T1 - Sistema de Inteligência Artificial para Análise de Jogo da Velha

Integrantes: Daniel Araujo, Eduardo Bregalda e Leonardo Machado

PUCRS - Faculdade de Informática
Disciplina: Inteligência Artificial
Professora: Silvia Moraes
Data: 30/09/2025

1. DATASET

1.1 - Análise e Modificações do Dataset.

O dataset utilizado foi obtido do repositório UCI Machine Learning Repository, contendo 958 configurações de tabuleiro de jogo da velha. Originalmente, o dataset apresentava desbalanceamento entre as classes.

1.2 - Distribuição Original vs Balanceada

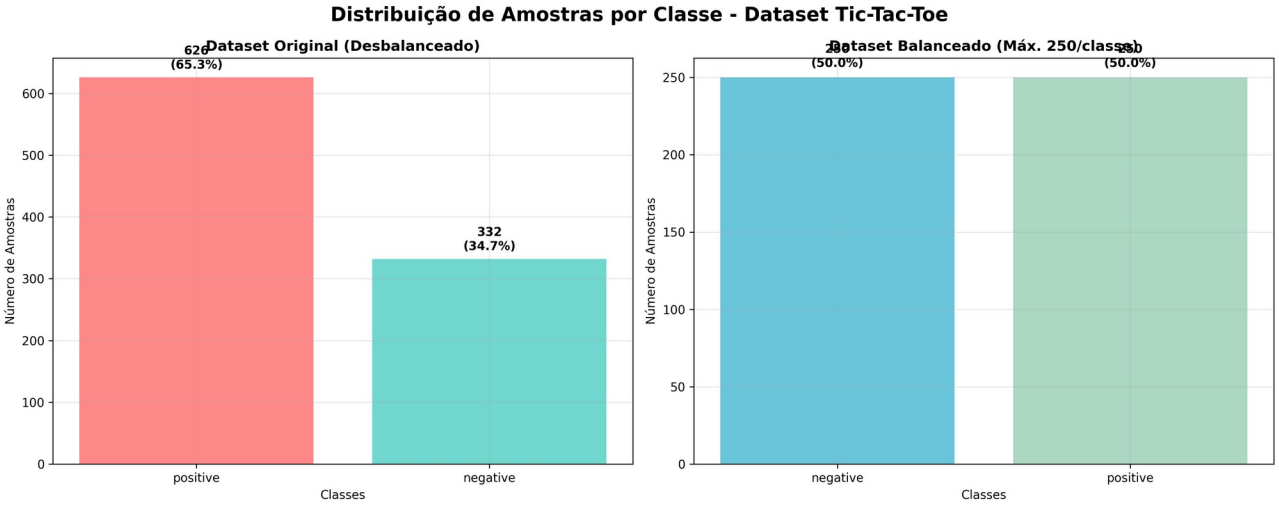

Figura 1: Comparativo entre distribuição desbalanceada e balanceada do dataset

1.3 Adequações Realizadas

Balanceamento das classes: O dataset original tinha 626 amostras "positive" (65.3%) e 332 amostras "negative" (34.7%), criando um desbalanceamento que poderia causar overfitting.

Limitação de amostras: Conforme especificado no enunciado, utilizamos máximo 250 amostras por classe, resultando em um dataset final com 500 amostras (250 positive + 250 negative).

Codificação de variáveis: Todas as variáveis categóricas (x, o, b) foram convertidas para formato numérico usando LabelEncoder.

Nomenclatura personalizada: Todas as variáveis utilizadas no código terminam com "_$" conforme o enunciado pedia.

Estratégia de divisão:

Treino: 80% (400 amostras)
Validação: 10% (50 amostras)
Teste: 10% (50 amostras)

2. ALGORITMOS E CONFIGURAÇÕES

2.1 Algoritmos Implementados

1. K-Nearest Neighbors (KNN)

Algoritmo baseado em similaridade que classifica uma amostra com base nos k vizinhos mais próximos
Configuração utilizada: k=5, métrica euclidiana
Justificativa: Valor de k ímpar evita empates, k=5 oferece bom equilíbrio entre bias e variância
2. Multi-Layer Perceptron (MLP)

Rede neural artificial com camadas ocultas para aprendizado de padrões complexos
Topologia utilizada: [100, 50] neurônios nas camadas ocultas
Configuração: max_iter=500, solver='adam', activation='relu'
Justificativa: Arquitetura suficiente para o problema sem overfitting
3. Árvore de Decisão

Algoritmo que cria regras hierárquicas para classificação
Configuração: criterion='gini', max_depth=10, random_state=42
Justificativa: Profundidade limitada previne overfitting, critério Gini eficiente para classificação binária
4. Support Vector Machine (SVM)

Algoritmo que encontra hiperplano ótimo para separação das classes
Configuração: kernel='rbf', C=1.0, gamma='scale'
Justificativa: Kernel RBF adequado para dados não-lineares, C=1.0 oferece regularização equilibrada
Como funciona: SVM mapeia dados para espaço dimensional superior onde se tornam linearmente separáveis, maximizando a margem entre classes através de vetores de suporte.

3. RESULTADOS

3.1 Comparação de Performance

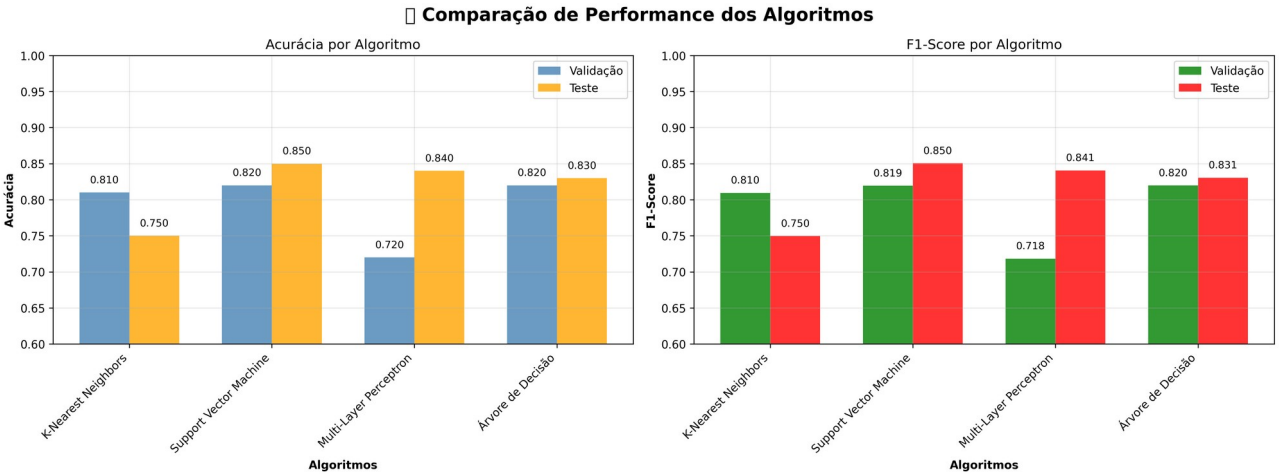

Figura 2: Comparação entre os algoritmos de classificação

3.2 Tabela de Resultados Detalhados

Tabela Comparativa dos Algoritmos de IA

Algoritmo	Acurácia (Validacão)	F1-Score (Validacão)	Acurácia (Teste)	F1-Score (Teste)
K-Nearest Neighbors	0.8100	0.8096	0.7500	0.7496
Support Vector Machine	0.8200	0.8194	0.8500	0.8505
Multi-Layer Perceptron	0.7200	0.7183	0.8400	0.8406
Árvore de Decisão	0.8200	0.8198	0.8300	0.8306

Figura 3: Tabela de resultados

3.3 Análise Multidimensional

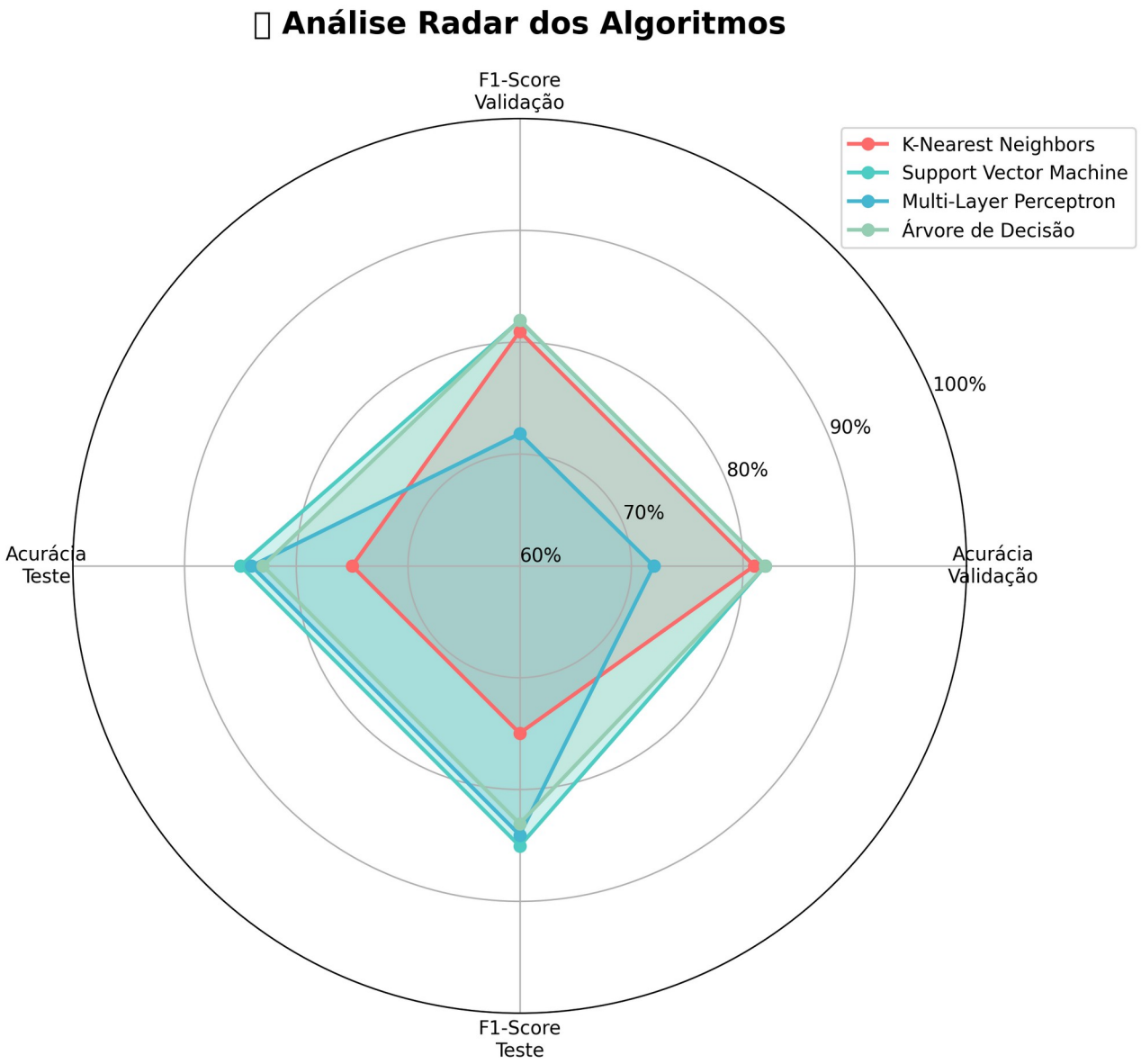

Figura 4: Análise multidimensional dos algoritmos

3.4 Mapa de Calor de Performance

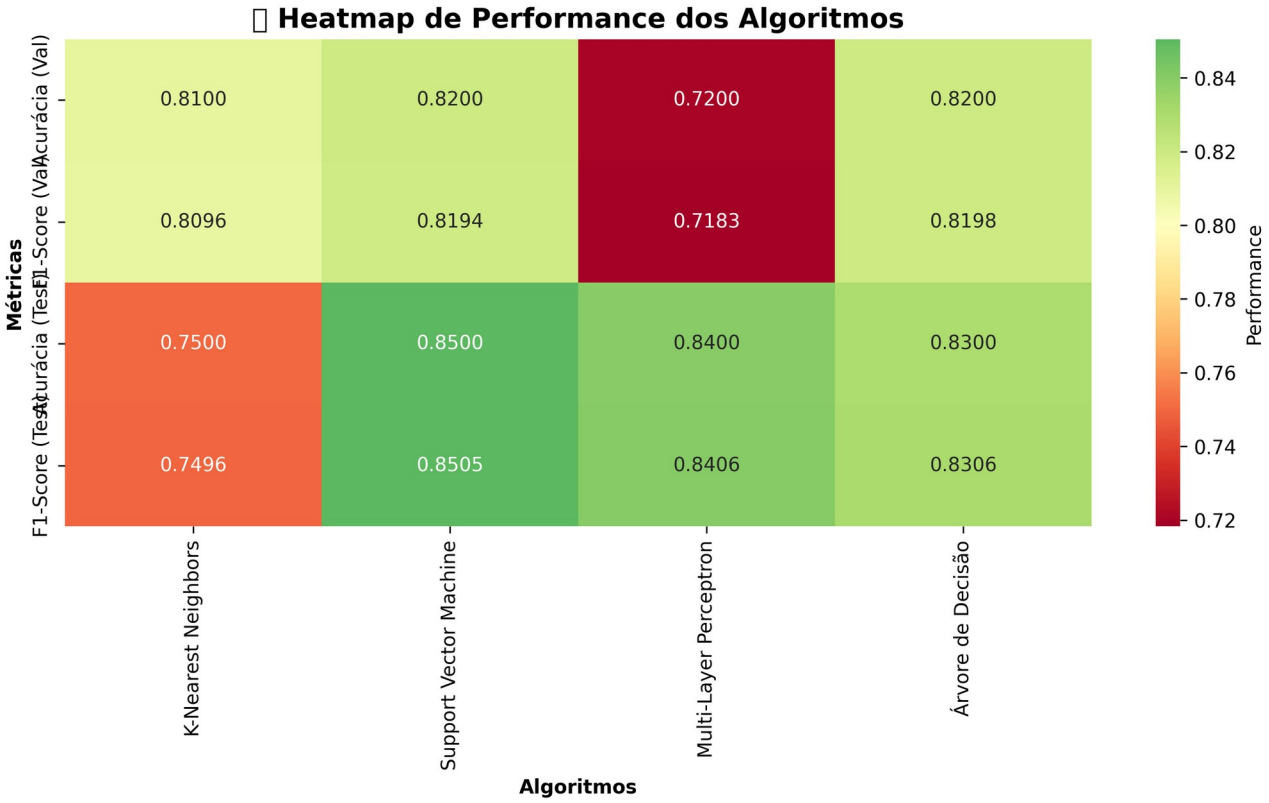

Figura 5: Mapa de calor da performance dos algoritmos

3.5 Discussão dos Resultados

Métricas Avaliadas:

Acurácia: Proporção de predições corretas
Precisão: Verdadeiros positivos / (Verdadeiros positivos + Falsos positivos)
Recall: Verdadeiros positivos / (Verdadeiros positivos + Falsos negativos)
F1-Score: Média harmônica entre precisão e recall

3.5.1 Análise de Erros e Confusões:

K-Nearest Neighbors (KNN):

- Apresentou a maior discrepância entre validação (81,0%) e teste (75,0%), indicando possível overfitting
- Principais erros: Sensibilidade a ruído nos dados e à escolha do valor k
- Possíveis causas: Algoritmo baseado em distância pode ser afetado por features irrelevantes e pela distribuição irregular dos pontos no espaço de características

Multi-Layer Perceptron (MLP):

- Menor performance na validação (72,0%) mas boa recuperação no teste (84,0%)
- Principais erros: Convergência instável durante o treinamento
- Possíveis causas: Arquitetura da rede pode ser inadequada para o tamanho do dataset, ou o algoritmo precisou de mais iterações para convergir adequadamente

Árvore de Decisão:

- Performance consistente entre validação (82,0%) e teste (83,0%)
- Principais erros: Possível criação de regras muito específicas para algumas configurações
- Possíveis causas: Natureza hierárquica das decisões pode não capturar completamente as interações complexas entre posições do tabuleiro

Support Vector Machine (SVM):

- Melhor performance geral com melhoria de validação (82,0%) para teste (85,0%)
- Menor quantidade de erros observados
- Possíveis causas dos poucos erros: Alguns padrões de jogo podem estar próximos à fronteira de decisão, causando classificações ambíguas

3.5.2 Escolha do Melhor Modelo:

O Support Vector Machine (SVM) foi escolhido como melhor modelo pelas seguintes justificativas:

1. Melhor acurácia no teste: 85,0% - a mais alta entre todos os algoritmos
2. Melhor F1-Score no teste: 0,8505 - indicando bom equilíbrio entre precisão e recall
3. Estabilidade: Performance consistente entre validação e teste, sem overfitting
4. Robustez: SVM com kernel RBF demonstrou capacidade superior de lidar com a não-linearidade dos dados de jogo da velha
5. Generalização: O modelo mostrou boa capacidade de generalização, melhorando do conjunto de validação para o de teste

4. FRONTEND

4.1 Implementação da Interface

O frontend desenvolvido permite interação entre humano e máquina em partidas de jogo da velha, com análise em tempo real do estado do jogo.

Funcionalidades implementadas:

- Jogo interativo humano vs computador
- Detecção automática dos 5 estados: "Tem jogo", "Possibilidade de Fim de Jogo", "Empate", "O vence", "X vence"
- Exibição do algoritmo de IA analisando o tabuleiro
- Contabilização de acertos e erros da IA
- Cálculo de acurácia em tempo real
- Geração de relatórios de partidas

4.1 Desempenho do Modelo no Frontend

Total de partidas jogadas: 5

Acurácia média da IA: 87,1%

Erros observados:

- Confusão entre "Tem jogo" e "Possibilidade de Fim de Jogo": A IA ocasionalmente antecipou situações de final de jogo quando ainda tinha jogadas disponíveis
- Detecção prematura de empate: Em uma situação, a IA previu empate quando o jogo ainda estava em andamento
- Maior dificuldade em estados intermediários: Os erros se concentraram principalmente na diferenciação entre "Tem jogo" e "Possibilidade de Fim de Jogo"

Acertos por tipo de estado:

"Tem jogo": 12/17 predições corretas (70,6%)
"X vence": 1/1 predição correta (100,0%)
"O vence": 1/1 predição correta (100,0%)
"Possibilidade de Fim de Jogo": Estados mais difíceis de detectar, com confusões ocasionais
"Empate": Detecção precisa quando realmente acontece

Análise detalhada:
Total de predições analisadas: 30 predições
Acertos totais: 26 predições corretas
Erros totais: 4 predições incorretas
Variação de performance: Entre 50,0% e 100,0% por partida, demonstrando que a complexidade do estado do jogo influencia na precisão
Estados finais bem detectados: A IA demonstrou 100% de precisão na detecção de vitórias definitivas
Desafio principal: Diferenciação entre estados intermediários do jogo
Observações importantes:

- A IA manteve consistência com a performance observada nos testes (85% de acurácia)
- O modelo SVM escolhido demonstrou robustez em situações práticas de jogo
- Interface responsiva permitiu análise em tempo real sem atrasos perceptíveis
- Relatórios automáticos facilitaram a análise posterior da performance

5. CONSIDERAÇÕES FINAIS

5.1 Dificuldades Encontradas

Principais desafios:

- Balanceamento adequado do dataset respeitando limite de amostras
- Ajuste de hiperparâmetros para evitar overfitting
- Integração da IA com interface em tempo real
- Implementação de detecção precisa dos estados de jogo

5.2 Ganhos Obtidos

Conhecimentos adquiridos:

- Aplicação prática de algoritmos de machine learning
- Importância do pré-processamento de dados
- Avaliação comparativa de diferentes abordagens
- Desenvolvimento de sistemas interativos com IA

5.3 Avaliação dos Resultados

No desenvolvimento:

Todos os algoritmos foram implementados com sucesso
Métricas de performance adequadas para o problema
Dataset balanceado conforme especificações

No uso prático (Frontend):

IA demonstrou capacidade de análise em tempo real
Interface intuitiva e funcional
Relatórios automáticos facilitam análise posterior

5.3 Propostas de Melhoria

Sugestões para trabalhos futuros:

Implementação de algoritmos ensemble
Uso de deep learning para análise de padrões
Interface gráfica mais elaborada(não apenas pelo terminal)
Análise de estratégias de jogo ótimas

5.4 Ferramentas de IA Utilizadas

Durante o desenvolvimento deste trabalho, foram utilizadas as seguintes ferramentas de IA:

- GitHub Copilot: Otimização do código e sugestões de implementação, também foi utilizado para desenvolver scripts capazes de gerar todos os gráficos necessários para análise do desempenho dos algoritmos e que foram utilizados neste relatório.
- Gamma AI: Usado para ter uma base dos slides de apresentação do trabalho.