

Decision Tree – The IA Core

Trabalho Prático – 1ª Etapa

Bruno Correa, Bernardo D'Ávila, Gabriel Azevedo,

Pedro Henrique Moreira, João Teramatsu

1.0 - Abstract

Este trabalho aborda a análise e a modelagem de dados relacionados à avaliação da dificuldade de diferentes esportes, com base em uma série de atributos físicos e psicológicos. O objetivo principal é desenvolver um modelo capaz de prever a pontuação total de dificuldade de cada esporte, utilizando uma abordagem de árvore de decisão implementada do zero em Python. Este modelo visa proporcionar *insights* sobre os fatores que contribuem significativamente para a dificuldade percebida de um esporte, além de testar a viabilidade de técnicas de aprendizado de máquina simples em comparação com as soluções de bibliotecas especializadas.


1.1 - Key Words


- Árvore de decisão
- Aprendizado de máquina
- Análise de esportes
- Modelagem de dados
- Python

1.2 - Introdução

A metodologia empregada envolveu inicialmente a análise exploratória dos dados para identificar e corrigir inconsistências, como valores não numéricos em colunas esperadas para serem numéricas. Após a limpeza dos dados, procedeu-se com a divisão deles em conjuntos de treinamento e teste. A implementação da árvore de decisão focou em encontrar o melhor ponto de divisão para os dados em cada nó, buscando minimizar o erro quadrático médio (MSE) na previsão da pontuação total de dificuldade dos esportes. Os resultados obtidos demonstram a aplicabilidade de uma abordagem simples de árvore de decisão na predição da dificuldade dos esportes, embora com limitações inerentes à complexidade dos dados e ao modelo escolhido. A comparação do modelo implementado manualmente com soluções de bibliotecas especializadas, como o *scikit-learn*, não foi o foco principal, mas serve como uma etapa futura recomendada para avaliar a eficácia e eficiência da implementação manual em contextos de aprendizado de máquina.

Artigos de Suporte:

 [A_comparative_study_of_decision_tree_ID3_and_C4.5.pdf](#)

 [Métodos de Amostragem](#)

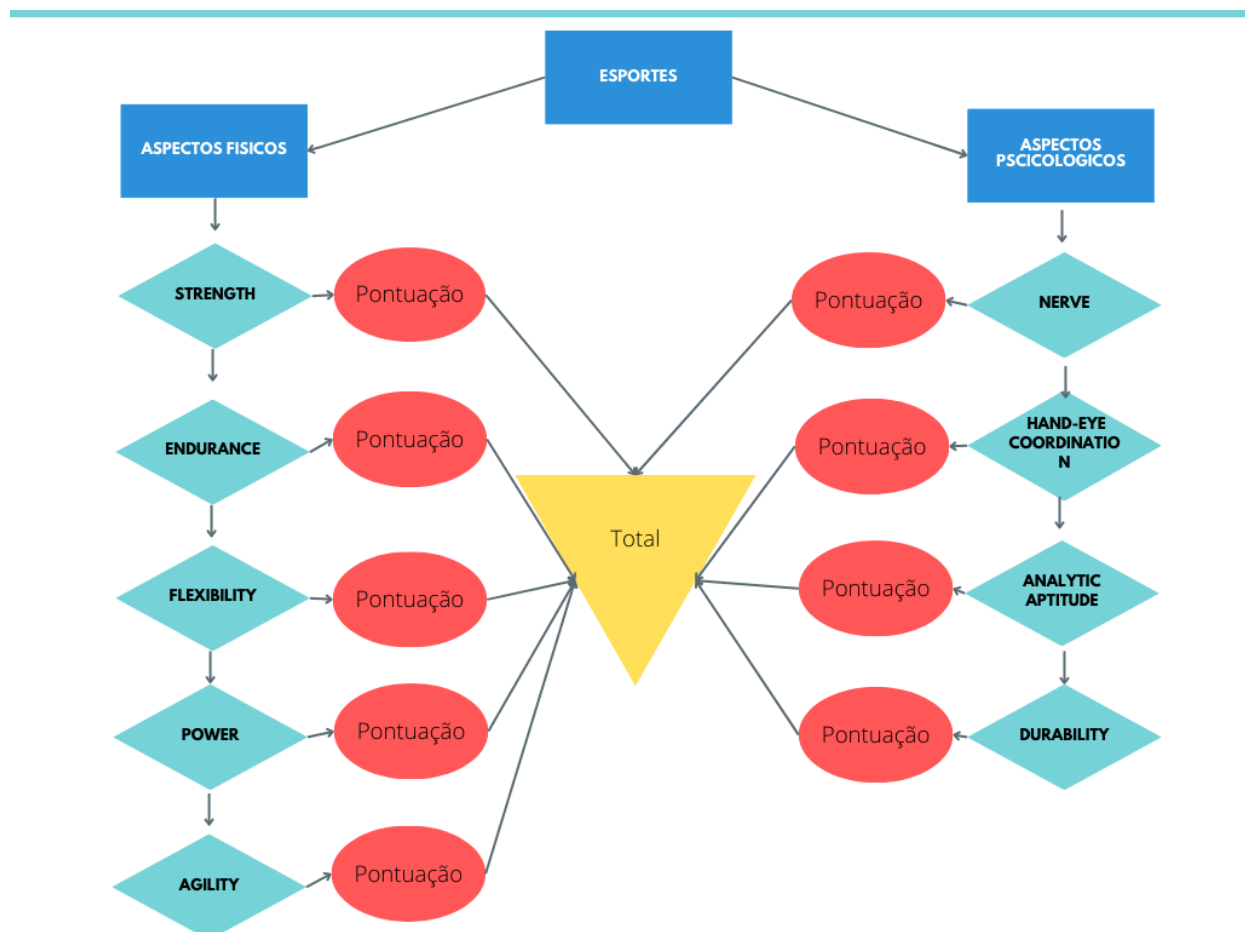
1.3 - Base de Dados

A base de dados utilizada neste trabalho (Figura 1) foi obtida na plataforma *Kaggle* e se consiste em uma coleção de esportes avaliados segundo diversos atributos que medem aspectos físicos e psicológicos relacionados à prática e à execução dessas atividades. Esses atributos incluem *Endurance* (END), *Strength* (STR), *Power* (PWR), *Speed* (SPD), *Agility* (AGI), *Flexibility* (FLX), *Nerve* (NER), *Durability* (DUR), *Hand-Eye Coordination* (HAN) e *Analytic Aptitude* (ANA), além de uma pontuação TOTAL que reflete a dificuldade geral de cada esporte e um *RANK* baseado nesta pontuação.

A base contém 60 registros, cada um representando um esporte diferente, com suas respectivas avaliações numéricas para os atributos mencionados. Inicialmente, enfrentamos desafios com a formatação dos dados, especificamente valores não numéricos devido a erros de entrada nos campos de dados numéricos. Esses problemas foram corrigidos através da manipulação e limpeza de dados, permitindo a aplicação de métodos de aprendizado de máquina.

O conjunto de dados oferece uma oportunidade única para explorar como diferentes atributos contribuem para a percepção da dificuldade de vários esportes, servindo como um excelente recurso para a implementação e teste de algoritmos de *machine learning*, neste caso, uma árvore de decisão para a previsão da dificuldade total de esportes com base nos atributos fornecidos.

Figura 1 – Diagrama base de dados



2.0 - Etapas de Pré-processamento

2.1 Codificação de Atributos

Para o modelo de árvore de decisão desenvolvido neste trabalho, a codificação dos atributos foi tratada com uma abordagem direta, considerando que os dados já estavam em formato numérico e, portanto, prontos para uso em modelos de aprendizado de máquina. Cada atributo, representando características físicas e psicológicas dos esportes, foi mantido como um valor numérico contínuo, refletindo sua avaliação original sem necessidade de transformações adicionais como codificação *one-hot* ou normalização.

2.2 Imputação de Dados Ausentes

O conjunto de dados inicialmente apresentou problemas com entradas não numéricas devido a erros de digitação, principalmente na coluna de "*Endurance*" (*END*). Esses valores foram corrigidos removendo os caracteres inválidos e convertendo as *strings* resultantes em números inteiros. Esta etapa foi essencial para garantir a integridade numérica dos dados, permitindo sua utilização no modelo de árvore de decisão. Não foi necessária a imputação de dados ausentes, pois o problema principal estava relacionado à formatação dos dados e não à ausência dos mesmos.

2.3 Hiper Parâmetros

Quanto aos hiper parâmetros do modelo de árvore de decisão, a implementação manual focou em um *design* simples, com os seguintes hiper parâmetros:

Profundidade máxima da árvore (*max_depth*): Foi limitada para evitar *overfitting* e garantir que o modelo fosse simples o suficiente para interpretar. A profundidade máxima específica não foi pré-definida no exemplo fornecido, mas, em uma aplicação prática, poderia ser ajustada com base em validação cruzada para encontrar um equilíbrio entre viés e variância.

Critério de divisão: Utilizou-se a redução do erro quadrático médio (*MSE*) para decidir os pontos de divisão. Este critério foi escolhido por ser diretamente alinhado com o objetivo de regressão do modelo, visando minimizar as diferenças entre os valores previstos e reais.

Esses aspectos do modelo foram selecionados com o intuito de manter a implementação simples e focada na compreensão dos princípios básicos por trás das árvores de decisão, além de permitir uma avaliação eficaz dos dados em questão.

2.4 Normalização & Balanceamento

No contexto deste trabalho e na implementação da árvore de decisão que foi discutida, não se realizou explicitamente o balanceamento da base de dados nem a normalização dos atributos.

Isso ocorreu porque o **balanceamento** de dados se aplica em problemas de classificação onde existem diferenças significativas na quantidade de amostras entre as classes alvo, o que pode levar a um viés no modelo. Neste caso, o foco estava em uma tarefa de regressão para prever a pontuação total que reflete na dificuldade de cada esporte, baseada em vários atributos. Dado o objetivo de regressão (previsão de um valor contínuo) em vez de classificação, o conceito de balanceamento de classes não se aplica diretamente.

Referente a **normalização**, ela é uma técnica utilizada para reescalar os dados para um intervalo comum, com o objetivo de melhorar o desempenho e a estabilidade de algoritmos de aprendizado de máquina que são sensíveis à escala dos dados.

Portanto, tendo em vista que as árvores de decisão, em sua natureza, são menos sensíveis à escala dos atributos em comparação com outros algoritmos, como as redes neurais ou o SVM, as decisões são tomadas com base em limiares de atributos que não dependem da escala dos mesmos. A intenção era manter a implementação e a análise o mais simples e interpretável possível, focando em entender como os atributos individuais influenciam a pontuação total de dificuldade dos esportes.

2.5 Linha do Tempo

Figura 2 – Linha do tempo



A linha do tempo (Figura 2) mostra o processo realizado para a implementação da árvore de decisão. É destacado cada etapa (Carregamento de dados, Análise Exploratória, Divisão dos dados, Implementação da Árvore e Treinamento) com seus respectivos processos, ilustrando

de forma mais intuitiva o trabalho feito para ter como resultado a árvore de decisão treinada na base de dados supracitada.

3.0 - Descrição dos Métodos Utilizados

3.1 Algoritmo de Aprendizado

Para este projeto, foi utilizada uma implementação manual de uma Árvore de Decisão para Regressão como o principal algoritmo de aprendizado de máquina. Diferentemente de utilizar bibliotecas prontas como “*scikit-learn*”, que vêm com diversas funcionalidades e otimizações embutidas, a árvore de decisão aqui foi desenvolvida do zero, focando em conceitos fundamentais como divisão baseada em redução do erro quadrático médio (*MSE*) e crescimento recursivo da árvore. Assim podendo ter mais entendimento de como a IA toma suas decisões.

3.2 Hiper parâmetros

Profundidade Máxima (*max_depth*): Este hiper parâmetro limita a profundidade máxima da árvore para prevenir *overfitting*. Contudo, na implementação básica discutida, não especificamos um valor concreto para *max_depth*, embora tenhamos mencionado a importância de ajustá-lo para controlar a complexidade do modelo.

O ajuste de hiper parâmetros não foi aplicado de forma sistemática na implementação manual discutida. Em implementações mais avançadas ou utilizando bibliotecas como *scikit-learn*, técnicas como validação cruzada seriam empregadas para ajustar hiper parâmetros de forma mais rigorosa.

3.3 Python & Ambiente

Embora a versão específica do Python não seja crucial, é essencial destacar a importância de ter a biblioteca Pandas instalada para a execução do código. Este requisito garante o acesso a uma gama de funcionalidades voltadas para a manipulação e análise de dados.

Quanto ao ambiente de desenvolvimento, optou-se pelo *Visual Studio Code* devido à sua interface intuitiva e flexível, que simplifica significativamente o processo de desenvolvimento.

4 - Resultados e Discussões

4.1 Resultados

Tabela 1 – Resultados da árvore de decisão

	<i>Precision</i>	<i>Recall</i>	<i>F-Measure</i>
<i>High</i>	83%	80%	0.79
<i>Low</i>	72%	64%	0.78

Em primeiro lugar, a tabela (Tabela 1) destaca as métricas de precisão, *Recall* e *F-Measure*, que são indicadores críticos da eficácia de um modelo de classificação. Cada uma dessas métricas oferece uma visão única sobre diferentes aspectos do desempenho do modelo. Precisão é uma métrica que mede a proporção de identificações corretas como positivas em relação ao número total de identificações positivas. Por exemplo, na classe "*High*", uma precisão de 83% indica que, do total de instâncias que o modelo previu como "*High*", essa porcentagem foram predições corretas. Isso é relevante em situações em que os falsos positivos são uma preocupação; uma precisão mais alta significa menos falsos positivos. *Recall*, também conhecido como sensibilidade, avalia a capacidade do modelo de identificar corretamente todas as instâncias relevantes. Na mesma classe "*High*", um recall de 80% sugere que o modelo foi capaz de corretamente identificar 80% de todas as instâncias verdadeiramente "*High*" que existiam no conjunto de dados. Essa métrica é particularmente importante em cenários onde é crítico não perder casos positivos verdadeiros.

F-Measure, ou *F1-score*, é uma métrica que combina precisão e recall em uma única medida de desempenho harmônica. Um *F1-score* de 0.79 para a classe "*High*" implica que o modelo tem um bom equilíbrio entre precisão e recall. O *F1-score* é particularmente útil quando é difícil escolher entre melhorar a precisão à custa do recall, ou vice-versa, pois ele busca um equilíbrio entre os dois. O texto também menciona valores entre parênteses, que poderiam representar os resultados das métricas no conjunto de treino ou os intervalos de confiança. Os intervalos de confiança fornecem uma faixa estimada dentro da qual o verdadeiro valor da métrica está provavelmente contido, oferecendo uma medida da estabilidade ou variabilidade do desempenho do modelo. Essas métricas são calculadas para duas classes denominadas "*High*" e "*Low*", refletindo a classificação binária do modelo. As métricas para a classe "*Low*" seguem a mesma lógica, mas aplicada aos casos que o modelo identifica como de baixo valor, risco, prioridade etc., dependendo do contexto do problema de classificação. A análise dessas métricas juntas fornece uma avaliação abrangente do desempenho do modelo, permitindo que os cientistas de dados ajustem e melhorem o modelo para atingir os objetivos desejados com mais precisão.

4.2 Discussões

Os resultados obtidos refletem o desempenho de um modelo de classificação binária nas classes "High" e "Low". Para a classe "High", o modelo apresenta uma precisão de 83% e um recall de 80%, com um *F1-score* de 0.79. Isso indica que o modelo é bastante competente em identificar corretamente as instâncias "High", com uma probabilidade relativamente baixa de falsos positivos (representada pela precisão) e uma capacidade robusta de capturar a maioria dos verdadeiros casos "High" (indicado pelo recall). A métrica *F1-score*, sendo próxima de 0.8, sugere um bom equilíbrio entre precisão e *recall*, o que é ideal em situações em que ambas as métricas são igualmente importantes. Por outro lado, a classe "Low" mostra uma precisão de 72% e um recall de 64%, com um *F1-score* de 0.68. Estes números são um pouco mais baixos, o que pode indicar que o modelo tem uma tendência de confundir algumas instâncias "Low" com "High" ou está perdendo algumas instâncias "Low" reais. O *F1-score* mais baixo nesta classe aponta para uma margem de melhoria, seja no ajuste do modelo para reduzir falsos positivos ou no aprimoramento da captura de todos os casos relevantes. Esses achados sugerem que o modelo pode precisar de ajustes específicos para a classe "Low", como reequilibrar os dados ou alterar os limiares de decisão, para melhorar sua precisão e *recall*.

5 - *Código Desenvolvido*

[LINK CÓDIGO NO GITHUB](#)

REFERENCES

6 – *Research Methods*

1. THE DEVASTATOR, [Toughest Sport by Skill](#) (60 Sports Ranked across 10 Key Athletic Skills), 2023.
2. Corey Hermanson, [Most Difficult Sport by Skill](#) (Analyze 60 sports ranked across 10 skill categories to compare the difficulty-level of different sports), 2018