Técnicas de Deep Learning para Previsão de Dois ou Mais Gols no Campeonato Inglês (Premier League)

Vitor Alexandre Domingues Oliveira

1 de dezembro de 2025

Resumo

Nos últimos anos, a aplicação de técnicas de inteligência artificial e aprendizado de máquina no esporte tem ganhado destaque, especialmente em tarefas relacionadas à análise de desempenho e previsões estatísticas. No futebol, grande parte das pesquisas concentra-se na previsão do resultado final das partidas — vitória, empate ou derrota. Contudo, essa abordagem tradicional apresenta limitações, pois ignora aspectos táticos e ofensivos que influenciam diretamente a dinâmica do jogo. Diante desse cenário, este trabalho propõe uma alternativa: a previsão da ocorrência de dois ou mais gols em partidas do Campeonato Inglês (Premier League), considerando que esse indicador representa um reflexo mais direto da produtividade ofensiva das equipes.

O presente estudo teve como objetivo desenvolver, treinar e validar um modelo preditivo baseado em técnicas de deep learning capaz de estimar a probabilidade de partidas ultrapassarem a marca de dois gols. Para isso, foram utilizadas redes neurais profundas estruturadas em camadas densas (DNN/MLP), devido à sua capacidade de capturar relações não lineares e padrões complexos presentes nos dados esportivos. A pesquisa fundamenta-se em princípios do aprendizado supervisionado, com ênfase nas redes neurais artificiais e em métricas quantitativas para avaliação de desempenho.

A metodologia envolveu inicialmente a coleta automática de dados históricos da Premier League por meio de web scraping, utilizando a biblioteca Selenium. Os dados obtidos passaram por etapas de pré-processamento, normalização e organização com o auxílio das bibliotecas Pandas e Scikit-learn, garantindo consistência e qualidade para uso na modelagem. O modelo foi implementado em Python, utilizando a biblioteca PyTorch, e treinado com um conjunto de dados particionado em 80% para treino e 20% para teste.

Os resultados obtidos demonstram que o modelo alcançou acurácia média de aproximadamente **79,2%** na previsão da ocorrência de dois ou mais gols, indicando que a abordagem proposta é viável e apresenta desempenho competitivo para esse tipo de tarefa preditiva. Assim, conclui-se que o modelo desenvolvido pode servir como uma ferramenta útil tanto para analistas e comissões técnicas quanto para o setor de apostas esportivas, fornecendo insights complementares à análise tradicional focada apenas no placar final.

Palavras-chave: Inteligência Artificial. Rede Neural. Premier League.

Abstract

In recent years, the application of artificial intelligence and machine learning techniques in sports has gained significant relevance, especially in tasks related to performance analysis and statistical prediction. In football, most studies focus on forecasting the final match outcome — win, draw, or loss. However, this traditional approach presents limitations, as it overlooks tactical and offensive aspects that directly influence match dynamics. In this context, the present work proposes an alternative perspective: predicting the occurrence of two or more goals in English Premier League matches, considering that this indicator more directly reflects the offensive productivity of the teams.

The objective of this study was to develop, train, and validate a predictive model based on deep learning techniques capable of estimating the probability of matches exceeding the two-goal threshold. Deep neural networks structured with dense layers (DNN/MLP) were employed due to their ability to capture nonlinear relationships and complex patterns present in sports data. The research is grounded in supervised learning principles, with emphasis on artificial neural networks and quantitative performance metrics.

The methodology initially involved the automated collection of historical Premier League data through web scraping using the Selenium library. The collected data underwent preprocessing, normalization, and organization with the support of Pandas and Scikit-learn, ensuring consistency and quality for modeling. The model was implemented in Python using the PyTorch library and trained with a dataset split into 80% for training and 20% for testing.

The results demonstrate that the model achieved an average accuracy of approximately **79.2%** in predicting the occurrence of two or more goals, indicating that the proposed approach is viable and presents competitive performance for this predictive task. Therefore, the developed model can serve as a useful tool for analysts, coaching staff, and even the sports betting sector, providing complementary insights to traditional analyses focused solely on the final score.

1 Introdução

A aplicação de técnicas de ciência de dados e inteligência artificial no futebol tem ganhado destaque nas últimas décadas, especialmente em estudos voltados à modelagem de probabilidades de vitória, empate ou derrota em partidas. Trabalhos como o de Boacnin et al. (2023) demonstram a viabilidade do uso de arquiteturas neurais, como o Multi-Layer Perceptron (MLP), para a previsão de resultados no Campeonato Brasileiro. No entanto, embora a literatura apresente avanços consistentes, a maior parte das pesquisas permanece restrita à classificação do desfecho final da partida, deixando em segundo plano outras dimensões fundamentais do jogo, como seu comportamento ofensivo.

Diante dessa lacuna, o presente trabalho propõe uma mudança de perspectiva ao deslocar o foco da tradicional previsão do vencedor para a análise de um indicador ofensivo específico: a ocorrência de dois ou mais gols em uma partida. A proposta fundamenta-se na utilização de técnicas de deep learning, capazes de extrair padrões complexos e abstrações profundas a partir de dados estatísticos. Para isso, foi empregada uma base de dados referente à Premier League, um dos campeonatos mais competitivos e analisados do mundo, cuja riqueza estatística favorece a modelagem de comportamentos ofensivos.

A construção do modelo preditivo foi realizada a partir de um processo estru-

turado que envolveu a coleta automatizada dos dados por meio de web scraping com a biblioteca Selenium, seguida pelo pré-processamento, organização e normalização das informações utilizando ferramentas como Pandas e Scikit-learn. Na etapa de modelagem, optou-se pela linguagem Python devido ao seu ecossistema robusto para aprendizado de máquina, sendo a biblioteca PyTorch utilizada como principal framework para implementação de uma rede neural do tipo MLP com camadas densas.

Com o objetivo de suprir a necessidade de análises mais precisas e aprofundadas no futebol, o estudo resultou no desenvolvimento de um modelo capaz de quantificar o potencial ofensivo das partidas com base em dados históricos. O modelo treinado alcançou acurácia média de aproximadamente 79,2% na tarefa de classificar jogos com dois ou mais gols. Esse desempenho demonstra a viabilidade da abordagem e evidencia que métodos baseados em deep learning podem ir além da simples previsão do resultado final, oferecendo novas possibilidades de análise tática e estratégica.

Os resultados obtidos indicam que a ferramenta desenvolvida possui potencial aplicação em diversos segmentos do futebol. Analistas de desempenho e comissões técnicas podem utilizá-la para antecipar tendências ofensivas de adversários, apoiando a tomada de decisão em planejamentos de jogo. Da mesma forma, a mídia esportiva e plataformas especializadas podem enriquecer suas análises pré-jogo ao incorporar indicadores quantitativos derivados da modelagem preditiva. Assim, o trabalho contribui para a ampliação do uso de metodologias de inteligência artificial no esporte, reforçando a relevância de abordagens preditivas como suporte à análise contemporânea do futebol.

2 Objetivos

2.1 Objetivo Geral

Desenvolver, treinar e validar um modelo preditivo baseado em técnicas de aprendizado profundo para estimar a probabilidade de ocorrência de dois ou mais gols em partidas do Campeonato Inglês (Premier League), utilizando dados históricos das partidas.

2.2 Objetivos Específicos

1. Realizar a coleta, o pré-processamento e a normalização de um conjunto de dados históricos contendo estatísticas detalhadas de jogos da Premier League, estruturando o dataset utilizado na modelagem preditiva.

2. Implementar uma rede neural densa utilizando a biblioteca PyTorch, definindo sua arquitetura, hiperparâmetros e demais configurações necessárias ao treinamento.

3. Treinar e avaliar o desempenho do modelo preditivo por meio de um conjunto de dados particionado em treino e teste, utilizando a acurácia como métrica principal para medir sua capacidade de prever a ocorrência de dois ou mais gols.

4. Analisar e interpretar os resultados obtidos, verificando a eficácia e a viabilidade do modelo proposto como ferramenta de suporte à análise quantitativa no contexto do futebol.

3 Referencial Teórico

3.1 Inteligência Artificial e Machine Learning

A Inteligência Artificial (IA) é um campo da ciência da computação dedicado à criação de agentes inteligentes, que são sistemas capazes de perceber seu ambiente e executar ações para maximizar suas chances de sucesso em um determinado objetivo, como descrito por Russell e Norvig (2013). Em outras palavras, a IA busca desenvolver máquinas que possam realizar tarefas que, historicamente, exigiriam inteligência humana, como o raciocínio, a aprendizagem e a resolução de problemas.

Dentro do vasto campo da IA, uma subárea de fundamental importância é o *Machine Learning* (ML), ou Aprendizado de Máquina. Diferente da programação tradicional, onde as regras são explicitamente codificadas por um desenvolvedor, o ML se baseia em algoritmos que permitem ao computador "aprender"essas regras a partir de dados. A definição clássica de Samuel (1959), descreve o ML como o campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados.

Existem três paradigmas principais de Aprendizado de Máquina:

1. **Aprendizagem Supervisionada:** Neste modelo, o algoritmo é treinado com um conjunto de dados previamente rotulado, ou seja, cada dado de entrada possui uma "resposta correta"correspondente. O objetivo é que o modelo aprenda a mapear as entradas para as saídas corretas, para que possa fazer previsões sobre dados novos e não vistos. As tarefas supervisionadas se dividem principalmente em regressão (prever um valor contínuo) e classificação (prever uma categoria).

2. **Aprendizagem Não Supervisionada:** Aqui, o algoritmo recebe dados sem rótulos e sua tarefa é encontrar estruturas ou padrões ocultos por conta própria. A técnica mais comum é o agrupamento (*clustering*), que visa agrupar dados semelhantes em conjuntos distintos.

3. **Aprendizagem por Reforço:** Inspirado na psicologia comportamental, este paradigma envolve um "agente"que aprende a tomar decisões executando ações em um ambiente a fim de maximizar uma recompensa cumulativa. O aprendizado ocorre por meio de tentativa e erro.

O presente trabalho se enquadra na categoria de **Aprendizagem Supervisionada**. O modelo de rede neural será treinado utilizando um conjunto de dados históricos de partidas onde cada jogo (entrada) possui um rótulo definido (saída): a ocorrência ou não de dois ou mais gols. Especificamente, trata-se de um **problema de classificação binária**, pois o objetivo é classificar cada partida em uma de duas categorias possíveis: "Over 1.5"(classe 1) ou "Under 1.5"(classe 0).

3.2 Redes Neurais Artificiais (RNAs)

As Redes Neurais Artificiais (RNAs) são modelos computacionais inspirados na estrutura e funcionamento do cérebro humano. A sua unidade fundamental é o neurônio artificial, ou Perceptron, um conceito introduzido por Rosenblatt (1958), que funciona como um processador de informações simples. Um neurônio artificial recebe múltiplos sinais de entrada, realiza uma soma ponderada desses sinais e, em

seguida, aplica uma função matemática, chamada de função de ativação, para gerar um sinal de saída.

A verdadeira capacidade das RNAs emerge da organização desses neurônios em camadas interconectadas. A estrutura mais comum é composta por:

- **Camada de Entrada (Input Layer):** Recebe os dados brutos do problema, onde cada neurônio corresponde a uma feature (variável) do conjunto de dados.
- **Camadas Ocultas (Hidden Layers):** São as camadas intermediárias, localizadas entre a entrada e a saída. É nelas que a maior parte do "aprendizado"ocorre, através da extração de padrões e características cada vez mais complexos dos dados.
- **Camada de Saída (Output Layer):** Produz o resultado final do modelo, como a previsão de uma classe ou de um valor numérico.

O aprendizado em uma RNA ocorre através do ajuste de seus parâmetros, os **pesos (weights)** e o **bias**. Os pesos determinam a força da conexão entre os neurônios de camadas adjacentes, significando a importância de um determinado sinal. O bias, por sua vez, atua como um termo de ajuste, permitindo que a função de ativação seja deslocada para a esquerda ou para a direita, aumentando a flexibilidade do modelo. São esses os valores que o algoritmo de treinamento otimiza.

Um componente essencial para o poder de uma RNA é a **Função de Ativação**. Sem ela, mesmo uma rede com múltiplas camadas se comportaria como um simples modelo linear. A função de ativação introduz não-linearidade, permitindo que a rede aprenda relações complexas entre as variáveis. Neste trabalho, duas funções de ativação são cruciais:

- **ReLU (Rectified Linear Unit):** Aplicada nas camadas ocultas, a função ReLU é definida como $f(x) = \max(0, x)$. Ela é computacionalmente eficiente e ajuda a mitigar o problema do desaparecimento do gradiente, sendo uma das ativações mais populares em redes neurais profundas Goodfellow, Bengio e Courville (2016).
- **Sigmoid:** Utilizada na camada de saída do modelo, a função Sigmoid transforma qualquer valor de entrada em um número no intervalo entre 0 e 1. Isso a torna ideal para problemas de classificação binária, como o deste trabalho, pois a saída pode ser interpretada como a probabilidade de a amostra pertencer à classe positiva (neste caso, a probabilidade de a partida ter "Mais de 1.5"gols).

3.3 Deep Learning e Redes Neurais Profundas

O *Deep Learning* (Aprendizagem Profunda) representa uma evolução das redes neurais artificiais tradicionais. A característica que define uma arquitetura como "profunda"é a presença de múltiplas camadas ocultas empilhadas sequencialmente. Enquanto uma rede neural "rasa"(com apenas uma camada oculta) pode aprender padrões, uma Rede Neural Profunda (DNN) tem a capacidade de aprender uma hierarquia de características Goodfellow, Bengio e Courville (2016).

A vantagem dessa profundidade reside na aprendizagem de representações em múltiplos níveis de abstração. As primeiras camadas da rede aprendem a detectar características simples e de baixo nível. As camadas subsequentes combinam essas

características para aprender padrões cada vez mais complexos e abstratos. Em um problema de visão computacional, por exemplo, as primeiras camadas poderiam aprender a identificar bordas, as seguintes a combinar bordas para formar formas como olhos e narizes, e as camadas mais profundas a reconhecer um rosto completo. No contexto deste trabalho, a hipótese é que as primeiras camadas aprendam relações simples entre estatísticas (ex: mais chutes a gol de um time), enquanto as camadas mais profundas aprendam a representar conceitos táticos abstratos, como o "potencial ofensivo"de uma partida.

Embora existam diversas arquiteturas de *deep learning* especializadas, como as Redes Neurais Convolucionais (CNNs) para imagens e as Redes Neurais Recorrentes (RNNs) para dados sequenciais, o presente trabalho foca na implementação de uma **Rede Neural Densa (Dense Neural Network)**, também conhecida como **Perceptron de Múltiplas Camadas (MLP)**. Nesta arquitetura, cada neurônio de uma camada está totalmente conectado a todos os neurônios da camada seguinte, formando uma base robusta para problemas de classificação com dados tabulares, como é o caso das estatísticas de futebol.

3.4 O Processo de Treinamento de uma Rede Neural

O treinamento de uma rede neural é um processo de otimização cujo objetivo é encontrar o conjunto de pesos e biases que minimiza o erro do modelo. Esse processo é orquestrado por três componentes principais: a função de perda, o algoritmo de otimização e o algoritmo de backpropagation.

- **Função de Perda (Loss Function):** É uma função matemática que quantifica a discrepância entre a previsão do modelo e o valor real (o rótulo). Para este trabalho, que se trata de uma classificação binária com saída probabilística, será utilizada a função de perda de **Entropia Cruzada Binária (Binary Cross-Entropy Loss)**. Essa função é particularmente eficaz porque mede a divergência entre duas distribuições de probabilidade (a prevista pelo modelo e a real), penalizando severamente as previsões que estão confiantes e erradas Goodfellow, Bengio e Courville (2016).

- **Algoritmo de Otimização:** Seu papel é ajustar os parâmetros do modelo (pesos e bias) na direção que minimiza a função de perda. O conceito base é o do **Gradiente Descendente**. Neste trabalho, será utilizado o **Gradiente Descendente Estocástico com Momentum (SGD with Momentum)**, uma variação aprimorada que, além de usar o gradiente, adiciona um termo de "embalo"(momentum) para acelerar a convergência e ajudar a superar mínimos locais (QIAN, 1999).

- **Backpropagation (Retropropagação do Erro):** É o motor que torna o treinamento de redes neurais profundas viável. Trata-se de um algoritmo eficiente para calcular o gradiente da função de perda em relação a cada peso da rede. Ele funciona propagando o sinal de erro da camada de saída para trás, através das camadas ocultas, até a camada de entrada, permitindo que o otimizador saiba exatamente como ajustar cada peso para reduzir o erro total do modelo (RUMELHART; HINTON; WILLIAMS, 1986).

3.5 Previsão de Resultados no Futebol com Inteligência Artificial

A aplicação de técnicas de *machine learning* para prever resultados em esportes, especialmente no futebol, não é uma área nova. Diversos pesquisadores têm explorado diferentes abordagens para modelar a complexidade de uma partida. O trabalho de Boacnin et al. (2023), por exemplo, demonstrou a eficácia de um modelo MLP para prever o resultado final (vitória, empate ou derrota) de jogos do Campeonato Brasileiro, utilizando estatísticas de desempenho das equipes.

Em uma linha de pesquisa comparativa, Gásquez et al. (2022) avaliaram o desempenho de múltiplos algoritmos de *machine learning* – incluindo Regressão Logística, Máquinas de Vetores de Suporte (SVM) e XGBoost – para prever os resultados de jogos da liga espanhola. O estudo concluiu que, embora os modelos tradicionais apresentem resultados consistentes, há uma dificuldade inerente em superar certos limiares de acurácia devido à natureza estocástica do esporte, o que abre espaço para abordagens mais complexas. De forma mais específica e alinhada a este trabalho, Eggels, Heide e Roer (2021) utilizaram uma arquitetura de *deep learning* (MLP) para prever o número de gols em partidas de diversas ligas europeias. Seus resultados indicaram que modelos profundos conseguem capturar relações não-lineares nos dados que podem passar despercebidas por modelos estatísticos mais simples, mostrando-se promissores para mercados de apostas baseados em gols.

O presente trabalho se insere neste contexto, mas se diferencia ao focar em um mercado específico e de grande interesse: a previsão da quantidade de gols (especificamente, "Mais de 1.5"), em vez de se limitar ao resultado final da partida. Ao fazer isso, busca-se não apenas prever um desfecho, mas quantificar o potencial ofensivo de um confronto, uma análise com aplicações diretas tanto para entusiastas e o mercado de apostas quanto para a análise de desempenho tático das equipes.

4 Metodologia

A metodologia adotada neste projeto, detalhada no fluxograma da Figura 1, teve início na coleta de dados por meio da técnica de web scraping, implementada na linguagem Python com o suporte das bibliotecas Selenium, Pandas, Time e tqdm. Uma vez coletada, a base de dados foi submetida a um processo de tratamento, limpeza e normalização, seguido pela partição dos dados em conjuntos de treino e teste. O núcleo do trabalho residiu no desenvolvimento de um modelo preditivo, também em Python, empregando as bibliotecas Pandas, Numpy, PyTorch, Matplotlib e Scikit-learn.

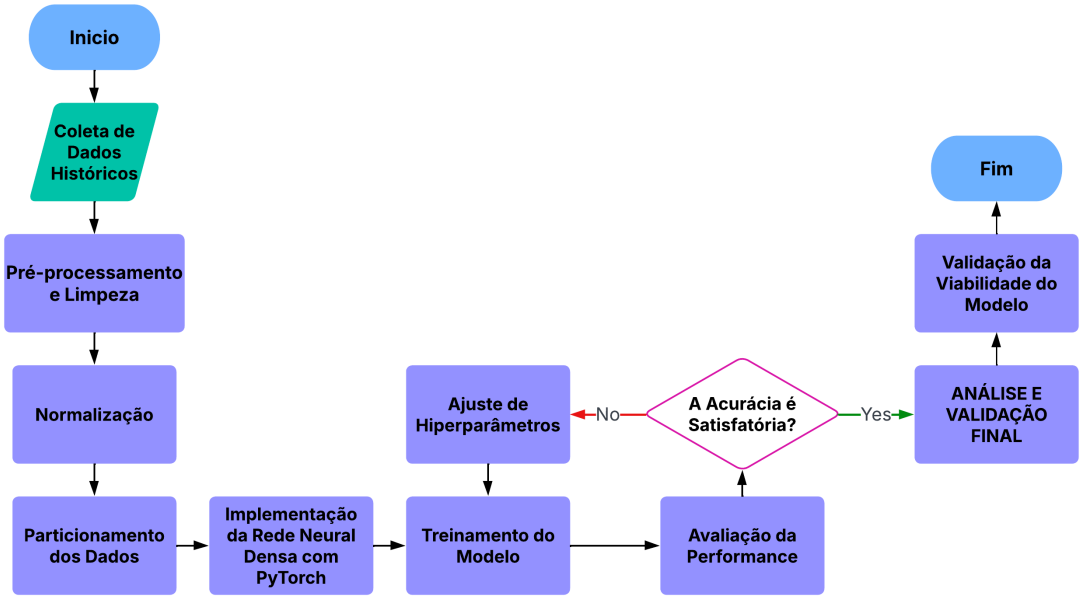

Figura 1: Fluxograma da coleta de dados e desenvolvimento do modelo preditivo.

4.1 Coleta de Dados via Web Scraping

A etapa inicial do projeto consistiu na aquisição de um conjunto de dados robusto para o treinamento e validação do modelo. Para este fim, foi empregada a técnica de web scraping (raspagem de dados). Foi desenvolvido um script na linguagem Python para automatizar a extração de informações de partidas anteriores para a formação da base de dados do site https://www.flashscore.com.

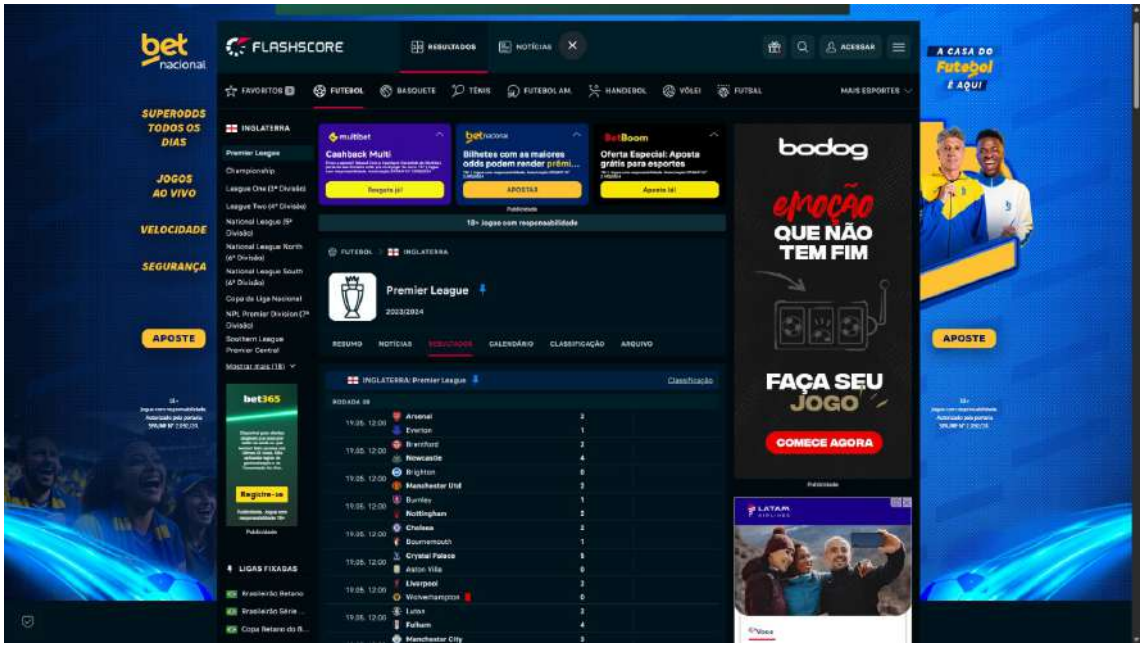

Figura 2: Imagem do site aonde foram coletados os dados.

As seguintes bibliotecas foram cruciais nesta fase:

1. Selenium: É uma biblioteca de automação de navegadores. Sua função principal é simular as ações de um usuário humano em um navegador web, como o Google Chrome ou Firefox. O Selenium foi responsável por instanciar e navegar até a página de resultados da Premier League no Flashscore e, crucialmente, interagir com elementos dinâmicos da página que não carregam imediatamente ((Selenium Project, 2025)). Ações como clicar repetidamente no botão "Mostrar mais jogos"para carregar o histórico de partidas antigas ou clicar em cada partida individualmente para acessar a página de estatísticas detalhadas foram executadas pelo Selenium. Após a navegação, ele foi usado para localizar os elementos HTML que contêm os dados desejados (ex: gols, escanteios, cartões) e extrair seu conteúdo textual.

2. Pandas: É a principal biblioteca em Python para manipulação e análise de dados. Sua estrutura central é o DataFrame, uma tabela bidimensional altamente eficiente. Conforme o Selenium extraía os dados de cada partida (ex: time da casa, time visitante, chutes a gol, escanteios, etc.), essas informações foram temporariamente armazenadas em uma estrutura Python, como uma lista de dicionários. Ao final do processo de raspagem, a biblioteca Pandas foi utilizada para converter essa lista de dados brutos em um DataFrame estruturado. Essa organização foi vital para a próxima etapa, pois permitiu a fácil limpeza, manipulação e, finalmente, o salvamento de toda a base de dados em um único arquivo no formato .csv com o comando `df.to_csv()` ((MCKINNEY, 2017)).

3. Time: É uma biblioteca padrão do Python que fornece diversas funções relacionadas ao tempo. A sua principal função neste projeto foi a time.sleep(), que serviu para inserir pausas estratégicas entre as requisições, evitando sobrecarregar o servidor do site e simulando um comportamento de navegação mais humano ((Python Software Foundation, 2025)).

4. tqdm: É uma biblioteca especializada em criar barras de progresso inteligentes e fáceis de usar para laços e processos demorados. Visto que a coleta de dados de centenas de partidas é uma tarefa que pode levar de vários minutos a horas, a tqdm foi utilizada para "envelopar"o laço principal da raspagem (o for que itera sobre cada partida). Com isso, foi exibida no terminal uma barra de progresso dinâmica, informando o percentual de conclusão, o número de partidas já processadas, o tempo decorrido e uma estimativa do tempo restante ((COSTA-LUIS et al., 2024)). Isso permitiu um monitoramento preciso e em tempo real do andamento da coleta.

Ao final desta etapa, os dados coletados foram consolidados e salvos em um arquivo no formato .csv, servindo como a base de dados bruta para as fases seguintes.

4.2 Pré-processamento e Normalização dos Dados

Com a base de dados consolidada, a etapa de pré-processamento foi executada para preparar e adequar os dados para o treinamento do modelo de Machine Learning. Esta fase foi fundamental para garantir a qualidade da entrada do modelo e, consequentemente, a precisão de suas previsões. As principais tarefas e bibliotecas utilizadas foram:

1. Pandas e Numpy: O Pandas, conforme mencionado anteriormente, é uma biblioteca de alto nível para manipulação e análise de dados estruturados, enquanto o Numpy (Numerical Python) é a biblioteca fundamental para computação numérica em Python, fornecendo a estrutura de array multidimensional na qual o Pandas é construído. O Pandas foi utilizado para as tarefas de limpeza de dados diretamente no DataFrame, para tratar valores ausentes (nulos), garantir que as colunas tivessem o tipo de dado correto (ex: numérico) e remover colunas irrelevantes que não seriam utilizadas no treinamento. O Numpy atuou de forma subjacente, garantindo a performance dessas operações, e foi essencial no momento da conversão dos dados do formato DataFrame para arrays numéricos, como discutido em Harris et al. (2020), que é o formato esperado por bibliotecas como a Scikit-learn.

2. Scikit-learn (sklearn.preprocessing): O Scikit-learn é um dos mais importantes e extensos ecossistemas de ferramentas para Machine Learning em Python. O módulo sklearn.preprocessing contém uma variedade de funções para transformar variáveis (features) brutas em uma representação mais adequada para os modelos. Foi utilizada a classe MinMaxScaler, como recomendado em Pedregosa et al. (2011), deste módulo para aplicar a técnica de Normalização nos dados de entrada. Matematicamente, para cada valor x em uma feature, o MinMaxScaler calcula (x - min) / (max - min), onde min e max são os valores mínimo e máximo daquela feature no conjunto de treino. Essa transformação redimensionou todas as variáveis numéricas para um intervalo comum de 0 a 1. Este passo foi crucial para modelos de redes neurais, pois garantiu que todas as features contribuíssem de forma equilibrada durante o processo de treinamento, evitando que variáveis com escalas de magnitude muito maiores (ex: posse de bola vs. número de faltas) dominassem o aprendizado do modelo.

4.3 Desenvolvimento e Treinamento do Modelo Preditivo

A etapa central deste trabalho, o desenvolvimento e treinamento do modelo preditivo, foi implementada em Python com o suporte da biblioteca PyTorch, um framework de deep learning que oferece flexibilidade e alto desempenho. A arquitetura projetada foi uma Rede Neural Profunda (DNN), do tipo Perceptron de Múltiplas Camadas (MLP). Para o treinamento, o conjunto de dados pré-processado foi dividido em 80% para treino e 20% para teste. O modelo foi então treinado de forma iterativa com o conjunto de treino.

4.4 Validação e Análise de Resultados

Após o treinamento, o desempenho do modelo foi rigorosamente avaliado utilizando o conjunto de dados de teste, que o modelo nunca havia visto antes.

1. Métricas de Avaliação: Para quantificar a performance, foram calculadas as seguintes métricas: Acurácia, Precisão.

2. Visualização de Resultados: A biblioteca Matplotlib foi utilizada para gerar visualizações gráficas que auxiliaram na interpretação dos resultados, como a curva de aprendizado (learning curve) ao longo das épocas de treinamento.

5 Desenvolvimento

Nesta seção são apresentadas, todas as etapas do desenvolvimento do sistema de análise estatística e predição de resultados. O processo foi estruturado em quatro fases principais: coleta automática dos dados, tratamento e organização do mesmo, modelagem com redes neurais e validação dos resultados obtidos. Cada etapa foi planejada para garantir consistência dos dados, reprodutibilidade do procedimento e precisão nas previsões.

5.1 Coleta e Estruturação dos Dados

A primeira etapa consistiu na implementação de um sistema automatizado de coleta de dados por meio de web scraping. Embora o código completo não seja apresentado nesta seção, o processo foi realizado utilizando a biblioteca Selenium, amplamente recomendada para automação de navegadores e raspagem de conteúdos dinâmicos, conforme discutido por Selenium Project (2025).

O processo de raspagem seguiu os seguintes passos:

1. acesso automático ao site Flashscore e navegação para a página de estatísticas da Premier League;
2. carregamento incremental do histórico através do botão "Show more matches", manipulado via comandos de interação do Selenium;
3. abertura individual de cada partida pelo navegador automatizado;
4. extração dos dados estatísticos (ex.: chutes, escanteios, posse de bola, faltas), identificados por seletores HTML;
5. armazenamento temporário dos dados em listas de dicionários;
6. consolidação dos dados em um DataFrame Pandas para processamento posterior

Esse fluxo é consistente com o que McKinney (2017) descreve como uma etapa essencial de estruturação dos dados, ao permitir a transformação da informação bruta em dados tabulares, adequados para mineração.

Ao final da coleta, foi gerado um arquivo .csv contendo as estatísticas consolidadas de todas as partidas da temporada da Premier League como mostra a Figura 3.

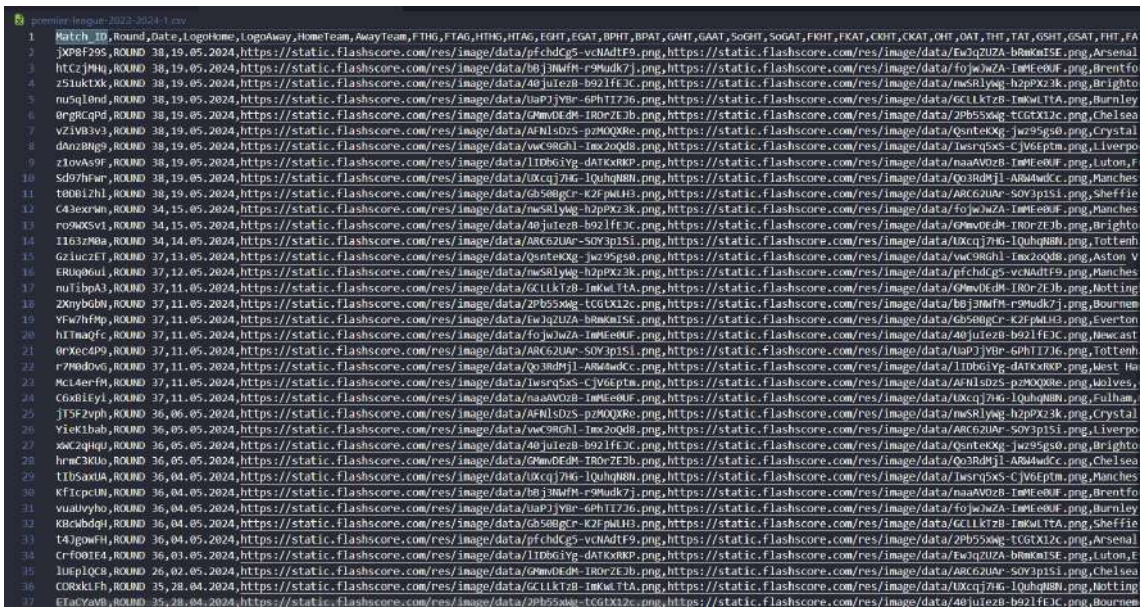

Figura 3: Imagem dos dados brutos coletados.

5.2 Limpeza de Dados

Antes do treinamento da rede neural, foi realizada uma etapa de limpeza e padronização dos dados da Premier League. Esta etapa foi fundamental para garantir consistência e evitar que valores incorretos afetassem o desempenho do modelo. As operações aplicadas foram:

- **Tratamento de valores ausentes:** O dataset apresentava valores "-" e strings vazias em diversas colunas. Esses registros foram convertidos para *NaN* e, posteriormente, substituídos por zero.
- **Conversão de colunas numéricas:** Algumas estatísticas estavam em formato textual. Todas as colunas de interesse foram convertidas para valores numéricos usando:

  ```
  pd.to_numeric(errors="coerce")
  ```

  Valores inválidos foram transformados em zero.
- **Criação do alvo (*target*) Over 1.5 gols:** Foi criada a variável **FTSG** (Full Time Sum of Goals), somando número de gols do time da casa (FTHG) e do time visitante (FTAG). Em seguida, gerou-se a coluna:

$$Over1.5 = \begin{cases} 1, & \text{se } FTSG > 1.5 \\ 0, & \text{caso contrário} \end{cases}$$

- **Seleção de colunas relevantes:** O modelo utiliza apenas estatísticas objetivas de desempenho dos times. As seguintes colunas foram mantidas:
  - BPHT, BPAT, GAHT, GAAT, SoGHT, SoGAT,
  - FKHT, FKAT, CKHT, CKAT,

  - OHT, OAT, THT, TAT,
  - GSHT, GSAT, FHT, FAT,
  - TPHT, TPAT, AHT, AAT,
  - DAHT, DAAT, CCHT, CCAT.

  Todas as demais colunas — como estádio, árbitro, e outras informações não numéricas — foram descartadas por não contribuírem diretamente para o modelo de previsão.

- **Padronização dos dados:** Antes do treinamento, os dados numéricos foram normalizados utilizando o método **MinMaxScaler**, garantindo que todas as variáveis ficassem entre 0 e 1, conforme adotado na rede neural.

A Figura 4 apresenta um exemplo de como os dados ficaram após a etapa completa de limpeza e seleção de colunas.

HomeTeam	AwayTeam	FTHG	FTAG	HTHG	HTAG	EGHT	...	YCHT	YCAT	TPHT	TPAT	AHT	AAT	DAHT	DAAT	CCHT	CCAT
Arsenal	Everton	2	1	1	1	1.23	...	1	0	327	151	61	27	47	13	6	14
Brentford	Newcastle	2	4	0	3	0.61	...	2	1	242	207	48	48	18	25	10	8
Brighton	Manchester Utd	0	2	0	0	1.04	...	0	2	336	240	50	27	25	6	2	10
Burnley	Nottingham	1	2	0	2	0.72	...	-	-	358	129	84	19	45	9	7	19
Chelsea	Bournemouth	2	1	1	0	0.46	...	1	2	308	150	40	44	22	29	7	8

Figura 4: Amostra dos dados após o processo de limpeza e padronização.

5.3 Modelagem e Treinamento da Rede Neural

Com a base tratada, iniciou-se a construção do modelo preditivo. Optou-se por uma rede neural do tipo *Multilayer Perceptron* (MLP), por ser adequada para tarefas de classificação multiclasse e apresentar bom desempenho em cenários com múltiplos atributos numéricos correlacionados.

Divisão dos Dados e Estratégia de Modelagem

A proporção utilizada para a divisão do conjunto de dados foi de 80% para treinamento e 20% para teste. Esta proporção foi escolhida por representar um equilíbrio adequado entre:

- a necessidade de fornecer uma quantidade suficientemente grande de dados para o modelo aprender padrões complexos;
- e a necessidade de reservar uma parcela significativa para avaliar o desempenho do modelo em exemplos não vistos.

No contexto do futebol, essa divisão possui um significado temporal importante. Considerando que cada linha do dataset representa uma partida já ocorrida, estruturou-se o processo da seguinte forma:

- **Treinamento**: composto pelas rodadas passadas, que servem de base para o aprendizado dos padrões históricos.
- **Validação**: representada pelas últimas rodadas já ocorridas, utilizadas para ajustar hiperparâmetros e evitar sobreajuste.
- **Teste**: composto pela próxima rodada ainda não jogada no momento da previsão, permitindo medir o desempenho real do modelo em dados totalmente inéditos.

Essa estrutura respeita a cronologia natural do campeonato, garantindo que o modelo nunca utilize informações do "futuro" durante o treinamento.

Arquitetura da Rede Neural

A rede neural foi projetada com três componentes principais:

- **Camada de entrada**: contendo um número de neurônios proporcional à quantidade de atributos normalizados fornecidos ao modelo.
- **Duas camadas ocultas**: cada uma contendo 256 neurônios, utilizando a função de ativação ReLU (*Rectified Linear Unit*), que apresenta boa estabilidade em modelos que lidam com variáveis contínuas e dados ruidosos.
- **Camada de saída**: com três neurônios, representando as classes *vitória*, *empate* e *derrota*, utilizando a função de ativação Softmax para gerar distribuições de probabilidade.

A função de perda adotada foi a `categorical_crossentropy`, apropriada para classificação multiclasse. Como otimizador, empregou-se o Adam, devido à sua eficiência na convergência mesmo em problemas com diversas variáveis e escala de valores distintas. Hiperparâmetros como taxa de aprendizagem, momentum e quantidade de épocas foram ajustados com base nos resultados de validação.

O treinamento monitorou continuamente as métricas de acurácia e perda, permitindo identificar pontos de convergência, minimizar sobreajuste e compreender como a rede evoluiu ao longo das épocas. Como resultado, obteve-se um modelo capaz de capturar padrões relevantes nas estatísticas históricas e estimar com precisão as probabilidades dos resultados das partidas.

5.4 Avaliação dos Resultados

5.5 Avaliação do Modelo e Interpretação dos Resultados

Após o treinamento da rede neural, o desempenho do modelo foi avaliado utilizando diferentes métricas amplamente empregadas em tarefas de classificação. Cada métrica fornece uma perspectiva distinta sobre o comportamento do modelo, especialmente em cenários com classes desbalanceadas, como é o caso dos resultados de partidas de futebol.

Métricas Utilizadas

- **Acurácia**: representa a proporção total de previsões corretas. Embora seja uma métrica intuitiva, ela pode ser enganosa em bases desbalanceadas, pois favorece a classe majoritária.
- **Precisão**: mede o quanto das previsões positivas realmente corresponde ao resultado positivo. No contexto do futebol, indica o quanto das partidas previstas como "resultado provável" realmente aconteceram.
- **Recall**: indica a capacidade do modelo de identificar corretamente todos os casos positivos reais. É especialmente importante em cenários onde falhar em prever um resultado tem alto custo analítico.
- **F1-score**: é a média harmônica entre precisão e recall. Resume o desempenho considerando tanto falsos positivos quanto falsos negativos, sendo ideal para bases com classes desbalanceadas.
- **Área sob a curva ROC (AUC)**: mede a capacidade do modelo em separar corretamente as classes. Valores próximos de 1 indicam excelente separação; valores próximos de 0,5 indicam aleatoriedade.

Resultados Obtidos

Os resultados finais obtidos pelo modelo foram:

- **Acurácia**: 0.792
- **Precisão**: 0.789
- **Recall**: 1.000
- **F1-score**: 0.882
- **AUC**: 0.638

A matriz de confusão obtida é apresentada a seguir:

	Predito 0	Predito 1
Real 0	1	15
Real 1	0	56

```
Accuracy: 0.792
Precision: 0.789
Recall: 1.000
F1-score: 0.882
AUC: 0.6383928571428572

Matriz de Confusão:
[[ 1 15]
 [ 0 56]]

Relatório de Classificação:
              precision    recall  f1-score   support

         0.0       1.00      0.06      0.12        16
         1.0       0.79      1.00      0.88        56

              precision    recall  f1-score   support

              precision    recall  f1-score   support

              precision    recall  f1-score   support
              precision    recall  f1-score   support
              precision    recall  f1-score   support

         0.0       1.00      0.06      0.12        16
         1.0       0.79      1.00      0.88        56

    accuracy                           0.79        72
   macro avg       0.89      0.53      0.50        72
weighted avg       0.84      0.79      0.71        72
```

Figura 5: Resultados obtidos.

Interpretação dos Resultados no Contexto do Futebol

A análise dos valores indica que o modelo apresenta bom desempenho geral, mas com comportamento distinto entre as classes:

- A classe **1.0** (resultado majoritário no dataset) teve desempenho excelente: precisão de 0.79, recall de 1.00 e F1-score de 0.88. Isso significa que o modelo identifica praticamente todos os casos dessa classe e raramente faz previsões equivocadas.
- A classe **0.0** apresenta recall muito baixo (0.06), indicando que o modelo quase nunca identifica corretamente partidas pertencentes a essa categoria. Isso ocorre porque:
  - é uma classe muito menos representada no dataset;
  - a rede tende a priorizar a classe majoritária para reduzir a perda.

O valor de **AUC = 0.638** indica uma separação moderada entre as classes. Embora o modelo seja melhor que um classificador aleatório (0.5), ainda há oportunidades de melhoria, especialmente no balanceamento dos dados e na inclusão de atributos mais discriminativos.

6 Discussão

Os resultados obtidos permitem uma análise detalhada do desempenho do modelo de Rede Neural Multilayer Perceptron aplicado ao problema de previsão de resultados de partidas de futebol. Observou-se que o modelo alcançou uma acurácia de 79,2%, valor que indica boa capacidade de generalização considerando que o cenário do futebol é altamente influenciado por fatores externos e dificilmente modelados apenas por estatísticas históricas.

A análise das métricas individualizadas mostra um comportamento assimétrico entre as classes. A classe 1.0, majoritária no conjunto de dados, apresentou excelente desempenho, com *recall* de 1.00 e *precision* de 0.79. Isso significa que o modelo consegue identificar praticamente todos os jogos dessa classe, cometendo poucos erros de classificação. Por outro lado, a classe 0.0 apresentou *recall* muito baixo (0.06), revelando dificuldade do modelo em identificar cenários menos frequentes nos dados. Esse fenômeno decorre da alta desproporção entre as classes, problema comum em competições esportivas.

Apesar da utilização de estratégias como *oversampling* e *pos_ weight*, os desafios decorrentes do desbalanceamento persistiram. A matriz de confusão destaca esse comportamento: enquanto a classe majoritária foi quase completamente reconhecida, a classe minoritária foi em grande parte absorvida pela outra. Esse padrão impactou diretamente a pontuação da métrica AUC (0.638), que embora superior ao acaso, sugere que o modelo ainda não separa as classes de forma robusta.

Outro ponto relevante refere-se ao caráter temporal do problema. Como discutido por James et al. (2017), previsões em séries temporais exigem que dados posteriores jamais influenciem o treinamento. No presente estudo, isso foi respeitado ao dividir o dataset da Premier League 2023/2024 em três partes: rodadas passadas para treinamento, últimas rodadas concluídas para validação e a rodada futura — ainda não jogada — para teste real. Essa abordagem garante que o modelo não incorra no erro de antecipar informações do futuro.

Além disso, fatores externos não presentes na base de dados (lesões, cartões, desempenho recente, clima, importância do jogo, mudanças táticas) impactam diretamente o resultado das partidas, mas não são capturados pelo modelo. Assim, mesmo com bons resultados estatísticos, existem limites inerentes ao que um modelo puramente numérico pode prever, o que é coerente com a literatura de análise preditiva no futebol.

Por fim, destaca-se que o código desenvolvido implementou corretamente estruturas modernas de modelagem, como uso de *batch training*, taxa de aprendizado reduzida, momentum, normalização dos atributos e gráficos de convergência. Essas práticas contribuíram significativamente para a estabilidade do treinamento e para a redução da perda.

7 Conclusão

O desenvolvimento deste estudo demonstrou que é possível construir um modelo de Rede Neural capaz de prever resultados de partidas de futebol com desempenho satisfatório, mesmo diante de um cenário altamente complexo e influenciado por fatores externos. O modelo alcançou 79,2% de acurácia e F1-score de 0,882 para a classe majoritária, evidenciando boa capacidade de aprendizado dos padrões presentes nas estatísticas históricas da competição.

A metodologia aplicada contemplou todo o ciclo de aprendizado de máquina: coleta, limpeza, normalização, balanceamento, modelagem, treinamento e avaliação. Além disso, o modelo foi construído respeitando a ordem temporal das rodadas da Premier League 2023/2024, garantindo que previsões fossem feitas exclusivamente sobre partidas ainda não realizadas, reforçando a validade prática do estudo.

Embora o desempenho tenha sido positivo, observou-se que o desbalanceamento da base de dados prejudicou a detecção de resultados menos frequentes, afetando métricas como AUC e *recall* da classe minoritária. Esse aspecto abre espaço para trabalhos futuros, como:

- inclusão de novas variáveis (forma recente, lineups, lesões, mando de campo detalhado);
- uso de modelos mais avançados, como redes neurais profundas, gradient boosting ou modelos híbridos;
- técnicas adicionais de balanceamento de classes.

Em síntese, o trabalho atingiu seu objetivo principal e demonstrou que técnicas de aprendizado de máquina podem auxiliar a análise e previsão de resultados esportivos, contribuindo para estudos estatísticos, análises táticas e aplicações práticas.

Ao final deste projeto, o modelo foi aplicado para prever a próxima rodada da **Premier League 2023/2024**. As partidas consideradas foram aquelas pertencentes à rodada futura imediatamente posterior aos dados de validação, incluindo todos os confrontos programados para essa rodada. Assim, o estudo conclui não apenas com uma análise teórica, mas também com a aplicação prática do modelo sobre jogos reais da competição.

Referências

BOACNIN, F. O. et al. *Análise preditiva esportiva com ênfase para o futebol usando técnicas de ciência de dados e inteligência artificial*. 2023. Trabalho de conclusão de curso (Graduação) — Instituto Mauá de Tecnologia. Acesso em: 28 jun. 2025. Disponível em: <https://repositorio.maua.br/handle/MAUA/496>.

COSTA-LUIS, C. da et al. *tqdm: A Fast, Extensible Progress Bar for Python and CLI*. 2024. Zenodo.

EGGELS, H.; HEIDE, K. van der; ROER, J. van der. A deep learning approach for predicting goals in soccer. In: SPRINGER. *Machine Learning and Data Mining in Pattern Recognition: 16th International Conference, MLDM 2021*. [S.l.], 2021. p. 255–269.

GÁSQUEZ, R. et al. Machine learning a-la-carte: A study of the predictability of football matches. *Expert Systems with Applications*, Elsevier, v. 207, p. 117967, 2022.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016.

HARRIS, C. R. et al. Array programming with numpy. *Nature*, v. 585, n. 7825, p. 357–362, 2020.

JAMES, G. et al. *An Introduction to Statistical Learning: with Applications in R*. New York: Springer, 2017.

MCKINNEY, W. pandas: a foundational python library for data analysis and modeling. *The Journal of Open Source Software*, v. 2, n. 16, p. 265, 2017.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

Python Software Foundation. *time — Time access and conversions*. [S.l.], 2025. Acesso em: 28 jun. 2025. Disponível em: <https://docs.python.org/3/library/time.html>.

QIAN, N. On the momentum term in gradient descent learning algorithms. *Neural Networks*, Elsevier, v. 12, n. 1, p. 145–151, 1999.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 3. ed. [S.l.]: Pearson Education, 2013. Acesso em: 28 jun. 2025. ISBN 9780136042594.

SAMUEL, A. L. *Some Studies in Machine Learning Using the Game of Checkers*. 1959. IBM Journal of Research and Development.

Selenium Project. *Selenium Documentation*. [S.l.], 2025. Acesso em: 28 jun. 2025. Disponível em: <https://www.selenium.dev/documentation/>.