Universidade Federal de Pernambuco
Centro de Informática
Programa de Pós-Graduação em Ciência da Computação
Curso de Mestrado

Entendimento dos Dados - MAGIC Gamma Telescope Dataset

Equipe: Giullia Braga, Luís Amaral, Manuel Ferreira, Thifany Souza

Disciplina: Aprendizagem de Máquina

---

1 Introdução

Após as etapas de Entendimento do Negócio e Entendimento dos Dados, ocorre a etapa de Preparação dos Dados, de acordo com as fases definidas na metodologia CRISP-DM [1]. Esta fase compreende a execução de fluxos de tarefas, dispostos na Figura 1. Estas tarefas incluem:

1. **Seleção dos dados:** etapa de decisão sobre os dados que serão utilizados para análise. É importante a certificação de que os dados são relevantes para o objetivo identificado na fase de Entendimento do Negócio. Nessa etapa, também serão listados os dados a serem excluídos e incluídos, além das razões para tais decisões.

2. **Limpeza de dados:** melhoramento da qualidade dos dados. Serão descritas as decisões e ações tomadas para mitigar os problemas de qualidade dos dados relatados durante a Verificação da Qualidade dos Dados na etapa de Entendimento dos Dados.

3. **Construção dos dados:** inclusão de atributos derivados, novos registros inteiros ou valores transformados para atributos existentes. Realização de métodos de codificação, especialmente para variáveis categóricas ou engenharia de atributos.

4. **Integração dos dados:** combinação de dados de múltiplas tabelas ou registros para criação de novos registos ou valores. Conhecimentos e habilidade em manipulação de banco de dados são relevantes nessa fase.

5. **Formatação dos dados:** etapa de transformação dos dados, porém sem necessariamente mudar seu significado, conforme necessário para a ferramenta de modelagem. Exemplos incluem a transformação dos dados por meio de padronização ou normalização.

Com base nas observações realizadas na etapa anterior, que focaram na identificação das variáveis e suas propriedades, é essencial avançar para uma análise mais detalhada dos dados gerados pelas simulações com um telescópio Cherenkov. A etapa de preparação dos dados desempenha um papel crucial no sucesso das fases subsequentes do projeto, pois é nessa fase que se garante a qualidade e a usabilidade do conjunto de dados para a modelagem preditiva.

A limpeza e manipulação de dados, elementos centrais desta etapa, visam identificar e tratar valores atípicos, inconsistências ou ausências que possam comprometer a robustez do modelo final. Essa abordagem conecta-se diretamente ao trabalho anterior de entendimento dos dados, uma vez que uma visão clara das variáveis e suas distribuições é o ponto

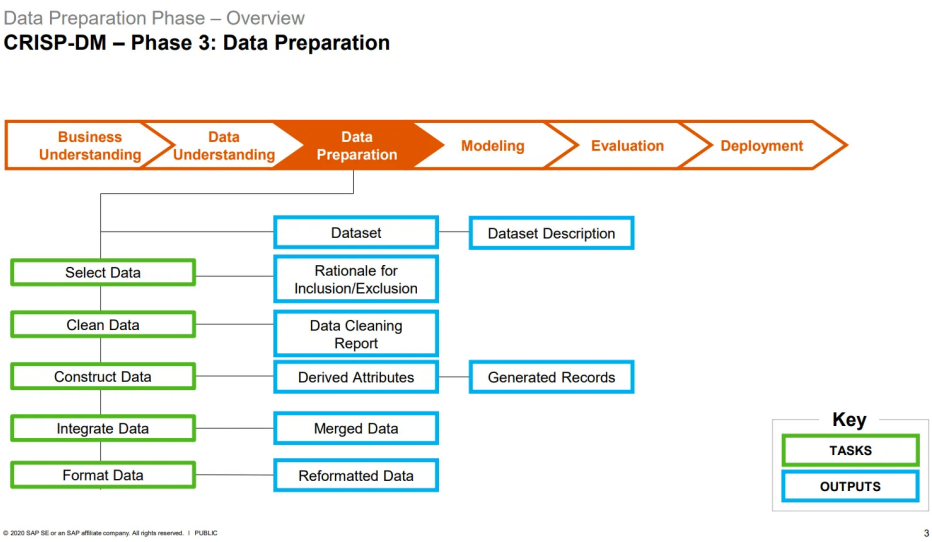

Figura 1: Fluxograma detalhando a etapa de Preparação dos Dados. Fonte: [1].

de partida para ações como a detecção e o tratamento de outliers, a imputação de valores faltantes e a aplicação de transformações que otimizem a representação das informações.

Nessa etapa, diferentes abordagens para tratar os dados serão testadas e avaliadas em modelos genéricos, permitindo identificar quais estratégias de preparação trazem os melhores resultados para a etapa de modelagem final. Essa prática não apenas assegura uma maior confiabilidade na escolha das técnicas de manipulação, mas também melhora a tomada de decisão ao basear as escolhas em evidências experimentais, otimizando o desempenho do modelo preditivo.

Além disso, ao explorar as distribuições e relações entre variáveis, bem como identificar padrões relevantes, busca-se garantir que os dados sejam adequados para o objetivo principal: desenvolver um modelo de classificação eficaz para distinguir partículas gama de hádrons com base nas características das imagens. A preparação detalhada dos dados não apenas reduz os riscos de vieses e erros na análise, mas também fortalece a capacidade do modelo de gerar insights relevantes e confiáveis, contribuindo significativamente para avanços no campo da astrofísica.

Esta etapa, portanto, é uma ponte essencial entre a compreensão inicial dos dados e a modelagem avançada, assegurando que as etapas futuras se baseiem em informações limpas, precisas e bem estruturadas, e otimizadas com base em testes exploratórios e validações preliminares.

2 Revisão das Propostas de Alterações

Na etapa anterior, foram elaboradas propostas de alterações dos dados utilizados no presente estudo. Estas indicação de alteração são:

1. **Remoção de duplicatas:** na análise exploratória foram observados 115 registros duplicados. Com o objetivo de evitar redundâncias, esses registros deveriam ser removidos.

2. **Tratamento de *outliers*** Foi observado que variáveis como $fAsym$, $fWidth$ e $fM3Long$ apresentaram uma quantidade considerável de *outliers*, o que pode im-

pactar negativamente o desempenho do modelo. Para lidar com esse problema, propõe-se a aplicação de transformações matemáticas que ajudem a comprimir os valores extremos, reduzindo sua influência sem eliminar informações importantes. Entre as possibilidades, destaca-se a winsorização, que limita a magnitude dos *outliers* a um intervalo pré-definido, preservando a estrutura geral dos dados. Essas abordagens buscam reduzir ruídos, melhorar a estabilidade e garantir maior robustez ao modelo preditivo.

3. **Normalização:**Variáveis como $fLength$, $fWidth$ e $fSize$, que possuem amplitudes muito diferentes, devem passar por um processo de normalização. Dessa forma, se busca uniformizar as escalas das variáveis, além de evitar que uma domine a outra durante o treinamento.

4. **Aplicação de PCA:** de forma a lidar com questões de dimensionalidade e multicolinearidade pode ser uma alternativa interessante, visto que existem variáveis altamente correlacionadas. Sendo assim, será comparada a modelagem com e sem o PCA de forma a entender se esta técnica possibilitou resultados mais robustos.

5. **Codificação numérica da variável de classe:** a fim de garantir a compatibilidade com os modelos utilizados, a classe gama "g"será codificada como 1 e hádrons "h", 0.

Com isso, o intuito das alterações propostas visou otimizar o desempenho do modelo, aumentar a robustez e garantir que os resultados fossem mais consistentes e interpretáveis.

3 Implementação das Alterações

Para definir a ordem como são feitas as transformações nos dados, este projeto se baseia no livro "*Data Preprocessing in Data Mining* [2]. Segundo ele, o pré-processamento dos dados deve seguir uma sequência lógica para garantir resultados confiáveis. O primeiro passo é a limpeza de dados, que envolve a remoção de duplicatas, tratamento de valores ausentes e inconsistências, garantindo uma base de dados coerente e de qualidade para as etapas subsequentes [2, p. 45]. Em seguida, é essencial detectar e lidar com *outliers*, pois estes podem distorcer medidas estatísticas como média e variância, impactando negativamente os processos de normalização e redução de dimensionalidade.[2, p. 45]. Após a limpeza de *outliers*, aplica-se a normalização para ajustar as escalas dos atributos, assegurando que todos contribuam igualmente na análise, algo fundamental para fazer transformações consistentes nas features, como a Análise do Componente Principal (PCA) [2, p. 179]. Por fim, a técnica de PCA pode ser implementada para redução de dimensionalidade, beneficiando-se de dados previamente limpos e normalizados.

É fundamental ressaltar que, neste primeiro momento, estamos analisando todo o conjunto de dados, buscando entender as características da base. Porém, em etapas posteriores, as transformações escolhidas serão refeitas, somente no conjunto de treino e validação, para preservar o de teste.

3.1 Importações e configurações gerais

Foram feitas as seguintes importações e definidas as configurações gerais:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats.mstats import winsorize
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split, cross_val_score, StratifiedKFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.utils import resample

# Configuracoes gerais
RANDOM_STATE = 42
sns.set(style="whitegrid")
```

3.2 Tratamento de duplicatas

Para remoção de duplicatas, utilizou-se o seguinte código:

```python

def load_and_clean_data(filepath):
    data = pd.read_csv(filepath)
    data['class'] = data['class'].replace({'g': 1, 'h': 0})
    num_rows_before = data.shape[0]
    data.drop_duplicates(inplace=True)
    num_rows_after = data.shape[0]
    num_duplicates_removed = num_rows_before - num_rows_after
    print(f"Numero de linhas duplicadas removidas: {num_duplicates_removed}")
    return data


# Caminho para o arquivo de dados
file_path = "/magic.csv"

# Carregar e limpar os dados
data = load_and_clean_data(file_path)
```

Além da remoção de 115 observações duplicatas, o código acima também realizou a substituição de 'g' por 1, e 'h' por 0 na feature "*class*".

A análise descritiva da tabela 1 mostra que $fLength$, $fDist$ e outras features apresentam altos valores médios e desvios padrão consideráveis, indicando maior variabilidade, enquanto outras como $fSize$ possuem distribuições mais concentradas com valores médios baixos, sugerindo maior estabilidade. $fAlpha$ e $fDist$ apresentam médias e quartis bem definidos, sugerindo maior potencial de discriminação. A variável *class*, com média de 0.65 e quartis distintos, reflete o desbalanceamento das classes. Esta análise destaca a necessidade de normalização ou transformação dadas as variáveis com alta dispersão.

Além disso, foi possível observar a distribuição das variáveis contínuas da base original, após a remoção de duplicatas, como mostra a Figura 2

Os histogramas indicam que variáveis como $fAlpha$ e $fM3Long$ apresentam diferenças mais evidentes entre as classes, sugerindo bom poder discriminativo. Por outro lado,

Tabela 1: Estatísticas descritivas da base original tratada, após remoção de duplicatas.

	count	mean	std	min	25%	50%	75%	max
fLength	18905.00	53.16	42.26	4.28	24.36	37.13	69.98	334.18
fWidth	18905.00	22.15	18.30	0.00	11.87	17.14	24.71	256.38
fSize	18905.00	2.83	0.47	1.94	2.48	2.74	3.10	5.32
fConc	18905.00	0.38	0.18	0.01	0.24	0.35	0.50	0.89
fConc1	18905.00	0.22	0.11	0.00	0.13	0.20	0.29	0.68
fAsym	18905.00	-4.18	59.01	-457.92	-20.48	4.06	24.13	575.24
fM3Long	18905.00	10.62	50.90	-331.78	-12.77	15.34	35.87	238.32
fM3Trans	18905.00	0.26	20.78	-205.89	-10.84	0.75	10.95	179.85
fAlpha	18905.00	27.55	26.08	0.00	5.52	17.53	45.70	90.00
fDist	18905.00	193.71	74.69	1.28	142.27	191.83	240.41	495.56
class	18905.00	0.65	0.48	0.00	0.00	1.00	1.00	1.00

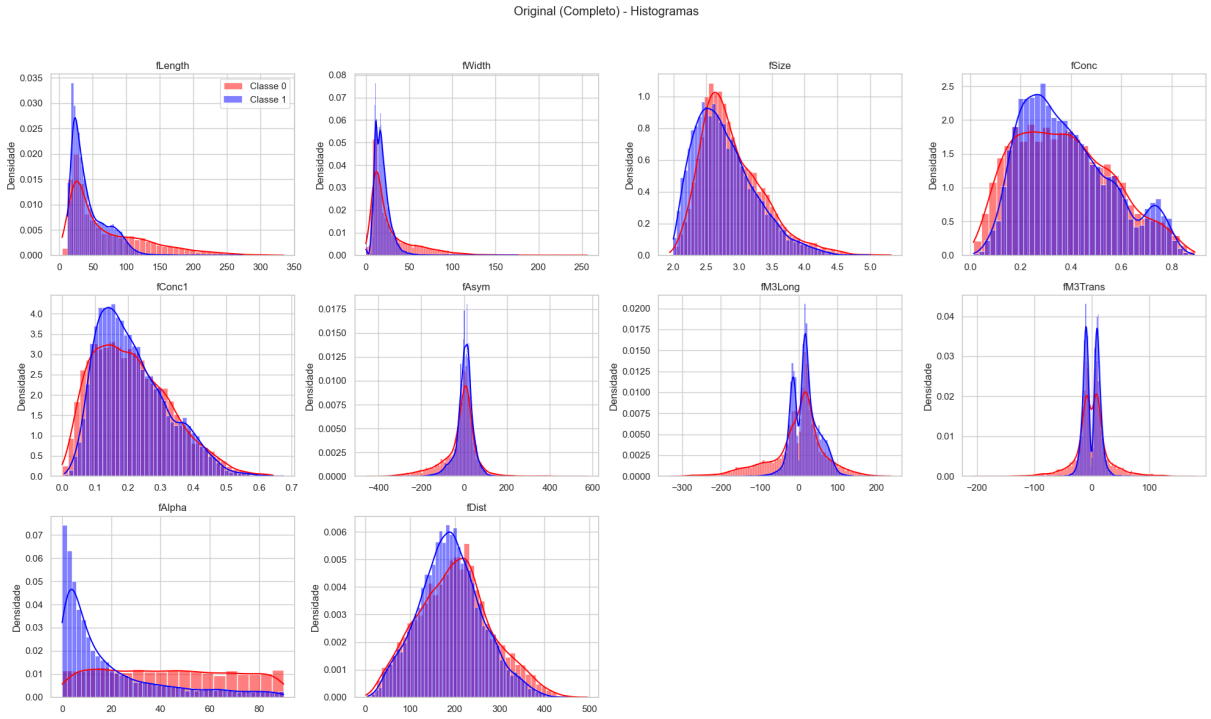

Figura 2: Histograma da distribuição das variáveis da base tratada, por classe.

variáveis como $fAsym$ têm distribuições quase idênticas entre as classes, indicando baixa utilidade para separação. Variáveis como $fLength$, $fWidth$, $fSize$ e $fDist$ mostram alguma sobreposição, mas com diferenças que podem ser suficientes para possivelmente contribuírem na classificação. Sendo assim, essa análise inicial ajuda a identificar quais variáveis são mais relevantes para o modelo preditivo.

Os boxplots da Figura 3 complementam a análise prévia mostrando a distribuição das variáveis entre as classes e destacando *outliers*. Entre as features, percebe-se que $fAlpha$ é a que apresentam maior diferença entre as medianas, reforçando que esta variável pode ser a que mais contribuirá para classificação.

Assim, seguindo a metodologia CRISP-DM e as orientações de Herrera, Luengo e García (2015)[2]; no próximo passo será abordado o tratamento de *outliers*.

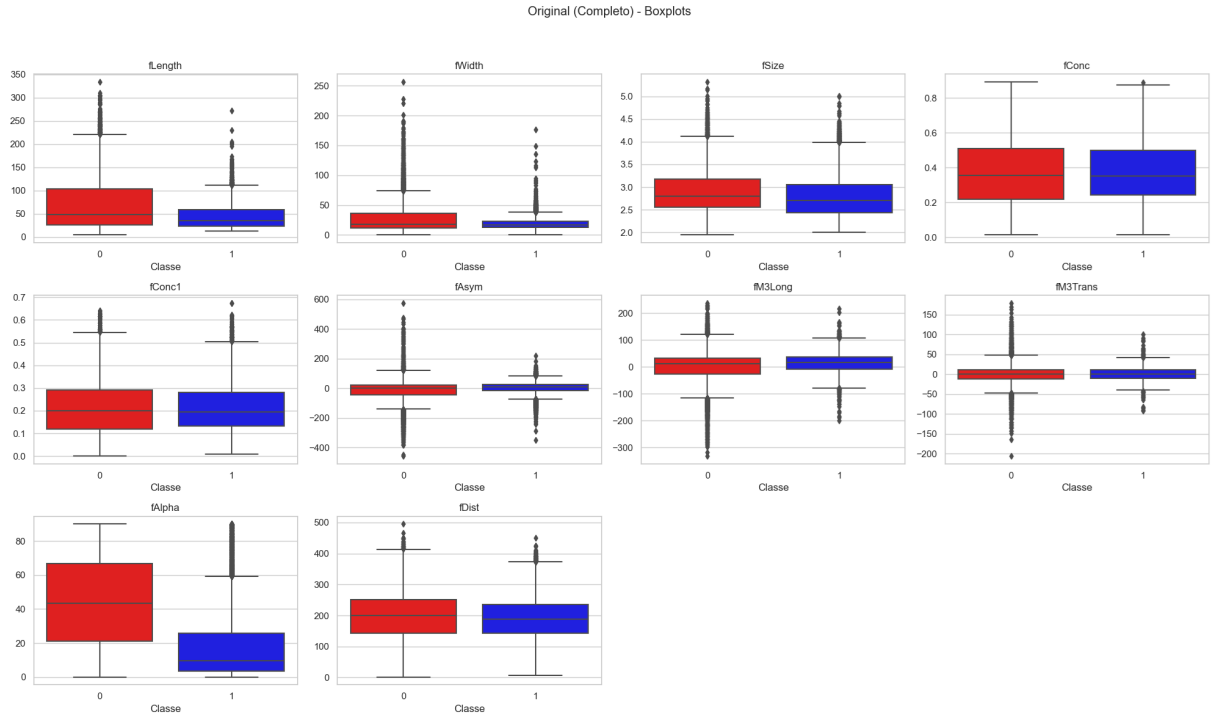

Figura 3: Boxplot da distribuição das variáveis da base tratada, por classe.

3.3 Tratamento de *outliers*

Os *outliers* são observações que destoam da dinâmica geral da série e podem distorcer suas características, afetando negativamente as análises. Na etapa anterior, foi avaliada a aplicação de transformações matemáticas como forma de mitigar esse problema. Entre elas, a transformação logarítmica foi considerada, pois aproxima os valores extremos da média da distribuição. No entanto, essa abordagem é inadequada para bases que contêm variáveis com valores negativos, como é o caso do conjunto de dados analisado.

Outra alternativa seria a remoção dos *outliers*. Embora essa técnica elimine a influência direta de valores extremos, ela é considerada menos rigorosa, pois descarta informações que podem ser relevantes para o entendimento do fenômeno em análise. Além disso, a exclusão de dados reduz o tamanho da amostra, comprometendo a representatividade.

Diante disso, optou-se pela winsorização, que é uma técnica de acomodação dos dados. Diferente de métodos que removem ou ignoram *outliers*, a winsorização busca integrar essas observações ao conjunto, mas reduzindo seu impacto. Essa abordagem substitui os valores extremos por limites definidos com base em percentis da distribuição, como o 5º e o 95º percentil. Por exemplo, valores abaixo do 5º percentil são ajustados para o limite inferior, enquanto valores acima do 95º percentil são ajustados para o limite superior. Isso permite manter a estrutura do conjunto de dados, minimizando a influência de *outliers* sem excluir observações. Neste estudo, serão testadas a winsorização a 5% e a 1% para identificar qual configuração melhor atende às necessidades da base proposta, garantindo um equilíbrio entre robustez estatística e preservação da integridade dos dados.

3.3.1 Winsorização a 1%

```
1 """
```

```
Para otimizar o processo, criamos um dicionario com as bases, para em
    seguida plotar os histogramas. Assim, as primeiras linhas do codigo
    abaixo se referem a essa otimizacao.
"""

bases_completas = {}

# Lista de colunas num ricas (excluindo 'class')
numeric_cols = data.select_dtypes(include=[np.number]).columns.drop('
    class')

winsor_limits_1 = (0.01, 0.01)
data_winsor_1 = data.copy()
for col in numeric_cols:
    data_winsor_1[col] = winsorize(data_winsor_1[col], limits=
    winsor_limits_1)
bases_completas['Winsorizado 1%'] = data_winsor_1
```

Como resultado da etapa de winsorização a 1%, tem-se a tabela 2.

Tabela 2: Estatísticas descritivas da base após Winsorização 1%

	count	mean	std	min	25%	50%	75%	max
fLength	18905.00	52.88	40.95	12.73	24.36	37.13	69.98	215.54
fWidth	18905.00	21.92	16.74	3.22	11.87	17.14	24.71	101.18
fSize	18905.00	2.82	0.47	2.08	2.48	2.74	3.10	4.20
fConc	18905.00	0.38	0.18	0.07	0.24	0.35	0.50	0.81
fConc1	18905.00	0.21	0.11	0.04	0.13	0.20	0.29	0.50
fAsym	18905.00	-4.42	53.96	-221.66	-20.48	4.06	24.13	116.71
fM3Long	18905.00	10.78	48.30	-168.47	-12.77	15.34	35.87	126.09
fM3Trans	18905.00	0.26	18.55	-61.87	-10.84	0.75	10.95	65.68
fAlpha	18905.00	27.54	26.06	0.15	5.52	17.53	45.70	88.30
fDist	18905.00	193.59	73.88	39.41	142.27	191.83	240.41	376.81
class	18905.00	0.65	0.48	0.00	0.00	1.00	1.00	1.00

A comparação entre as tabelas mostra que a winsorização a 1% (1% inferior e superior, ou seja, 2% do conjunto de dados.) reduziu os valores extremos, substituindo-os pelos limites dos percentis 1% e 99%, o que diminuiu o impacto de *outliers* sem alterar significativamente os quartis ou a mediana. Como resultado, as médias sofreram pequenas mudanças, enquanto os desvios padrão reduziram em variáveis como $fAsym$ e $fM3Long$, indicando menor dispersão. Assim, essa transformação aparenta preservar a estrutura central dos dados. Uma análise adicional pode ser feita observando os histogramas da Figura 4 e os boxplots da Figura 5 após a transformação.

Nos boxplots pós-winsorização observa-se que os "whiskers"estão mais próximos, e a presença de *outliers* visíveis diminuiu consideravelmente, especialmente para variáveis como *fWidth*, *fAsym* e *fM3Long*.

Os histogramas mostram que as distribuições gerais permanecem consistentes, preservando as características principais das variáveis, enquanto as caudas foram truncadas para mitigar a influência de valores extremos. Como foi explicado, essa abordagem mantém a integridade estatística dos dados ao minimizar o viés que poderia ser introduzido pelos *outliers*, favorecendo análises mais robustas. No entanto, a winsorização pode esconder padrões ou tendências interessantes associadas a esses valores extremos, sendo necessário

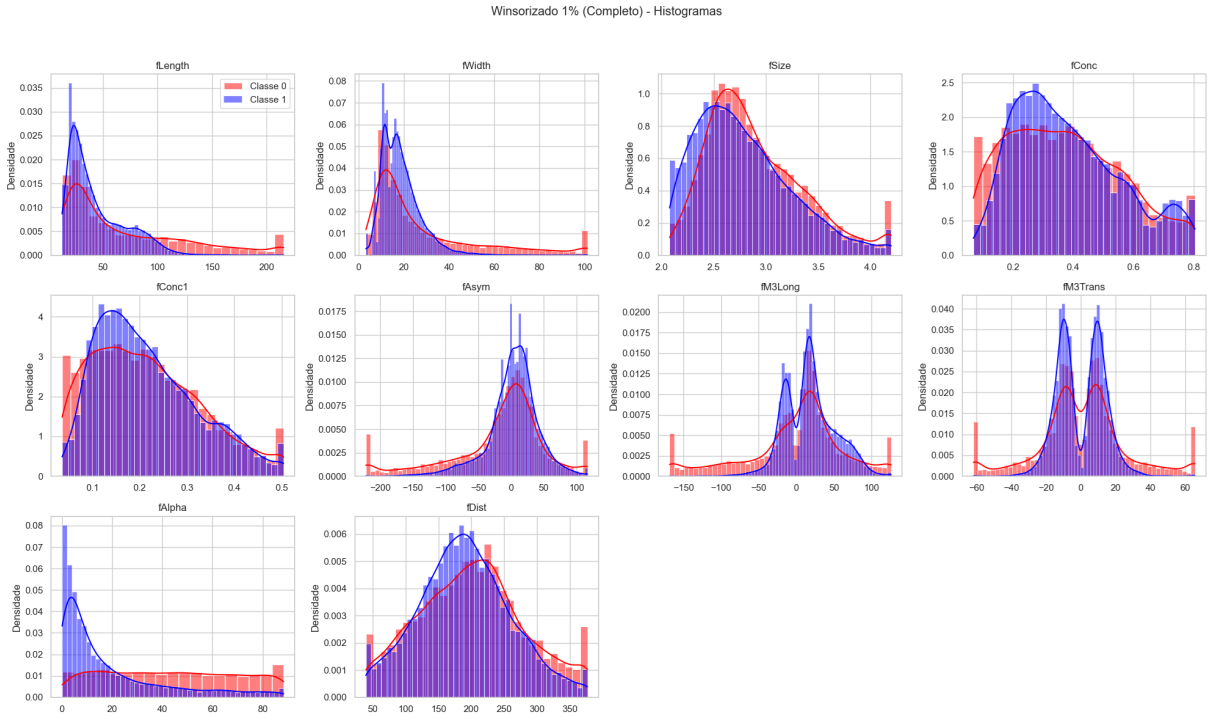

Figura 4: Histograma da distribuição das variáveis da base, após a winsorização a 1%.

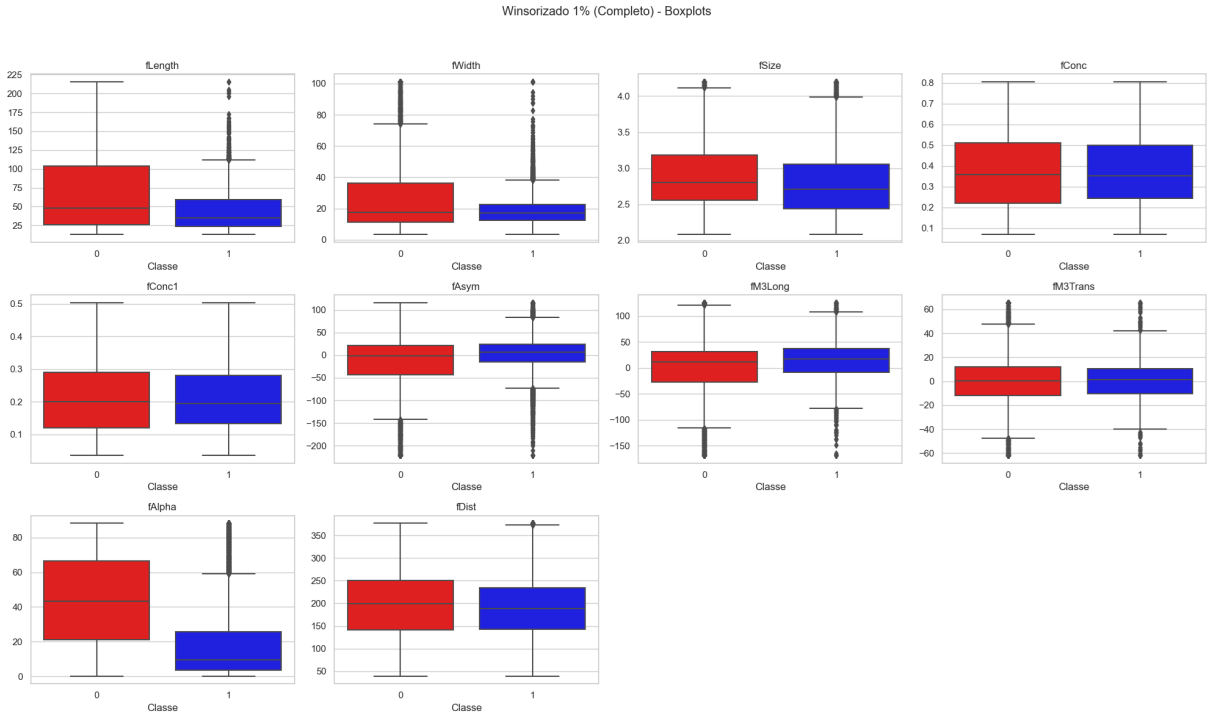

Figura 5: Boxplot da distribuição das variáveis da base, após a winsorização a 1%.

avaliar a pertinência dessa técnica. Como há indícios de que foi uma boa abordagem, seguiu-se com ela. Porém, na etapa de validação, a base transformada e a original serão comparadas afim de entender a real melhora no conjunto de dados.

3.3.2 Winsorização a 5%

```
winsor_limits_5 = (0.05, 0.05)
data_winsor_5 = data.copy()
for col in numeric_cols:
    data_winsor_5[col] = winsorize(data_winsor_5[col], limits=
   winsor_limits_5)
bases_completas['Winsorizado 5%'] = data_winsor_5
```

A winsorização a 5% (que ajusta 10% da base, delimitando os valores entre os percentis 5% e 95%) apresentou resultados mais problemáticos. Pela análise dos histogramas da Figura 6, observa-se limites mais amplos geraram truncamentos significativos nos extremos da distribuição, alterando de forma significante o padrão original dos dados. Considerando essas observações, na etapa de validação, a winsorização a 5% será incluída para comparação, mas os resultados preliminares sugerem que essa técnica não é a mais adequada para esta base de dados. Dado o impacto mais controlado e consistente da winsorização a 1%, optou-se por priorizá-la no desenvolvimento das análises.

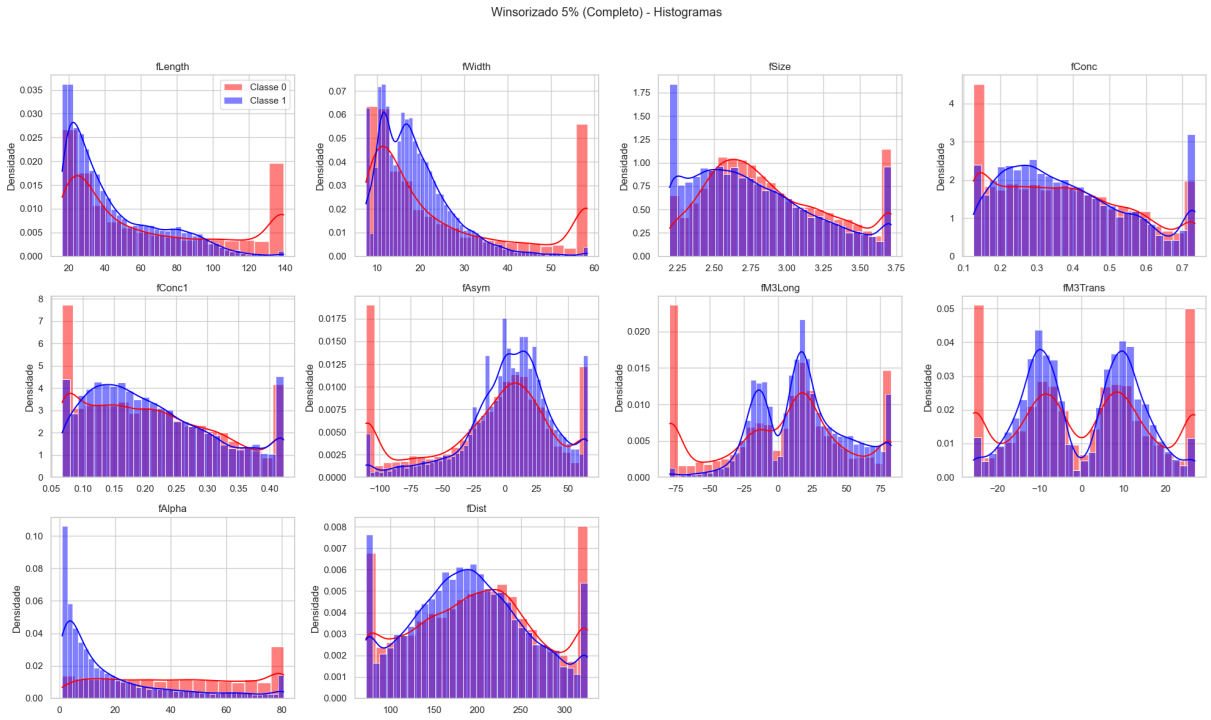

Figura 6: Histograma da distribuição das variáveis da base, após a winsorização a 5%.

3.4 Normalização

A normalização dos dados é fundamental para garantir que todas as variáveis contribuam de forma equilibrada para análises e algoritmos de aprendizado de máquina, independentemente de suas escalas originais. Como destacado no livro "*Data Preprocessing in Data Mining*"[2], diferenças significativas entre as escalas podem distorcer os resultados de métodos que dependem de distâncias ou correlações, como o K-means. Além disso, a normalização ajuda a mitigar a influência de valores extremos e a melhorar a convergência de algoritmos de otimização, assegurando resultados mais precisos e confiáveis.

Assim, após a limpeza da base e winsorização a 1% para tratar *outliers*, foi realizada a normalização dos dados.

```
data_winsor_1_norm = data_winsor_1.copy()
scaler_winsor_1 = MinMaxScaler()
data_winsor_1_norm[numeric_cols] = scaler_winsor_1.fit_transform(
    data_winsor_1_norm[numeric_cols])
bases_completas['Normalizado + Winsorizado 1%'] = data_winsor_1_norm
```

Como outputs novamente, serão avaliadas as estatísticas descritivas da Tabela 3, os histogramas da Figura 7 e os boxplots da Figura 8 dos dados normalizados.

Tabela 3: Estatísticas descritivas da base após normalização e winsorização a 1%

	count	mean	std	min	25%	50%	75%	max
fLength	18905.00	0.20	0.20	0.00	0.06	0.12	0.28	1.00
fWidth	18905.00	0.19	0.17	0.00	0.09	0.14	0.22	1.00
fSize	18905.00	0.35	0.22	0.00	0.19	0.31	0.48	1.00
fConc	18905.00	0.42	0.25	0.00	0.23	0.39	0.59	1.00
fConc1	18905.00	0.38	0.23	0.00	0.20	0.34	0.53	1.00
fAsym	18905.00	0.64	0.16	0.00	0.60	0.67	0.73	1.00
fM3Long	18905.00	0.61	0.16	0.00	0.53	0.62	0.69	1.00
fM3Trans	18905.00	0.49	0.15	0.00	0.40	0.49	0.57	1.00
fAlpha	18905.00	0.31	0.30	0.00	0.06	0.20	0.52	1.00
fDist	18905.00	0.46	0.22	0.00	0.31	0.45	0.60	1.00
class	18905.00	0.65	0.48	0.00	0.00	1.00	1.00	1.00

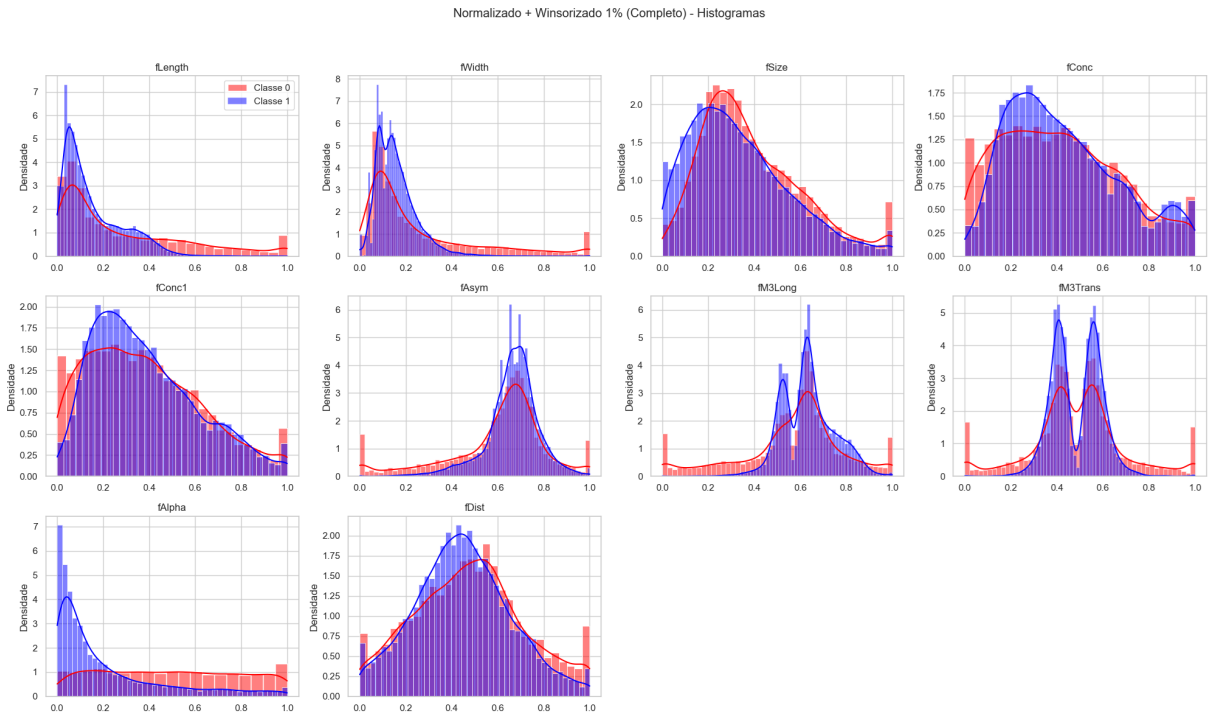

Figura 7: Histograma da distribuição das variáveis da base, após a winsorização a 1% e normalização.

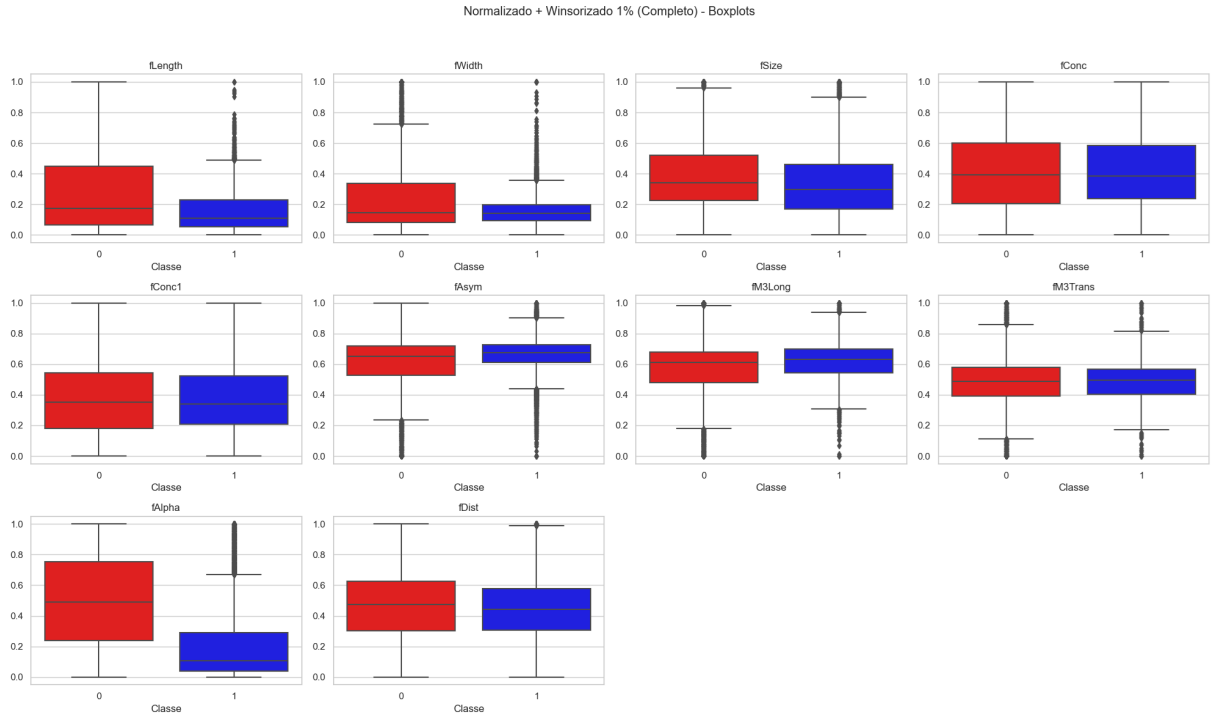

Figura 8: Boxplot da distribuição das variáveis da base, após a winsorização a 1% e normalização.

Na base final, após o processamento e limitação dos *outliers*, todos os valores foram reescalados para o intervalo [0, 1], permitindo uma padronização comparável entre variáveis. A variável $fAsym$, por exemplo, passou a ter valores entre 0,00 e 1,00, com uma média de 0,64 e um desvio padrão de 0,16, evidenciando uma compressão tanto da amplitude quanto da variabilidade (antes tinha um desvio padrão de 59,01 e valores variando de -457,92 até 575,24.).

Esse reescalonamento garante uma distribuição mais uniforme, eliminando o impacto desproporcional de variáveis originalmente mais amplas, porém sem alterar caraterísticas fundamentais da série.

3.5 Análise do PCA

Como observado na etapa anterior, há indícios de multicolinearidade nos dados, o que pode acarretar em viéses e problemas de dimensionalidade. Isso é observado através do *heatmap* apresentado na etapa anterior, que é novamente introduzido na Figura 9. Uma das formas de tratamento adicional que pode ser feita para minimizar essa questão é o *Principal Component Analysis* (PCA).

O PCA é uma técnica de redução de dimensionalidade amplamente utilizada em análise de dados, pois transforma variáveis originais em um conjunto de componentes principais não correlacionados, que são combinações lineares das variáveis originais. Esses componentes são ordenados de forma que o primeiro retém a maior variância dos dados, seguido pelos subsequentes. Essa abordagem reduz a dimensionalidade enquanto preserva o máximo possível de informação relevante. Sua importância é devido a habilidade de simplificar conjuntos de dados complexos, facilitando a visualização e interpretação. Além disso, o PCA auxilia na remoção de redundâncias e ruídos nos dados, melhorando a

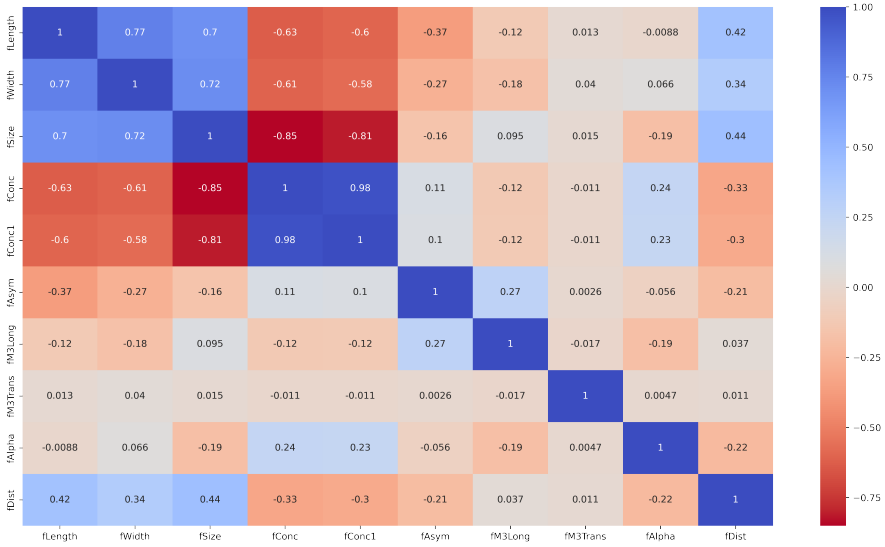

Figura 9: *Heatmap* das variáveis da base

eficiência de algoritmos de aprendizado de máquina ao trabalhar com um número reduzido de características.

Para implementar o PCA, colocou-se como parâmetros 95%. Ou seja, os componentes devem ser capazes de explicar 95% das variações nos dados.

```
1
2  def apply_pca(data, n_components=0.95):
3      features = data.drop(columns=['class'])
4      pca = PCA(n_components=n_components, random_state=RANDOM_STATE)
5      pca_features = pca.fit_transform(features)
6      pca_feature_names = [f'PC{i+1}' for i in range(pca_features.shape
       [1])]
7      data_pca = pd.DataFrame(pca_features, columns=pca_feature_names)
8      data_pca['class'] = data['class'].reset_index(drop=True)
9      return data_pca
10
11     # PCA na base normalizada e winsorizada a 1%
12 bases_completas['PCA Normalizado + Winsorizado 1%'] = apply_pca(
       data_winsor_1_norm)
```

Novamente, tem-se como outputs as estatísticas descritivas na Tabela 4, os histogramas na Figura 10 e os boxplots na Figura 11 dos dados winsorizados, normalizados e com PCA.

Após a aplicação do PCA em conjunto com a normalização e winsorização a 1%, a base de dados sofreu transformações significativas. O histograma apresenta distribuições mais uniformes para os componentes, quando comparados as classes 0 e 1, além de maior simetria. Já os boxplots pós todas as transformações mostram maior uniformidade entre os componentes, com reduções substanciais nos "whiskers"e eliminação de *outliers* extremos.

Porém, um problema relacionado ao PCA é a mudança na quantidade de features, sendo impossível modelar sem aplicar também essas alterações no conjunto de treino e teste. Essa abordagem, apesar de promissora, pode trazer problemas para a correta classificação dos modelos.

Tabela 4: Estatísticas descritivas da base após PCA, Normalização e Winsorização a 1%

	count	mean	std	min	25%	50%	75%	max
PC1	18905.00	0.00	0.46	-1.32	-0.34	0.02	0.34	1.06
PC2	18905.00	0.00	0.30	-0.43	-0.23	-0.12	0.19	1.24
PC3	18905.00	0.00	0.22	-0.71	-0.14	-0.01	0.13	1.04
PC4	18905.00	0.00	0.18	-1.00	-0.10	0.00	0.12	0.73
PC5	18905.00	0.00	0.15	-0.53	-0.09	0.01	0.09	0.55
PC6	18905.00	0.00	0.13	-0.55	-0.07	-0.01	0.06	0.97
PC7	18905.00	0.00	0.12	-0.47	-0.07	-0.01	0.06	0.83
class	18905.00	0.65	0.48	0.00	0.00	1.00	1.00	1.00

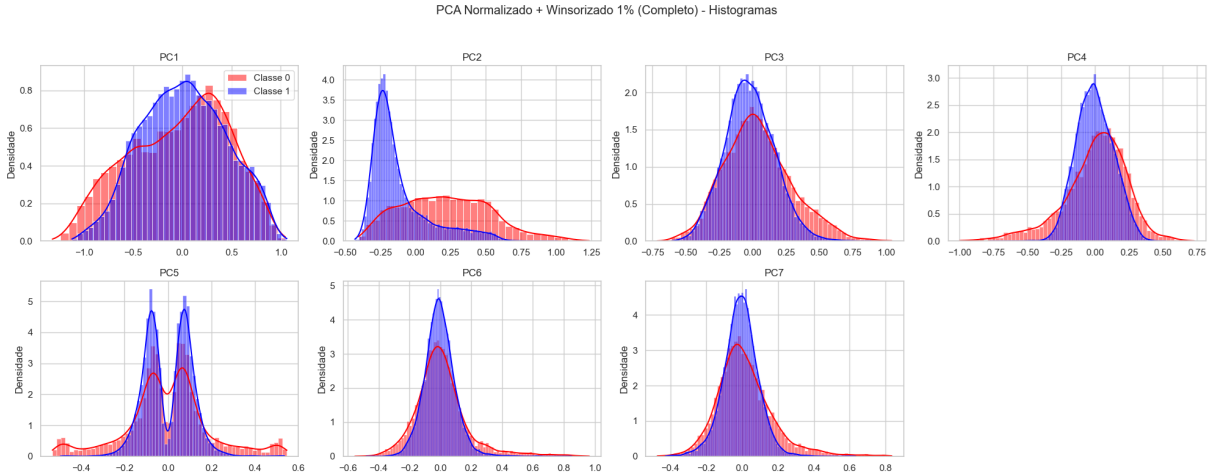

Figura 10: Histograma da distribuição das variáveis da base, após PCA, winsorização a 1% e normalização.

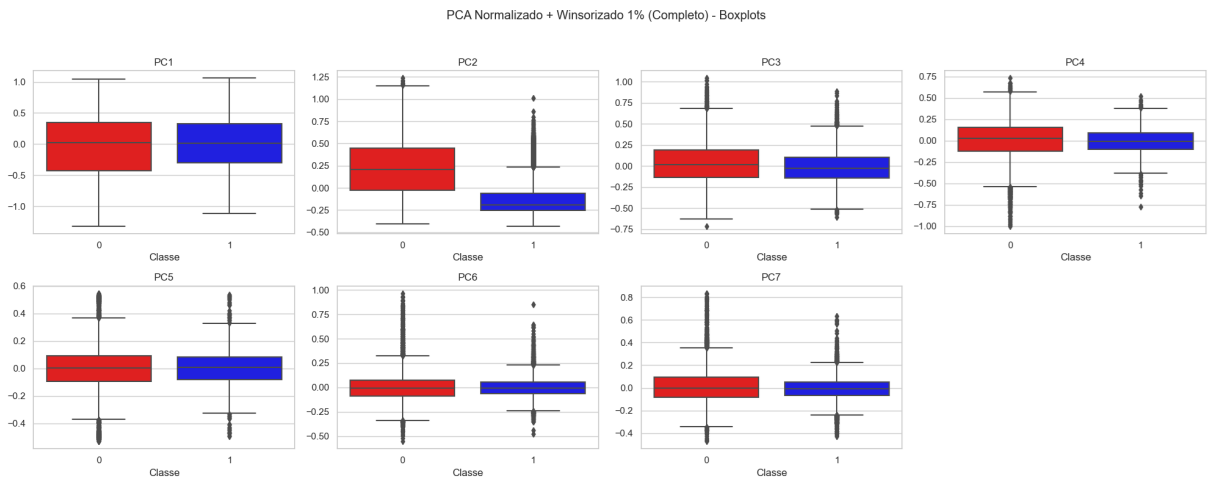

Figura 11: Boxplot da distribuição das variáveis da base, após PCA, winsorização a 1% e normalização.

Sendo assim, dadas as considerações apresentadas na análise exploratória, **acredita-se que a base mais robusta será a que passou pela limpeza dos dados, winsorização de 1% e normalização, sem o uso do pca**, que será testada na etapa de validação.

4 Resultado da preparação dos dados.

A preparação dos dados buscou melhorar a qualidade e usabilidade do conjunto, por meio de ações como winsorização, normalização e, em parte do estudo, PCA. Os objetivos centrais eram reduzir o impacto de *outliers*, uniformizar as escalas e avaliar a relevância da redução dimensional para a construção de modelos preditivos.

A Tabela 1, correspondente à base original após remoção de duplicatas, exibe valores altamente dispersos em diversas variáveis, como $fLength$ (min=4.28, max=334.18, std=42.26) e $fAsym$ (min=-457.92, max=575.24, std=59.01). Essa dispersão reflete a presença significativa de *outliers* e escalas variadas, que podem prejudicar o desempenho e estabilidade de modelos. Em contraste, a Tabela 3 apresenta os dados após winsorização a 1% e normalização. Nesse caso, todas as variáveis foram ajustadas ao intervalo [0, 1], reduzindo a dispersão (std para $fLength$ e $fAsym$ passaram para 0.20 e 0.16, respectivamente) e preservando a estrutura relativa das distribuições.

A comparação dos histogramas (Figuras 2 e 7) evidencia as mudanças nas distribuições: enquanto os dados originais exibiam caudas longas e sobreposições amplas entre as classes, especialmente em variáveis como $fAlpha$ e $fDist$, os dados transformados mostram caudas truncadas e maior concentração em torno das médias. Já os bloxplots (Figuras 3 e 8) confirmam uma redução significativa na presença de *outliers* visíveis, particularmente em $fAsym$, $fM3Long$, entre outras.

Embora o PCA, conforme observado na Tabela 4, tenha sido útil para reduzir a dimensionalidade e eliminar redundâncias, a interpretação dos componentes principais apresenta desafios. Cada componente é uma combinação linear das variáveis originais, dificultando a associação direta com características específicas do fenômeno analisado. Além disso, seu uso implicaria na necessidade de transformação do conjunto de testes, o que não é a estratégia mais adequada.

Optou-se, portanto, por não seguir com o PCA no conjunto final. A base que passou por winsorização a 1% e normalização mostrou-se mais robusta e interpretável, além de preservar as relações originais das variáveis. Essa abordagem equilibra a necessidade de controle de *outliers* com a integridade dos dados, otimizando o uso das informações sem a perda de interpretabilidade. Este conjunto será utilizado nas etapas posteriores para treinar e validar os modelos preditivos. Porém, a partir da base inicial limpa, será primeiro dividido o conjunto de treino, validação e teste, e as transformações: winsorização a 1% e normalização, serão aplicadas apenas ao primeiro e ao segundo.

5 Conjunto de Treinamento e Validação

A divisão do conjunto de dados para o treinamento (70%), validação (15%) e teste (15%) foi realizada com o objetivo de garantir que as análises e modelagens subsequentes sejam generalizáveis e robustas.

Para o cojunto de validação, optou-se por separar 15% dos dados, que foram winsorizados a 1% e normalizados com a escala estimada no conjunto de treinamento. Adicionalmente usaremos a estratégia de *k-fold cross-validation* estratificada, uma técnica amplamente recomendada para conjuntos de dados balanceados ou com leve desbalanceamento de classes. Essa abordagem divide os dados em k subconjuntos, onde cada um é utilizado como conjunto de validação exatamente uma vez, enquanto os outros $k-1$ subconjuntos são usados para o treinamento do modelo, assegurando a preservação da

distribuição das classes em cada partição.

Assim, na etapa de divisão e validação, a primeira ação foi dividir, a partir da base original com remoção de duplicatas, os conjutos propostos.

```
train_data, temp_data = train_test_split(data, test_size=0.30,
    random_state=RANDOM_STATE)

validation_data, test_data = train_test_split(
    temp_data, test_size=0.5, random_state=RANDOM_STATE
)

print(f"Tamanho do conjunto de treino: {len(train_data)}")
print(f"Tamanho do conjunto de valida  o: {len(validation_data)}")
print(f"Tamanho do conjunto de teste: {len(test_data)}")

"""
Novamente, criamos um dicion rio para armazenar as bases de treino,
    pensando na otimiza  o do nosso c digo
"""
# Dicion rio para armazenar as bases de treino
bases_treino = {}

# Base original de treino
bases_treino['Original'] = train_data.copy()
```

É possível ver a distribuição das variáveis de treino antes das transformações na Figura 12, e as estatísticas descritivas na Tabela 5.

Tabela 5: Estatísticas descritivas da base Original (Treino)

	count	mean	std	min	25%	50%	75%	max
fLength	13233.00	53.21	41.85	4.28	24.51	37.34	70.31	310.61
fWidth	13233.00	22.20	18.29	0.00	11.90	17.24	24.80	256.38
fSize	13233.00	2.83	0.47	1.94	2.48	2.74	3.11	5.32
fConc	13233.00	0.38	0.18	0.01	0.24	0.35	0.50	0.89
fConc1	13233.00	0.21	0.11	0.00	0.13	0.20	0.28	0.68
fAsym	13233.00	-4.12	58.22	-457.92	-20.65	3.94	24.07	575.24
fM3Long	13233.00	10.97	50.78	-331.78	-12.68	15.38	36.27	238.32
fM3Trans	13233.00	0.18	20.81	-205.90	-10.92	0.45	10.97	179.85
fAlpha	13233.00	27.48	26.13	0.00	5.47	17.36	45.57	90.00
fDist	13233.00	194.50	74.73	1.28	143.04	192.60	241.60	495.56
class	13233.00	0.65	0.48	0.00	0.00	1.00	1.00	1.00

A partir dessa análise, é perceptível pelo histograma que existe certo desbalanceamento entre as classes, em que há uma maior presença de classes negativas no conjunto de treino. Isso pode prejudicar os modelos de classificação que serão utilizados e gerar viéses. Por isso, será feito um *undersampling* aleatório, para equalizar a presença das classes.

```
def undersample_data_safe(X, y):
    """
    Realiza o undersampling para balancear as classes no conjunto de
    treinamento.
    """
    data = pd.concat([X, y], axis=1)
    class_counts = data['class'].value_counts()
```

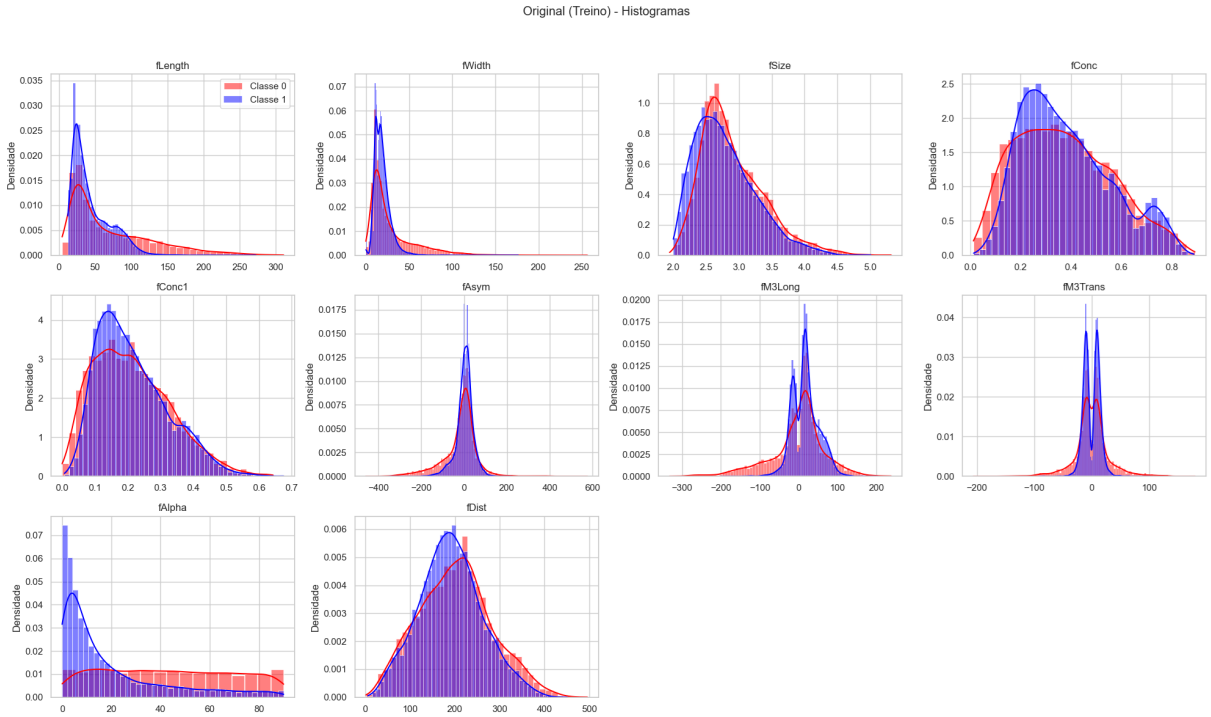

Figura 12: Histograma da distribuição das variáveis da base de treino, antes das transformações.

```
7
8       # Identificar a classe majorit ria e minorit ria
9       majority_class_value = class_counts.idxmax()
10      minority_class_value = class_counts.idxmin()
11
12      majority_class = data[data['class'] == majority_class_value]
13      minority_class = data[data['class'] == minority_class_value]
14
15      # Verificar se ambas as classes est o presentes
16      if len(minority_class) > 0 and len(majority_class) > 0:
17          # Igualar ao tamanho da classe minorit ria
18          majority_downsampled = resample(
19              majority_class,
20              replace=False,
21              n_samples=len(minority_class),
22              random_state=RANDOM_STATE
23          )
24          balanced_data = pd.concat([majority_downsampled, minority_class
    ])
25
26          return balanced_data.drop(columns=['class']).reset_index(drop=
    True), balanced_data['class'].reset_index(drop=True)
27      else:
28          print("Uma das classes est  ausente. Retornando os dados
    originais.")
29          return X.reset_index(drop=True), y.reset_index(drop=True)
30
31
32  # Dicion rio para armazenar as bases de treino com undersampling
33  bases_treino_undersampled = {}
```

```
34
35 for name, dataset in bases_treino.items():
36     print(f"\nAplicando undersampling na base de treino: {name}")
37     X_train = dataset.drop(columns=['class'])
38     y_train = dataset['class']
39     X_train_balanced, y_train_balanced = undersample_data_safe(X_train,
       y_train)
40     balanced_dataset = pd.concat([X_train_balanced, y_train_balanced],
       axis=1)
41     bases_treino_undersampled[name] = balanced_dataset
```

Como resultado, a base de treino, que antes possuia 13233 valores, sendo 4574 na classe 1, e 8659 na classe 0, foi balanceada para ambas terem 4574 valores, totalizando 9148 observações, como mostra a Tabela 6

Tabela 6: Estatísticas descritivas da base Original (Treino *Undersampled*)

	count	mean	std	min	25%	50%	75%	max
fLength	9148.00	57.35	46.58	4.28	24.85	38.93	77.08	310.61
fWidth	9148.00	23.80	20.99	0.00	11.69	17.29	26.43	256.38
fSize	9148.00	2.84	0.48	1.94	2.49	2.75	3.13	5.32
fConc	9148.00	0.38	0.19	0.01	0.23	0.35	0.51	0.89
fConc1	9148.00	0.21	0.11	0.00	0.13	0.20	0.29	0.67
fAsym	9148.00	-7.38	64.48	-457.92	-24.35	2.55	23.74	575.24
fM3Long	9148.00	7.76	56.44	-331.78	-14.97	14.51	35.58	238.32
fM3Trans	9148.00	0.29	23.38	-205.90	-11.04	0.73	11.25	179.85
fAlpha	9148.00	31.52	27.09	0.00	7.37	23.38	52.52	90.00
fDist	9148.00	196.28	76.63	1.28	143.08	194.39	244.00	495.56
class	9148.00	0.50	0.50	0.00	0.00	0.50	1.00	1.00

Visto a necessidade de *undersampling*, a base de treino será transformada, repetindo o processo que foi feito com a base completa no estudo exploratório: winsorização a 1% seguido pela normalização, em seguida fazendo o *undersampling*. O output mostra o histograma da distribuição na Figura 13, o boxplot na Figura 14 e as estatísticas descritivas na Tabela 7.

Tabela 7: Estatísticas descritivas da base Normalizada + Winsorizada 1% (Treino *Undersampled*)

	count	mean	std	min	25%	50%	75%	max
fLength	9148.00	0.22	0.22	0.00	0.06	0.13	0.32	1.00
fWidth	9148.00	0.20	0.20	0.00	0.08	0.14	0.23	1.00
fSize	9148.00	0.36	0.22	0.00	0.19	0.32	0.49	1.00
fConc	9148.00	0.42	0.25	0.00	0.22	0.39	0.60	1.00
fConc1	9148.00	0.38	0.24	0.00	0.20	0.35	0.54	1.00
fAsym	9148.00	0.63	0.17	0.00	0.58	0.66	0.72	1.00
fM3Long	9148.00	0.60	0.18	0.00	0.52	0.62	0.69	1.00
fM3Trans	9148.00	0.50	0.16	0.00	0.41	0.50	0.58	1.00
fAlpha	9148.00	0.36	0.31	0.00	0.08	0.26	0.59	1.00
fDist	9148.00	0.47	0.23	0.00	0.31	0.46	0.61	1.00
class	9148.00	0.50	0.50	0.00	0.00	0.50	1.00	1.00

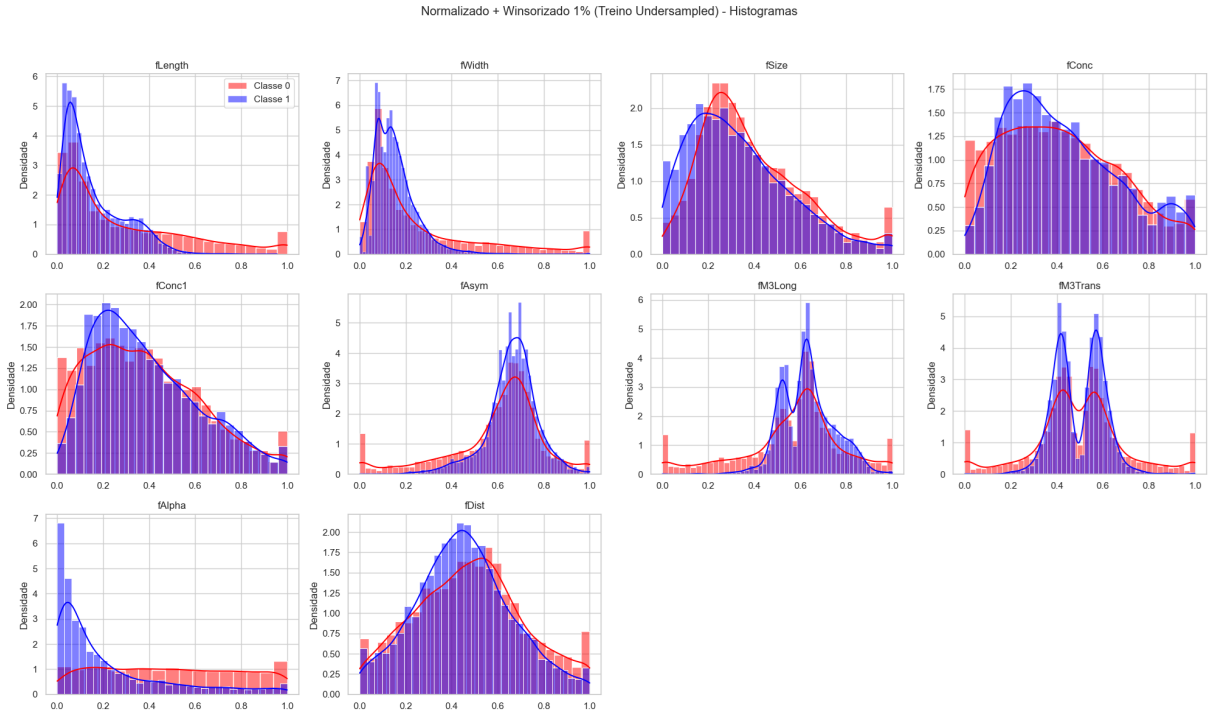

Figura 13: Histograma da distribuição das variáveis da base de treino, após as transformações.

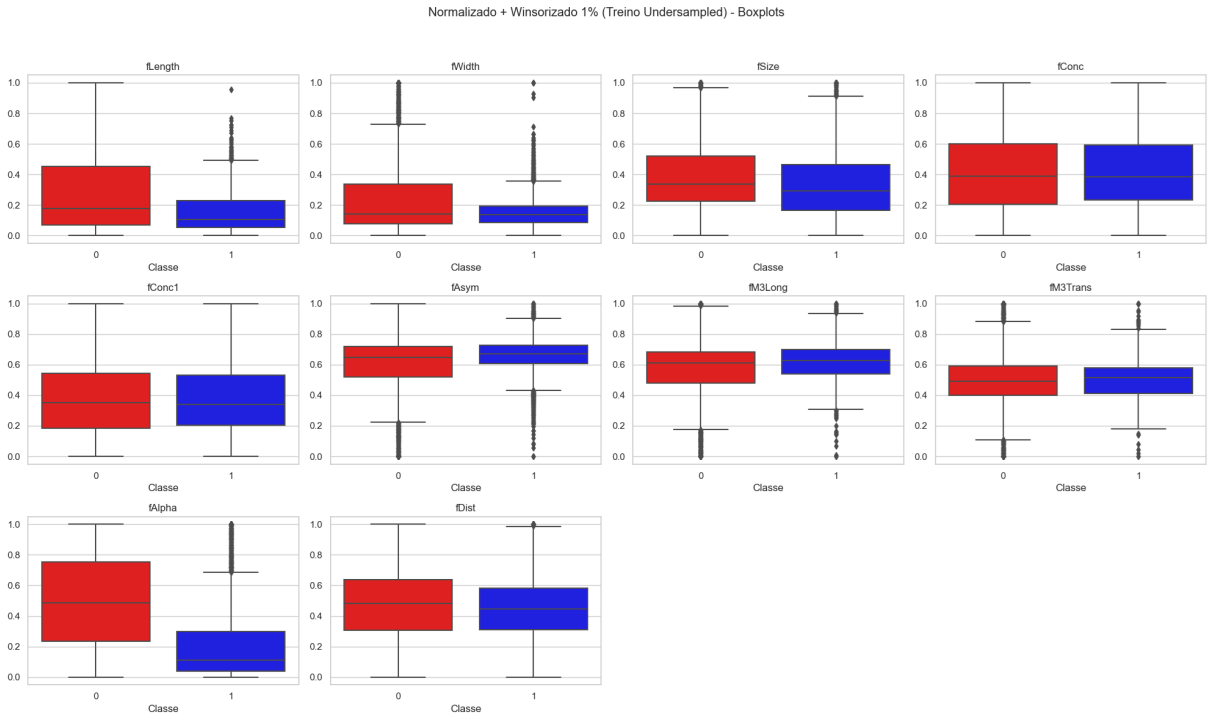

Figura 14: Boxplot da distribuição das variáveis da base de treino, após as transformações.

A comparação entre o treino original e o treino após *undersampling*, winsorização a 1% e normalização evidencia melhorias significativas na preparação dos dados, tal qual aconteceu com a base completa original. Repetimos o processo para a base de validação. Porém, utilizamos as escalas estimadas no conjunto de treino para normalização e winso-

rização, também para esse conjunto, que será mais utilizado na próxima entrega. Como resultado, temos estatísticas descritivas coerentes com o conjunto de treino (Tabela 8).

```
validation_winsor_1 = validation_data.copy()
for col in numeric_cols:
    validation_winsor_1[col] = winsorize(validation_winsor_1[col],
    limits=winsor_limits_1)

# Usamos o scaler 'scaler_train_winsor_1' que foi ajustado em '
    train_winsor_1' antes do undersampling
validation_winsor_1_norm = validation_winsor_1.copy()
validation_winsor_1_norm[numeric_cols] = scaler_train_winsor_1.transform
    (validation_winsor_1_norm[numeric_cols])
```

Tabela 8: Estatísticas descritivas da base Normalizada + Winsorizada 1% (Validação - Escala Treino)

	count	mean	std	min	25%	50%	75%	max
fLength	2836.00	0.20	0.20	0.00	0.06	0.12	0.28	0.997
fWidth	2836.00	0.18	0.17	-0.03	0.08	0.14	0.21	0.999
fSize	2836.00	0.35	0.22	-0.00	0.19	0.31	0.47	0.987
fConc	2836.00	0.43	0.25	0.01	0.23	0.39	0.59	1.004
fConc1	2836.00	0.39	0.23	0.01	0.20	0.35	0.54	0.985
fAsym	2836.00	0.64	0.16	-0.00	0.59	0.66	0.72	0.955
fM3Long	2836.00	0.60	0.16	0.02	0.53	0.62	0.69	0.980
fM3Trans	2836.00	0.50	0.14	0.04	0.41	0.50	0.58	1.007
fAlpha	2836.00	0.31	0.29	0.00	0.06	0.19	0.51	1.000
fDist	2836.00	0.45	0.22	-0.01	0.30	0.45	0.59	0.998
class	2836.00	0.66	0.48	0.00	0.00	1.00	1.00	1.000

Adicionalmente, para validar a base de treino, será aplicada uma validação cruzada. Para tanto, serão treinados modelos genéricos do KNN, DT, MLP e SVM. O objetivo dessa etapa não é achar o melhor modelo, mas entender como o tratamento proposto da base de treino pode ser comparativamente vantajoso em relações a outras técnicas. Assim, serão propostas diferentes bases de treino com *undersampling*, com diferentes combinações das técnicas usadas para tratamento na etapa anterior (winsorização a 1 e a 5%, normalização e PCA). Essa é uma etapa complementar, já que a análise exploratória já aponta que a winsorização a 1% seguida pela normazliação é a técnica que melhor trata os dados preservando suas caractéristicas.

Em todas as bases, foi respeitada a ordem: 1. winsorização (se houver), 2. normalização (se houver), 3. PCA (se houver) 4. *undersampling*.

```
def perform_cross_validation(X, y, models, cv_folds):
    """
    Realiza validacao cruzada usando k-folds e retorna as acur cias
    m dias.
    """
    results = {}
    skf = StratifiedKFold(n_splits=cv_folds, shuffle=True, random_state=
    RANDOM_STATE)
    for model_name, model in models.items():
        cv_scores = cross_val_score(model, X, y, cv=skf, scoring='
    accuracy')
        results[model_name] = cv_scores.mean()
```

```
    return results

    # Modelos a serem utilizados
models = {
    "Arvore de Decisao": DecisionTreeClassifier(random_state=
   RANDOM_STATE),
    "KNN": KNeighborsClassifier(),
    "SVM": SVC(),
    "Rede Neural": MLPClassifier(random_state=RANDOM_STATE),
}

# Dicionario para armazenar resultados da valida  o cruzada
cv_results = {}

# Realizando validacao cruzada
for name, dataset in bases_treino_undersampled.items():
    print(f"\nRealizando valida  o cruzada para a base: {name}")
    X = dataset.drop(columns=['class'])
    y = dataset['class']
    cv_scores = perform_cross_validation(X, y, models, cv_folds=5)
    cv_results[name] = cv_scores
```

O resultado da validação cruzada nas bases de treino com *undersampling*, Figura 15, demonstra que a escolha da técnica utilizada neste projeto não só é estatisticamente coerente, como gera os melhores resultados. A winsorização a 1% seguida pela normalização se divide como a melhor técnica para os modelos propostos, com a winsorização a 5% seguida de normalização. Isso reforça a coerencia da abordagem escolhida.

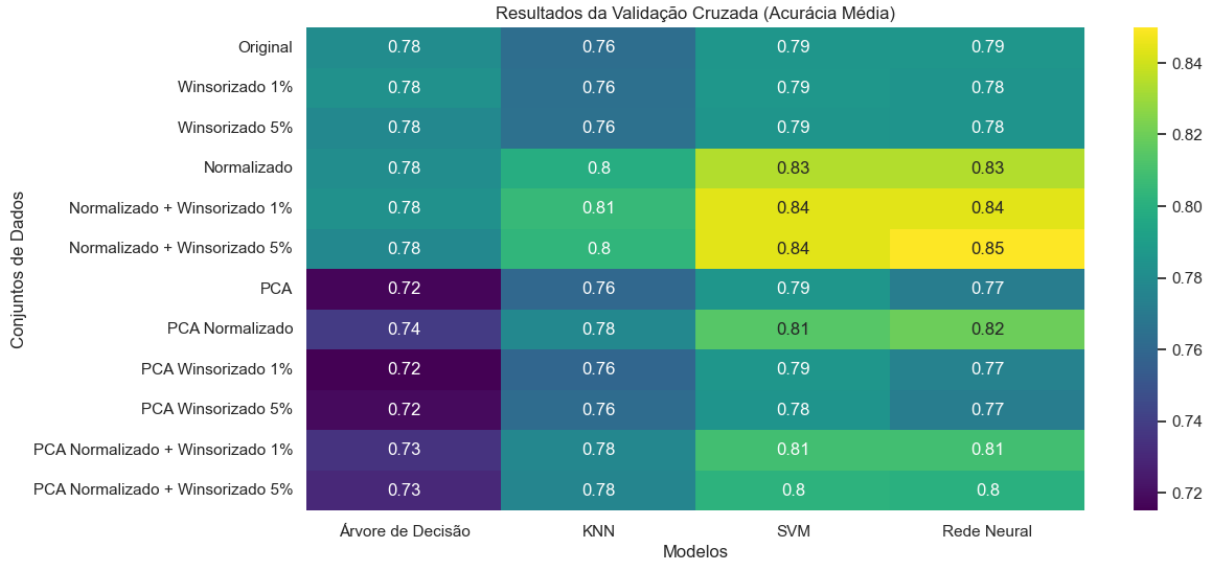

Figura 15: Resultado da validação cruzada em diferentes bases de treino com undersampling.

Entendido os resultados desta etapa, é relevante ressaltar que a escolha por manter um conjunto de validação é importante e que ele deve ser usado apenas para "avaliar o desempenho do modelo preditivo enquanto estiver sendo gerado, fazendo um papel de conjunto de teste durante o treinamento"[3, p. 221]. Adicionalmente, é importante reforçar que manter o conjunto de teste intacto foi fundamental para assegurar uma avaliação imparcial e confiável do desempenho dos modelos desenvolvidos. Ao não modificar o conjunto

de teste, garante-se que ele represente dados nunca antes vistos, preservando a integridade do processo de validação.

Isso é especialmente importante em *pipelines* onde técnicas de preparação de dados, como normalização, winsorização e *undersampling*, podem introduzir vieses se aplicadas de maneira inconsistente. Um conjunto de teste imaculado fornece uma métrica clara da capacidade de generalização do modelo, eliminando o risco de resultados inflados que poderiam ocorrer caso o modelo já tivesse "visto"as transformações aplicadas aos dados. Essa prática é essencial para garantir que os modelos desenvolvidos sejam adequados para aplicações reais, onde novos dados podem apresentar características inesperadas.

6 Reflexões Críticas

Nesta etapa do trabalho, foi possível observar a importância da correta preparação dos dados e o quanto isso poderia afetar o desempenho do modelo final. É possível observar diferenças relevantes entre a base original e a base tratada. Após as etapas de processamento, obteve-se uma base mais uniforme e com escalas otimizadas, porém mantendo a integridade estatística dos dados.

A winsorização a 1% e a normalização reduziram a dispersão dos dados (diminuindo a presença de *outliers*), mas preservaram a estrutura relativa das distribuições. É válido destacar que esse processo poderia ocultar informações relevantes contidas nesses *outliers* e limitar a identificação de padrões em determinadas modelagens. No entanto, é uma forma relevante de favorecer a estabilidade e robustez das análises, e por isso foi adotada.

Por outro lado, o PCA foi útil para reduzir a dimensionalidade dos dados, de forma a eliminar possíveis redundâncias. No entanto, possíveis desafios intrínsecos a essa técnica podem ser encontrados, por exemplo, a dificuldade de interpretação dos componentes principais ao tentar associar a componente linear com características específicas. Tem-se, então, uma análise mais abstrata e menos intuitiva (devido à redução da explicabilidade dos dados).

Outro aspecto crítico foi o balanceamento das classes por meio de *undersampling*. Essa técnica assegurou uma distribuição equitativa entre as classes, evitando vieses em algoritmos de classificação. No entanto, a redução do número de observações disponíveis para treinamento pode limitar a generalização dos modelos, especialmente em problemas complexos. Caso sejam identificados problemas nas próximas etapas, técnicas alternativas, como *oversampling* ou geração de dados sintéticos via SMOTE, podem ser exploradas para aumentar o volume de dados sem comprometer a representatividade.

Por fim, o treinamento de modelos genéricos reforça a robustez da escolha do processo de preparação de dados feita neste projeto, em que, para todos os modelos, a base winsorizada a 1% e normalizada configura como a de melhor acurácia média, ou segunda melhor, em uma validação cruzada.

7 Conclusão

A análise da etapa de preparação dos dados demonstrou o papel central dessa fase na construção de um pipeline robusto e eficiente para modelagem preditiva. As ações implementadas — como remoção de duplicatas, tratamento de *outliers* por winsorização a 1%, normalização e exploração do PCA — mostraram-se fundamentais para mitigar problemas de qualidade e inconsistência nos dados, fatores críticos para o sucesso do projeto. A

limpeza inicial garantiu a eliminação de redundâncias, enquanto a winsorização reduziu a influência de valores extremos de maneira controlada, sem descartar informações relevantes. A normalização, por sua vez, uniformizou as escalas das variáveis, permitindo que todas contribuíssem de forma equilibrada para a análise.

O PCA, embora tenha sido explorado como estratégia para lidar com multicolinearidade e dimensionalidade elevada, apresentou limitações significativas. Sua aplicação resultou na perda de interpretabilidade, uma vez que os componentes principais não mantêm associações diretas com variáveis originais. Além disso, a necessidade de replicar as transformações no conjunto de testes poderia introduzir vieses indesejados, comprometendo a validade do processo de validação. Por essas razões, a abordagem baseada exclusivamente na winsorização e normalização foi priorizada, pois equilibrava adequadamente a preservação da integridade dos dados e a mitigação de problemas técnicos.

A estratégia de divisão dos dados em treino, validação e teste, complementada pelo uso de técnicas de undersampling, garantiu um pipeline consistente para avaliação e modelagem. Já os resultados da validação cruzada reforçaram a coerência da abordagem escolhida, especialmente ao destacar a superioridade da base tratada com winsorização a 1% e normalização. Essa combinação não apenas garantiu melhor desempenho médio nos modelos testados, mas também preservou as relações fundamentais entre as variáveis. Isso valida a eficácia das etapas de preparação.

Em resumo, a preparação dos dados desempenhou um papel essencial no estabelecimento de bases sólidas para a modelagem. A etapa reafirma a importância de um processo preciso e crítico na ciência de dados, onde a preparação adequada é a base para o sucesso analítico e a geração de insights confiáveis. Em especial, metodologias como a CRISP-DM se mostram adequadas para esse processo.

8 Anexos

Em anexo, encontra-se o código completo desenvolvido. Ele foi projetado para gerar os outputs de todas as bases em todas as etapas. Consequentemente, ele produz mais informações do que as apresentadas neste artigo. Para aprimorar a fluidez e o entendimento do texto, foram destacados apenas os resultados relevantes para a interpretação da análise.

8.1 Imports e configurações

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats.mstats import winsorize
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split, cross_val_score,
    StratifiedKFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.utils import resample

```

```
RANDOM_STATE = 42
sns.set(style="whitegrid")
```

8.2 Funções Auxiliares

```
def load_and_clean_data(filepath):
    """
    Carrega o arquivo CSV, converte a coluna 'class' para valores bin rios e remove duplicatas.
    """
    data = pd.read_csv(filepath)
    data['class'] = data['class'].replace({'g': 1, 'h': 0})
    num_rows_before = data.shape[0]
    data.drop_duplicates(inplace=True)
    num_rows_after = data.shape[0]
    num_duplicates_removed = num_rows_before - num_rows_after
    print(f"N mero de linhas duplicadas removidas: {num_duplicates_removed}")
    return data

def plot_histograms_boxplots(data, title_prefix):
    """
    Plota histogramas e boxplots para cada feature num rica, diferenciando as classes.
    Todos os histogramas e boxplots de uma base s o plotados em uma  nica  imagem com m ltiplos subplots.
    """
    numeric_cols = data.select_dtypes(include=[np.number]).columns.drop('class')
    num_cols = len(numeric_cols)
    num_rows = (num_cols + 3) // 4

    plt.figure(figsize=(20, num_rows * 4))
    for idx, col in enumerate(numeric_cols):
        plt.subplot(num_rows, 4, idx + 1)
        for class_value, color in zip([0, 1], ['red', 'blue']):
            sns.histplot(
                data[data['class'] == class_value][col],
                color=color,
                label=f'Classe {class_value}',
                kde=True,
                stat='density',
                alpha=0.5
            )
        plt.title(f"{col}")
        plt.xlabel('')
        plt.ylabel('Densidade')
        if idx == 0:
            plt.legend()
    plt.suptitle(f"{title_prefix} - Histogramas")
    plt.tight_layout(rect=[0, 0, 1, 0.95])
    plt.show()

    plt.figure(figsize=(20, num_rows * 4))
    for idx, col in enumerate(numeric_cols):
        plt.subplot(num_rows, 4, idx + 1)
```

```
        sns.boxplot(x='class', y=col, data=data, palette=['red', 'blue'])
        plt.title(f"{col}")
        plt.xlabel('Classe')
        plt.ylabel('')
    plt.suptitle(f"{title_prefix} - Boxplots")
    plt.tight_layout(rect=[0, 0, 1, 0.95])
    plt.show()

def generate_latex_table(data, title):
    """
    Gera uma tabela LaTeX com estat sticas descritivas.
    """
    desc = data.describe().transpose()
    latex_table = desc.to_latex(float_format="%.3f")
    print(f"Tabela LaTeX para {title}:\n")
    print(latex_table)
    print("\n" + "-" * 80 + "\n")

def undersample_data_safe(X, y):
    """
    Realiza o undersampling para balancear as classes no conjunto de treinamento.
    """
    data = pd.concat([X, y], axis=1)
    class_counts = data['class'].value_counts()

    majority_class_value = class_counts.idxmax()
    minority_class_value = class_counts.idxmin()

    majority_class = data[data['class'] == majority_class_value]
    minority_class = data[data['class'] == minority_class_value]

    if len(minority_class) > 0 and len(majority_class) > 0:
        majority_downsampled = resample(
            majority_class,
            replace=False,
            n_samples=len(minority_class),
            random_state=RANDOM_STATE
        )
        balanced_data = pd.concat([majority_downsampled, minority_class])

        return balanced_data.drop(columns=['class']).reset_index(drop=True), balanced_data['class'].reset_index(drop=True)
    else:
        print("Uma das classes est  ausente. Retornando os dados originais.")
        return X.reset_index(drop=True), y.reset_index(drop=True)

def perform_cross_validation(X, y, models, cv_folds):
    """
    Realiza valida  o cruzada usando k-folds e retorna as acur cias m dias.
    """
    results = {}
    skf = StratifiedKFold(n_splits=cv_folds, shuffle=True, random_state=RANDOM_STATE)
```

```
    for model_name, model in models.items():
        cv_scores = cross_val_score(model, X, y, cv=skf, scoring='accuracy')
        results[model_name] = cv_scores.mean()
    return results

def apply_pca(data, n_components=0.95):
    """
    Aplica PCA ao conjunto de dados e retorna o DataFrame transformado.
    """
    features = data.drop(columns=['class'])
    pca = PCA(n_components=n_components, random_state=RANDOM_STATE)
    pca_features = pca.fit_transform(features)
    pca_feature_names = [f'PC{i+1}' for i in range(pca_features.shape[1])]
    data_pca = pd.DataFrame(pca_features, columns=pca_feature_names)
    data_pca['class'] = data['class'].reset_index(drop=True)
    return data_pca
```

8.3 Importação da base

```
file_path = "C:/Users/55819/Downloads/magic.csv"
data = load_and_clean_data(file_path)
```

8.4 Análise da base original

```
bases_completas = {}
numeric_cols = data.select_dtypes(include=[np.number]).columns.drop('class')

# Base original
bases_completas['Original'] = data.copy()

# Winsoriza  o a 1%
winsor_limits_1 = (0.01, 0.01)
data_winsor_1 = data.copy()
for col in numeric_cols:
    data_winsor_1[col] = winsorize(data_winsor_1[col], limits=winsor_limits_1)
bases_completas['Winsorizado 1%'] = data_winsor_1

# Winsoriza  o a 5%
winsor_limits_5 = (0.05, 0.05)
data_winsor_5 = data.copy()
for col in numeric_cols:
    data_winsor_5[col] = winsorize(data_winsor_5[col], limits=winsor_limits_5)
bases_completas['Winsorizado 5%'] = data_winsor_5

# Normaliza  o da base original
data_normalizado = data.copy()
scaler_original = MinMaxScaler()
data_normalizado[numeric_cols] = scaler_original.fit_transform(data_normalizado[numeric_cols])
bases_completas['Normalizado'] = data_normalizado
```

```
26
27 # Normaliza  o da base winsorizada a 1%
28 data_winsor_1_norm = data_winsor_1.copy()
29 scaler_winsor_1 = MinMaxScaler()
30 data_winsor_1_norm[numeric_cols] = scaler_winsor_1.fit_transform(
       data_winsor_1_norm[numeric_cols])
31 bases_completas['Normalizado + Winsorizado 1%'] = data_winsor_1_norm
32
33 # Normaliza  o da base winsorizada a 5%
34 data_winsor_5_norm = data_winsor_5.copy()
35 scaler_winsor_5 = MinMaxScaler()
36 data_winsor_5_norm[numeric_cols] = scaler_winsor_5.fit_transform(
       data_winsor_5_norm[numeric_cols])
37 bases_completas['Normalizado + Winsorizado 5%'] = data_winsor_5_norm
38
39 # PCA na base original
40 bases_completas['PCA'] = apply_pca(data)
41
42 # PCA na base normalizada
43 bases_completas['PCA Normalizado'] = apply_pca(data_normalizado)
44
45 # PCA na base winsorizada a 1%
46 bases_completas['PCA Winsorizado 1%'] = apply_pca(data_winsor_1)
47
48 # PCA na base winsorizada a 5%
49 bases_completas['PCA Winsorizado 5%'] = apply_pca(data_winsor_5)
50
51 # PCA na base normalizada e winsorizada a 1%
52 bases_completas['PCA Normalizado + Winsorizado 1%'] = apply_pca(
       data_winsor_1_norm)
53
54 # PCA na base normalizada e winsorizada a 5%
55 bases_completas['PCA Normalizado + Winsorizado 5%'] = apply_pca(
       data_winsor_5_norm)
56
57 for name, dataset in bases_completas.items():
58     print(f"\nAnalisando a base completa: {name}")
59     plot_histograms_boxplots(dataset, f"{name} (Completo)")
60     generate_latex_table(dataset, f"{name} (Completo)")
```

8.5 Divisão dos conjuntos de treino, validação e teste

```
1 train_data, temp_data = train_test_split(data, test_size=0.30,
      random_state=RANDOM_STATE)
2
3 validation_data, test_data = train_test_split(
4     temp_data, test_size=0.5, random_state=RANDOM_STATE
5 )
6
7 print(f"Tamanho do conjunto de treino: {len(train_data)}")
8 print(f"Tamanho do conjunto de valida  o: {len(validation_data)}")
9 print(f"Tamanho do conjunto de teste: {len(test_data)}")
```

8.6 Análise da base de treino

```
bases_treino = {}

# Base original de treino
bases_treino['Original'] = train_data.copy()

# Winsoriza  o a 1% no conjunto de treino
train_winsor_1 = train_data.copy()
for col in numeric_cols:
    train_winsor_1[col] = winsorize(train_winsor_1[col], limits=
    winsor_limits_1)
bases_treino['Winsorizado 1%'] = train_winsor_1

# Winsoriza  o a 5% no conjunto de treino
train_winsor_5 = train_data.copy()
for col in numeric_cols:
    train_winsor_5[col] = winsorize(train_winsor_5[col], limits=
    winsor_limits_5)
bases_treino['Winsorizado 5%'] = train_winsor_5

# Normaliza  o da base original de treino
train_normalizado = train_data.copy()
scaler_train_original = MinMaxScaler()
train_normalizado[numeric_cols] = scaler_train_original.fit_transform(
    train_normalizado[numeric_cols])
bases_treino['Normalizado'] = train_normalizado

# Normaliza  o da base winsorizada a 1% de treino
train_winsor_1_norm = train_winsor_1.copy()
scaler_train_winsor_1 = MinMaxScaler()
train_winsor_1_norm[numeric_cols] = scaler_train_winsor_1.fit_transform(
    train_winsor_1_norm[numeric_cols])
bases_treino['Normalizado + Winsorizado 1%'] = train_winsor_1_norm

# Normaliza  o da base winsorizada a 5% de treino
train_winsor_5_norm = train_winsor_5.copy()
scaler_train_winsor_5 = MinMaxScaler()
train_winsor_5_norm[numeric_cols] = scaler_train_winsor_5.fit_transform(
    train_winsor_5_norm[numeric_cols])
bases_treino['Normalizado + Winsorizado 5%'] = train_winsor_5_norm

# PCA na base original de treino
bases_treino['PCA'] = apply_pca(train_data)

# PCA na base normalizada de treino
bases_treino['PCA Normalizado'] = apply_pca(train_normalizado)

# PCA na base winsorizada a 1% de treino
bases_treino['PCA Winsorizado 1%'] = apply_pca(train_winsor_1)

# PCA na base winsorizada a 5% de treino
bases_treino['PCA Winsorizado 5%'] = apply_pca(train_winsor_5)

# PCA na base normalizada e winsorizada a 1% de treino
bases_treino['PCA Normalizado + Winsorizado 1%'] = apply_pca(
    train_winsor_1_norm)

# PCA na base normalizada e winsorizada a 5% de treino
bases_treino['PCA Normalizado + Winsorizado 5%'] = apply_pca(
```

```
    train_winsor_5_norm)

for name, dataset in bases_treino.items():
    print(f"\nAnalisando a base de treino: {name}")
    plot_histograms_boxplots(dataset, f"{name} (Treino)")
    generate_latex_table(dataset, f"{name} (Treino)")
```

8.7 Análise da base de treino com undersampling

```
bases_treino_undersampled = {}

for name, dataset in bases_treino.items():
    print(f"\nAplicando undersampling na base de treino: {name}")
    X_train = dataset.drop(columns=['class'])
    y_train = dataset['class']
    X_train_balanced, y_train_balanced = undersample_data_safe(X_train,
    y_train)
    balanced_dataset = pd.concat([X_train_balanced, y_train_balanced],
    axis=1)
    bases_treino_undersampled[name] = balanced_dataset

for name, dataset in bases_treino_undersampled.items():
    print(f"\nAnalisando a base de treino com undersampling: {name}")
    plot_histograms_boxplots(dataset, f"{name} (Treino Undersampled)")
    generate_latex_table(dataset, f"{name} (Treino Undersampled)")
```

8.8 Análise da base de validação

```
validation_winsor_1 = validation_data.copy()
for col in numeric_cols:
    validation_winsor_1[col] = winsorize(validation_winsor_1[col],
    limits=winsor_limits_1)
validation_winsor_1_norm = validation_winsor_1
validation_winsor_1_norm[numeric_cols] = scaler_train_winsor_1.transform
    (validation_winsor_1_norm[numeric_cols])


print("\nAnalisando o conjunto de valida  o ap s Winsoriza  o a 1%
    e Normaliza  o (usando escala do treino antes do undersampling):")
plot_histograms_boxplots(validation_winsor_1_norm, "Valida  o -
    Winsorizado 1% e Normalizado (Escala Treino)")
generate_latex_table(validation_winsor_1_norm, "Valida  o -
    Winsorizado 1% e Normalizado (Escala Treino)")
```

8.9 Validação cruzada

```
"""
Modelos utilizados
"""
models = {
    " rvore  de Decis o": DecisionTreeClassifier(random_state=
    RANDOM_STATE),
    "KNN": KNeighborsClassifier(),
```

```
    "SVM": SVC(),
    "Rede Neural": MLPClassifier(random_state=RANDOM_STATE),
}

"""
Valida  o cruzada
"""
cv_results = {}
for name, dataset in bases_treino_undersampled.items():
    print(f"\nRealizando valida  o cruzada para a base: {name}")
    X = dataset.drop(columns=['class'])
    y = dataset['class']
    cv_scores = perform_cross_validation(X, y, models, cv_folds=5)
    cv_results[name] = cv_scores

cv_results_df = pd.DataFrame(cv_results).transpose()
print("\nResultados da Valida  o Cruzada:")
print(cv_results_df)
plt.figure(figsize=(12, 6))
sns.heatmap(cv_results_df, annot=True, cmap='viridis')
plt.title('Resultados da Valida  o Cruzada (Acur cia M dia)')
plt.xlabel('Modelos')
plt.ylabel('Conjuntos de Dados')
plt.show()
```

Referências

[1] Z. Luna, "Crisp-dm phase 3: Data preparation." https://medium.com/analytics-vidhya/crisp-dm-phase-3-data-preparation-faf5ee8dc38e.

[2] S. García, J. Luengo e F. Herrera, *Data Preprocessing in Data Mining*, vol. 72 of *Intelligent Systems Reference Library*. Cham, Switzerland: Springer, 2015.

[3] L. A. d. Silva, S. M. Peres e C. Boscarioli, *Introdução à Mineração de Dados: Com Aplicações em R*. Rio de Janeiro, Brasil: Elsevier Editora Ltda., 1 ed., 2016.