Previsão de resultados futebolísticos com *machine-learning*

Football results prediction using machine-learning

Leonardo Benvinda
Instituto Politécnico de Beja
Escola de Tecnologia e Gestão
Beja, Portugal
18792@stu.ipbeja.pt

Resumo — **Este trabalho explora a implementação de modelos de *machine-learning* na previsão de resultados desportivos, especificamente jogos de futebol, no software Orange.**

Palavras-Chave – futebol; previsões; apostas; machine-learning.

Abstract — **This document explores the implementation of machine-learning in football results predictions using Orange.**

Keywords – football; soccer; predictions; bets; machine-learning.

I. Introdução

O futebol é o desporto mais popular do mundo [1]. Com esta popularidade, é um desporto que atrai muita atenção de outra atividade lúdica muito popular: apostas desportivas.

As apostas desportivas são um mercado que movimenta grandes volumes de dinheiro: em 2020 estima-se que tenha movimentado entre 60 e 73 mil milhões de dólares apenas nos Estados Unidos da América [2]. Num mercado tão arriscado como este, para garantir que as empresas que investem neste mercado são sustentáveis, implementam-se mecanismos de equilíbrio de probabilidades para garantir que, independentemente do resultado desportivo, a empresa consiga sempre lucrar:

"As casas de apostam começam por definir uma margem de lucro. […]. Outro exemplo: imaginemos que fazemos uma aposta Acima/Abaixo, onde ambos os desfechos da aposta têm 1.91 em *odds* decimais. Se um apostador apostar $100 Acima e outro $100 Abaixo, o apostador que ganhar recebe $191 enquanto outro perde $100. Isto significa que a casa de apostas recebeu $200 em apostas, mas apenas pagou $191, ficando com $9 de lucro. Neste caso, se o montante apostado por todos os apostadores for dividido entre ambos os resultados possíveis, seja qual for o resultado, as casas de apostas recebem sempre uma parte do montante apostado." – tradução livre de um trecho do artigo [3].

Para além da fixação de uma margem de lucro mínima, casas de apostas combinam previsões baseadas em dados estatísticos, que se assume como probabilidades reais, que são depois alteradas para refletir as tendências e opiniões de especialistas e do publico em geral sobre cada evento específico:

"A probabilidade não é calculada apenas em dados estatísticos e objetivos. Muitas casas de apostas também tomam em consideração fatores subjetivos, como opiniões dos próprios especialistas, opiniões de outras casas de apostas e a opinião geral do público. Isto ajuda as casas de apostas a perceber melhor o potencial comportamento dos apostadores, podendo ajustar as probabilidades de maneira a maximizar os lucros" - tradução livre de um trecho do artigo [3].

Para entender melhor os diferentes tipos de apostas, consultar o artigo [4] e para entender as principais representações de *odds,* consultar o artigo [5].

Esta manipulação, embora que necessária para garantir a sustentabilidade deste tipo de atividade, leva a que as probabilidades reais sejam mal representadas, levando os apostadores a assumir probabilidades erradas para os diferentes desfechos possíveis de um evento desportivo.

Neste artigo, irei treinar um modelo *machine-learning* com dados estatísticos para encontrar as probabilidades reais associadas a eventos desportivos, mais especificamente jogos de futebol, tentando, assim, equilibrar as probabilidades a favor do apostador.

II. Metodologia

Para desenvolver este trabalho, decidi usar a metodologia KDD (*Knowledge Discovery Database*) [6].

A. Processo ETL

O *dataset* que escolhi foi retirado do sítio web Kaggle [7], contendo dados retirados do sítio web Transfermarkt [8]. Este *dataset* contém informação sobre jogos, equipas e jogadores das principais competições europeias.

Dos vários ficheiros .csv disponibilizados no dataset, utilizei apenas o ficheiro "games.csv" contento jogos das épocas desportivas de 2012 até ao início de 2024.

- Limpeza dos dados

Deste ficheiro, comecei por eliminar várias colunas de dados meramente informativas que em nada influenciariam o modelo, como nome do estádio, o nome do treinador de cada equipa, etc. De seguida, apaguei todas as entradas nas quais houvesse uma coluna relevante com valores nulos, eliminando assim possíveis influências negativas que pudessem ter no modelo.

- Cálculo de variáveis

Com os dados já limpos, utilizei os dados presentes para calcular novas colunas que serão posteriormente usadas como variável-alvo e variáveis de análise:

- Variável-Alvo: Resultado do jogo:

Para calcular o resultado do jogo, utilizei as colunas referentes aos golos marcados por cada equipa no jogo em análise, para determinar se a equipa vencedora foi a equipa da casa, a equipa visitante, ou um empate.

- Variáveis de análise:

A escolha de variáveis de análise para um tema como resultados desportivos será sempre algo subjetiva, porque há dados que serão baseados em opiniões pessoais. Por exemplo, uma das mais referidas variáveis de análise é o desempenho individual dos jogadores de uma equipa em jogos passados. Isto é algo altamente subjetivo pois a apreciação e classificação de uma exibição irá depender de opiniões e gostos pessoais de quem avalia. Para evitar este tipo de subjetividades, decidi utilizar varáveis que fossem objetivas e factuais. As variáveis que escolhi baseiam-se nos 5 jogos anteriores de cada equipa individualmente e nos 5 jogos entre as duas equipas em questão:

- Vitórias;
- Empates;
- Derrotas.

São variáveis talvez um pouco simplistas, mas são dados factuais e objetivos que refletem a forma de cada equipa ao chegar ao confronto em questão.

Todo este processo foi efetuado através de código pyhton disponível em apêndice A.

- Identificação e eliminação de *outliers*

Um dos pontos importantes num bom processo ETL é a identificação e eliminação de *outliers,* mas atendendo à natureza dos dados, valores extremos também são necessários para conseguir classificar com algum realismo resultados desportivos. Peguemos num exemplo: A equipa A tem um registo 100% vitorioso, quer nos seus últimos 5 jogos, quer nos últimos 5 confrontos com a equipa B. A equipa B, por sua vez, tem apenas derrotas, tanto nos seus últimos 5 jogos, como nos últimos 5 confrontos com a equipa A. Um resultado que não seja uma vitória para a equipa A será sempre extremamente anormal, estatisticamente falando, mas, como se costuma dizer, "a bola é redonda", e o futebol tem-nos habituado a resultados muito surpreendentes, mostrando que qualquer resultado é possível, por mais improvável que seja.

- Normalização dos dados

A normalização dos dados também é muito importante, mas foi implementada já no software Orange. Optei por normalizar os dados nessa fase posterior pois considerei que poderia ser importante visualizar os dados originais e os dados normalizados, para uma compreensão mais fácil dos resultados.

Os dados foram normalizados através do *widget "preprocess"*, isto permite-me adicionar um visualizador "data table" antes e depois do processamento, podendo assim ver os dados normalizados e não normalizados.

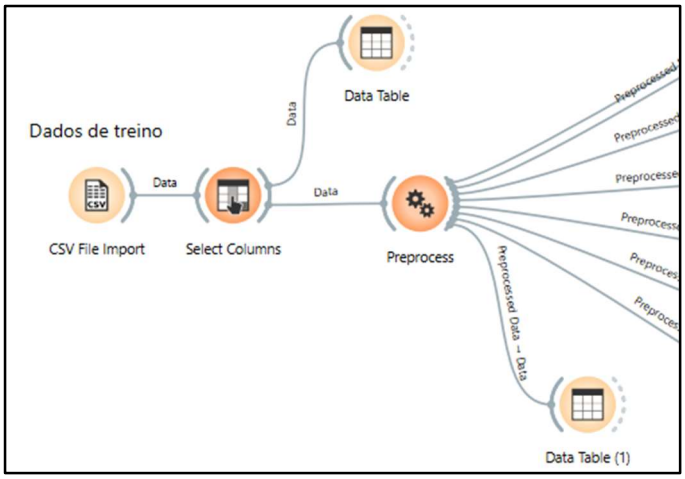

Figure 1 - fluxo de trabalho Orange referente à normalizaão de dados de treino

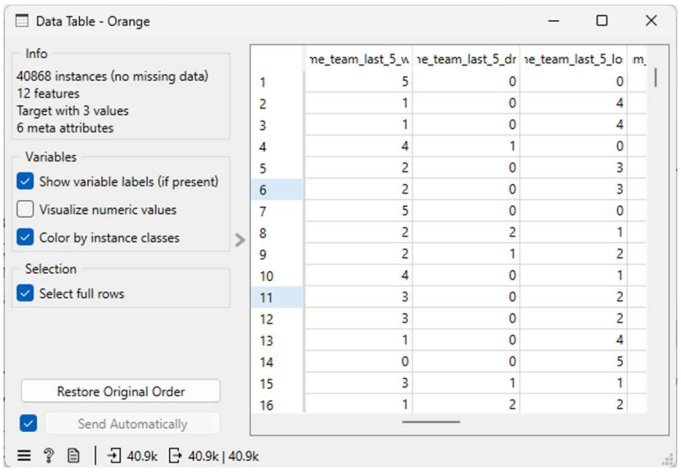

Figure 2 - Dados de treino não normalizados (Data Table)

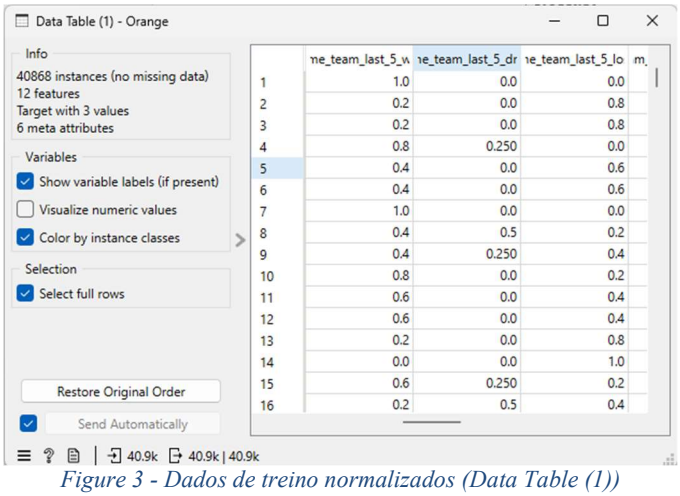

Figure 3 - Dados de treino normalizados (Data Table (1))

B. *Modelos escolhidos*

Um jogo de futebol tem 3 resultados possíveis: vitória caseira (*home_win*), empate (*draw*) e vitória visitante (*away_win*), o que faz deste um problema de classificação.

Sendo assim, escolhi os modelos Floresta Aleatória, Regressão Logistíca, kNN, Gradient Bosting e Rede Neural, por serem os mais adequados a este tipo de problema [9], [10].

Passarei então a explicar sucintamente cada um dos algoritmos utilizados:

- Random Forest

O modelo random forest é um algoritmo do tipo *ensemble[11],* que cria várias árvores de decisão, treinadas com um subconjunto aleatório dos dados de treino. Chega-se à decisão final de uma previsão através da contagem das previsões de cada árvore, no caso de uma variável discreta, ou da média do valor previsto de cada árvore, no caso de uma variável contínua [12].

- Regressão Logística

O nome pode ser enganador, pois sugere a resolução de problemas de regressão. Na verdade, o modelo de Regressão Logística [13] é um poderoso classificador. Pode ser implementado de duas formas:

  - Regressão Logística Binomial

Esta variante é utilizada para classificar dados de forma binária, apenas havendo duas classificações possíveis. Utilizando o tema deste artigo, poderia prever apenas se uma equipa ganhou ou não ganhou. Isto não seria assim tão útil pois pretendo que sejam previstos todos os 3 resultados possíveis. Isso é possível utilizando a Regressão Logistíca Multinominal, explicada já de seguida, pelo que não entrarei em mais detalhe sobre a versão binomial. Para uma melhor compreensão consultar os artigos [13] e [14].

  - Regressão Logística Multinomial

Mais versátil, a Regressão Logística Multinomial [14] adapta-se perfeitamente ao problema do artigo, podendo prever variáveis com 3 ou mais desfechos possíveis. Para traduzir os valores em probabilidades, este algoritmo utiliza a função softmax [14], transformando as previsões do modelo em probabilidades que somam 1, permitindo assim a classificação em múltiplas categorias. A função softmax é dada como:

$$P(y = c|x) = \frac{e^{z_x}}{\sum_{j=1}^{C} e^{z_j}} \qquad (1)$$

Onde:
x: uma entrada de dados;
c: uma classe da variável-alvo;
P (y = c|x): Probabilidade da entrada x pertencer à classe c;
z_c: Combinação linear das variáveis para a classe c;
C: Número total de classes.

- kNN (K-Nearest Neighbors)

O modelo kNN [15] é um algoritmo utilizado tanto para classificação como regressão. Baseia-se no conceito de proximidade, classificando os dados de acordo com as classes dos **k** vizinhos mais próximos, ou fazendo a média dos seus valores, no caso de uma variável discreta.

Para calcular a distância entre a entrada a classificar e os seus vizinhos mais próximos, é costume utilizar métricas como a **distância Euclidiana** definida por:

$$d(p, q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \qquad (2)$$

Onde p e q são dois pontos no espaço n-dimensional.

O kNN é um algoritmo sem treino, o que significa que o processamento é todo feito na fase de previsão, o que o torna ideal para *datasets* de dimensão reduzida, mas pouco eficiente para grandes volumes de dados, tendo de fazer muitos cálculos para cada previsão.

- Gradient Boosting

À semelhança do kNN, o ***Gradient Boosting*** [16] também pode ser utilizado tanto para classificações, como regressões. É também um algoritmo *ensemble,* como o modelo Floresta Aleatória.

Este modelo começa por implementar um modelo simples, como, por exemplo, uma previsão de probabilidade para a classificação de uma entrada. Após esta previsão, é calculado o erro residual através da diferença entre as previsões atuais e os valores reais. Este erro servirá para indicar ao próximo modelo o que terá de corrigir.

Ao treinar o modelo seguinte, dá-se prioridade aos exemplos com maior erro residual, focando-se nas entradas mais mal classificadas.

O modelo recém treinado é então adicionado em série ao conjunto de modelos implementados, caso a taxa de aprendizagem (η) se verifique significativa. Esta taxa de aprendizagem pondera o impacto do novo modelo em relação ao anterior.

O processo de calculo de erro, treino de novos algoritmos e adição dos mesmos ao modelo pode ser repetida até se atingir um número desejado de iterações.

A principal formula deste modelo é a seguinte:

$$F_m(x) = F_{m-1}(x) - \eta . h_m(x) \qquad (3)$$

Onde:

$F_m(x)$: é a previsão final após m iterações;
$F_{m-1}(x)$: é a previsão anterior;
η: Taxa de aprendizagem;
$h_m(x)$: Modelo ajustado para corrigir os erros atuais.

O Gradient Boosting é então um algoritmo muito preciso e flexível devido à sua atenção especial a entradas mal

classificadas ou calculadas. Isto vem com alguma exigência computacional, especialmente em grandes conjuntos de dados.

- Neural Network

O algoritmo Rede Neural [17] é inspirado no comportamento de um cérebro, formando várias camadas de unidades de processamento, convenientemente chamadas de neurónios artificiais, que transformam os dados de entrada em dados de saída através de uma série de cálculos e operações matemáticas.

A primeira camada de neurónios é a **camada de entrada**, que recebe os dados a serem processados. A **camada de saída** é a última, responsável por gerar os dados de saída, como uma classificação ou regressão. Entre estas, situam-se **camadas de neurónios ocultas,** que transformam os dados através de variadas operações.

C. Métricas de avaliação

Para avaliar a eficácia de modelos *machine-learning,* existem várias métricas a que se pode recorrer para ganhar uma melhor perspetiva sobre o seu desempenho [18][21]. Identifiquei as métricas Acurácia (CA), F1-Score, Precisão e Recall, que passo a explicar mais detalhadamente:

- Acurácia (CA)

A acurácia é a proporção de previsões corretas feitas pelo modelo em relação ao número total de previsões. Noutras palavras, é a medida de quantas previsões corretas o modelo teve em relação ao total de previsões feitas. É obtida através da seguinte formula:

$$Acurácia = \frac{TP+T}{TP+TN+FP+F} \qquad (4)$$

Onde:

TP = Verdadeiros Positivos;
TN = Verdadeiros Negativos;
FP = Falsos Positivos;
FN = Falsos Negativos.

Se o modelo conseguir prever corretamente 80% das vitórias caseiras e 70% das vitórias visitantes, a acurácia seria a soma dos acertos das previsões em relação ao total de jogos. No entanto, a acurácia pode ser enganadora se as classes não forem equilibradas, como em competições onde a vitória caseira ocorre mais frequentemente [19].

- F1-Score

O F1-Score é a média harmónica entre precisão e recall. É útil quando as classes não são equilibradas, pois pondera tanto os falsos positivos, quanto os falsos negativos [19]. É dado pela função:

$$F1 = 2 . \frac{Precisão . Recall}{Precisão + Recall} \qquad (5)$$

Ao prever uma vitória caseira, pode ser importante considerar o equilíbrio entre acertar o maior número de vitórias caseiras (precisão) e garantir que o modelo não deixe de prever vitórias caseiras quando elas acontecem (recall). O F1-Score é útil para garantir que o modelo tem um bom desempenho tanto em termos de precisão quanto de recall.

- Precisão

A precisão mede a proporção de previsões positivas corretas em relação ao total de previsões positivas feitas. Ou seja, de todas as vezes que o modelo previu um evento positivo (como uma vitória caseira), qual a percentagem de vezes que isso foi realmente correto [20]. Obtida com a função:

$$Precisão = \frac{TP}{TP+F} \qquad (6)$$

Se o modelo prever 10 vitórias caseiras, mas apenas 7 delas forem realmente vitórias caseiras, a precisão seria 0.7 (70%). Isso significa que o modelo tem uma taxa de erro de 30% nas previsões de vitórias caseiras.

- Recall

O recall mede a capacidade do modelo de identificar todas as ocorrências positivas. Em outras palavras, quantos dos eventos realmente positivos (como vitórias caseiras) o modelo foi capaz de prever corretamente [20]. Dada pela função:

$$Recall = \frac{TP}{TP+FN} \qquad (7)$$

Se em 10 jogos com vitórias caseiras, o modelo só previu 7 vitórias caseiras, o recall seria 0.7 (70%). Isso significa que 30% das vitórias caseiras não foram previstas pelo modelo.

D. Ferramentas utilizadas

Na implementação dos modelos de *machine-learning* utilizei duas principais ferramentas:

- Código Python

Utilizei alguns códigos python para o tratamento e separação dos dados. Escolhi esta linguagem por ser altamente versátil, bem documentada e com uma comunidade muito ativa onde é fácil encontrar exemplos de implementação de código.

- Orange

Considerei o **Orange** uma excelente escolha para a implementação dos modelos de previsão de resultados. Devido à sua **interface gráfica intuitiva**, **facilidade de implementação de modelos**, **ferramentas de pré-processamento de dados**, **avaliação de métricas de desempenho** e **extensibilidade**, torna-se uma ferramenta poderosíssima. Além disso, ao ser uma ferramenta visual, permite uma rápida adaptação e iteração, o que acelera o desenvolvimento do modelo e a análise de resultados [22].

III. Aplicação dos Modelos

Neste capítulo irei explicar como apliquei os modelos de machine-learning, começando por explicar a separação dos dados, seguido do fluxo de trabalho no Orange.

A. Divisão de dados

Com os dados limpos e as variáveis necessárias já calculadas fiz a divisão dos dados. Tendo em conta que o tema tratado são jogos de futebol e as previsões são baseadas em dados de confrontos anteriores, achei que faria sentido dividir os dados por épocas desportivas, utilizando para treino jogos de 2013 a 2021, para teste 2022 e 2023, e, para simular a introdução de dados novos, dados de 2024. Esta divisão foi feita também através de código python em apêndice B.

B. Fluxo de trabalho

O fluxo de trabalho começa com a importação dos ficheiros .csv gerados para treino, teste e previsões. Segue uma seleção de colunas, escolhendo quais serão meramente informativas (*metas*) usadas para identificar o jogo em questão, qual será a variável-alvo (*target*), quais serão as variáveis com peso nas previsões (*features*) e quais serão ignoradas (*ignored*).

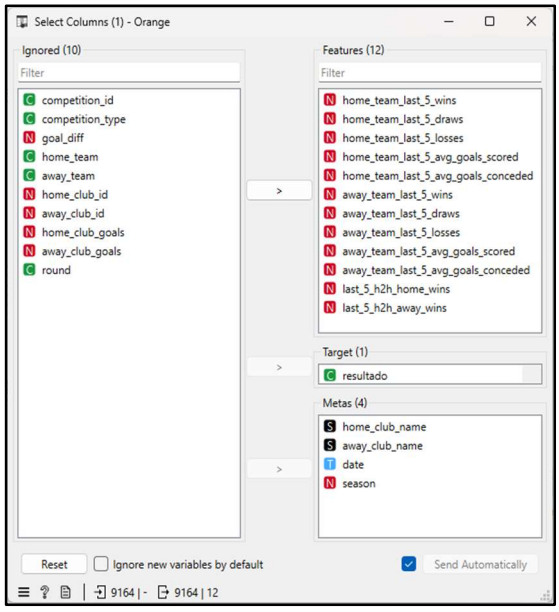

Figure 4 - Seleção de colunas

Ligado ao widget de seleção de colunas, estará o pré-processador de dados, que utilizei para fazer a normalização das variáveis, como referido na secção sobre normalização de dados. Esta tranche do fluxo Orange pode ser toda observada na Figura 1 e é repetido para a importação dos 3 ficheiros de dados utilizados neste fluxo Orange.

Com os dados importados e devidamente processados, são utilizados para alimentar os restantes widgets. O ficheiro de dados de treino alimentará o widget '*Test and Score'*, responsável por testar e avaliar os modelos, bem como os próprios modelos de machine-learning. Os dados de teste alimentarão também o widget '*Test and Score'*. Finalmente, os dados para previsão serão utilizados para alimentar o *widget* '*Predictions*', que será alimentado também com os modelos já treinados e mostrará as previsões feitas pelos diferentes modelos. Para uma melhor visualização de resultados, os *widgets* '*Test and Score'* e '*Predictions*' foram ligados a um *widget* que gera uma matriz de confusão.

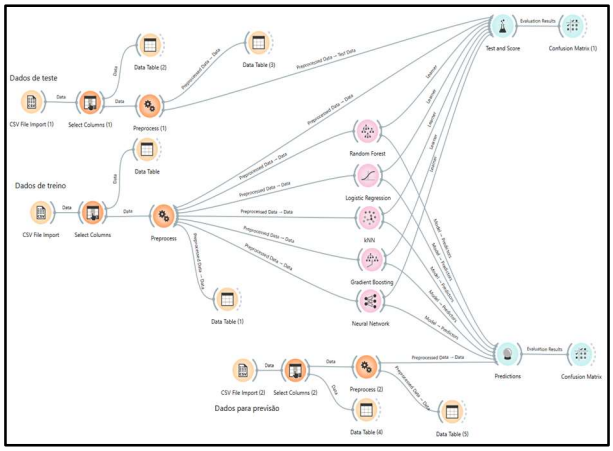

Figure 5 - Fluxo Orange

IV. Resultados

Após os modelos de *machine-learning* serem treinados, e testados, recorri às próprias métricas geradas pelo Orange para avaliar os resultados. Ainda no Orange utilizei também matrizes de confusão para uma melhor visualização das estatísticas. Neste capítulo demonstro os resultados com os recursos referidos e interpreto os mesmos.

Evaluation results for target (None, show average over classes)

Model	CA	F1	Prec	Recall
Logistic Regression	0.509	0.468	0.504	0.509
Random Forest	0.478	0.469	0.465	0.478
kNN	0.457	0.454	0.457	0.457
Gradient Boosting	0.522	0.483	0.523	0.522
Neural Network	0.513	0.482	0.505	0.513

Figure 6 - Valores das métricas para a média entre classes

- Acurácia (CA)

Gradient Boosting apresenta a maior acurácia (**52.2%**), seguido pela **Rede Neural** (**51.3%**) e **Regressão Logística** (**50.9%**). Estes valores são relativamente baixos, dado que um modelo aleatório num problema de classificação de três classes teria uma acurácia esperada de **33.3%**. Apesar disso, superar a base aleatória indica que os modelos estão a conseguir encontrar algum tipo de padrão.

- F1-Score

O **Gradient Boosting** também lidera o F1-Score (**48.3%**), com a **Rede Neural** e a **Regressão Logística** em segundo lugar (**48.2% e 46.8%**, respectivamente). O F1-Score equilibra a precisão e o recall, sendo útil em problemas de classes desequilibradas. Os valores em torno de **48%** sugerem que os modelos têm dificuldade em prever todas as classes com alta consistência.

- Precisão (Precision)

O **Gradient Boosting** possui a maior precisão (**52.3%**), indicando que consegue previsões corretas mais frequentemente do que outros modelos. A **Regressão Logística** e a **Rede Neural** também se destacam, com valores de **50.4% e 50.5%**, respetivamente.

- Recall

O **Gradient Boosting** e a **Rede Neural** apresentam os maiores valores de recall (**52.2% e 51.3%**, respectivamente), sugerindo que esses modelos têm maior capacidade de identificar corretamente as instâncias das classes reais. A **Regressão Logística** também é competitiva com **50.9%**.

- Avaliação Geral

O **Gradient Boosting** destaca-se como o modelo com melhor desempenho geral nas métricas analisadas (**CA, F1, Precision e Recall**). Isso indica que oferece um equilíbrio mais robusto entre previsões corretas, previsões positivas corretas e a capacidade de detetar instâncias reais.

A **Rede Neural** e a **Regressão Logística** são alternativas competitivas, mas ligeiramente inferiores.

Random Forest e **kNN** têm desempenho significativamente mais baixo em todas as métricas, sugerindo que não são os melhores para este conjunto de dados.

Seguem as matrizes de confusão para uma melhor visualização das previsões feitas pelos modelos durante os testes.

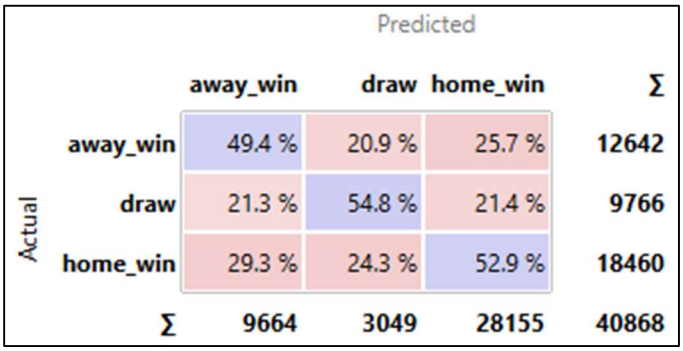

Figure 7 - Martiz de Confusão 'Test and Score' do modelo Gradient Boosting.

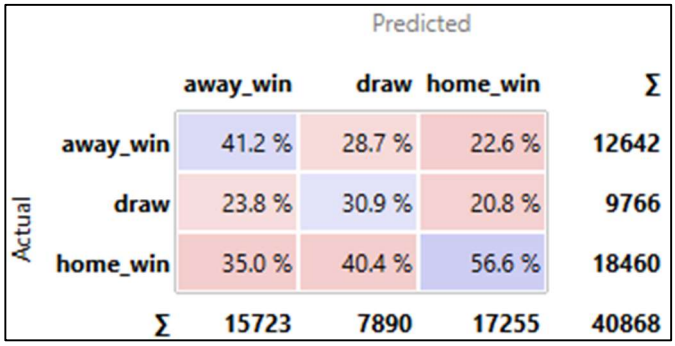

Figure 8 - Martiz de Confusão 'Test and Score' do modelo kNN

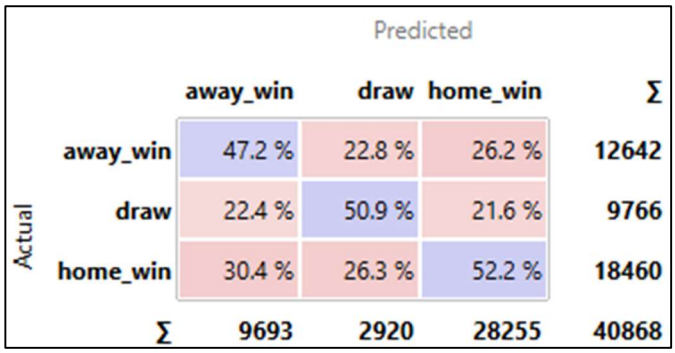

Figure 9 - Martiz de Confusão 'Test and Score' do modelo Regressão Logística

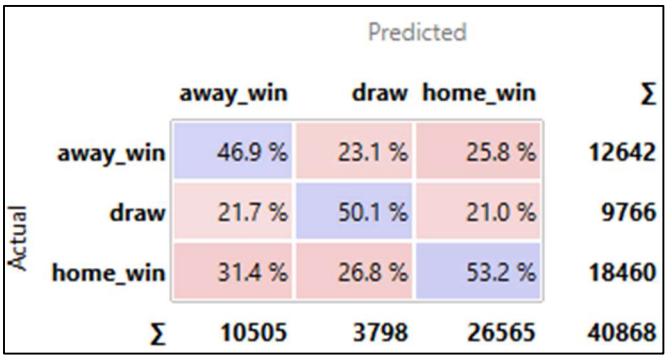

Figure 10 - Martiz de Confusão 'Test and Score' do modelo Rede Neural

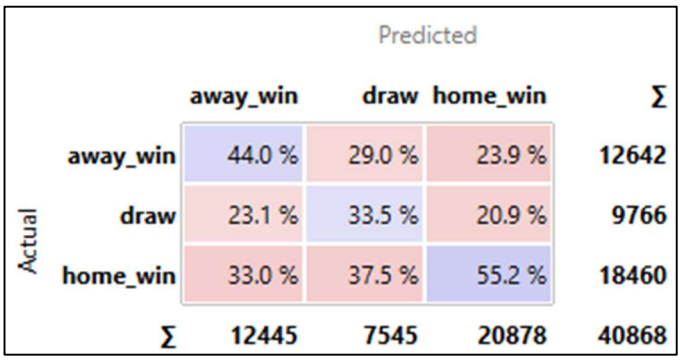

Figure 11 - Martiz de Confusão 'Test and Score' do modelo Random Forest

V. Análise a Trabalhos semelhantes

Neste capítulo irei comparar o trabalho realizado neste artigo com outro por mim identificado que considero ser uma boa comparação, não só pelas semelhanças, mas também pelas divergências.

A. *"Using Bookmaker Odds to Predict the Final Result of Football Matches" [23]*

Este estudo utiliza as odds fornecidas por casas de apostas como principal fonte de dados para prever resultados de futebol. Apesar de uma abordagem diferente, algumas comparações podem ser feitas:

- Fonte de dados

O trabalho baseado em odds utiliza as probabilidades já calculadas pelas casas de apostas, o que significa que incorpora subjetividades humanas e estatísticas implícitas. Por outro lado, o meu trabalho constrói modelos baseados exclusivamente em dados históricos objetivos, como vitórias, empates e derrotas nos últimos jogos.

- Modelos

O estudo utiliza regressões logísticas e métodos probabilísticos diretos. O este trabalho, além de incluir a regressão logística, explora algoritmos o como Gradient Boosting e Redes Neurais.

- Resultados

O estudo de odds reporta uma acurácia em torno de **53%–58%**, mas destaca que é difícil superar as odds das casas de apostas devido ao fator de manipulação das probabilidades. O meu trabalho apresenta resultados ligeiramente inferiores, mas é mais transparente em relação à construção e manipulação dos dados, sendo uma abordagem mais educacional.

- Objetivo

O estudo visa auxiliar na identificação de apostas lucrativas, enquanto o meu trabalho procura um modelo equilibrado que ajude a prever probabilidades reais para desfechos desportivos.

VI. Conclusão

Este trabalho demonstrou o uso de machine learning para prever resultados de jogos de futebol utilizando um fluxo implementado no software **Orange**. Comparado a trabalhos semelhantes, destaca-se pela utilização de variáveis objetivas e factuais, o que facilita a replicabilidade e elimina subjetividades. O **Gradient Boosting** mostrou-se o modelo mais robusto, com uma acurácia de **52.2%**, superando a base aleatória de **33.3%**, mas ficando ligeiramente atrás de outros estudos que incluíram variáveis mais complexas.

Ao analisar trabalhos semelhantes, como o do artigo [23] observa-se que a inclusão de variáveis mais detalhadas e contextuais, como métricas táticas ou probabilidades de casas de apostas, pode melhorar o desempenho do modelo. Contudo, a simplicidade e acessibilidade do fluxo de trabalho no **Orange** tornam este projeto ideal para fins didáticos e exploratórios.

Futuras melhorias que identifiquei incluem:

- A integração de variáveis adicionais, como dados individuais de jogadores ou fatores contextuais (lesões, condições meteorológicas).
- Testes com técnicas *ensemble* mais avançadas.

Por fim, este trabalho contribui para o avanço do uso de machine learning em previsões desportivas, oferecendo insights tanto para apostadores como para analistas desportivos, com foco na transparência e simplicidade de implementação.

Referências Bibliográfica

[1] "What are the most popular sports in the world," *World Atlas*. Available: https://www.worldatlas.com/articles/what-are-the-most-popular-sports-in-the-world.html.

[2] "Global sports betting market," *Sports Betting Dime*. Available: https://www.sportsbettingdime.com/guides/finance/global-sports-betting-market/.

[3] "Bookmaker odds," *OddsMatrix*. Available: https://oddsmatrix.com/bookmaker-odds/.

[4] "Types of sports bets," *FlashPicks*. Available: https://flashpicks.com/guides/types-of-sports-bets.

[5] "Understanding the math behind betting odds," *Investopedia*. Available: https://www.investopedia.com/articles/dictionary/042215/understand-math-behind-betting-odds-gambling.asp.

[6] "KDD process in data mining," *GeeksforGeeks*. Available: https://www.geeksforgeeks.org/kdd-process-in-data-mining/.

[7] "Player scores dataset," *Kaggle*. Available: https://www.kaggle.com/datasets/davidcariboo/player-scores.

[8] "Transfermarkt," *Transfermarkt*. Available: https://www.transfermarkt.us/.

[9] A. Freitas et al., "Previsão de resultados desportivos com machine learning," Universidade Federal do Rio Grande do Sul, 2023. Available: https://lume.ufrgs.br/bitstream/handle/10183/261788/001172666.pdf?sequence=1.

[10] E. Nogueira, "Data mining aplicado à previsão de resultados de futebol," Instituto Politécnico do Porto, 2019. Available: https://recipp.ipp.pt/bitstream/10400.22/15552/1/DM_EduardoNogueira_2019_MEI.pdf.

[11] "Ensemble methods in Python," *GeeksforGeeks*. Available: https://www.geeksforgeeks.org/ensemble-methods-in-python/.

[12] "Understanding random forest," *Analytics Vidhya*. Available: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/.

[13] "Logistic regression: Detailed overview," *Towards Data Science*. Available: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc.

[14] "Logistic regression," *GeeksforGeeks*. Available: https://www.geeksforgeeks.org/understanding-logistic-regression/.

[15] "K-Nearest Neighbors (kNN)," *Scikit-learn Documentation*. Available: https://scikit-learn.org/stable/modules/neighbors.html.

[16] "ML Gradient Boosting," *GeeksforGeeks*. Available: https://www.geeksforgeeks.org/ml-gradient-boosting/?ref=header_outind.

[17] IBM, "Neural Networks," *IBM Think Blog*. Available: https://www.ibm.com/think/topics/neural-networks.

[18] M. Filho, "As métricas mais populares para avaliar modelos de machine learning," 2019. Available: https://mariofilho.com/as-metricas-mais-populares-para-avaliar-modelos-de-machine-learning/.

[19] "F1 Score, Accuracy, ROC AUC, and PR AUC," *Neptune.ai*. Available: https://neptune.ai/blog/f1-score-accuracy-roc-auc-pr-auc.

[20] "Accuracy, precision, recall," *Google Developers*. Available: https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall?hl=pt-br.

[21] A. Alzahrani et al., "Evaluating classification performance measures: an empirical study," *BMC Genomics*, vol. 20, no. 1, pp. 1–12, 2019. Available: https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-019-6413-7.

[22] "Orange Data Mining Documentation," *Orange Data Mining*. Available: https://orangedatamining.com/docs/.

[23] M. Hvattum and H. Arntzen, "Using Bookmaker Odds to Predict the Final Result of Football Matches," *International Journal of Forecasting*, vol. 26, no. 3, pp. 460–470, 2010. Available: https://www.researchgate.net/publication/262395354_Using_Bookmaker_Odds_to_Predict_the_Final_Result_of_Football_Matches.

APÊNDICE

A. *Data_processing.py*

B. *Data_separator.py*

C. *Machine_learning.ows*