

$$\epsilon \approx \frac{2\pi p_F^4}{h^3}, \quad P \approx \frac{1}{3}\epsilon. \quad (2.18)$$

A 2.18 é dita forma politrópica da equação de estado, e pode ser escrita genericamente como

$$P = K\rho^\Gamma, \quad (2.19)$$

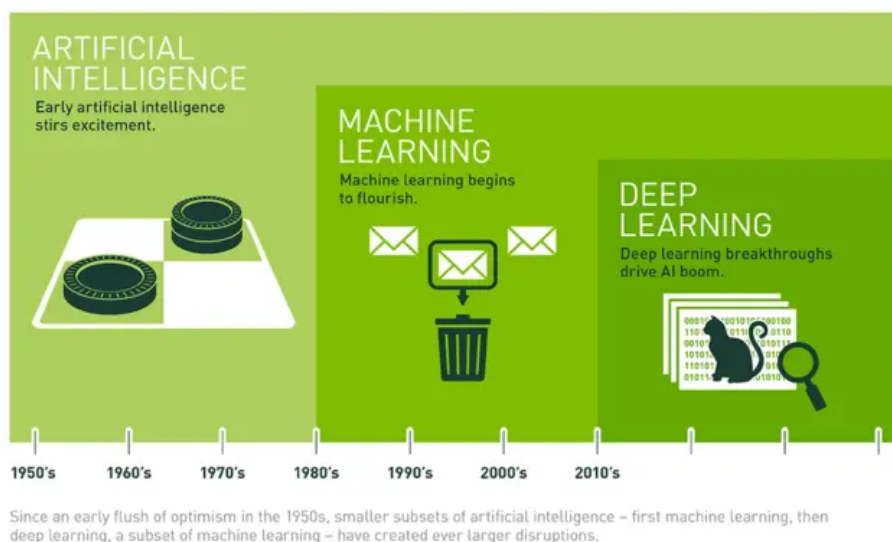
sendo K uma constante de proporcionalidade, Γ o índice adiabático e ρ a densidade de massa, que através da primeira lei da termodinâmica se relaciona com a densidade de energia.

A forma politrópica é uma boa aproximação para o comportamento termodinâmico de uma estrela, sendo uma equação de estado que permite o ajuste de seus parâmetros livres de modo a reproduzir as propriedades observadas da estrela. Esta é a forma da equação de estado que será utilizada neste trabalho durante a geração de dados sintéticos para o treinamento dos modelos. A metodologia utilizada será apresentada e discutida no Capítulo 4.

3 Aprendizado de Máquina

Apesar de ser um termo que vem ganhando popularidade na última década, o aprendizado de máquina (do inglês *machine learning*), não é algo recente. Desde meados do século XX a ideia de uma máquina que conseguiria reproduzir a forma de pensar dos humanos já havia sido concebida, e em 1959 Arthur Samuel definiu o aprendizado de máquina como sendo o campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados (SIMON, 2013). Usando algoritmos que aprendem iterativamente a partir de dados, o aprendizado de máquina permite que computadores encontrem *insights* ocultos nos dados, mesmo que não tenham sido programados para tal fim. A evolução da inteligência pode ser acompanhada na linha do tempo da figura a seguir.

FIGURA 3.1 – Evolução da inteligência artificial.



Fonte: (COPELAND, 2016)

Um dos usos mais populares do aprendizado de máquina (senão o mais popular) é realizar previsões através do reconhecimento de padrões. Através de dados históricos é possível treinar a máquina para reconhecer padrões até então ocultos aos olhos humanos,

de modo a prever situações ou valores futuros. A grosso modo, usa-se informações que já existem para se obter informações ausentes (AGRAWAL *et al.*, 2020). O uso de modelos preditivos se mostra então como um grande potencial de redução de custos, riscos e melhoria de produtividade, o que explica sua crescente popularidade.

O objetivo principal deste trabalho é treinar o computador para que ele possa identificar a relação entre as equações de estado de estrelas de nêutrons e suas respectivas curvas massa-raio, isto é, a relação dada pela TOV, porém sem que haja cálculos de forma explícita. Para tal, este capítulo se dedica a apresentar os principais conceitos referentes ao aprendizado de máquina, bem como os algoritmos que serão utilizados, alguns dos processos de tratamento dos dados e as métricas de avaliação dos modelos gerados.

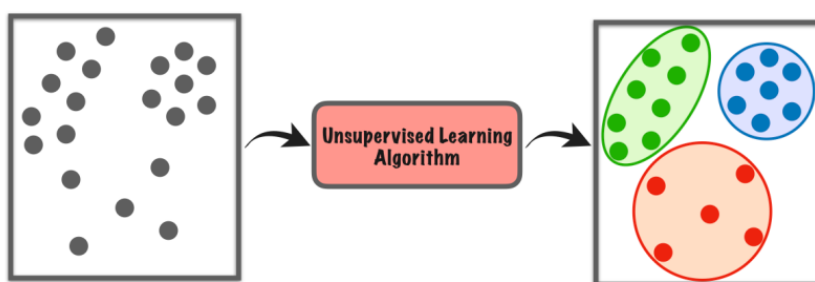
3.1 Tipos de aprendizagem

A forma de aprendizado e o tipo de algoritmo utilizado varia muito de acordo com o problema a ser resolvido e o objetivo desejado. De modo geral existem duas grandes formas de aprendizagem: a aprendizagem não-supervisionada e a aprendizagem supervisionada (existe ainda a aprendizagem por reforço, que não será explorada neste trabalho). A seguir ambas as formas serão detalhadas a fim de que fique clara a escolha para a realização deste trabalho.

3.1.1 Aprendizagem Não Supervisionada

No aprendizagem não supervisionada a máquina tem acesso aos dados históricos, mas não recebe a informação explícita sobre o que deve aprender (MICHALSKI *et al.*, 2014). A máquina tenta encontrar padrões de similaridade entre os dados buscando homogeneidade no espaço de parâmetros, resultando nos dados semelhantes entre si agrupados em *clusters*, como ilustrado na figura abaixo.

FIGURA 3.2 – Aprendizagem Não Supervisionada



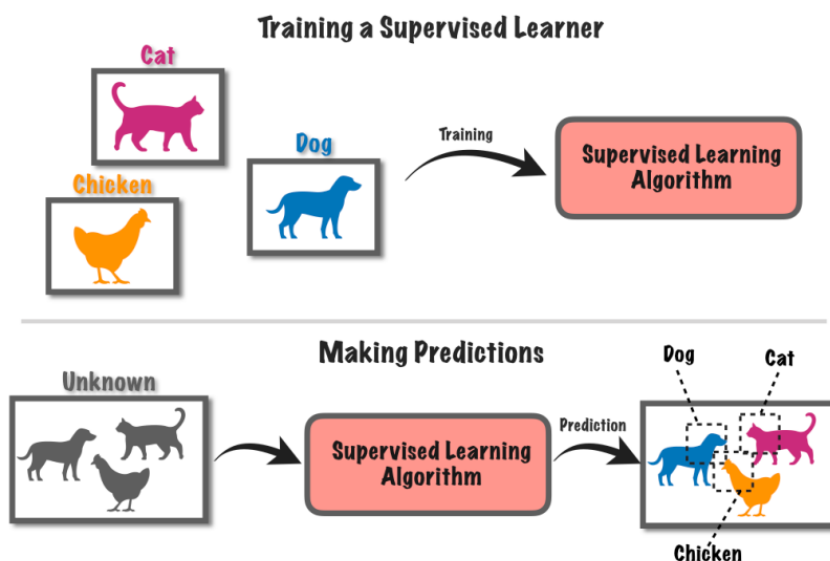
Fonte: (JEFFARES, 2018)

O principal objetivo desse tipo de aprendizagem não é necessariamente fazer uma previsão, mas sim descobrir algum tipo de conhecimento sobre os dados, como por exemplo descobrir perfis específicos de clientes com base nos seus dados de consumo. Um dos algoritmos mais utilizados para aprendizagem não supervisionada é o *K-Means*.

3.1.2 Aprendizagem Supervisionada

No aprendizagem supervisionado a máquina tem acesso não só aos dados históricos, mas também sabe exatamente o que deve aprender (MICHALSKI *et al.*, 2014). Exemplos seriam aprender o preço de uma casa, a reconhecer um animal com base em uma foto, ou se um cliente deve ou não receber crédito de um banco. Enquanto o preço de uma casa é uma variável numérica, um animal ou a liberação ou não de crédito são variáveis categóricas, e por este motivo a aprendizagem supervisionada pode ser dividida em duas: regressão e classificação. Exemplos desses tipos de aprendizagem podem ser acompanhados na imagem abaixo.

FIGURA 3.3 – Aprendizagem Supervisionada.



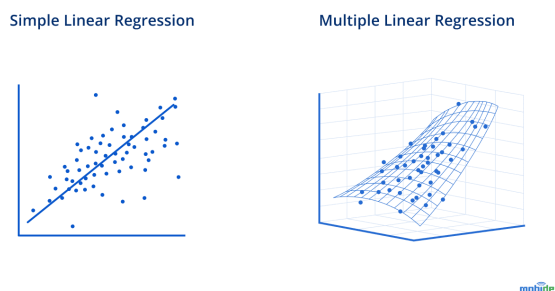
Fonte: (JEFFARES, 2018)

3.1.2.1 Regressão

Na regressão o objetivo do algoritmo é prever um valor numérico. Voltando ao exemplo da casa, um algoritmo de regressão receberia diferentes informações sobre casas que já possuem preços definidos. Ele então aprende a relação entre as características da casa e seus preços para, ao ser apresentado com novas características de uma casa que não estava nos dados históricos, ele possa entregar o preço correspondente. Algoritmos de regressão

linear são os mais utilizados para esta tarefa, mas outros tipos como árvores de decisão e *support vector machines* também podem realizar esta atividade. Outra característica da regressão linear é que ela pode ser simples, com apenas uma variável preditora, ou múltipla, quando mais de uma variável é utilizada na predição, conforme ilustrado na figura abaixo.

FIGURA 3.4 – Regressão Linear Simples e Múltipla.

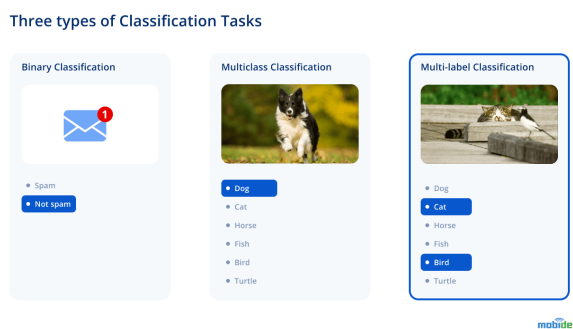


Fonte: (MOLODORIA, 2022)

3.1.2.2 Classificação

Na classificação o objetivo é prever uma classe, um rótulo. Por exemplo, bancos possuem diversas informações sobre os seus clientes, bem como quais tiveram análise de crédito positiva. É possível então treinar um algoritmo de classificação para, assim que um novo cliente ingressar no banco, prever se aquele cliente tem ou não crédito aprovado. Ou então usando um banco de fotos de gatos e cachorros, treinar o algoritmo para identificar a qual classe pertence uma nova foto de um animal que não estava presente no banco de fotos do treinamento. Alguns exemplos de classificação podem ser vistos na figura à seguir.

FIGURA 3.5 – Diferentes tipos de classificação.



Fonte: (MOLODORIA, 2022)

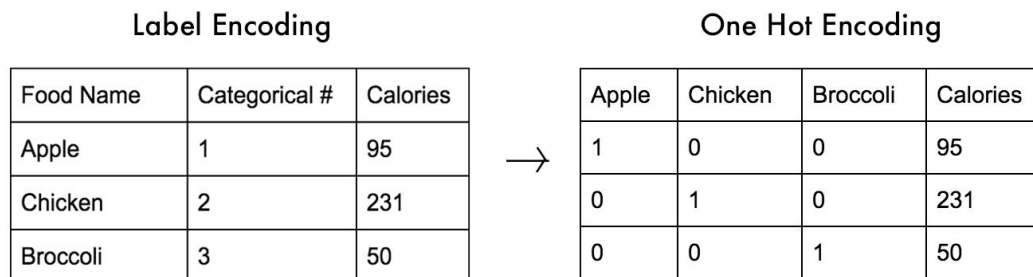
Como o objetivo deste trabalho é determinar quais equações de estado são consistentes com as regiões observacionais estabelecidas por observatórios como o NICER e o LIGO, o ideal é o uso de algoritmos de classificação. Ao final espera-se que as equações sejam classificadas em duas classes: passam ou não passam as condições impostas. Mais detalhes sobre estas condições serão discutidos no Capítulo 4.

3.2 Tratamento dos Dados

Como visto anteriormente, para que seja possível treinar um algoritmo de *machine learning* é necessário que haja dados que permitam que a máquina aprenda. Esses dados podem conter diferentes informações, que podem ser tanto variáveis numéricas (altura, peso, preço) quanto categóricas (sexo, profissão, marca), de modo que é preciso que haja uma padronização nos dados para que o aprendizado seja facilitado. Alguns algoritmos, por exemplo, fazem uso da distância euclidiana para calcular a similaridade entre os dados dentro do espaço de parâmetros, o que pode trazer complicações caso os dados estejam em escalas muito diferentes (MURPHY, 2012). O problema se agrava ainda mais quando se trata de variáveis categóricas, que não podem ser utilizadas no cálculo da distância, de modo que se faz necessário o tratamento desses dados antes de serem apresentados ao algoritmo. Na sequência serão apresentadas três técnicas de pré-processamento dos dados visando os algoritmos de classificação.

3.2.1 *Encoding*

O ideal é que variáveis categóricas sejam transformadas em valores numéricos durante o pré-processamento dos dados, e isto pode ser feito de diferentes formas. É importante, porém, levar em consideração o tipo de variável categórica, que pode ser tanto ordinal como nominal. Exemplos de variáveis ordinais seriam graus de formação (bacharel, mestre, doutor) ou cargos em uma empresa (estagiário, analista, gerente), onde existe uma ordem entre as categorias. Já variáveis nominais não possuem uma hierarquia (cor, cidade, marca). O objetivo do *encoding* é converter essas variáveis categóricas em valores mantendo a hierarquia no caso das ordinais, o que geralmente é feito com o uso de *label encoding*. Já para as variáveis nominais são usadas técnicas mais robustas, como o *one-hot encoding* (RASCHKA; MIRJALILI, 2019). A diferença entre esses dois tipos de *encoding* pode ser observada na figura à seguir.

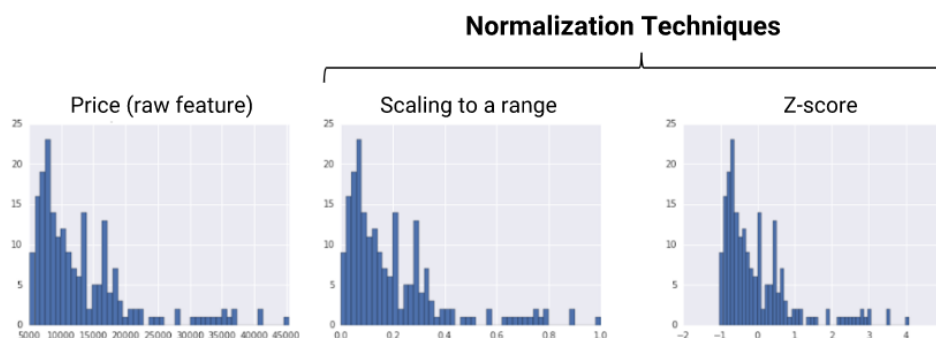
FIGURA 3.6 – *Label Encoding* e *One-Hot Encoding*.

Fonte: (DELSOLE, 2018)

3.2.2 Normalização

A normalização de dados é um dos processos a serem realizados durante o tratamento dos mesmos visando transformar as entradas para que todas as variáveis possuam a mesma escala. Este procedimento é necessário já que muitos algoritmos assumem que os dados possuem distribuição normal, o que nem sempre ocorre, e variáveis com escalas diferentes podem levar a resultados errados ou até mesmo desempenho abaixo do esperado do modelo (RASCHKA; MIRJALILI, 2019). Alguns exemplos de normalização estão na figura à seguir.

FIGURA 3.7 – Exemplos de normalização dos dados.



Fonte: (GOOGLE, 2022)

Existem várias técnicas de normalização que podem ser utilizadas durante a normalização dos dados, incluindo a normalização min-max e a normalização *Z-score*. A normalização min-max transforma os valores dos dados para um intervalo de 0 a 1, já a normalização *Z-score* transforma estes valores para uma distribuição com média zero e desvio padrão igual a um. É importante ressaltar que nem todos os algoritmos necessitam que os dados sejam normalizados. Algoritmos como árvores de decisão por exemplo

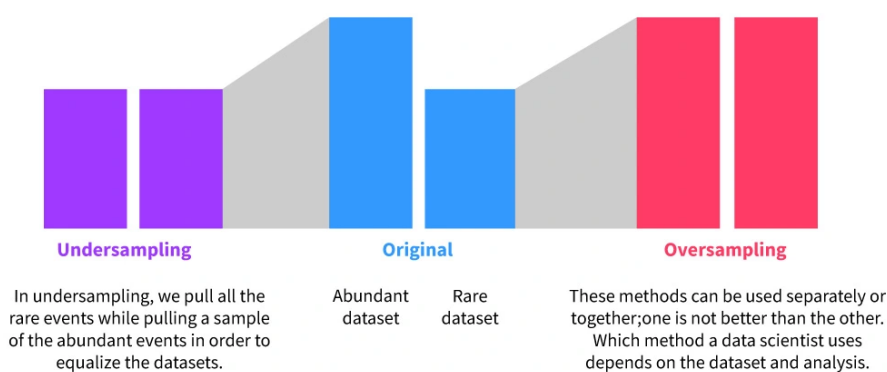
não são sensíveis às escalas dos dados, de modo que a normalização pode prejudicar o desempenho do modelo (RASCHKA; MIRJALILI, 2019).

3.2.3 Balanceamento de Classe

É comum que nos dados de treinamento de um algoritmo de classificação existam mais dados referentes a uma classe específica do que outra, por exemplo mais casos de clientes com crédito negado (classe 0) do que aprovado (classe 1). Nestes casos o algoritmo tende a favorecer a classe majoritária, prejudicando o desempenho da classificação da classe minoritária e levando a decisões equivocadas em aplicações práticas (HE; GARCIA, 2009).

Algumas técnicas podem ser utilizadas para lidar com o desbalanceamento de classe nos dados de treinamento. Uma opção seria reduzir os dados da classe majoritária para que seja igual à classe minoritária, uma técnica conhecida como *undersampling*. Porém nem sempre é possível aplicar o *undersampling*, já que em alguns casos a classe minoritária pode representar 1% dos dados de treinamento. Nestes casos pode-se aplicar o *oversampling* da classe minoritária, com técnicas como o SMOTE (*Synthetic Minority Over-sampling Technique*) (CHAWLA *et al.*, 2002), que cria exemplos sintéticos da classe minoritária com base em padrões dos dados já existentes. É necessário porém cautela em sua aplicação, já que excessos podem levar ao *overfitting*, um ajuste muito particular para os dados do treinamento de modo que o algoritmo não consiga ter um bom desempenho em dados novos.

FIGURA 3.8 – Balanceamento de Classe



Fonte: (EDX, 2022)

3.3 Algoritmos de Classificação

Os algoritmos de classificação surgiram devido à necessidade de classificar dados de diferentes categorias em diversos campos, como finanças, medicina e ciências. Um dos primeiros algoritmos de classificação foi criado em 1957 por Frank Rosenblatt, chamado perceptron. O perceptron é um modelo de rede neural capaz de aprender a classificar dados em duas categorias e é a base de muitos algoritmos de classificação utilizados até hoje (ROSENBLATT, 1958).

Outros algoritmos de classificação foram desenvolvidos com o passar dos anos, como árvores de decisão, redes neurais convolucionais, SVMs e Random Forests. Cada algoritmo tem suas próprias vantagens e desvantagens, sendo mais adequado para diferentes tipos de dados e tarefas. As árvores de decisão são frequentemente usadas em problemas de classificação com muitas características, enquanto as SVMs são comuns em tarefas de classificação binárias (HASTIE *et al.*, 2009). Hoje, com o avanço da tecnologia, novos algoritmos de classificação continuam a ser desenvolvidos e aprimorados. Esses algoritmos podem ser usados em vários campos, incluindo reconhecimento de voz, diagnóstico médico e detecção de fraudes. À medida que a quantidade de dados disponíveis continua a crescer exponencialmente, a importância e a relevância dos algoritmos de classificação também crescem, tornando-se uma parte fundamental do arsenal de ferramentas de muitos profissionais em ciência de dados e aprendizagem de máquina (ALPAYDIN, 2020).

A seguir serão apresentados cinco algoritmos de classificação mais utilizados na aprendizagem de máquina. Estes serão os algoritmos que serão utilizados neste trabalho e terão seus desempenhos comparados no Capítulo 5. Vale ressaltar que, como dito anteriormente, a proposta deste trabalho é classificar as equações de estado com base em alguma restrição imposta a elas, de modo que se faz necessário o uso de algoritmos de classificação e não regressão.

3.3.1 Regressão Logística

A regressão logística é um algoritmo amplamente utilizado em problemas de classificação binária. Diferentemente dos problemas de regressão, onde o objetivo é prever um valor numérico contínuo, a regressão logística tem como objetivo prever a probabilidade de uma amostra pertencer a uma determinada classe, geralmente representada pelos valores 0 ou 1 (JR. *et al.*, 2013). Essa probabilidade é então transformada em uma previsão definitiva utilizando uma função logística.

A regressão logística é um modelo linear e paramétrico, o que significa que a relação entre as variáveis independentes e dependentes é modelada por uma equação linear com

um conjunto de parâmetros ajustáveis. Essa equação é geralmente representada como

$$P(y = 1|x) = \frac{1}{(1 + \exp(-z))}, \quad (3.1)$$

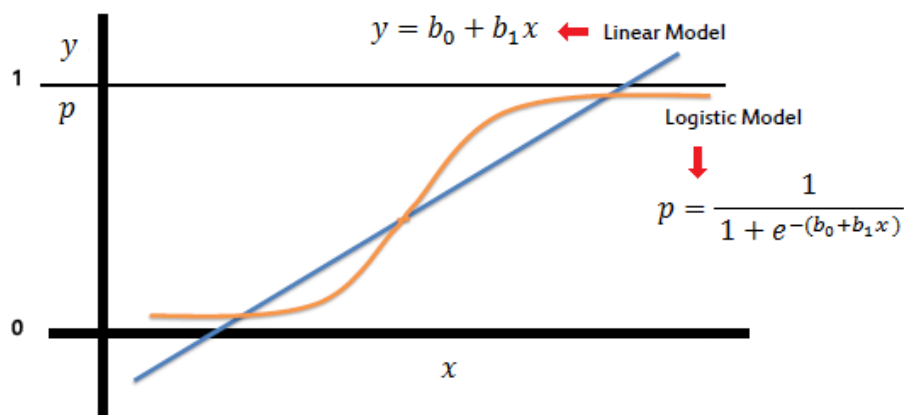
onde y é a variável dependente binária, x é o vetor de variáveis independentes, $P(y=1|x)$ é a probabilidade de y ser 1 dado x , e z é uma combinação linear das variáveis independentes com pesos ajustáveis, representada por

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n, \quad (3.2)$$

onde β_n e x_n são, respectivamente, os coeficientes de regressão e os valores das variáveis independentes.

O objetivo da regressão logística é encontrar os valores dos coeficientes β_n que melhor se ajustam aos dados de treinamento, de modo que a probabilidade de cada objeto pertencer à classe positiva seja maximizada. Uma das formas de encontrar esses coeficientes é através de funções de custo que calculam a diferença da probabilidade prevista e a probabilidade real. O ideal é que essa diferença seja minimizada, o que é feito a cada iteração através de métodos de otimização, como o gradiente descendente.

FIGURA 3.9 – Regressão Logística



Fonte: (SAYAD, 2021)

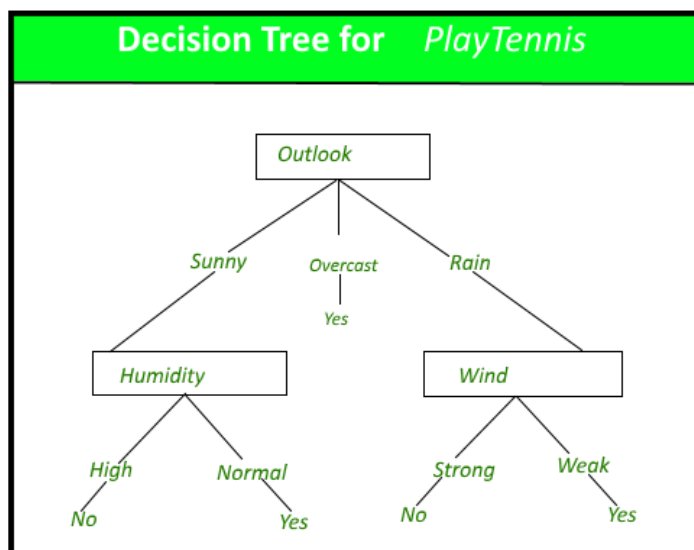
Uma vez que o modelo de regressão logística é treinado, ele pode ser usado para prever a probabilidade de uma nova amostra pertencer a classe positiva (classe 1). A previsão é feita calculando a probabilidade usando a equação da regressão logística (3.1) e, em seguida, aplicando um limiar de decisão para determinar se a amostra pertence a classe positiva ou não. Na Figura 3.9 a curva laranja representa a equação da regressão logística, enquanto a linha horizontal preta representa o limiar de decisão. Dado um valor x aplicado

na equação da regressão logística, se o valor de $P(x)$ estiver acima do limiar estabelecido a classificação atribuída será 1, e caso o contrário será 0.

3.3.2 Árvores de Decisão

As árvores de decisão são um modelo de aprendizado de máquina que permite a classificação e previsão de dados por meio de uma série de decisões binárias em um formato de árvore (QUINLAN, 1986). A partir dos dados de treinamento as árvores são construídas de modo que cada nó represente uma variável, e cada ramo represente uma decisão binária ou uma escolha entre duas opções possíveis, como visto na figura à seguir.

FIGURA 3.10 – Árvore de decisão



Fonte: (GEEKSFORGEES, 2023)

O processo de construção da árvore começa com a seleção da variável que melhor separa as classes do conjunto de dados de treinamento. Essa variável é colocada no nó raiz da árvore e divide os dados em duas subárvores, cada uma representando uma das opções possíveis. A divisão dos ramos é baseada em condições que minimizam a impureza do nó, como a entropia ou índice de Gini (BREIMAN *et al.*, 1984). Esse processo é repetido recursivamente para cada subconjunto de dados até que todas as folhas da árvore representem uma classe ou categoria. *Random Forest* e *Extra Trees* são algoritmos de aprendizado de máquina que pertencem à categoria de árvore de decisão.

3.3.3 *Extra Trees e Random Forest*

A *Random Forest* é uma técnica de aprendizado supervisionado que combina várias árvores de decisão independentes para criar um modelo de classificação. Cada árvore é construída com um subconjunto aleatório dos dados de treinamento e um subconjunto aleatório dos atributos, e usando a média das previsões de todas as árvores ela produz uma previsão final. A principal vantagem da *Random Forest* é reduzir o *overfitting* (ajuste excessivo aos dados de treinamento) e melhorar a generalização do modelo. Isso ocorre devido a aleatoriedade do subconjunto dos dados de treinamento, o que aumenta a diversidade do conjunto de árvores na floresta. Além disso, ela é menos suscetível a *outliers* e dados ruidosos, porque a média das previsões de várias árvores reduz o efeito desses dados (BREIMAN, 2001).

Já as *Extra Trees* são uma variante da *Random Forest*. A principal diferença entre elas é que as *Extra Trees* tomam decisões de forma ainda mais aleatória. Enquanto as árvores de decisão tradicionais dividem um nó com base em uma única condição (por exemplo, o valor de um atributo), as *Extra Trees* selecionam várias condições aleatórias para fazer a divisão (GEURTS *et al.*, 2006). Essa aleatoriedade adicionada também ao processo de divisão reduz a tendência de *overfitting* e melhora a generalização do modelo. A figura a seguir compara as características da *Random Forest* com as *Extra Trees*.

FIGURA 3.11 – Comparação entre *Random Forest* e *Extra Trees*

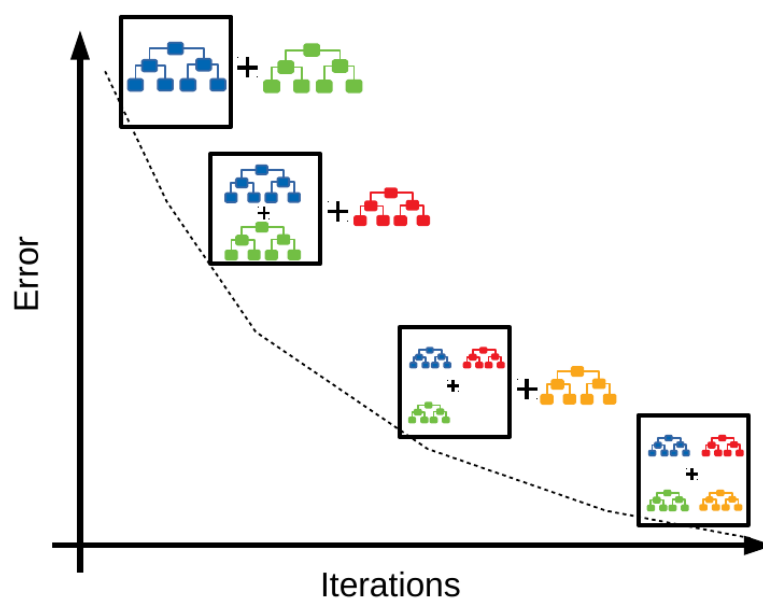
Random Forest	Extremely Randomized Trees
Samples subsets through bootstrapping	Samples the entire dataset
Nodes are split looking at the best split	Randomized node split
Medium Variance	Low Variance
It takes time to find the best node to split on	Faster since node splits are random

Fonte: (BUDU, 2020)

3.3.4 *Gradient Boosting*

Gradient Boosting é uma técnica que cria um modelo de previsão combinando várias árvores de decisão simples, chamadas de “árvores fracas”. Ao contrário do *Random Forest*, onde as árvores são construídas de forma independente, o *Gradient Boosting* constrói as árvores em série, uma após a outra, ajustando cada árvore para corrigir os erros da árvore anterior (FRIEDMAN, 2001). O princípio é similar ao da Regressão Logística, que é minimizar a função de perda, sendo que para isso ele utiliza uma abordagem de otimização iterativa. A cada iteração, uma nova árvore é adicionada ao modelo para corrigir os erros do modelo atual. A saída final é uma combinação ponderada das previsões de todas as árvores no modelo e com o erro reduzido, como exemplificado na Figura 3.12.

O Gradient Boosting tem vários hiperparâmetros que podem ser ajustados para otimizar o modelo, como o número de árvores, a profundidade das árvores, a taxa de aprendizado e a função de perda. A escolha dos hiperparâmetros pode afetar significativamente o desempenho do modelo, e pode ser otimizada usando técnicas de validação cruzada. Além disso, ele possui algumas variações como o *Light Gradient Boosting* e o *Extreme Gradient Boosting*.

FIGURA 3.12 – *Gradient Boosting*

Fonte: (SHUKLA, 2022)

3.3.4.1 *Light Gradient Boosting*

Light Gradient Boosting (LGBM) se diferencia de outras implementações de *Gradient Boosting* por sua eficiência computacional e velocidade, além de ser capaz de lidar com grandes conjuntos de dados com facilidade. O LGBM usa o conceito de *split finding* para construir árvores de decisão de forma mais eficiente do que as técnicas tradicionais de *Gradient Boosting* (KE *et al.*, 2017). O *split finding* se refere ao processo de escolher a melhor divisão possível de um conjunto de dados em dois subconjuntos em cada nó da árvore de decisão, e no LGBM a técnica utiliza uma abordagem baseada em histograma, que agrupa os valores dos atributos em “baldes”(ou *bins*) e a partir disso encontra a melhor divisão possível.

Outro ponto importante do LGBM é que ele utiliza uma técnica chamada “*leaf-wise*” para construir as árvores de decisão de forma mais eficiente. A abordagem “*leaf-wise*” constrói a árvore de decisão de forma mais profunda do que as técnicas tradicionais de *Gradient Boosting*, permitindo que o modelo alcance melhores resultados com menos nós

da árvore (KE *et al.*, 2017). Usar amostragem em linha para lidar com grandes conjuntos de dados em vez de amostragem de coluna é outro aspecto crucial do LGBM. Em vez de selecionar aleatoriamente os atributos para cada amostra de dados, a amostragem em linha seleciona aleatoriamente os dados de treinamento para cada árvore de decisão.

3.3.4.2 *Extreme Gradient Boosting*

O *Extreme Gradient Boosting* (XGBoost) é uma versão aprimorada do algoritmo de *Gradient Boosting* e se destaca por sua eficiência e desempenho em grandes conjuntos de dados. Como o *Gradiente Boosting* e o LGBM, o XGBoost opera de maneira semelhante, mas uma de suas características distintivas é que ele emprega um mecanismo de *pruning* (poda) para evitar o *overfitting* (CHEN; GUESTRIN, 2016), que ocorre quando o modelo se ajusta demais aos dados de treinamento e não generaliza bem para novos dados. O procedimento de “poda” apara os ramos desnecessários das árvores de decisão para evitar que o modelo superajuste o conjunto de treinamento. A regularização é outra técnica usada pelo XGBoost para diminuir o *overfitting*. Modelos com muitas características (alta complexidade) ou com características altamente correlacionadas são penalizados pela regularização. Além de evitar o *overfitting*, a regularização ajuda a manter o modelo simples. O XGBoost também suporta a paralelização do treinamento do modelo, o que significa que ele pode treinar modelos mais rapidamente em grandes conjuntos de dados (CHEN; GUESTRIN, 2016).

3.3.5 *Support Vector Machines*

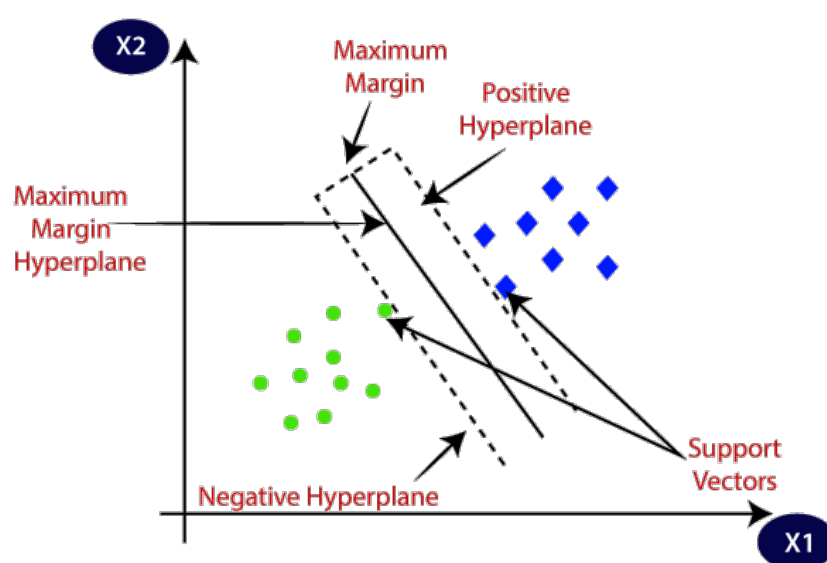
Os *Support Vector Machines* são algoritmos extremamente populares usados tanto para regressão como para classificação. O objetivo do SVM é encontrar um hiperplano que possa separar as classes de dados em um espaço de alta dimensão. O hiperplano é definido como um limite de decisão entre as classes, onde a distância entre o hiperplano e os pontos de dados mais próximos, chamados vetores de suporte, é maximizada. Basicamente, se no espaço de parâmetros atual não for possível encontrar um plano de separação entre as classes, o SVM eleva a dimensão desse espaço até que a divisão seja possível (CRISTIANINI; SHAW-TAYLOR, 2000).

Para elevar os dados a uma dimensão superior o SVM utiliza uma função, também chamada de *kernel*. O *kernel* é uma função matemática que calcula a semelhança entre dois pontos de dados no espaço original e projeta esses pontos para um espaço de dimensão superior. Existem vários tipos de *kernels* que podem ser usados com o SVM, incluindo o linear, polinomial, RBF (*Radial Basis Function*) e sigmoidal (CRISTIANINI; SHAW-TAYLOR, 2000). O *kernel* mais apropriado depende do conjunto de dados e do problema

em questão.

Uma vez que os dados são projetados em um espaço de dimensão superior, o SVM encontra o hiperplano que maximiza a distância entre os vetores de suporte e o hiperplano. Essa distância é chamada de margem e representa a confiança do modelo na classificação dos dados. Se os dados não podem ser separados por um hiperplano linear, o SVM usa um truque chamado *soft margin*. Nesse caso, o SVM permite que alguns pontos de dados sejam classificados incorretamente, a fim de encontrar um hiperplano que minimize a soma das margens e o número de erros de classificação.

FIGURA 3.13 – *Support Vector Machine*



Fonte: (JAVATPOINT, 2020)

3.4 Métricas de Avaliação dos Modelos

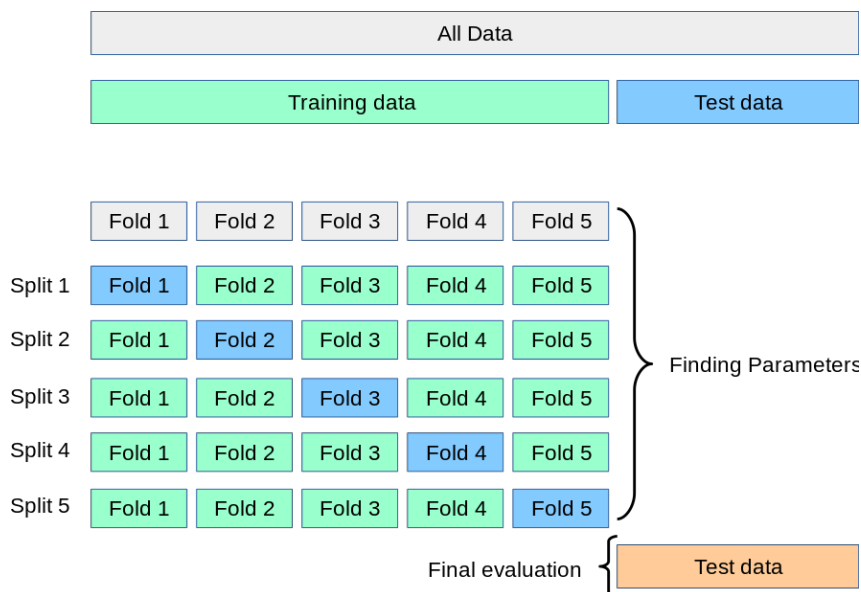
Independente de qual algoritmo seja utilizado para o treinamento de um modelo de aprendizado de máquina, ele precisa ser avaliado a fim de garantir que as previsões sejam confiáveis e que erros não sejam cometidos durante sua aplicação. As métricas de avaliação são utilizadas para medir a qualidade de um modelo de *machine learning* e determinar quão bem ele está se ajustando aos dados. Essas métricas são importantes para ajudar a selecionar o melhor modelo para um determinado conjunto de dados e para entender o desempenho do modelo. A seguir serão apresentadas doze métricas de avaliação que serão utilizadas nos modelos treinados para este trabalho, bem como a função delas durante a análise dos resultados. Como cada métrica mede um aspecto do modelo, é importante analisá-las em conjunto, e não de modo isolado.

3.4.1 *Cross Validation*

Cross Validation (validação cruzada) é uma técnica comum usada para avaliar a capacidade de generalização de um modelo e ajudar a evitar *overfitting* (VARMA; SIMON, 2006). O principal propósito da validação cruzada é avaliar como o modelo treinado performará em dados que nunca foram vistos antes, uma vez que um modelo pode se ajustar demais aos dados de treinamento e não conseguir generalizar bem para novos dados.

Para realizar a validação cruzada, divide-se o conjunto de dados em dois subconjuntos: um subconjunto é usado para treinar o modelo (conjunto de treinamento) e o outro subconjunto é usado para testar o modelo (conjunto de teste), conforme Figura 3.14. O processo é repetido várias vezes, variando os subconjuntos utilizados para treinamento e teste. Isso significa que o modelo é treinado em diferentes conjuntos de treinamento e testado em diferentes conjuntos de teste. Ao final, as métricas de desempenho são computadas e avaliadas para determinar a qualidade geral do modelo.

FIGURA 3.14 – *Cross Validation*



Fonte: (SCIKIT-LEARN, 2011)

Existem várias técnicas de validação cruzada disponíveis, sendo as mais comuns a validação cruzada *k-fold* e *leave-one-out*. Na validação cruzada *k-fold*, o conjunto de dados é dividido em k subconjuntos (ou *folds*), e o modelo é treinado k vezes, cada vez usando um subconjunto diferente para teste e os outros $k-1$ subconjuntos para treinamento. Na validação cruzada *leave-one-out*, cada amostra é usada uma vez como conjunto de teste e as outras são usadas para treinamento.

3.4.2 *Confusion Matrix*

A *Confusion Matrix* (matriz de confusão) é uma das métricas mais utilizadas para avaliar a qualidade de um modelo de classificação. Ela mostra a distribuição das previsões do modelo em relação aos dados reais de teste, permitindo uma análise detalhada do desempenho geral do modelo, sendo uma tabela que mostra o número de previsões corretas e incorretas feitas pelo modelo em cada classe (PROVOST; FAWCETT, 2001). Usualmente as linhas representam as classes reais e as colunas representam as classes previstas pelo modelo, conforme ilustrado na Figura 3.15. A tabela é dividida em quatro quadrantes, que correspondem às quatro possíveis combinações de previsões corretas e incorretas:

- a) Verdadeiro positivo (TP): o modelo previu corretamente que a amostra pertence a uma classe específica;
- b) Falso positivo (FP): o modelo previu erroneamente que a amostra pertence a uma classe específica, quando na verdade ela pertence a outra;
- c) Falso negativo (FN): o modelo previu erroneamente que a amostra não pertence a uma classe específica, quando na verdade ela pertence;
- d) Verdadeiro negativo (TN): o modelo previu corretamente que a amostra não pertence a uma classe específica.

FIGURA 3.15 – *Confusion Matrix*

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Fonte: (DRAELOS, 2019)

As matrizes de confusão são muito úteis para avaliar a qualidade do modelo de classificação e para identificar quais classes estão sendo mais bem ou mal classificadas pelo modelo. Elas também ajudam a identificar possíveis problemas de viés no modelo e a ajustar seus hiperparâmetros (os parâmetros específicos de cada algoritmo de treinamento) para melhorar seu desempenho. Outro ponto importante é que a partir dela é possível

calcular várias outras métricas de desempenho do modelo, como a acurácia, *precision*, *recall* e *F1-Score*.

3.4.3 Acurácia

A acurácia é uma métrica comum em modelos de *machine learning* de classificação que indica a proporção de amostras classificadas corretamente pelo modelo. Essa métrica é uma medida geral do desempenho do modelo e indica o quão preciso ele é na classificação das amostras (POWERS, 2020).

A acurácia é calculada dividindo o número de amostras classificadas corretamente pelo modelo pelo número total de amostras:

$$\text{Acurácia} = \frac{\text{TP} + \text{TN}}{\text{Total de Amostras}}. \quad (3.3)$$

Se um modelo de classificação corretamente classificou 80 amostras de um total de 100 amostras, a acurácia desse modelo seria de 80%. Em outras palavras, o modelo acertou 80% das suas previsões.

3.4.4 Precision

Precision (precisão) é uma métrica de avaliação que mede a proporção dos dados classificados como positivos pelo modelo que realmente são positivos (POWERS, 2020). Ela é uma métrica muito importante pois, em alguns casos, é mais importante minimizar os falsos positivos do que os falsos negativos. Por exemplo, em um modelo que classifica transações bancárias como fraudulentas ou não, é preferível ter uma precisão alta (ou seja, classificar poucas transações legítimas como fraudulentas) mesmo que isso signifique ter uma taxa maior de falsos negativos (ou seja, deixar passar algumas transações fraudulentas). A precisão é calculada pela fórmula

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \quad (3.4)$$

3.4.5 Recall

O *Recall* (revocação) mede a proporção dos dados positivos que foram corretamente identificados pelo modelo (POWERS, 2020). Ele é importante por motivos similares ao *precision*, já que em alguns casos é mais importante minimizar os falsos negativos do que os falsos positivos. Um exemplo seria um modelo que classifica pacientes como doentes ou não, onde é preferível ter um *Recall* alto (ou seja, identificar a maioria dos pacientes doen-

tes) mesmo que isso signifique ter uma taxa maior de falsos positivos (ou seja, identificar alguns pacientes saudáveis como doentes). A revocação é calculada pela fórmula

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \quad (3.5)$$

3.4.6 F1-Score

O *F1-Score* é uma métrica de avaliação que combina o *precision* e o *recall* em uma única medida resumida. Ele acaba sendo de extrema importância para modelos que exigem tanto um *precision* quanto um *recall* elevados (POWERS, 2020). O *F1-Score* é então a maneira de resumir o desempenho do modelo em ambas as métricas. Seu cálculo é feito pela seguinte fórmula

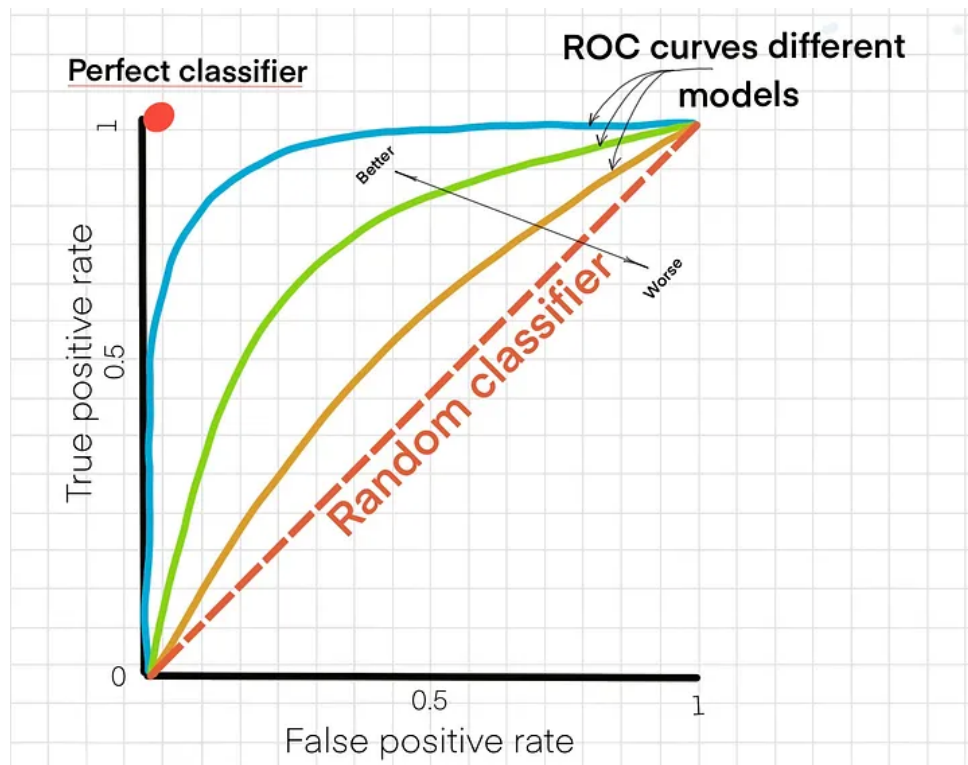
$$\text{F1-Score} = \frac{2 \cdot (\text{Precision} \cdot \text{Recall})}{\text{Precision} + \text{Recall}}. \quad (3.6)$$

O *F1-Score* é uma média harmônica da precisão e revocação, dando igual peso a ambas as métricas. Como resultado, o *F1-Score* é uma medida útil para avaliar o desempenho de um modelo quando há um equilíbrio entre a precisão e a revocação desejada.

3.4.7 Curva ROC

A curva ROC (*Receiver Operating Characteristic*) é outra técnica de avaliação de modelos de classificação binária. A curva ROC é um gráfico que representa a relação entre a taxa de verdadeiros positivos (*true positive rate*, ou TPR) e a taxa de falsos positivos (*false positive rate*, ou FPR) para diferentes valores de *threshold* (limiar de decisão) do modelo (HANLEY; MCNEIL, 1982). Ela é útil porque permite avaliar a capacidade do modelo de distinguir entre dados positivos e negativos em diferentes níveis de sensibilidade e especificidade. Sensibilidade é a capacidade do modelo de identificar corretamente os dados positivos, enquanto especificidade é a capacidade do modelo de identificar corretamente os dados negativos.

FIGURA 3.16 – Curva ROC



Fonte: (GUSAROVA, 2022)

A curva ROC é construída traçando-se a taxa de verdadeiros positivos (TPR) no eixo Y e a taxa de falsos positivos (FPR) no eixo X para diferentes valores de *threshold*. Cada ponto na curva ROC representa uma combinação de sensibilidade e especificidade para um determinado *threshold*. Um modelo ideal teria uma taxa de verdadeiros positivos (TPR) igual a 1 e uma taxa de falsos positivos (FPR) igual a 0, o que seria representado pelo ponto no canto superior esquerdo da Figura 3.16. Um modelo aleatório teria uma curva ROC que se aproximaria de uma linha reta diagonal do canto inferior esquerdo ao superior direito (FAWCETT, 2006).

A área sob a curva ROC (AUC - *Area Under Curve*) é uma medida de desempenho geral do modelo, e quanto maior o valor da AUC, melhor o desempenho do modelo. A AUC varia de 0 a 1, sendo que um valor de 0,5 indica um modelo que não é melhor do que um modelo aleatório, enquanto um valor de 1 indica um modelo perfeito. A AUC é amplamente utilizada quando se tratam de modelos de classificação multiclasse, isto é, quando existem mais de duas classes possíveis.

3.4.8 Calibration Curve

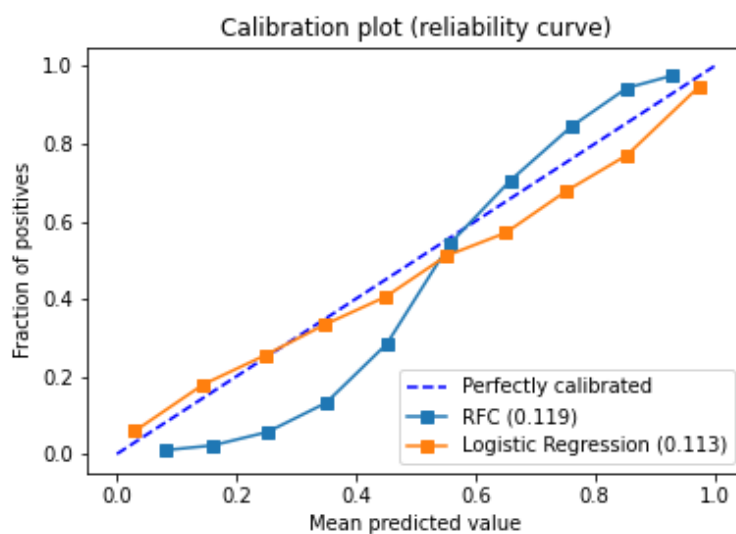
A *Calibration Curve* (curva de calibração) visa avaliar o quão bem as probabilidades de classificação do modelo estão calibradas. A ideia por trás da curva de calibração é verificar se a probabilidade de classificação prevista pelo modelo é consistente com a probabilidade real de que uma determinada instância pertença a uma determinada classe (GUO *et al.*, 2017).

Para um modelo bem calibrado, o que se espera é que para os dados com uma probabilidade prevista de 70% de pertencer a uma classe (0 ou 1, por exemplo), aproximadamente 70% realmente pertençam à classe prevista. Se a calibração do modelo estiver correta, a curva de calibração será próxima de uma diagonal ($y = x$), como visto na Figura 3.17. Isto significa que a probabilidade prevista é muito semelhante à probabilidade real.

A curva de calibração é útil para avaliar modelos que produzem estimativas de probabilidade, como regressão logística, árvores de decisão e florestas aleatórias. Ela ajuda a identificar casos em que o modelo é tendencioso na previsão de probabilidades para diferentes classes e, portanto, pode ser necessário ajustá-lo.

Para gerar a curva de calibração, o modelo divide as instâncias em um número fixo de *bins* (intervalos), com base na probabilidade prevista, e para cada *bin* calcula a proporção de instâncias verdadeiramente positivas em relação ao total de instâncias desse *bin*. Em seguida, a curva de calibração é gerada plotando essas proporções em relação às probabilidades previstas médias para cada *bin*.

FIGURA 3.17 – Calibration Curve



Fonte: (CINELLI, 2020)

3.4.9 Gini

O índice de Gini é uma métrica comumente usada em modelos de classificação para avaliar a qualidade de uma classificação binária, mede o quão bem o modelo separa as duas classes e é calculado a partir da curva ROC (SEO *et al.*, 2020). O índice de Gini é calculado como a área entre a curva ROC e a linha diagonal, que representa um classificador aleatório

$$\text{Gini} = 2 \cdot \text{AUC} - 1. \quad (3.7)$$

O valor do índice de Gini varia de 0 a 1, onde 0 indica um classificador aleatório e 1 indica um classificador perfeito. Ele se torna útil para avaliar modelos de classificação pois é uma medida que não depende da proporção de instâncias em cada classe. Ele também é menos sensível a desequilíbrios de classe do que outras medidas de avaliação, como a acurácia. Isso o torna uma métrica mais robusta para avaliar a qualidade de modelos de classificação em conjuntos de dados com classes desbalanceadas (POWERS, 2020).

3.4.10 *Kappa*

O índice *Kappa* (também conhecido como coeficiente *Kappa* ou coeficiente de concordância de Cohen) é uma métrica usada para avaliar a concordância entre as previsões do modelo e os rótulos verdadeiros das classes. O índice *Kappa* é especialmente útil em situações em que os dados estão desbalanceados e a acurácia não é uma métrica confiável para avaliar o modelo (POWERS, 2020).

O índice *Kappa* é calculado comparando a concordância observada entre as previsões do modelo e os rótulos verdadeiros com a concordância esperada entre as previsões e os rótulos verdadeiros se as previsões fossem aleatórias. Ele é calculado pela fórmula

$$Kappa = \frac{\text{Acurácia Observada} - \text{Acurácia Esperada}}{\text{Acurácia Esperada}}, \quad (3.8)$$

onde a acurácia observada é a proporção de instâncias corretamente classificadas e a acurácia esperada é a proporção que seria corretamente classificada se as previsões fossem aleatórias.

O valor do índice *Kappa* varia de -1 a 1, onde -1 indica uma discordância total, 0 indica concordância aleatória e 1 indica concordância perfeita (POWERS, 2020). Um valor de *Kappa* de 0,5 ou mais é geralmente considerado como um indicador de uma concordância boa entre as previsões do modelo e os rótulos verdadeiros das classes.

3.4.11 MCC

O índice MCC (coeficiente de correlação de Matthews) mede a qualidade geral da classificação de um modelo e leva em conta as quatro possibilidades de classificação: verdadeiro positivo (TP), falso positivo (FP), verdadeiro negativo (TN) e falso negativo (FN) (POWERS, 2020). O índice varia de -1 a 1, onde -1 indica uma classificação totalmente incorreta, 0 indica uma classificação aleatória e 1 indica uma classificação perfeita. Valores positivos indicam uma classificação correta, enquanto valores negativos indicam uma classificação incorreta. O cálculo do índice MCC é feito pela seguinte fórmula

$$\text{MCC} = \frac{\text{TP.TN-FP.FN}}{\sqrt{(\text{TP+FP}).(\text{TP+FN}) * (\text{TN+FP}) * (\text{TN+FN})}}, \quad (3.9)$$

O índice MCC é útil para avaliar modelos de classificação binária em situações de dados desequilibrados, em que a acurácia pode ser enganosa. Além disso, o índice MCC pode ser usado para comparar diferentes modelos de classificação binária.

3.4.12 KS

O índice KS (Kolmogorov-Smirnov) mede a diferença máxima entre as curvas ROC do modelo e a curva ROC ideal, que representa a melhor classificação possível (POWERS, 2020). Ele é calculado a partir das probabilidades de previsão do modelo para a classe positiva e negativa. O primeiro passo é ordenar as instâncias de acordo com as probabilidades de previsão da classe positiva. Em seguida, é calculada a curva acumulada da distribuição de probabilidade das instâncias verdadeiras positivas e falsas positivas em função das probabilidades de previsão da classe positiva. A diferença máxima entre as curvas acumuladas é o índice KS.

O índice KS varia de 0 a 1, onde valores mais altos indicam uma melhor classificação do modelo (POWERS, 2020). Geralmente, um valor de KS acima de 0,6 é considerado bom, enquanto um valor acima de 0,8 é considerado excelente.

4 Metodologia

O objetivo primordial deste trabalho é criar um modelo que saiba identificar quais equações de estado geram curvas M-R que passam por determinadas regiões, mais especificamente as regiões obtidas pelos observatórios NICER (MILLER *et al.*, 2019; RILEY *et al.*, 2019; MILLER *et al.*, 2021; RILEY *et al.*, 2021) e LIGO (ABBOTT *et al.*, 2018). Para que isto fosse possível, o trabalho se estruturou em 5 pilares principais:

- a) geração dos dados sintéticos: gerar e integrar equações de estado para obter um conjunto de curvas M-R;
- b) etiquetagem dos dados: realizar a verificação de quais curvas passam pelas regiões de interesse e preparar os dados para serem apresentados aos modelos;
- c) divisão dos dados: dividir os dados para que sejam usados no treinamento e na avaliação;
- d) treinamento dos modelos: treinar os modelos com os dados de treino;
- e) avaliação dos modelos: avaliar os modelos com os dados de teste a partir das métricas apropriadas.

Cada um destes pilares é de suma importância para o objetivo final. Sendo assim, a seguir cada um deles será explorado de forma mais detalhada, sendo apresentados os procedimentos realizados em cada etapa e sua importância dentro do contexto do trabalho. Outro ponto de grande importância é a plataforma onde os dados foram gerados, processados e os modelos treinados. Toda a parte computacional deste trabalho foi realizada na plataforma em *cloud* disponibilizada pela *Power of Data*, empresa cadastrada no CNPJ sob o nº 18.216.263/0001-06, com endereço na cidade de Santana do Parnaíba, Estado de São Paulo, na Avenida Yojiro Takaoka, nº 4384, Sala 701, CJ 5654, CEP 06542-001, sob um termo de cooperação firmado com o Instituto Tecnológico de Aeronáutica. O *cluster* disponibilizado possui as seguintes configurações:

- **Sistema operacional:** Linux;

- **Versão do Sistema Operacional:** Debian 10 com *kernel* 5.10.0;
- **Data de Lançamento da Versão:** 28 de julho de 2022;
- **Arquitetura de Hardware:** x86-64 (64 *bits*);
- **Memória RAM Disponível:** : 252 GB

4.1 Geração dos Dados Sintéticos

Para que os modelos possam ser treinados de forma eficiente é necessário que haja uma grande quantidade de dados disponíveis. Tendo isto em vista, é preciso esclarecer o que se deseja que os modelos aprendam para então definir quais dados iremos fornecer. Recapitulando, queremos que os modelos aprendam quais equações de estado representam curvas M-R que passam pelas regiões observacionais de interesse, porém sem que seja necessário aplicar a TOV para realizar essa verificação. Tem-se então as duas variáveis necessárias para realizar o treinamento dos modelos: os dados que serão apresentados aos modelos (as equações de estado) e o que deseja-se aprender (quais curvas M-R passam nas regiões observacionais). A Figura 4.1 mostra a relação entre essas duas variáveis: temos a equação de estado que, após passar pela TOV, retorna uma curva M-R. Pode-se então verificar se a curva passa ou não pelas regiões observacionais desejadas, representadas em verde e azul.

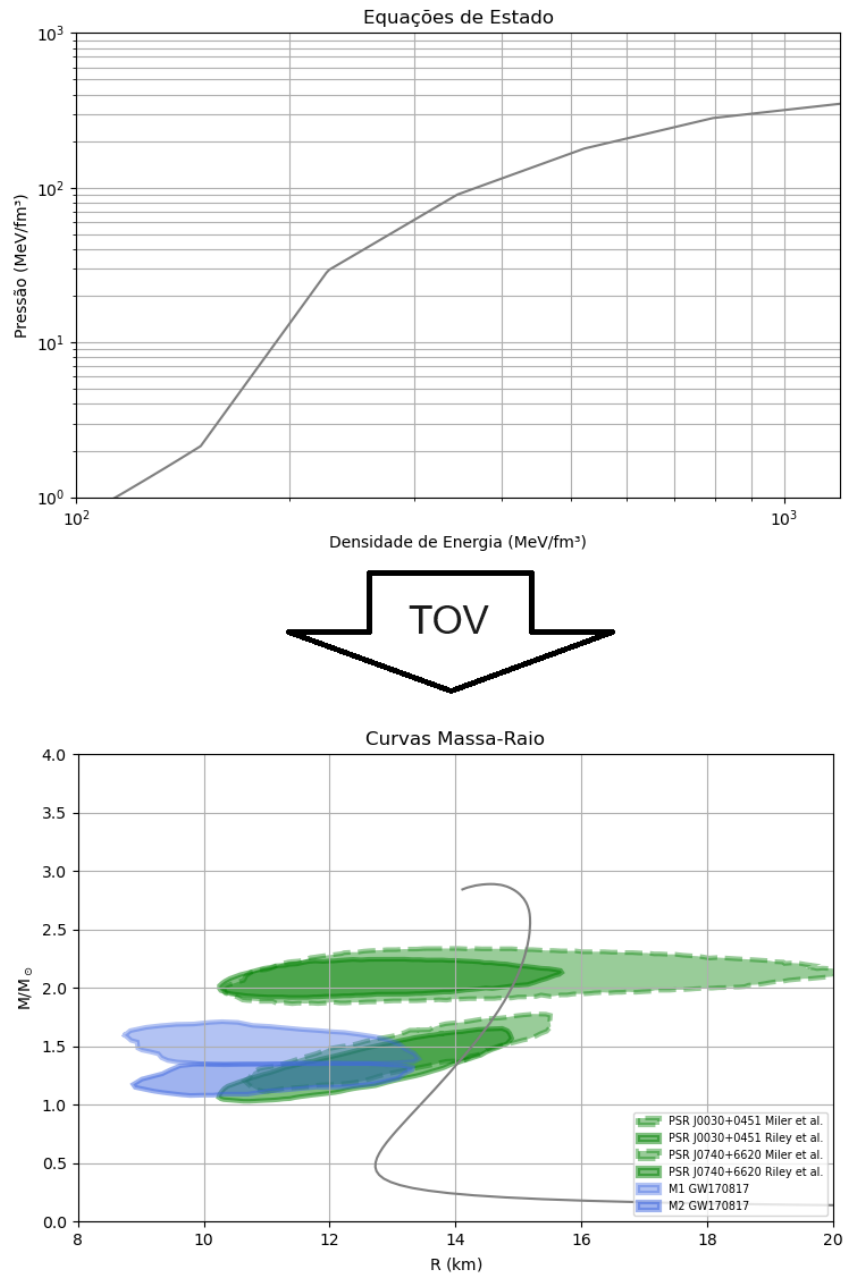
Tendo esclarecido quais serão os dados utilizados no treinamento, o próximo passo é gerar esses dados. Como dito anteriormente, é necessário que haja uma grande quantidade de equações para serem apresentadas aos modelos para que o aprendizado ocorra de forma eficiente. A geração das equações de estado segue o mesmo modelo visto no trabalho apresentado por Fujimoto (FUJIMOTO *et al.*, 2021), onde elas serão geradas de forma aleatória seguindo algumas restrições físicas:

- a) p deve ser uma função monótona e crescente em ϵ ;
- b) a velocidade do som, c_s , deve ser menor do que a velocidade da luz, sendo que:

$$c_s^2 = \frac{\partial p}{\partial \epsilon}. \quad (4.1)$$

Assim sendo, a velocidade do som se mostra uma boa opção como o parâmetro para aplicar a aleatoriedade.

FIGURA 4.1 – Equação de Estado (esquerda) e sua respectiva curva M-R (direita)



Fonte: Elaborado pelo autor.

Outro fator importante é que as equações de estado geradas devem se conectar com uma equação de estado empírica para a região de baixa densidade. A escolhida foi a SLy4 (DOUCHIN; HAENSEL, 2001) por ser uma boa equação de estado para a descrição de regiões mais próximas a superfície da estrela, e foi aplicada até a densidade de energia de saturação da matéria nuclear ϵ_0 . Para densidades de energia superiores a ϵ_0 foi aplicada parametrização politrópica por partes, sendo o range de densidade de energia, $[\epsilon_0, 8\epsilon_0]$, dividido em 5 segmentos com 6 pontos de energia ϵ_i separados igualmente na escala logarítmica, como pode ser visto na tabela abaixo.

TABELA 4.1 – Faixas de densidade de energia

Pontos	Densidade de energia (MeV/fm ³)
ϵ_0	150
ϵ_1	227
ϵ_2	345
ϵ_3	522
ϵ_4	792
ϵ_5	1200

Fonte: Elaborado pelo autor.

O parâmetro utilizado na aleatoriedade é a velocidade do som média no i -ésimo segmento, $c_{s,i}^2 \equiv \langle c_s^2 \rangle = \langle \partial p / \partial \epsilon \rangle$, ($i=1, \dots, 5$), sendo que a pressão correspondente a estes pontos é obtida através da relação:

$$p_i = p_{i-1} + c_{s,i}^2 (\epsilon_i - \epsilon_{i-1}), \quad (4.2)$$

e p_0 é determinada pela SLy4, sendo $p_0 = p(\epsilon_0)$. A interpolação politrópica foi feita por $p = k_i \epsilon^{\gamma_i}$, onde:

$$\gamma_i = \frac{\ln(p_i/p_{i-1})}{\ln(\epsilon_i/\epsilon_{i-1})}, \quad (4.3)$$

e

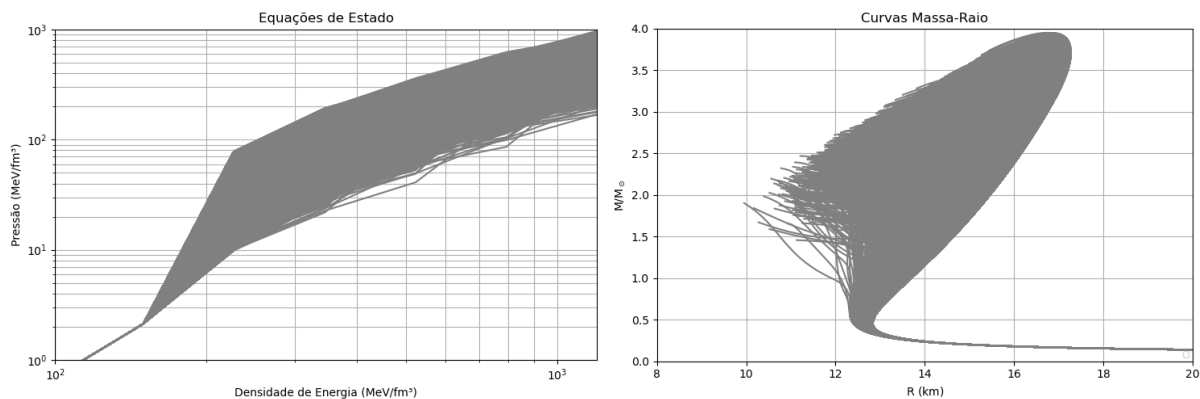
$$k_i = \frac{p_i}{\epsilon_i^{\gamma_i}}. \quad (4.4)$$

Para gerar as equações foram atribuídas de forma aleatória valores para a velocidade do som em cada um dos segmentos, seguindo uma distribuição uniforme $0,01 \leq c_{s,i}^2 \leq 0,99$, sendo o limite superior para manter a causalidade ($c_s^2 < c^2 = 1$), e o limite inferior para manter a estabilidade termodinâmica ($\partial p / \partial \epsilon \geq 0$). Seguindo esses passos foram geradas 10 mil equações de estado utilizando o código apresentado no Apêndice A, possui a implementação de todos os passos citados até aqui, como a aplicação da SLy4, a definição das faixas de energia, a escolha aleatória para as velocidades do som, o cálculo das pressões de cada segmento e a interpolação politrópica. O mesmo código possui também a função obter as curvas massa-raio de cada equação de estado através da TOV (conforme Equação 2.11). Para esta integração foi utilizado o pacote em *python* chamado *tovsolver* e sua classe *TOVSolver*, que além da equação de estado, recebe 3 parâmetros: valores inicial e final para a densidade central, onde foram utilizados $dc_i = 1,5 \cdot 10^{-8}$ MeV/fm³ e $dc_f = 1200$ MeV/fm³, e um valor para dr , sendo utilizado $dr = 100$ m. Todo este processo de geração das equações de estado e obtenção das curvas M-R levou cerca 6 horas na plataforma disponibilizada pela *Power of Data*.

É importante ressaltar que gerando a velocidade de som de forma aleatória se exclui

qualquer tipo de viés nas equações de estado, permitindo o surgimento curvas M-R não condizentes com dados observacionais, porém isto garante que os modelos possam aprender sem nenhum tipo de tendência específica, sem que aprendam mais sobre uns tipos de curvas do que outras. As equações de estado geradas e suas respectivas curvas M-R obtidas através da TOV podem ser vistas na figura à seguir.

FIGURA 4.2 – 10 mil Equações de Estado (esquerda) e Curvas M-R (direita) geradas para o treinamento dos modelos



Fonte: Elaborado pelo autor.

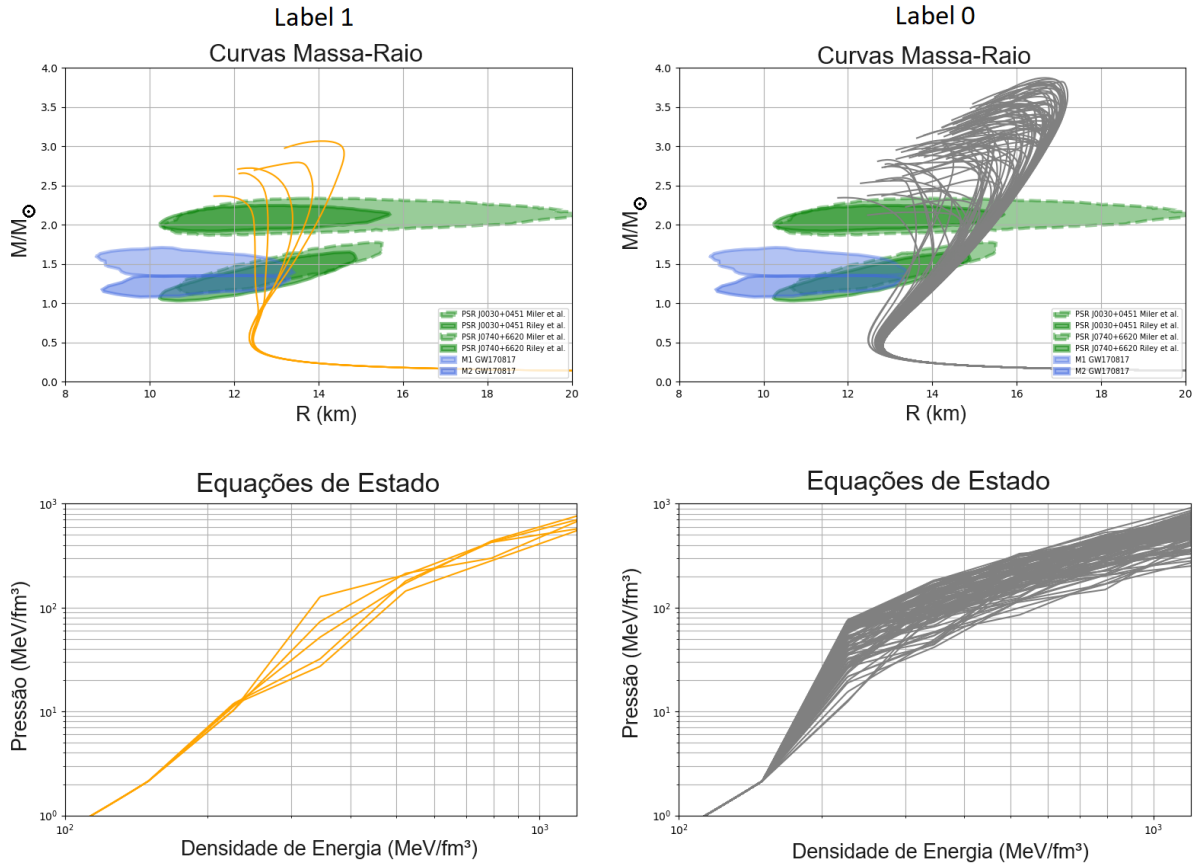
Tendo as equações de estado e as curvas M-R, é preciso verificar quais equações geraram curvas que passam pelas regiões de interesse, ou seja, atribuir um etiqueta “passa” ou “não passa” para cada uma das equações, procedimento conhecido como *labeling*, ou etiquetagem dos dados. Também é necessário definir quais informações das equações de estado serão utilizadas para o treinamento dos modelos.

4.2 Etiquetagem e Preparação dos Dados

As equações de estado precisam ser marcadas como aquelas que geram curvas M-R que passam pelas regiões de interesse e as que não passam. Para isto é necessário verificar quais curvas passam por estas regiões e atribuir um *label*, ou etiqueta, na equação de estado correspondente. O *label* escolhido foi 1 para as equações que passam pelas regiões e 0 para as que não passam, sendo que foram analisadas três situações distintas: as equação que passam pela região do NICER, as que passam pela região do LIGO, e as que passam simultaneamente em ambas as regiões. Os modelos foram treinados para identificar os três casos, de modo que há três versões para cada modelo. Na Figura 4.3 temos um exemplo deste processo. Do lado esquerdo temos as equações de estados e as respectivas curvas M-R que passam simultaneamente em todas as regiões observacionais, marcadas em azul e verde. Por terem passado na restrição imposta, elas receberam o *label* 1 e foram marcadas de laranja para facilitar sua observação nos gráficos. Do lado direito temos as

equações e curvas M-R que não cumpriram o requisito de passar simultaneamente em todas as regiões, de modo que receberam o *label* 0 e foram marcadas de cinza.

FIGURA 4.3 – Exemplo do processo de *Labeling*



Fonte: Elaborado pelo autor.

Todo o processo de etiquetagem das curvas M-R foi feito com o código apresentado no Apêndice B. Nele as curvas M-R geradas pelo código do Apêndice A são carregadas juntamente com os pontos que representam as regiões obtidas pelos observatórios NICER (MILLER *et al.*, 2019; RILEY *et al.*, 2019; MILLER *et al.*, 2021; RILEY *et al.*, 2021) e LIGO (ABBOTT *et al.*, 2018). Para cada equação é então feita a verificação de passagem em cada uma das regiões e atribuído o *label* 0 ou 1, e como. No total, cada equação possui 3 *label*: um para indicar a passagem na região do NICER, um para indicar a passagem pela região do LIGO, e um para indicar a passagem simultânea em ambas as regiões.

Com a etiqueta de cada equação definida o próximo passo é definir quais informações das equações de estado serão usadas no treinamento. As equações de estado possuem duas informações, os valores de densidade de energia e seus respectivos valores de pressão. A faixa de densidade de energia é a mesma para todas as equações, variando de 150 a 1200 MeV/fm³, porém cada equação possui uma relação $p = p(\epsilon)$ distinta, de modo que o que diferencia as equações uma das outras são os valores de pressão. Desta forma foram

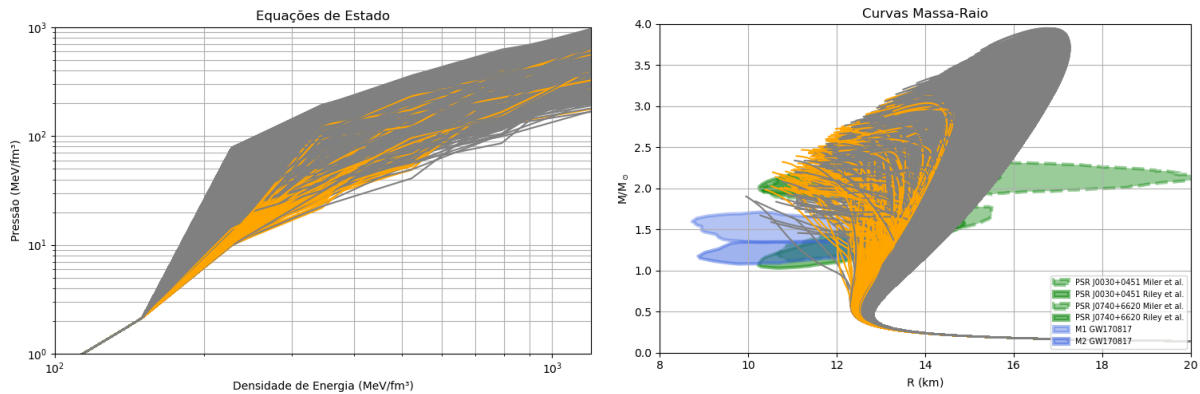
utilizados 200 valores de pressão correspondentes a 200 valores de densidade de energia, distribuídos linearmente entre 150 e 1200 MeV/fm³, sendo esses valores de densidade de energia iguais para todas as equações. Assim sendo os modelos receberam duzentas variáveis, sendo os duzentos valores de pressão, além dos *labels* informando se cada uma das equações passava ou não nas restrições impostas, conforme a tabela abaixo:

TABELA 4.2 – Forma que as equações são apresentadas aos modelos

EoS	p0	p1	pi	p199	Label
1	V_p0_1	V_p1_1	...	V_p199_1	1
2	V_p0_2	V_p1_2	...	V_p199_2	1
3	V_p0_3	V_p1_3	...	V_p199_3	0
j	V_pi_j
9998	V_p0_9998	V_p1_9998	...	V_p199_9998	0
9999	V_p0_9999	V_p1_9999	...	V_p199_9999	1
10000	V_p0_10000	V_p1_10000	...	V_p199_10000	0

Fonte: Elaborado pelo autor.

onde $Vp_{i,j}$ representa o i -ésimo valor de pressão da j -ésima equação de estado. Toda essa separação e organização dos valores de pressão e organização dos dados também é realizado pelo código do Apêndice B, que no final exporta um *dataframe*, que nada mais é do que um formato estruturado dos dados para ser apresentado aos algoritmos.

FIGURA 4.4 – Equações de Estado e Curvas M-R com *label* 1 (laranja) e *label* 0 (cinza).

Fonte: Elaborado pelo autor.

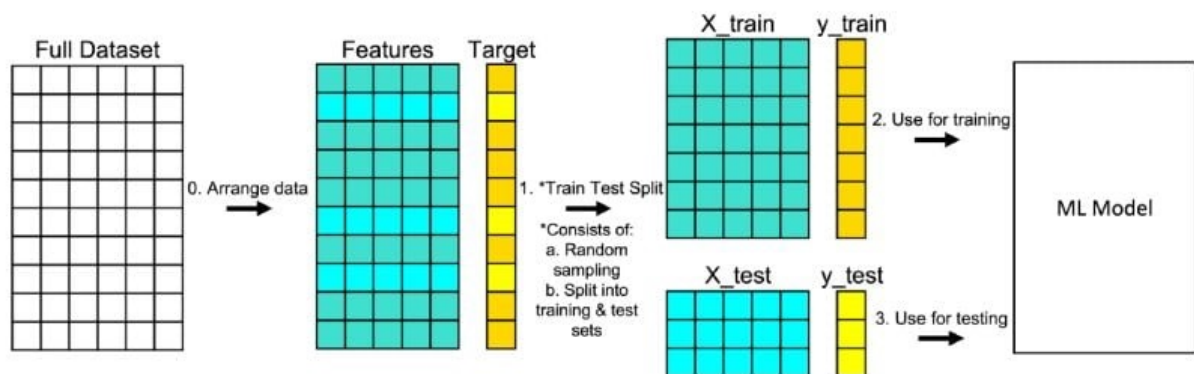
A Figura 4.4 mostra todas as equações de estado já identificadas com o devido *label*, com as equações em cinza tendo recebido o *label* 0 e as em laranja o *label* 1. Também estão representadas as respectivas curvas M-R, seguindo a mesma lógica de cores. Neste caso receberam o *label* 1 a equações de estado que geraram curvas M-R que passavam simultaneamente em todas as regiões observacionais, representadas em azul e verde.

4.3 Divisão dos Dados

É importante que, durante a avaliação dos modelos, sejam utilizados dados que não foram vistos durante o treinamento, já que os dados de treino já são conhecidos e se espera que a taxa de acerto seja elevada quando os modelos são testados com estes dados. Para garantir que o aprendizado ocorreu é necessário que os modelos sejam capazes de realizar a previsão da classe (*label*) das equações mesmo que nunca tenha tido contato com elas. Para isso é necessário dividir os dados em dois subconjuntos: dados de treino e dados de teste, conforme exemplificado na Figura 4.5.

Os dados de treino são os dados que serão utilizados pelos modelos durante o processo de aprendizagem. É através deles que os modelos aprenderão a relação entre os valores de pressão e a classe de cada equação de estado. Com base neste aprendizado eles poderão então receber novas equações de estado e dizer a qual classe ela pertence, e isso será feito com os dados de teste. Como os modelos nunca viram os dados de teste, a sua previsão ocorrerá sem nenhum tipo de viés. Como essas equações já foram etiquetadas previamente, poderemos comparar a classe atribuída pelos modelos à classe real de cada uma das equações para então medir o desempenho dos modelos com o auxílio das métricas de avaliação.

FIGURA 4.5 – Divisão dos dados entre treino e teste.



Fonte: (GALARNYK, 2022).

Usualmente são separados 70% dos dados para treino e 30% para teste, de forma completamente aleatória. Porém para este trabalho foi adotada uma abordagem diferente. Após o treinamento dos modelos com as 10 mil equações geradas, serão geradas 10 mil novas equações para realização dos testes, totalizando 20 mil equações geradas. Além disto, alguns modelos ainda realizam divisões adicionais nos dados de treino, além do processo de *cross-validation* que também faz subdivisões dos dados de treino para avaliar o modelo ainda durante o processo de aprendizagem.

4.4 Treinamento dos Modelos

Para o treinamento dos modelos em si, foram escolhidas 4 técnicas das apresentadas no Capítulo 3: *Gradient Boosting*, LGMB, XGBoost e SVM (com *kernel* linear). O intuito foi comparar 3 modelos que usam como base árvores de decisão entre si (GB, LGBM e XGB), mas também com um modelo que utiliza uma outra abordagem (SVM). Para todos os modelos foram aplicados os seguintes procedimentos:

- a) balanceamento das classes: para os casos onde os dados estavam desbalanceados, foi necessário aplicar o SMOTE para equilibrar as classe;
- b) normalização dos dados: os dados foram normalizados utilizando a padronização *z-score*;
- c) escolha dos hiperparâmetros: cada algoritmo possui uma série de hiperparâmetros que podem ser ajustados e que afetam significativamente a performance do modelo. Foi aplicada uma técnica chamada *grid-search*, que busca os melhores valores e combinações para os hiperparâmetros visando aumentar a performance do modelo.

Todo o treinamento dos modelos foi realizado no *framework* proprietário da *Power of Data*. Com ele foi possível apresentar os parâmetros desejados para o treinamento (conforme explicitado anteriormente) e os dados de treinamento e teste, e ao final obter os modelos treinados. Este *framework* foi desenvolvido pela *Power of Data* justamente para facilitar a criação de modelos de *machine learning*, já que através apenas do ajuste de alguns parâmetros é possível realizar todos os passos que envolvem o treinamento de um modelos, desde o pré-processamento dos dados até a avaliação dos modelos em si. Utilizando este *framework* o treinamento de cada modelo levou cerca de 2 horas.

Com os modelos treinados o próximo passo foi testá-los com as 10 mil novas equações geradas utilizando as métricas de avaliação apresentadas no Capítulo 3.

4.5 Avaliação dos Modelos

A avaliação de quais modelos performam melhor ou pior ocorre observando todo o conjunto de métricas que foram escolhidas para a realização da avaliação, já que olhar apenas para algumas métricas de forma isolada pode levar a conclusões erradas. Sendo assim, os modelos foram avaliados seguindo as seguintes métricas:

- a) *cross-validation*: ainda com os dados em treino, a acurácia, *Recall*, *Precision*, F1, Kappa, MCC, KS e Gini serão avaliadas para 5 diferentes subconjuntos dos dados

de treino. Um bom modelo deve apresentar um baixo desvio padrão para cada uma das métricas;

- b) *confusion matrix*: através da matriz de confusão se tem uma ideia geral da performance do modelo avaliando-se a taxa de acertos para cada uma das classes;
- c) acurácia: a acurácia mostra a taxa geral de acertos do modelo. O desejado neste trabalho é uma taxa de acertos acima de 90%;
- d) *precision*: a precisão medirá quantas equações que foram classificadas como 1 realmente eram daquela classe. O buscado é uma precisão acima de 85%;
- e) *recall*: a revocação medirá quantas equações que eram da classe 1 foram corretamente classificadas. O buscado é uma revocação acima de 85%;
- f) F1: mostra a precisão e revocação de forma resumida, sendo a média harmônica das duas medidas. O desejado é que seja acima de 85%;
- g) curva de calibração: mede o quão bem as probabilidades de classificação do modelo estão calibradas. O desejado é uma curva perto de uma calibração perfeita;
- h) Gini: mede a qualidade de uma classificação binária (0 ou 1). O desejado é um valor acima de 0,8;
- i) *Kappa*: avalia a concordância entre as previsões do modelo e os rótulos verdadeiros das classes. O desejado é um kappa acima de 0,5;
- j) MCC: mede a qualidade geral da classificação do modelo. O desejado é um valor acima de 0,5;
- k) KS: mede a diferença máxima entre as curvas ROC do modelo e a curva ROC ideal. O desejado é que seja acima de 0,8.

Os valores para todas as métricas foram obtidos diretamente do *framework* da *Power of Data*. Após avaliação de cada um dos modelos eles foram comparados a fim de verificar qual performou melhor de acordo com o resultado esperado para este trabalho. A discussão dos resultados, a apresentação das métricas e a comparação dos modelos em si será apresentada no próximo capítulo.

5 Resultados

Como dito no capítulo anterior, os modelos foram treinados utilizando quatro técnicas diferentes (*Gradient Boosting*, LGBM, XGBoost e SVM - linear) e para identificar as equações de estado que geram curvas M-R que atendam as restrições impostas (passam pela região LIGO, NICER, e em ambas simultaneamente), totalizando doze modelos diferentes. Os modelos foram comparados entre si de acordo com a proposta de cada um, ou seja, os quatro modelos que identificam apenas a passagem das curvas M-R pela região LIGO serem comparados entre si, e o mesmo vale para as demais situações.

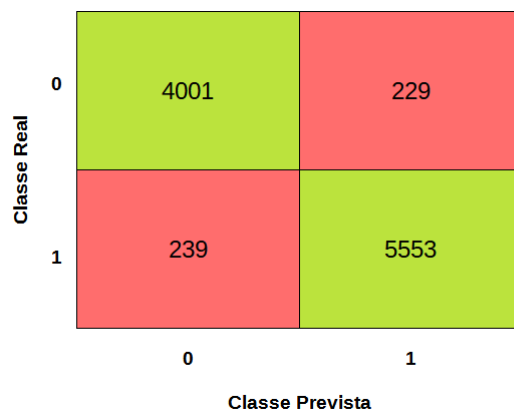
5.1 Modelos para a região LIGO

O primeiro passo para se avaliar os modelos é identificar a taxa de acerto para cada uma das classes, aqui sendo as equações que passam na região do LIGO (classe 1) e as que não passam. A forma mais imediata de se ter um primeiro vislumbre desta taxa de acertos é observando a *confusion matrix* de cada um deles. Nesta matrix pode-se, de forma rápida, avaliar o total das equações que foram classificadas corretamente. As Figuras 5.1, 5.2, 5.3 e 5.4 apresentam as matrizes para os quatro modelos.

FIGURA 5.1 – *Confusion Matrix* do modelo *Gradient Boosting* - LIGO

	0	1
0	4012	219
1	250	5540

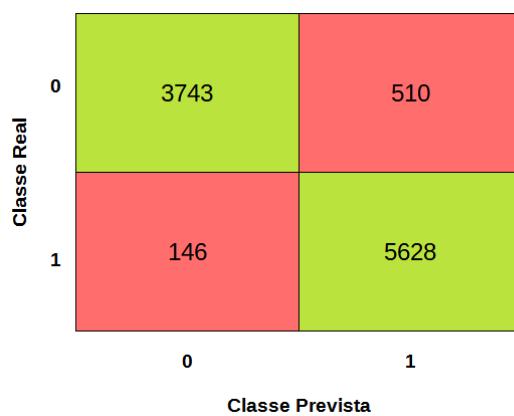
Fonte: Elaborado pelo autor.

FIGURA 5.2 – *Confusion Matrix* do modelo LGBM - LIGO

A 2x2 confusion matrix for the LGBM model. The y-axis is labeled 'Classe Real' with values 0 and 1. The x-axis is labeled 'Classe Prevista' with values 0 and 1. The cells contain the following counts: (0,0) is 4001 (green), (0,1) is 229 (red), (1,0) is 239 (red), and (1,1) is 5553 (green).

Classe Real \ Classe Prevista	0	1
0	4001	229
1	239	5553

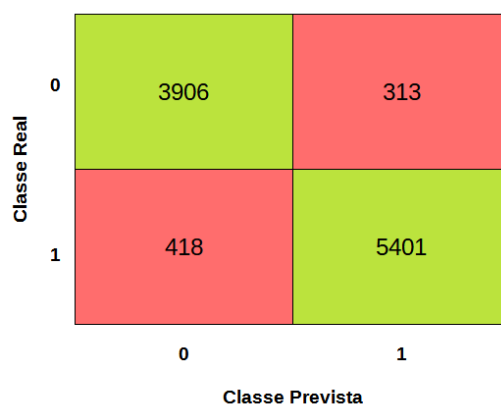
Fonte: Elaborado pelo autor.

FIGURA 5.3 – *Confusion Matrix* do modelo XGBoost - LIGO

A 2x2 confusion matrix for the XGBoost model. The y-axis is labeled 'Classe Real' with values 0 and 1. The x-axis is labeled 'Classe Prevista' with values 0 and 1. The cells contain the following counts: (0,0) is 3743 (green), (0,1) is 510 (red), (1,0) is 146 (red), and (1,1) is 5628 (green).

Classe Real \ Classe Prevista	0	1
0	3743	510
1	146	5628

Fonte: Elaborado pelo autor.

FIGURA 5.4 – *Confusion Matrix* do modelo SVM - LIGO

A 2x2 confusion matrix for the SVM model. The y-axis is labeled 'Classe Real' with values 0 and 1. The x-axis is labeled 'Classe Prevista' with values 0 and 1. The cells contain the following counts: (0,0) is 3906 (green), (0,1) is 313 (red), (1,0) is 418 (red), and (1,1) is 5401 (green).

Classe Real \ Classe Prevista	0	1
0	3906	313
1	418	5401

Fonte: Elaborado pelo autor.

De modo geral, todos os modelos tiveram um bom desempenho na classificação das equações de estado, independente da classe original. Esse desempenho fica claro quando se observa a quantidade das equações que foram classificadas corretamente em relação as que foram classificadas erroneamente. Ainda assim, se faz necessário avaliar as demais métricas para se chegar a uma conclusão de qual foi o modelo que melhor performou, métricas essas que se utilizam dos resultados obtidos na *confusion matrix*. Resumidamente, a *confusion matrix* serve de ponto de partida e de insumo para o cálculo das demais métricas.

Como dito no Capítulo 3, o *cross validation* serve para avaliar como diversas métricas variam durante o treinamento quando se utiliza diversos subconjuntos dos dados de treino. Para os modelos deste trabalho foi utilizada a versão *k-fold* da validação cruzada, com $k = 5$. As Tabelas 5.1, 5.2, 5.3 e 5.4 contêm os valores das métricas acurácia, *recall*, *precision*, *F1-score*, Kappa, MCC e Gini para cada um dos modelos e para cada *fold*.

TABELA 5.1 – *Cross Validation* do modelo *Gradient Boosting* - LIGO

Fold	Acurácia	Recall	Precision	F1	Kappa	MCC	KS	Gini
0	0,9592	0,9643	0,9620	0,9631	0,9174	0,9174	0,9200	0,9516
1	0,9549	0,9573	0,9610	0,9591	0,9088	0,9088	0,9100	0,9348
2	0,9568	0,9612	0,9608	0,9610	0,9126	0,9126	0,9100	0,9416
3	0,9575	0,9565	0,9663	0,9614	0,9141	0,9141	0,9200	0,9448
4	0,9476	0,9448	0,9597	0,9522	0,8941	0,8943	0,9000	0,9290

Fonte: Elaborado pelo autor.

TABELA 5.2 – *Cross Validation* do modelo *LGBM* - LIGO

Fold	Acurácia	Recall	Precision	F1	Kappa	MCC	KS	Gini
0	0,9549	0,9599	0,9584	0,9592	0,9088	0,9088	0,9100	0,9416
1	0,9536	0,9611	0,9551	0,9581	0,9061	0,9061	0,9100	0,9424
2	0,9562	0,9525	0,9676	0,9600	0,9116	0,9117	0,9200	0,9436
3	0,9530	0,9549	0,9597	0,9573	0,9049	0,9049	0,9100	0,9370
4	0,9592	0,9650	0,9612	0,9631	0,9174	0,9174	0,9200	0,9398

Fonte: Elaborado pelo autor.

TABELA 5.3 – *Cross Validation* do modelo XGBoost - LIGO

Fold	Acurácia	Recall	Precision	F1	Kappa	MCC	KS	Gini
0	0,9328	0,9728	0,9115	0,9412	0,8629	0,8654	0,9100	0,9364
1	0,9426	0,9732	0,9268	0,9494	0,8833	0,8847	0,9100	0,9380
2	0,9319	0,9763	0,9076	0,9407	0,8610	0,8642	0,9100	0,9368
3	0,9420	0,9771	0,9226	0,9491	0,8818	0,8838	0,9200	0,9488
4	0,9362	0,9740	0,9160	0,9441	0,8699	0,8722	0,9200	0,9436

Fonte: Elaborado pelo autor.

TABELA 5.4 – *Cross Validation* do modelo SVM - LIGO

Fold	Acurácia	Recall	Precision	F1	Kappa	MCC	KS	Gini
0	0,9298	0,9225	0,9488	0,9355	0,8584	0,8588	0,8600	0,9306
1	0,9287	0,9276	0,9423	0,9349	0,8561	0,8562	0,8600	0,9210
2	0,9280	0,9327	0,9367	0,9347	0,8546	0,8546	0,8600	0,9312
3	0,9358	0,9315	0,9511	0,9412	0,8704	0,8707	0,8700	0,9420
4	0,9312	0,9264	0,9478	0,9370	0,8613	0,8616	0,8700	0,9346

Fonte: Elaborado pelo autor.

Como informado no Capítulo 4, um bom modelo é aquele que, além de ter valores dentro de uma faixa específica para cada métrica, também deve apresentar um baixo desvio padrão na validação cruzada. A Tabelas 5.5 compara a média e a Tabela 5.6 o desvio padrão dos modelos para cada uma das métricas, e destaca aqueles que melhor desempenharam em cada métrica.

TABELA 5.5 – Comparação da média para os modelos da região LIGO

Modelo	Acurácia	Recall	Precision	F1	Kappa	MCC	KS	Gini
GB	0,9552	0,9568	0,9620	0,9594	0,9094	0,9094	0,9150	0,9404
LGBM	0,9554	0,9587	0,9604	0,9595	0,9098	0,9098	0,9140	0,9409
XGBoost	0,9371	0,9747	0,9169	0,9449	0,8718	0,8741	0,9140	0,9407
SVM	0,9307	0,9281	0,9453	0,9367	0,8602	0,8604	0,8640	0,9319

Fonte: Elaborado pelo autor.

TABELA 5.6 – Comparação do desvio padrão para os modelos da região LIGO

Modelo	Acurácia	Recall	Precision	F1	Kappa	MCC	KS	Gini
GB	0,0045	0,0074	0,0026	0,0042	0,0091	0,0090	0,0071	0,0088
LGBM	0,0025	0,0050	0,0046	0,0022	0,0050	0,0050	0,0055	0,0026
XGBoost	0,0050	0,0019	0,0079	0,0042	0,0104	0,0098	0,0055	0,0054
SVM	0,0031	0,0041	0,0058	0,0027	0,0063	0,0064	0,0055	0,0076

Fonte: Elaborado pelo autor.

Observando as médias de cada métrica dos modelos na Tabela 5.5, o modelo LGBM performou melhor em 5 métricas, sendo elas a acurácia, F1-score, Kappa, MCC e Gini. O modelo *Gradient Boosting* teve melhor o desempenho nas métricas *precision* e KS, enquanto o modelo XGBoost performou melhor na métrica *recall*. Já o modelo SVM não teve a melhor performance em nenhuma das métricas, mas vale ressaltar que seus valores para todas elas também estão acima dos valores desejados e estabelecidos no Capítulo 4, bem como todos os demais modelos.

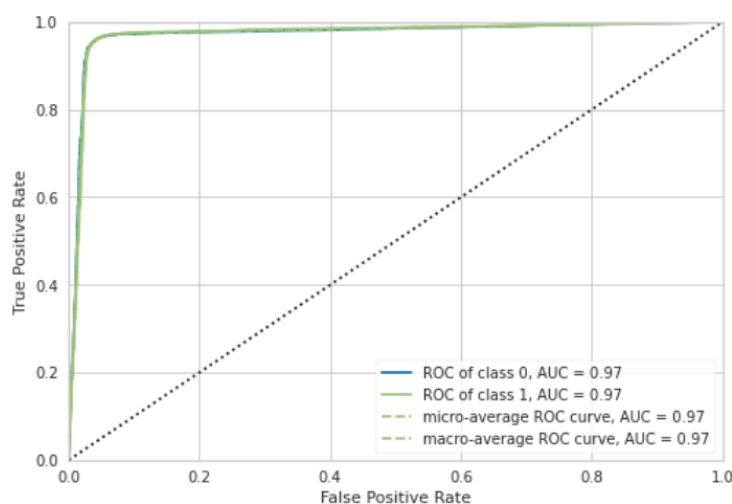
Em relação ao desvio padrão o resultado não se mostrou muito diferente, como observado na Tabela 5.6. O modelo com a melhor performance se mostrou novamente sendo o LGBM, com o menor valor para as métricas acurácia, F1-score, Kappa, MCC, KS e Gini. O modelo *Gradient Boosting* teve o menor desvio padrão para as métricas *precision*, o modelo XGBoost para as métricas *recall* e KS, e o modelo SVM para a métrica KS, com o mesmo valor obtido para os modelos LGBM e XGBoost.

A partir desses valores é possível concluir que o modelo LGBM obteve a melhor performance geral dos quatro modelos, tendo o melhor resultado para cinco das oito métricas apresentadas em relação à média, e seis das oito métricas em relação ao desvio padrão, sendo que cinco delas são as mesmas em ambas as comparações. Sendo o modelo com a melhor acurácia ele é o modelo que mais acertou a classificação das equações de estado, acertando, em média, 95,5% dos dados. Com um F1-score de 95,9%, o modelo também se mostrou capaz de classificar corretamente as equações, independente de suas classes, isto é, classificou corretamente, em média, 95,9% das equações que passavam pela região do LIGO (classe 1) e também as que não passavam (classe 0). A qualidade dessa classificação também ficou evidente nos índices MCC, Gini e Kappa, usados para medir a qualidade geral de uma classificação binária e a concordância entre as previsões do modelo e os rótulos verdadeiros das classes; seus valores foram, MCC = 0,9, Gini = 0,94 e Kappa = 0,9, sendo que o desejado era que fossem MCC > 0,5, Gini > 0,8 e Kappa > 0,5. O *precision* e *recall*, por mais que não tenham tido a melhor performance dentre todos os modelos, ainda assim obtiveram valores muito satisfatórios para o modelo LGBM, sendo ambos acima do valor desejado, que era de 85%. Já o índice KS, que mede a diferença máxima entre as curvas ROC do modelo e a curva ROC ideal também ficou acima do

valor desejado de 80%.

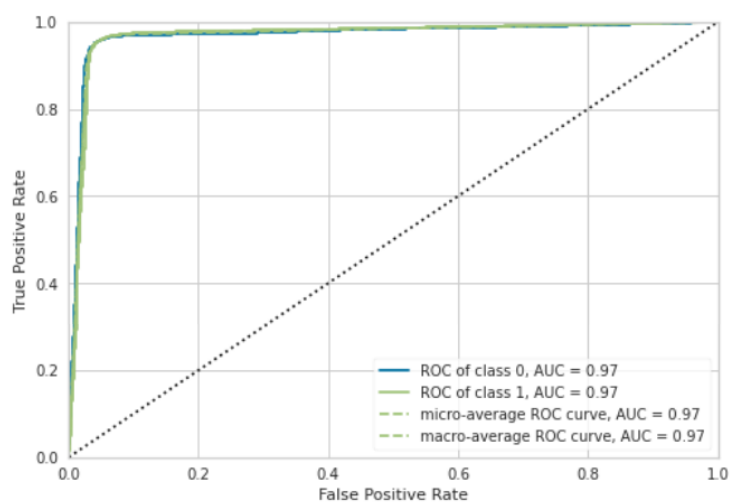
Outra métrica utilizada para avaliar os modelos foi a curva ROC, que como visto anteriormente é utilizada para o cálculo do índice KS e serve para avaliar o quanto o modelo se aproxima de um classificador perfeito, porém de uma forma mais visual. É através dela que se pode observar também se o modelo está ou não com *overfitting*, ou seja, se ajustou demais aos dados de treino e pode não apresentar bons resultados em dados mais gerais. As Figuras 5.5, 5.6, 5.7 e 5.8 apresentam as curvas ROC de cada um dos modelos.

FIGURA 5.5 – Curva ROC do modelo *Gradient Boosting*



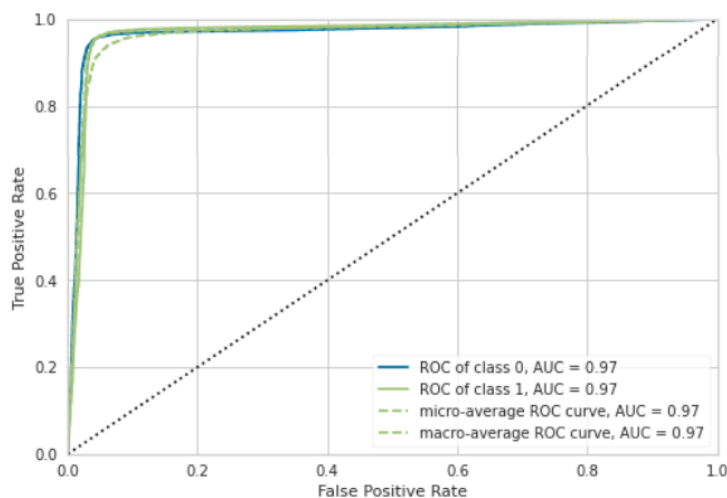
Fonte: Elaborado pelo autor.

FIGURA 5.6 – Curva ROC do modelo LGBM



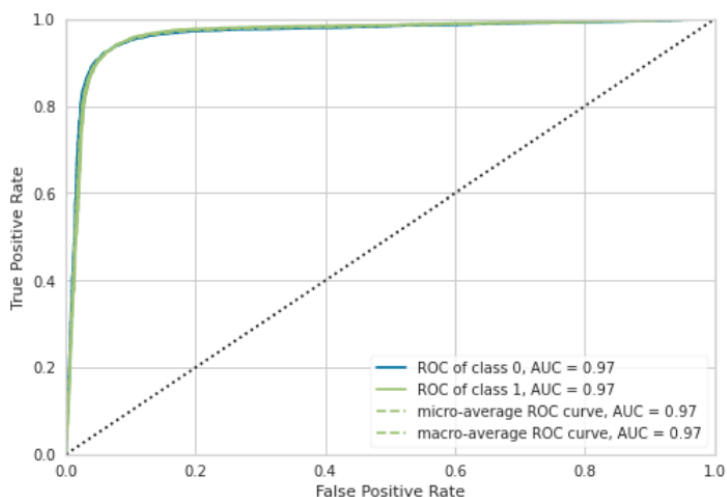
Fonte: Elaborado pelo autor.

FIGURA 5.7 – Curva ROC do modelo XGBoost



Fonte: Elaborado pelo autor.

FIGURA 5.8 – Curva ROC do modelo SVM



Fonte: Elaborado pelo autor.

É possível observar que a curva de todos os modelos se aproxima do ponto de um modelo de classificação perfeito, mas não o suficiente para indicar uma situação de *overfitting*. Outro ponto interessante de ser observado é na Figura 5.8, em que o afastamento mais acentuado do modelo SVM do ponto de um modelo perfeito reflete seu desempenho sendo menor do que os demais modelos quando foram avaliadas as demais métricas. Ainda assim vale repetir que seu desempenho também foi muito satisfatório.

Apesar do bom desempenho para todos os modelos, é importante lembrar que essas métricas foram calculadas na validação cruzada durante o treinamento, ou seja, utilizando os próprios dados de treino. Por mais que a validação cruzada utilize subconjuntos dos dados de treino que não foram utilizados no aprendizado para testar o modelo e calcular

as métricas, é importante que o desempenho real dos modelos seja testado com os dados de teste, que não foram vistos em momento algum pelo modelo. Os modelos foram então apresentados às equações geradas para realização dos testes, e a Tabela 5.7 apresenta a comparação do seus desempenhos, com destaque para os que melhor performaram em cada métrica.

TABELA 5.7 – Comparação dos modelos com os dados de teste - LIGO

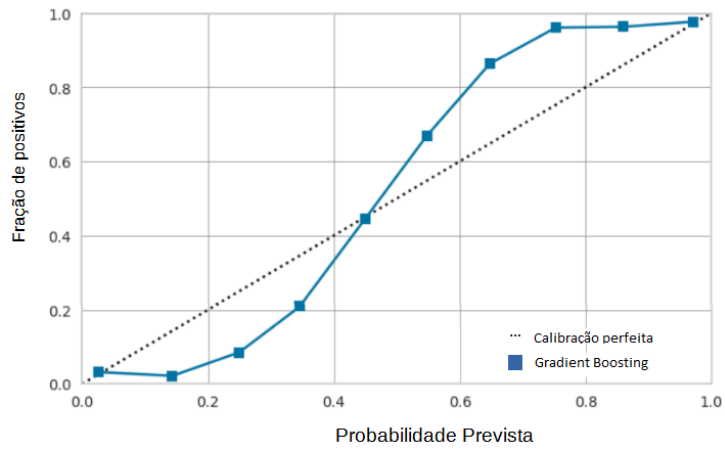
Modelo	Acurácia	Recall	Precision	F1	Kappa	MCC	KS	Gini
GB	0,9453	0,9544	0,9640	0,9592	0,9099	0,9099	0,9100	0,9446
LGBM	0,9545	0,9556	0,9796	0,9675	0,9196	0,9099	0,9240	0,9405
XGBoost	0,9393	0,9771	0,9179	0,9466	0,8765	0,8788	0,9200	0,9452
SVM	0,9336	0,9317	0,9468	0,9392	0,8661	0,8662	0,8700	0,9340

Fonte: Elaborado pelo autor.

Todos os modelos tiveram uma performance com valores acima dos desejados, com destaque para o modelo LGBM que teve a melhor performance em cinco das oito métricas analisadas. Esse resultado com os dados de teste reflete o observado com os dados de treino, e confirma o fato de que os modelos não estavam em uma situação de *overfitting*, como observado na análise das curvas ROC. Ainda assim há um último aspecto dos modelos a ser analisado antes de concluir obteve a melhor performance.

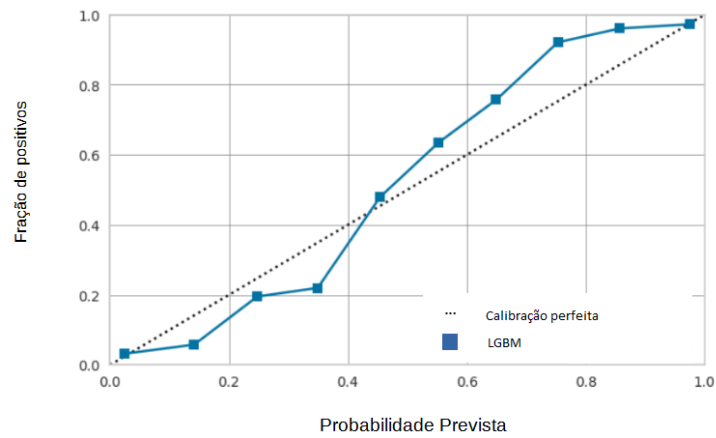
O próximo ponto de análise dos modelos é a curva de calibração, que mostra se as probabilidades informadas pelas previsões condizem com as observadas nos dados de teste. Apesar dos dados de teste não terem sido vistos pelos modelos em momento algum durante o treinamento, eles também possuem a informação de sua classe original, de modo análises das métricas, como feito previamente, e da curva de calibração sejam possíveis. De forma objetiva, se o modelo previu que, por exemplo, 100 equações possuem uma probabilidade de 70% de pertencerem a classe 1 (passam pela região do LIGO), então 70 dessas equações devem ser de fato da classe 1, e 30 da classe 0 (não passam). Para um modelo perfeitamente calibrado, esse relação vale para todas as probabilidades calculadas, de modo que em um gráfico de probabilidades previstas pela fração de positivos tem-se uma reta. Esse gráfico é justamente a curva de calibração. As Figuras 5.9, 5.10, 5.11 e 5.12 apresentam as curvas para cada um dos modelos.

FIGURA 5.9 – Curva de calibração do modelo *Gradient Boosting*



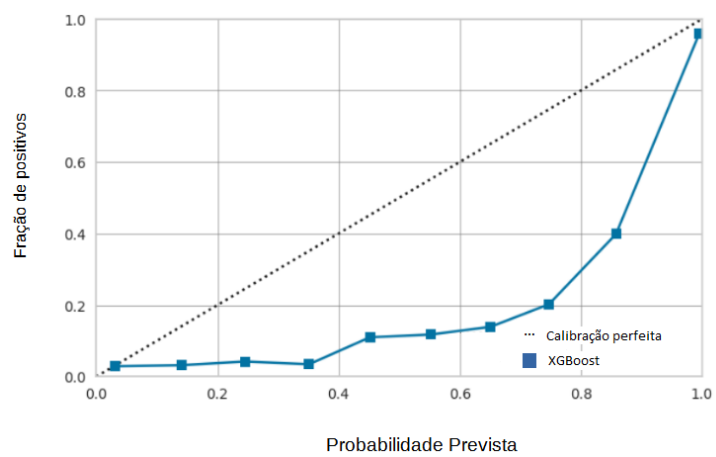
Fonte: Elaborado pelo autor.

FIGURA 5.10 – Curva de calibração do modelo LGBM



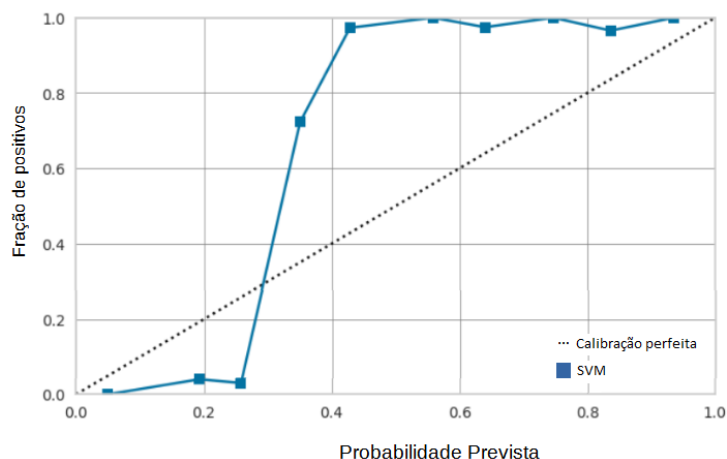
Fonte: Elaborado pelo autor.

FIGURA 5.11 – Curva de calibração do modelo XGBoost



Fonte: Elaborado pelo autor.

FIGURA 5.12 – Curva de calibração do modelo SVM



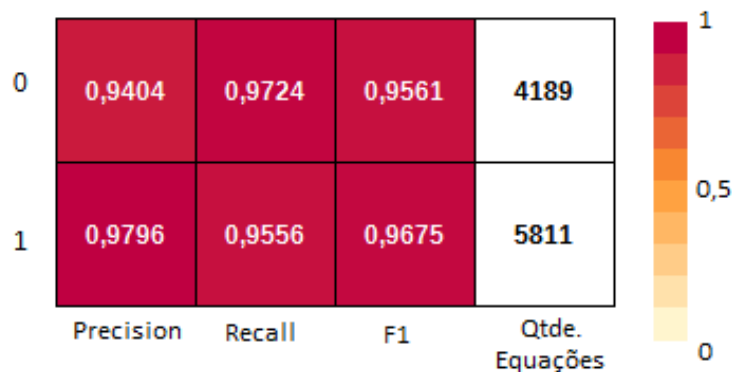
Fonte: Elaborado pelo autor.

Os quatro modelos apresentaram curvas de calibração bem distintas, sendo os modelos *Gradient Boosting* e LGBM os que mais se aproximaram de uma calibração perfeita. Já os modelos XGBoost e SVM mostraram uma calibração um pouco desviada do ideal. Observando a Figura 5.11 é possível notar que o modelo previu probabilidades muito acima das observadas, enquanto na Figura 5.12 ocorreu o oposto, o modelo previu probabilidades muito abaixo das reais. É importante ressaltar que curvas descalibradas não indicam um modelo necessariamente ruim, já que é possível que um modelo tenha altas métricas de acurácia, *recall* e *precision*, mas ainda apresente uma curva de calibração descalibrada. Isso ocorre porque essas métricas medem diferentes aspectos do desempenho do modelo em relação aos rótulos das classes, enquanto a curva de calibração está relacionada à confiança atribuída às previsões de probabilidade. Uma alta acurácia, *recall* e *precision* (como no caso dos modelos XGBoost e SVM) não garantem automaticamente que o modelo tenha uma curva de calibração bem ajustada. É fundamental avaliar a calibração do modelo separadamente, mas usá-lo em conjunto com as demais métricas para tomar alguma decisão sobre qual modelo melhor performa.

Após a análise do desempenho dos modelos com dados de treino e de teste, é possível afirmar que para o caso em que se deseja identificar as equações de estado que geram curvas que passam pela região do LIGO, o modelo LGBM foi o que obteve a melhor performance geral. Os valores de todas as métricas, tanto em treino quanto em teste, ficaram acima dos valores desejados e estabelecidos no Capítulo 4, além do modelo ter mostrado uma boa calibração. O mesmo pode ser dito do modelo *Gradiente Boosting*, que também apresentou bons valores para as métricas e uma calibração aceitável. Os demais modelos, XGBoost e SVM, apesar de também terem obtido bons valores para as métricas, se mostraram um pouco inferiores apenas por possuírem curvas de calibração descalibradas, o que acaba afetando a confiança nas probabilidades previstas. A Figura

5.13 resume a performance geral do modelo LGBM.

FIGURA 5.13 – Performance geral do modelo LGBM para o caso LIGO



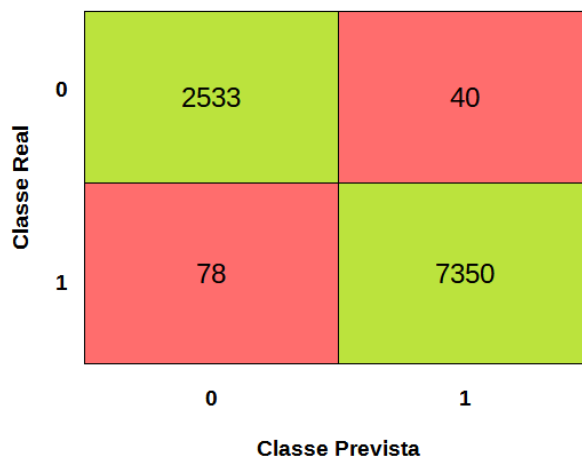
Fonte: Elaborado pelo autor.

Dentre todas as 10.000 equações utilizadas durante o teste, 42% pertenciam à classe 0 (não passam pela região LIGO), enquanto 58% pertenciam à classe 1 (passam pela região LIGO). O modelo apresentou um ótimo desempenho tanto para classificar as equações da classe 0 quanto da classe 1.

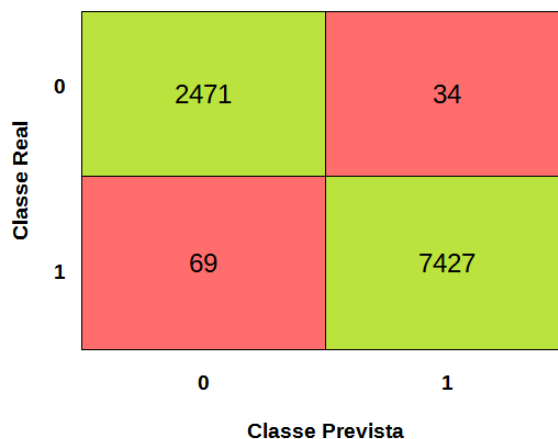
5.2 Modelos para a região NICER

Os modelos discutidos a seguir foram treinados para classificar as curvas considerando se elas passam (classe 1) ou não (classe 0) nas regiões observacionais do NICER. Iniciando pela *confusion matrix*, as Figuras 5.14, 5.15, 5.16 e 5.17 apresentam as matrizes para os quatro modelos treinados.

FIGURA 5.14 – *Confusion Matrix* do modelo *Gradient Boosting* - NICER



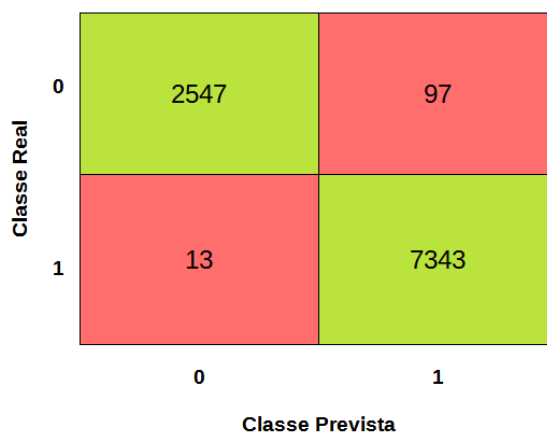
Fonte: Elaborado pelo autor.

FIGURA 5.15 – *Confusion Matrix* do modelo LGBM - NICER

A 2x2 confusion matrix for the LGBM model. The vertical axis is labeled 'Classe Real' with values 0 and 1. The horizontal axis is labeled 'Classe Prevista' with values 0 and 1. The cells contain the following counts: (0,0) is 2471, (0,1) is 34, (1,0) is 69, and (1,1) is 7427.

Classe Real \ Classe Prevista	0	1
0	2471	34
1	69	7427

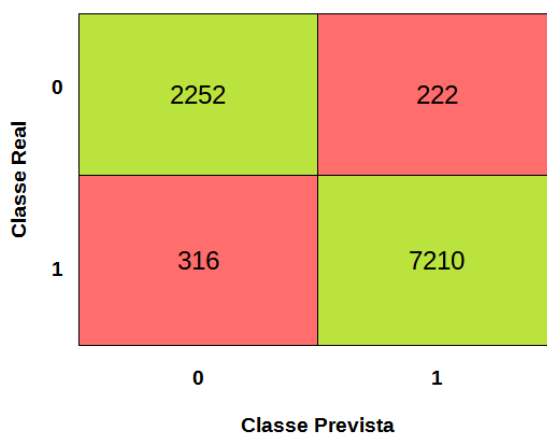
Fonte: Elaborado pelo autor.

FIGURA 5.16 – *Confusion Matrix* do modelo XGBoost - NICER

A 2x2 confusion matrix for the XGBoost model. The vertical axis is labeled 'Classe Real' with values 0 and 1. The horizontal axis is labeled 'Classe Prevista' with values 0 and 1. The cells contain the following counts: (0,0) is 2547, (0,1) is 97, (1,0) is 13, and (1,1) is 7343.

Classe Real \ Classe Prevista	0	1
0	2547	97
1	13	7343

Fonte: Elaborado pelo autor.

FIGURA 5.17 – *Confusion Matrix* do modelo SVM - NICER

A 2x2 confusion matrix for the SVM model. The vertical axis is labeled 'Classe Real' with values 0 and 1. The horizontal axis is labeled 'Classe Prevista' with values 0 and 1. The cells contain the following counts: (0,0) is 2252, (0,1) is 222, (1,0) is 316, and (1,1) is 7210.

Classe Real \ Classe Prevista	0	1
0	2252	222
1	316	7210

Fonte: Elaborado pelo autor.

Observando as matrizes é possível observar dois pontos: o modelo LGBM foi o que menos cometeu erros, enquanto o modelo SVM foi o que mais cometeu. Apesar disto, a classificação de todos os modelos se mostrou bastante satisfatória. É importante observar que neste caso há muito mais equações da classe 1 do que da classe 0, diferente do caso da região LIGO. Isto pode ser explicado observando a Figura 4.1, onde fica evidente que a região do NICER é muito mais ampla do que a região do LIGO. Desta forma, durante a geração das equações de estado e obtenção das curvas M-R, é muito mais provável que uma curva passe pela região do NICER, de modo que mais equações receberam a classe 1 do que a classe 0.

Analisando a validação cruzada realizada durante o treinamento pode-se acompanhar como variaram as diversas métricas dos modelos. As Tabelas 5.8, 5.9, 5.10 e 5.11 apresentam os valores obtidos durante o *cross validation* para os quatro modelos.

TABELA 5.8 – *Cross Validation* do modelo *Gradient Boosting* - NICER

Fold	Acurácia	Revocação	Precisão	F1	Kappa	MCC	KS	Gini
0	0,9873	0,9875	0,9953	0,9914	0,9675	0,9676	0,9800	0,9988
1	0,9886	0,9895	0,9950	0,9923	0,9707	0,9708	0,9800	0,9990
2	0,9912	0,9933	0,9948	0,9940	0,9773	0,9773	0,9800	0,9992
3	0,9882	0,9898	0,9942	0,9920	0,9696	0,9696	0,9700	0,9988
4	0,9860	0,9875	0,9936	0,9905	0,9641	0,9642	0,9700	0,9986

Fonte: Elaborado pelo autor.

TABELA 5.9 – *Cross Validation* do modelo LGBM - NICER

Fold	Acurácia	Revocação	Precisão	F1	Kappa	MCC	KS	Gini
0	0,9901	0,9898	0,9968	0,9933	0,9748	0,9749	0,9800	0,9994
1	0,9905	0,9927	0,9944	0,9936	0,9758	0,9758	0,9800	0,9992
2	0,9890	0,9909	0,9941	0,9925	0,9720	0,9720	0,9800	0,9990
3	0,9910	0,9927	0,9950	0,9939	0,9769	0,9769	0,9800	0,9992
4	0,9890	0,9877	0,9973	0,9925	0,9721	0,9723	0,9800	0,9992

Fonte: Elaborado pelo autor.

TABELA 5.10 – *Cross Validation* do modelo XGBoost - NICER

Fold	Acurácia	Revocação	Precisão	F1	Kappa	MCC	KS	Gini
0	0,9878	0,9985	0,9851	0,9918	0,9680	0,9683	0,9980	0,9994
1	0,9871	0,9977	0,9850	0,9913	0,9663	0,9666	0,9980	0,9992
2	0,9886	0,9977	0,9970	0,9923	0,9703	0,9705	0,9980	0,9992
3	0,9893	0,9988	0,9868	0,9928	0,9719	0,9722	0,9980	0,9992
4	0,9845	0,9985	0,9808	0,9896	0,9594	0,9600	0,9970	0,9988

Fonte: Elaborado pelo autor.

TABELA 5.11 – *Cross Validation* do modelo SVM - NICER

Fold	Acurácia	Revocação	Precisão	F1	Kappa	MCC	KS	Gini
0	0,9484	0,9636	0,9664	0,9650	0,8673	0,8673	0,8900	0,9770
1	0,9495	0,9601	0,9711	0,9656	0,8711	0,8713	0,9000	0,9788
2	0,9469	0,9592	0,9685	0,9638	0,8644	0,8645	0,9000	0,9784
3	0,9456	0,9504	0,9752	0,9626	0,8630	0,8639	0,8900	0,9760
4	0,9456	0,9566	0,9693	0,9629	0,8614	0,8617	0,8900	0,9760

Fonte: Elaborado pelo autor.

Já as Tabelas 5.12 e 5.13 comparam a média e o desvio padrão dos modelos para cada uma das métricas, e destaca aqueles que melhor desempenharam em cada métrica.

TABELA 5.12 – Comparação da média para os modelos da região NICER

Modelo	Acurácia	Recall	Precision	F1	Kappa	MCC	KS	Gini
GB	0,9883	0,9895	0,9946	0,9920	0,9698	0,9699	0,9760	0,9989
LGBM	0,9899	0,9908	0,9955	0,9932	0,9743	0,9744	0,9800	0,9992
XGBoost	0,9875	0,9982	0,9869	0,9916	0,9672	0,9675	0,9978	0,9992
SVM	0,9472	0,9580	0,9701	0,9640	0,8654	0,8657	0,8940	0,9772

Fonte: Elaborado pelo autor.

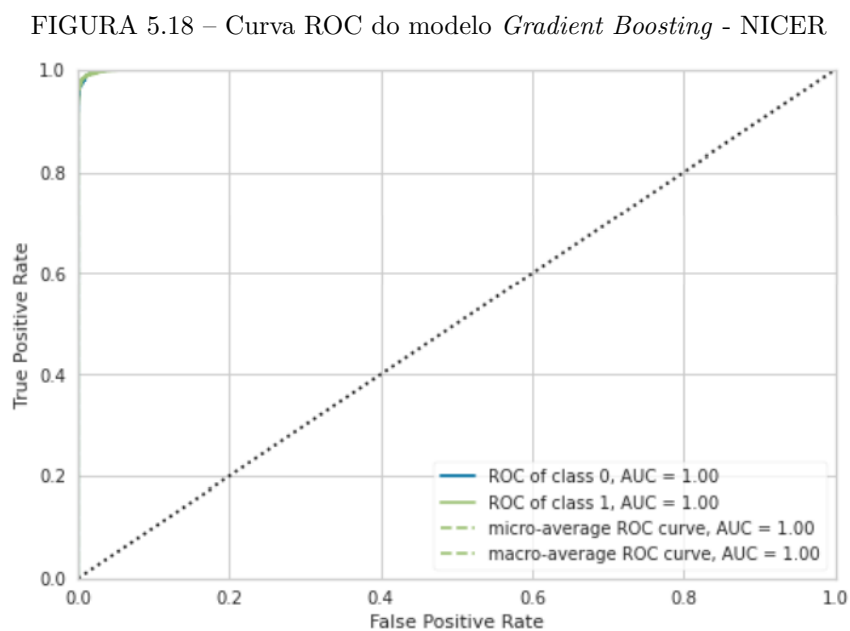
TABELA 5.13 – Comparação do desvio padrão para os modelos da região NICER

Modelo	Acurácia	Recall	Precision	F1	Kappa	MCC	KS	Gini
GB	0,0019	0,0024	0,0007	0,0013	0,0049	0,0048	0,0055	0,0002
LGBM	0,0009	0,0021	0,0014	0,0006	0,0022	0,0022	0,0000	0,0001
XGBoost	0,0019	0,0005	0,0060	0,0012	0,0048	0,0047	0,0004	0,0002
SVM	0,0017	0,0049	0,0033	0,0013	0,0038	0,0037	0,0055	0,0013

Fonte: Elaborado pelo autor.

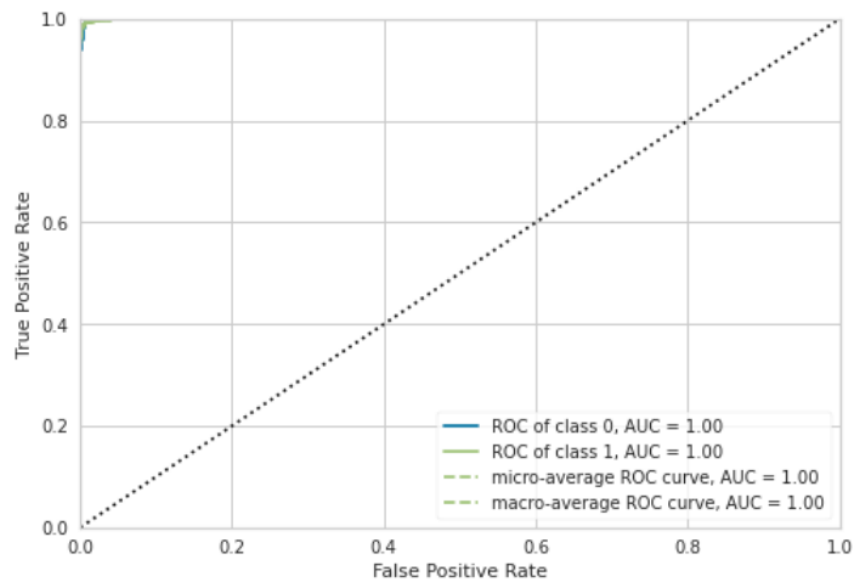
O modelo LGBM foi o que melhor performou, tendo o melhor resultado em seis das oito métricas tanto na comparação da média quanto do desvio padrão. É interessante observar que todas as métricas que medem tanto a taxa de acertos quanto a qualidade da classificação obtiveram valores muito elevados, como a acurácia que atingiu praticamente 99% para o modelo LGBM. À primeira vista isso pode levar a compreensão de que esse modelo beira à perfeição e consegue classificar precisamente praticamente todas as equações que forem apresentadas, mas isso não pode ser tido como verdade. É importante lembrar que muitas das equações pertenciam a classe 1, de modo que os modelos podem ter entrado em *overfitting* e aprendido demais apenas sobre este tipo de equação. Ainda assim, quando observamos as matrizes das Figuras 5.14, 5.15, 5.16, e 5.17, vemos que eles também conseguiram classificar corretamente uma grande quantidade das equações da classe 0, de forma que apenas com essas informações ainda não é possível obter uma conclusão.

As curvas ROC são bons indicativos de um eventual caso de *overfitting*. As Figuras 5.18, 5.19, 5.20 e 5.21 apresentam as curvas ROC para os quatro modelos.



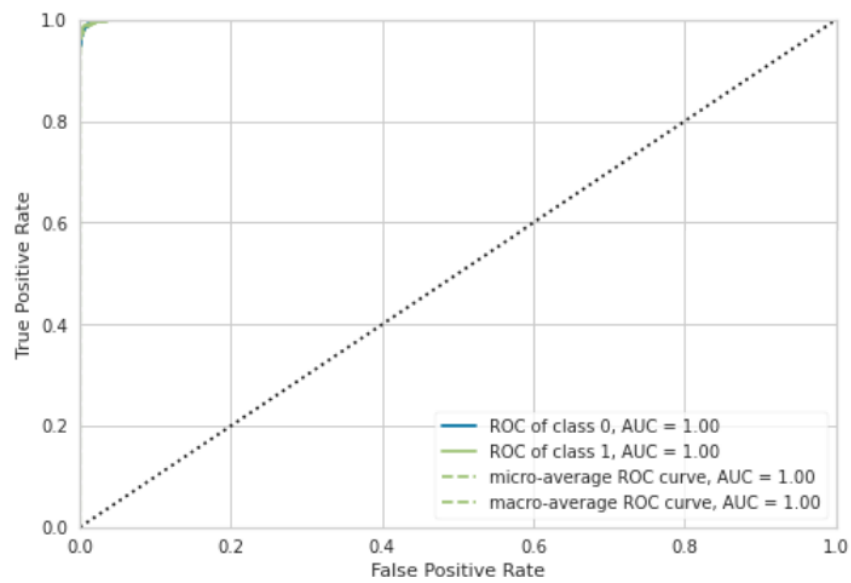
Fonte: Elaborado pelo autor.

FIGURA 5.19 – Curva ROC do modelo LGBM - NICER



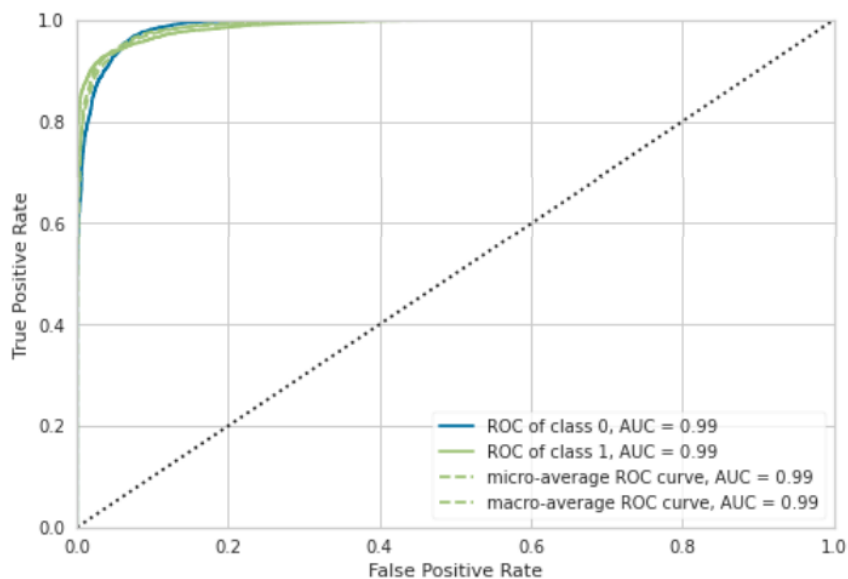
Fonte: Elaborado pelo autor.

FIGURA 5.20 – Curva ROC do modelo XGBoost - NICER



Fonte: Elaborado pelo autor.

FIGURA 5.21 – Curva ROC do modelo SVM - NICER



Fonte: Elaborado pelo autor.

As curvas dos modelos *gradient boosting*, LGBM e XGBoost estão extremamente próximas do ponto de um modelo perfeito, o que a princípio mostra um forte indício de *overfitting*. Já a curva ROC do modelo SVM se encontra em uma situação mais aceitável, e reflete os valores levemente menores obtidos nas métricas apresentadas na Tabela 5.12, como a acurácia de $\approx 95\%$. Para concluir se os modelos que aparentam estar em *overfitting* realmente se encontram nesta situação, é preciso avaliar seus desempenhos com os dados de teste, nunca vistos pelo modelo. Um modelo em *overfitting* aprendeu demais sobre os dados de treino e acabou se tornando muito particular para esses dados, de modo que, ao se deparar com dados novos, seu desempenho tende a ser muito inferior. Logo, a única forma de concluir se há ou não o *overfitting* é testando os modelos com novos dados.

Ao serem apresentados com as equações de teste, os modelos apresentaram desempenho conforme Tabela 5.14.

TABELA 5.14 – Comparação dos modelos com os dados de teste - NICER

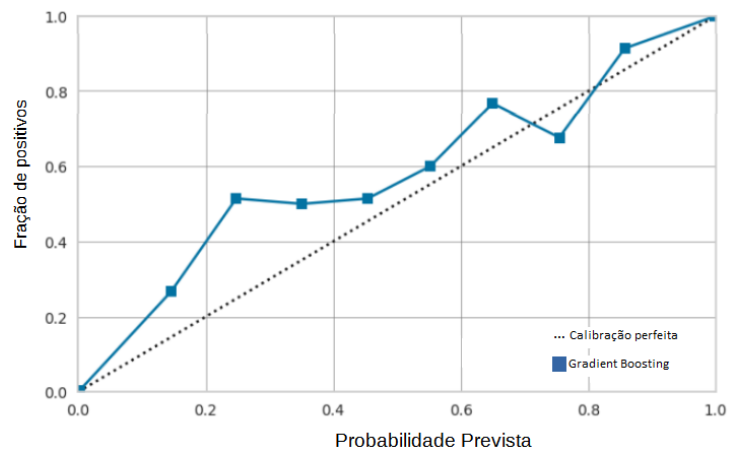
Modelo	Acurácia	Recall	Precision	F1	Kappa	MCC	KS	Gini
GB	0,9902	0,9910	0,9956	0,9933	0,9751	0,9751	0,9751	0,9994
LGBM	0,9887	0,9809	0,9835	0,9796	0,9729	0,9734	0,9700	0,9862
XGBoost	0,9907	0,9989	0,9885	0,9937	0,9759	0,9761	0,9900	0,9996
SVM	0,9709	0,9594	0,9732	0,9663	0,8762	0,8765	0,9000	0,9802

Fonte: Elaborado pelo autor.

O modelo XGBoost obteve o melhor desempenho, tendo a melhor performance em sete das oito métricas. De modo geral, todos os modelos performaram muito bem, com valores

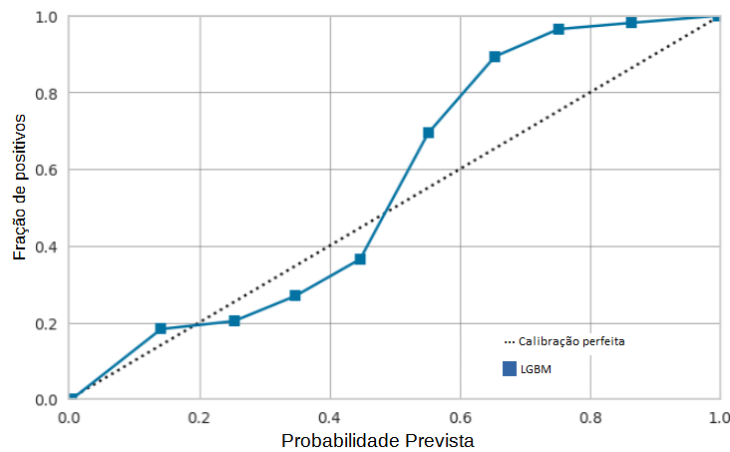
acima dos estabelecidos para todas as métricas. Como os dados utilizados para o teste não foram vistos pelos modelos durante o treinamento, é possível concluir que houve uma boa classificação das equações, o que afasta a possibilidade de *overfitting*. Na sequência foram avaliadas as curvas de calibração dos modelos, apresentadas nas Figuras 5.22, 5.23, 5.24 e 5.25.

FIGURA 5.22 – Curva de calibração do modelo *Gradient Boosting* - NICER



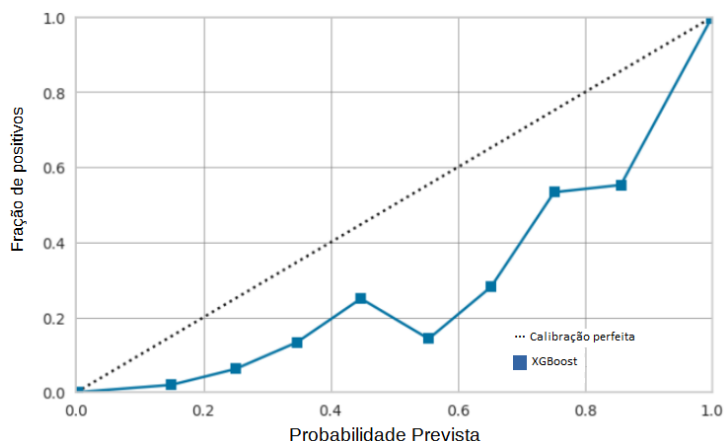
Fonte: Elaborado pelo autor.

FIGURA 5.23 – Curva de calibração do modelo LGBM - NICER



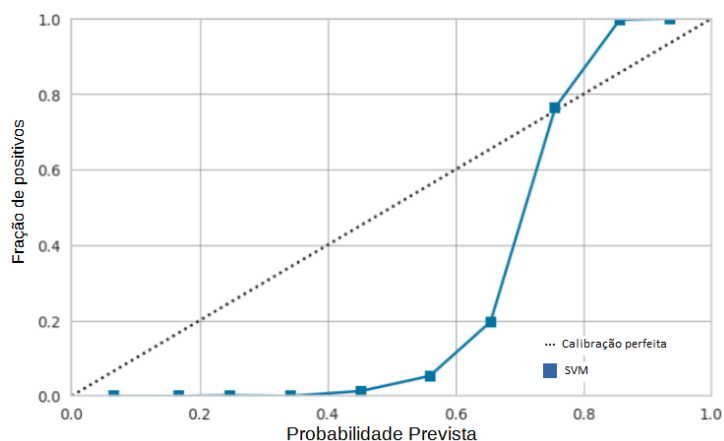
Fonte: Elaborado pelo autor.

FIGURA 5.24 – Curva de calibração do modelo XGBoost - NICER



Fonte: Elaborado pelo autor.

FIGURA 5.25 – Curva de calibração do modelo SVM - NICER

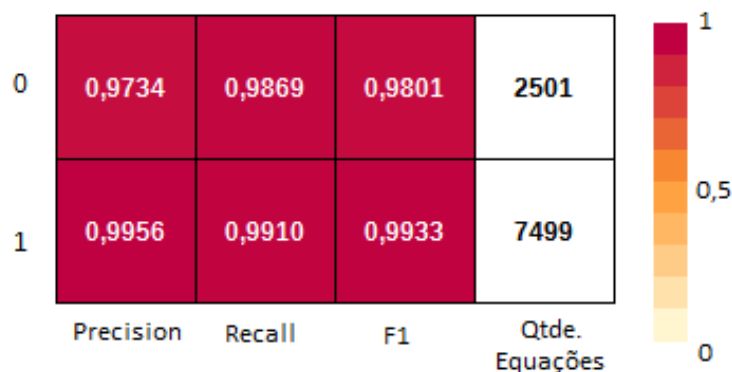


Fonte: Elaborado pelo autor.

O comportamento das curvas foi muito similar aos dos modelos LIGO, com destaque para os modelos *Gradient Boosting* e LGBM. O modelo XGBoost, apesar de obter os melhores valores para as métricas, apresentou uma curva de calibração mais descalibrada se comparada com os modelos GB e LGBM. Como os valores para as métricas não foram muito discrepantes entre os modelos, e devido a curva de calibração do modelo *gradient boosting* ter apresentando um comportamento próximo de uma calibração ideal, é seguro afirmar que ele foi o modelo superior. Logo, para o caso onde se deseja identificar as equações de estado que geram curvas que passam pela região do NICER, o modelo *gradient boosting* foi o que obteve a melhor performance geral. O modelo LGBM também apresentou valores acima dos desejados para todas as métricas, tanto em treino como em teste, conforme estabelecido no Capítulo 4, e também obteve uma curva de calibração aceitável. Os demais modelos apresentaram bons resultados para as métricas, porém pos-

suíam curvas de calibração descalibradas, o que afeta a confiança do modelo. A Figura 5.26 resume a performance geral do modelo *gradient boosting*.

FIGURA 5.26 – Performance geral do modelo *Gradient Boosting* para o caso NICER



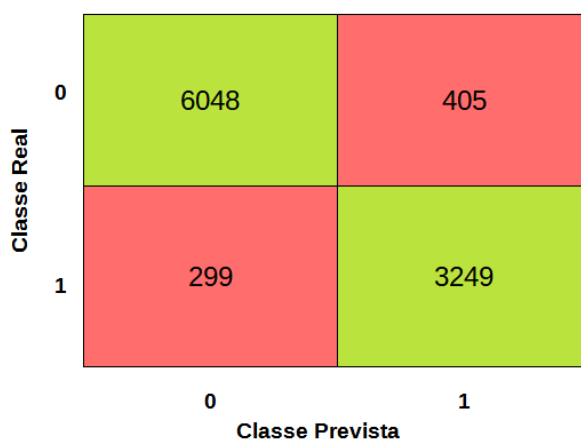
Fonte: Elaborado pelo autor.

Dentre todas as 10.000 equações utilizadas durante o teste, 25% pertenciam à classe 0 (não passam pela região NICER), enquanto 75% pertenciam à classe 1 (passam pela região NICER). O modelo apresentou um ótimo desempenho tanto para classificar as equações da classe 0 quanto da classe 1.

5.3 Modelos para as regiões LIGO e NICER

Os modelos discutidos a seguir foram treinados para classificar as curvas considerando se elas passam (classe 1) ou não (classe 0) nas regiões observacionais do NICER e LIGO simultaneamente. Iniciando pela *confusion matrix*, as Figuras 5.27, 5.28, 5.29 e 5.30 apresentam as matrizes para os quatro modelos treinados.

FIGURA 5.27 – *Confusion Matrix* do modelo *Gradient Boosting* - LIGO e NICER



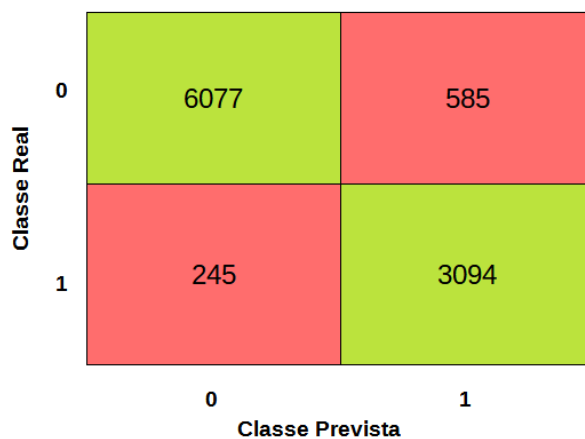
Fonte: Elaborado pelo autor.

FIGURA 5.28 – *Confusion Matrix* do modelo LGBM - LIGO e NICER

A 2x2 confusion matrix for the LGBM model. The y-axis is labeled 'Classe Real' with values 0 and 1. The x-axis is labeled 'Classe Prevista' with values 0 and 1. The cells contain the following counts: (0,0) is 6029, (0,1) is 445, (1,0) is 311, and (1,1) is 3215.

Classe Real \ Classe Prevista	0	1
0	6029	445
1	311	3215

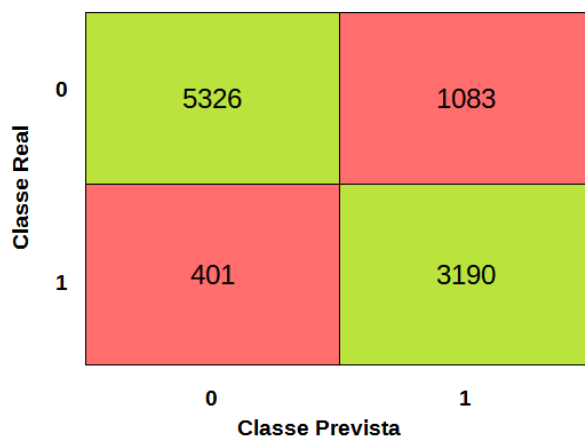
Fonte: Elaborado pelo autor.

FIGURA 5.29 – *Confusion Matrix* do modelo XGBoost - LIGO e NICER

A 2x2 confusion matrix for the XGBoost model. The y-axis is labeled 'Classe Real' with values 0 and 1. The x-axis is labeled 'Classe Prevista' with values 0 and 1. The cells contain the following counts: (0,0) is 6077, (0,1) is 585, (1,0) is 245, and (1,1) is 3094.

Classe Real \ Classe Prevista	0	1
0	6077	585
1	245	3094

Fonte: Elaborado pelo autor.

FIGURA 5.30 – *Confusion Matrix* do modelo SVM - LIGO e NICER

A 2x2 confusion matrix for the SVM model. The y-axis is labeled 'Classe Real' with values 0 and 1. The x-axis is labeled 'Classe Prevista' with values 0 and 1. The cells contain the following counts: (0,0) is 5326, (0,1) is 1083, (1,0) is 401, and (1,1) is 3190.

Classe Real \ Classe Prevista	0	1
0	5326	1083
1	401	3190

Fonte: Elaborado pelo autor.

É imediato observar que, neste caso, há muito mais equações da classe 0 do que da classe 1, o que é compreensível quando se leva em consideração que as curvas M-R devem passar em duas regiões simultaneamente para serem consideradas da classe 1. Apesar disto, de modo geral os modelos conseguiram acertar de forma satisfatória a classe das equações, com exceção do modelo SVM que classificou incorretamente uma quantidade considerável das equações da classe 0. As Tabelas 5.15, 5.16, 5.17 e 5.18 apresentam os resultados obtidos para a validação cruzada de cada um dos modelos.

TABELA 5.15 – *Cross Validation* do modelo *Gradient Boosting* - LIGO e NICER

Fold	Acurácia	Revocação	Precisão	F1	Kappa	MCC	KS	Gini
0	0,9231	0,9062	0,8841	0,8950	0,8344	0,8345	0,8400	0,8876
1	0,9323	0,9228	0,8936	0,9080	0,8545	0,8548	0,8600	0,8974
2	0,9282	0,9252	0,8822	0,9032	0,8462	0,8468	0,8600	0,9042
3	0,9250	0,9086	0,8870	0,8976	0,8385	0,8386	0,8500	0,8820
4	0,9323	0,9157	0,8991	0,9073	0,8540	0,8541	0,8600	0,9008

Fonte: Elaborado pelo autor.

TABELA 5.16 – *Cross Validation* do modelo LGBM - LIGO e NICER

Fold	Acurácia	Revocação	Precisão	F1	Kappa	MCC	KS	Gini
0	0,9255	0,9066	0,8914	0,8989	0,8399	0,8400	0,8500	0,8908
1	0,9248	0,9207	0,8793	0,8995	0,8395	0,8401	0,8500	0,8966
2	0,9207	0,9219	0,8693	0,8948	0,8313	0,8322	0,8500	0,8958
3	0,9184	0,9119	0,8710	0,8910	0,8258	0,8264	0,8400	0,8918
4	0,9183	0,8972	0,8816	0,8893	0,8247	0,8247	0,8300	0,8790

Fonte: Elaborado pelo autor.

TABELA 5.17 – *Cross Validation* do modelo XGBoost - LIGO e NICER

Fold	Acurácia	Revocação	Precisão	F1	Kappa	MCC	KS	Gini
0	0,9072	0,9273	0,8356	0,8791	0,8041	0,8070	0,8840	0,8808
1	0,9141	0,9256	0,8512	0,8869	0,8178	0,8197	0,8850	0,9018
2	0,9115	0,9286	0,8439	0,8842	0,8128	0,8153	0,8840	0,8918
3	0,9033	0,9256	0,8288	0,8745	0,7963	0,7995	0,8840	0,8840
4	0,9119	0,9262	0,8462	0,8844	0,8135	0,8156	0,8840	0,8878

Fonte: Elaborado pelo autor.

TABELA 5.18 – *Cross Validation* do modelo SVM - LIGO e NICER

Fold	Acurácia	Revocação	Precisão	F1	Kappa	MCC	KS	Gini
0	0,8498	0,8878	0,7471	0,8114	0,6882	0,6953	0,7300	0,8348
1	0,8507	0,8813	0,7514	0,8112	0,6890	0,6950	0,7300	0,8256
2	0,8524	0,8943	0,7489	0,8152	0,6939	0,7015	0,7300	0,8338
3	0,8410	0,8878	0,7321	0,8025	0,6715	0,6802	0,7200	0,8242
4	0,8539	0,8907	0,7529	0,8160	0,6963	0,7030	0,7400	0,8410

Fonte: Elaborado pelo autor.

Já as Tabelas 5.19 e 5.20 comparam respectivamente a média e o desvio padrão dos modelos para cada uma das métricas, e destaca aqueles que melhor desempenharam em cada métrica.

TABELA 5.19 – Comparação da média para os modelos das regiões LIGO e NICER

Modelo	Acurácia	Recall	Precision	F1	Kappa	MCC	KS	Gini
GB	0,9282	0,9157	0,8892	0,9022	0,8455	0,8458	0,8540	0,8944
LGBM	0,9215	0,9117	0,8785	0,8947	0,8322	0,8327	0,8440	0,8908
XGBoost	0,9096	0,9267	0,8411	0,8818	0,8089	0,8114	0,8842	0,8892
SVM	0,8496	0,8884	0,7465	0,8113	0,6878	0,6950	0,7300	0,8319

Fonte: Elaborado pelo autor.

TABELA 5.20 – Comparação do desvio padrão para os modelos das regiões LIGO e NICER

Modelo	Acurácia	Recall	Precision	F1	Kappa	MCC	KS	Gini
GB	0,0042	0,0084	0,0070	0,0058	0,0090	0,0091	0,0089	0,0093
LGBM	0,0034	0,0103	0,0089	0,0046	0,0073	0,0073	0,0089	0,0071
XGBoost	0,0043	0,0013	0,0089	0,0050	0,0086	0,0081	0,0004	0,0081
SVM	0,0050	0,0048	0,0083	0,0054	0,0097	0,0090	0,0071	0,0070

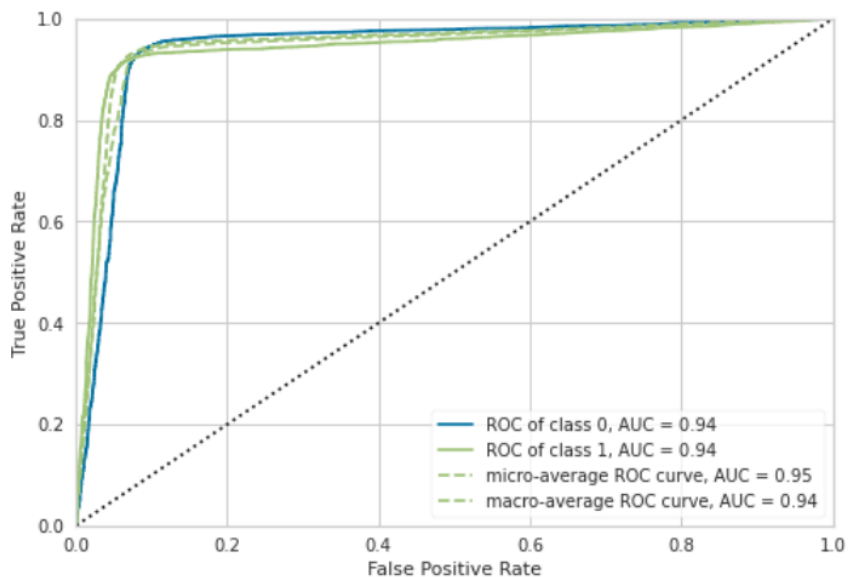
Fonte: Elaborado pelo autor.

O modelo *gradient boosting* performou melhor em cinco das oito métricas em relação à média, e em uma métrica em relação ao desvio padrão. Todos os valores ficaram acima dos desejados, com exceção do modelo SVM que obteve por exemplo um índice KS abaixo de 0,8. É importante reparar que os valores foram abaixo dos obtidos pelos modelos nas outras situações, quando estavam prevendo a passagem das curvas M-R apenas nas regiões LIGO e NICER separadamente. Os índices inferiores aos demais modelos pode ser reflexo da complexidade do caso em questão, onde deve-se detectar a passagem das curvas nas

duas regiões simultaneamente. Ainda assim os valores obtidos são acima dos desejados, o que demonstram que os modelos foram satisfatórios em sua performance durante o treino.

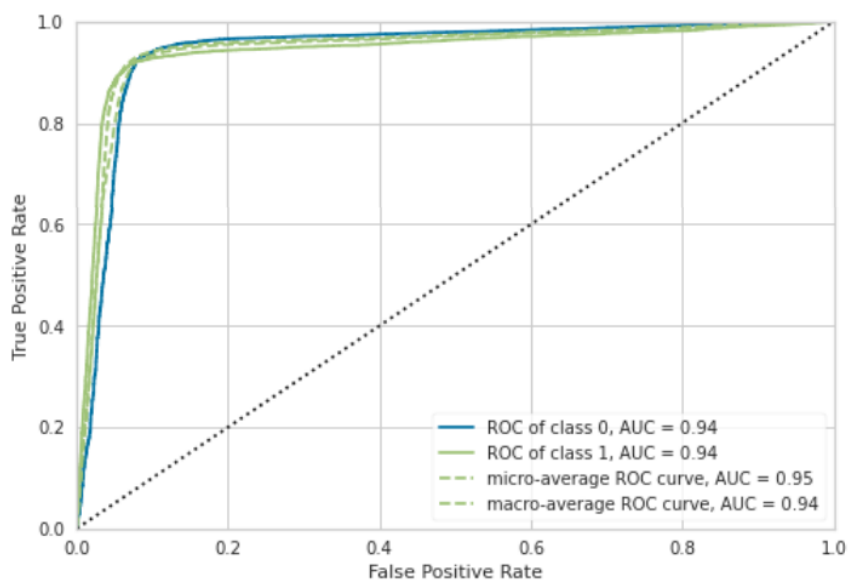
Para as curvas ROC obteve-se os resultados mostrados nas Figuras 5.31, 5.32, 5.33 e 5.34.

FIGURA 5.31 – Curva ROC do modelo *Gradient Boosting* - LIGO e NICER



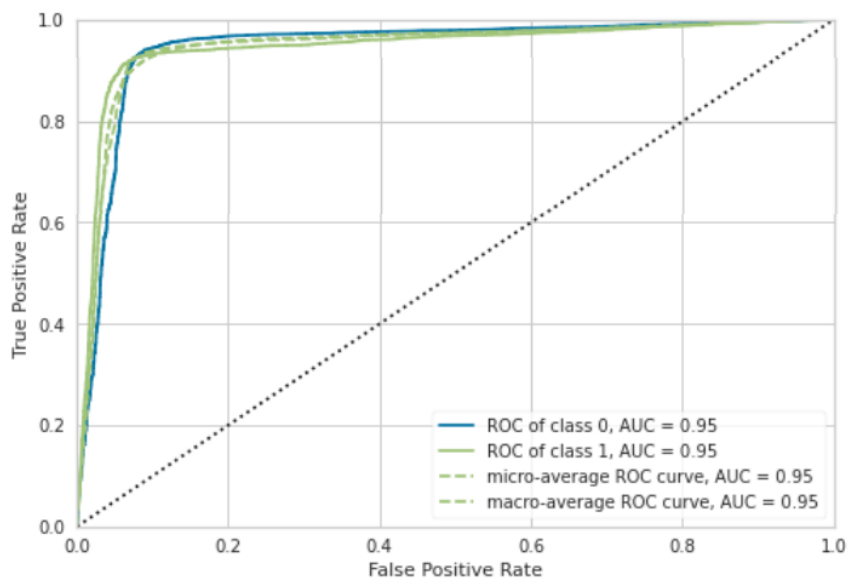
Fonte: Elaborado pelo autor.

FIGURA 5.32 – Curva ROC do modelo LGBM - LIGO e NICER



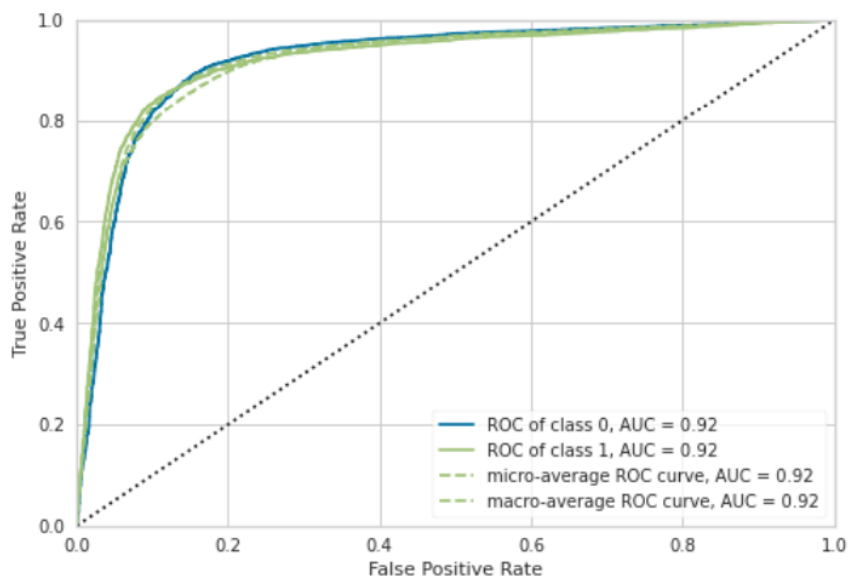
Fonte: Elaborado pelo autor.

FIGURA 5.33 – Curva ROC do modelo XGBoost - LIGO e NICER



Fonte: Elaborado pelo autor.

FIGURA 5.34 – Curva ROC do modelo SVM - LIGO e NICER



Fonte: Elaborado pelo autor.

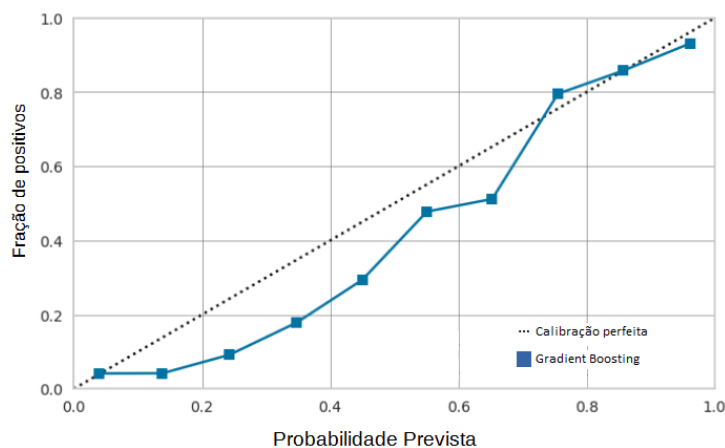
Analisando apenas as curvas não se observa uma situação de possível *overfitting* para nenhum dos modelos. Ainda assim é necessário avaliar suas performances em teste para que seja possível tirar alguma conclusão. Com as equações de teste, os modelos obtiveram os resultados mostrados na Tabela 5.21.

TABELA 5.21 – Comparação dos modelos com os dados de teste - LIGO e NICER

Modelo	Acurácia	Recall	Precision	F1	Kappa	MCC	KS	Gini
GB	0,9289	0,9106	0,8946	0,9025	0,8465	0,8466	0,8500	0,8896
LGBM	0,9013	0,9115	0,8867	0,8845	0,8308	0,8234	0,8400	0,8980
XGBoost	0,9157	0,9266	0,8531	0,8883	0,8209	0,8227	0,8400	0,8904
SVM	0,8547	0,8900	0,7532	0,8159	0,6973	0,7039	0,7400	0,8374

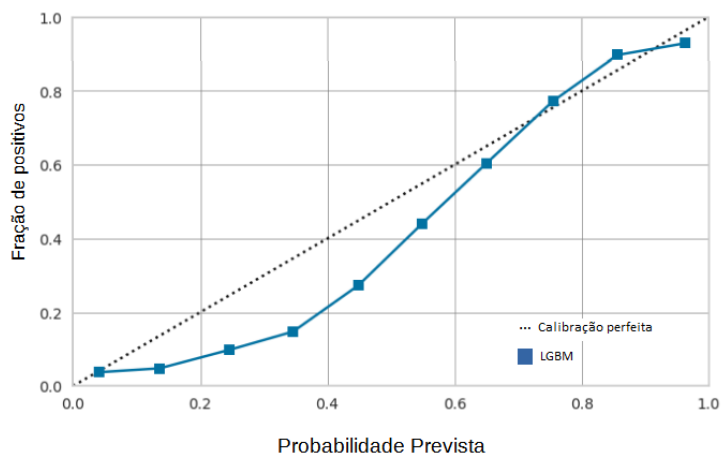
Fonte: Elaborado pelo autor.

Confirmando os resultados obtidos com os dados de treino, o modelo *gradient boosting* foi o que melhor performou, obtendo os melhores resultados para seis das oito métricas. De modo geral todos os modelos obtiveram desempenhos satisfatórios, com o modelo SVM tendo os resultados menos expressivos. Restava avilar as curvas de calibração, apresentadas nas Figuras 5.35, 5.36, 5.37 e 5.38.

FIGURA 5.35 – Curva de calibração do modelo *Gradient Boosting* - LIGO e NICER

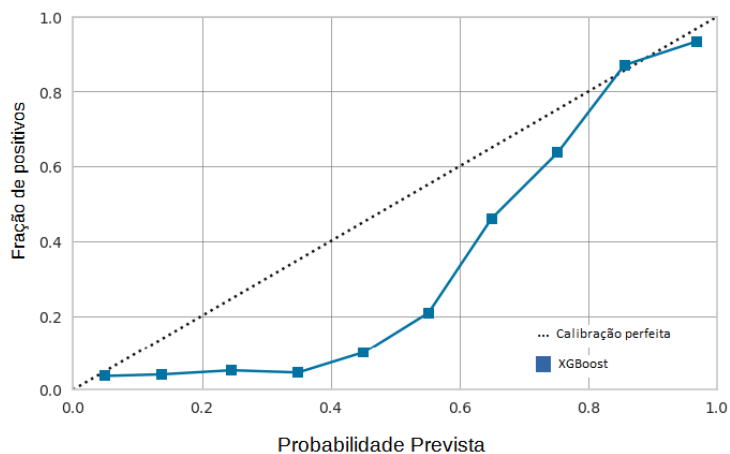
Fonte: Elaborado pelo autor.

FIGURA 5.36 – Curva de calibração do modelo LGBM - LIGO e NICER



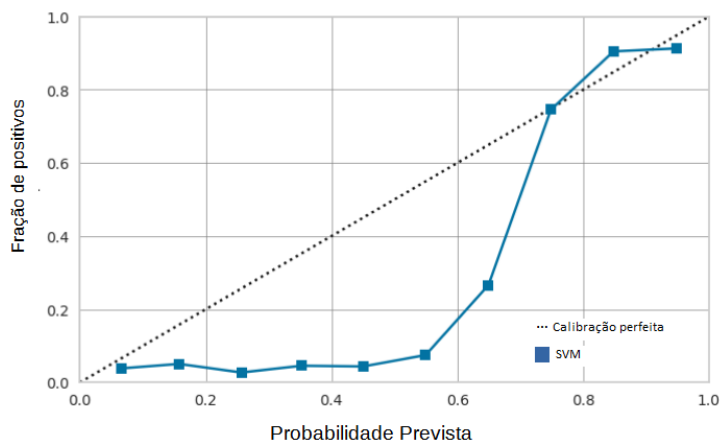
Fonte: Elaborado pelo autor.

FIGURA 5.37 – Curva de calibração do modelo XGBoost - LIGO e NICER



Fonte: Elaborado pelo autor.

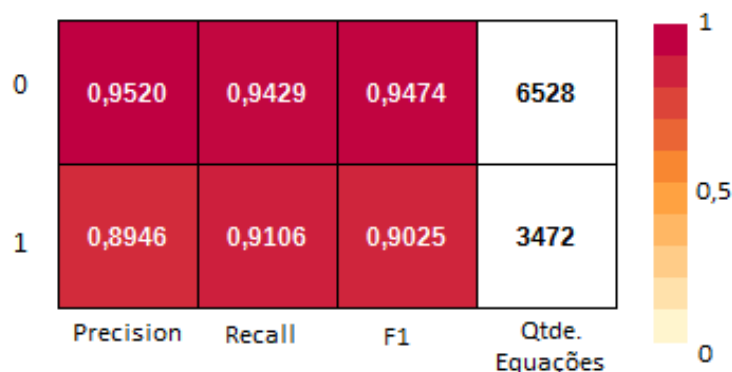
FIGURA 5.38 – Curva de calibração do modelo SVM - LIGO e NICER



Fonte: Elaborado pelo autor.

Repetindo o desempenho dos modelos para a região LIGO e para a região NICER, para as duas regiões simultaneamente os modelos que obtiveram as melhores curvas de calibração foram o *gradient boosting* e LGMB, não havendo uma diferença tão grande entre ambas. Os valores das métricas também foram semelhantes entre os dois modelos, com um maior destaque para o modelo *gradient boosting* que obteve maiores valores para oito das seis métricas. Logo, para o caso onde se deseja identificar as equações de estado que geram curvas M-R que passam simultaneamente pela região do LIGO e do NICER, o modelo *gradient boosting* foi o que obteve a melhor performance geral. Por mais que seu desempenho tenha sido levemente inferior aos modelos para as regiões LIGO e NICER separadamente, todos os valores estavam acima dos desejados e estabelecidos no Capítulo 4, além de possuir uma boa curva de calibração. Todos os demais modelos também apresentaram bons resultados (até mesmo o modelo SVM que ficou abaixo dos demais), porém alguns apresentaram curvas de calibração levemente descalibradas, o que afeta a confiança do modelo. A Figura 5.39 resume a performance do modelo *gradient boosting*.

FIGURA 5.39 – Performance geral do modelo *Gradient Boosting* para o caso LIGO e NICER



Fonte: Elaborado pelo autor.

Dentre todas as 10.000 equações utilizadas durante o teste, 65% pertenciam à classe 0 (não passam pelas regiões LIGO e NICER simultaneamente), enquanto 35% pertenciam à classe 1 (passam pelas regiões LIGO e NICER simultaneamente). O modelo apresentou um ótimo desempenho tanto para classificar as equações da classe 0 quanto da classe 1.

Em resumo, dos quatro modelos treinados para cada um dos três casos de interesse, um deles se saiu superior aos demais. Para o caso onde se deseja identificar as equações de estado que geram curvas M-R que passam região do LIGO, a melhor performance foi do modelo LGMB, para a região do NICER foi o modelo *gradient boosting*, e para ambas regiões simultaneamente também foi o modelo *gradient boosting*. Os demais modelos, por mais que não tenham tido o melhor desempenho entre todos os avaliados, também apresentaram bons resultados quando observadas as métricas de avaliação analisadas.

6 Conclusão

Este trabalho foi dedicado à exploração das capacidades dos modelos de *machine learning* na classificação de equações de estado de estrelas de nêutrons, com base nas regiões observacionais que se alinham com suas respectivas curvas massa-raio. Com os modelos devidamente treinados, uma nova equação de estado pode ser apresentada e como resultado se obtém a classificação da sua respectiva curva M-R entre duas possibilidades: se ela passa ou não pelas regiões observacionais de interesse. A aplicação bem sucedida desses modelos em um contexto tão complexo como esse é, sem dúvida, um marco no campo da astrofísica e da ciência de dados.

Os resultados alcançados são notáveis. Com uma taxa de acurácia superior a 95%, os modelos demonstraram a capacidade de realizar classificações confiáveis. Esses resultados são particularmente empolgantes, uma vez que a natureza das estrelas de nêutrons e suas equações de estado são fundamentais para a compreensão de eventos cósmicos extremos, como supernovas e fusões de estrelas de nêutrons. Portanto, a aplicação de técnicas de *machine learning* para classificar essas equações de estado pode ter implicações profundas na astrofísica e na física nuclear.

Os modelos LGBM e *gradient boosting* se destacaram, obtendo resultados acima do desejado para as regiões observacionais do NICER e LIGO, bem como para ambas as regiões simultaneamente. Esses resultados reforçam a versatilidade dessas técnicas e destacam a importância de selecionar abordagens de *machine learning* adequadas para problemas específicos. A curva de calibração próxima do ideal desses modelos também indica uma confiabilidade promissora em suas previsões.

Além dos resultados significativos, este estudo também aponta para direções promissoras para pesquisas futuras, já que à medida que avançamos, é importante reconhecer algumas áreas de melhoria e considerar direções futuras para a pesquisa. Primeiramente, embora os modelos tenham alcançado taxas de acurácia notáveis, deve-se continuar a aprimorar a compreensão das incertezas associadas às classificações. A astrofísica é uma ciência na qual a incerteza desempenha um papel crucial, e a capacidade de quantificar e comunicar essas incertezas é de importância crítica. Portanto, futuros estudos podem se concentrar em desenvolver métodos que permitam a estimativa e a interpretação das

incertezas nas previsões de equações de estado.

Outro ponto de melhoria é a exploração de técnicas mais avançadas de aprendizado de máquina, como redes neurais profundas (*deep learning*). Embora os modelos atuais tenham alcançado resultados satisfatórios, as redes neurais profundas têm a capacidade de capturar relações complexas e não lineares em dados, o que pode ser vantajoso para problemas astrofísicos. A investigação de abordagens de *deep learning* pode ser um próximo passo para melhorar ainda mais os modelos de classificação apresentados neste trabalho. Além disso, considerando a expansão contínua dos observatórios espaciais e terrestres, a disponibilidade de dados astronômicos está em constante crescimento. Portanto, a inclusão de mais fontes de dados e um conjunto de dados mais abrangente podem aprimorar a capacidade dos modelos de *machine learning* em classificar equações de estado com precisão.

Esta dissertação representa um passo significativo no uso de *machine learning* para a compreensão das equações de estado de estrelas de nêutrons, assim como abre portas para a aplicação de técnicas semelhantes em outros ramos da astrofísica e da ciência espacial. A utilização de *machine learning* na identificação de estrelas variáveis, na análise de dados de exoplanetas ou na detecção de eventos astrofísicos extraordinários pode ser um campo fértil para futuras pesquisas interdisciplinares. Os resultados obtidos apontam para diversas possibilidades e abrem caminho para uma pesquisa futura rica e diversificada, embora é de se reconhecer que há necessidade contínua de inovação e colaboração na pesquisa astrofísica. À medida que se avança na interseção da astrofísica e da ciência de dados, espera-se que este trabalho contribua para uma compreensão mais profunda das maravilhas do cosmos.

Referências

- ABBOTT, B. P.; ABBOTT, R.; ABBOTT, T. D.; ACERNESE, F.; ACKLEY, K. Gw170817: Measurements of neutron star radii and equation of state. **Physical Review Letters**, v. 121, 2018.
- AGRAWAL, A.; GANS, J.; GOLDFARB, A. **Máquinas Preditivas: a simples economia da inteligência artificial**. [S.l.]: Alta Books, 2020. 272 p.
- ALPAYDIN, E. **Introduction to machine learning**. [S.l.]: MIT Press, 2020.
- BAADE, W.; ZWICKY, F. Supernovae and cosmic rays. **Physical Review**, v. 46, n. 1, p. 76–77, 1933.
- BREIMAN, L. Random forests. **Machine Learning**, 2001.
- BREIMAN, L.; FRIEDMAN, J.; STONE, C. J.; OLSHEN, R. A. **Classification and Regression Trees**. [S.l.: s.n.], 1984.
- BUDU, E. **Random Forest vs. Extremely Randomized Trees**. 2020. Disponível em: <<https://www.baeldung.com/cs/random-forest-vs-extremely-randomized-trees>>. Acessado em: 23 de Março de 2023.
- CARROLL, S. M. **Spacetime and Geometry: An Introduction to General Relativity**. University of Chicago: Addison Wesley, 2004.
- CHADWICK, J. On the possible existence of a neutron. **Nature**, v. 129, n. 312, 1932.
- CHAWLA, N. V.; BOWYER, K. W.; HALL, L. O.; KEGELMEYER, W. P. **SMOTE: Synthetic Minority Over-sampling Technique**. [S.l.]: Journal of artificial intelligence research, 2002.
- CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. *In*: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Proceedings [...]**. [S.l.: s.n.], 2016. p. 785–794.
- CINELLI, M. **How Probability Calibration Works**. 2020. Disponível em: <<https://medium.com/analytics-vidhya/how-probability-calibration-works-a4ba3f73fd4d>>. Acessado em: 23 de Março de 2023.

- COPELAND, M. **What's the Difference Between Artificial Intelligence, Machine Learning and Deep Learning?** 2016. Disponível em: <<https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>>. Acessado em: 23 de Março de 2023.
- CRISTIANINI, N.; SHAWE-TAYLOR, J. **An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods**. [S.l.]: Cambridge University Press, 2000.
- DELSOLE, M. **What is One-Hot Encoding and How to Do It**. 2018. Disponível em: <<https://medium.com/@michaeldelsole/what-is-one-hot-encoding-and-how-to-do-it-f0ae272f1179>>. Acessado em: 26 de Março de 2023.
- DOUCHIN, F.; HAENSEL, P. A unified equation of state of dense matter and neutron star structure. **Astron. Astrophys.**, v. 380, 2001.
- DRAELOS, R. **Measuring Performance: The Confusion Matrix**. 2019. Disponível em: <<https://glassboxmedicine.com/2019/02/17/measuring-performance-the-confusion-matrix/>>. Acesso em: 26 de Março de 2023.
- EDX. **Undersampling in Data Science: What Is It and How Is It Used**. 2022. Disponível em: <<https://www.mastersindatascience.org/learning/statistics-data-science/undersampling/>>. Acessado em 26 de Março de 2023.
- FAWCETT, T. An introduction to roc analysis. **Pattern recognition letters**, Elsevier, v. 27, n. 8, p. 861–874, 2006.
- FRIEDMAN, J. H. Greedy function approximation: A gradient boosting machine. **The Annals of Statistics**, v. 29, n. 5, p. 1189–1232, 2001.
- FUJIMOTO, Y.; FUKUSHIMA, K.; MURASE, K. Extensive studies of the neutron star equation of state from the deep learning inference with the observational data augmentation. **Journal of High Energy Physics**, v. 2021, p. 273, 2021.
- GALARNYK, M. **Understanding Train Test Split**. 2022. Disponível em: <https://builtin.com/data-science/train-test-split>. Acessado em: 23 de Março de 2023.
- GEEKSFORGEEEKS. **Decision Tree - GeeksforGeeks**. 2023. Disponível em: <<https://www.geeksforgeeks.org/decision-tree/>>. Acessado em: 26 de Março de 2023.
- GEURTS, P.; ERNST, D.; WEHENKEL, L. Extremely randomized trees. **Machine Learning**, v. 63, n. 1, p. 3–42, 2006.
- GLENDENNING, N. K. **Compact Stars: Nuclear Physics, Particle Physics and General Relativity**. Estados Unidos: Springer Science Business Media, 2000. ISBN 9971509896.

- GOOGLE. **Normalization - Pré-processamento de dados** | Google Developers. 2022. Disponível em: <<https://developers.google.com/machine-learning/data-prep/transform/normalization?hl=pt-br>>. Acessado em 26 de Março de 2023.
- GUO, C.; PLEISS, G.; SUN, Y.; WEINBERGER, K. Q. On calibration of modern neural networks. **arXiv preprint arXiv:1706.04599**, 2017.
- GUSAROVA, M. **Understanding AUC — ROC and Precision-Recall Curves**. 2022. Disponível em: <<https://medium.com/@data.science.enthusiast/auc-roc-curve-ae9180eaf4f7>>. Acessado em: 23 de Março de 2023.
- HAENSEL, P.; POTEKHIN, A.; YAKOVLEV, D. **Neutron stars 1: Equation of state and structure**. New York, USA: Springer, 2007.
- HANLEY, J. A.; MCNEIL, B. J. The meaning and use of the area under a receiver operating characteristic (roc) curve. **Radiology**, Radiological Society of North America, v. 143, n. 1, p. 29–36, 1982.
- HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The elements of statistical learning: data mining, inference, and prediction**. [S.l.]: Springer Science & Business Media, 2009.
- HE, H.; GARCIA, E. **Learning from imbalanced data**. [S.l.]: IEEE Transactions on knowledge and data engineering, 2009.
- JAVATPOINT. **Machine Learning - Support Vector Machine Algorithm**. 2020. Disponível em: <<https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm>>. Acessado em: 23 de Março de 2023.
- JEFFARES, A. **Supervised vs Unsupervised Learning in 3 Minutes**. 2018. Disponível em: <<https://towardsdatascience.com/supervised-vs-unsupervised-learning-in-2-minutes-72dad148f242>>. Acessado em: 23 de Março de 2023.
- JR., D. W. H.; LEMESHOW, S.; STURDIVANT, R. X. **Applied logistic regression**. [S.l.]: John Wiley & Sons, 2013.
- KE, G.; MENG, Q.; FINLEY, T.; WANG, T.; CHEN, W.; MA, W.; YE, Q.; LIU, T. Y. Lightgbm: A highly efficient gradient boosting decision tree. *In: Proceedings of the 31st Conference on Neural Information Processing Systems. Proceedings [...]*. [S.l.: s.n.], 2017. p. 3146–3154.
- MICHALSKI, R. S.; CARBONELL, J. G.; MITCHELL, T. M. **Machine Learning: An Artificial Intelligence Approach**. [S.l.]: Morgan Kaufmann, 2014. 572 p.
- MILLER, M. C.; LAMB, F. K.; DITTMANN, A. J.; BOGDANOV, S.; ARZOUMANIAN, Z. Psr j0030+0451 mass and radius from nicer data and implications for the properties of neutron star matter. **The Astrophysical Journal Letters**, 2019.
- MILLER, M. C.; LAMB, F. K.; DITTMANN, A. J.; BOGDANOV, S.; ARZOUMANIAN, Z.; GENDREAU, K. C. The radius of psr j0740+6620 from nicer and xmm-newton data. **The Astrophysical Journal Letters**, 2021.

- MOLODORIA, A. **5 Essential Machine Learning Algorithms For Business Applications**. 2022. Disponível em: <<https://mobidev.biz/blog/essential-machine-learning-algorithms-for-business-applications>>. Acessado em: 26 de Março em 2023.
- MURPHY, K. P. **Machine Learning: A Probabilistic Perspective**. [S.l.]: MIT Press, 2012.
- OPPENHEIMER, J. R.; VOLKOFF, G. M. On massive neutron cores. **Physical Review**, v. 55, n. 4, p. 374–381, 1916.
- POWERS, D. M. W. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. **Journal of Machine Learning Technologies**, Cognition Research, v. 2, n. 1, p. 37–63, 2020.
- PROVOST, F.; FAWCETT, T. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. **Knowledge and Information Systems**, v. 3, n. 3, p. 43–66, 2001.
- QUINLAN, J. R. Induction of decision trees. **Machine learning**, Springer, v. 1, n. 1, p. 81–106, 1986.
- RASCHKA, S.; MIRJALILI, V. **Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2**. [S.l.]: Packt Publishing, 2019.
- RILEY, T. E.; WATTS, A. L.; BOGDANOV, S.; RAY, P. S.; LUDLAM, R. M.; GUILLOT, S.; ARZOUMANIAN, Z.; BAKER, C. L.; BILOUS, A. V.; CHAKRABARTY, D.; GENDREAU, K. C.; HARDING, A. K.; HO, W. C. G.; LATTIMER, J. M.; MORSINK, S. M.; STROHMAYER, T. E. A nicer view of psr j0030+0451: Millisecond pulsar parameter estimation. **The Astrophysical Journal Letters**, 2019.
- RILEY, T. E.; WATTS, A. L.; RAY, P. S.; BOGDANOV, S.; GUILLOT, S.; MORSINK, S. M.; BILOUS, A. V.; ARZOUMANIAN, Z.; CHOUDHURY, D.; DENEVA, J. S.; GENDREAU, K. C.; HARDING, A. K.; HO, W. C. G.; LATTIMER, J. M.; LOEWENSTEIN, M.; LUDLAM, R. M.; MARKWARDT, C. B.; OKAJIMA, T.; PRESCOD-WEINSTEIN, C.; REMILLARD, R. A.; WOLFF, M. T.; FONSECA, E.; CROMARTIE, H. T.; KERR, M.; PENNUCCI, T. T.; PARTHASARATHY, A.; RANSOM, S.; STAIRS, I.; GUILLEMOT, L.; COGNARD, I. A nicer view of the massive pulsar psr j0740+6620 informed by radio timing and xmm-newton spectroscopy. **The Astrophysical Journal Letters**, 2021.
- ROMANI, R. W.; KANDEL, D.; FILIPPENKO, A. V.; BRINK, T. G.; ZHENG, W. Psr j09520607: The fastest and heaviest known galactic neutron star. **The Astrophysical Journal Letters**, v. 934, 2022.
- ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, American Psychological Association, v. 65, n. 6, p. 386–408, 1958.

SABBATA, V. de; GASPERINI, M. **Introduction to Gravitation**. Singapura: World Scientific, 1985. ISBN 9971509896.

SAYAD, S. **Logistic Regression**. 2021. Disponível em: <https://www.saedsayad.com/logistic_regression.htm>. Acessado em: 26 de Março de 2023.

SCHWARZSCHILD, K. Über das gravitationsfeld eines massenpunktes nach der einsteinschen theorie (1916). **Sitzungsberichte der Kniglich Preussischen Akademie der Wissenschaften**, v. 1, p. 189–196, 1916.

SCIKIT-LEARN. **Cross-validation: evaluating estimator performance**. 2011. Disponível em: <https://scikit-learn.org/stable/modules/cross_validation.html>. Acessado em: 27 de Março de 2023.

SEO, Y.; HAN, Y.; LEE, T. Gini coefficient as a criterion for feature selection. **Expert Systems with Applications**, Elsevier, v. 154, p. 113374, 2020.

SHAPIRO, S. L.; TEUKOLSKY, S. A. **Black Holes, White Dwarfs, and Neutron Stars: The Physics of Compact Objects**. 3rd. ed. Nova Iorque: Wiley-Vch, 1983.

SHUKLA, P. **Top 10 Interview Questions on Gradient Boosting**. 2022. Disponível em:<<https://www.analyticsvidhya.com/blog/2022/11/top-10-interview-questions-on-gradient-boosting/>>. Acessado em: 23 de Março de 2023.

SIMON, P. **Too Big to Ignore: The Business Case for Big Data**. [*S.l.*]: Wiley, 2013. 89 p.

VARMA, S.; SIMON, R. Bias in error estimation when using cross-validation for model selection. **BMC Bioinformatics**, v. 7, n. 1, p. 91, 2006.

Apêndice A - Código para geração das equações de estado e obtenção das curvas M-R

```
# Imports
from random import random, seed
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mt
import numpy as np
import math
import os
import warnings
from tovsolver.tov import *
from tovsolver.constants import *
import multiprocessing as mp
warnings.filterwarnings('ignore')

# Gerador das velocidades do som no \\
intervalo delta < sv < 1-delta
from random import random, seed

def soundv(min,max):
    return min+(random()*(max-min))

# Interpolacao Politropica
def poli(n,k,gamma,epoints):
    if n<=epoints[1]:
```

```

        return k[1]*(n**gamma[1])
    elif n<=epoints[2]:
        return k[2]*(n**gamma[2])
    elif n<=epoints[3]:
        return k[3]*(n**gamma[3])
    elif n<=epoints[4]:
        return k[4]*(n**gamma[4])
    else:
        return k[5]*(n**gamma[5])

# Funcao para gerar as equacoes de estado
def equations(path,ei=150,ef=1200,nfaixas=5):

    # Lendo o arquivo da equacao inicial
    sly4=pd.read_csv('gs://bkt-upload-raw-data/sly4oficial.csv')
    X,Y = sly4.den_energy.values,sly4.press.values

    #Obtendo os valores para densidade de energia ate 150Mev/fm3
    sly4 = sly4.loc[0:523]

    # Obtendo os pontos de energia separados na escala \\
    #logaritmica
    epoints = np.logspace(np.log10(ei),np.log10(ef),nfaixas+1)
    epoints=[int(round(i,0)) for i in epoints]
    # Gerando os pontos de den. energia
    faixas_energia=np.linspace(ei,ef,500)

    #Atribuindo velocidades do som aleatorias para cada segmento
    crand=[soundv(0.1,0.99) for i in range(nfaixas)]
    velocidades=[0]
    pressoes=[np.interp(150,X,Y)]
    for i in range(len(crand)):
        velocidades.append(crand[i])

    #Gerando as pressoes de forma iterativa
    for i in range(1,nfaixas+1):
        p=pressoes[i-1]+velocidades[i]*(epoints[i]-epoints[i-1])
        pressoes.append(p)
    k=[0]

```

```

gamma=[0]

# Calculando os coeficientes para a politrotica
for i in range(1,nfaixas+1):
    g=(np.log(pressoes[i]/pressoes[i-1]))\\
    /(np.log(epoints[i]/epoints[i-1]))
    gamma.append(g)
    nk=pressoes[i]/(epoints[i]**gamma[i])
    k.append(nk)
faixas_pressao = np.array(list(map\\
(lambda p: poli(p,k,gamma,epoints),faixas_energia)))

# Salvando o arquivo .csv
d={'den_energy':faixas_energia,'press':faixas_pressao}
eq=pd.DataFrame(d)
eq=pd.concat([sly4,eq],ignore_index=True)
eq.to_csv(path,index=False)

#Rotina para resolver a TOV
def TOVSolver(paths):
    print("Calculando "+paths[1])
    eos=pd.read_csv(paths[0])
    X=eos.den_energy.values
    Y=eos.press.values

    tov_s = TOV(X, Y, plot_eos=False, add_crust=0)
    m_arr = []
    R_arr = []

    for dens_c in np.linspace(1.50e-08,1200,500):
        R, M, prof = tov_s.solve(dens_c, dr=100)
        m_arr.append(M)
        R_arr.append(R)

    MR = pd.DataFrame({'radius':R_arr,'mass':m_arr})
    MR.to_csv(paths[1],index=False)

if __name__ == "__main__":

```

```

#Loop para gerar e salvar diversas
equacoes de estado e curvas MR
n=10000 #Total de equacoes geradas
ei=150 # Densid. energia inicial
ef=1200 # Densid. energia final
faixas_energia=5
cpus = mp.cpu_count()
paths1=['gs://bkt-output\\
data/EOS/EOS'+str(j)+'.csv' for j in range(n)]
paths2=['gs://bkt-output-\\
data/MR/MR'+str(j)+'.csv' for j in range(n)]
with mp.Pool(cpus) as pool:
    print('Gerando '+str(n)+' Equacoes de Estado...')
    pool.map(equations, paths1)
    pool.map(TOVSolver, zip(paths1, paths2))
    
```

Apêndice B - Código para verificações das curvas M-R e organização dos dados

```
# Imports
from tqdm.auto import tqdm
import pandas as pd
import multiprocessing as mp
from multiprocessing.pool import ThreadPool as Pool
import warnings
import numpy as np
import os
import time
from shapely.geometry import Point, Polygon
warnings.filterwarnings('ignore')

def readPress(i):
#Leitura das pressoes
    global pressoes
    global epoints
    global params
    global energy_values
    path='gs://bkt-output-data/EOS_BPS/EOS'+str(i)+'.csv'
    eos=pd.read_csv(path)
    X=eos.den_energy.values
    Y=eos.press.values
    p=np.interp(energy_values,X,Y)
    pressoes.append((i,p))

def readMR(j):
#Leitura das curvas MR
```



```

global cond_mass_min
global cond_mass_max
global cond_raio_min
global cond_raio_max
global nicer030_riley
global nicer030_miler
global gw_m2
global gw_m1
global nicer740_miler
global nicer740_riley

path='gs://bkt-output-data/MR_BPS/MR'+str(j)+'_csv'

ms = pd.read_csv(path)

# Realizando a verificacao da passagem da curva M-R
# pelas regioes observacionais
nicer_verification = False
regiao_nicer1 = Polygon(zip(\
nicer030_riley.Radius, nicer030_riley.Mass))
regiao_nicer2 = Polygon(zip(\
nicer030_miler.Radius, nicer030_miler.Mass))
regiao_nicer3 = Polygon(zip(\
nicer740_miler.Radius, nicer740_miler.Mass))
regiao_nicer4 = Polygon(zip(\
nicer740_riley.Radius, nicer740_riley.Mass))

for i, j in zip(ms.radius, ms.mass):
    if regiao_nicer1.contains(Point((i, j))) or \
regiao_nicer2.contains(Point((i, j))) or \
regiao_nicer3.contains(Point((i, j))) or \
regiao_nicer4.contains(Point((i, j))):
        nicer_verification=True
        break

ligo_verification = False
regiao_ligo1 = Polygon(zip(gw_m1.Radius, gw_m1.Mass))
regiao_ligo2 = Polygon(zip(gw_m2.Radius, gw_m2.Mass))

```

```

for i ,j in zip(ms.radius ,ms.mass):
    if regioao_ligo1.contains(Point((i ,j))) or \
regioao_ligo2.contains(Point((i ,j))):
        ligo_verifiction = True
        break

nicer_ligo_verifiction = False
if nicer_verifiction and ligo_verifiction:
    nicer_ligo_verifiction = True

if nicer_verification:
    v=(j ,1)
else:
    v=(j ,0)
nicer.append(v)

if ligo_verification:
    v=(j ,1)
else:
    v=(j ,0)
ligo.append(v)

if nicer_ligo_verification:
    v=(j ,1)
else:
    v=(j ,0)
nicer_ligo.append(v)

# Verificando massa maxima
#(acabou n o sendo utilizado)
massas.append((j ,ms.mass.max()))

if __name__ == "__main__":
    start = time.time()

##### PARaMETROS PRINCIPAIS #####

energy_values = [i for i in range(200 ,1200 ,5)]
params = len(energy_values)

```

```

eqnumber = 1000
pressoes = []
cpus = mp.cpu_count()
massas = []
ligo = []
nicer = []
nicer_ligo = []
# Lendo os dados observacionais
path2 = 'gs://bkt-upload-raw-data/DadosArtigos'
nicer030_riley = pd.read_csv\\
(path2+"NICER-PSRJ0030+0451-2022-EPJWoC260-04001\\
-solid_line-Riley-90perc.csv")
nicer030_miler = pd.read_csv\\
(path2+"NICER-PSRJ0030+0451-2022-EPJWoC260-04001\\
-dashed_line-Miler-90perc.csv")
gw_m2 = pd.read_csv\\
(pat2h+"GW170817-rosa-PhysRevD.105.123004--ang-li.csv")
gw_m1 = pd.read_csv\\
(path2+"GW170817-violeta-PhysRevD.105.123004--ang-li.csv")
nicer740_miler = pd.read_csv(path2+"NICER-PSRJ0740+0030-\\
arXiv-2201.11767-Miler-90perc.csv")
nicer740_riley = pd.read_csv(path2+"NICER-PSRJ0740+0030-\\
arXiv-2201.11767-Riley-90perc.csv")

##### Paralelizar aqui #####
# Gerando e integrando as equacoes
# paralelamente

with Pool(cpus) as pool:
    print("\n Total de arquivos:", eqnumber*2)
    print("\n Lendo arquivos das EoS...")
    pool.map(readPress, range(eqnumber))
    print("\n Lendo arquivos das curvas M-R...")
    pool.map(readMR, range(eqnumber))
#####
#Vetores para diversas informacoes
#das equacoes

```

```

p_order = []
p_values = []
nicer_value = []
nicer_order = []
ligo_value = []
ligo_order = []
nicer_ligo_value = []
nicer_ligo_order = []
m_order = []
m_values = []
# Obtendo as leituras de pressao e suas ordens
print("\n Ordenando as leituras das pressoes...")
for (i, p) in pressoes:
    p_order.append(i)
    p_values.append(p)
# Obtendo as verificacoes e suas ordens
print("\n Ordenando as verificacoes das curvas M-R...")
for (i,v) in nicer:
    nicer_order.append(i)
    nicer_value.append(v)
for (i,v) in ligo:
    ligo_order.append(i)
    ligo_value.append(v)
for (i,v) in nicer_ligo:
    nicer_ligo_order.append(i)
    nicer_ligo_value.append(v)
# Obtendo as leituras da massa maxima e suas ordens
print("\n Ordenando as leituras das massas maximas...")
for (i, m) in massas:
    m_order.append(i)
    m_values.append(m)

# Ajeitando o array de pressoes para o dataframe
p_values=np.array(p_values).transpose()

print("\n Criando o Dataframe...")
# Criando o dataframe de pressoes e ordenando
colnames = ['p'+str(i) for i in range(params)]
dic = {i:j for (i,j) in zip(colnames, p_values)}

```

```

dic ['ordem'] = p_order
data_p = pd.DataFrame(dic)
data_p=data_p.sort_values\\
(by='ordem', ascending = 1)[colnames]

# Criando o dataframe de verificacoes e ordenando
dic={'ordem':nicer_order,'NICER':nicer_value}
data_nicer=pd.DataFrame(dic)
data_nicer=data_nicer.sort_values(by='ordem', ascending = 1)
data_nicer.reset_index(inplace=True, drop = True)
data_nicer=data_nicer.NICER

dic={'ordem':ligo_order,'LIGO':ligo_value}
data_ligo=pd.DataFrame(dic)
data_ligo=data_ligo.sort_values(by='ordem', ascending = 1)
data_ligo.reset_index(inplace=True, drop = True)
data_ligo=data_ligo.LIGO

dic={'ordem':nicer_ligo_order,\\
'NICER-LIGO':nicer_ligo_value}
data_nicer_ligo=pd.DataFrame(dic)
data_nicer_ligo=data_nicer_ligo.sort_values\\
(by='ordem', ascending = 1)
data_nicer_ligo.reset_index(inplace=True, drop = True)
data_nicer_ligo=[ 'NICER-LIGO']

# Criando o dataframe de massas maximas e ordenando
dic = {'ordem':m_order,'massa_max':m_values}
data_m = pd.DataFrame(dic)
data_m=data_m.sort_values(by='ordem', ascending = 1)
data_m.reset_index(inplace=True, drop = True)
data_m=data_m.massa_max

#Unindo os datasets
arquivo="gs://bkt-output-data/eos_dataframe_BPS.csv"
data2=pd.concat([data_p,data_nicer,\\
data_ligo,data_nicer_ligo,data_m], axis = 1)
data2.to_csv(arquivo,index = False)
print("\\n Arquivo ",arquivo," salvo em ", os.getcwd())

```

```
end=time.time()  
print("\n Duracao:",round(end-start,2),"s.")
```

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO DM	2. DATA 31 de janeiro de 2024	3. DOCUMENTO Nº DCTA/ITA/DM-144/2023	4. Nº DE PÁGINAS 102
5. TÍTULO E SUBTÍTULO: Uso de <i>Machine Learning</i> para Classificação de Equações de Estado de Estrelas de Nêutrons			
6. AUTOR(ES): Gabriel Coelho Teles de Moura			
7. INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES): Instituto Tecnológico de Aeronáutica – ITA			
8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR: 1. Machine learning 2. Equações de estado 3. Estrelas de nêutrons			
9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO: Estrelas de nêutrons; Equações de estado; Aprendizagem (inteligência artificial); Algoritmos; Astrofísica; Física.			
10. APRESENTAÇÃO: (X) Nacional () Internacional TA, São José dos Campos. Curso de Mestrado. Programa de Pós-Graduação em Física. Área de Física Nuclear. Orientador: Prof. Dr. César Henrique Lenzi; coorientadora: Prof ^ª . Dr ^ª . Nadja Simão Magalhães. Defesa em 13/11/2023. Publicada em 2023.			
11. RESUMO: Esta dissertação concentra-se na avaliação da eficácia de quatro algoritmos de classificação (<i>Gradient Boosting</i> , <i>Extreme Gradient Boosting</i> , <i>Light Gradient Boosting</i> e <i>Support Vector Machines</i>) na categorização de equações de estado de estrelas de nêutrons. O objetivo primordial é realizar essa categorização com base nas curvas massa-raio que coincidem com regiões observacionais específicas, nomeadamente as do NICER, LIGO, bem como aquelas que simultaneamente abrangem ambas as regiões. Como resultado deste estudo, foram desenvolvidos 12 modelos de classificação. Observou-se que o melhor modelo para a região LIGO foi o LGBM, enquanto para a região NICER e em ambas as regiões simultaneamente, o <i>gradient boosting</i> demonstrou ser a escolha mais eficaz. Esses modelos proporcionaram uma análise objetiva e imparcial das capacidades desses algoritmos no contexto altamente especializado das estrelas de nêutrons. Palavras-chave: estrelas de nêutrons, equações de estado, aprendizado de máquina, classificação, curvas massa-raio, astrofísica.			
12. GRAU DE SIGILO: (X) OSTENSIVO () RESERVADO () SECRETO			