Otimização de Florestas Aleatórias com metaheurísticas evolucionárias para a predição de doenças cardíacas.

Daniel C. Anselmo[1]

[1]Programa de Pós-Graduação em Ciência da Computação(PPGCC)
Instituto Federal de Educação, Ciência e Tecnologia do Ceará. (IFCE)
Fortaleza – CE – Brazil

daniel.carvalho60@aluno.ifce.edu.br

Abstract.

Resumo. *Segundo a Organizaço Mundial da Sade (OMS), as doenças cardiovasculares representam a principal causa de morte no mundo, vitimando em torno de 20,5 pessoas por ano. Existem alguns fatores de risco que contribuem para esse índice de mortes elevado, como tabagismo, sedentarismo, consumo em excesso de álcool, alimentação inadequada, e outros mais. Uma estratégia para reduzir esse índice de mortes é a detecção de risco de doença cardíaca em uma pessoa. Identificar precocemente se uma pessoa tem potencial para desenvolver algum problema cardiovascular, ou se ela já tem algum problema, é essencial para definir um tratamento eficaz e evitar mortes prematuras. Assim sendo, dado a urgência do problema e a necessidade de se desenvolver métodos eficientes de detecção, este trabalho propõe uma estratégia baseada em metaheurísticas evolutivas para otimizar os hiperparâmetros de árvores de decisão, para realizar a predição de risco de doença cardíaca.*

1. Indrodução

As doenças cardiovasculares (DCV) continuam sendo a principal causa de morte em todo o mundo. Se trata de doenças letais e crônicas e que Segundo a Organização Mundial da Saúde (OMS), em torno de 20,5 milhões de pessoas morrem por causa de doenças cardiovasculares, representando cerca de 31,5% das mortes ocorridas no mundo. É estimado também que haja um aumento desse número de mortes até 2030 para 24,2 milhões de mortes a cada ano. Aproximadamente 85% dos casos de mortes provocados por doenças cardiovasculares têm como causa ataques cardíacos e derrames. A doença cardíaca ocorre quando o coração não consegue levar sangue suficiente para o restante do corpo. Esse bloqueio do fluxo sanguíneo ocorre por causa do acúmulo de placas nas artérias. Isso gera alguns sintomas, alguns deles são: falta de ar, batimentos irregulares, tontura, suor frio, entre outros. É necessário um diagnóstico preciso e rápido, com o objetivo de diminuir a taxa de mortalidade dos pacientes (REDDY, 2021).

Com a tecnologia atual, é possível realizar realizar previsões de risco cardíaco através de técnicas de Aprendizado de Máquina (AM), construindo modelos estatísticos capazes de aprender com o histórico médico dos pacientes. É possível, através de algoritmos de aprendizado de máquina, identificar padrões e correlações que podem passar despercebidos em análises tradicionais (NASUTION et al., 2025). Um algoritmo muito relevante para esse problema é o *Random Forest* (RF). Esse algoritmo de classificação

tem ótimas características como boa precisão, robustez e uma ótima aptidão para evitar overfitting, ou seja, quando o modelo consegue atingir um bom nível de generalização. A previsão gerada por esse classificador é baseada nos resultados gerados por múltiplas árvores de decisão. (SEN e BARMAN, 2025)

A eficácia desse algoritmo depende de uma boa seleção e combinação de seus hiperparâmetros, como profundidade máxima das árvores, o número de árvores na floresta, e o número mínimo de amostras para dividir um nó. O objetivo desse estudo é aplicar duas metaheurísticas evolutivas, o algoritmo genético e o algoritmo de evolução diferencial, para realizar uma melhor combinação desses hiperparâmetros, a fim de verificar qual estratégia evolutiva obteve um melhor desempenho, tanto de classificação quanto de custo computacional.

2. Fundamentação teórica

Esta seção apresenta alguns conceitos de aprendizagem de máquina, mais especificamente sobre árvores de decisão e florestas aleatórias e sobre os algoritmos de otimização evolucionária.

2.1. Árvores de Decisão

A técnica de árvore de decisão funciona da seguinte forma: primeiro se estabelece a representação, que é o espaço de hipótese, e depois uma forma de aprender uma boa hipótese. Uma árvore de decisão é representada por uma função que recebe como entrada um vetor contendo valores de atributos e tem como retorno uma ”decisão”, um valor único de saída. Ela consegue retornar sua decisão após a execução de uma sequência de testes. Cada nó interno na árvore é equivalente a um teste do valor de algum dos atributos de entrada, A_i, e as ramificações dos nós são classificadas com os possíveis valores do atributo, $A_i = v_i k$. Os nós de folha da árvore representam o valor de retorno da função. (RUSSELL e NORVIG, 2013)

As árvores de decisão constroem sua estrutura com base na seleção de atributos que melhor separam as classes nos dados, utilizando medidas de impureza como o índice de Gini e o ganho de informação (Information Gain). Em cada nó da árvore, o algoritmo escolhe o atributo que gera a maior redução de incerteza na predição, dividindo o conjunto de dados em subconjuntos mais homogêneos. Essa abordagem recursiva continua até que determinados critérios de parada sejam atendidos, como profundidade máxima, número mínimo de amostras em um nó, ou a ausência de ganho significativo em novas divisões (RUSSELL e NORVIG, 2013).

Apesar de sua simplicidade, árvores de decisão são sensíveis a overfitting, especialmente quando crescem demais e memorizam os dados de treinamento. Para mitigar esse problema, técnicas como poda (pruning) são aplicadas para eliminar divisões pouco significativas. Outro desafio é a instabilidade: pequenas variações nos dados de entrada podem levar à construção de árvores completamente diferentes. No entanto, sua capacidade de lidar com dados categóricos e numéricos, ausência de necessidade de normalização e boa interpretabilidade tornam as árvores de decisão uma das escolhas mais utilizadas em problemas de classificação, particularmente em domínios onde a explicabilidade é essencial (MAIMON e ROKACH, 2014).

No contexto da predição de doenças cardíacas, a escolha de árvores de decisão se justifica por sua clareza na tomada de decisão e facilidade de interpretação dos resultados. Como os dados médicos geralmente estão organizados em formato tabular e incluem variáveis heterogêneas (idade, pressão arterial, colesterol, histórico familiar, etc.), a estrutura de decisão sequencial permite que profissionais da saúde compreendam facilmente os critérios utilizados pelo modelo. Além disso, estudos recentes demonstram que árvores de decisão, embora simples, apresentam desempenho competitivo na classificação de doenças cardíacas, especialmente quando combinadas com técnicas de ensemble como Random Forest ou quando otimizadas com algoritmos evolutivos (REDDY et al., 2021; NASUTION et al., 2025).

2.2. Florestas Aleatórias

As Florestas Aleatórias (FA) são um algoritmo de ensemble baseado na combinação de múltiplas árvores de decisão independentes, cada uma treinada com subconjuntos aleatórios dos dados e atributos. O objetivo principal é reduzir a variância do modelo e melhorar a generalização, superando o problema de overfitting comum em árvores únicas. Durante o treinamento, cada árvore é construída a partir de um bootstrap dos dados (amostragem com reposição), e, em cada nó, é considerado apenas um subconjunto aleatório de atributos para decidir a melhor divisão. A predição final é feita por votação majoritária (para classificação) ou média (para regressão) entre as árvores (BREIMAN, 2001).

Essa abordagem traz vantagens significativas: além de robustez a ruídos e outliers, o modelo lida bem com alta dimensionalidade e dados mistos (categóricos e numéricos). Outra característica importante é sua capacidade de estimar a importância dos atributos, o que o torna útil também em tarefas de seleção de variáveis. Apesar de menos interpretável do que uma árvore isolada, o Random Forest é frequentemente considerado um ótimo ponto de equilíbrio entre desempenho preditivo e interpretabilidade parcial, sendo amplamente utilizado em problemas de ciência de dados estruturada (GUEX, 2021; MAIMON e ROKACH, 2014).

A predição de doenças cardíacas envolve dados clínicos complexos, com múltiplos atributos interdependentes, como idade, pressão arterial, colesterol, frequência cardíaca, entre outros. Nesse contexto, o Random Forest se destaca por sua alta acurácia, estabilidade e tolerância a correlações entre variáveis, além de ser capaz de identificar os fatores que mais influenciam o risco. Diversos estudos recentes mostram que esse algoritmo supera modelos lineares como regressão logística em bases como o UCI Heart Disease Dataset, atingindo acurácias superiores a 90% (REDDY et al., 2021; NASUTION et al., 2025). Além disso, sua estrutura baseada em múltiplas árvores favorece a construção de soluções otimizáveis com algoritmos evolutivos, o que justifica sua escolha neste trabalho.

2.3. Otimização

A otimização é uma etapa crucial em diversos problemas de aprendizado de máquina, especialmente na busca por configurações que maximizem o desempenho dos modelos. No contexto da predição de doenças cardíacas, por exemplo, a escolha adequada dos hiperparâmetros pode impactar significativamente a acurácia e a generalização

do algoritmo. Entre as abordagens existentes, os métodos inspirados na natureza, conhecidos como modelos evolucionários, destacam-se por sua capacidade de explorar eficientemente grandes espaços de busca sem depender de derivadas ou suposições sobre a forma da função objetivo.

2.3.1. Modelos Evolucionários

Modelos evolucionários são algoritmos baseados em princípios da seleção natural e evolução biológica. Esses métodos utilizam mecanismos como seleção, cruzamento, mutação e competição para iterativamente evoluir soluções candidatas a um problema de otimização. Por serem heurísticos estocásticos, são particularmente úteis em cenários onde o espaço de busca é complexo, multimodal ou de natureza não linear — características comuns na otimização de hiperparâmetros de algoritmos de aprendizado de máquina (EIBEN e SMITH, 2015).

2.3.2. Algoritmo Genético

O Algoritmo Genético (AG) é um dos modelos evolucionários mais conhecidos e amplamente utilizados. Ele simula o processo de evolução natural ao operar sobre uma população de indivíduos representados por cromossomos, que codificam possíveis soluções do problema. A cada geração, indivíduos com melhor desempenho (fitness) são selecionados para reprodução, passando seus "genes" para novas soluções geradas por crossover e mutação. O AG é eficaz para encontrar soluções próximas do ótimo global mesmo em espaços de busca com muitas variáveis ou múltiplos picos de desempenho. No contexto deste trabalho, ele é utilizado para ajustar hiperparâmetros do Random Forest, como o número de árvores e a profundidade máxima, de forma automática e adaptativa (HOLLAND, 1992).

2.3.3. Evolução Diferencial

A Evolução Diferencial (ED) é um algoritmo evolucionário projetado principalmente para problemas de otimização contínua. Diferente do AG, que atua sobre cadeias codificadas de variáveis, a ED manipula diretamente vetores reais como candidatos à solução. O principal mecanismo da ED consiste em combinar indivíduos da população atual por meio de operações de diferença vetorial escalada, gerando candidatos mais diversos com potencial para escapar de mínimos locais. A seleção é feita de forma competitiva, onde apenas os indivíduos que melhoram a função objetivo permanecem. Graças à sua simplicidade e eficiência, a ED tem se destacado em aplicações como otimização de hiperparâmetros e ajuste fino de modelos de aprendizado de máquina (STORN e PRICE, 1997).

Para cada indivíduo da população(chamado de "vetor-alvo"), o ED escolhe três indivíduos diferentes: a b e c, e cria um indivíduo mutante usando uma diferença vetorial, com a seguinte fórmula:

$$\text{mutante} = a + F \cdot (b - c) \tag{1}$$

Sendo F um fator de escala que controla os passos da mutação. Esse mutante criado é então combinado com o "vetor-alvo", gerando assim um filho. Se esse filho for melhor que o "vetor-alvo", o "vetor-alvo" é então substituído pelo filho na próxima geração. Esse processo é repetido para toda a população.

3. Metodologia

Para realizar os experimentos desse estudo, foi utilizado uma base de dados contendo indicadores clínicos relacionados ao risco de doença cardíaca. Essa base de dados foi coletada no site da Kaggle. Os experimentos foram divididos em duas etapas: Random Forest + GA e Random Forest + ED. A tabela abaixo mostra os hiperparâmetros do algoritmo florestas aleatórias que foram escolhidos para serem otimizados.

Table 1. Hiperparâmetros selecionados

Hiperparâmetros	Definição
n_estimators	Número de árvores da floresta.
max_depth	Profundidade das árvores.
min_samples_split	Número mínimo de amostras.

Foram feitos testes com diferentes espaços de busca da profundidade das árvores, a fim de verificar se árvores menos profundas melhoram o desempenho da classificação e reduz o tempo de execução. A próxima seção apresenta a base de dados utilizada nos experimentos.

3.1. Base de dados e pré-processamento

A base de dados "Heart Disease" foi coletada no site da Kaggle, e contém muitos indicadores de saúde relacionados ao risco de doença cardíaca, como histórico de doença cardíaca na família, diabetes, colesterol, etc. Esse conjunto de dados possui 10000 casos de pacientes. A distribuição inicial das classes está dividida em 8000 casos de "não tem doença cardíaca" e 2000 casos de "tem doença cardíaca".

Inicialmente, foi feito uma análise exploratória no dados, a fim de verificar dados faltantes. Alguns atributos apresentaram dados faltantes. Foi então feito um tratamento de imputação preenchendo esses espaços vazios pela mediana da coluna, no caso de colunas numéricas, e pela moda(valor mais frequênte), no caso de colunas categóricas.

Após isso, foi feito um tratamento em dados categóricos, como por exemplo o atributo "diabetes" que tinha valores "sim" ou "não". Para essas colunas binárias, foi aplicado o LabelEncoder, transformando essas valores para 0 ou 1. Para as colunas com mais de duas categorias, foi aplicado o OneHotEncoder.

Foi observado também que havia um desbalanceamento entre as classes, sendo 8000 entradas para a classe "sem doença" e 2000 entradas para a classe "com doença". Para solucionar esse problema foi aplicado a técnica de SMOTE (Synthetic Minority Over-sampling Technique). O SMOTE é uma técnica de balanceamento de classes que

em vez de duplicar instâncias da classe minoritária, gera novos exemplos sintéticos ao interpolar dados existentes dessa classe. Isso ajuda os modelos a aprenderem melhor os padrões da classe minoritária, reduzindo o viés em favor da classe majoritária e melhorando o desempenho preditivo, especialmente em métricas como o recall e o F1-score. Esta abordagem foi proposta por Chawla et al. (2002), sendo amplamente adotada em problemas de classificação com dados desbalanceados. Após a aplicação dessa técnica, o balanceamento ficou assim: 8000 entradas para a classe ”sem doença” e 6000 entradas para a classe ”com doença”.

A figura abaixo mostra um exemplo de árvore de decisão com uma amostra de 100 pacientes. Para plotar essa árvore foi aplicado antes uma técnica de redução de dimensionalidade utilizando a Análise de Componentes Principais(PCA). O PCA representa o conjunto de dados em um espaço com menos dimensões, preservando ao máximo a variância original. No exemplo abaixo, para facilitar a visualização, os dados foram reduzidos a 2 componentes.

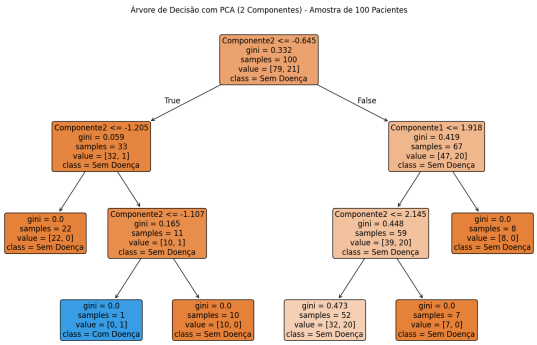

Figure 1. Árvore de decisão com PCA.

Com base nessa árvore decisão gerada, observou-se que a segunda componente foi o fator decisivo mais relevante, separando inicialmente pacientes com menor probabilidade de doença. Já a primeira componente atuou como refinamento, especialmente entre os casos com maior incerteza. Em diversos nó, observou-se pureza total (gini = 0), especialmente para a classe ”Sem doença”, indicando que a combinação dos dois componentes foi eficaz em capturar padrões de risco mesmo com apenas duas variáveis sintéticas.

Para realizar os experimentos, foi aplicado o PCA com 95% de componentes em relação aos atributos originais. Com isso, o algoritmo de PCA seleciona quantos componentes forem necessários para manter 95% da variância dos dados originais, ou seja, foi mantido 95% da informação dos dados originais. No gráfico abaixo é possível visualizar a curva de variância em relação ao número de componentes até ela atingir os 95%.

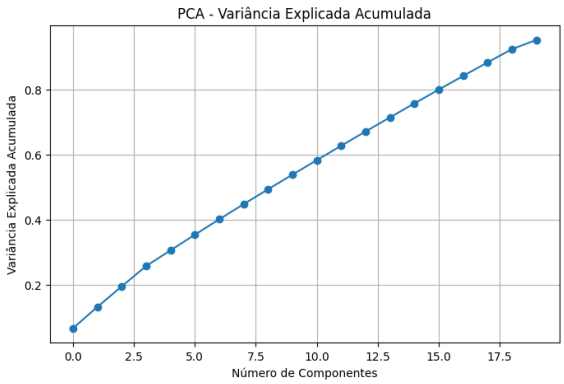

Figure 2. Árvore de decisão com PCA.

As proximas seções descrevem a implementação de cada estratégia de otimização.

3.2. Random Forest + GA

A implementação da otimização do Random Forest com GA foi feita utilizando a biblioteca *deap* do python.

Inicialmente, foi definida uma função fitness, que recebe como parâmetro um indivíduo da população contendo seus genes, que são os hiperparâmetros selecionados no espaço de busca pelo algoritmo, no seguinte formato: [n_estimators, max_depth, min_samples_split]. Após isso o modelo de Random Forest é treinado com os hiperparâmetros selecionados. Para realizar a avaliação de desempenho do modelo, foi feita uma validação cruzada com 3 dobras (folds). Essa validação funciona da seguinte forma: o conjunto de treinamento é dividido em três partes, duas partes pra treino e 1 para teste, isso acontece 3 vezes trocando o grupo de teste a cada vez. A vantagem é que o modelo é avaliado com três subconjuntos diferentes, reduzindo o viés da avaliação. A métrica dessa avaliação é a F1-score, que calcula a média harmônica entre precisão e revocação. Para calcular o retorno da função fitness, foi aplicado a seguinte fórmula:

$$\text{Fitness} = \bar{F}_1 - (0.5 \cdot \sigma + 0.001 \cdot d) \tag{2}$$

Onde:

- $\bar{F}_1$ representa a média do F1-score obtido por validação cruzada (3 folds);
- σ é o desvio padrão dos F1-scores entre os folds, penalizando instabilidade nos resultados;
- d corresponde à profundidade máxima da árvore (*max_depth*), penalizada levemente para evitar sobreajuste.

Após definir a função de fitness, foi definido o espaço de busca, a população de indivíduos, e as operações do algoritmo genético, mutação e cruzamento. O processo para selecionar os melhores indivíduos da população é o torneio. Nesse processo, é selecionado um subconjunto aleatório da população, e nesse subconjunto o que apresentar maior valor fitness é escolhido para reprodução. Esse processo é repetido várias vezes até formar uma nova geração. Essa seleção favorece indivíduos mais aptos, mas ainda assim preserva a diversidade populacional, permitindo que indivíduos menos adaptados também tenham chances de participar da reprodução.

3.3. Random Forest + ED

Essa implementação com ED foi feita utilizando o método differential_evolution da biblioteca scipy.optimize do python, que implementa o algoritmo da Evolução Diferencial. A função fitness foi implementada da mesma forma como foi no GA.

Na proxima seção é apresentado os resultados dos experimentos.

4. Resultados e Discussão

Esta seção apresenta os resultados dos experimentos feitos. O objetivo dos experimentos é avaliar o desempenho das duas metaheurísticas, com diferentes espaços de busca.

4.1. Resultados de Random Forest + GA

Para realizar os experimentos com o GA, foram definidas as configurações apresentadas na tabela abaixo.

Table 2. Parâmetros utilizados no algoritmo genético (GA)

Parâmetro	Definição	Valor
`cxpb`	Probabilidade de cruzamento entre indivíduos.	0,5
`mutpb`	Probabilidade de mutação dos genes.	0,3
`ngen`	Número total de gerações.	8

Inicialmente foi feito um experimento com o seguinte espaço de busca:

Table 3. Hiperparâmetros selecionados e seus respectivos espaços de busca do 1º experimento com GA

Hiperparâmetro	Espaço de Busca
`n_estimators`	[50, 150]
`max_depth`	[5, 40]
`min_samples_split`	[2, 20]

Os resultados são apresentados na tabela abaixo.

Table 4. Desempenho do algoritmo genético na otimização da Random Forest no 1º experimento.

Métrica	Valor
Tempo de Execução (s)	484,28
Memória Usada (MB)	8,18
Melhor Solução	[109, 13, 10]
Fitness da Solução	0,7042
Acurácia (treino)	0,9576
Acurácia (teste)	0,7983
F1-score (teste)	0,7375
ROC AUC (teste)	0,8472

Espaço de Busca	`n_estimators: (50, 150),`
	`max_depth: (5, 40),`
	`min_samples_split: (2, 20)`

Na tabela seguinte está o relatório de classificação com os hiperparâmetros ótimos.

Table 5. Relatório de classificação do 1º experimento com GA

Classe	Precision	Recall	F1-score	Support
0	0.78	0.90	0.84	2400
1	0.83	0.66	0.74	1800
Accuracy		0.80		4200
Macro avg	0.81	0.78	0.79	4200
Weighted avg	0.80	0.80	0.79	4200

Foi feito outro experimento com outro espaço de busca, reduzindo o valores do intervalo da profundidade da árvore. O resultado é apresentado nas tabelas abaixo.

Table 6. Desempenho do algoritmo genético na otimização da Random Forest no 2º experimento.

Métrica	Valor
Tempo de Execução (s)	388,01
Memória Usada (MB)	5,90
Melhor Solução	[89, 15, 5]
Fitness da Solução	0,7046
Acurácia (treino)	0,9945
Acurácia (teste)	0,8062
F1-score (teste)	0,7447
ROC AUC (teste)	0,8526
Espaço de Busca	`n_estimators: (50, 150), max_depth: (2, 15), min_samples_split: (2, 20)`

Table 7. Relatório de classificação do 2º experimento com GA

Classe	Precision	Recall	F1-score	Support
0	0.78	0.92	0.84	2400
1	0.86	0.66	0.74	1800
Accuracy	0.81			4200
Macro avg	0.82	0.79	0.79	4200
Weighted avg	0.81	0.81	0.80	4200

Observa-se que o tempo de execução foi reduzido no segundo experimento, assim como o consumo de memória. Algumas métricas também melhoraram no segundo experimento, como a acurácia e a ROC AUC, assim como também a precisão da classe 1 e o recall da classe 0.

4.2. Resultados de Random Forest + ED

Nos experimentos com o ED foi feita a seguinte configuração de parâmetros:

Table 8. Parâmetros utilizados na Evolução Diferencial (ED)

Parâmetro	Definição	Valor
`strategy`	Estratégia de mutação utilizada.	`best1bin`
`mutation`	Fator de mutação diferencial F, intervalo de variação.	(0,5, 1)
`recombination`	Taxa de recombinação (crossover).	0,7
`seed`	Semente para reprodutibilidade.	42
`tol`	Tolerância de convergência.	0,01
`maxiter`	Número máximo de iterações (gerações).	20
`popsize`	Fator multiplicativo do tamanho da população.	10

Foram feitos 2 experimentos também com os mesmos espaços de busca utilizados nos experimentos com GA.

O primeiro experimento feito com o primeiro espaço de busca utilizado no GA, obteve os resultados apresentados na tabela abaixo.

Table 9. Desempenho do algoritmo de Evolução Diferencial na otimização da Random Forest no 1º experimento.

Métrica	Valor
Tempo de Execução (s)	80,52
Memória Usada (MB)	6,35
Melhor Solução	[87, 13, 4]
Fitness da Solução	0,7106
Acurácia (treino)	0,9834
Acurácia (teste)	0,8040
F1-score (teste)	0,7443
ROC AUC (teste)	0,8489

Espaço de Busca	`n_estimators: (50, 150),`
	`max_depth: (5, 40),`
	`min_samples_split: (2, 20)`

Na próxima tabela está o relatório de classificacão com os hiperparâmetros ótimos.

(Olhar lá o relatório no visual studio)

Table 10. Relatório de classificação do 1º experimento com ED

Classe	Precision	Recall	F1-score	Support
0	0,7736	0,9267	0,8432	2400
1	0,8672	0,6383	0,7354	1800
Accuracy	0,8031			4200
Macro avg	0,8204	0,7825	0,7893	4200
Weighted avg	0,8137	0,8031	0,7970	4200

Novamente, foi feito um 2º experimento com o segundo espaço de busca.

Table 11. Desempenho do algoritmo de Evolução Diferencial na otimização da Random Forest no 2º experimento.

Métrica	Valor
Tempo de Execução (s)	154,54
Memória Usada (MB)	6,18
Melhor Solução	[98, 13, 4]
Fitness da Solução	0,7132
Acurácia (treino)	0,9835
Acurácia (teste)	0,8055
F1-score (teste)	0,7457
ROC AUC (teste)	0,8498
Espaço de Busca	`n_estimators: (50, 150), max_depth: (2, 15), min_samples_split: (2, 20)`

Abaixo está o relatório de classificação desse 2º experimento.

Table 12. Relatório de classificação do 2º experimento com ED

Classe	Precision	Recall	F1-score	Support
0	0,7746	0,9267	0,8439	2400
1	0,8676	0,6406	0,7370	1800
Accuracy		0,8040		4200
Macro avg	0,8211	0,7836	0,7904	4200
Weighted avg	0,8145	0,8040	0,7981	4200

Foi observado que o ED consegue obter praticamente os mesmos resultados que o GA em um tempo bem menor. Então, foi feito um 3º experimento com um espaço de busca com um maior número de árvores de decisão. O resultado é apresentado nas tabelas abaixo.

Table 13. Desempenho do algoritmo de Evolução Diferencial na otimização da Random Forest no experimento com $n\_estimators \in (150, 200)$.

Métrica	Valor
Tempo de Execução (s)	249,31
Memória Usada (MB)	-43,38
Melhor Solução	[188, 13, 2]
Fitness da Solução	0,7114
Acurácia (treino)	0,9899
Acurácia (teste)	0,8107
F1-score (teste)	0,7521
ROC AUC (teste)	0,8558
Espaço de Busca	`n_estimators: (150, 200), max_depth: (2, 15), min_samples_split: (2, 20)`

Table 14. Relatório de classificação do 3º experimento com Evolução Diferencial

Classe	Precision	Recall	F1-score	Support
0	0.7767	0.9350	0.8486	2400
1	0.8810	0.6417	0.7425	1800
Accuracy		0.8093		4200
Macro avg	0.8289	0.7883	0.7955	4200
Weighted avg	0.8214	0.8093	0.8031	4200

5. Conclusões

O estudo feito nesse artigo concluiu que o algoritmo ED possui um dempenho melhor que o GA, conseguindo praticamente os mesmos valores de acurácia, precisão, revocação e f1-score do GA, e reduzindo consideravelmente o tempo de execução. Concluiu-se também que árvores menos profundas melhoram o desempenho da classificação, enquanto árvores mais profundas podem aumentar o risco de overfitting, perdendo assim a capacidade de generalização. Em todos os experimentos, os melhores valores do hiperparâmetro max_depth foi 13 ou 15. Foi observado também que o ED consegue avaliar florestas aleatórias com um número grande de árvores de decisão em um tempo razoável em comparação com o GA. Mesmo com um esbaço de busca com n_estimators = (150, 200) e max_depth = (2, 15), ele conseguiu um tempo de 249,31 segundos. Já o GA, com um espaço de busca com n_estimators = (50, 150), e max_depth = (2, 15) demorou 388,01 segundos. O ED também consegue avaliar árvores de decisão profundas em tempos excelentes, em comparação com o GA. Com n_estimators = (50, 150) e max_depth = (5, 40), o ED executou em 80,52 segundos. O GA, com esse mesmo espaço de busca, executou em 484,28 segundos.

6. Referências

REDDY, K. V. V.; et al. Heart disease risk prediction using machine learning classifiers with attribute evaluators. Applied Sciences, v. 11, n. 18, p. 8352, 2021.

NASUTION, N.; HASAN, M. A.; NASUTION, F. B. Predicting Heart Disease Using Machine Learning: An Evaluation of Logistic Regression, Random Forest, SVM, and KNN Models on the UCI Heart Disease Dataset. IT Journal Research and Development, v. 9, n. 2, p. 140-150, 2025.

SEN, J.; BARMAN, P. C. A hybrid machine learning approach for heart disease prediction: Integrating Random Forest and Multi-Layer Perceptron for enhanced diagnostic accuracy. International Journal of Engineering Research & Technology (IJERT), v. 14, n. 5, maio 2025. Disponível em: http://www.ijert.org. ISSN 2278-0181.

RUSSELL, S. J.; NORVIG, P. Inteligência Artificial. 3. ed. São Paulo: Elsevier, 2013.

MAIMON, O. Z.; ROKACH, L. Data Mining with Decision Trees: Theory and Applications. 2. ed. Singapore: World Scientific Publishing, 2014.

BREIMAN, L. Random Forests. Machine Learning, v. 45, n. 1, p. 5–32, 2001.

GUEX, N. Random Forests with R. Springer, 2021.

HOLLAND, J. H. Adaptation in Natural and Artificial Systems. Cambridge: MIT Press, 1992.

STORN, R.; PRICE, K. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization, v. 11, n. 4, p. 341–359, 1997.

EIBEN, A. E.; SMITH, J. E. Introduction to Evolutionary Computing. 2. ed. Springer, 2015.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321–357.