# T1 - Inteligencia Artificial

Guilherme Melos Vieira

## Introdução:

O objetivo deste projeto foi desenvolver um sistema de IA para o jogo da velha, capaz de verificar o estado de um tabuleiro e classificar o jogo em quatro possíveis saídas: “continue” (jogo ainda em andamento), “positive” (jogador X venceu), “negative” (jogador O venceu) e “tie” (empate). Além disso, o sistema também deveria interagir com o jogador por meio de um front-end simples, permitindo que um humano jogue contra a máquina (que faz jogadas aleatórias).

O problema consiste em treinar um modelo de classificação que, com base no estado atual do tabuleiro (9 entradas), determine uma das quatro saídas possíveis. O desafio principal era garantir que o modelo pudesse generalizar bem para diferentes estados de jogo, evitando overfitting e ao mesmo tempo garantindo alta precisão.

## Dataset:

O dataset originalmente tinha 958 casos, de fim de jogo, com 10 colunas, 9 para as casas do jogo da velha, onde ‘b’ eram os espaços vazios e ‘x’ e ‘o’ representavam os ‘x’ e ‘O’ que tinham sido jogados e 1 coluna para a classificação, “positive” para o jogador ‘x’ ser vencedor e “negative” para o caso contrário, que poderia ser empate ou vitória do jogador ‘O’.

As mudanças feitas ao dataset foram criar duas novas classes, a classe “tie” para empate e a classe “continue” para jogo ainda em andamento, também tive que balancear o dataset para fazer com que tivesse 200 amostras para cada classe, tive que replicar os casos da classe tie pois só existem 16 casos possíveis de empate no jogo da velha.

Dentro do código todas as entradas foram convertidas para valores numéricos sendo:

b = 0
o = 1
x = 2
continue = 0
negative = 1
positive = 2
tie = 3

## Desenvolvimento das Soluções de IA:

Foram testados quatro algoritmos de classificação para a solução do problema:

- **K-Nearest Neighbors (KNN)**: Utilizou-se k=3 para a classificação.
- **Multi-Layer Perceptron (MLP)**: Uma rede neural com 15 neurônios na camada oculta, tendo uma topologia de 9 x 15 x 4.
- **Árvore de Decisão**: Profundidade máxima de 15 para a árvore de decisão.
- **Random Forest**: Floresta aleatória com 70 árvores e profundidade máxima de 30.

Os valores de hiperparâmetros foram ajustados com base em iterações anteriores e avaliação de desempenho. Foi dada ênfase em evitar overfitting nos modelos mais complexos, como Random Forest e MLP.

Foi utilizada uma divisão de treino/validação/teste com 40% dos dados reservados para validação e teste, garantindo uma avaliação justa dos modelos.

## Resultados:

| Algoritmo | Acurácia | Precisão | Recall | F1-Score |
|---|---|---|---|---|
| **KNN** | 0.7125 | 0.7091 | 0.7125 | 0.6875 |
| **MLP** | 0.7625 | 0.7617 | 0.7625 | 0.7617 |
| **Árvore de Decisão** | 0.8125 | 0.8100 | 0.8125 | 0.8101 |
| **Random Forest** | 0.8375 | 0.8386 | 0.8375 | 0.8349 |

**Valores aproximados**

- **K-Nearest Neighbors (KNN)**: O desempenho foi razoável, porém inferior aos modelos mais complexos, como Random Forest e MLP. A acurácia de 71% reflete limitações em capturar padrões complexos no jogo da velha.
- **Multi-Layer Perceptron (MLP)**: A rede neural obteve bons resultados, com uma acurácia de 76%, mostrando sua capacidade de capturar interações não lineares.
- **Árvore de Decisão:** A árvore de decisão teve uma performance superior (81% de acurácia), balanceando simplicidade e precisão.
- **Random Forest:** Este foi o algoritmo com melhor desempenho, com uma acurácia de 83,75%, sugerindo que a combinação de múltiplas árvores melhora a robustez e a generalização do modelo.

## Conclusão:

‘ Este projeto permitiu o melhor entendimento sobre como configurar e usar alguns algoritmos de aprendizado de máquina, no fim o não achei que ficou muito preciso, a IA erra diversas vezes a previsão. A principal dificuldade foi gerar dados artificiais realistas para a classe "continue", além de ajustar os hiperparâmetros dos modelos para evitar overfitting.

O Random Forest se destacou como o melhor modelo, mas o MLP e a Árvore de Decisão também mostraram bons resultados embora o MLP tenha sido o modelo mais demorado na hora de treiná-lo.