CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS DIVINÓPOLIS
CURSO DE ENGENHARIA DE COMPUTAÇÃO

DISCIPLINA: INTELIGÊNCIA ARTIFICIAL

Estudo Comparativo de Paradigmas de IA

Sistemas Simbólicos, Aprendizado de Máquina e Computação Natural

Autores:
Bruno Prado Dos Santos
João Francisco Teles da Silva

Professor:
Tiago Alves de Oliveira

Divinópolis – MG
Dezembro de 2025

Resumo

Este trabalho apresenta a implementação e análise comparativa de quatro paradigmas fundamentais da Inteligência Artificial, abordando desde sistemas baseados em regras até metaheurísticas bioinspiradas. Inicialmente, desenvolveu-se um sistema especialista simbólico utilizando uma Árvore de Decisão manual para recomendação de filmes. Em seguida, aplicaram-se algoritmos de Aprendizado de Máquina Supervisionado (KNN, SVM e Árvore de Decisão) no problema de classificação de doenças cardíacas (*Cleveland Heart Disease*), avaliando métricas como acurácia, precisão e *F1-score*. A terceira etapa explorou a Computação Evolutiva através de um Algoritmo Genético (GA) aplicado ao Problema do Caixeiro Viajante (TSP) com 20 cidades. Por fim, realizou-se um estudo comparativo entre Inteligência de Enxame (ACO) e Sistemas Imunológicos Artificiais (CLONALG) no mesmo cenário de otimização. Os resultados experimentais demonstram os *trade-offs* entre a explicabilidade dos modelos simbólicos, a capacidade de generalização dos estatísticos e a eficiência de convergência das abordagens naturais.

Sumário

1 Introdução

A Inteligência Artificial (IA) evoluiu historicamente de sistemas baseados em conhecimento explícito e regras rígidas para modelos capazes de aprender padrões complexos a partir de dados e algoritmos estocásticos inspirados na natureza. Segundo Russell e Norvig (2013), a IA busca projetar agentes que percebem seu ambiente e tomam ações que maximizam suas chances de sucesso. Compreender as nuances, vantagens e limitações de cada paradigma é essencial para a seleção da técnica adequada a cada classe de problema.

Este relatório documenta o desenvolvimento, a implementação e a análise crítica de quatro abordagens distintas de IA:

1. **Sistemas Simbólicos (Parte 1):** Uma abordagem determinística baseada em regras de decisão manuais ("se-então"), exemplificada por um sistema de recomendação.
2. **Aprendizado de Máquina (Parte 2):** Modelos estatísticos supervisionados aplicados à classificação de dados médicos reais, comparando o desempenho de vizinhança (KNN), margem máxima (SVM) e entropia (Árvores de Decisão).
3. **Computação Evolutiva (Parte 3):** O uso de mecanismos de seleção natural e variação genética simulada para resolver problemas de otimização combinatória (TSP).
4. **Inteligência de Enxame e Imunes (Parte 4):** Metaheurísticas baseadas em comportamento social coletivo e mecanismos de defesa biológica, contrastando suas capacidades de exploração e convergência.

O objetivo central deste trabalho não é apenas a obtenção de soluções funcionais, mas a avaliação da eficiência computacional e da qualidade das soluções geradas por cada método frente a problemas de diferentes naturezas.

2 Parte 1: Sistemas Simbólicos (Árvore de Decisão Manual)

Esta etapa do trabalho explora a implementação de um Sistema Especialista baseado em regras, um dos paradigmas clássicos da Inteligência Artificial Simbólica. O objetivo foi desenvolver um sistema de recomendação de filmes que, através de uma interação guiada, classifica a preferência do usuário em um gênero cinematográfico específico.

2.1 Fundamentação Teórica

2.1.1 Sistemas Simbólicos e Baseados em Conhecimento

A IA Simbólica, predominante nas décadas de 1970 e 1980, baseia-se na premissa de que a inteligência pode ser simulada através da manipulação de símbolos e regras lógicas explícitas. Diferente das abordagens conexionistas (como redes neurais), que aprendem padrões a partir de dados, os sistemas simbólicos operam sobre um conhecimento previamente estruturado e inserido por um especialista humano.

Um exemplo clássico são os **Sistemas Especialistas**, que utilizam um conjunto de regras "Se-Então"(*If-Then*) para inferir conclusões a partir de fatos observados. A principal vantagem desta abordagem é a **explicabilidade**: o caminho lógico percorrido pelo sistema para chegar a uma conclusão é transparente e facilmente auditável.

2.1.2 Árvores de Decisão Estáticas

No contexto deste trabalho, a base de conhecimento foi modelada como uma Árvore de Decisão estática. Uma árvore de decisão é um grafo acíclico dirigido onde:

- Cada **nó interno** representa um teste ou pergunta sobre um atributo (ex: "Gosta de adrenalina?");
- Cada **aresta** (ramo) representa o resultado desse teste (ex: "Sim"ou "Não");
- Cada **nó folha** representa uma classe ou decisão final (ex: Gênero "Terror").

Nesta implementação manual ("hardcoded"), a estrutura da árvore e os pontos de corte são definidos a priori pelo programador, sem o uso de algoritmos de indução como ID3 ou C4.5.

2.2 Modelagem do Problema

O domínio do problema foi restringido à recomendação de gêneros cinematográficos. Para garantir a granularidade exigida de 10 perguntas, a árvore foi projetada com profundidade variável, dividindo o espaço de busca em dois eixos macroscópicos de preferência do usuário:

1. **Eixo da Adrenalina/Tensão:** Destinado a usuários que buscam estímulos intensos. Este ramo discrimina entre o medo (Terror), a ação física (Ação), o mistério intelectual (Suspense) e a aventura exploratória (Ficção/Aventura).
2. **Eixo da Calma/Emoção:** Destinado a usuários que buscam entretenimento leve ou reflexivo. Este ramo diferencia o humor (Comédia), a catarse emocional (Drama), o romance (subdividido em leve ou dramático) e o escapismo fantástico (Fantasia).

2.3 Implementação

O sistema foi implementado em Python, utilizando uma estrutura de controle de fluxo aninhada para representar os nós da árvore. A função de interação `ask(prompt)` garante que a entrada do usuário seja normalizada, aceitando variações como "s", "sim", "n"ou "nao".

O Algoritmo 1 ilustra a lógica de decisão implementada, demonstrando como o fluxo de execução percorre a árvore até atingir uma folha.

Algorithm 1: Sistema Especialista de Recomendação de Filmes

Entrada: Respostas do usuário (Sim/Não)
Saída: Gênero de filme recomendado

```
1  if Gosta de Adrenalina/Tensão? then
2      if Objetivo é sentir Medo? then
3          return Terror;
4      else
5          if Gosta de Explosões/Perseguições? then
6              return Ação;
7          else
8              if Prefere resolver Mistérios? then
9                  return Suspense/Policial;
10             else
11                 if Envolve Futuro/Espaço? then
12                     return Ficção Científica;
13                 else
14                     return Aventura;
15 else
16     if Objetivo é dar Risada? then
17         return Comédia;
18     else
19         if Quer se Emocionar/Chorar? then
20             return Drama;
21         else
22             if Foco em Romance? then
23                 if Prefere final Feliz/Leve? then
24                     return Comédia Romântica;
25                 else
26                     return Romance Dramático;
27             else
28                 if Gosta de Magia/Mundos Irreais? then
29                     return Fantasia;
30                 else
31                     return Documentário/Biografia;
```

2.4 Resultados e Discussão

A validação do sistema foi realizada através de testes de caixa-branca, percorrendo todos os caminhos possíveis da árvore para garantir que cada folha fosse alcançável. O diagrama visual da estrutura de decisão (Figura 1) permite compreender a cobertura dos gêneros.

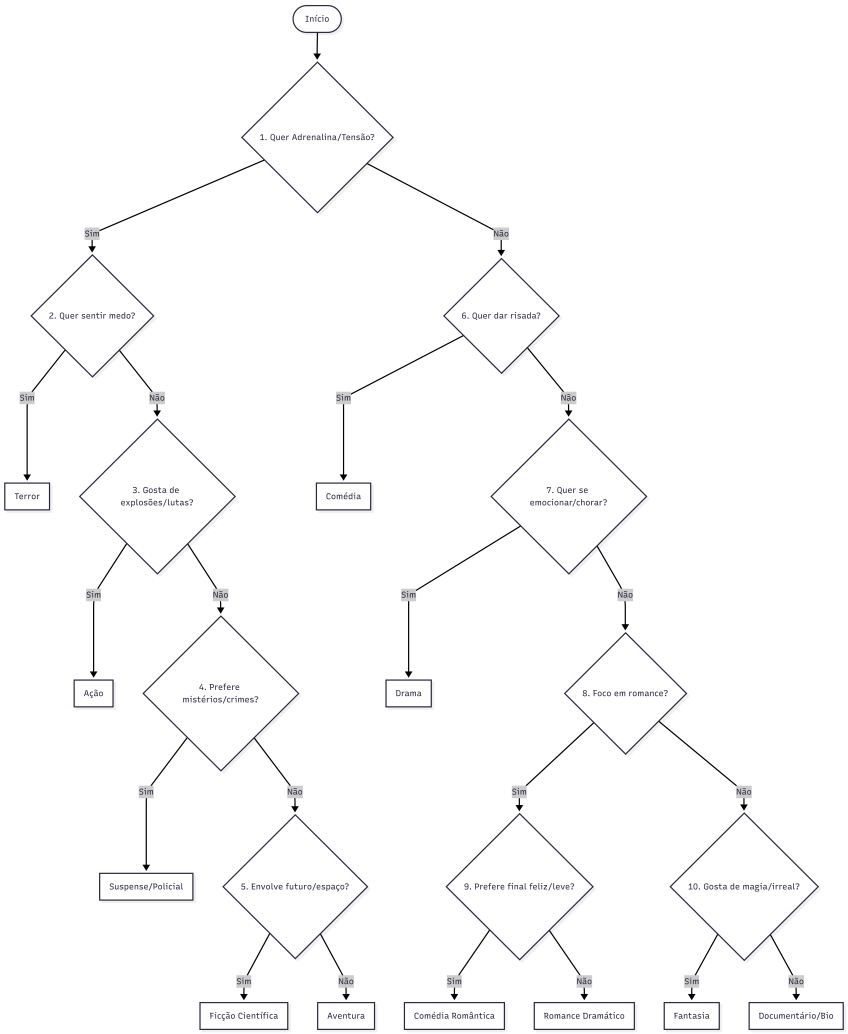

Figura 1: Diagrama da Árvore de Decisão Manual gerado via Mermaid.

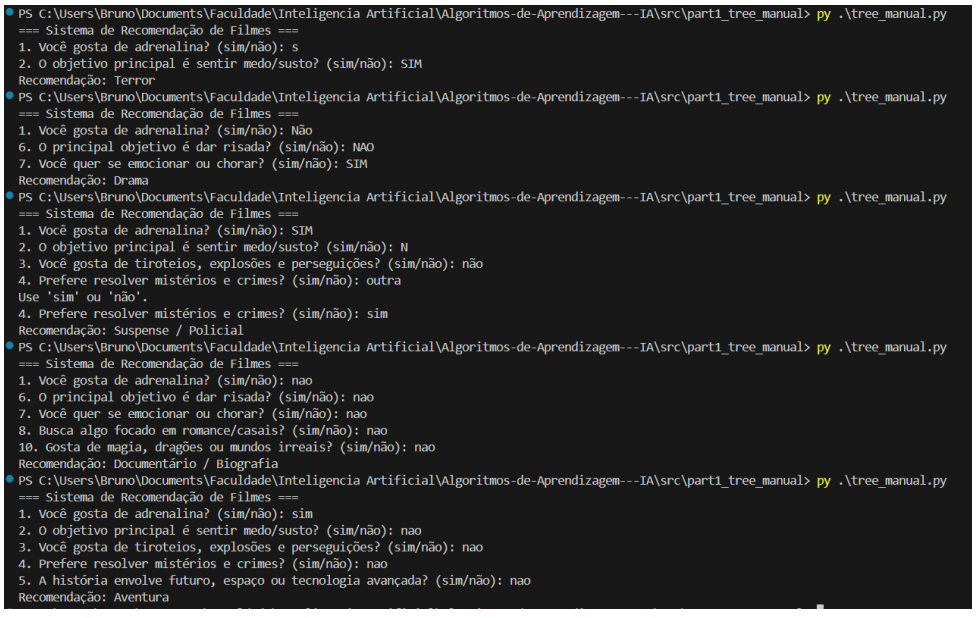

Figura 2: Imagem de teste inserida para validação de layout.

A principal vantagem observada nesta abordagem é a **precisão determinística**: para um conjunto específico de entradas, a saída é sempre previsível e justificável. No entanto, a limitação da escalabilidade é evidente; adicionar novos gêneros ou refinar as recomendações exigiria a reescrita manual do código ("hardcoding"), tornando a manutenção custosa em sistemas complexos. Essa limitação motiva o uso de Aprendizado de Máquina, abordado na próxima seção.

3 KNN, SVM Árvore de Decisão

3.1 Fundamentação Teórica

3.1.1 KNN

O K-Nearest Neighbors(KNN) é um algoritmo supervisionado baseado em instâncias, utilizado principalmente para classificação. A ideia central é simples: para classificar uma nova amostra, o algoritmo calcula a distância dessa amostra para todos os exemplos do conjunto de treino e seleciona os k vizinhos mais próximos. Em seguida, a classe predita é definida por votação (maioria) entre esses vizinhos.

Em geral, utiliza-se a distância Euclidiana quando os atributos são numéricos. Por isso, é fundamental aplicar normalização/escala (ex.: *StandardScaler*), pois atributos com escalas maiores podem dominar o cálculo das distâncias e distorcer o resultado.

O hiperparâmetro k controla o compromisso entre sensibilidade e generalização: valores pequenos de k tendem a produzir fronteiras de decisão mais complexas e sensíveis a ruídos, enquanto valores maiores de k suavizam a fronteira, tornando o modelo mais estável, porém podendo reduzir a capacidade de capturar padrões locais. Assim, a escolha de k costuma ser feita via validação cruzada.

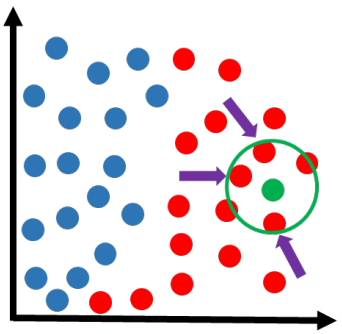

Figura 3: Intuição do KNN: classificação por votação entre os k vizinhos mais próximos.

A Figura 3 ilustra a intuição do KNN: a amostra de teste (ponto verde) é classificada observando os k vizinhos mais próximos dentro de uma vizinhança (círculo), que "votam" na classe final.

3.1.2 SVM

O *Support Vector Machine* (SVM) é um algoritmo supervisionado de aprendizado utilizado para classificação e regressão. Seu objetivo principal é encontrar um hiperplano que separe

as classes de forma ótima, maximizando a margem entre os exemplos de diferentes classes mais próximos ao limite de decisão, chamados de vetores de suporte.

Quando os dados são linearmente separáveis, o SVM busca um hiperplano linear. Entretanto, em muitos problemas reais, essa separação não é possível no espaço original. Para lidar com esse cenário, o SVM utiliza funções de kernel, que realizam uma transformação implícita dos dados para um espaço de maior dimensão, tornando possível a separação linear nesse novo espaço.

Neste trabalho, foram utilizados os kernels linear e radial (*Radial Basis Function* – RBF). O parâmetro C controla o compromisso entre maximizar a margem e penalizar erros de classificação, enquanto o parâmetro γ influencia a flexibilidade da fronteira de decisão no kernel RBF. Valores elevados de γ geram fronteiras mais complexas, enquanto valores menores produzem superfícies de decisão mais suaves.

Assim como no KNN, a normalização dos atributos é essencial no SVM, pois o algoritmo é sensível à escala dos dados. Por esse motivo, foi aplicado o método *StandardScaler* antes do treinamento do modelo.

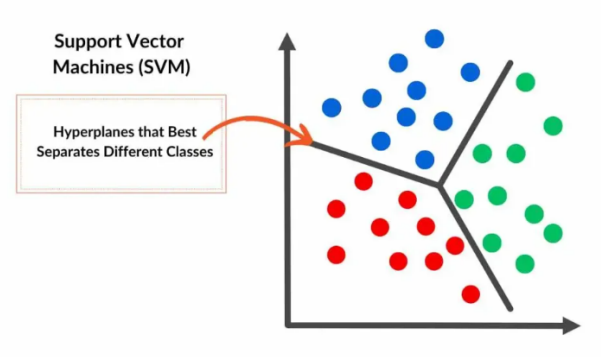

Figura 4: SVM: hiperplano que maximiza a margem entre classes.

A Figura 4 ilustra o princípio de funcionamento do SVM, evidenciando diferentes hiperplanos capazes de separar as classes e destacando aquele que maximiza a margem entre elas.

3.1.3 Árvore de Decisão

A Árvore de Decisão é um algoritmo de aprendizado supervisionado amplamente utilizado em problemas de classificação e regressão. Seu funcionamento baseia-se na construção de uma estrutura hierárquica de decisões, na qual cada nó interno representa um teste sobre um atributo, cada ramo corresponde ao resultado desse teste e cada folha indica a classe final atribuída à amostra.

Durante o processo de treinamento, o algoritmo seleciona automaticamente os atributos e limiares que melhor separam as classes, buscando maximizar a pureza dos subconjuntos gerados a cada divisão. Neste trabalho, foi utilizado o critério de impureza de Gini, que mede o grau de heterogeneidade das classes em um nó da árvore.

O índice de Gini é definido pela Equação 1, sendo utilizado para avaliar a qualidade de uma divisão. Quanto menor o valor do índice, maior a homogeneidade das classes no nó analisado.

$$Gini = 1 - \sum_{i=1}^{C} p_i^2 \quad (1)$$

onde C representa o número de classes e p_i corresponde à proporção de amostras da classe i no nó. Em problemas de classificação binária, como o considerado neste trabalho, a expressão se reduz a $Gini = 1 - (p_0^2 + p_1^2)$.

A construção da árvore ocorre de forma recursiva até que critérios de parada sejam satisfeitos. Para evitar o sobreajuste aos dados de treino, foi utilizado o parâmetro *max_ depth*, que limita a profundidade máxima da árvore. O valor desse parâmetro foi definido por meio de validação cruzada, buscando um equilíbrio entre capacidade de generalização e complexidade do modelo.

3.2 Modelagem do problema

3.2.1 Descrição do problema e tipo de tarefa

O problema abordado nesta etapa do trabalho consiste na classificação automática de pacientes quanto à presença ou ausência de doença cardíaca, com base em informações clínicas e demográficas. A partir de um conjunto de atributos extraídos de exames e observações médicas, busca-se prever se um paciente apresenta indícios da doença.

Os dados utilizados contêm variáveis como idade, sexo, pressão arterial em repouso, nível de colesterol, frequência cardíaca máxima alcançada, entre outros fatores clínicos relevantes. A variável alvo indica o diagnóstico do paciente, originalmente representado por diferentes níveis de severidade da doença.

Para os experimentos realizados, o problema foi formulado como uma tarefa de classificação binária. Dessa forma, pacientes sem diagnóstico de doença cardíaca foram associados à classe 0, enquanto pacientes com qualquer nível de diagnóstico positivo foram associados à classe 1.

Assim, trata-se de um problema de aprendizado supervisionado, no qual os modelos são treinados a partir de exemplos previamente rotulados, com o objetivo de aprender padrões que permitam classificar corretamente novos pacientes.

3.2.2 Base de dados

Os experimentos realizados neste trabalho utilizaram o conjunto de dados *Cleveland Heart Disease Dataset*, disponibilizado no repositório UCI Machine Learning Repository. Esse dataset é amplamente utilizado como referência em estudos de aprendizado de máquina na área da saúde.

O conjunto é composto por registros de pacientes descritos por atributos clínicos e demográficos, incluindo variáveis numéricas e categóricas. Os dados encontram-se organizados em formato tabular, no qual cada linha representa um paciente e cada coluna corresponde a um atributo específico.

A Tabela 1 apresenta uma descrição resumida dos principais atributos originais presentes no conjunto de dados, antes da aplicação das etapas de pré-processamento.

Tabela 1: Descrição dos principais atributos do dataset Cleveland Heart Disease

Atributo	**Descrição**	**Tipo**
age	Idade do paciente	Numérico
sex	Sexo do paciente	Categórico
cp	Tipo de dor no peito	Categórico
trestbps	Pressão arterial em repouso	Numérico
chol	Colesterol sérico	Numérico
fbs	Glicemia em jejum	Binário
restecg	Resultado do eletrocardiograma	Categórico
thalach	Frequência cardíaca máxima	Numérico
exang	Angina induzida por exercício	Binário
oldpeak	Depressão do segmento ST	Numérico
slope	Inclinação do segmento ST	Categórico
ca	Número de vasos principais	Categórico
thal	Resultado do exame de talassemia	Categórico
target	Diagnóstico de doença cardíaca	Binário

3.2.3 Pré-processamento dos dados

Antes do treinamento dos modelos de aprendizado supervisionado, os dados passaram por uma etapa de pré-processamento com o objetivo de garantir consistência e adequação às técnicas utilizadas.

Inicialmente, valores ausentes presentes no conjunto de dados, identificados por caracteres especiais, foram tratados por meio da remoção das amostras incompletas. Em seguida, todos os atributos foram convertidos para o tipo numérico, garantindo compatibilidade com os algoritmos de aprendizado.

A variável alvo, originalmente representada por diferentes níveis de severidade da doença cardíaca, foi binarizada. Dessa forma, amostras sem diagnóstico de doença foram associadas à classe 0, enquanto amostras com qualquer nível positivo de diagnóstico foram associadas à classe 1.

As variáveis categóricas foram transformadas utilizando codificação *one-hot*, permitindo sua representação em formato numérico sem introduzir relações ordinais artificiais entre as categorias.

Por fim, o conjunto de dados foi dividido em conjuntos de treino e teste por meio de divisão estratificada, utilizando 70% das amostras para treinamento e 30% para teste. Essa estratégia garante a preservação da proporção entre as classes em ambos os conjuntos, contribuindo para uma avaliação mais confiável do desempenho dos modelos.

3.2.4 Implementação e ajuste dos classificadores

KNN: O classificador K-Nearest Neighbors foi implementado utilizando a biblioteca scikit-learn. Como o algoritmo baseia-se no cálculo de distâncias entre amostras, os atributos foram previamente normalizados por meio do método *StandardScaler*.

A escolha do parâmetro k foi realizada por meio de validação cruzada estratificada com 5 partições, avaliando valores de k no intervalo de 1 a 30. Para cada valor testado, foi calculada a acurácia média obtida na validação cruzada.

Conforme apresentado na Figura 5, o valor $k = 25$ apresentou a maior acurácia média, sendo, portanto, selecionado para o treinamento final do modelo KNN.

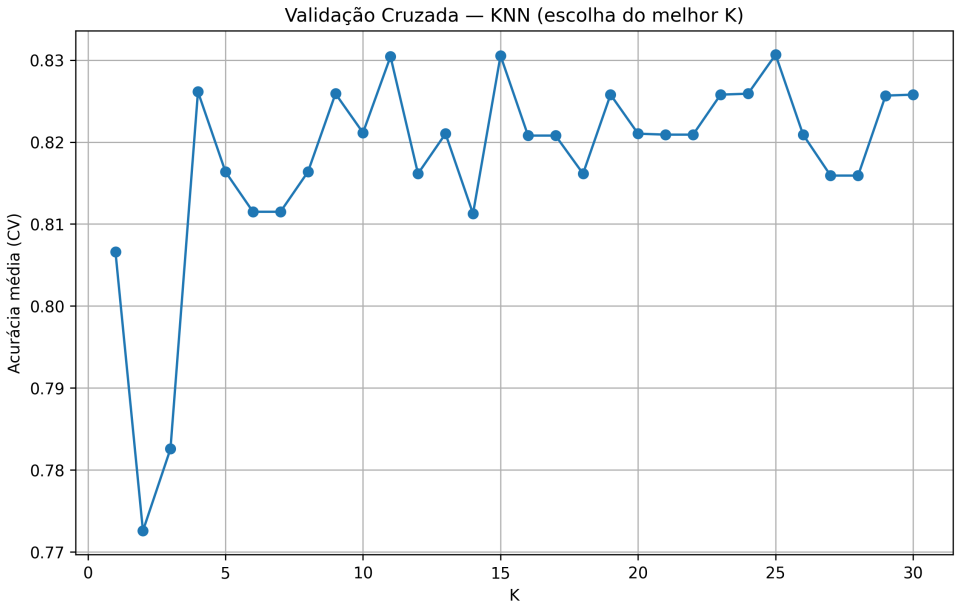

Figura 5: Validação cruzada para escolha do melhor valor de k no classificador KNN

SVM: O classificador Support Vector Machine foi implementado utilizando a biblioteca scikit-learn, considerando os kernels linear e radial (*RBF*). Assim como no KNN, os atributos foram normalizados por meio do método *StandardScaler*, devido à sensibilidade do algoritmo à escala dos dados.

A seleção dos hiperparâmetros foi realizada por meio de validação cruzada estratificada, avaliando diferentes combinações dos parâmetros C, γ e do tipo de kernel. A melhor combinação encontrada foi o kernel RBF com $C = 0{,}1$ e $\gamma = \text{scale}$, que apresentou a maior acurácia média na validação cruzada.

Essa configuração foi então utilizada para o treinamento final do modelo SVM.

Árvore de Decisão: A Árvore de Decisão foi implementada utilizando o critério de impureza de Gini para orientar as divisões dos nós. Com o objetivo de controlar a complexidade do modelo e evitar sobreajuste, foi realizada uma validação cruzada variando o parâmetro de profundidade máxima da árvore.

Foram avaliados valores de profundidade no intervalo de 1 a 15. O valor $max\_depth = 9$ apresentou a maior acurácia média na validação cruzada e foi selecionado para o treinamento final do classificador.

Para fins de interpretação, a árvore treinada foi visualizada graficamente, sendo exibidos apenas os primeiros níveis da estrutura, conforme ilustrado na Figura 6.

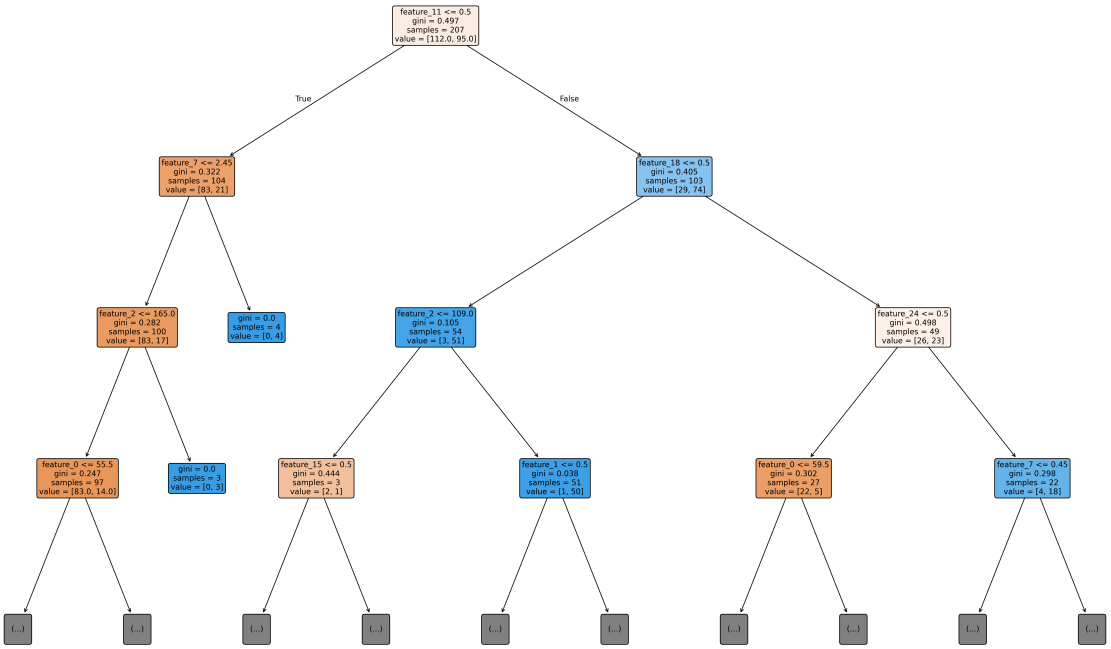

Figura 6: Visualização parcial da Árvore de Decisão treinada (profundidade limitada para fins de interpretação)

3.3 Resultados

Os classificadores KNN, SVM e Árvore de Decisão foram avaliados no conjunto de teste utilizando as métricas de acurácia, precisão, recall e F1-score. Essas métricas permitem analisar tanto o desempenho global dos modelos quanto seu comportamento em relação à classificação correta das classes.

A Tabela 2 apresenta os valores obtidos por cada classificador. Observa-se que o modelo KNN apresentou os maiores valores em todas as métricas avaliadas, seguido pelo SVM, enquanto a Árvore de Decisão apresentou desempenho inferior.

Tabela 2: Métricas de desempenho dos classificadores no conjunto de teste

Modelo	**Acurácia**	**Precisão**	**Recall**	**F1-score**
KNN	0.87	0.87	0.86	0.87
SVM	0.84	0.85	0.84	0.84
Árvore de Decisão	0.72	0.73	0.71	0.71

A Figura 7 apresenta uma visualização gráfica da comparação entre os classificadores, facilitando a análise do desempenho relativo de cada modelo em relação às métricas consideradas.

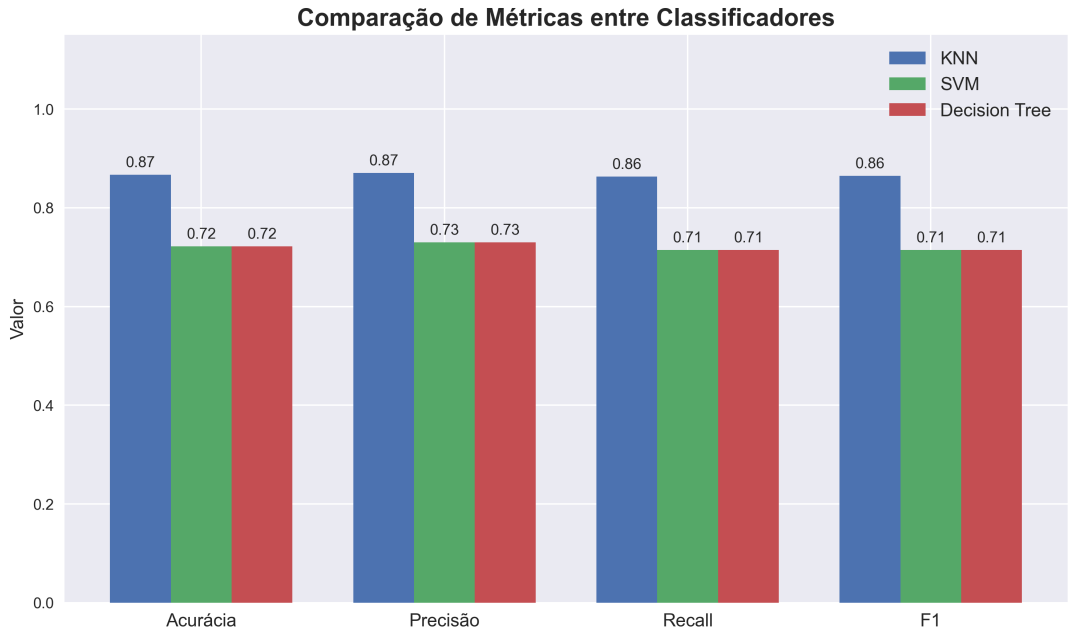

Figura 7: Comparação das métricas de desempenho entre os classificadores

A análise dos resultados evidencia que o classificador KNN obteve o melhor desempenho geral, alcançando valores superiores de acurácia, precisão, recall e F1-score. Esse resultado indica que o modelo foi capaz de capturar de forma eficaz os padrões presentes nos dados após o pré-processamento e a normalização dos atributos.

O classificador SVM apresentou desempenho consistente, embora ligeiramente inferior ao KNN. Os valores obtidos sugerem que o modelo construiu uma fronteira de decisão adequada, porém com menor flexibilidade para lidar com a distribuição dos dados quando comparado ao método baseado em vizinhos mais próximos.

Por sua vez, a Árvore de Decisão apresentou desempenho inferior em todas as métricas avaliadas. Apesar de sua interpretabilidade e simplicidade, o modelo demonstrou menor capacidade de generalização, o que pode estar relacionado à complexidade do conjunto de dados e às limitações impostas pela profundidade máxima da árvore utilizada.

De forma geral, os resultados indicam que o método KNN foi o mais adequado para o problema de classificação de doença cardíaca abordado neste trabalho, superando tanto o SVM quanto a Árvore de Decisão no conjunto de teste.

4 Parte 3: Computação Evolutiva (Algoritmo Genético)

Nesta etapa, aplicou-se a Computação Evolutiva para resolver um problema de otimização combinatória clássico: o Problema do Caixeiro Viajante (TSP). O objetivo foi demonstrar como mecanismos inspirados na seleção natural podem ser utilizados para encontrar soluções sub-ótimas de alta qualidade em espaços de busca vastos, onde métodos exatos seriam inviáveis.

4.1 Fundamentação Teórica

4.1.1 Algoritmos Genéticos (AG)

Os Algoritmos Genéticos são uma classe de algoritmos de busca estocástica baseados na teoria da evolução de Charles Darwin. Eles operam sobre uma população de soluções candidatas (indivíduos), que evoluem ao longo de gerações através de processos análogos à biologia:

- **Seleção Natural:** Indivíduos mais aptos (*fitness* superior) têm maior probabilidade de sobreviver e reproduzir.
- **Crossover (Recombinação):** Combinação do material genético de dois pais para gerar descendentes que herdam características de ambos.
- **Mutação:** Alterações aleatórias nos genes para introduzir diversidade e evitar a convergência prematura para ótimos locais.

Essa abordagem é particularmente eficaz em problemas NP-difíceis, onde o espaço de busca é grande demais para ser varrido sistematicamente.

4.2 Definição e Modelagem do Problema

O problema consiste em encontrar a rota de menor distância que visita um conjunto de N cidades exatamente uma vez e retorna à cidade de origem.

Configuração do Experimento:

- **Instância:** 20 cidades distribuídas aleatoriamente em um plano cartesiano 100×100.
- **Espaço de Busca:** O número total de rotas possíveis é dado por $(N-1)!/2$. Para $N = 20$, isso resulta em aproximadamente 6×10^{16} permutações distintas, inviabilizando a força bruta.
- **Otimização:** Para reduzir o custo computacional, foi pré-calculada uma *Matriz de Distâncias* 20×20, evitando o recálculo repetitivo da distância Euclidiana durante a avaliação de fitness.

4.3 Implementação do Algoritmo Genético

O algoritmo foi implementado "do zero"em Python, sem o uso de bibliotecas de "caixa preta"como *DEAP*, para evidenciar a compreensão dos operadores internos.

4.3.1 Representação e Fitness

Cada indivíduo (cromossomo) é representado por uma lista de inteiros $[0, 1, ..., 19]$ que denota a ordem de visitação das cidades (permutação). A função de *fitness* foi definida como o negativo da distância total, uma vez que o algoritmo busca maximizar o fitness, mas o objetivo do TSP é minimizar o caminho:

$$Fitness(ind) = -\sum_{i=0}^{N-1} dist(cidade_i, cidade_{i+1})$$

4.3.2 Operadores Genéticos

Para garantir a validade das rotas (evitar cidades duplicadas ou faltantes), utilizaram-se operadores específicos para permutações:

1. **Seleção por Torneio:** Escolhe-se aleatoriamente um subconjunto da população e o melhor indivíduo deste grupo é selecionado para reprodução. Isso mantém a pressão seletiva controlável.
2. **Crossover OX1 (Order Crossover):** Preserva uma sub-rota de um pai e preenche o restante com a ordem relativa das cidades do segundo pai. Isso transmite características estruturais (sequências de cidades) para a prole.
3. **Mutação Swap:** Com uma probabilidade baixa (10%), duas cidades aleatórias da rota trocam de posição. Isso introduz diversidade e evita a estagnação em ótimos locais.

O Algoritmo 2 resume a estrutura evolutiva implementada.

Algorithm 2: Algoritmo Genético para TSP

Entrada: População Inicial, Taxas (Crossover, Mutação), Gerações
Saída: Melhor Rota Encontrada

Inicializar população aleatória P;
Avaliar fitness de cada indivíduo em P;
for $g \leftarrow 1$ **to** Max_Geraes **do**
    $Nova\_Pop \leftarrow \emptyset$;
    while $|Nova\_Pop| < |P|$ **do**
        $Pai1 \leftarrow$ Torneio(P); $Pai2 \leftarrow$ Torneio(P);
        if *rand()* $<$ $Taxa\_Crossover$ **then**
            $Filho1, Filho2 \leftarrow$ OX1_Crossover($Pai1, Pai2$);
        else
            $Filho1, Filho2 \leftarrow Pai1, Pai2$;
        Mutar($Filho1$); Mutar($Filho2$);
        Adicionar $Filhos$ em $Nova\_Pop$;
    $P \leftarrow Nova\_Pop$;
    Avaliar e Registrar estatísticas (Melhor, Média);
return Melhor indivíduo histórico;

4.4 Resultados e Análise Experimental

Para garantir a robustez estatística dos resultados, o algoritmo foi executado **30 vezes independentes** com sementes aleatórias distintas. Os parâmetros utilizados foram: População = 100, Gerações = 2000, Taxa de Crossover = 0.9 e Taxa de Mutação = 0.1.

4.4.1 Convergência e Estabilidade

A Figura 8 apresenta a curva de convergência média das 30 execuções. O eixo X representa as gerações e o eixo Y a distância total (menor é melhor).

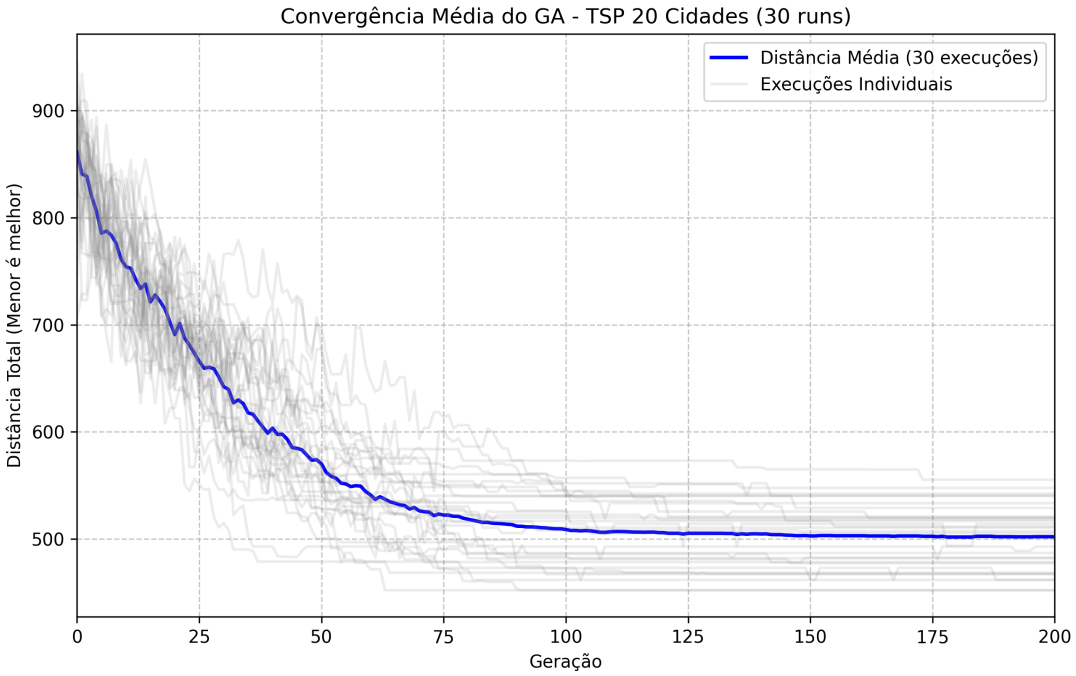

Figura 8: Curva de Convergência do GA: Evolução da distância ao longo das gerações (Número de gerações foi condensado para 200 apenas para melhor vizualização).

Observa-se um decaimento acentuado da distância nas primeiras 50 gerações ("fase de exploração"), onde o algoritmo refina rapidamente as soluções aleatórias iniciais. Após a geração 150, a curva tende à estabilização ("fase de exploração fina"), indicando que a população convergiu para uma região de ótimo (global ou local).

4.4.2 Análise de Desempenho

A Tabela 3 resume os dados estatísticos coletados. O baixo desvio padrão em relação à média final indica que o algoritmo é robusto, encontrando soluções de qualidade consistente independentemente da população inicial aleatória.

Tabela 3: Estatísticas de Desempenho do GA (30 Execuções)

Métrica	**Valor Obtido**
Melhor Distância Encontrada (Mínimo)	**452.09**
Pior Distância Encontrada (Máximo)	540.52
Distância Média Final	488.26
Desvio Padrão	27.60
Tempo Médio de Execução (s)	1.8923 s

Discussão: Em comparação com a força bruta, que levaria anos para testar as 2.4×10^{18} combinações possíveis, o GA encontrou uma solução competente em menos de 2 segundos (tempo médio). Embora não se possa garantir matematicamente que a melhor solução encontrada é o ótimo global absoluto, a consistência dos resultados sugere que ela está muito próxima dele, validando a eficácia da abordagem evolutiva para problemas NP-difíceis.

5 Parte 4: Inteligência de Enxame e Sistemas Imunes

Nesta etapa final, exploram-se duas abordagens de Inteligência Computacional inspiradas em sistemas biológicos descentralizados: a Inteligência de Enxame (*Swarm Intelligence*) e os Sistemas Imunológicos Artificiais (*Artificial Immune Systems*). O objetivo é comparar o desempenho do algoritmo ACO (*Ant Colony Optimization*) e do CLONALG (*Clonal Selection Algorithm*) no mesmo problema de otimização (TSP 20 cidades) abordado anteriormente.

5.1 Fundamentação Teórica

5.1.1 Otimização por Colônia de Formigas (ACO)

O ACO é uma metaheurística inspirada no comportamento de formigas reais na busca por alimentos. Formigas depositam uma substância química chamada **feromônio** no solo enquanto caminham. Outras formigas tendem a seguir caminhos com maior concentração de feromônio, reforçando essa trilha (feedback positivo). Com o tempo, o feromônio evapora (feedback negativo), o que evita a convergência prematura para caminhos subótimos. A probabilidade de uma formiga k escolher ir da cidade i para j depende de dois fatores:

- **Trilha de Feromônio (τ_{ij}):** Memória coletiva do enxame (o que funcionou no passado).
- **Visibilidade/Heurística (η_{ij}):** Informação gulosa local (geralmente $1/distncia_{ij}$), indicando que cidades mais próximas são mais atrativas.

5.1.2 Algoritmo de Seleção Clonal (CLONALG)

O CLONALG é inspirado na resposta imune adquirida, especificamente na teoria da seleção clonal. Quando o organismo é invadido por um antígeno (o problema a ser resolvido), as células B (anticorpos/soluções) que melhor reconhecem esse antígeno são selecionadas para proliferar (clonagem). Os princípios-chave são:

- **Seleção e Clonagem:** Os melhores indivíduos geram mais cópias de si mesmos.
- **Maturação de Afinidade (Hipermutação):** Os clones sofrem mutações com taxas inversamente proporcionais à sua aptidão (indivíduos muito bons mudam pouco; indivíduos ruins mudam muito).
- **Edição de Receptores:** Os piores indivíduos da população são substituídos por novos aleatórios, garantindo diversidade contínua.

5.2 Implementação

Ambos os algoritmos foram aplicados à mesma instância do TSP (20 cidades) utilizada no Algoritmo Genético, permitindo uma comparação direta.

5.2.1 Parametrização do ACO

Implementou-se o *Ant System* clássico com os seguintes parâmetros:

- **Formigas:** 30 agentes construindo soluções a cada iteração.
- **Pesos:** $\alpha = 1.0$ (importância do feromônio) e $\beta = 2.0$ (importância da distância). Isso dá prioridade à heurística local (cidades próximas).
- **Evaporação (ρ):** Taxa de 0.1 (10% do feromônio evapora por ciclo).

5.2.2 Parametrização do CLONALG

Adaptou-se o algoritmo para problemas de permutação:

- **População:** 50 anticorpos (rotas).
- **Fator de Clonagem (β):** 5 (gera clones proporcionalmente ao rank).
- **Mutação:** Operador *Swap* com taxa dinâmica. Os melhores clones sofrem apenas 1 troca (refinamento fino), enquanto clones de pior rank sofrem múltiplas trocas (exploração agressiva).
- **Diversidade:** 20% da população é renovada a cada geração.

5.3 Resultados e Discussão Comparativa

Os algoritmos foram executados por 1000 iterações em 30 rodadas independentes. A Figura 9 apresenta o comparativo das curvas de convergência média.

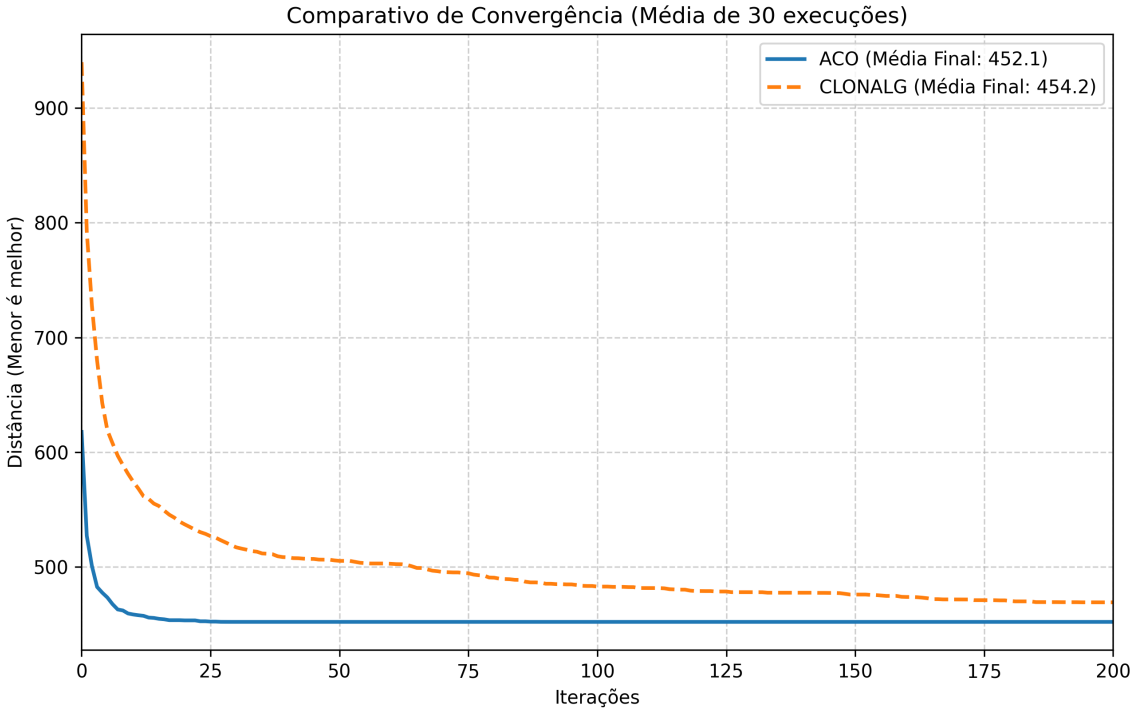

Figura 9: Comparativo de Convergência: ACO vs CLONALG (Média de 30 execuções) (Número de iterações foi condensado para 200 apenas para melhor vizualização).

5.3.1 Análise de Comportamento

Observa-se comportamentos distintos de busca:

1. **ACO (Enxame):** Apresenta uma convergência inicial extremamente rápida. A combinação da heurística gulosa ($\beta = 2$) com o reforço de feromônio permite encontrar rotas "boas"quase instantaneamente. No entanto, tende a estabilizar mais cedo, exigindo ajuste fino na taxa de evaporação para não estagnar.

2. **CLONALG (Imunes):** Demonstra uma evolução mais gradual. A mecânica de hipermutação permite que o algoritmo continue refinando soluções por mais tempo, e a inserção de indivíduos aleatórios (edição de receptores) previne a perda total de diversidade, permitindo ocasionais "saltos"para fora de ótimos locais.

5.3.2 Tabela Resumo

A Tabela 4 consolida o desempenho final das metaheurísticas em comparação ao Algoritmo Genético (Parte 3).

Tabela 4: Comparativo Final: GA vs ACO vs CLONALG

Algoritmo	Melhor Solução	Média	Tempo (s)	Característica
Genético (GA)	**452.09**	**488.26**	**1.8923**	Equilíbrio Global
Enxame (ACO)	**452.09**	**452.09**	**5.1371**	Convergência Rápida
Imunes (CLONALG)	**452.09**	**454.16**	**7.1804**	Refinamento Local

Conclusão da Etapa: Para instâncias pequenas do TSP (20 cidades), o ACO mostrou-se ligeiramente superior em velocidade de convergência inicial, enquanto o CLONALG e o GA demonstraram maior robustez para manter a diversidade genética a longo prazo.

6 Conclusão e Trabalhos Futuros

Este trabalho apresentou um estudo comparativo entre diferentes paradigmas da Inteligência Artificial, abrangendo sistemas simbólicos, aprendizado de máquina supervisionado e técnicas bioinspiradas de otimização. Ao longo das etapas desenvolvidas, foi possível analisar não apenas o desempenho numérico dos algoritmos, mas também suas características conceituais, limitações e adequação a diferentes tipos de problemas.

Na Parte 1, a abordagem simbólica baseada em uma Árvore de Decisão manual evidenciou a principal vantagem dos sistemas baseados em regras: a interpretabilidade. O sistema especialista desenvolvido produziu decisões determinísticas e facilmente explicáveis, porém apresentou limitações claras de escalabilidade e adaptabilidade, uma vez que a expansão do domínio exige modificações manuais na base de conhecimento.

Na Parte 2, foram aplicados algoritmos de Aprendizado de Máquina supervisionado ao problema real de classificação de doenças cardíacas. Os resultados experimentais mostraram que o classificador KNN obteve o melhor desempenho geral, superando o SVM e a Árvore de Decisão nas métricas de acurácia, precisão, recall e F1-score. O SVM

apresentou desempenho consistente, enquanto a Árvore de Decisão, embora interpretável, demonstrou menor capacidade de generalização. Esses resultados reforçam a importância do pré-processamento adequado e da seleção criteriosa de hiperparâmetros no desempenho de modelos estatísticos.

Na Parte 3, a Computação Evolutiva foi explorada por meio da aplicação de um Algoritmo Genético ao Problema do Caixeiro Viajante. Os experimentos evidenciaram a capacidade dos algoritmos evolutivos de encontrar soluções de alta qualidade em espaços de busca extremamente grandes, onde métodos exatos se tornam inviáveis. A análise de convergência demonstrou um equilíbrio entre exploração e refinamento, com resultados robustos ao longo de múltiplas execuções.

Por fim, na Parte 4, foram comparadas abordagens de Inteligência de Enxame e Sistemas Imunológicos Artificiais no mesmo problema de otimização. Observou-se que o ACO apresentou convergência inicial mais rápida, enquanto o CLONALG demonstrou maior capacidade de refinamento contínuo e manutenção da diversidade. Essa comparação evidenciou que diferentes metaheurísticas podem ser mais ou menos adequadas dependendo dos objetivos do problema, como velocidade de convergência ou robustez da solução.

De forma geral, os resultados obtidos ao longo do trabalho destacam que não existe um único paradigma de Inteligência Artificial superior em todos os cenários. A escolha da abordagem mais adequada depende diretamente da natureza do problema, dos requisitos de interpretabilidade, da disponibilidade de dados e das restrições computacionais envolvidas.

Como trabalhos futuros, sugere-se a ampliação dos experimentos com técnicas adicionais de aprendizado de máquina, como Redes Neurais Artificiais e métodos de ensemble, bem como a aplicação de técnicas de balanceamento de classes e validação cruzada mais extensiva no problema de classificação médica. Além disso, para os problemas de otimização, pode-se explorar instâncias maiores do TSP, bem como híbridos entre metaheurísticas, combinando, por exemplo, algoritmos genéticos com métodos de enxame ou sistemas imunes. Essas extensões permitiriam uma análise ainda mais abrangente do comportamento e das limitações dos diferentes paradigmas de Inteligência Artificial.

7 Reprodutibilidade

Todos os experimentos realizados neste trabalho podem ser reproduzidos a partir do código-fonte disponibilizado em um repositório público. O projeto foi desenvolvido em Python (versão 3.10 ou superior) e utiliza bibliotecas amplamente consolidadas, como NumPy, Pandas, Matplotlib e Scikit-learn, cujas dependências estão listadas no arquivo `requirements.txt`.

A execução dos experimentos é feita por meio de scripts independentes, organizados de acordo com cada parte do trabalho. Para a etapa de aprendizado supervisionado, os dados são inicialmente pré-processados e, em seguida, os classificadores são treinados e avaliados conforme descrito no repositório.

Quando aplicável, sementes aleatórias foram utilizadas para garantir a reprodutibilidade dos resultados. As instruções completas de execução e configuração do ambiente estão documentadas no arquivo `README.md` do projeto.

Referências

[1] RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 4. ed. Upper Saddle River: Pearson, 2021.

[2] UCI MACHINE LEARNING REPOSITORY. *Heart Disease Dataset*. Disponível em: `https://archive.ics.uci.edu/ml/datasets/Heart+Disease`. Acesso em: 21 dez. 2025.

[3] SANTOS, B. P.; SILVA, J. F. T. *Algoritmos de Aprendizagem e Otimização – IA*. 2025. Disponível em: `https://github.com/joaofranciscoteles/Algoritmos-de-Aprendizagem---IA`. Acesso em: 21 dez. 2025.