Universidade de São Paulo
Instituto de Astronomia, Geofísica e Ciências Atmosféricas
Departamento de Astronomia

Marcele Pâmela Santos Nunes

Inteligência Artificial Explicável: aplicação à classificação de fontes astronômicas

São Paulo
2024

Marcele Pâmela Santos Nunes

Inteligência Artificial Explicável: aplicação à classificação de fontes astronômicas

Trabalho de Conclusão de Curso apresentado ao Departamento de Astronomia do Instituto de Astronomia, Geofísica e Ciências Atmosféricas da Universidade de São Paulo como requisito parcial para a obtenção do título de Bacharel em Astronomia.

Vertente: Pesquisa Básica
Orientador: Prof. Dr. Laerte Sodré Junior

São Paulo
2024

À minha família e aos meus amigos.

Agradecimentos

À minha mãe Márcia, minha tia Dita e aos meus irmãos Bruna e João Vitor.

Aos meus amigos de longa data Robson e Diego.

Ao meu orientador Prof. Laerte Sodre, pela confiança, paciência e apoio.

A todos os professores que tive nos anos de escola.

Aos professores do IAG;

À USP pelo apoio.

Aos companheiros de curso do IAG.

Aos meus amigos do CRUSP.

Esta tese/dissertação foi escrita em LaTeX com a classe IAGTESE, para teses e dissertações do IAG.

"*The cosmos is also within us, we're made of star-stuff. We are a way for the cosmos to know itself.*"

Carl Sagan

Resumo

O objetivo deste trabalho é verificar como cada filtro fotométrico influencia na classificação de objetos astronômicos, galáxias e estrelas, com base em suas características estritamente fotométricas. A amostra, obtida a partir do Shouthern Photometric Local Universe Survey (S-PLUS), consiste de 12 bandas fotométricas e a classificação para 10 mil objetos. Algoritmos de *Machine Learning* (ML), a saber, *Random Forest*, *XGBoost* e *Regressão Logística* foram aplicados aos dados. Foi realizado o uso de *K-Fold Cross Validation* (validação cruzada) para a proporção de 80% da amostra como treinamento e 20% como conjunto de teste, dividindo a base em 5 subamostras (*folds*). Sob o ponto de vista da acurácia dos modelos, essa métrica é máxima utilizando o algoritmo de XGBoost com 96,28%, enquanto que a mínima é obtida para Regressão Logística, de 86,21%. Como resultado, os valores SHAP para XGBoost mostraram que os filtros J0410_petro e r_petro mais auxiliaram o modelo a caracterizar as fontes astronômicas em suas respectivas classes, enquanto que a banda i_petro se mostrou menos relevante. É possível também realizar algumas diferentes abordagens para justificar os resultados aqui obtidos. Este estudo contribui para uma compreensão sobre como os diferentes comprimentos de onda do espectro visível atuam na separação de objetos astronômicos, destacando o uso e a importância de modelos de Aprendizado de Máquina na Astronomia.

Abstract

The objective of this work is to examine how each photometric band influences the classification of astronomical objects, galaxies and stars, based on their strictly photometric characteristics. The sample, obtained from the Southern Photometric Local Universe Survey (S-PLUS), consists of 12 photometric bands and the classification of 10,000 objects. Machine Learning (ML) algorithms, namely Random Forest, XGBoost, and Logistic Regression, were applied to the data. The use of K-Fold Cross Validation was performed with 80% of the sample as training and 20% as the test set, dividing the data into 5 subsamples (folds). In terms of model accuracy, the highest metric was achieved using the XGBoost algorithm with 96.28%, while the lowest was obtained for Logistic Regression at 86.21%. As a result, the SHAP values for XGBoost showed that the filters J0410_petro and r_petro most helped the model characterize astronomical sources in their respective classes, while the i_petro band proved to be less relevant. It is also possible to perform different approaches to justify the results obtained here. This study contributes to the understanding of how different wavelengths of the visible spectrum act in the separation of astronomical objects, highlighting the use and importance of Machine Learning models in Astronomy.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Inteligência Artificial Explicável (ou *Explainable Artificial Intelligence - XAI*) é um campo de estudo que busca tornar modelos de Inteligência Artificial mais claros e de melhor entendimento pelos seres humanos. Ao invés de fornecer apenas uma resposta ou previsão, modelos explicáveis também se empenham em justificar ou detalhar o raciocínio por trás de suas decisões. Deve ser capaz de explicar o que fez, o que está fazendo agora e o que acontecerá a seguir (Bellotti e Edwards (2001), Gunning et al. (2019)).

Modelos complexos, como redes neurais profundas, que apresentam camadas ocultas ou algoritmos de emsemble[1] são bastante precisos. Porém a forma como funcionam são como "caixas-pretas", uma vez que interpretá-los é difícil e pode ser problemático em áreas críticas, como saúde e finanças, sendo fundamental entender os motivos que levaram o modelo a tomar determinada decisão.

A expressão "Inteligência Artificial" (IA) pode soar de forma intrigante e despertar curiosidade e até receio ao sugerir a ideia de uma máquina capaz de realizar atividades exclusivas dos seres humanos. Métodos de IA estão alcançando níveis de desempenho excepcional ao aprenderem a resolver tarefas computacionais cada vez mais sofisticadas. Atualmente, a evolução dos sistemas baseados em IA chegou a tal ponto que pouca ou nenhuma intervenção humana é requerida para seu desenvolvimento e implementação (Arrieta et al. (2020)).

Técnicas agnósticas de explicabilidade post-hoc (do latim "depois disso") são utilizadas para entender o comportamento de modelos de ML após o modelo ter sido treinado. O termo "agnósticas" significa que as técnicas podem ser aplicadas a qualquer tipo de modelo

[1] Métodos de aprendizado de máquina que combinam previsões de múltiplos modelos diferentes, como *Random Forest* e *XGBoost*

de aprendizado de máquina, independente da forma como o modelo opera internamente, utilizando apenas as saídas do modelo. A ideia é fornecer insigths sobre como o modelo toma decisões, mesmo que ele seja uma "caixa-preta". Um exemplo é a técnica de SHAP (SHapley Additive exPlanations).

Lundberg e Lee propuseram o SHAP para explicar o modelo de aprendizado de máquina baseado em árvore, como Random Forest (Floresta Aleatória, explicada na seção 3.3.1) e XGBoost (em 3.3.2), estimando a contribuição de cada previsão da variável com base na abordagem da teoria dos jogos, discutida mais a frente na seção 3.4.

Capítulo 2

Base de dados

A base de dados fotométricos utilizada foi obtida do Southern Photometric Local Universe Survey (S-PLUS), um levantamento astronômico que cobre aproximadamente 9300° quadrados do céu no hemisfério Sul. O telescópio em operação no Chile possui um sistema óptico fotométrico de 12 bandas Javalambre[1], as quais também são encontradas na base de dados utilizada neste trabalho. Ao todo, serão estudados 10 mil objetos: 5000 galáxias e 5000 estrelas, classificadas como 0 e 1, respectivamente, tendo, portanto, a classificação como uma informação já disponível e encontrada na última coluna da base.

O sistema fotométrico de 12 bandas do S-PLUS foi idealizado para o projeto Javalambre Photometric Local Universe (J-PLUS), em que se tem como um de seus objetivos, dado um certo conjunto de critérios, maximizar a precisão e a eficiência na classificação dos objetos. Tal sistema é constituído por sete filtros de banda estreita (*J0378, J0395, J0410, J0430, J0515, J0660, J0861*), além de cinco filtros de banda larga (*u, g, r, i e z*), que podem ser verificadas conforme as curvas de transmissão na Figura 2.1. Sendo *g, r, i* e *z* semelhantes em valores com os do SDSS[2] (Mendes de Oliveira et al. (2019), Fukugita et al. (1996)).

Na mesma imagem das curvas de transmissão, os filtros fotométricos estão associados à legenda e distribuídos ao longo dos valores de comprimento de onda do espectro eletromagnético, variando aproximadamente de 3.000 Å a 10.000 Å, indo do ultravioleta ao infravermelho próximo, como é caracterizado um estudo fotométrico no espectro visível.

[1] Em referência ao Observatório Astrofísico Javalambre na Espanha

[2] Sloan Digital Sky Survey

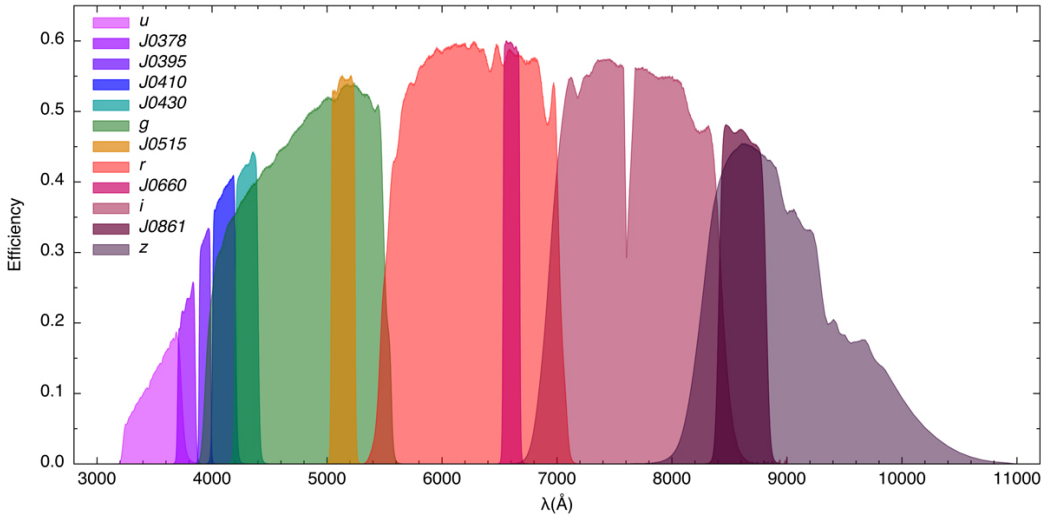

Figura 2.1: Curvas de transmissão dos 12 filtros do sistema S-PLUS (Cenarro et al. (2019)).

Na Tabela 2.1, conforme Cenarro et al. (2019), são identificados os valores de comprimento de onda central (λ_C) em que a emissão de cada filtro é mais intensa. Há também a largura ($\Delta\lambda$) dessas bandas na metade do valor máximo e um comentário quando o filtro é comum com o Javalambre Physics of the Accelerating Universe Astrophysical Survey (J-PAS) ou com o Sloan Digital Sky Survey (SDSS), além da característica referente ao filtro.

Tabela 2.1 - Comprimentos de onda centrais (λ_C) e as larguras ($\Delta\lambda$) das bandas dos 12 filtros do sistema S-PLUS. Sendo (a) em comum com J-PAS e (b) com SDSS.

Filtro	λ_C (Å)	$\Delta\lambda$ (Å)	Comentário
u	3485	508	(a)
J0378	3785	168	$[O_{II}]$; (a)
J0395	3950	100	$Ca\ H+K$
J0410	4100	200	H_δ
J0430	4300	200	$Gband$
g	4803	1409	(b)
J0515	5150	200	$Mg\ b\ triplet$
r	6254	1388	(b)
J0660	6600	138	H_α; (a)
i	7668	1535	(b)
J0861	8610	400	$Ca\ triplet$
z	9114	1409	(b)

As magnitudes da base referem-se às que são calculadas segundo a métrica "Razão de Petrosian" (R_P[3]), a qual compara o brilho superficial em um anel, a uma dada distância r do centro do objeto, com o brilho médio dentro de r, conforme pode ser visto na equação 2.1.

$$R_P = \frac{\text{Brilho superficial em r}}{\text{Brilho médio em r}} \tag{2.1}$$

As magnitudes de Petrosian são medidas fotométricas que representam a quantidade total de luz emitida por uma galáxia, por exemplo, dentro de um raio r específico (também chamado raio de Petrosian r_P, valor no qual a razão R_P se torna constante; em geral, essa razão é $0,2$). Dessa forma, as regiões mais externas das galáxias são desconsideradas, pois a intensidade superficial da luz diminui com o aumento da distância radial, permitindo manter uma homogeneidade do objeto.

O comprimento de onda central de um filtro, como o u, com valor de 3485 Å, conforme apresentado na Tabela 2.1, é o mesmo para u_{petro}, pois ambos utilizam o mesmo intervalo de comprimento de onda, o que tornam fixas suas propriedades físicas. No entanto, a diferença entre u e u_{petro} está no método de cálculo da magnitude. Enquanto u representa a magnitude total, u_{petro} é calculada com base no critério estabelecido pelo "raio de Petrosian". Esse mesmo método se aplica aos demais filtros da base de dados, indicados, portanto, como magnitudes de Petrosian.

2.1 Tratamento dos dados

Em uma análise preliminar dos dados, fez-se uma verificação de valores ausentes, uma vez que dados faltantes podem enviesar e resultar em tomadas de decisões errôneas. A ocorrência dessa situação exige uma intervenção apropriada para tratar o problema de forma eficaz. Para isso, têm-se disponíveis diversas técnicas de Aprendizado Profundo (*Deep Learning*, ou ainda *DL*) que podem ser adaptadas aos diferentes tipos de dados (tabulares, imagens, dentre outros).

Nesse sentido, com a execução do código `df.isna().sum()` em `python` e a subsequente análise da saída no terminal, verificou-se que não há valores ausentes na base de dados.

[3] Fonte: http://www.astro.iag.usp.br/ laerte/aga295/5_luminosidades.pdf

Todas as 12 bandas fotométricas e o campo “classe” apresentaram soma de valores ausentes igual a 0. Consequentemente, nenhum método de *DL* para preenchimento de dados indisponíveis foi necessário.

Outra abordagem de pré-processamento é a conversão das variáveis categóricas para o formato numérico. No entanto, tal procedimento não se fez necessário, visto que todos os dados da base são quantitativos. Adicionalmente, outro ponto a se considerar foi o balanceamento das classes que, da mesma forma, também não foi requerido, dado que a base apresenta uma distribuição uniforme, tendo exatos 50% de galáxias e 50% de estrelas.

2.2 Análise gráfica dos dados

Com a finalidade de visualizar como os dados se comportam a partir de uma análise visual, foram construídos diagramas cor-cor e histogramas, conforme apresentados na Figura 2.2. As galáxias são representadas pela cor azul, enquanto as estrelas são representadas em vermelho. A diferença nas distribuições pelos diagramas cor-cor (gráficos à esquerda) entre os dois tipos de objetos pode sugerir que eles ocupariam regiões distintas no espaço de cores, o que facilitaria uma possível separação. Observa-se cuidadosamente que as galáxias tendem a ser mais dispersas, enquanto as estrelas estão concentradas em uma faixa mais restrita.

Galáxias são sistemas mais complexos, com populações estelares diversas, além da presença de gás e poeira, o que as caracterizam com uma maior variabilidade nas magnitudes. Enquanto as estrelas, como objetos únicos apresentam propriedades mais bem definidas, o que reflete no padrão de magnitudes mais concentrado em determinada área do diagrama. Entretanto a separação dos dois tipos de objetos pela distribuição em um diagrama cor-cor não foi tão eficiente, sendo necessário um método mais eficaz e complexo do que fazer apenas uso do método linear, como foi utilizado na construção dos diagrmas, ou a necessidade de se incorporar mais informações.

Os histogramas, por outro lado, encontrados à direita e com número de bins igual a 100, ilustram mais facilmente os dois conjuntos de objetos. Alguns com mais sobreposições que outros, mas ainda assim visualmente mais favoráveis que os diagramas. Cada barra indica a frequência com que cada intervalo de cor ocorre. A linha tracejada nos histogramas representa a moda (o intervalo mais frequente) e é indicada separadamente para cada tipo

de objeto. De maneira geral, as galáxias apresentam uma moda ligeiramente inferior em comparação com as estrelas, sugerindo que as estrelas tendem a ter valores mais elevados para o mesmo intervalo de cor.

Essa abordagem visual dos dados pode ajudar a identificar padrões e tendências, podendo destacar onde ocorrem sobreposições, que poderiam configurar como ambiguidade na classificação e, portanto, não ajudar muito no desempenho de classificação do modelo. Bem como onde há uma maior separação, demonstrariam quais combinações de variáveis apresentam maior relevância no treinamento dos modelos. Neste trabalho, os dois tipos de objetos se mostraram mais misturados entre si, sugerindo que modelos preditores mais complexos deverão ser utilizados para que se possa estabelecer a melhor separabilidade possível entre as duas classes de objetos.

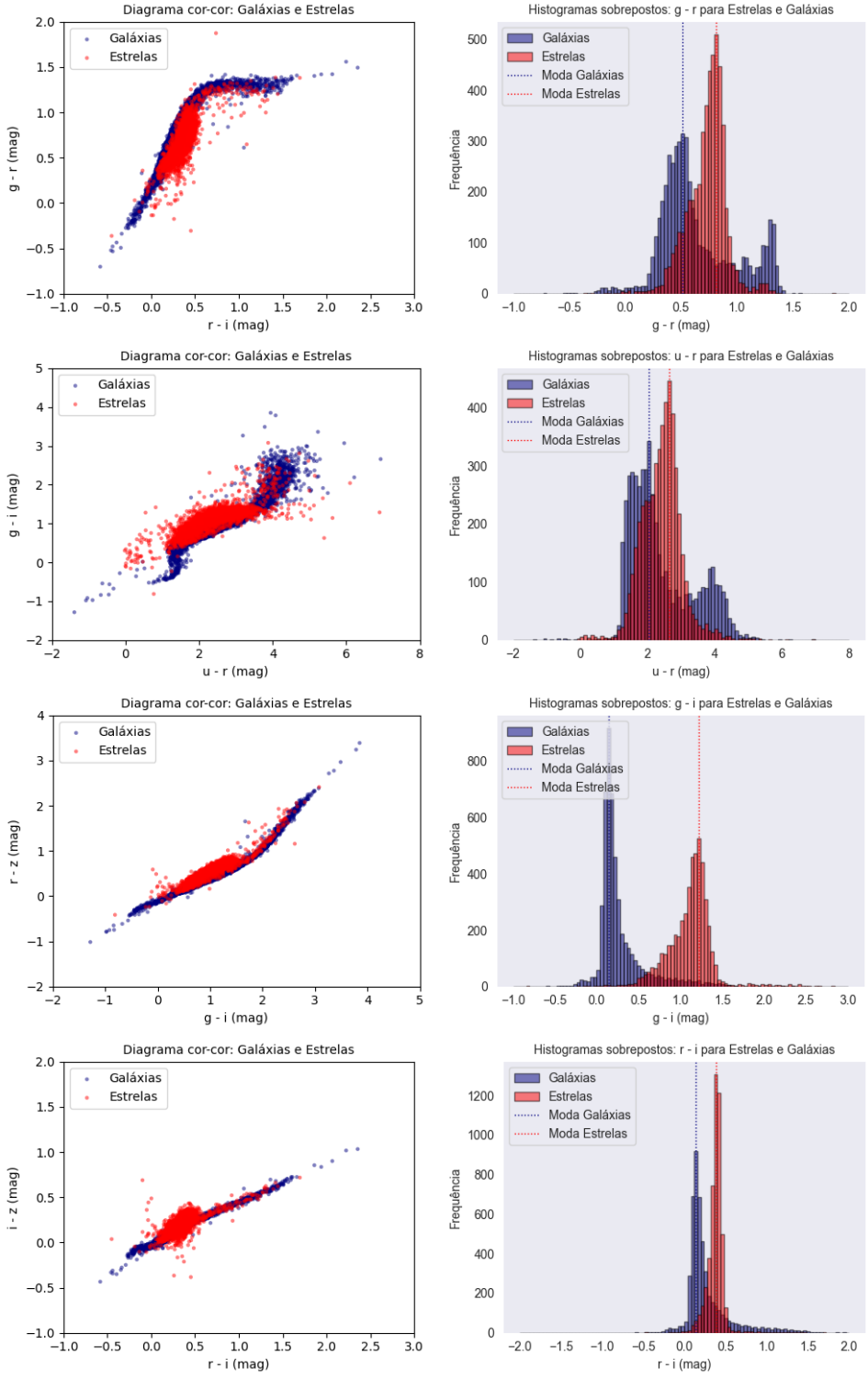

Figura 2.2: Diagramas cor-cor à esquerda e histogramas à direita.

Uma matriz de correlação das 12 bandas fotométricas é encontrada na Figura 2.3. A análise de correlação das variáveis é uma técnica estatística usada para medir e interpretar o grau de vínculo entre elas. A correlação identifica se duas variáveis se movem juntas e de que maneira, ajudando a entender se elas têm uma relação linear, ou seja, se uma aumenta ou diminui em relação à outra. Esse tipo de análise é fundamental em muitos estudos porque permite insights sobre dependências e associações entre variáveis, facilitando inferências e a construção de modelos preditivos.

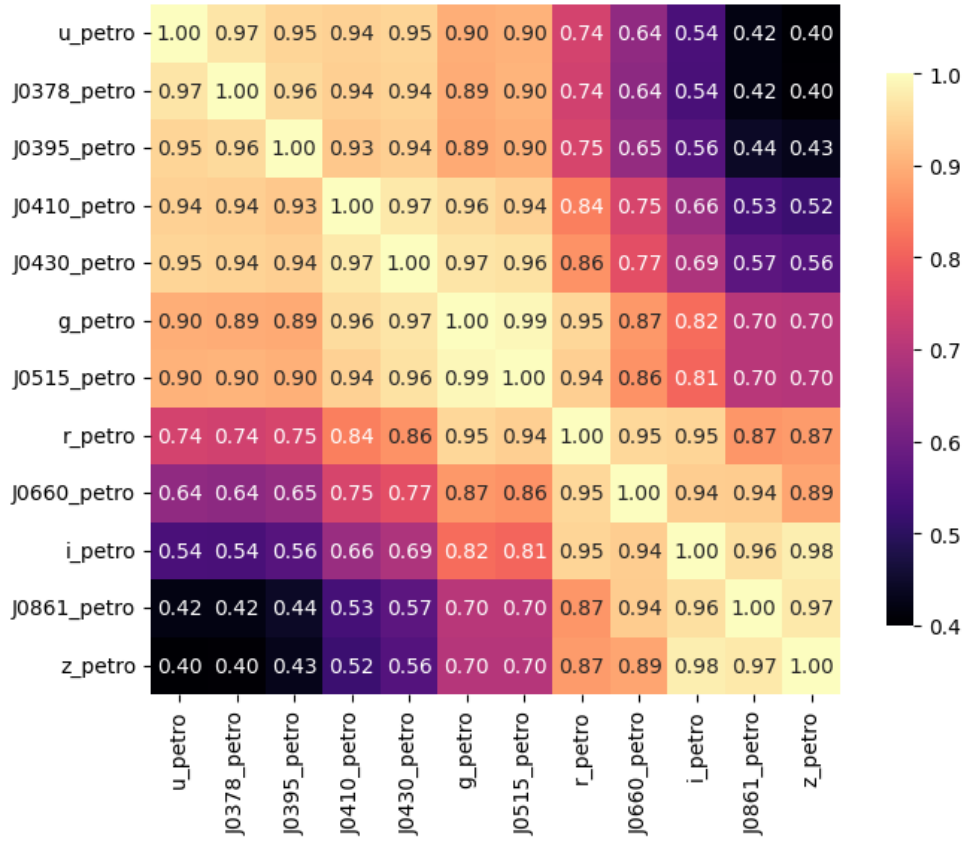

Figura 2.3: Matriz de correlação das 12 bandas fotométricas.

Em matrizes de correlação as diagonais apresentam sempre valor 1 na correlação aos pares, representando a correlação perfeita de cada variável com ela própria. Uma matriz de correlação tem por base o cálculo dos coeficientes (ρ) de correlação de Pearson[4] entre duas variáveis. Os coeficientes podem ter valor entre -1 e 1. Ter $\rho = 1$ indica uma relação positiva perfeita, isto é, se uma variável aumenta a outra também aumenta perfeitamente linear. Se $\rho = -1$ significa que se uma variável aumenta a outra diminui perfeitamente

[4] Grande contribuidor da estatística matemática.

linear. Enquanto que para $\rho = 0$ mostra que as duas variáveis não dependem linearmente uma da outra, podendo existir uma dependência que não seja linear.

Com base nos dados analisados neste estudo, a matriz da Figura 2.3 possui uma escala de cor na qual tons mais claros representam maiores correlações lineares, enquanto que as tonalidades mais escuras, as correlações mais fracas. Como pode ser notado, ultravioleta (u) e infravermelho próximo (z) que, conforme as curvas de transmissão da Figura 2.1 ilustram, encontram-se nas extremidades opostas. Ambos os filtros são amplamente discutidos na literatura, em especial no contexto de formação estelar e evolução de populações estelares, portanto se sobressaem em contextos astrofísicos diferentes.

Capítulo 3

Processamento dos dados

3.1 Normalização

Para diminuir a complexidade computacional de modelos, técnicas de ajuste de tamanho do conjunto de dados, ou seja, ajuste de seu dimensionamento, são essenciais na precisão de um modelo utilizado. Segundo Sujon et al. (2024), técnicas de pré-processamento de dados devem ser apropriadas em função das características dos conjuntos de dados e da compatibilidade com diferentes algoritmos, e que, embora a padronização melhore o desempenho de modelos lineares como Support Vector Machine (SVM)[1] e Logistic Regression (LR), abordado na seção 3.3.3, em bases de dados grandes e médias, a normalização para modelos lineares de dados pequenos melhora o desempenho.

Assim, para a normalização da base utilizada neste trabalho, foram importadas duas classes utilizando o código

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler
```

inserido no cabeçalho de exportações. A equação 3.1 representa a técnica de normalização aplicada transformando os dados de seus valores originais para uma faixa padrão entre 0 e 1.

$$x_{norma} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.1}$$

O termo x_{norma} é o valor normalizado, x é o valor que queremos normalizar, x_{min} é o valor mínimo e x_{max} é valor máximo de uma determinada coluna específica da base de

[1] Algoritmo de aprendizado de máquina supervisionado usado para classificação e regressão, que busca encontrar a melhor linha ou hiperplano que separa diferentes classes de dados.

dados. Dessa forma, o entendimento da equação 3.1 tem por base a interpretação de que a subtração no numerador, $x - x_{min}$, desloca todos os valores para que o menor deles se torne 0. Assim, os valores passam a variar entre 0 e a diferença máxima possível, x_{max} - x_{min}, pelo escalonamento no denominador.

3.2 *K-Fold Cross-Validation*

A Figura 3.1 esquematiza o processo de *K-Fold Cross-Validation* (Validação Cruzada K-Fold), técnica comum do campo interdisciplinar das Ciência de Dados que auxilia a avaliar a performance de modelos de *Machine Learning*. No topo da Figura, "All data" representa todos os dados disponíveis e que serão divididos em várias partes (*folds*) para executar a validação. Abaixo, o conjunto total foi dividido em 5 *folds* iguais. Na figura, a configuração é de *5-Fold Cross-Validation*.

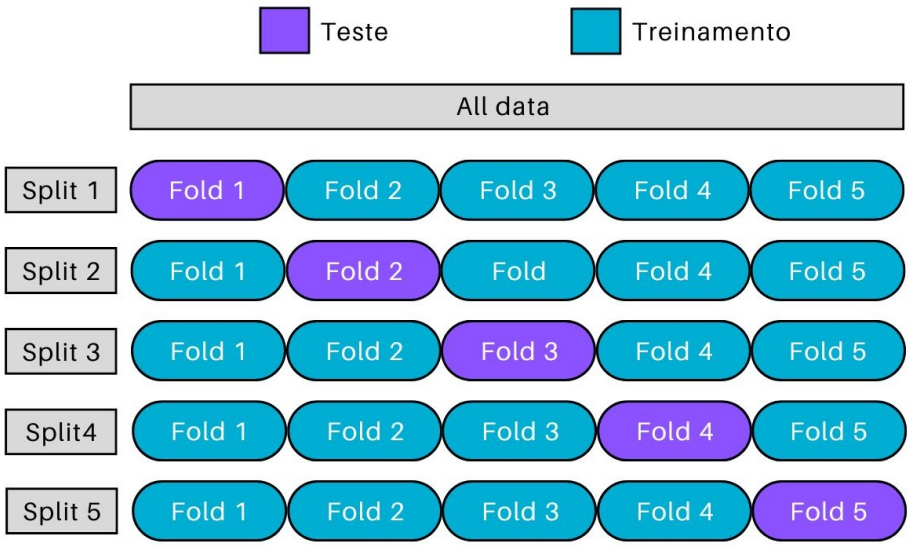

Figura 3.1: Ilustração do método *K-Fold Cross Validation*.

A cada rodada (*Split*), cada uma das partes é utilizada como conjunto de teste, enquanto os outros *K-1 folds* são utilizadas como conjunto de treinamento. Dessa forma, cada *fold* é utilizado em algum momento como conjunto de teste. Nesse sentido, os *Splits* representam cada rodada de validação: o *fold* em roxo seria o conjunto de teste, enquanto que os demais, em azul, formariam o conjunto de treinamento. Cada dado é utilizado uma única vez para teste e quatro vezes para treinamento do processo de validação.

Como consequência, para avaliar o desempenho de algoritmos de classificação é útil fazê-lo por meio do método *K-Fold Cross Validation* com algumas repetições. Isto porque, segundo (Wong e Yeh (2020)), vários estudos demonstraram que ao implementar esta técnica, repetidamente, as estimativas do quão preciso é um modelo, seriam mais confiáveis em termos de comparação estatística, além de diminuir a dependência entre elas mesmas, o que pode impactar no cálculo de suas variâncias[2].

Para este processo do trabalho, foram importadas duas classes conforme o código a seguir:

```
from sklearn.model_selection import train_test_split, KFold
```

as quais foram estruturadas ao longo do código com 20% dos dados utilizados como conjunto de teste e 80% como conjunto de treinamento, o número de folds igual a 5 e a "classe" como a *feature* que se deseja prever. Então, para cada iteração do método de *K-Fold Cross-Validation*, 1 *fold* era conjunto de teste e 4 eram de treinamento. É importante destacar que um mesmo dado não é utilizado em *folds* distintos, sendo empregado apenas em um único *fold*. Em cada um dos três modelos preditivos utilizados neste trabalho foi utilizado a técnica de Validação Cruzada.

3.3 Exploração de modelos preditivos

Modelos preditivos são algoritmos ou técnicas de *Machine Learning* e Estatística utilizados para prever resultados ou tendências. Isso é possível com base em dados históricos utilizados como treinamento de aprendizado e como teste do modelo. A utilização dos dados com o alvo já disponível na base, configura os modelos como sendo do tipo preditivos supervisionados. Em outras palavras, cada entrada no conjunto de treinamento já possui uma saída desejada. Neste trabalho foram utilizados algoritmos de *Random Forest*, *XGBoost* e *Logistic Regression*, obtidos por importações de bibliotecas do `python`.

Ao final da execução de cada modelo, é possível obter as métricas derivadas das matrizes de confusão. A matriz de confusão é uma ferramenta analítica que organiza as previsões de um modelo de classificação em relação aos valores reais observados, permitindo identificar acertos e erros de forma estruturada. Para um problema de classificação binária, ela é

[2] Medem a dispersão em relação à média.

composta por quatro quadrantes que correspondem às combinações entre os valores reais e previstos, categorizados de forma padrão como verdadeiros positivos (VP), verdadeiros negativos (VN), falsos positivos (FP) e falsos negativos (FN), como ilustra a Tabela 3.1.

Na prática, as quatro siglas são substituídas por valores e são melhor exemplificadas nas matrizes obtidas, encontradas nas seções 3.3.1, 3.3.2 e 3.3.3.

Tabela 3.1 - Matriz de Confusão 2x2.

	Previsão Positiva	Previsão Negativa
Real Positivo	VP	FN
Real Negativo	FP	VN

Adaptando a matriz para refletir a classificação entre galáxia ou estrela, os elementos da matriz representam os acertos e erros do modelo na identificação de cada tipo de objeto. Esse ajuste pode ser visto na Tabela 3.2 em que galáxia é positivo e estrela é negativo.

Tabela 3.2 - Matriz de Confusão para classificação galáxia/estrela.

	Previsão: Galáxia	Previsão: Estrela
Real: Galáxia	Nº de Galáxias classificadas como Galáxias	Nº de Galáxias classificadas como Estrelas
Real: Estrela	Nº de Estrelas classificadas como Galáxias	Nº de Estrelas classificadas como Estrelas

Assim, as métricas médias obtidas para cada modelo preditivo em relação ao número de *Splits* igual a 5, seguem as Equações 3.2, 3.3, 3.4 e 3.5, para cada *Split*, adequando-se ao caso de classificação galáxia/estrela. A correspondência dos rótulos "Galáxia"e "Estrela"nos eixos é definida diretamente no corpo do código.

A acurácia é a proporção de previsões corretas em relação ao total de previsões, ou seja, a porcentagem de acertos do modelo:

$$Acurácia = \frac{VP + VN}{VP + FP + FN + VN} \tag{3.2}$$

A precisão mede a proporção de acertos para cada classificação prevista:

$$Precisão\ (Galáxia) = \frac{VP}{VP + FP} \quad \text{e} \quad Precisão\ (Estrela) = \frac{VN}{VN + FN}$$

$$\textit{Precisão Média} = \frac{\textit{Precisão (Galáxia)} + \textit{Precisão (Estrela)}}{2} \tag{3.3}$$

A completeza mede a proporção do quanto a classificação real foi corretamente obtida em função de todos os objetos que verdadeiramente são daquela classe:

$$\textit{Recall (Galáxia)} = \frac{VP}{VP + FN} \quad \text{e} \quad \textit{Precisão (Estrela)} = \frac{VN}{VN + FP}$$

$$\textit{Recall Médio} = \frac{\textit{Recall (Galáxia)} + \textit{Recall (Estrela)}}{2} \tag{3.4}$$

E, por último, o F1-Score Médio combina precisão e completeza, tratando-se de um único valor denominado média harmônica. Ela é particularmente útil quando se busca um equilíbrio entre as duas métricas. Um alto valor de F1-Score indica que o modelo é consistente em identificar verdadeiros positivos e em evitar falsos positivos.

$$\textit{F1-Score (Galáxia)} = 2 \cdot \frac{\textit{Precisão (Galáxia)} \cdot \textit{Recall (Galáxia)}}{\textit{Precisão (Galáxia)} + \textit{Recall (Galáxia)}}$$

$$\text{e} \quad \textit{F1-Score (Estrela)} = 2 \cdot \frac{\textit{Precisão (Estrela)} \cdot \textit{Recall (Estrela)}}{\textit{Precisão (Estrela)} + \textit{Recall (Estrela)}}$$

$$\textit{F1-Score Médio} = \frac{\textit{F1-Score (Galáxia)} + \textit{F1-Score (Estrela)}}{2} \tag{3.5}$$

3.3.1 Randon Forest

Randon Forest (ou Floresta Aleatória) é um modelo supervisionado que consiste de um algoritmo preditivo de *Machine Learning* baseado em várias "árvores de decisão". Em outras palavras, a estrutura por trás de seu funcionamento se assemelha aos ramos de uma

árvore, os quais seriam os diferentes "caminhos" que o algoritmo toma para chegar no valor previsto. Como exemplo, supondo um dado valor da base, ele passará por uma série de perguntas (os ramos da árvore) para se prever a melhor resposta a que ele se encaixa (Anselmo (2020)). Um esquema de *Random Forest* pode ser visto na Figura 3.2.

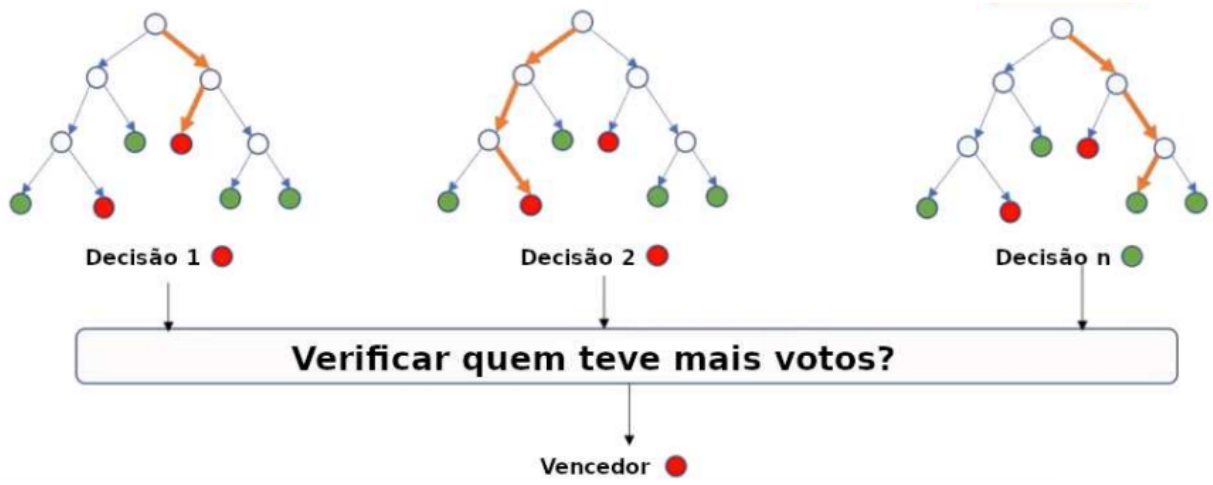

Figura 3.2: Representação de árvores de decisão (Anselmo (2020)).

Aplicando a ideia para este trabalho, pode-se considerar que o primeiro nível a ser submetido o valor da banda fotométrica r_petro normalizado, por exemplo, poderia ser tal que:

- se r_petro < 0.5, o valor segue para o próximo nó onde a classificação é mais provável de ser uma galáxia;
- se r_petro ≥ 0.5, o valor segue para uma análise mais profunda em relação a outros filtros para confirmar sua classificação.

No próximo nível, a classificação verifica outro filtro seguindo o mesmo princípio, estando o valor dentro ou fora de um intervalo, que o configuraria como mais provável de ser galáxia ou estrela. Assim, seguindo as ramificações possíveis, cada caminho na árvore termina em uma folha, representando a classe final do objeto.

A Figura 3.3 é a matriz de confusão do modelo *Random Forest*. Segundo ela, para $K = 5$, em média 4865 objetos (97,30%) foram corretamente classificados como galáxia e 4744 (94,88%) corretamente como estrelas.

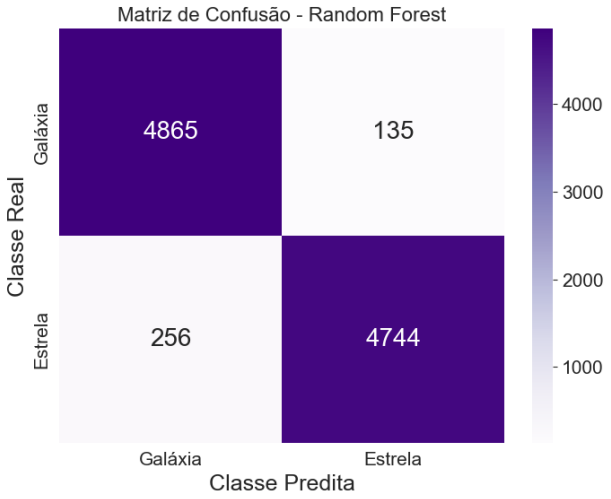

Figura 3.3: Matriz de Confusão para o modelo Random Forest.

As métricas médias obtidas por esse modelo, com o hiperparâmetro `n_estimators`, ou seja, o número de árvores de decisão igual a 100, foram conforme a Tabela 3.3.

Tabela 3.3 - Métricas do modelo *Random Forest* para 5-Fold Cross Validation.

Métrica	**Valor**
Acurácia Média	96,09%
Precisão Média	97,23%
Completeza Média	94,88%
F1-Score Médio	96,04%

3.3.2 XGboost

O *XGBoost* (*Extreme Gradient Boosting*) é um modelo eficiente e de alto desempenho de Gradient Boosting Machines (GBMs), um algoritmo de aprendizado supervisionado que utiliza técnicas para combinar múltiplos modelos fracos (geralmente árvores de decisão) e criar um modelo forte e preditivo. O *XGBoost* é amplamente utilizado devido à sua eficiência computacional, capacidade de lidar com grandes volumes de dados e robustez

em uma variedade de problemas, especialmente em tarefas de classificação (como é o caso deste trabalho) e de regressão.

O XGBoost segue a abordagem de *boosting*, que consiste em treinar modelos de forma sequencial, onde cada modelo posterior tenta corrigir os erros do modelo anterior. O termo "Extreme" se refere ao foco na otimização de diversos aspectos do algoritmo, como a regularização, a melhoria no cálculo das derivadas e o uso de uma técnica de divisão eficiente que acelera a construção das árvores (Chao et al. (2019)).

Na Figura 3.4 encontra-se a matriz de confusão do modelo. Em termos de porcentagem, 97,6% dos objetos que são realmente galáxias foram corretamente classificados, enquanto que para estrelas, 94,96% foram corretamente classificados como estrelas.

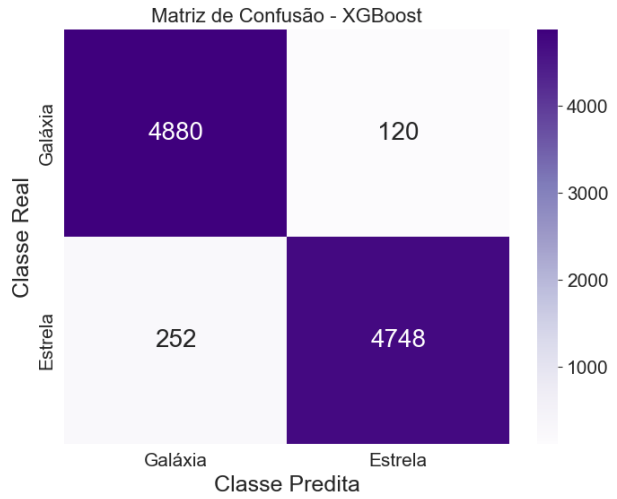

Figura 3.4: Matriz de Confusão para o modelo XGBoost.

As métricas para este modelo se encontram na Tabela 3.4.

Tabela 3.4 - Métricas do modelo *XGBoost* com 5-Fold Cross Validation.

Métrica	**Valor**
Acurácia Média	96,28%
Precisão Média	94,96%
Completeza Média	97,54%
F1-Score Médio	96,04%

3.3.3 *Logistic Regression*

O modelo estatístico de aprendizado supervisionado *Logistic Regression* (ou Regressão Logística) é utilizado para prever a probabilidade de um evento ocorrer. Enquanto a regressão linear é usada para prever uma variável contínua, a regressão logística é aplicada quando a variável dependente (a que estamos tentando prever) é categórica, geralmente binária (ex: 0 ou 1, sim ou não, sucesso ou fracasso, ou como neste trabalho, galáxia ou estrela) (Couronné et al. (2018)).

Para estimar a probabilidade do evento ocorrer, a regressão logística utiliza-se de uma função sigmoide gerando uma saída entre 0 e 1. Assim, supondo Y a variável de resposta binária de interesse e $X_1, ..., X_p$ as variáveis da base de dados (as bandas fotométricas), a probabilidade condicional de Y ocorrer (ser 0 ou 1) dado a existência das variáveis é dada pela Equação 3.6.

$$P(Y = 1) = \frac{e^{(\beta_0+\beta_1 X_1+...+\beta_p X_p)}}{1 + e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}} \tag{3.6}$$

Em que $P(Y = 1)$ é a probabilidade do evento de interesse ocorrer (por exemplo, $Y = 1$), enquanto que $\beta_0 + \beta_1 X_1 + ... + \beta_p X_p$ é uma combinação linear com seus respectivos coeficientes de regressão, estimados pela máxima verossimilhança com base no conjunto de dados analisados. A função de verossimilhança (Equação 3.7) é maximizada para estimar os coeficientes.

$$L(\beta) = \prod_{i=1}^{n} P((Y_i|Xi)^{y_i} \cdot (1 - P(Y_i|X_i))^{1-Y_i} \tag{3.7}$$

O termo $P(Y_i|X_i)$ é a probabilidade predita para a observação i, Y_i é o valor observado (0 ou 1), X_i é o vetor de variáveis independentes para a observação i e n é o número total de observações.

Assim, após realizar *K-Fold Cross Validation*, o modelo de *Logistic Regression* foi aplicado em cada iteração do processo de validação nos dados de treinamento. Como resultado, esse simples e eficaz modelo de classificação binária fornece uma probabilidade de pertencimento a uma classe (galáxia ou estrela) que, como forma de verificação de sua performance,

uma matriz de confusão, na Figura 3.5, possibilita verificar a quantidade de classificações corretas e incorretas.

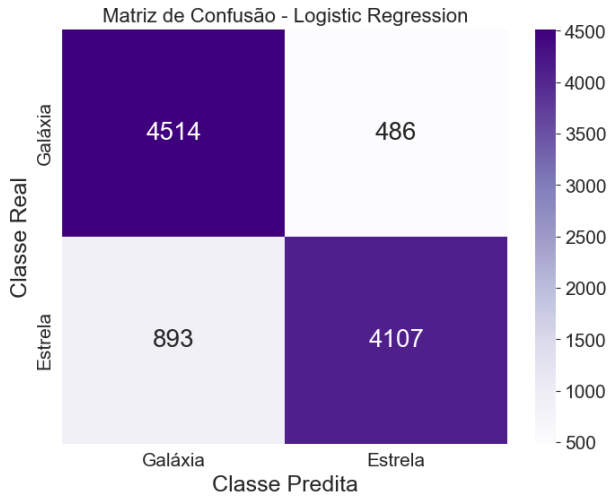

Figura 3.5: Matriz de Confusão para o modelo *Logistic Regression.*

Como pode ser verificado na matriz, no eixo y seriam as classificações reais dos objetos, enquanto que no eixo x seria a classificação que o modelo de *Logistic Regression* previu. No primeiro quadrante, em roxo, 4514 (90,28%) das 5000 galáxias foram possíveis de ser identificadas corretamente pelo modelo, enquanto que 4107 (82,14%) dos objetos que são estrelas foram classificadas como estrelas.

Com base nesses dados visuais, as métricas obtidas para esse modelo estão na Tabela 3.5. Comparando com os demais modelos, foi o que obteve as menores métricas.

Tabela 3.5 - Métricas do modelo *Logistic Regression com 5-Fold Cross Validation.*

Métrica	**Valor**
Acurácia Média	86,21%
Precisão Média	89,41%
Completeza Média	82,13%
F1-Score Médio	85,61%

3.4 Valores SHAP aplicados a XGBoost

Entender por que um modelo consegue fazer uma determinada previsão pode ser tão importante quanto a precisão com que foram obtidos os resultados. Vários métodos foram

propostos para tentar ajudar os usuários a entender as previsões de modelos complexos, porém não deixam muito clara a relação entre a precisão e a interpretabilidade.

Nesse sentido, Lundberg e Lee (2017), para abordar esse problema, apresentaram uma forma de interpretar previsões, por meio do método *SHapley Additive exPlanations* (SHAP), que combina os conceitos de valores de Shapley da teoria dos jogos com um modelo explicativo aditivo (ou seja, a soma da contribuição de todas as variáveis, que depende fortemente da presença ou não de uma determinada variável na predição).

Os valores de Shapley são uma solução para questões envolvendo alocação de recursos ou de ganhos em jogos cooperativos. Foram introduzido por Lloyd Shapley, em 1953, baseados na ideia de distribuir os ganhos totais de um grupo de participantes cooperando, de forma justa para cada jogador. As principais propriedades envolvidas nos valores de Shapley satisfazem os axiomas:

- **Simetria:** jogadores/filtros que contribuíram igualmente recebem valores iguais;
- **Eficiência:** a soma dos valores de Shapley de todos os jogadores/filtros é igual ao ganho total na coalizão/desempenho do modelo em função dos filtros utilizados ou não;
- **Atividade:** na combinação de dois jogos/processos de classificação, os valores de Shapley de um jogador/filtro são a soma de suas alocações em cada jogo/processo de classificação;
- **Contriuição nula:** se não há contribuição, o valor de Shapley é zero para o jogador/filtro.

Para exemplificar de maneira bem simples, suponha que três amigos se unem para comprar um presente. Cada um contribui com uma quantia diferente de dinheiro. Depois, eles pretendem dividir o crédito pelo presente de forma justa: quem deu mais dinheiro tem um pouco mais de crédito que os demais. Em um modelo de *Machine Learning* utilizado para classificar objetos como galáxia ou estrela, os "amigos"são as variáveis (as magnitudes dos filtros), e os valores de Shapley ajudam a descobrir qual variável mais favoreceu o modelo a fazer uma boa previsão.

Se, por exemplo, o filtro g_petro contribuiu significativamente para distinguir estrelas de galáxias, ele terá um valor de Shapley maior. Por outro lado, se z_petro tiver pouca

influência, seu valor de Shapley será menor. Assim, é possível ter uma visão mais clara de como as variáveis individualmente impactam na previsão. Uma descrição mais bem detalhada de outro exemplo pode ser encontrado também em Vidal e Machado (2023).

A técnica de valores SHAP foi aplicada somente para o modelo que teve o melhor desempenho. Com base nas acurácias dos modelos, conforme a Tabela 3.6, *XGBoost* se mostrou como o modelo que melhor se comportou na classificação dos objetos, com uma taxa de 96,28%. Espera-se que o modelo não apresente 100% de acurácia pois indicaria que o modelo estaria "memorizando"os dados ao invés de aprender os padrões e a avaliação da robustez de um modelo implica na sua capacidade de generalizar para novos dados.

Tabela 3.6 - Comparação das acurácias médias dos modelos *Random Forest, XGBoost e Logistic Regression.*

Acurácia Média	**Modelo**
Random Forest	96,09%
XGBoost	96,28%
Logistic Regression	86,21%

Os valores SHAP podem ser representados em um gráfico de distribuição, conforme a Figura 3.6. Nele, cada ponto é a relação entre o valor do filtro e o impacto no modelo. No eixo y se encontram os filtros e no x, os valores SHAP. À esquerda do eixo central estão os valores referentes à classificação como galáxia e à direita, como estrela. A coloração varia do azul (baixo valor do filtro) ao rosa (alto valor do filtro). Isso reflete a magnitude do valor do filtro no impacto do modelo. Os filtros no eixo y estão ordenados em ordem decrescente de influência.

Observando o filtro J0410_petro, a cor azul indica que valores baixos influenciaram o modelo na classificação como galáxias (tomando o módulo do valor), e a cor rosa, à extrema direita, indica que valores altos influenciaram na classificação do objeto como estrela. Portanto, valores baixos favorecem a classificação como galáxia e valores altos como estrelas. Analisando a dispersão, a concentração dos pontos numa região, como é possível notar na maioria dos filtros, indica que para muitos de seus valores o impacto é praticamente o mesmo. Isso pode ocorrer por serem valores muito similares no conjunto de dados.

Continuando a interpretar o gráfico, o filtro i_petro foi considerado pelo modelo como o que possui a mais baixa influência na determinação da classificação do objeto, independente

de seu valor ser alto ou baixo, como pode ser notado os baixos valores SHAP.

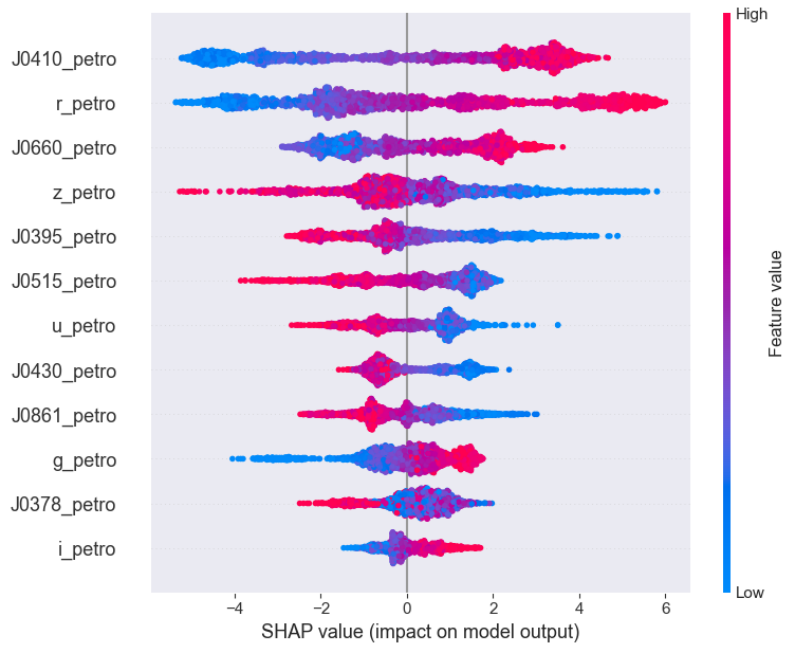

Figura 3.6: Gráfico de dispersão dos valores SHAP para o modelo XGBoost.

O gráfico de barras na Figura 3.7 mostra a distribuição geral dos valores médios de SHAP para todos os filtros fotométricos sem distinção entre as classes.

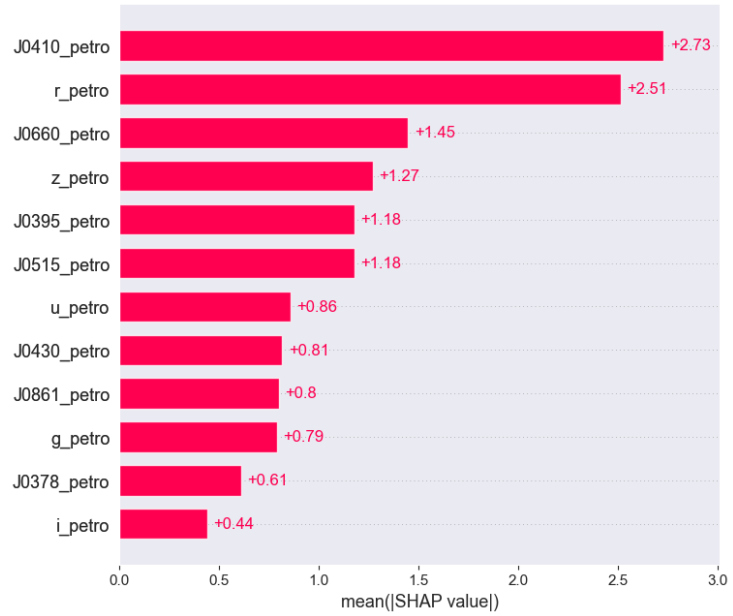

Figura 3.7: Gráfico de barras dos valores SHAP dos 12 filtros (direita).

Diferentemente da Figura 3.6, ele não apresenta uma visão mais detalhada de como

cada variável se distribui, nem identifica padrões, outliers e concentrações. Entretanto, ainda assim permanece a ordem de influência das variáveis.

Capítulo 4

Análise

O trabalho focou na busca pelas bandas fotométricas que mais favorecem a classificação de objetos astronômicos entre estrelas e galáxias. Para tal, utilizou-se uma base de dados composta por 12 bandas fotométricas e a classe como alvo. Durante o pré-processamento, verificou-se que os dados estavam completos e uniformemente distribuídos, com 50% de exemplos de cada classe, o que permitiu descartar a necessidade de balanceamento.

A avaliação dos modelos foi conduzida utilizando métricas como acurácia, precisão, completeza e F1-score. Essas métricas foram computadas a partir das matrizes de confusão, considerando os rótulos reais e previstos para cada classe. Para complementar a precisão dos modelos, na Figura 4.1 se encontra um gráfico com as Curvas ROC (Receiver Operating Characteristic) e sua AUC (Área sob a Curva), que fornecem mais uma perspectiva do desempenho dos modelos.

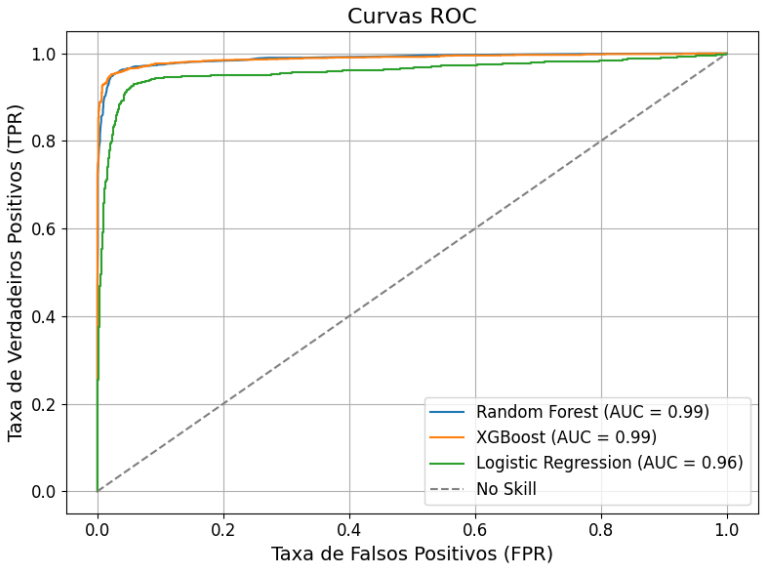

Figura 4.1: Enter Caption

A Curva ROC é particularmente útil por avaliar a capacidade do modelo de distinguir entre duas classes. A AUC, que varia de 0 a 1, quantifica essa habilidade, sendo valores mais próximos de 1 indicativos de um modelo bem-sucedido. Sumariamente, em termos matemáticos, a curva ROC relaciona a taxa de verdadeiros positivos e a taxa de falsos positivos. No caso deste trabalho, como pode ser notado pela legenda, as curvas ROC dos modelos empregados indicaram um desempenho robusto, com AUC variando entre 0,96 e 0,99 para os melhores classificadores.

Dessa forma, a ROC reforça o uso do modelo escolhido (*XGBoost*) para a aplicação da técnica *XAI* na identificação das variáveis relevantes. A reta tracejada em cinza representa um modelo que não possui habilidade de classificação, onde a taxa de verdadeiros positivos é igual à de falsos positivos, sem conseguir diferenciar entre as classes. O modelo que tivesse a curva abaixo dessa reta diagonal, estaria classificando cometendo mais erros do que acertos. As previsões seriam tão ruins que estaria basicamente invertendo as classes, gerando mais falsos positivos e falsos negativos do que acertos.

Os resultados alcançados, no geral, demonstram que as métricas tradicionais e a Curva ROC são ferramentas que se complementam e são essenciais para avaliar os classificadores em problemas de classificação binária. A combinação dessas abordagens fortalece a confiabilidade dos modelos e oferece insights para melhorar sua performance em futuras iterações. A literatura corrobora o uso, como evidenciado em trabalhos de Fawcett (2006), Flach (2016) e vários outros em diversas áreas de pesquisa, que destacam a ROC como uma métrica padrão para avaliação de classificadores. Assim, este trabalho valida a aplicabilidade das técnicas empregadas (*k-Fold Cross Validation*, modelos preditivos, valores SHAP) e fornece uma concepção de cunho fundamental para mais aplicações futuras.

Capítulo 5

Conclusões

Os modelos de aprendizado de máquina utilizados neste trabalho foram *XGBoost, Random Forest e Logistic Regression*. Com base no processamento dos dados, *XGBoost* se mostrou como o melhor classificador, considerando como métrica de comparação os valores das acurácias dos algoritmos. Embora *Random Forest* tenha se mostrado um bom modelo, *XGBoost* teve uma melhor performance. Enquanto que a Regressão Logística se mostrou mais inferior na predição das classes. Assim, com a técnica SHAP, obteve-se como resultado final que as bandas fotométricas J0410_petro e r_petro demonstraram ser as mais importantes e a i_petro menos relevante na categorização dos objetos.

A fim de se estabelecer uma continuidade ao projeto, outros modelos de aprendizado supervisionado e outras bases de dados, de dimensões distintas, poderiam corroborar ou não com o resultado obtido neste trabalho. Tal qual, a expansão da análise de cada um dos algoritmos para um melhor ajuste de seus hiperparâmetros, poderia aprimorar a performance dos modelos utilizados.

Uma abordagem complementar que poderia ser também explorada é a investigação das propriedades físicas de cada filtro, com o intuito de identificar correlações com os gráficos cor-cor, histogramas e a matriz de correlação. Em outras palavras, uma análise astrofísica mais aprofundada poderia fornecer insights sobre possíveis relações entre os filtros e os próprios objetos. Adicionalmente, o intervalo dos valores de cada um dos filtros pode ser mais um aspecto a ser mais abordado, analisando quais intervalos de valores poderiam ser favoráveis nas predições dos modelos em função das características dos objetos.

Referências Bibliográficas

Anselmo F., Machine Learning na Prática. Publicação Independente, 2020, 103 p.

Arrieta A. B., Díaz-Rodríguez N., Del Ser J., Bennetot A., Tabik S., Barbado A., García S., Gil-López S., Molina D., Benjamins R., et al., Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI, Information fusion, 2020, vol. 58, p. 82

Bellotti V., Edwards K., Intelligibility and accountability: human considerations in context-aware systems, Human–Computer Interaction, 2001, vol. 16, p. 193

Cenarro A. e., Moles M., Cristóbal-Hornillos D., Marín-Franch A., Ederoclite A., Varela J., López-Sanjuan C., Hernández-Monteagudo C., Angulo R., Ramió H. V., et al., J-PLUS: The javalambre photometric local universe survey, Astronomy & Astrophysics, 2019, vol. 622, p. A176

Chao L., Wen-hui Z., Ji-ming L., Study of star/galaxy classification based on the xgboost algorithm, Chinese Astronomy and Astrophysics, 2019, vol. 43, p. 539

Couronné R., Probst P., Boulesteix A.-L., Random forest versus logistic regression: a large-scale benchmark experiment, BMC bioinformatics, 2018, vol. 19, p. 1

Fawcett T., An introduction to ROC analysis, Pattern recognition letters, 2006, vol. 27, p. 861

Flach P. A., , 2016 in , Encyclopedia of machine learning and data mining. Springer pp 1–8

Fukugita M., Shimasaku K., Ichikawa T., Gunn J., et al.,, 1996 Technical report The Sloan digital sky survey photometric system. SCAN-9601313

Gunning D., Stefik M., Choi J., Miller T., Stumpf S., Yang G.-Z., XAI—Explainable artificial intelligence, Science robotics, 2019, vol. 4, p. eaay7120

Lundberg S., Lee S.-I., , 2017 A Unified Approach to Interpreting Model Predictions

Mendes de Oliveira C. e., Ribeiro T., Schoenell W., Kanaan A., Overzier R., Molino A., Sampedro L., Coelho P., Barbosa C. E., Cortesi A., et al., The Southern Photometric Local Universe Survey (S-PLUS): improved SEDs, morphologies, and redshifts with 12 optical filters, Monthly Notices of the Royal Astronomical Society, 2019, vol. 489, p. 241

Sujon K. M., Hassan R. B., Towshi Z. T., Othman M. A., Samad M. A., Choi K., When to Use Standardization and Normalization: Empirical Evidence from Machine Learning Models and XAI, IEEE Access, 2024

Vidal M. C., Machado A. C., Inteligência Artificial Explicável (XAI) na área médica, 2023

Wong T.-T., Yeh P.-Y., Reliable Accuracy Estimates from k-Fold Cross Validation, IEEE Transactions on Knowledge and Data Engineering, 2020, vol. 32, p. 1586

Apêndice

Apêndice A

Diagrama Cor-Cor

```
# Separação das classes e quantidade de galáxias e estrelas
galaxias = df[df['classe'] == 0] # galáxias
estrelas = df[df['classe'] == 1] # estrelas

print("Total de galáxias: ", len(galaxias))
print("Total de estrelas: ", len(estrelas))

# Calcula as cores 'g - r' vs. 'r - i' para estrelas e galáxias
estrelas_color_ri = estrelas['r_petro'] - estrelas['i_petro']
estrelas_color_gr = estrelas['g_petro'] - estrelas['r_petro']
galaxias_color_ri = galaxias['r_petro'] - galaxias['i_petro']
galaxias_color_gr = galaxias['g_petro'] - galaxias['r_petro']

# Cria uma figura
plt.figure(figsize=(5, 4)

# Gráfico de dispersão sobreposto
plt.scatter(galaxias_color_ri, galaxias_color_gr, alpha=0.5, color='navy',
            s=10, label='Galáxias', edgecolor='none')
plt.scatter(estrelas_color_ri, estrelas_color_gr, alpha=0.5, color='red',
            s=10, label='Estrelas', edgecolor='none')

# Configurações do gráfico
```

```
plt.xlim(-1, 3)
plt.ylim(-1, 2)
plt.title("Diagrama cor-cor: Galáxias e Estrelas", fontsize=10)
plt.xlabel('r - i (mag)')
plt.ylabel('g - r (mag)')
plt.legend(loc='upper left')

# Ajusta o layout para melhor visualização
plt.tight_layout()

# Exibir o gráfico
plt.show()
```

Apêndice B

Histograma

```
# Cria histograma para as cores 'g - r' para estrelas e galáxias
bins = 100
range_values = (-1, 2)

plt.figure(figsize=(5, 4))

# Histogramas e retorno dos valores de contagem
counts_galaxias, bin_edges_galaxias, _ = plt.hist(galaxias_color_gr,
                                          bins=bins, range=range_values,
                                          alpha=0.5, color='navy',
                                          label='Galáxias', edgecolor='k')
counts_estrelas, bin_edges_estrelas, _ = plt.hist(estrelas_color_gr,
                                          bins=bins, range=range_values,
                                          alpha=0.5, color='red',
                                          label='Estrelas',
                                          edgecolor='k')

# Encontrando o índice da barra mais alta para galáxias
max_index_galaxias = np.argmax(counts_galaxias)
max_index_estrelas = np.argmax(counts_estrelas)

# Encontrando o centro do intervalo onde a barra é maior
center_galaxias = (bin_edges_galaxias[max_index_galaxias] +
```

```
                    bin_edges_galaxias[max_index_galaxias + 1]) / 2
center_estrelas = (bin_edges_estrelas[max_index_estrelas] +
                    bin_edges_estrelas[max_index_estrelas + 1]) / 2

# Adicionando linhas pontilhadas nos intervalos de maior frequência
plt.axvline(center_galaxias, color='navy', linestyle=':', linewidth=1.0,
                                    label='Moda Galáxias')
plt.axvline(center_estrelas, color='red', linestyle=':', linewidth=1.0,
                                    label='Moda Estrelas')

# Configurações do gráfico
plt.xlabel('g - r (mag)')
plt.ylabel('Frequência')
plt.title("Histogramas sobrepostos: g - r para Estrelas e Galáxias",
          fontsize=10)
plt.legend(loc='upper left')

# Ajusta o layout para melhor visualização
plt.tight_layout()
plt.show()
```

Apêndice C

Matriz de Correlação

```
# Separando a variável alvo e as features
X = df.drop(columns=['classe'])
y = df['classe']

# Descrição das variáveis
X.describe()

# Matriz de correlçação das variáveis
correlation_matrix = X.corr()
print(correlation_matrix)
plt.figure(figsize=(10, 8))

sns.heatmap(correlation_matrix, annot=True, cmap='magma',
            linewidth=0.5, cbar_kws={'shrink': .8},
            annot_kws={'fontsize': 10})
plt.title("Matriz de correlação das bandas fotométricas")
plt.show()
```

Apêndice D

Métricas dos modelos preditivos

Aqui se encontram os códigos utilizados para cada um dos três modelos (XGBoost, Random Forest e Regressão Logística).

D.1 XGBoost

```
# Bibliotecas importadas
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split, KFold
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, recall_score,
                            precision_score,
                            f1_score, confusion_matrix

galaxias = df[df['classe'] == 0] # galáxias
estrelas = df[df['classe'] == 1] # estrelas

# Normalização (entre 0 e 1)
scaler = MinMaxScaler()
df_normalized = pd.DataFrame(scaler.fit_transform(df),
```

```
                columns=df.columns)

# Exibir as primeiras linhas do DataFrame normalizado
print("Dados Normalizados:")
print(df_normalized.head())

# 1. Divisão em conjuntos de treino e teste
X_train, X_test, y_train, y_test =
                    train_test_split(df_normalized.drop('classe',
                    axis=1), df_normalized['classe'],
                    test_size=0.2, random_state=22)

# Exibindo tamanhos dos conjuntos
print(f"Tamanho do conjunto de treinamento: {X_train.shape[0]}")
print(f"Tamanho do conjunto de teste: {X_test.shape[0]}")

# 2. Implementando k-fold cross-validation com XGBoost
k = 5  # número de folds
kf = KFold(n_splits=k, shuffle=True, random_state=22)

# Inicializando listas para armazenar métricas em cada fold
accuracies, recalls, precisions, f1_scores = [], [], [], []
confusion_matrices = []

for fold, (train_index,test_index) in enumerate(kf.split(df_normalized)):
    X_train_fold, X_test_fold = df_normalized.drop('classe',
                                axis=1).iloc[train_index],
                                df_normalized.drop('classe',
                                axis=1).iloc[test_index]
    y_train_fold, y_test_fold =
                        df_normalized['classe'].iloc[train_index],
                        df_normalized['classe'].iloc[test_index]
```

```
# Inicializa o modelo XGBoost
model = XGBClassifier(use_label_encoder=False,
        eval_metric='logloss')

# Treina o modelo
model.fit(X_train_fold, y_train_fold)

# Faz previsões para o conjunto de teste
y_pred_fold = model.predict(X_test_fold)

# Calcula as métricas
accuracy = accuracy_score(y_test_fold, y_pred_fold)
recall = recall_score(y_test_fold, y_pred_fold)
precision = precision_score(y_test_fold, y_pred_fold)
f1 = f1_score(y_test_fold, y_pred_fold)
conf_matrix = confusion_matrix(y_test_fold, y_pred_fold)

# Armazena as métricas e a matriz de confusão
accuracies.append(accuracy)
recalls.append(recall)
precisions.append(precision)
f1_scores.append(f1)
confusion_matrices.append(conf_matrix)

# Exibe as métricas do fold atual
print(f"\nFold {fold + 1}")
print(f"Acurácia: {accuracy * 100:.2f}%")
print(f"Recall: {recall * 100:.2f}")
print(f"Precisão: {precision * 100:.2f}")
print(f"F1-Score: {f1 * 100:.2f}")
```

```
# Calcula e exibe a média e o desvio padrão das métricas
print("\nMétricas Médias e Desvios Padrão:")
print(f"Acurácia Média:
  {np.mean(accuracies) * 100:.2f}% ± {np.std(accuracies) * 100:.2f}%")
print(f"Recall Médio:
  {np.mean(recalls) * 100:.2f} ± {np.std(recalls) * 100:.2f}")
print(f"Precisão Média:
  {np.mean(precisions) * 100:.2f} ± {np.std(precisions)* 100:.2f}")
print(f"F1-Score Médio:
  {np.mean(f1_scores) * 100:.2f} ± {np.std(f1_scores)* 100:.2f}")

# Soma das matrizes de conf. para análise combinada e conversão para int
conf_matrix_sum = sum(confusion_matrices).astype(int)

# Plotando a matriz de confusão combinada
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_sum, annot=True, fmt="d", cmap="Purples",
                                    xticklabels=['Galáxia', 'Estrela'],
                                    yticklabels=['Galáxia', 'Estrela'],
                                    annot_kws={"fontsize":20})
plt.xlabel("Classe Predita",fontsize=18)
plt.ylabel("Classe Real", fontsize=18)

plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.title("Matriz de Confusão - XGBoost", fontsize=16)

cbar = plt.gca().collections[0].colorbar
cbar.ax.tick_params(labelsize=15)

plt.show()
```

D.2 Random Forest

```
# Bibliotecas exportadas
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split, KFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
                     precision_score, recall_score, f1_score
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Normalização (entre 0 e 1)
scaler = MinMaxScaler()
df_normalized = pd.DataFrame(scaler.fit_transform(df),
    columns=df.columns)

# Exibir as primeiras linhas do DataFrame normalizado
print("Dados Normalizados:")
print(df_normalized.head())

# 1. Divisão em conjuntos de treino e teste
X_train, X_test, y_train, y_test =
                    train_test_split(df_normalized.drop('classe',
                    axis=1), df_normalized['classe'], test_size=0.2,
                    random_state=22)

# Exibindo tamanhos dos conjuntos
print(f"Tamanho do conjunto de treinamento: {X_train.shape[0]}")
```

```
print(f"Tamanho do conjunto de teste: {X_test.shape[0]}")

# 2. Implementando k-fold cross-validation com Random Forest
k = 5  # número de folds
kf = KFold(n_splits=k, shuffle=True, random_state=22)

# Listas para armazenar as métricas
accuracies = []
precisions = []
recalls = []
f1_scores = []
conf_matrices = np.zeros((2, 2))  # Matriz de confusão acumulada

# Loop K-Fold para Random Forest
for fold, (train_index,test_index) in enumerate(kf.split(df_normalized)):
    X_train_fold, X_test_fold = df_normalized.drop('classe',
                                axis=1).iloc[train_index],
                                df_normalized.drop('classe',
                                axis=1).iloc[test_index]
    y_train_fold,y_test_fold = df_normalized['classe'].iloc[train_index],
                                df_normalized['classe'].iloc[test_index]

    # Inicializa o modelo Random Forest
    model = RandomForestClassifier(n_estimators=100, random_state=22)

    # Treina o modelo
    model.fit(X_train_fold, y_train_fold)

    # Faz previsões para o conjunto de teste
    y_pred_fold = model.predict(X_test_fold)

    # Calcula as métricas
```

```
    accuracy = accuracy_score(y_test_fold, y_pred_fold)
    precision = precision_score(y_test_fold, y_pred_fold)
    recall = recall_score(y_test_fold, y_pred_fold)
    f1 = f1_score(y_test_fold, y_pred_fold)

    # Adiciona as métricas às listas
    accuracies.append(accuracy)
    precisions.append(precision)
    recalls.append(recall)
    f1_scores.append(f1)

    # Acumula a matriz de confusão
    fold_conf_matrix = confusion_matrix(y_test_fold, y_pred_fold)
    conf_matrices += fold_conf_matrix

    # Exibe as métricas do fold
    print(f"\nFold {fold + 1}:")
    print(f"Acurácia: {accuracy * 100:.2f}%")
    print(f"Precisão: {precision * 100:.2f}%")
    print(f"Completeza (Recall): {recall * 100:.2f}%")
    print(f"F1-Score: {f1 * 100:.2f}%")

    # Exibe os hiperparâmetros do modelo
    print("Hiperparâmetros do Random Forest para este fold:")
    print(model.get_params())

# Exibe a acurácia média dos folds
print(f"\nAcurácia Média dos {k} Folds:
    {np.mean(accuracies) * 100:.2f}%")
print(f"Precisão Média dos {k} Folds:
    {np.mean(precisions) * 100:.2f}%")
print(f"Completeza Média dos {k} Folds:
```

```
    {np.mean(recalls) * 100:.2f}%")
print(f"F1-Score Médio dos {k} Folds:
    {np.mean(f1_scores) * 100:.2f}%")

# Exibe a matriz de confusão acumulada
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrices, annot=True, fmt=".0f", cmap="Purples",
            xticklabels=['Galáxia', 'Estrela'], yticklabels=['Galáxia',
            'Estrela'], annot_kws={"fontsize": 20})
plt.xlabel('Classe Predita', fontsize=18)
plt.ylabel('Classe Real', fontsize=18)

# Ajusta o tamanho das labels dos eixos
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.title('Matriz de Confusão - Random Forest', fontsize=16)

# Ajusta o tamanho dos números na escala de cor
cbar = plt.gca().collections[0].colorbar
cbar.ax.tick_params(labelsize=15)

plt.show()
```

D.3 Regressão Logística

```
# Bibliotecas exportadas
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
```

```
from sklearn.model_selection import train_test_split, KFold
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,
    precision_score, recall_score, f1_score
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Normalização (entre 0 e 1)
scaler = MinMaxScaler()
df_normalized = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)

# Exibir as primeiras linhas do DataFrame normalizado
print("Dados Normalizados:")
print(df_normalized.head())

# 1. Divisão em conjuntos de treino e teste
X_train, X_test, y_train, y_test =
    train_test_split(df_normalized.drop('classe', axis=1),
    df_normalized['classe'], test_size=0.2, random_state=22)

# Exibindo tamanhos dos conjuntos
print(f"Tamanho do conjunto de treinamento: {X_train.shape[0]}")
print(f"Tamanho do conjunto de teste: {X_test.shape[0]}")

# 2. Implementando k-fold cross-validation com Regressão Logística
k = 5  # número de folds
kf = KFold(n_splits=k, shuffle=True, random_state=22)

# Listas para armazenar as métricas
accuracies = []
precisions = []
```

```
recalls = []
f1_scores = []
conf_matrices = np.zeros((2, 2))  # Matriz de confusão acumulada

# Loop K-Fold para Regressão Logística
for fold, (train_index, test_index) in enumerate(kf.split(df_normalized)):
    X_train_fold, X_test_fold = df_normalized.drop('classe',
            axis=1).iloc[train_index], df_normalized.drop('classe',
            axis=1).iloc[test_index]
    y_train_fold, y_test_fold = df_normalized['classe'].iloc[train_index],
            df_normalized['classe'].iloc[test_index]

    # Inicializa o modelo de Regressão Logística
    model = LogisticRegression(random_state=22, max_iter=1000)

    # Treina o modelo
    model.fit(X_train_fold, y_train_fold)

    # Faz previsões para o conjunto de teste
    y_pred_fold = model.predict(X_test_fold)

    # Calcula as métricas
    accuracy = accuracy_score(y_test_fold, y_pred_fold)
    precision = precision_score(y_test_fold, y_pred_fold)
    recall = recall_score(y_test_fold, y_pred_fold)
    f1 = f1_score(y_test_fold, y_pred_fold)

    # Adiciona as métricas às listas
    accuracies.append(accuracy)
    precisions.append(precision)
    recalls.append(recall)
    f1_scores.append(f1)
```

```
    # Acumula a matriz de confusão
    fold_conf_matrix = confusion_matrix(y_test_fold, y_pred_fold)
    conf_matrices += fold_conf_matrix

    # Exibe as métricas do fold
    print(f"\nFold {fold + 1}:")
    print(f"Acurácia: {accuracy * 100:.2f}%")
    print(f"Precisão: {precision * 100:.2f}%")
    print(f"Completeza (Recall): {recall * 100:.2f}%")
    print(f"F1-Score: {f1 * 100:.2f}%")

    # Exibe os hiperparâmetros do modelo
    print("Hiperparâmetros do modelo de Regressão Logística
            para este fold:")
    print(model.get_params())

# Exibe a acurácia média dos folds
print(f"\nAcurácia Média dos {k} Folds:
                          {np.mean(accuracies) * 100:.2f}%")
print(f"Precisão Média dos {k} Folds:
                          {np.mean(precisions) * 100:.2f}%")
print(f"Completeza Média dos {k} Folds:
                          {np.mean(recalls) * 100:.2f}%")
print(f"F1-Score Médio dos {k} Folds:
                          {np.mean(f1_scores) * 100:.2f}%")

# Exibe a matriz de confusão acumulada
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrices, annot=True, fmt=".0f", cmap="Purples",
    xticklabels=['Galáxia', 'Estrela'], yticklabels=['Galáxia',
    'Estrela'], annot_kws={"fontsize": 20})
```

```
plt.xlabel('Classe Predita', fontsize=18)
plt.ylabel('Classe Real', fontsize=18)

# Ajusta o tamanho das labels dos eixos
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.title('Matriz de Confusão - Logistic Regression', fontsize=16)

# Ajusta o tamanho dos números na escala de cor
cbar = plt.gca().collections[0].colorbar
cbar.ax.tick_params(labelsize=15)

plt.show()
```