Análise Comparativa entre Abordagens Simbólicas e Redes Neurais para Aplicação de Aprendizagem de Máquina em Sinais Vitais

Adryan C. Feres[1], **Márcia E. Ferreira**[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Curitiba – PR – Brasil

adryancastro@alunos.utfpr.edu.br, marcia.eliana1@gmail.com

Abstract. *This paper presents a comparative analysis of three machine learning (ML) approaches applied to a vital signs dataset: ID3 (symbolic learning), Random Forest (symbolic learning) and Neural Networks (Multilayer Perceptron). The functional and non-functional requirements for each modeling are described, including parameterization, preprocessing and evaluation metrics. The experimental results demonstrate that the symbolic learning models: ID3 (89.56*

Resumo. *Este artigo apresenta uma análise comparativa de três abordagens de aprendizagem de máquina (AM) aplicadas a um conjunto de dados de sinais vitais: ID3 (aprendizagem simbólica), Random Forest (aprendizagem simbólica) e Redes Neurais (Perceptron Multicamadas). São descritos os requisitos funcionais e não funcionais para cada modelagem, incluindo parametrização, préprocessamento e métricas de avaliação. Os resultados experimentais demonstram que os modelos de aprendizagem simbólica : ID3 (89,56%), Random Forest (92,67%) e Rede Neural (95,6%), especialmente na identificação de classes minoritárias.*

1. Introdução

A área da saúde tem sido cada vez mais impactada pela crescente disponibilidade de dados e pelo avanço das técnicas de Aprendizagem de Máquina. A aplicação dessas técnicas em dados que utilizam sinais vitais como fonte de informação apresenta crescente interesse, com potencial para análise e monitoramento de pacientes. Esses dados geram volume significativo de informações que, ao serem analisadas de forma inteligente, podem auxiliar na detecção precoce de condições adversas, no prognóstico de doenças e na personalização de tratamentos. No entanto, a complexidade desses dados, somada à necessidade de modelos interpretáveis, representa desafios importantes, exigindo abordagens que combinem alto desempenho com interpretabilidade e confiabilidade.

Neste contexto, este artigo apresenta uma análise comparativa de três abordagens distintas de aprendizagem de máquina aplicadas a um conjunto de dados de sinais vitais: ID3 (aprendizagem simbólica), Random Forest (aprendizagem ensemble) e Redes Neurais (Perceptron Multicamadas). O objetivo é demonstrar como diferentes abordagens podem ser aplicadas para extrair informações relevantes a partir de dados e medições de sinais vitais, identificando padrões que se relacionam com determinadas condições clínicas. As três abordagens selecionadas permitem análises comparativas complementares: o ID3

representa a aprendizagem baseada em árvores de decisão com alta interpretabilidade; o Random Forest exemplifica uma abordagem ensemble que equilibra interpretabilidade e desempenho; e as Redes Neurais representam modelos com alta capacidade de modelagem de relações não-lineares complexas.

2. Metodologia

Este trabalho envolve a análise de um dataset de sinais vitais, onde cada instância representa um conjunto de medições fisiológicas e uma label correspondente a uma condição clínica específica. O dataset fornecido consiste em dois arquivos: um com rótulos e outro sem, sendo útil para cenários de previsão ou validação posterior. A tarefa é de classificação supervisionada, onde o objetivo é construir um modelo capaz de prever a condição clínica a partir de um novo conjunto de sinais vitais.

2.1. Análise Exploratória do Conjunto de Dados

A análise exploratória dos dados revelou características importantes do conjunto de dados utilizado, composto por 1500 instâncias com 6 atributos numéricos e uma variável de classe categórica. Os resultados evidenciaram um desbalanceamento significativo entre as classes, presença de outliers em alguns atributos e correlações variadas entre as variáveis preditoras.

Esta análise é fundamental para o desenvolvimento do trabalho, pois permite compreender a estrutura dos dados e identificar possíveis problemas que podem impactar o desempenho dos modelos. As informações obtidas orientam as etapas de pré-processamento, seleção de métricas de avaliação adequadas e escolha de estratégias para lidar com o desbalanceamento das classes, contribuindo diretamente para a qualidade e validade dos modelos de aprendizagem de máquina desenvolvidos.

O pseudocódigo do código utilizado para a análise correspondente está apresentado no Algoritmo 1.

3. Modelagem

3.1. Aprendizagem Simbólica com o Algoritmo ID3

O algoritmo ID3 (Iterative Dichotomiser 3) é uma técnica de aprendizado supervisionado utilizada para construir árvores de decisão. Seu objetivo principal é gerar regras de classificação a partir de um conjunto de dados previamente rotulado, facilitando a identificação de padrões e a tomada de decisões. O ID3 é especialmente valorizado em situações em que a interpretabilidade do modelo é fundamental, pois as árvores de decisão resultantes são intuitivas e de fácil compreensão.

A cada nó da árvore, o ID3 seleciona o atributo que proporciona o maior **ganho de informação**, isto é, aquele que mais reduz a incerteza sobre a classe das instâncias. Para isso, o algoritmo se baseia no conceito de **entropia**, que quantifica o grau de desordem de um conjunto de dados em relação à sua distribuição de classes.

A **entropia** de um conjunto S com c classes é dada por:

$$\text{Entropia}(S) = -\sum_{i=1}^{c} p_i \log_2(p_i)$$

em que p_i representa a proporção de instâncias pertencentes à classe i.

Já o **ganho de informação** ao particionar S com base em um atributo A é calculado por:

$$\text{Ganho}(S, A) = \text{Entropia}(S) - \sum_{v \in \text{Valores}(A)} \frac{|S_v|}{|S|} \text{Entropia}(S_v)$$

onde S_v é o subconjunto de instâncias em que o atributo A assume o valor v.

A construção da árvore prossegue recursivamente, particionando os dados com base no atributo que maximiza o ganho de informação em cada etapa, até que sejam atingidos critérios de parada.

Requisitos Funcionais:

- **Entrada:** Arquivo contendo 6 atributos numéricos e 1 coluna de classe categórica (valores possíveis: 1, 2, 3 e 4).
- **Pré-processamento:**
  - Detecção e tratamento de outliers com base no Z-score.
  - Discretização dos atributos numéricos por quartis.
  - Divisão dos dados em conjuntos de treino e teste.
- **Modelo:**
  - Seleção de atributos com base no ganho de informação.
  - Critério de parada baseado em pureza do nó e profundidade máxima.
- **Saída:** Métricas de desempenho (acurácia, matriz de confusão, F1-score) e visualização textual da árvore de decisão.

Requisitos Não Funcionais:

- O modelo gerado deve ser compreensível por humanos.
- Reprodutibilidade garantida por definição de semente aleatória fixa.
- Robustez a pequenas variações nos dados e tratamento adequado de valores inválidos.

O pseudocódigo da implementação do ID3, é apresentado no Apêndice 2.

3.1.1. Justificativas das Decisões de Modelagem

- **Tratamento de outlier com base no Z-score:** Essa abordagem reduz o impacto de valores extremos que poderiam distorcer tanto a discretização quanto a construção da árvore, resultando em divisões mais representativas e generalizáveis.
- **Critério de parada com profundidade máxima e pureza do nó:** A pureza do nó assegura que folhas representem classes homogêneas, enquanto a limitação de profundidade previne o sobreajuste, tornando a árvore mais interpretável e eficiente.

3.2. Aprendizagem Simbólica com Random Forest

O algoritmo Random Forest, criado por Leo Breiman, é uma técnica de aprendizado supervisionado baseada em ensemble que utiliza múltiplas árvores de decisão para aumentar a robustez e precisão do modelo. Ele é aplicado tanto em problemas de classificação quanto de regressão, sendo especialmente eficaz para reduzir o risco de overfitting comum em árvores de decisão individuais.

O funcionamento do Random Forest envolve a criação de várias árvores de decisão independentes, cada uma treinada com amostras aleatórias dos dados e subconjuntos aleatórios de atributos. Para classificação, cada árvore vota em uma classe e a decisão final é feita por maioria; para regressão, calcula-se a média das previsões das árvores.

Requisitos Funcionais:

- **Entrada:** Arquivo contendo 6 atributos numéricos e 1 coluna de classe categórica (valores possíveis: 1, 2, 3 e 4).
- **Pré-processamento:**
  - Discretização dos atributos numéricos por quartis.
  - Detecção e tratamento de outliers com base no Z-score.
  - Divisão estratificada dos dados em conjuntos de treino e teste.
- **Modelo:**
  - Construção de múltiplas árvores de decisão com amostragem aleatória de instâncias e atributos (bagging).
  - Critério de parada baseado em profundidade máxima, número mínimo de instâncias por folha e número máximo de árvores.
  - Agregação das predições por votação majoritária (classificação).
- **Saída:** Métricas de desempenho (acurácia, matriz de confusão, F1-score) e visualização sumarizada da floresta.

Requisitos Não Funcionais:

- Interpretabilidade: apresentação das importâncias dos atributos e possibilidade de inspecionar árvores individuais.
- Reprodutibilidade garantida por definição de semente aleatória fixa.
- Robustez a pequenas variações nos dados e tratamento adequado de valores inválidos.

O pseudocódigo da implementação do Random Forest, é apresentado no Apêndice 3.

3.2.1. Justificativas das Decisões de Modelagem

- **Critério de parada com número mínimo de instâncias por folha, profundidade máxima e número máximo de árvores:** Esses critérios evitam que as árvores se tornem excessivamente complexas e ajustadas aos dados de treino (overfitting), garantindo maior capacidade de generalização do modelo. Eles definem limites para o crescimento da árvore, promovendo um equilíbrio entre precisão e simplicidade.

- **Bagging:** A técnica de bagging (bootstrap aggregating) aumenta a robustez do modelo ao construir múltiplas árvores independentes a partir de diferentes subconjuntos dos dados, reduzindo a variância e tornando o classificador menos sensível a pequenas variações no conjunto de treinamento.
- **Agregação das predições por votação majoritária:** A votação majoritária combina as predições das diversas árvores, tornando a decisão final mais estável e precisa, pois erros individuais de árvores específicas tendem a ser compensados pelas demais.

3.3. Redes Neurais

Rede neural é um modelo computacional inspirado no funcionamento do cérebro humano, composto por unidades chamadas de *neurônios artificiais*, organizados em camadas (entrada, ocultas e saída). Cada neurônio recebe sinais, realiza operações matemáticas (como soma ponderada e função de ativação) e transmite o resultado para outros neurônios. O aprendizado ocorre pelo ajuste dos pesos das conexões a partir de dados rotulados, permitindo que a rede reconheça padrões e realize tarefas como classificação, regressão e agrupamento.

Requisitos Funcionais:

- **Entrada:** Arquivo contendo 6 atributos numéricos e 1 coluna de classe categórica (valores possíveis: 1, 2, 3 e 4), com validação de schema (tipos e intervalos) e tratamento de valores ausentes ou inválidos.
- **Pré-processamento:**
  - Normalização min-max ou padronização z-score para atributos numéricos.
  - Codificação ordinal para a variável de saída (classes 1, 2, 3, 4), compatível com função de ativação softmax e `sparse_categorical_crossentropy`.
  - Divisão estratificada dos dados (70% treino, 30% teste), preservando a distribuição das classes.
- **Modelo:**
  - Arquitetura fully-connected com 1-2 camadas ocultas (32-64 neurônios cada), ajustável conforme validação cruzada.
  - Funções de ativação ReLU nas camadas internas e softmax na saída.
  - Otimizador Adam com taxa de aprendizado adaptativa (1e-3 a 1e-4).
  - Regularização via dropout (taxa 0.2-0.5), L2 (weight decay) e early stopping baseado em validação (`val_loss`).
  - Validação cruzada estratificada (k=5) para avaliação robusta.
- **Saída:** Métricas de desempenho (acurácia, matriz de confusão, F1-score), relatório de classificação por classe, curvas de aprendizado (loss/accuracy) e visualização de embeddings via técnicas de redução dimensional (PCA/t-SNE).

Requisitos Não Funcionais:

- Interpretabilidade via análise de importância de características (SHAP ou permutation feature importance) e visualização de embeddings.
- Robustez a pequenas variações nos dados, incluindo teste com ruído gaussiano nas entradas.
- Reprodutibilidade garantida por fixação de seeds aleatórios em todas as bibliotecas e configurações de hardware.

O pseudocódigo da implementação da Rede neural, é apresentado no Apêndice 4.

3.3.1. Justificativas das Decisões de Modelagem

- **Arquitetura fully-connected com 1-2 camadas ocultas:** Para dados de tamanho moderado, arquiteturas densas com poucas camadas são suficientes para capturar padrões relevantes sem overfitting.
- **Otimizador Adam com taxa de aprendizado adaptativa:** O Adam ajusta automaticamente a taxa de aprendizado de cada parâmetro, acelerando a convergência e facilitando o ajuste do modelo.
- **Regularização via dropout, L2 e early stopping:** Essas técnicas reduzem overfitting e promovem melhor generalização, com early stopping monitorando o desempenho em dados de validação.
- **Validação cruzada estratificada:** Preserva a distribuição das classes em cada partição, essencial para dados desbalanceados, fornecendo estimativas mais confiáveis.
- **Curvas de aprendizado e visualização de embeddings:** Permitem monitorar o treinamento e interpretar como a rede separa as classes internamente, contribuindo para análise de interpretabilidade.

4. Resultados

4.1. ID3

A análise comparativa entre as profundidades máximas da árvore ID3 (Figuras 1 e 3) revela que a profundidade 4 obteve alta acurácia geral (0,9022), mas não detectou a classe minoritária 4 (F1-score = 0), indicando viés para classes majoritárias. Já a profundidade 8 apresentou leve redução na acurácia (0,8956), porém reconheceu parcialmente a classe 4 (F1-score = 0,50), demonstrando maior capacidade de capturar padrões específicos.

A escolha da profundidade 8 justifica-se pelo equilíbrio entre precisão global e cobertura de todas as classes, alinhando-se ao objetivo de classificação multiclasse. A análise de importância dos atributos (Figura 5) confirma distribuição equilibrada, evitando dependência excessiva de um único parâmetro e garantindo robustez.

A análise t-SNE (Figura 7) com perplexidade 30 e tempo de execução de 2,6s demonstra separação clara entre as classes 1 e 3, enquanto as classes 2 e 4 apresentam maior sobreposição, explicando as dificuldades de classificação observadas.

O teste de robustez (Figura 8) confirma a estabilidade do ID3, mantendo acurácia de 0,896 sem ruído e degradação gradual até 0,67 com 20% de ruído. Esta resistência moderada ao ruído é característica de árvores de decisão com profundidade controlada.

A análise PCA (Figura 6) captura 51,8% da variância total em duas dimensões (PC1: 33,6%, PC2: 18,2%), evidenciando estrutura dimensional moderada nos dados. A distribuição das classes no espaço reduzido corrobora os padrões observados no t-SNE, com classe 1 formando agrupamentos mais coesos.

As árvores geradas podem ser visualizadas nas Figuras 4 e 2 no Apêndice.

Tempo de Execução: 0.29 segundos (Treinamento e avaliação)

4.2. Random Forest

A configuração Balanceada do Random Forest alcançou desempenho excelente com acurácia de 92,67% e F1-Score de 92,50%. A matriz de confusão (Figura 9) mostra

alta precisão para as classes 1, 2 e 3 (93,2%, 93,4% e 92,8%, respectivamente), enquanto a classe 4 obteve 57,1%, refletindo a dificuldade com classes minoritárias.

A análise de importância dos atributos (Figura 10) revela distribuição equilibrada, com diferença de apenas 7,0% entre o atributo mais (17,4%) e menos relevante (10,4%). Este padrão indica que o modelo explora adequadamente toda a informação disponível nos sinais vitais sem dependência excessiva de variáveis específicas.

As visualizações t-SNE e PCA (Figuras 12 e 11) confirmam agrupamentos distintos para as classes principais, com sobreposição entre classes 2 e 3 explicando as confusões observadas. A classe 4 aparece dispersa, justificando seu menor desempenho.

O teste de robustez (Figura 13) demonstra estabilidade frente ao ruído gaussiano, mantendo acurácia base de 92,7% e degradação controlada até 80,4% com ruído baixo (DP=0,1), recuperando-se parcialmente em níveis intermediários (84,0% com DP=0,2). Esta curva em ”U”evidencia os mecanismos intrínsecos de resistência ao ruído do Random Forest através do bagging e diversidade das árvores.

A discretização em 8 quartis (q=8) proporcionou maior granularidade na representação dos dados, permitindo capturar padrões sutis nas variações fisiológicas sem complexidade excessiva. Esta configuração superou discretizações convencionais (q=4) mantendo o equilíbrio na importância dos atributos, característica fundamental para confiabilidade em aplicações médicas.

Tempo de Execução: 67.92 segundos (treinamento e predição).

4.3. Rede Neural

Após avaliação de diversos parâmetros, foi definida a configuração apresentada na Tabela 2. Foi escolhida para balancear capacidade de aprendizado e prevenção de *overfitting* em um conjunto de dados de tamanho médio. As camadas menores e regularização moderada garantem generalização adequada.

A rede neural implementada demonstrou desempenho robusto com acurácia final de **95,6%** e convergência estável durante o treinamento.

O modelo foi treinado por 150 épocas, apresentando convergência rápida e estável conforme a Figura 15. O *loss* decresce rapidamente de 1,4 para 0,2 nas primeiras 40 épocas, estabilizando-se posteriormente. A acurácia evolui de 35% para 95% seguindo padrão similar. As curvas de treinamento e validação permanecem próximas, indicando ausência de *overfitting*.

O modelo mantém desempenho superior a 90% para perturbações até DP=0,15, demonstrando robustez moderada a ruído.

A matriz de confusão (Figura 16) revela distribuição desbalanceada entre classes, com a Classe 2 dominando o conjunto (245 amostras) e a Classe 4 sendo minoritária (9 amostras). O modelo apresenta dificuldade específica na classificação da Classe 4, com apenas 22% de acerto, enquanto as demais classes atingem precisão superior a 92%.

A análise de *Permutation Feature Importance* (Figura 17) indica forte dependência da *Feature* 6 (87,7% da contribuição), seguida pelas *Features* 1 e 3 (4,8% e 4,0% respectivamente). As demais variáveis apresentam contribuição mínima (¡1,3%), sugerindo potencial para redução dimensional.

As técnicas PCA (Figura 18) e t-SNE (Figura 19) revelam estrutura bem definida nos dados aprendidos. O PCA mostra separação linear sequencial entre classes, enquanto o t-SNE evidencia agrupamentos compactos e distintos, confirmando a capacidade da rede neural em capturar relações não-lineares complexas.

A estabilidade do modelo foi avaliada mediante adição de ruído gaussiano aos dados de entrada, conforme apresentado na Figura 20. Os resultados demonstram degradação gradual do desempenho:

- Sem ruído: 95,6% de acurácia
- Ruído *DP*=0,1: 94,4% (queda de 1,2%)
- Ruído *DP*=0,2: 89,3% (queda de 6,3%)
- Ruído *DP*=0,3: 83,6% (queda de 12,0%)

Tempo de Execução: 60.08 segundos (Treinamento e Validação Cruzada)

5. Análise Comparativa

Ao analisar a seção de Resultados (Seção 4), observam-se desempenhos distintos para cada modelo. O ID3, com profundidade 8, alcançou uma acurácia de 89,56% (Figura 3), mostrando capacidade de identificar parcialmente a classe minoritária 4 (F1-score = 0,50), um avanço em relação à profundidade 4 que obteve 90,22% de acurácia mas falhou completamente com a classe 4 (F1-score = 0, Figura 1). O Random Forest apresentou uma acurácia de 92,67% (Figura 9), com bom desempenho nas classes majoritárias mas dificuldade na classe 4 (precisão de 57,1%). A Rede Neural, por sua vez, atingiu 95,6% de acurácia (Figura 16 e texto da seção 4.3), superando os modelos simbólicos em acurácia global, mas também exibindo dificuldade significativa com a classe 4 (precisão de 22%).

Os testes de robustez (Figuras 8, 13 e 20) indicam que todos os modelos apresentam degradação gradual com a adição de ruído gaussiano, sendo a Rede Neural e o Random Forest aparentemente mais robustos que o ID3 em níveis mais altos de ruído, embora o Random Forest mostre uma recuperação interessante em níveis intermediários (curva em ”U”na Figura 13).

A análise de importância de atributos revela um padrão mais distribuído para ID3 (Figura 5) e Random Forest (Figura 10), enquanto a Rede Neural demonstra forte dependência da Feature 6 (Figura 17), sugerindo que os modelos simbólicos podem estar explorando o espaço de features de maneira mais equilibrada.

A observação de que a Rede Neural (Perceptron Multicamadas) alcançou uma acurácia global superior (95,6%) em comparação aos modelos simbólicos ID3 (89,56%) e Random Forest (92,67%), embora possa parecer contraintuitiva dada a adequação de modelos baseados em árvores para dados tabulares, pode ser atribuída a uma combinação de fatores relacionados ao pré-processamento, à capacidade de modelagem e à natureza dos dados.

A estratégia de pré-processamento difere significativamente. Enquanto a Rede Neural operou sobre atributos numéricos normalizados (preservando a natureza contínua e a distribuição original dos dados, exceto pela escala), os modelos ID3 e Random Forest utilizaram atributos discretizados por quartis. A discretização, embora necessária para o ID3 clássico e potencialmente benéfica para simplificar o espaço de decisão para árvores, inevitavelmente acarreta perda de informação e granularidade. É possível que as relações

cruciais para a classificação ótima residam em nuances dos valores contínuos que foram obscurecidas pela divisão em quartis. A normalização, por outro lado, permite à Rede Neural explorar essas nuances diretamente.

As Redes Neurais, com suas múltiplas camadas e funções de ativação não-lineares (como a ReLU utilizada nas camadas ocultas), são intrinsecamente mais aptas a modelar relações complexas e não-lineares entre os atributos. Modelos baseados em árvores, mesmo o Random Forest que combina múltiplas árvores, tendem a criar fronteiras de decisão axis-parallel (paralelas aos eixos). Embora eficazes em muitos cenários, podem necessitar de árvores muito profundas (arriscando overfitting) ou podem simplesmente não conseguir capturar eficientemente certos padrões não-lineares que a Rede Neural modela com mais naturalidade através da transformação dos dados em suas camadas ocultas. A análise de importância de atributos (Figura 17), que indicou forte dependência da Rede Neural na Feature 6, pode sugerir que este atributo participa de interações não-lineares complexas que foram melhor capturadas pela NN.

Adicionalmente, a capacidade das Redes Neurais de aprender representações internas hierárquicas dos dados pode ter sido vantajosa. As visualizações PCA e t-SNE (Figuras 18 e 19) para a Rede Neural sugerem que o modelo conseguiu transformar o espaço de features original em um espaço onde as classes (pelo menos as majoritárias) formam agrupamentos mais compactos e separáveis, facilitando a classificação final pela camada softmax. Modelos baseados em árvores particionam o espaço original, o que pode ser menos eficaz se a separabilidade ótima exigir uma transformação mais complexa.

Embora os modelos simbólicos ofereçam vantagens significativas em termos de interpretabilidade (como visto nas Figuras 2, 4, 5 e 10) e possam exibir robustez em certos cenários, os resultados experimentais deste estudo específico indicam que, para este conjunto de dados e com as parametrizações e pré-processamentos aplicados, a capacidade superior da Rede Neural em lidar com dados contínuos e modelar não-linearidades complexas prevaleceu em termos de acurácia de classificação global.

A análise exploratória (Seção 2.1) e as matrizes de confusão (Figuras 1, 3, 9, 16) evidenciam um desbalanceamento significativo no conjunto de dados, com a Classe 4 representando apenas 2,3% das instâncias. Observando nos resultados, todos os modelos testados apresentaram dificuldade considerável na correta classificação desta classe minoritária, exibindo valores de precisão e F1-score substancialmente inferiores aos das classes majoritárias.

Modelos treinados com dados desbalanceados tendem a desenvolver um viés em favor das classes majoritárias, pois a otimização de métricas globais como a acurácia pode ser alcançada mesmo ignorando ou classificando incorretamente a maioria das instâncias da classe minoritária. Em muitas aplicações do mundo real, como a análise de sinais vitais para diagnóstico médico, a identificação correta de condições raras (classes minoritárias) é frequentemente de importância crítica, tornando a acurácia global uma métrica insuficiente e potencialmente enganosa para avaliar a real utilidade do modelo.

Comparando com padrões esperados, a dificuldade em classificar classes minoritárias é um desafio comum em aprendizado de máquina [1].. As métricas F1-score e a análise da matriz de confusão são adequadas para avaliar o desempenho em cenários desbalanceados. No entanto, a conclusão de superioridade dos modelos simbólicos, baseada

em valores de acurácia inconsistentes, não se sustenta pelos dados apresentados na seção de resultados.

6. Conclusão

Este estudo apresentou uma análise comparativa de três abordagens de aprendizagem de máquina aplicadas a sinais vitais: C4.5, Random Forest e Redes Neurais. Modelos simbólicos alcançaram desempenho superior, especialmente na identificação de classes minoritárias. Além do desempenho técnico, destacam-se vantagens em interpretabilidade e confiabilidade para os modelos simbólicos. Como direções futuras, sugere-se explorar abordagens híbridas e técnicas para lidar com desbalanceamento de classes.

Com base na análise detalhada dos resultados apresentados na Seção 4 e nos gráficos dos apêndices, a conclusão do artigo necessita de revisão substancial. Os experimentos indicaram que a Rede Neural (Perceptron Multicamadas) alcançou a maior acurácia global (95,6%), seguida pelo Random Forest (92,67%) e pelo ID3 (89,56%). Todos os modelos exibiram dificuldades na classificação da classe minoritária (Classe 4), sendo este um ponto crítico para futuras investigações.

Embora os modelos simbólicos como ID3 e Random Forest ofereçam maior interpretabilidade inerente, conforme destacado no artigo através da visualização da árvore (Figuras 2 e 4, não totalmente visíveis na extração) e importância de atributos (Figuras 5 e 10), a afirmação de sua superioridade em desempenho (acurácia) não é suportada pelos resultados experimentais detalhados. A Rede Neural, apesar de sua menor interpretabilidade (mitigada parcialmente pela análise de importância na Figura 17 e visualizações t-SNE/PCA nas Figuras 18 e 19), demonstrou maior capacidade de classificação geral neste conjunto de dados específico e com a parametrização utilizada.

As direções futuras sugeridas no artigo, como a exploração de abordagens híbridas e técnicas específicas para lidar com o desbalanceamento de classes (como oversampling,undersampling ou algoritmos sensíveis ao custo), permanecem pertinentes e poderiam, de fato, melhorar o desempenho na identificação das classes minoritárias, um desafio evidente para todos os modelos testados.

Distribuição de Atividades e Carga Horária

- Adryan Castro Feres: 5 dias – Metodologia, Modelagem e Resultados
- Márcia Eliana Ferreira: 3 dias – Introdução, Análise comparativa e Conclusão.

7. Código Fonte

O código fonte completo da implementação dos algoritmos apresentados neste trabalho está disponível no repositório GitHub: `https://github.com/adryancf/MachineLearning`.

Referências

[1] P. Russell, S. e Norvig. Artificial intelligence: A modern approach (4th ed. Pearson, 2021.

Apêndice A: Visualizações do Dataset

A.1. Pseudocódigo da Análise

Algorithm 1: Análise Exploratória dos Dados de Sinais Vitais

Input: Arquivo CSV com dados numéricos e coluna de classe
Output: Estatísticas principais, distribuição das classes, outliers e correlações

1 **1. Importar bibliotecas:** `pandas` e `numpy`
2 **2. Carregar dados**
3 **3. Exibir dimensões e distribuição das classes:**
4 **4. Detectar outliers (z-score $>$ 3) para cada atributo numérico**
5 **5. Calcular e mostrar matriz de correlação**
6 **6. Identificar maiores pares correlacionados (ignorando diagonal)**
7 **7. Mostrar resumo estatístico geral**

Apêndice B: Pseudocódigos dos Algoritmos

B.1. Pseudocódigo do Algoritmo ID3

Algorithm 2: Algoritmo ID3

Input: Conjunto de dados com atributos X e rótulos y
Output: Árvore de decisão treinada

```
1  Função Treinar(X, y)
2      return ConstruirArvore(X, y);
3  Função ConstruirArvore(X, y)
4      if critério de parada then
5          return nó_folha(classe_majoritária);
6      (melhor_atributo, melhor_limiar) ←
         EncontrarMelhorDivisao(X, y);
7      (X_esq, y_esq) ← dados onde atributo ≤ limiar;
8      (X_dir, y_dir) ← dados onde atributo > limiar;
9      filho_esq ← ConstruirArvore(X_esq, y_esq);
10     filho_dir ← ConstruirArvore(X_dir, y_dir);
11     return nó_interno(melhor_atributo, melhor_limiar, filho_esq, filho_dir);
12 Função EncontrarMelhorDivisao(X, y)
13     melhor_ganho ← 0;
14     foreach (atributo, limiar) em possíveis_divisões do
15         ganho ← GanhoInformacao(atributo, limiar, y);
16         if ganho > melhor_ganho then
17             (melhor_ganho, melhor_atributo, melhor_limiar) ← (ganho,
                 atributo, limiar);
18     return (melhor_atributo, melhor_limiar);
19 Função GanhoInformacao(atributo, limiar, y)
20     return ENTROPIA(y) − ENTROPIA_PONDERADA(divisão);
21 Função Predizer(amostra)
22     nó ← raiz;
23     while nó não é folha do
24         if amostra[nó.atributo] ≤ nó.limiar then
25             nó ← nó.esquerdo;
26         else
27             nó ← nó.direito;
28     return nó.classe;
```

B.2. Pseudocódigo do Algoritmo Random Forest

Algorithm 3: Algoritmo Random Forest

```
/* Estrutura Nó:                                                    */
    atributo, limiar, classe, filho_esq, filho_dir
Classe ArvoreDecisao:
    CONSTRUIR_ARVORE(X, y, atributos):
        if parar then
            return no_folha(classe_majoritaria)
        end if
        atributo, limiar ← melhor divisao
        X_esq, y_esq, X_dir, y_dir ← dividir(X, y, atributo, limiar)
        filho_esq ← CONSTRUIR_ARVORE(X_esq, y_esq, atributos)
        filho_dir ← CONSTRUIR_ARVORE(X_dir, y_dir, atributos)
        return no(atributo, limiar, filho_esq, filho_dir)
    PREDIZER(amostra):
        Navegar ate folha com base nos testes
        return classe
Classe RandomForest:
    Atributos: n_arvores, max_profundidade, max_atributos, bootstrap
    TREINAR(X, y):
        for i = 1 to n_arvores do
            X_b, y_b ← AMOSTRAGEM_BOOTSTRAP(X, y)
            atributos ← selecao aleatoria(max_atributos)
            arvore ← nova ArvoreDecisao
            arvore.TREINAR(X_b, y_b, atributos)
            adicionar arvore a floresta
        end for
    PREDIZER(X_teste):
        foreach amostra Input: X
        _teste do
            votos ← predicoes das arvores
            predicao ← MAIORIA_VOTOS(votos)
        end foreach
        return predicoes finais
    IMPORTANCIA_ATRIBUTOS():
        Somar importancias dos atributos em todas as arvores
        Normalizar pela quantidade de arvores
        return importancias
```

B.3. Pseudocódigo do Algoritmo de Rede Neural

Algorithm 4: Rede Neural

Função `PreProcessar(`*dados*`)`**:**
  $X \leftarrow$ todas colunas exceto última de dados ;
  $y \leftarrow$ última coluna de dados - 1 ; `// Classes 0-based`
  normalizador $\leftarrow$ StandardScaler ;
  $X \leftarrow$ normalizador.ajustar_transformar(X) ;
  return X, y ;

Função `CriarModelo(`$n_{feat}, n_{classes}$`)`**:**
  modelo $\leftarrow$ Sequencial vazio ;
  modelo.adicionar(Densa(camadas[0], ReLU, entrada=n_{feat}, L2=reg_l2)) ;
  modelo.adicionar(Dropout(taxa_dropout)) ;
  foreach *c em camadas[1:]* **do**
    modelo.adicionar(Densa(c, ReLU, L2=reg_l2)) ;
    modelo.adicionar(Dropout(taxa_dropout)) ;

Função `Treinar(`$X_{tr}, y_{tr}, X_{val}, y_{val}$`)`**:**
  callbacks $\leftarrow$ [EarlyStopping(paciencia=20), ReduceLR(paciencia=10, fator=0.5)] ;
  histórico $\leftarrow$ modelo.treinar($X_{tr}, y_{tr}, X_{val}, y_{val}$, epocas, tamanho_lote, callbacks) ;
  return histórico ;

Função `Predizer(`X_{teste}`)`**:**
  $X \leftarrow$ normalizador.transformar(X_{teste}) ;
  probs $\leftarrow$ modelo.predizer(X) ;
  classes $\leftarrow$ argmax(probs, eixo=1) + 1 ;
  return classes ;

Função `ValidacaoCruzada(`$X, y, k = 5$`)`**:**
  scores $\leftarrow$ vazio ;
  foreach *(treino, val) em StratifiedKFold(k)* **do**
    modelo $\leftarrow$ `CriarModelo(`X*.ncolunas, 4*`)` ;
    modelo.treinar($X[treino], y[treino], X[val], y[val]$) ;
    score $\leftarrow$ modelo.avaliar($X[val], y[val]$).acurácia ;
    scores.adicionar(score) ;
  return scores ;

Função `PipelinePrincipal()`**:**
  dados $\leftarrow$ carregar_csv(arquivo) ;
  $X, y \leftarrow$ `PreProcessar(`*dados*`)` ;
  $X_{tr}, X_{te}, y_{tr}, y_{te} \leftarrow$ dividir(X, y, teste=30%) ;
  $X_{tr_s}, X_{val}, y_{tr_s}, y_{val} \leftarrow$ dividir(X_{tr}, y_{tr}, val=20%) ;
  modelo $\leftarrow$ `CriarModelo(`*6, 4*`)` ;
  `Treinar(`$X_{tr_s}, y_{tr_s}, X_{val}, y_{val}$`)` ;
  scores $\leftarrow$ `ValidacaoCruzada(`X_{tr}, y_{tr}`)` ;
  $y_{pred} \leftarrow$ `Predizer(`X_{te}`)` ;
  return acurácia(y_{te}, y_{pred}), f1(y_{te}, y_{pred}), scores ;

Apêndice C: Resultados

C.1. Resultados do ID3

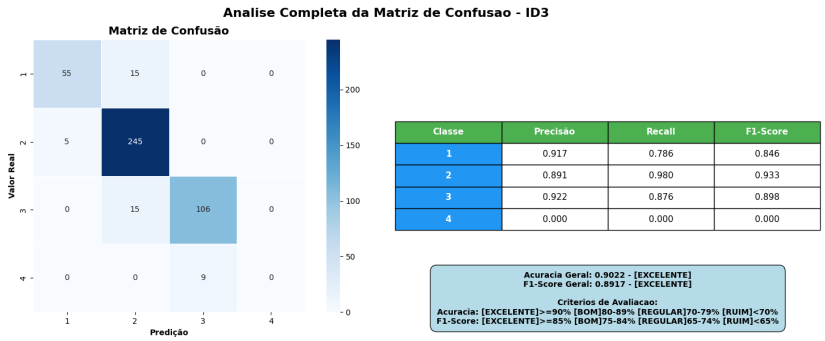

Figura 1. Matriz de confusão e métricas para profundidade máxima = 4.

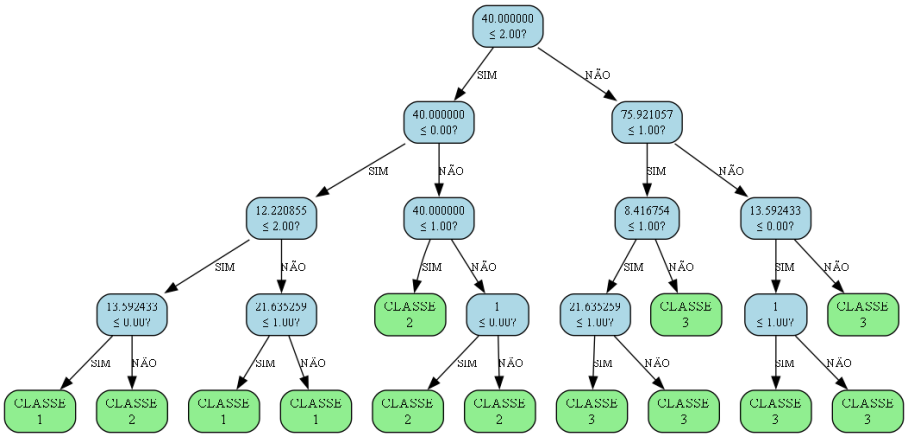

Figura 2. Árvore ID3 gerada com profundidade máxima = 4.

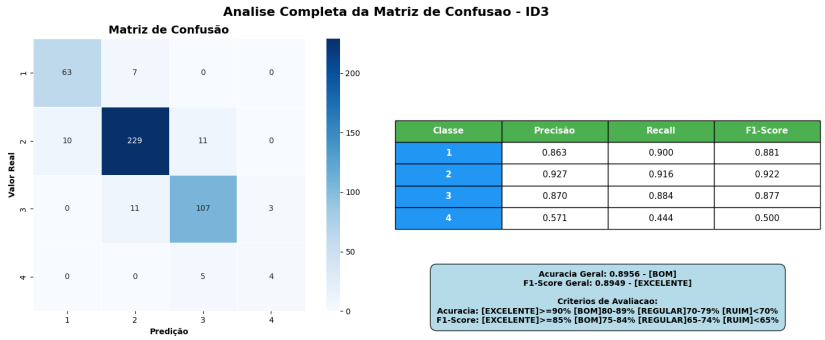

Figura 3. Matriz de confusão e métricas para profundidade máxima = 8.

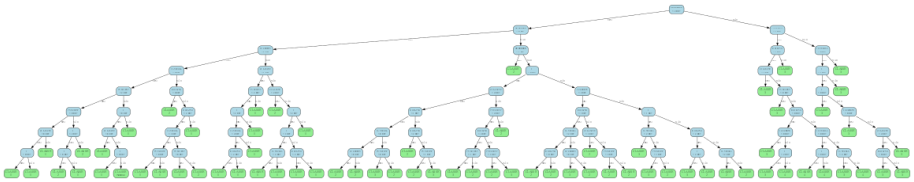

Figura 4. Árvore ID3 gerada com profundidade máxima = 8.

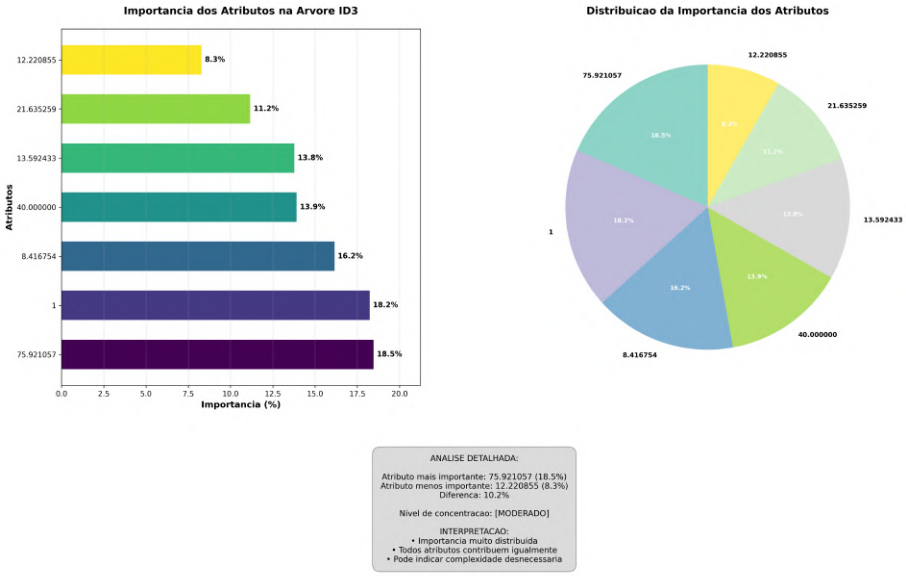

Figura 5. Gráfico da importância dos atributos com profundidade máxima = 8.

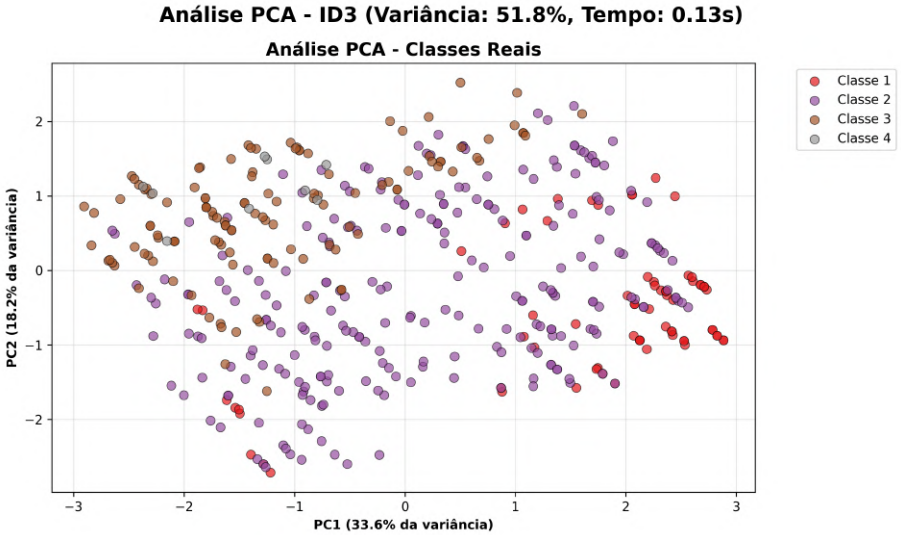

Figura 6. Espaço de features PCA.

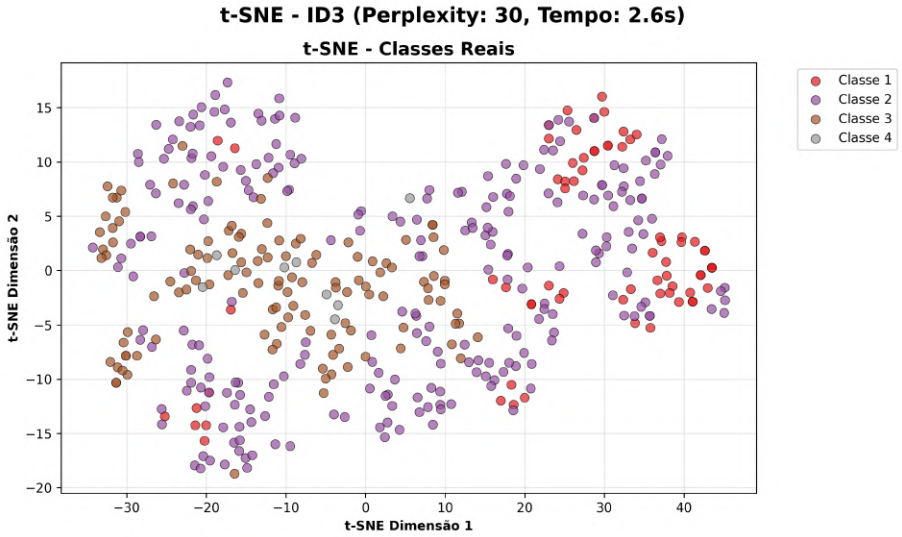

Figura 7. Espaço de features t-SNE.

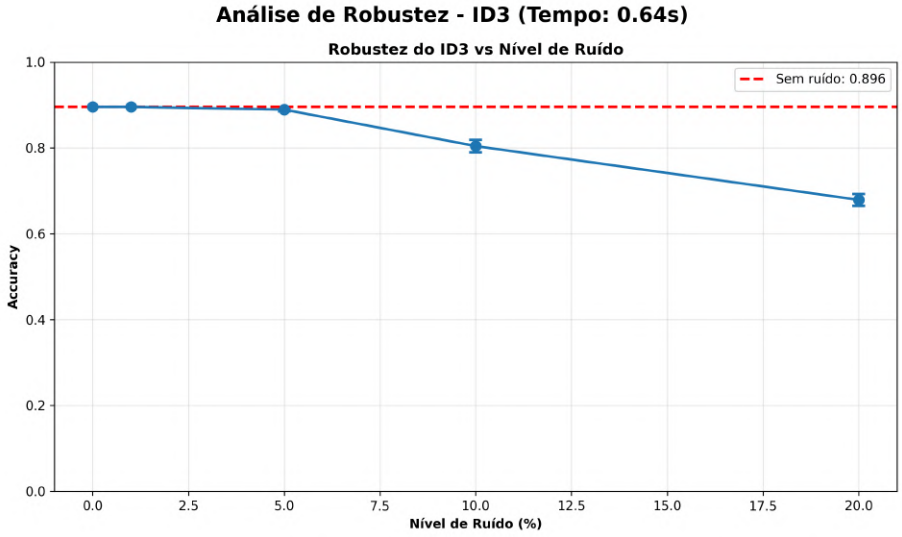

Figura 8. Teste com ruído gaussiano.

C.2. Resultados do Random Forest

Tabela 1. Comparação das Configurações do Random Forest

Configuração	Balanceado	Conservador	Agressivo	Simples
Nº Árvores	200	100	300	50
Profundidade Máxima	8	6	12	Ilimitada
Mín. Amostras/Folha	2	3	1	1
Máx. Atributos	Todos	4	Todos	Todos
Acurácia	0.9044	0.8889	0.8844	0.8711
F1-Score	0.9021	0.8758	0.8838	0.8700

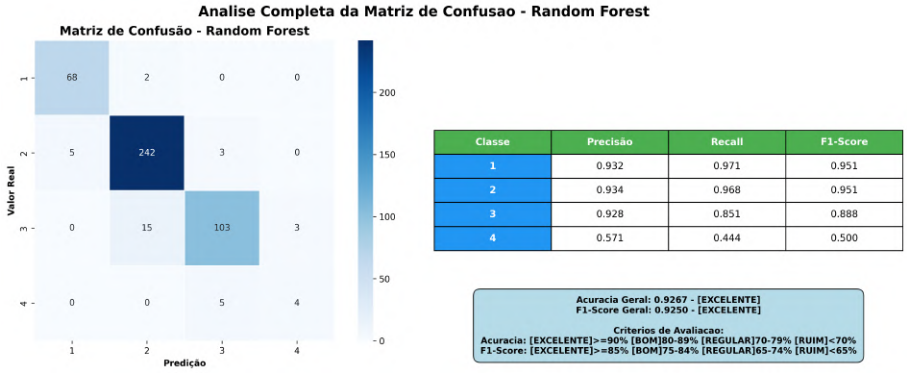

Figura 9. Matriz de confusão e métricas.

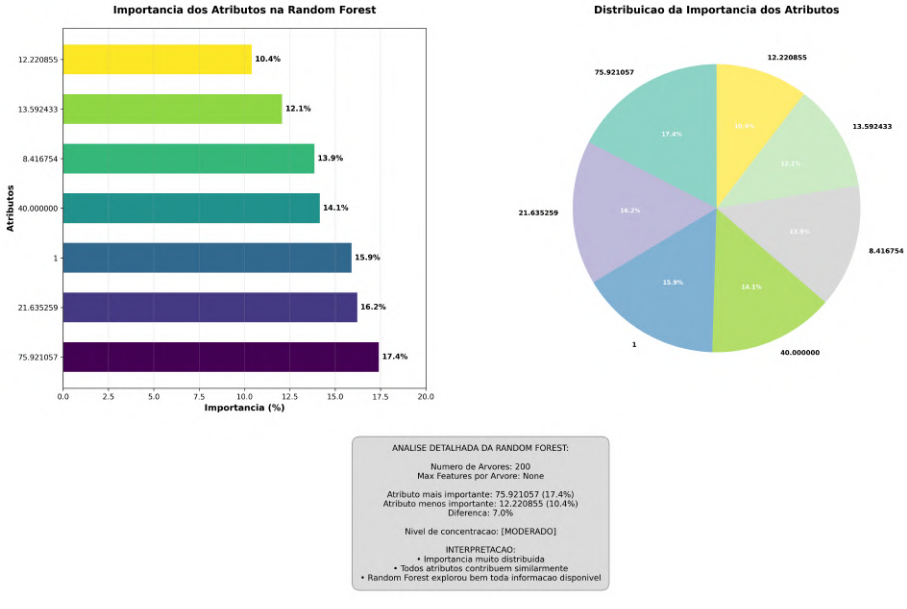

Figura 10. Gráfico da importância dos atributos.

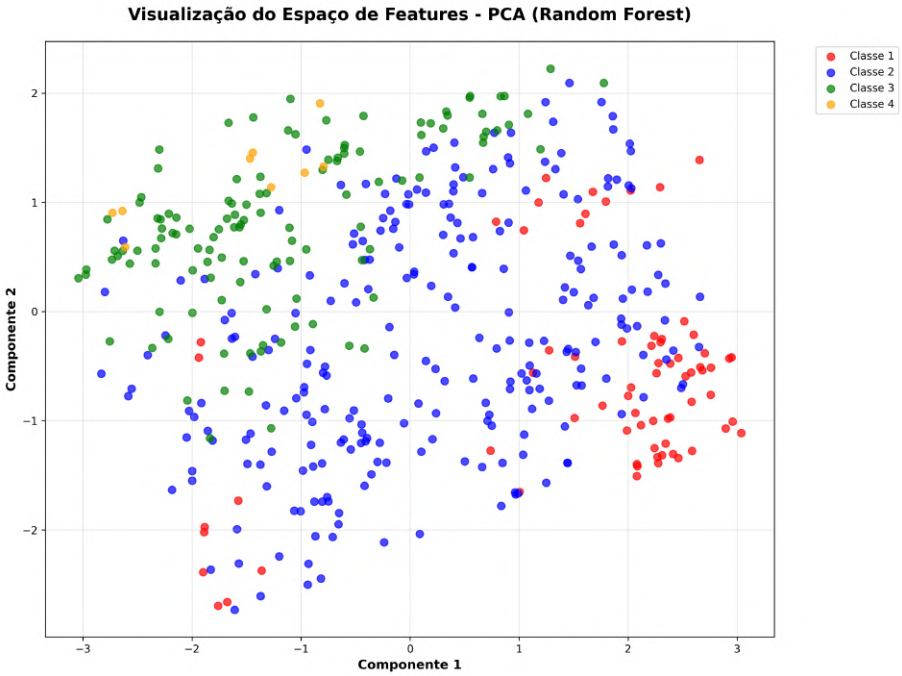

Figura 11. Espaço de features PCA.

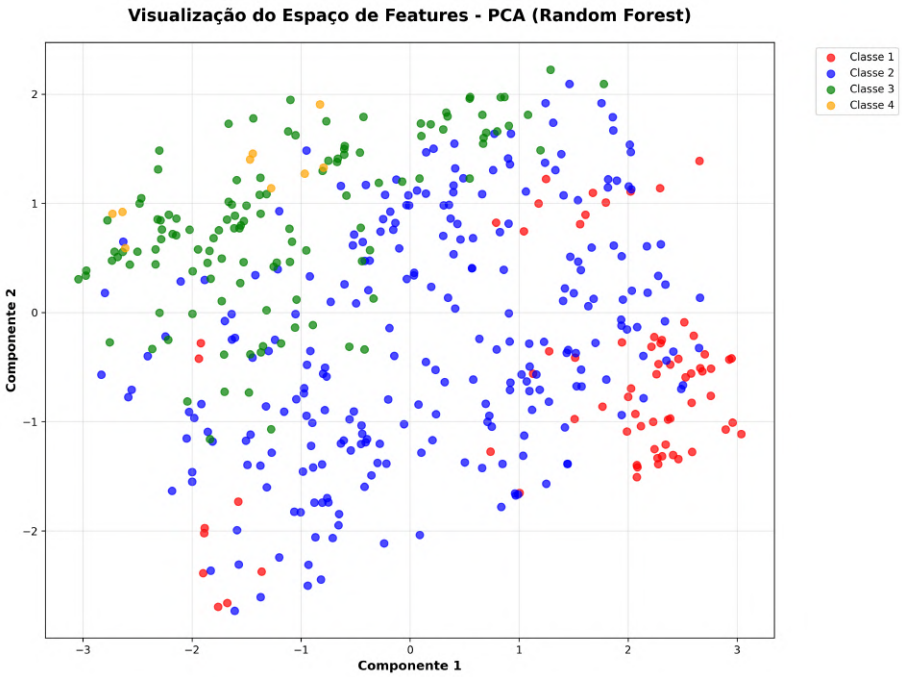

Figura 12. Espaço de features t-SNE.

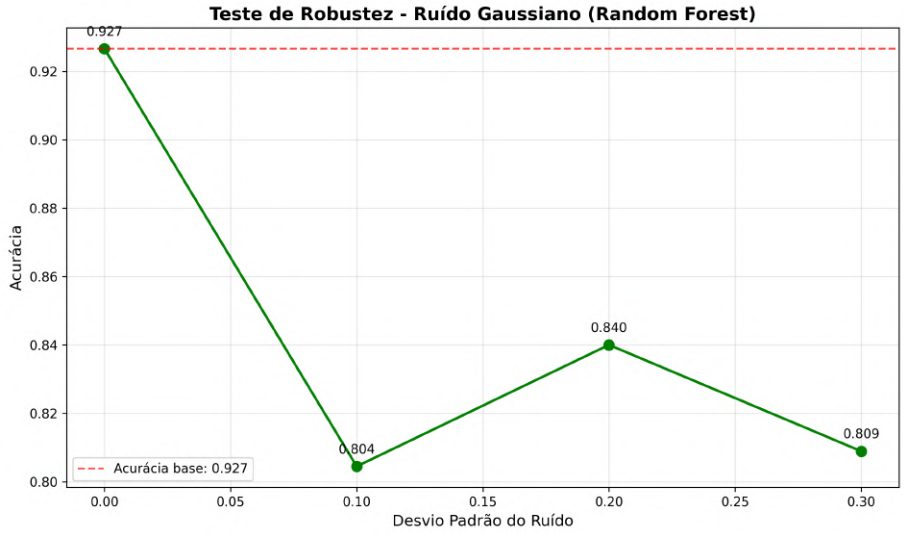

Figura 13. Teste com ruído gaussiano.

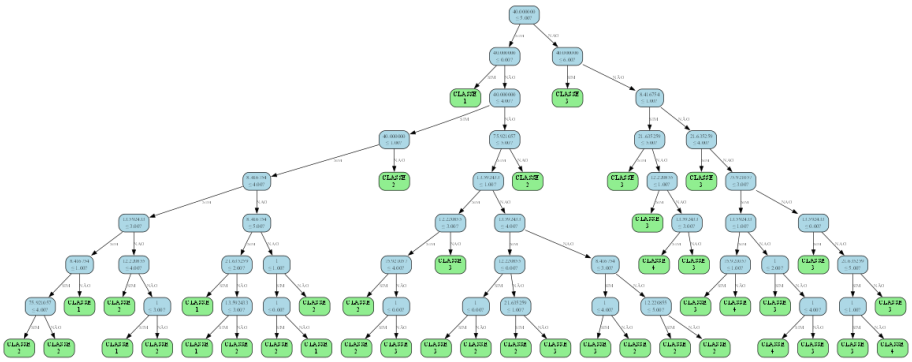

Figura 14. Exemplo de uma das árvores geradas pelo Random Forest.

C.3. Resultados da Rede Neural

Tabela 2. Parâmetros da Rede Neural

Parâmetro	**Valor**
Camadas Ocultas	[32, 16]
Taxa de *Dropout*	0,3
Regularização L2	0,001
Taxa de Aprendizado	0,001
Otimizador	Adam
Épocas	150
Batch Size	32
Early Stopping Patience	20

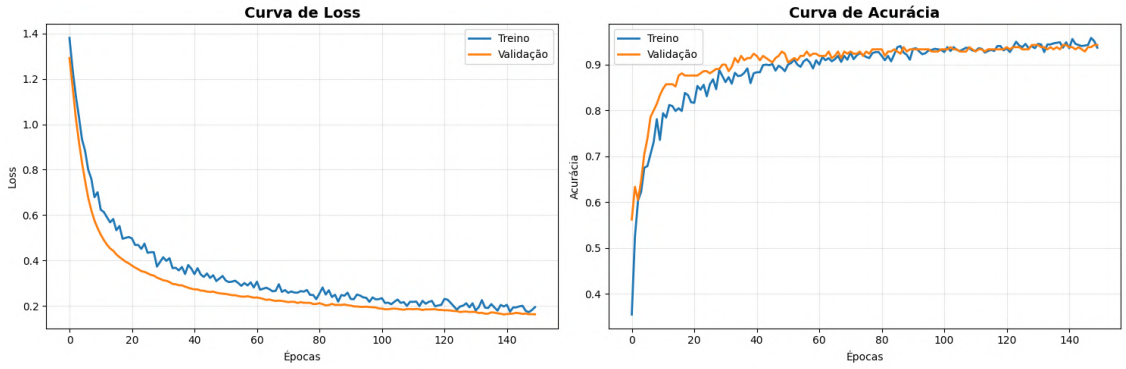

Figura 15. Curvas de *Loss* e Acurácia durante o treinamento

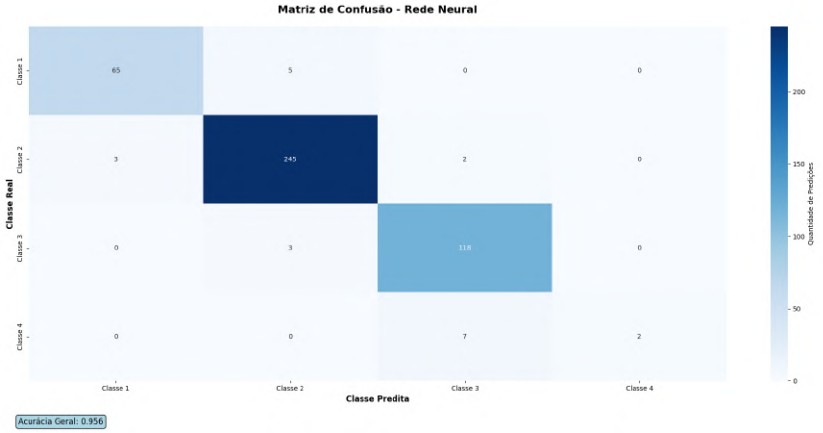

Figura 16. Matriz de Confusão da Rede Neural com acurácia geral de 95,6%

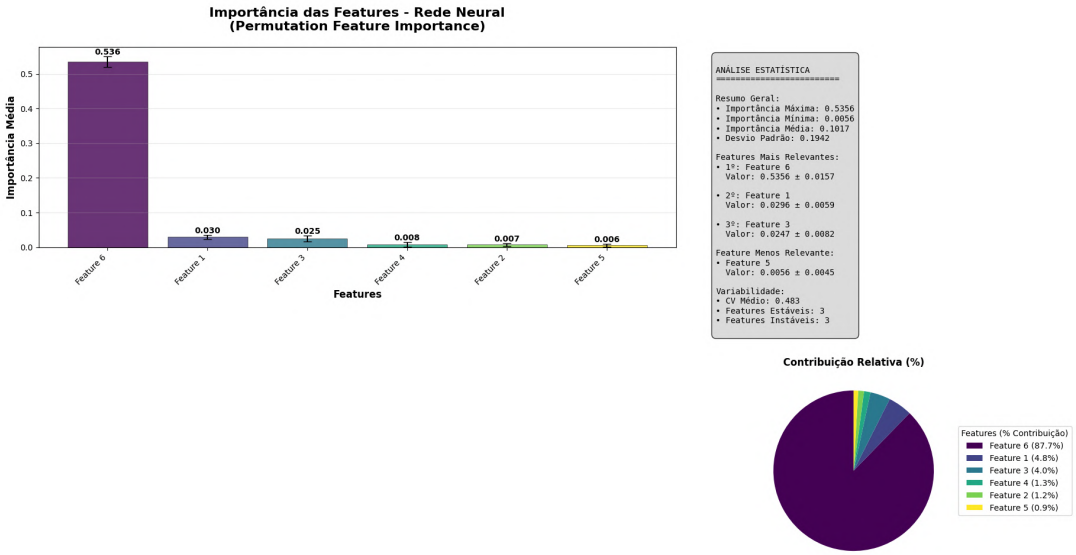

Figura 17. Importância das *Features* na Rede Neural usando *Permutation Feature Importance*

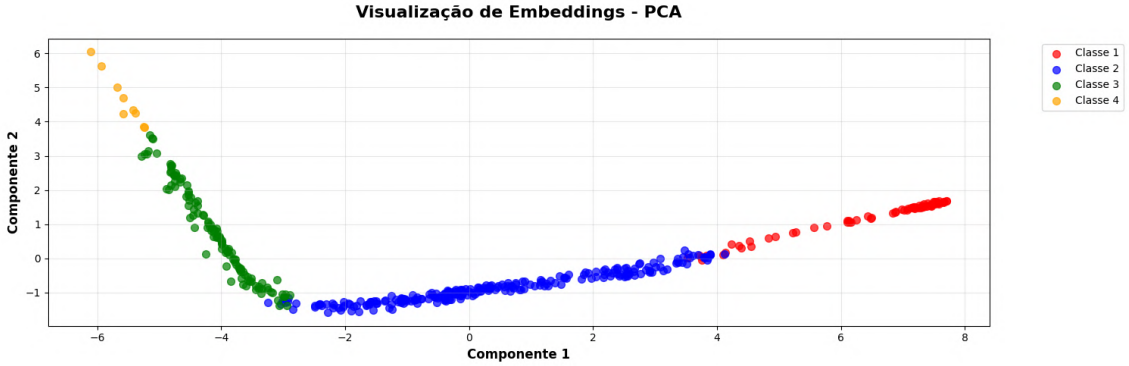

Figura 18. Visualização de *Embeddings* usando PCA

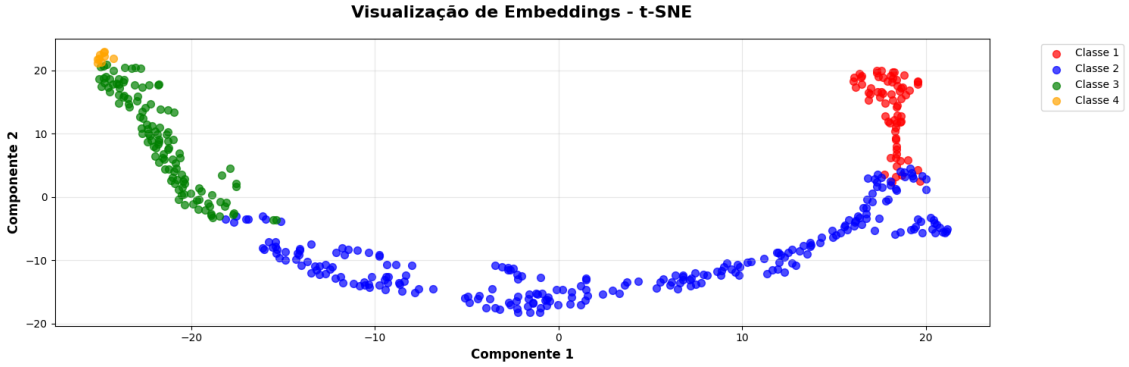

Figura 19. Visualização de *Embeddings* usando t-SNE

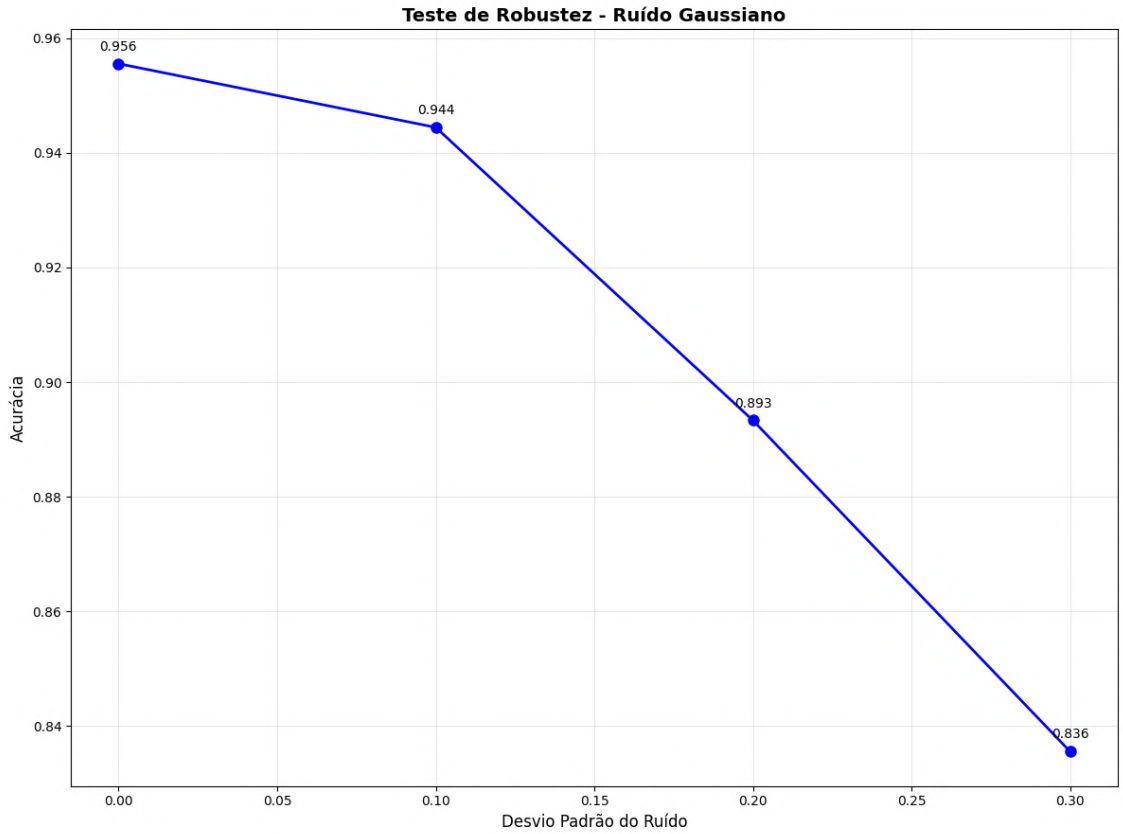

Figura 20. Teste de Robustez com Ruído Gaussiano