Trabalho 1 - Tic Tac Toe com ML

Inteligência Artificial

Natália Dal Pizzol e Sofia Sartori

4 de setembro de 2024

Resumo

1 Introdução

O Jogo da Velha é um dos jogos mais simples e populares ao redor do mundo. Sua simplicidade, aliada à sua capacidade de representar um desafio estratégico, o torna um objeto de estudo interessante para diversas áreas, incluindo a inteligência artificial. Neste trabalho, será explorado a aplicação de técnicas de aprendizado de máquina para classificação do estado do jogo.

Ao longo do trabalho, serão abordados os desafios enfrentados durante o processo de preparação dos dados, o treinamento dos modelos e a avaliação de seu desempenho. Além disso, serão discutidas as estratégias adotadas para lidar com questões como desbalanceamento de classes e escolha de métricas de avaliação adequadas.

2 Dataset

O conjunto de dados original era composto por registros de partidas do jogo da velha, onde cada entrada consistia em uma lista de nove elementos representando as posições do tabuleiro e o resultado da partida. Inicialmente, esse conjunto de dados tinha apenas duas classes: "positiva"para a vitória de 'X' e "negativa"para qualquer outro resultado.

No entanto, para seguir as requisições propostas, foi necessário realizar modificações no dataset. Se tornou necessário expandir as classes para quatro: "inGame"para partidas em curso, "xwins"para vitórias de 'x', 'owins"para vitórias de 'o' e "Draw"para empates.

Para realizar essa modificação, geramos um script em Python capaz de reestruturar o dataset original e gerar um novo conjunto de dados que atendesse às novas exigências. Esse script foi responsável por criar todas as possíveis combinações de posições do tabuleiro e classificá-las de acordo com o resultado do jogo, seguindo as novas classes estabelecidas.

Após uma análise mais aprofundada, foi identificado um desafio significativo relacionado ao desbalanceamento das classes no dataset, especialmente em relação à classe "Draw". Foi possível observar que havia uma quantidade substancialmente menor de instâncias dessa classe em comparação com as outras, devido à escassez de empates no jogo da velha em relação às outras possíveis situações.

Para lidar com esses desbalanceamento e garantir uma distribuição melhor entre as classes, foi adotada uma estratégia de aumento sintético das instâncias da classe "Draw". Essa abordagem envolveu a replicação das instâncias existentes, aumentando sua representatividade no dataset, e a distribuição após a alteração é ilustrada na Figura 1. Essa medida contribuiu para melhorar a capacidade do algoritmo de classificação em lidar com todas as condições possíveis de jogo de forma mais justa e precisa.

Após isso, o dataset foi dividido em três conjuntos distintos: treino, teste e validação. 70% do dataset foi destinado ao conjunto de treino, 20% ao conjunto de teste e 10% ao conjunto de

validação. Essa divisão permite avaliar o desempenho do modelo em dados não vistos, ajustar hiperparâmetros e evitar o overfitting.

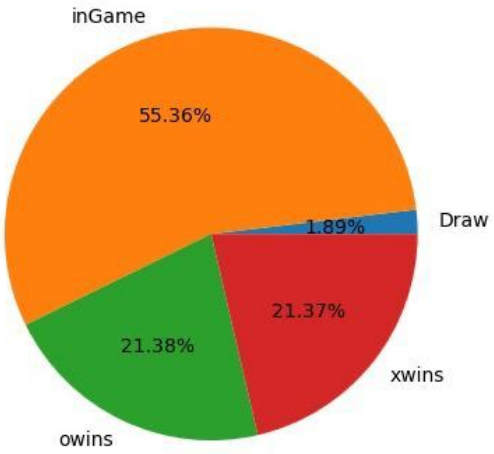

Figura 1: Distribuição d

3 Treinamento dos Modelos

Após a etapa de pré-processamento do dataset e sua divisão em conjuntos de treino, teste e validação, procedemos com o treinamento de quatro modelos de classificação: K-Nearest Neighbors (KNN), Árvore de Decisão (Decision Tree), Multi-Layer Perceptron (MLP) e Naive Bayes.

O algoritmo (K — Nearest Neighbors) é um algoritmo classificador de aprendizagem supervisionada baseado em instâncias. Classificadores baseados em instância também são chamados de "aprendizes preguiçosos", pois armazenam todas as amostras de treinamento e não constroem um classificador até que uma nova amostra, não rotulada, precise ser classificada.

Já um algoritmo de Árvore de Decisão funciona construindo uma estrutura de árvore onde cada nó representa uma decisão baseada em um atributo específico dos dados. A cada passo, o algoritmo escolhe o atributo que melhor divide os dados, buscando maximizar a pureza das classes em cada ramo da árvore.

O algoritmo MLP (Multilayer Perceptron) é composto por uma estrutura com múltiplas camadas de neurônios, incluindo uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada neurônio em uma camada está conectado a todos os neurônios na próxima camada, formando uma rede altamente interconectada. Durante o treinamento, o MLP ajusta os pesos das conexões entre os neurônios para minimizar a diferença entre as saídas previstas e os valores reais dos dados de treinamento.

Por fim, o algoritmo Naive Bayes também foi testado. Ele é baseado no teorema de Bayes, que descreve a probabilidade de um evento, dado o conhecimento prévio de condições relacionadas a esse evento. O termo "naive"(ingênuo) deriva da suposição de independência entre os recursos (ou características) utilizados para a classificação, o que simplifica o cálculo das probabilidades.

Durante o treinamento, os parâmetros de cada modelo foram ajustados de acordo com os dados de treino, permitindo que eles aprendessem os padrões presentes nos dados e fossem capazes de fazer previsões precisas.

Após o treinamento, os modelos foram avaliados utilizando o conjunto de teste. Essa etapa permitiu avaliar o desempenho dos modelos em dados não vistos durante o treinamento, fornecendo uma estimativa realista de sua capacidade de generalização.

Após implementação o jogo e testar o modelo em situações reais, observamos que, apesar das métricas de desempenho estarem altas, o modelo ainda cometia erros na classificação dos casos de

empate. Reconhecendo a importância de melhorar a precisão nessa condição específica, decidimos adotar uma abordagem especializada.

Para lidar com esse problema, desenvolvemos um modelo especialista dedicado a diferenciar entre as classes "inGame"(partidas em curso) e "Draw"(empates). Para isso, filtramos o dataset original, selecionando apenas as instâncias com a classe "Draw", e treinamos um modelo específico com base nesses dados.

Após treinar o modelo especialista, integramos sua decisão ao processo de classificação final. Criamos um comitê de modelos, combinando o modelo original, capaz de classificar todas as classes, com o modelo especialista treinado para distinguir entre "inGame"e "Draw".

Na abordagem do comitê, se o modelo especialista votasse pela classe "Draw", essa classificação seria automaticamente considerada como resultado final. No entanto, se o modelo especialista não votasse em "Draw", consideraríamos apenas o voto do modelo original para determinar a classe final.

Essa estratégia de comitê permitiu aproveitar as vantagens do modelo especialista em casos específicos, como empates, enquanto ainda se beneficiava da capacidade de classificação geral do modelo original. Essa abordagem combinada aumentou a confiabilidade do modelo fina em situações de empate.

4 Resultados

Durante a avaliação dos modelos, duas métricas principais foram utilizadas: precisão e F1-score. A precisão é uma métrica que mede a proporção de instâncias classificadas corretamente como positivas em relação ao total de instâncias classificadas como positivas, ou seja, a habilidade do modelo de evitar a classificação incorreta de instâncias negativas como positivas. É calculada pela fórmula:

$$Precisao = \frac{VP}{VP + FP}, \tag{1}$$

onde VP são as instâncias corretamente classificadas como positivas e FP são as instâncias incorretamente classificadas como positivas, quando na verdade são negativas.

O F1-score é uma medida que combina precisão e recall em uma única métrica. É particularmente útil em conjuntos de dados desbalanceados, nos quais existe uma disparidade significativa no número de instâncias entre as classes. O F1-score é calculado como a média harmônica entre precisão e recall e é definido pela seguinte fórmula:

$$F1 = \frac{2 \cdot (Precisao \cdot Recall)}{Precisao + Recall}, \tag{2}$$

O F1-score oferece uma medida do quão efetivamente o modelo equilibra a precisão (quão assertivo é o modelo ao prever uma classe) e o recall (quão cuidadoso é o modelo para não perder casos dessa classe). Em conjuntos de dados desbalanceados, onde uma classe é muito mais prevalente do que outra, a mera precisão pode ser enganosa, já que o modelo pode alcançar alta precisão simplesmente prevendo a classe majoritária.

Por levar em consideração tanto os falsos positivos quanto os falsos negativos, O F1-score torna-se uma métrica mais confiável em tais cenários. Portanto, a decisão de usar o F1-score em conjunto com a precisão no processo de avaliação foi crucial para garantir uma avaliação equilibrada do desempenho dos modelos, especialmente diante do desbalanceamento das classes no conjunto de dados.

Os resultados obtidos para cada algoritmo estão listados na Tabela 1. Durante os testes, ficou evidente que tanto o modelo Especialista quanto o MLP se destacaram com desempenho excepcionalmente elevado, seguidos pela árvore de decisão. Enquanto isso, o k-NN demonstrou um desempenho aceitável, e o Naive Bayes ficou atrás dos demais algoritmos em termos de desempenho de acurácia e f1.

Tabela 1: Resultados da avaliação dos algoritmos.

Algortimo	**Acurácia**	**F1-Score**
k-NN	0.8895	0.8687
Decision Tree	0.9922	0.9687
MLP	0.9997	0.9996
Naive Bayes	0.7624	0.7106
Especialista	0.9997	0.9996

5 Conclusão

A abordagem especializada que foi adotada para lidar com os erros de classificação nos casos de empate mostrou-se eficaz na melhoria da precisão do modelo. Ao desenvolver um modelo especialista dedicado exclusivamente à distinção entre as classes "inGame"e "Draw", se tornou possível integrar sua decisão ao processo de classificação final por meio de um comitê de modelos.

Essa estratégia permitiu capitalizar as vantagens do modelo especialista em situações específicas, como os empates, ao mesmo tempo em que ainda aproveitava-se a capacidade de classificação geral do modelo original. Como resultado, foi possível aumentar significativamente a confiabilidade do modelo final em situações de empate, o que é crucial para a precisão e usabilidade do sistema em cenários reais.

No geral, os resultados obtidos demonstram a eficácia da abordagem especializada e a importância de considerar cuidadosamente o desbalanceamento das classes ao desenvolver e avaliar modelos de aprendizado de máquina. Essa experiência destaca a necessidade de abordagens flexíveis e adaptáveis que possam lidar com desafios específicos do domínio e garantir a robustez e confiabilidade dos modelos em diferentes contextos de aplicação.