UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
BACHARELADO EM ESTATÍSTICA

Letycia Milene Neves Correia

Simulação do Campeonato Brasileiro de 2013 e Estimativa de Lucros em um Sistema de Apostas Hipotético

Goiânia
2025

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC no 1240/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei no 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo dos Trabalhos de Conclusão dos Cursos de Graduação disponibilizado no RI/UFG é de responsabilidade exclusiva dos autores. Ao encaminhar(em) o produto final, o(s) autor(a)(es)(as) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG)

Nome(s) completo(s) do(a)(s) autor(a)(es)(as): Letycia Milene Neves Correa.

Título do trabalho: Simulação do Campeonato Brasileiro de 2013 e Estimativa de Lucros em um Sistema de Apostas Hipotético.

2. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador) Concorda com a liberação total do documento [x] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante: a) consulta ao(à)(s) autor(a)(es)(as) e ao(à) orientador(a); b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo do TCCG. O documento não será disponibilizado durante o período de embargo.

Casos de embargo:

- Solicitação de registro de patente;

- Submissão de artigo em revista científica;

- Publicação como capítulo de livro.

Obs.: Este termo deve ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Valdivino Vargas Junior**, **Professor do Magistério Superior**, em 28/11/2025, às 19:07, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Letycia Milene Neves Correia**, **Discente**, em 10/12/2025, às 18:46, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5822499** e o código CRC **1D9D3014**.

Referência: Processo nº 23070.060282/2025-32 SEI nº 5822499

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
BACHARELADO EM ESTATÍSTICA

Letycia Milene Neves Correia

Simulação do Campeonato Brasileiro de 2013 e Estimativa de Lucros em um Sistema de Apostas Hipotético

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Estatística da Universidade Federal de Goiás para aprovação no componente curricular TCC, como parte das exigências para a obtenção do título de bacharel em Estatística. **Orientador**: Valdivino Vargas Júnior

Goiânia

2025

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Correia, Letycia Milene Neves
Simulação do Campeonato Brasileiro de 2013 e Estimativa de Lucros em um Sistema de Apostas Hipotético [manuscrito] / Letycia Milene Neves Correia. - 2025.
61 f.: il.

Orientador: Prof. Dr. Valdivino Vargas Júnior.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Instituto de Matemática e Estatística (IME), Estatística, Goiânia, 2025.
Bibliografia. Anexos.

1. Probabilidade. 2. Futebol. 3. Odds. 4. Lei dos grandes números. I. Júnior, Valdivino Vargas, orient. II. Título.

CDU 519.22

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA

ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO

Aos vinte e sete dias do mês de novembro do ano de 2025 iniciou-se a sessão pública de defesa do Trabalho de Conclusão de Curso (TCC) intitulado “Simulação de Campeonato e Avaliação de Estratégia de Apostas”, de autoria de Letycia Milene Neves Correa, do curso de Estatística, do Instituto de Matemática e Estatística da UFG. Os trabalhos foram instalados pelo Prof. Dr. Valdivino Vargas Junior com a participação dos demais membros da Banca Examinadora: Marley Apolinário Saraiva (IME/UFG) e Joelmir Divino Carlos Feliciano (IME/UFG). Após a apresentação, a banca examinadora realizou a arguição da estudante. Registrou-se a solicitação de alteração do título do trabalho. Após deliberação, o orientador definiu como novo título: "Simulação do Campeonato Brasileiro de 2013 e Estimativa de Lucros em um Sistema de Apostas Hipotético”.' Posteriormente, de forma reservada, a Banca Examinadora atribuiu a nota final de 8,6, tendo sido o TCC considerado aprovado.

Proclamados os resultados, os trabalhos foram encerrados e, para constar, lavrou-se a presente ata que segue assinada pelos Membros da Banca Examinadora.

Documento assinado eletronicamente por **Valdivino Vargas Junior**, **Professor do Magistério Superior**, em 27/11/2025, às 18:14, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Joelmir Divino Carlos Feliciano**, **Professor do Magistério Superior**, em 27/11/2025, às 18:35, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Marley Apolinario Saraiva**, **Professor do Magistério Superior**, em 28/11/2025, às 18:30, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5801937** e o código CRC **6C7344D6**.

Agradecimentos

Gostaria de agradecer aos meus pais, agradeço pelo apoio constante, incentivo e por me motivarem a seguir firme mesmo nos momentos de dificuldade. Ao meu orientador, Valdivino, pela paciência, orientação e me incentivar a participar ativamente de eventos do IME, me orientando no desenvolvimento e na apresentação do meu trabalho, o que contribuiu significativamente para meu crescimento acadêmico e profissional. À minha tia, meu agradecimento pelo suporte, motivação e pelas palavras de incentivo que tornaram este caminho mais leve.

Resumo

O Campeonato Brasileiro de Futebol é uma das competições esportivas mais populares e competitivas do Brasil. Diante disso, este trabalho tem como objetivo aplicar conceitos de probabilidade para a previsão da classificação final e estimar a média de pontos do Campeonato Brasileiro de Futebol de 2013. Em continuidade, simular o lucro de uma casa de apostas utilizando as probabilidades geradas pelo modelo desenvolvido. A metodologia consiste no desenvolvimento de um modelo probabilístico que se baseia na análise da convergência de variáveis aleatórias, fundamentada na Lei dos Grandes Números, que assegura que, conforme o número de observações aumenta, a média dos resultados tende a se aproximar do valor esperado. Além disso, foi utilizada a técnica de suavização exponencial para atualizar os vetores de probabilidades das equipes ao longo da competição, atribuindo pesos distintos para a partida mais recente e para o histórico. As 280 primeiras partidas do campeonato foram utilizadas para estimar os pesos por meio do cálculo de uma medida de distância entre as probabilidades do modelo desenvolvido e as probabilidades consideradas por uma banca de casa de apostas esportivas na construção de ODDs de abertura dos jogo. Com o peso ótimo definido, o modelo foi treinado e, em seguida, foi aplicado o método de Monte Carlo, implementado em linguagem Python, para estimar a média de pontos e a probabilidade de título de cada equipe nas partidas restantes. Para um resultado confiável foi realizada 1 milhão de simulações, tendo em vista que ao aumentar as repetiçõos o valor estimado se aproxima mais do valor real. Inclusive, o modelo também é utilizado para analisar o lucro de uma casa de apostas utilizando as probabilidades estimadas pelo modelo e assumindo que as apostas são uniformes. Dessa forma, o trabalho pretende tanto demonstrar a aplicabilidade de métodos estatísticos em cenários esportivos quanto evidenciar o lucro das casas de apostas.

Palavras-chave: Probabilidade. Futebol. Odds. Lei dos grandes números.

Abstract

The Brazilian Football Championship is one of the most popular and competitive sporting competitions in Brazil. Therefore, this study aims to apply probability concepts to predict the final standings and estimate the average points of the 2013 Brazilian Football Championship. In addition, it seeks to estimate the profit of a betting house using the probabilities generated by the developed model.

The methodology consists of developing a probabilistic model based on the analysis of the convergence of random variables, grounded in the Law of Large Numbers, which ensures that, as the number of observations increases, the average of the results tends to approach the expected value. Furthermore, the exponential smoothing technique was used to update the probability vectors of the teams throughout the competition, assigning distinct weights to the most recent matches and to historical performance.

The first 280 matches of the championship were used to estimate the weights through the calculation of a distance measure between the probabilities generated by the developed model and the probabilities considered by a betting house in constructing opening odds for the games. Once the optimal weight was defined, the model was trained and then the Monte Carlo method, implemented in Python, was applied to estimate the average points and the probability of winning the championship for each team in the remaining matches. For reliable results, one million simulations were performed, considering that increasing the number of repetitions brings the estimated value closer to the actual value.

Moreover, the model is also used to analyze the profit of a betting house using the probabilities estimated by the model, assuming uniform bets. In this way, the study aims both to demonstrate the applicability of statistical methods in sports scenarios and to highlight the profit of betting houses.

Keywords: Probability. Football. Odds. Law of large numbers.

Sumário

Introdução

O Campeonato Brasileiro de Futebol é uma das disputas esportivas mais tradicionais do Brasil e ele ocorre anualmente e reúne os melhores times para disputarem entre si em 38 rodadas, que são compostas de 10 partidas cada rodada. Todos os times jogam contra todos em turno e returno, e é realizada a contagem de pontos no formato de: 3 pontos para o time vencedor e 0 pontos para o perdedor; e 1 ponto para cada time em caso de empate. A classificação final do campeonato é definida pela quantidade de pontos acumulados. No entanto, em caso de empate, outros critérios são utilizados para decidir o campeão, como saldo de gols, número de vitórias ou confronto direto.

Para fins deste estudo, serão utilizados os dados do Campeonato Brasileiro de 2013 para o desenvolvimento do trabalho, que será dividido em duas partes principais. A primeira parte envolve a modelagem probabilística de resultados de partidas de futebol utilizando suavização exponencial, que permite atualizar dinamicamente as probabilidades de vitória, empate ou derrota de cada equipe ao longo do campeonato. E a segunda parte analisa o lucro de uma casa de apostas com base em simulações fundamentadas no modelo desenvolvido. Essa etapa busca ilustrar o quão desvantajoso pode ser apostar do ponto de vista do jogador, bem como estimar os ganhos potenciais das casas de apostas nesse contexto.

1 Revisão Bibliográfica

Este trabalho se fundamenta na aplicação de conceitos probabilísticos e métodos de simulação estocástica para a previsão de resultados esportivos. A abordagem combina técnicas de suavização exponencial com simulações Monte Carlo, tendo como base teórica os teoremas da probabilidade. A seguir será apresentada as definições e teoremas utilizados no projeto.

1.1 Convergência

O estudo da convergência de variáveis aleatórias é fundamental para a validação teórica das simulações que serão realizadas.

Definição 1. ***Convergência Quase Certa***
Sejam $X_1, X_2, \ldots, X$ variáveis aleatórias em um espaço de probabilidade $(\Omega, \mathcal{F}, \mathbb{P})$. Dizemos que X_n converge para X quase certamente ($X_n \xrightarrow{q.c.} X$) se o evento $\{\omega \in \Omega : X_n(\omega) \to X(\omega)$ quando $n \to \infty\}$ tem probabilidade 1. (BILLINGSLEY, 1995)

1.2 Lei dos Grandes Números

A base teórica principal deste trabalho se fundamenta na aplicação da Lei dos Grandes Números, descrita por:

Teorema 1. ***Primeira Lei Forte dos Grandes Números de Kolmogorov (1933)***
Seja $X_1, X_2, X_3, \ldots$ uma sequência de variáveis independentes e identicamente distribuídas, com média comum (μ) finita. Defina $S_n = X_1 + X_2 + \cdots + X_n$. Então,

$$\frac{S_n}{n} \xrightarrow{q.c.} \mu$$

(ROSS, 2010, Teorema 4.1, p. 473).

Uma vez que a Lei Forte dos Grandes Números garante a convergência quase certa da frequência relativa para a probabilidade teórica, o método de Monte Carlo foi empregado para estimar as probabilidades dos eventos do campeonato, utilizando-se um elevado número de simulações.

Exemplo 1. ***Aplicação no Contexto Esportivo:***
No contexto das simulações do campeonato de futebol que serão desenvolvidas, seja I_k a variável indicadora de um evento qualquer (por exemplo, título de um time específico) na k-ésima simulação:

Então, a proporção de ocorrências do evento em n simulações será:

$$\hat{p}_n = \frac{1}{n}\sum_{k=1}^{n} I_k \xrightarrow{q.c.} p$$

onde p é a probabilidade real do evento.

1.3 Método de Monte Carlo

O método de Monte Carlo consiste na geração de números aleatórios de maneira computacional para aproximação de parâmetros estatísticos, permitindo resolver problemas complexos de integração e otimização por meio de simulações repetidas (METROPOLIS; ULAM, 1949). No contexto de campeonatos esportivos, sua aplicação ocorre na simulação de milhares de possíveis sequências de partidas a partir de probabilidades associadas às partidas. A cada simulação, os resultados dos jogos são sorteados conforme a distribuição uniforme, e com a repetição de um grande número de simulações permitiu-se construir distribuições empíricas para variáveis de interesse, como a probabilidade de cada equipe se classificar, a chance de conquistar o título ou a frequência de determinados cenários no campeonato.

1.4 Suavização Exponencial

A suavização exponencial é uma técnica estatística utilizada para gerar previsões a partir de séries temporais, atribuindo pesos decrescentes aos dados conforme eles se tornam mais antigos. Diferente das médias móveis simples, a suavização exponencial dá maior importância às observações mais recentes, permitindo que o modelo responda mais rapidamente a mudanças nas tendências dos dados (MORETTIN; TOLOI, 2018). Essa característica a torna uma ferramenta bastante usada em análises econômicas, financeiras e de demanda.

O modelo mais elementar, conhecido como suavização exponencial simples, utiliza uma equação recursiva que combina o valor observado mais recente e a previsão anterior, ponderados por um parâmetro de suavização α (α). A fórmula do modelo de suavização exponencial simples é a seguinte:

$$S_t = \alpha X_t + (1-\alpha)S_{t-1}$$

onde:

S_t = valor suavizado (ou previsão) no tempo t;

X_t = valor observado (real) no tempo t;

S_{t-1} = valor suavizado no período anterior;

α = parâmetro de suavização, com$0 < \alpha < 1$.

Interpretação:

Essa equação mostra que o valor suavizado atual S_t é uma média ponderada entre o valor observado mais recente X_t e a estimativa anterior S_{t-1}. O parâmetro α controla o peso relativo de cada um:

Se α for alto (próximo de 1), o modelo dá mais peso aos dados recentes, reagindo rapidamente a mudanças.

Se α for baixo (próximo de 0), o modelo suaviza mais a série, sendo menos sensível a variações.

Esse parâmetro controla a sensibilidade do modelo às novas informações: valores de α próximos de 1 tornam as previsões mais reativas às variações recentes, enquanto valores menores produzem séries mais estáveis e suavizadas (HYNDMAN; ATHANASOPOULOS, 2021). Assim, o método equilibra a influência entre o passado e o presente de forma dinâmica.

Além da versão simples, existem extensões do método, como a suavização exponencial dupla e a tripla (modelo de Holt-Winters), que incorporam componentes de tendência e sazonalidade. Essas variações ampliam o uso do modelo em contextos onde os dados apresentam comportamento periódico ou crescimento constante (MONTGOMERY; JOHNSON; GARDINER, 2015). Por sua simplicidade e eficiência, a suavização exponencial é considerada uma das abordagens mais práticas para previsão de séries temporais.

1.5 Modelos Probabilísticos na Previsão de Resultados do Futebol

Com o intuito de realizar previsões nos campeonatos de futebol, foram desenvolvidas diferentes abordagens utilizando os conceitos citados na seção anterior. A seguir serão citados alguns estudos.

Lima et al. (LIMA *et al.*, 2010) abordam a aplicação de modelos probabilísticos no futebol, realizando simulações computacionais para a estimativa de resultados. Os autores fundamentam seu método na Lei dos Grandes Números, utilizando variáveis aleatórias com distribuição uniforme ($U(0,1)$) para gerar os resultados de partidas de forma aleatória e controlada. Essa escolha teórica garante que, à medida que o número de simulações aumenta, as estimativas se tornem mais estáveis e próximas das probabilidades reais. O estudo é pioneiro no cenário nacional ao propor a formalização da previsão de resultados em campeonatos de pontos corridos com base em vetores de força, que diferenciam o desempenho das equipes como mandantes e visitantes. A atualização contínua desses vetores, de acordo com o rendimento dos adversários,

permite incorporar o fator mando de campo e o efeito psicológico das vitórias e derrotas ao longo da competição.

Esse modelo, apesar de simples, serviu de base para o desenvolvimento de trabalhos posteriores que buscaram aprimorar a representação da incerteza no futebol. Ao empregar sorteios aleatórios uniformes para simular partidas e aplicar a Lei dos Grandes Números como sustentação teórica Lima et al. (LIMA *et al.*, 2010) demonstram que é possível aproximar a matemática da realidade esportiva sem perder o rigor formal. O estudo contribui para a popularização da matemática aplicada ao esporte e mostra que a aleatoriedade, quando tratada de forma estruturada, pode ser uma poderosa ferramenta para estimar probabilidades em sistemas complexos.

Kuhnert e Possato (KUHNERT; POSSATO, 2023) desenvolveram um modelo mais robusto, propondo uma abordagem utilizando distribuições Poisson duplas para modelar o número de gols, assumindo independência entre os times, ajustado por paramêtros de força ofensiva e defensiva com base nos dados históricos das partidas. Essa configuração permite estimar probabilidades de placares específicos, incorporando elementos como o fator casa, que adiciona em média 0,5 gols ao mandante, e a forma recente dos times, ajustada por média móvel exponencial (MME) para dar mais peso a desempenhos atuais. Eles corrigem distorções comuns, como a subestimação de empates, via fator rho, que infla essa probabilidade em cerca de 10% e redistribui as chances de vitória e derrota para manter a soma em 100%.

Kuhnert e Possato (KUHNERT; POSSATO, 2023) validam o modelo com simulações Monte Carlo, simulando os cenários diversas vezes para gerar distribuições empíricas de resultados, e avaliam a precisão pela distância de De Finetti, que mede o erro quadrático entre previsões e resultados reais. Eles testam com dados reais do futebol brasileiro, alcançando acurácia comparável a modelos internacionais, mas apontam limitações como sensibilidade a dados incompletos ou flutuações inesperadas.

Outro estudo na área de previsões é o modelo de Ramos, Lemos e Batista (RAMOS; LEMOS; BATISTA, 2021) para previsão de resultados em jogos de futebol baseado na distribuição de Poisson para representar o número de gols marcados por cada equipe, assumindo independência entre os gols feitos pelos times e também adicionando a influência de fatores de ataque e defesa. Cada equipe é caracterizada por dois vetores de médias: um para atuações como mandante, abrangendo gols feitos e sofridos em casa, e outro para visitante, com gols feitos e sofridos fora. Esses vetores são inicializados com médias globais do campeonato anterior e atualizados rodada a rodada, calculando-se a média aritmética entre os valores iniciais e os gols acumulados divididos pelo número de rodadas mais um, permitindo uma adaptação progressiva ao desempenho observado. O fator mandante é obtido pela média entre gols feitos em casa pela equipe mandante e gols sofridos fora pela visitante, enquanto o fator visitante segue a média inversa, formando um vetor que orienta a simulação de placares via geração de números aleatórios em subintervalos probabilísticos vindos da função de Poisson, com probabilidades acumuladas para gols de zero a cinco ou mais.

A avaliação da qualidade das previsões também emprega a medida de De Finetti, que calcula a distância euclidiana quadrática entre o vetor de probabilidades estimadas (vitória do mandante, empate, vitória do visitante) e o vértice correspondente ao resultado real no espaço probabilístico unitário. Aplicado a jogos da Série A do Brasileirão a partir da rodada intermediária, o modelo realiza múltiplas simulações de cenários completos do campeonato, atualizando classificações e vetores após cada rodada simulada, para estimar probabilidades de campeões, classificados para torneios continentais e rebaixados. Comparado ao modelo da UFMG (LIMA *et al.*, 2010), que adota simulações estocásticas semelhantes, o método proposto exibe coerência em previsões, enfatizando a efetividade da abordagem Poisson em contextos onde o efeito mandante impacta os resultados, com uma porção significativa das medidas de De Finetti indicando qualidade aceitável ao ficar abaixo do limiar de referência comumente adotado.

2 Metodologia

2.1 Coleta e Tratamento de Dados

Os dados das partidas foram coletados os por meio de fontes secundárias usando a biblioteca `BeautifulSoup` em Python, realizando *web scraping* para extrair automaticamente os resultados do Campeonato Brasileiro de 2013. A tabela extraída inclui: nome dos times, gols marcados, gols sofridos, data e hora da partida.

Após a coleta, foi realizado o tratamento dos dados, que conta com as etapas de:

- Formatação de datas.
- Ordenação cronológica dos jogos.
- Diferenciação entre mandante e visitante com os sufixos "_M" e "_V", pois o desempenho varia conforme o local da partida.

2.2 Treinamento do Modelo

2.2.1 Inicialização dos Perfis de Performance

Cada time possui dois vetores de perfil (mandante e visitante), cada um com três componentes: vitória, empate e derrota. Todos os vetores foram inicializados igualmente, dada a ausência de histórico:

$$\left(\tfrac{1}{3}, \tfrac{1}{3}, \tfrac{1}{3}\right).$$

2.2.2 Atualização dos Perfis via Suavização Exponencial

Após cada partida, os perfis foram atualizados conforme:

$$\mathbf{PM}_n = h\,\mathbf{PM}_{n-1} + (1-h)\,\mathbf{RUP}, \quad \mathbf{PV}_n = h\,\mathbf{PV}_{n-1} + (1-h)\,\mathbf{RUP}$$

onde h é o parâmetro de suavização e $\mathbf{RUP}$ é o vetor unitário correspondente ao resultado (vitória, empate ou derrota).

O parâmetro de suavização h controla a importância atribuída ao histórico das partidas. Quando h está próximo de 1, o modelo dá mais peso às informações passadas, resultando em perfis mais estáveis, com menor sensibilidade às últimas partidas. Por outro lado, quando h está próximo de 0, o modelo se torna mais responsivo, priorizando fortemente os resultados mais recentes.

Exemplo 2. *Suponha:*

$$\mathbf{PM}_A = (0.4, 0.5, 0.1), \quad \mathbf{PV}_B = (0.3, 0.5, 0.2), \quad h = 0.8.$$

Probabilidades antes da partida:

$$P_{VM} = 0.40, \quad P_E = 0.50, \quad P_{VV} = 0.10.$$

Se A vence, o perfil atualizado do mandante A é:

$$\mathbf{PM}_A^{(novo)} = 0.8(0.4, 0.5, 0.1) + (1 - 0.8)(1,0,0) = (0.52, 0.40, 0.08).$$

E o perfil atualizado do visitante B:

$$\mathbf{PV}_B^{(novo)} = 0.8 \cdot (0.3, 0.5, 0.2) + (1 - 0.8) \cdot (0, 0, 1) = (0.24, 0.4, 0.36).$$

2.2.2.1 Estimação do h

Estimamos o valor ótimo de h para o modelo por meio de um processo iterativo, variando o h de 0 a 1, com incrementos de 0,01, e avaliamos o desempenho em cada ponto. Para essa análise, utilizamos as odds de abertura da casa de apostas Bet365 em cada rodada, que serviram como base para calcular as probabilidades implícitas assumidas pela banca. Assim, realizamos:

$$P_r^i = \frac{1}{odds_{resultado}}$$

onde P_r^i corresponde a probabilidade inflada de um dado resultado. Após calculado a probabilidade inflada é retirado o lucro médio estimado pela banca que aparece como a quantidade em que a soma de probabilidades (de vitória, empate e derrota) ultrapassa 100 $\%$. Estamos admitindo, por hipótese, que a banca infla as probabilidades de forma igualitária. Assim, usamos como probabilidade o valor inflado dividido pela soma das probabilidades dos resultados.

$$P_r = \frac{P_r^i}{P_v + P_d + P_e}.$$

Após a obtenção das probabilidades estimadas para a banca de apostas foi realizada a comparação com as probabilidades calculadas pelo modelo. Usamos uma função $\delta(h)$ que calcula a diferença das probabilidades do nosso modelo com o valor estimado para a banca. Para cada valor de h treinamos o modelo nas 28 rodadas iniciais e calculamos $\delta(h)$ do seguinte modo:

$$\delta(h) = \sum_{i=181}^{380} [|PVM_b(i) - PVM_m(i)| + |PE_b(i) - PE_m(i)| + |PDM_b(i) - PDM_m(i)|]$$

onde i diz respeito a i-ésima partida do campeonato, $PVM_b(i), PE_b(i), PDM_b(i)$ são as probabilidades estimadas da banca para a i-ésima partida e $PVM_m(i), PE_m(i), PDM_m(i)$ são as probabilidades estimadas pelo nosso modelo para a i-ésima partida. Escolhemos o valor de h que minimiza $\delta(h)$. Isto é, se $\delta(h_c) = \min\{\delta(h); h \in \mathbb{N}\}$, então h_c o valor ótimo estimado, dado que esse peso gera probabilidades mais próximas das observadas pela banca.

2.3 Simulação de Partidas

Com o valor ótimo de h definido, foi realizada a simulação das partidas restantes do campeonato (a partir da 29ª rodada). Foram executadas 1 milhão de simulações completas do torneio, utilizando o vetor de probabilidades calculado no treinamento do modelo até a 28ª rodada, que representa o desempenho estimado de cada equipe até aquele ponto da competição. Ao final de cada simulação, o modelo retornava ao vetor de treinamento, garantindo que cada iteração se iniciasse a partir das mesmas condições iniciais. Foi definido que o modelo seria treinado até a 28ª rodada, de modo a garantir uma quantidade consistente de partidas para a estimação dos parâmetros e, ao mesmo tempo, preservar uma janela de aproximadamente 100 jogos para a etapa de simulação. A simulação do resultado de cada partida tem a seguinte metodologia. Se A (mandante) e B (visitante) vão se enfrentar e tem perfis

$$PM_A = (PM_{A,\text{vitória}}, PM_{A,\text{empate}}, PM_{A,\text{derrota}}),$$
$$PV_B = (PV_{B,\text{vitória}}, PV_{B,\text{empate}}, PV_{B,\text{derrota}}).$$

definimos a probabilidade dos resultados da partida $A \times B$ como:

$$P_{\text{VM}} = \frac{PM_{A,\text{vitória}} + PV_{B,\text{derrota}}}{2}$$
$$P_{\text{E}} = \frac{PM_{A,\text{empate}} + PV_{B,\text{empate}}}{2}$$
$$P_{\text{VV}} = \frac{PM_{A,\text{derrota}} + PV_{B,\text{vitória}}}{2}$$

Daí, o resultado de cada jogo é determinado por sorteio, considerando uma variável aleatória $U \sim \mathcal{U}(0,1)$, da seguinte forma:

- $U \leq P_{\text{VM}}$: vitória mandante;
- $P_{\text{VM}} < U \leq P_{\text{VM}} + P_E$: empate;
- restante: vitória visitante.

Exemplo 3. *Se $P_{VM} = 0.3$, $P_E = 0.5$, $U = 0.72$, então empate.*

Aplicando o método de Monte Carlo ao simular este processo 1 milhao de vezes, obtemos as probabilidades de interesse.

2.4 Simulação de Apostas em Bancas

A simulação do sistema de apostas de banca foi realizada a partir do vetor de probabilidades gerado pelo modelo, calculado com base nos resultados reais até a 28ª rodada. Esse vetor reflete as probabilidades estimadas de vitória, empate e derrota de cada equipe, considerando

o peso histórico h previamente estimado no modelo. A partir desse vetor, foram simuladas as apostas, assumindo que fosse investida uma unidade monetária em cada resultado possível. Nesse processo:

- As probabilidades reais foram inflacionadas para que a banca garanta lucro.
- As *odds* foram calculadas como inverso da probabilidade inflacionada.

Exemplo 4. ***Vamos calcular as Odds para os seguintes dados:***
Dados:

$$P_{VM} = 0.50, \quad P_E = 0.40, \quad P_{VV} = 0.10$$

Aplicando inflação de 20%:

$$P_{VM}^{infl} = 0.60, \quad P_E^{infl} = 0.48, \quad P_{VV}^{infl} = 0.12$$

As odds são:

$$Odd_{VM} = 1/0.60 \approx 1.67, \quad Odd_E = 1/0.48 \approx 2.08, \quad Odd_{VV} = 1/0.12 \approx 8.33$$

Tabela 1 – Probabilidades reais, inflacionadas e odds calculadas.

Resultado	Prob. Real	Prob. Inflacionada	Odd
Vitória Mandante	0.50	0.60	1.67
Empate	0.40	0.48	2.08
Vitória Visitante	0.10	0.12	8.33

Fonte: Elaboração própria.

O retorno do jogador, em caso de acerto do resultado, é o valor da odd multiplicado por essa aposta.

Considere uma partida entre os times A (mandante) e B (visitante), com as seguintes odds:

$$ODD_{\text{vitória mandante}} = 1.67, \quad ODD_{\text{empate}} = 2.08, \quad ODD_{\text{vitória visitante}} = 8.33.$$

Interpretação: *As odds indicam o retorno de uma aposta de R$1,00 em caso de acerto.*

- *Apostando R$1,00 na vitória do mandante, o retorno será R$1,67, gerando um lucro líquido de* $1{,}67 - 1 = R\$0{,}67$.
- *Apostando R$1,00 no empate, o retorno será R$2,08, com lucro líquido de* $2{,}08 - 1 = R\$1{,}08$.

- *Apostando R$1,00 na vitória do visitante, o retorno será R$8,33, com lucro líquido de* $8{,}33 - 1 = R\$7{,}33$.

O lucro da banca foi avaliado comparando as odds ajustadas e os resultados reais das partidas, o que permitiu medir o ganho da banca e comparar com o ganho esperado em função do vetor de probabilidades e do parâmetro h utilizado no modelo.

2.5 Descrição dos procedimentos

A seguir, será apresentada uma sequência resumida dos procedimentos, a qual está separada em duas etapas principais.

2.5.1 Primeira Parte – Simulação do Campeonato

Foi utilizado um modelo de suavização exponencial para alimentar dois vetores de perfil para cada time: um como mandante e outro como visitante. Cada vetor possui três entradas, correspondentes às tendências do time vencer, empatar ou perder. A soma dessas três tendências é igual a 1.

Até a 28ª rodada, os vetores foram atualizados com base nos resultados reais. Após cada jogo, aplicamos a suavização exponencial para atualizar os respectivos vetores dos times envolvidos.

A partir dos perfis atualizados até a 28ª rodada, realizamos múltiplas simulações do restante do campeonato. Durante essas simulações:

- Os perfis continuaram sendo atualizados por suavização exponencial, agora a partir dos jogos simulados.
- Cada partida foi simulada a partir de uma variável aleatória uniforme no intervalo $(0{,}1)$.

As probabilidades dos resultados de cada jogo foram estimadas da seguinte forma:

- A probabilidade de vitória do time mandante (A) é a média aritmética entre o perfil de vitória de A como mandante e o perfil de derrota de B como visitante.
- A probabilidade de vitória do visitante (B) é a média entre o perfil de vitória de B como visitante e o perfil de derrota de A como mandante.
- A probabilidade de empate é a média entre os perfis de empate de A como mandante e de B como visitante.

Após cada simulação, retornamos aos perfis da 28ª rodada, garantindo consistência nos experimentos.

O modelo de suavização exponencial possui um parâmetro h, que representa a influência dos jogos passados. Quanto maior h, maior a influência histórica nos vetores de perfil.

Para encontrar o melhor valor de h, utilizamos uma base de dados de *odds* (cotações) retiradas da internet. A partir dessas *odds*, foram estimadas as probabilidades implícitas atribuídas pelas casas de apostas e elas foram comparadas com as probabilidades estimadas pelo nosso modelo.

Foi avaliada a distância entre essas duas distribuições de probabilidades, adotando-se como melhor valor de h aquele que minimiza essa distância.

Com o valor ótimo de h, o modelo foi utilizado para responder perguntas como:

- Qual a probabilidade de um time ser campeão?
- Qual a probabilidade de determinado time ser rebaixado?
- Qual a probabilidade de o campeão atingir certa pontuação?

2.5.2 Segunda Parte – Simulação dos Ganhos da Banca

Na segunda parte do trabalho, foi utilizado o modelo para simular os ganhos de uma casa de apostas, ilustrando o quão desvantajoso pode ser apostar do ponto de vista do jogador.

Com base no melhor parâmetro h encontrado, foi simulado um sistema de apostas a partir da 29ª rodada, com foco em 100 partidas. O procedimento adotado será o seguinte:

1. Na rodada 29, foram utilizados os perfis acumulados até a rodada 28 para estimar as probabilidades de cada resultado.
2. Essas probabilidades foram inflacionadas em diversos valores representando a margem da casa e, em seguida, foram geradas as respectivas *odds*. Obtivemos resultados para os valoes de inflação $2{,}5\%, 5\%, \ldots, 17{,}5\%, 20\%$.
3. Foi suposto que uma unidade monetária foi apostada em cada resultado possível (mandante, empate, visitante), totalizando três unidades por partida.
4. Foram utilizados os resultados reais da rodada 29 para calcular o lucro da banca para cada valor de inflção dado.
5. Os perfis foram atualizados com os resultados da rodada 29 e o processo foi repetido para as rodadas seguintes até o fim do campeonato.

Ao final da simulação, foi obtido o balanço total da banca ao longo das 100 partidas, permitindo estimar o seu lucro acumulado e compará-lo com o lucro esperado por ela para cada valor de inflação dado.

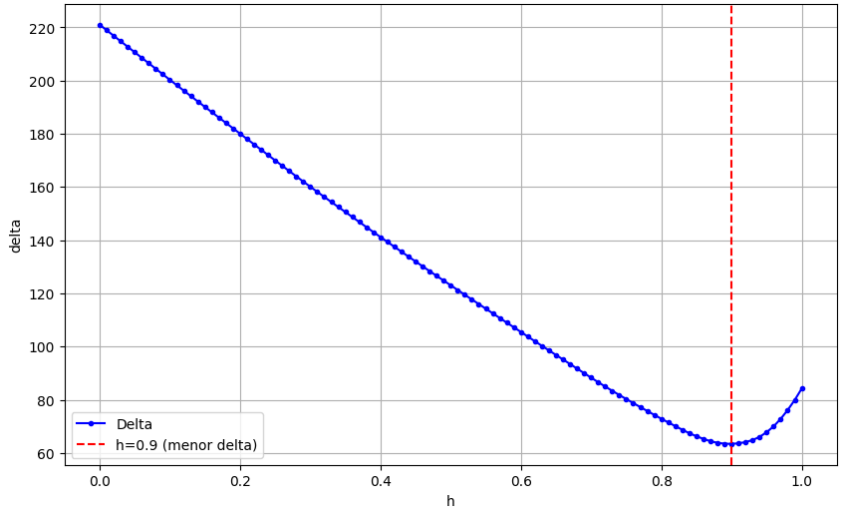
É importante observar que foi considerada uma unidade apostada em cada resultado por falta de dados sobre o comportamento real dos apostadores. Não sabemos, por exemplo, se eles tendem a apostar mais nos favoritos, nas zebras ou em seus times do coração. Portanto, foi adotada uma abordagem neutra para fins de simulação.

3 Resultados

3.1 Estimação do h

Para o desenvolvimento do modelo, inicialmente realizamos o estudo dos pesos e após minimizar o δ, que representa a soma das diferenças entre as probabilidades do modelo e as da banca, foi obtido o valor 0.9 para o peso do histórico. Na Figura 1, observa-se o comportamento de $\delta(h)$ em função de h, com a linha vermelha indicando o ponto de mínimo.

Figura 1 – Delta em função de h



Fonte: Elaboração própria.

3.2 Probabilidades das classificações finais

Com o modelo utilizando o h ótimo definido acima e após a realizar de 1 milhão de simulações de Monte Carlo a partir da 29ª rodada, obtivemos as seguintes probabilidades associadas a cada resultado possível de campeão.

Na Tabela 2 temos as probabilidades da classificação final das equipes do Campeonato Brasileiro de 2013. Observa-se que o Cruzeiro possui uma probabilidade significativamente superior às demais equipes, alcançando cerca de 90,2%, seguido do do Grêmio (4,4%), Botafogo (3,2%) e Athletico-PR (2,2%). Além disso, equipes como Atlético-MG, Goiás, Vitória e Internacional estão com probabilidades próximas ou inferiores a 0,001%, o que sugere baixíssima possibilidade de título. Já as demais equipes estão, na prática, fora da disputa por essa colocação específica, conforme indicado pelo valor zero de probabilidade.

Tabela 2 – Quadro de probabilidades de Título de Campeão

Times	Probabilidades
Cruzeiro	0.902042
Grêmio	0.043705
Botafogo	0.031841
Athletico-PR	0.021622
Atlético-MG	0.000404
Goiás	0.000144
Vitória	0.000133
Internacional	0.000055
Santos	0.000040
Flamengo	0.000004
Bahia	0.000006
Corinthians	0.000002
Fluminense	0.000001
São Paulo	0.000001
Coritiba	0.000000
Náutico	0.000000
Ponte Preta	0.000000
Vasco	0.000000
Portuguesa	0.000000

Fonte: Elaboração própria.

3.3 Probabilidades de rebaixamento

A Tabela 3 apresenta as probabilidades estimadas de rebaixamento das equipes participantes do campeonato. O Náutico apresenta risco máximo de rebaixamento, com probabilidade igual a 1,00. Os times Ponte Preta, Vasco e Criciúma também se encontram em situação preocupante, com valores elevados de 0,90, 0,43 e 0,38, respectivamente, indicando grandes chances de rebaixamento.

Outras equipes, como Bahia, Fluminense, São Paulo e Portuguesa, aparecem com probabilidades moderadas de rebaixamento, variando entre 0,10 e 0,23. Por outro lado, times como Grêmio, Botafogo e Cruzeiro não apresentam nenhum risco de rebaixamento de acordo com as simulações.

Tabela 3 – Probabilidade de rebaixamento

Time	Probabilidade
Náutico	1.00
Ponte Preta	0.90
Vasco	0.43
Criciúma	0.38
Coritiba	0.31
Portuguesa	0.23
São Paulo	0.19
Fluminense	0.19
Bahia	0.12
Flamengo	0.10
Corinthians	0.07
Internacional	0.02
Santos	0.03
Vitória	0.02
Goiás	0.01
Atlético-MG	0.00
Athletico-PR	0.00
Grêmio	0.00
Botafogo	0.00
Cruzeiro	0.00

Fonte: Elaboração própria.

3.4 Estimação de pontos ao final do campeonato

Outra estatística calculada por meio do modelo desenvolvido foi a quantidade média de pontos finais dos times. A Tabela 4 apresenta uma comparação entre a média de pontos calculada pelo modelo e os pontos observados no Campeonato para cada time. De modo geral, as estimativas para cada equipe se aproximaram razoavelmente dos valores reais, com uma diferença média de aproximadamente 2,7 pontos.

Tabela 4 – Pontos por time ao final do campeonato

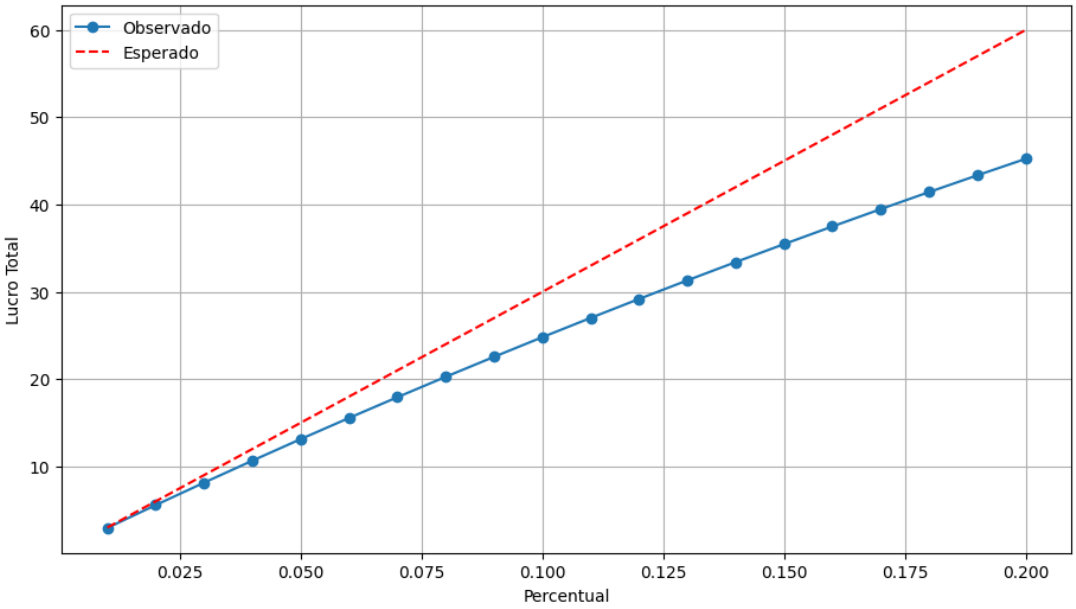
Time	Pontos Observados	Média de Pontos Estimada	Diferença Absoluta de Pontos
Cruzeiro	76	74.8	1.2
Grêmio	65	63.8	1.2
Botafogo	61	63.7	2.7
Athletico-PR	64	62.9	1.1
Atlético-MG	57	56.1	0.9
Goiás	59	54.2	4.8
Vitória	59	53.5	5.5
Santos	57	52.7	4.3
Internacional	48	52.4	4.4
Corinthians	50	50.4	0.4
Flamengo	45	50.1	5.1
Bahia	48	49.2	1.2
Fluminense	46	48.2	2.2
Portuguesa	44	47.9	3.9
São Paulo	50	47.7	2.3
Coritiba	48	46.1	1.9
Criciúma	46	45.5	0.5
Vasco	44	44.9	0.9
Ponte Preta	37	38.6	1.6
Náutico	20	28.0	8.0

Fonte: Elaboração própria.

3.5 Lucro simulado da banca

Ao simular diferentes níveis de inflação nas odds da casa de apostas, de 1% a 20%, e rodar o modelo desenvolvido com apostas de 1 unidade monetária em cada resultado possível (vitória, empate ou derrota) por partida, foi possível comparar o lucro esperado, calculado a partir das probabilidades infladas, com o lucro real obtido com base no placar efetivo do jogo. A Figura 2 apresenta o lucro estimado da banca para cada percentual de inflação aplicado às probabilidades, permitindo visualizar como alterações nas probabilidades afetam os lucros da banca.

Figura 2 – Lucro da banca x Lucro esperado em função do percentual de lucro esperado pela banca



Fonte: Elaboração própria.

Para comparar o ganho efetivo com o valor projetado pela banca, foi calculada a proporção entre essas duas medidas. Os resultados, apresentados na Tabela 5, mostram um comportamento consistente: conforme o percentual de inflação aplicado pela banca aumenta, essa proporção diminui. Isso indica que o lucro observado se afasta progressivamente do lucro esperado à medida que o carregamento nas probabilidades é intensificado. Esse efeito revela que o ajuste inflacionário não altera apenas os valores absolutos, mas também a relação entre expectativa e resultado, o que reforça a importância de analisar essas proporções na avaliação do desempenho do modelo.

Tabela 5 – Lucro obtido e Lucro esperado pela banca em função do valor de inflação

Inflação	**Lucro obtido**	**Lucro esperado**	**Proporção**
0.01	2.96	3.00	0.99
0.02	5.57	6.00	0.93
0.03	8.14	9.00	0.90
0.04	10.66	12.00	0.89
0.05	13.13	15.00	0.88
0.06	15.55	18.00	0.86
0.07	17.93	21.00	0.85
0.08	20.26	24.00	0.84
0.09	22.55	27.00	0.84
0.10	24.80	30.00	0.83
0.11	27.01	33.00	0.82
0.12	29.18	36.00	0.81
0.13	31.31	39.00	0.80
0.14	33.41	42.00	0.80
0.15	35.46	45.00	0.79
0.16	37.49	48.00	0.78
0.17	39.47	51.00	0.77
0.18	41.43	54.00	0.77
0.19	43.35	57.00	0.76
0.20	45.24	60.00	0.75

Fonte: Elaboração própria.

Conclusão

O estudo apresentou conceitos teóricos de probabilidade aplicados à classificação final do Campeonato Brasileiro de Futebol de 2013, resultando na estimativa das probabilidades de cada time se tornar campeão, ser rebaixado e na média final de pontos de cada equipe. Como o número de possibilidades de resultados (vitória, empate, derrota) de um campeonato com 380 jogos é 3^{380}, torna-se obviamente inviável estudar analiticamente todas as possibilidades. Mesmo faltando apenas dez rodadas para o final do campeonato, o número de desfechos possíveis ainda é da ordem de 3^{100}.

É aqui que entra a simulação computacional e o método de Monte Carlo. A ideia é replicar o campeonato inúmeras vezes e fazer uso da Lei Forte dos Grandes Números, que se mostra extremamente eficiente nesse contexto. Realizamos um grande número de simulações para obter estimativas mais precisas das probabilidades.

Se o número de repetições não for suficientemente grande, não haverá convergência dos resultados simulados para as probabilidades reais. Esses resultados não apenas ampliam o entendimento teórico sobre os teoremas limites, mas também abrem caminhos para futuras investigações para aperfeiçoamento das estimativas, tal como os pesos utilizados para ponderar os resultados recentes e históricos no cálculo das probabilidades.

Vale ressaltar as limitações do modelo, visto que ele não leva em consideração fatores externos e situacionais que podem impactar o desempenho das equipes, como cartões amarelos e vermelhos, trocas de técnico, conflitos internos no elenco, contusões, mudanças nos valores de premiação ou outros problemas de bastidores.

Para trabalhos futuros, sugere-se o desenvolvimento de valores do parâmetro h que sejam dinâmicos ao longo do campeonato, refletindo melhor as variações de desempenho das equipes. Além disso, podem ser definidos valores distintos de h para diferentes times, de modo a representar melhor suas particularidades e comportamentos específicos.

Referências

BILLINGSLEY, P. **Probability and Measure**. 3. ed. [S.l.]: John Wiley Sons, Inc., 1995. Citado na página 11.

HYNDMAN, R. J.; ATHANASOPOULOS, G. **Forecasting: Principles and Practice**. 3. ed. Melbourne: OTexts, 2021. Disponível em: <https://otexts.com/fpp3/>. Citado na página 13.

KUHNERT, H.; POSSATO, A. Modelagem probabilística de resultados de futebol com distribuição de poisson dupla e simulações de monte carlo. **Revista Brasileira de Estatística Aplicada**, v. 12, n. 1, p. 45–68, 2023. Citado na página 14.

LIMA, B. N. B. de *et al.* Probabilidades no futebol. **Matemática Universitária**, n. 48/49, 2010. Disponível em: <https://rmu.sbm.org.br/wp-content/uploads/sites/27/2018/03/n48_n49_Artigo02.pdf>. Citado 3 vezes nas páginas 13, 14 e 15.

METROPOLIS, N.; ULAM, S. The monte carlo method. **Journal of the American Statistical Association**, New York, v. 44, n. 247, p. 335–341, 1949. Citado na página 12.

MONTGOMERY, D. C.; JOHNSON, L. A.; GARDINER, J. S. **Forecasting and Time Series Analysis**. 2. ed. New York: McGraw-Hill, 2015. Citado na página 13.

MORETTIN, P. A.; TOLOI, C. M. C. **Análise de Séries Temporais**. 3. ed. São Paulo: Blucher, 2018. Citado na página 12.

RAMOS, L. F. P.; LEMOS, H. C. F.; BATISTA, B. D. de O. Modelagem matemática para previsão de jogos de futebol. **Revista Sergipana de Matemática e Educação Matemática**, v. 6, n. 1, p. 46–64, 2021. Disponível em: <https://doi.org/10.34179/revisem.v6i1.13637>. Citado na página 14.

ROSS, S. **Probabilidade: um curso moderno com aplicações**. 8. ed. Porto Alegre: Bookman, 2010. Citado na página 11.

ANEXO A – Tratamento dos dados

```
import pandas as pd
import numpy as np
from unidecode import unidecode

# Dados
df_extraidas_1 = pd.read_csv("C:\\Users\\lmcorreia\\OneDrive -
Stefanini\\Desktop\\campeonato_2013\\datasets\\
odds_extraidas_2013.csv")

df_extraidas_2 = pd.read_csv("C:\\Users\\lmcorreia\\OneDrive -
Stefanini\\Desktop\\campeonato_2013\\datasets\\
odds_extraidas_faltantes_2013.csv")

df_extraidas = pd.concat([df_extraidas_1, df_extraidas_2])
df_extraidas = df_extraidas.dropna().drop_duplicates()

# Remover acentos e converter para minúsculas nas colunas 'mandante' e
'visitante'
df_extraidas['mandante'] = df_extraidas['mandante'].apply(lambda x:
unidecode(x.lower()))
df_extraidas['visitante'] = df_extraidas['visitante'].apply(lambda x:
unidecode(x.lower()))

len(df_extraidas)

df_jogos_datas = pd.read_csv("C:\\Users\\lmcorreia\\OneDrive -
Stefanini\\Desktop\\campeonato_2013\\datasets\\jogos_data.csv")

df_jogos_datas = pd.read_csv("C:\\Users\\lmcorreia\\OneDrive -
Stefanini\\Desktop\\campeonato_2013\\datasets\\jogos_data.csv")
# Converter a coluna 'Date' para o tipo datetime com formato
dia/mês/ano
df_jogos_datas['Date'] = pd.to_datetime(df_jogos_datas['Date'],
format='%d/%m/%Y')

# Filtrar as linhas que estão no ano de 2013
df_2013 = df_jogos_datas[df_jogos_datas['Date'].dt.year == 2013]

df_2013 = df_2013.rename(columns={'Home': 'mandante',
                                  'Away': 'visitante',
                                  'Date': 'data',
                                  'HG':'gols_mandante',
                                  'AG': 'gols_visitante'})

df_2013 = df_2013[['data', 'mandante', 'visitante','gols_mandante',
'gols_visitante', 'Res']]

# Remover acentos e converter para minúsculas nas colunas 'mandante' e
'visitante'
df_2013['mandante'] = df_2013['mandante'].apply(lambda x:
unidecode(x.lower()))
```

```
df_2013['visitante'] = df_2013['visitante'].apply(lambda x:
unidecode(x.lower()))
df_2013['mandante'] = df_2013['mandante'].str.replace(' rj', '',
regex=False)
df_2013['visitante'] = df_2013['visitante'].str.replace(' rj', '',
regex=False)

df_2013.head()

len(df_2013)
```

Tratamento da base de dados das ODDS

```
df_parte1 = pd.merge(df_extraidas,df_2013, on=['mandante',
'visitante'], how='inner')
df_parte1 = df_parte1.dropna()
df_parte1 = df_parte1.drop_duplicates()
len(df_parte1)

df_que_faltam = pd.read_csv('C:\\Users\\lmcorreia\\OneDrive -
Stefanini\\Desktop\\campeonato_2013\\datasets\\odds_que_faltam.txt',
sep='\t')
df_que_faltam = df_que_faltam.drop(df_que_faltam.columns[0], axis=1)

df_completo = pd.concat([df_parte1, df_que_faltam])

df_completo['data'] = pd.to_datetime(df_completo['data'])

len(df_completo)

pd.set_option('display.max_rows', None)
#df_completo

df_ordenado =
df_completo.sort_values(by=['data']).reset_index(drop=True)

#df_ordenado

df_ordenado.isna().sum()

df_ate_280 = df_ordenado.iloc[:280]
df_ate_280.isna().sum()

df = df_ate_280.copy()

len(df)

df.to_csv('C:\\Users\\lmcorreia\\OneDrive - Stefanini\\Desktop\\
campeonato_2013\\datasets\\df_final_treino.csv', index=False,
encoding='utf-8')
```

```
#df_nulos = df_ordenado[df_ordenado.isnull().any(axis=1)]
```

ANEXO B – Suavização exponencial

```
import pandas as pd
import numpy as np

# Dados
df_resultados = pd.read_csv("C:\\Users\\lmcorreia\\OneDrive -
Stefanini\\Desktop\\campeonato_2013\\datasets\\
df_tratado_2013_v2.csv")


df = df_resultados.sort_values(by=['data']).reset_index(drop=True)

len(df)

df.isna().sum()
```

Cálculo das probabilidades da banca

```
df['p_M_ganha_inflada'] = 1 / df['bet365_M_ganha']
df['p_M_empate_inflada'] = 1 / df['bet365_empate']
df['p_M_perde_inflada'] = 1 / df['bet365_M_perde']

df['p_geral'] = (df['p_M_perde_inflada'] + df['p_M_empate_inflada'] +
df['p_M_ganha_inflada'])

df['lucro_medio_banca'] = (df['p_M_perde_inflada'] +
df['p_M_empate_inflada'] + df['p_M_ganha_inflada']) - 1

df['p_M_ganha_banca'] = df['p_M_ganha_inflada']/df['p_geral']
df['p_M_empate_banca'] = df['p_M_empate_inflada']/df['p_geral']
df['p_M_perde_banca'] = df['p_M_perde_inflada']/df['p_geral']

df
```

Treino

```
import copy
df_tratamento = copy.deepcopy(df)
df_tratamento

# Selecionar todos os times
times =
set(df_tratamento['Mandante']).union(set(df_tratamento['Visitante']))

# Todos os times com 0 pontos inicialmente
pontos_times = {time: 0 for time in times}

# Lista com todos os times para contagem de pontos (removendo "_M" e
"_V")
```

```
lista_cont_pontos = [s[:-2] for s in times]

# Times sem o "_M" e "_V" para classificacao final
classificacao_times = {time: 0 for time in lista_cont_pontos}

# Criar dicionario com todos os times iniciando com probabilidade 1/3
e vetor recente zerado (pq nao tiveram jogos ainda)
prob_times = {}
for index, row in df_tratamento.iterrows():
    times = [row['Mandante'], row['Visitante']]
    for time in times:
        if time not in prob_times:
            prob_times[time] = {
                'prob': [1/3, 1/3, 1/3],
                'recente': [0, 0, 0]}

# Dicionario com todas as partidas
partidas = {}
for i in range(len(df_tratamento)):
    partidas[f'jogo_{i+1}'] = [df_tratamento.loc[i, 'Mandante'],
df_tratamento.loc[i, 'Visitante']]


#lista = np.arange(0, 1.05, 0.05).tolist()
#for prob_h in lista:
#    print(prob_h)

lista = np.arange(0, 1.01, 0.01).tolist()

df_delta = pd.DataFrame(lista, columns=['h'])

# Coluna vazia para o delta de cada h
df_delta['delta_total'] = None

df_delta


def atualizar_prob(time):
  h = peso_h
  r = 1-peso_h
  prob_times[time]['prob'] = [(prob_times[time]['prob'][i] * h) +
(prob_times[time]['recente'][i] * r) for i in range(3)]
  if prob_times[time]['recente'] != [0, 0, 0]:
    prob_times[time]['prob'] = [x / (r+h) for x in prob_times[time]
['prob']]

def calcular_probabilidades(M, V):
    p_M_ganhar = (prob_times[M]['prob'][0] + prob_times[V]['prob'][2])
/ 2
    p_M_empatar = (prob_times[M]['prob'][1] + prob_times[V]['prob']
[1]) / 2
```

```
    p_M_perder = (prob_times[M]['prob'][2] + prob_times[V]['prob'][0])
/ 2

    total_prob = p_M_ganhar + p_M_empatar + p_M_perder
    p_M_ganhar /= total_prob
    p_M_empatar /= total_prob
    p_M_perder /= total_prob

    return p_M_ganhar, p_M_empatar, p_M_perder

id_partida = 280

# vetor de prob [ganhou, empatou, perdeu]
for peso_h in lista:
  # Inicializar variaveis
  df_tratamento = copy.deepcopy(df)
  df_tratamento = df_tratamento.head(id_partida)
  print("\n\n")
  print(peso_h)
  prob_times = {}
  for index, row in df_tratamento.iterrows():
      times = [row['Mandante'], row['Visitante']]
      for time in times:
          if time not in prob_times:
              prob_times[time] = {
                  'prob': [1/3, 1/3, 1/3],
                  'recente': [0, 0, 0]}

  # Partidas que ocorreram
  for i in range(id_partida):
    M = df_tratamento.loc[i, 'Mandante']
    V = df_tratamento.loc[i, 'Visitante']
    gols_mandante = df_tratamento.loc[i, 'gols_mandante']
    gols_visitante = df_tratamento.loc[i, 'gols_visitante']

    print("Partida: ", i)
    print("Mandante: ", M)
    print("Visitante: ", V)
    print("Data do jogo:", df_tratamento.loc[i, 'data'])

    # Calcular probabilidade
    p_M_ganhar, p_M_empatar, p_M_perder = calcular_probabilidades(M,
V)
    print("Prob M ganhar:", p_M_ganhar)
    print("Prob M empatar:", p_M_empatar)
    print("Prob M perder:", p_M_perder)

    # Salvar diferenças
    if pd.isna(df_tratamento.loc[i, 'p_M_ganha_banca']):
      df_tratamento.loc[i,'delta'] = 0
```

```
    else: df_tratamento.loc[i,'delta'] = abs(df_tratamento.loc[i,
'p_M_ganha_banca'] - p_M_ganhar) + abs(df_tratamento.loc[i,
'p_M_empate_banca'] - p_M_empatar) + abs(df_tratamento.loc[i,
'p_M_perde_banca'] - p_M_perder)

    print("Prob M ganhar (BANCA):", df_tratamento.loc[i,
'p_M_ganha_banca'])
    print("Prob M empatar (BANCA):", df_tratamento.loc[i,
'p_M_empate_banca'])
    print("Prob M perder (BANCA):", df_tratamento.loc[i,
'p_M_perde_banca'])
    print("DELTA:", df_tratamento.loc[i,'delta'], "\n")


    if gols_mandante > gols_visitante:
      prob_times[M]['recente'] = [1, 0, 0] # Mandante ganhou
      prob_times[V]['recente'] = [0, 0, 1] # Visitante perdeu
      pontos_times[M] += 3

    elif gols_mandante < gols_visitante:
      prob_times[M]['recente'] = [0, 0, 1] # Mandante perdeu
      prob_times[V]['recente'] = [1, 0, 0] # Visitante ganhou
      pontos_times[V] += 3

    else:
      prob_times[M]['recente'] = [0, 1, 0] # Mandante empatou
      prob_times[V]['recente'] = [0, 1, 0] # Visitante empatou
      pontos_times[M] += 1
      pontos_times[V] += 1

    atualizar_prob(M)
    atualizar_prob(V)
  print(df_tratamento)
  #df_tratamento.fillna(0, inplace=True)
  print("Apos preencher NA")
  print(df_tratamento)
  df_delta.loc[df_delta['h'] == peso_h, 'delta_total'] =
sum(df_tratamento['delta'])


pd.set_option('display.max_rows', None)  # Exibir todas as linhas
pd.set_option('display.max_columns', None)  # Exibir todas as colunas

df_delta

import matplotlib.pyplot as plt
# Criando o gráfico
plt.figure(figsize=(10,6))
plt.plot(df_delta['h'], df_delta['delta_total'], marker='o',
linestyle='-', color='b', label='Delta', markersize=3)
# Adicionando uma linha vertical passando pelo valor 0.9 no eixo X
```

```
plt.axvline(x=0.9, color='r', linestyle='--', label='h=0.9 (menor
delta)')


# Adicionando título e rótulos aos eixos
plt.title('Delta total por h')
plt.xlabel('h')
plt.ylabel('delta')

plt.grid(True)
# Exibindo a legenda
plt.legend()

# Exibindo o gráfico
plt.show()
```

h será a 0.9 pois tem o menor delta

ANEXO C – Simulação do campeonato

Simulação da Classificação Final do Campeonato de 2013

```
#import requests
#from bs4 import BeautifulSoup
import pandas as pd
#import requests
#import os
#from dotenv import load_dotenv
#from google.colab import files
#import random
#import numpy as np
from unidecode import unidecode
import numpy as np
from collections import Counter
import random


df_jogos_datas = pd.read_csv("C:\\Users\\lmcorreia\\OneDrive -
Stefanini\\Desktop\\campeonato_2013\\datasets\\jogos_data.csv")
# Converter a coluna 'Date' para o tipo datetime com formato
dia/mês/ano
df_jogos_datas['Date'] = pd.to_datetime(df_jogos_datas['Date'],
format='%d/%m/%Y')

# Filtrar as linhas que estão no ano de 2013
df_2013 = df_jogos_datas[df_jogos_datas['Date'].dt.year == 2013]

df_2013 = df_2013.rename(columns={'Home': 'Mandante',
                                  'Away': 'Visitante',
                                  'Date': 'data',
                                  'HG':'gols_mandante',
                                  'AG': 'gols_visitante'})

df_2013 = df_2013[['data', 'Mandante', 'Visitante','gols_mandante',
'gols_visitante', 'Res']]

# Remover acentos e converter para minúsculas nas colunas 'mandante' e
'visitante'
df_2013['Mandante'] = df_2013['Mandante'].apply(lambda x:
unidecode(x.lower()))
df_2013['Visitante'] = df_2013['Visitante'].apply(lambda x:
unidecode(x.lower()))
df_2013['Mandante'] = df_2013['Mandante'].str.replace(' rj', '',
regex=False)
df_2013['Visitante'] = df_2013['Visitante'].str.replace(' rj', '',
regex=False)
```

```
df_2013.head()

#df_tratamento = df_completo.dropna()

# Ordenar pela data
df_tratamento =
df_2013.sort_values(by=['data']).reset_index(drop=True)

# Adicionar "_M" e "_V" nos nomes dos times
df_tratamento['Mandante'] = df_tratamento['Mandante'].astype(str) +
'_M'
df_tratamento['Visitante'] = df_tratamento['Visitante'].astype(str) +
'_V'

df_tratamento.head()

len(df_tratamento)
```

Funções

```
# Funções
def atualizar_prob(time):
  h=0.9
  r=1-h
  prob_times[time]['prob'] = [(prob_times[time]['prob'][i] * h) +
(prob_times[time]['recente'][i] * r) for i in range(3)]
  if prob_times[time]['recente'] != [0, 0, 0]:
    prob_times[time]['prob'] = [x / (r+h) for x in prob_times[time]
['prob']]

def calcular_probabilidades(M, V):
    p_M_ganhar = (prob_times[M]['prob'][0] + prob_times[V]['prob'][2])
/ 2
    p_M_empatar = (prob_times[M]['prob'][1] + prob_times[V]['prob']
[1]) / 2
    p_M_perder = (prob_times[M]['prob'][2] + prob_times[V]['prob'][0])
/ 2

    total_prob = p_M_ganhar + p_M_empatar + p_M_perder
    p_M_ganhar /= total_prob
    p_M_empatar /= total_prob
    p_M_perder /= total_prob

    return p_M_ganhar, p_M_empatar, p_M_perder

def calcular_odds(p_M_ganhar, p_M_empatar, p_M_perder, infl):

   odds_M_ganhar = 1 / (p_M_ganhar + (infl * p_M_ganhar))
```

```
   odds_empatar = 1 / (p_M_empatar + (infl * p_M_empatar))
   odds_M_perder = 1 / (p_M_perder + (infl * p_M_perder))

   return odds_M_ganhar, odds_empatar, odds_M_perder

# Selecionar todos os times
times =
set(df_tratamento['Mandante']).union(set(df_tratamento['Visitante']))

# Todos os times com 0 pontos inicialmente
pontos_times = {time: 0 for time in times}

# Lista com todos os times para contagem de pontos (removendo "_M" e
"_V")
lista_cont_pontos = [s[:-2] for s in times]

# Times sem o "_M" e "_V" para classificacao final
classificacao_times = {time: 0 for time in lista_cont_pontos}

# Criar dicionario com todos os times iniciando com probabilidade 1/3
e vetor recente zerado (pq nao tiveram jogos ainda)
prob_times = {}
for index, row in df_tratamento.iterrows():
    times = [row['Mandante'], row['Visitante']]
    for time in times:
        if time not in prob_times:
            prob_times[time] = {
                'prob': [1/3, 1/3, 1/3],
                'recente': [0, 0, 0]}

# Dicionario com todas as partidas
partidas = {}
for i in range(len(df_tratamento)):
    partidas[f'jogo_{i+1}'] = [df_tratamento.loc[i, 'Mandante'],
df_tratamento.loc[i, 'Visitante']]


# Criar um DataFrame com as linhas como os times e as colunas como
p_rodada_30, ..., p_rodada_37
rodadas = [f'p_rodada_{rodada}' for rodada in range(29, 39)]
df_prob_classificacao_final =
pd.DataFrame(index=classificacao_times.keys(), columns=rodadas)
df_prob_classificacao_final[:] = 0 # Preencher o DataFrame com 0
df_prob_classificacao_final
```

Treino até a 28ª rodada

Obter as probabilidades.

```
id_partida = 280
rep = 1000000
#col = 0

# vetor de prob [ganhou, empatou, perdeu]

# Partidas que ocorreram
for i in range(id_partida):
  M = df_tratamento.loc[i, 'Mandante']
  V = df_tratamento.loc[i, 'Visitante']
  gols_mandante = df_tratamento.loc[i, 'gols_mandante']
  gols_visitante = df_tratamento.loc[i, 'gols_visitante']

  if gols_mandante > gols_visitante:
    prob_times[M]['recente'] = [1, 0, 0] # Mandante ganhou
    prob_times[V]['recente'] = [0, 0, 1] # Visitante perdeu
    pontos_times[M] += 3

  elif gols_mandante < gols_visitante:
    prob_times[M]['recente'] = [0, 0, 1] # Mandante perdeu
    prob_times[V]['recente'] = [1, 0, 0] # Visitante ganhou
    pontos_times[V] += 3

  else:
    prob_times[M]['recente'] = [0, 1, 0] # Mandante empatou
    prob_times[V]['recente'] = [0, 1, 0] # Visitante empatou
    pontos_times[M] += 1
    pontos_times[V] += 1

  atualizar_prob(M)
  atualizar_prob(V)

prob_times_ocorreram = prob_times
pontos_times_ocorreram = pontos_times
```

ADICIONAR

- Adicionar as odds até o jogo 280 e calcular o H
- Alterar o grafico adicionando a linha vermelha com o esperado 300*p.

Simulação

```
import copy

def combinar(dicionario):
    #Soma resultados do mandante e visitante
    novo_dicionario = {}
```

```
    for time_, valor in dicionario.items():
        time = time_.rsplit('_', 1)[0]
        if time in novo_dicionario:
            novo_dicionario[time] += valor
        else:
            novo_dicionario[time] = valor

    return novo_dicionario


# Simulação
jogos_para_simulacao = list(partidas.keys())[id_partida:]

for j in range(rep):
  prob_times = copy.deepcopy(prob_times_ocorreram)
  pontos_times = copy.deepcopy(pontos_times_ocorreram)
  # vetor prob [ganhar, empatar, perder]
  print("rodada:", j)

  for i in jogos_para_simulacao:

    M = list(partidas[i])[0]
    V = list(partidas[i])[1]

    p_M_ganhar, p_M_empatar, p_M_perder = calcular_probabilidades(M,
V)

    unif = np.random.uniform(size=1)
    if unif < p_M_ganhar:
      prob_times[M]['recente'] = [1, 0, 0] # Mandante ganhou
      prob_times[V]['recente'] = [0, 0, 1] # Visitante perdeu
      pontos_times[M] += 3

    elif (unif >= p_M_ganhar) & (unif < (p_M_ganhar + p_M_empatar)) :
      prob_times[M]['recente'] = [0, 1, 0] # Mandante empatou
      prob_times[V]['recente'] = [0, 1, 0] # Visitante empatou
      pontos_times[M] += 1
      pontos_times[V] += 1
    else:
      prob_times[M]['recente'] = [0, 0, 1] # Mandante perdeu
      prob_times[V]['recente'] = [1, 0, 0] # Visitante ganhou
      pontos_times[V] += 3


    atualizar_prob(M)
    atualizar_prob(V)

  pontos_times_totais = combinar(pontos_times)

  # Maior quantidade de pontos
```

```
  max_pontos = max(pontos_times_totais.values())

  # Todos os times com max quantidade de pontos
  times_com_max_pontos = [time for time, pontos in
pontos_times_totais.items() if pontos == max_pontos]

  # Sortear
  time_sorteado = random.choice(times_com_max_pontos)

  classificacao_times[time_sorteado] += 1

# Calcular a probabilidade
classificacao_times_prob = {chave: valor / rep for chave, valor in
classificacao_times.items()}

classificacao_times_prob
```

simula o resto do campeonato probabilidade de titulo de cada titulo

em media qunto cada equipe vai alcançar

monta uma tabela com a probabilidade de titulo de cada time tabela com a probabilidade de rebaixamento

tabela com a pontuação media de cada time tabela com a pontuação do campeao

```
classificacao_times_prob

import copy
import pandas as pd  # adicionado para criar DataFrame no final

def combinar(dicionario):
    #Soma resultados do mandante e visitante
    novo_dicionario = {}

    for time_, valor in dicionario.items():
        time = time_.rsplit('_', 1)[0]
        if time in novo_dicionario:
            novo_dicionario[time] += valor
        else:
            novo_dicionario[time] = valor

    return novo_dicionario


# Simulação
jogos_para_simulacao = list(partidas.keys())[id_partida:]

# --- ADIÇÃO: inicializar acumulador de pontos para calcular média ---
# usa as chaves combinadas para garantir nomes consistentes (sem
sufixos)
```

```
soma_pontos = {time: 0 for time in
combinar(pontos_times_ocorreram).keys()}

for j in range(rep):
  prob_times = copy.deepcopy(prob_times_ocorreram)
  pontos_times = copy.deepcopy(pontos_times_ocorreram)
  # vetor prob [ganhar, empatar, perder]
  print("rodada:", j)

  for i in jogos_para_simulacao:

    M = list(partidas[i])[0]
    V = list(partidas[i])[1]

    p_M_ganhar, p_M_empatar, p_M_perder = calcular_probabilidades(M,
V)

    unif = np.random.uniform(size=1)
    if unif < p_M_ganhar:
      prob_times[M]['recente'] = [1, 0, 0] # Mandante ganhou
      prob_times[V]['recente'] = [0, 0, 1] # Visitante perdeu
      pontos_times[M] += 3

    elif (unif >= p_M_ganhar) & (unif < (p_M_ganhar + p_M_empatar)) :
      prob_times[M]['recente'] = [0, 1, 0] # Mandante empatou
      prob_times[V]['recente'] = [0, 1, 0] # Visitante empatou
      pontos_times[M] += 1
      pontos_times[V] += 1
    else:
      prob_times[M]['recente'] = [0, 0, 1] # Mandante perdeu
      prob_times[V]['recente'] = [1, 0, 0] # Visitante ganhou
      pontos_times[V] += 3


    atualizar_prob(M)
    atualizar_prob(V)

  pontos_times_totais = combinar(pontos_times)

  # --- ADIÇÃO: acumular os pontos desta simulação ---
  for time, pontos in pontos_times_totais.items():
      soma_pontos[time] += pontos

  # Maior quantidade de pontos
  max_pontos = max(pontos_times_totais.values())

  # Todos os times com max quantidade de pontos
  times_com_max_pontos = [time for time, pontos in
pontos_times_totais.items() if pontos == max_pontos]
```

```
  # Sortear
  time_sorteado = random.choice(times_com_max_pontos)

  classificacao_times[time_sorteado] += 1

# Calcular a probabilidade
classificacao_times_prob = {chave: valor / rep for chave, valor in
classificacao_times.items()}

# --- ADIÇÃO: calcular média de pontos por time (soma / rep) ---
media_pontos = {time: soma / rep for time, soma in
soma_pontos.items()}

# Opcional: transformar em DataFrame para visualização ordenada
df_media = pd.DataFrame({
    'Time': list(media_pontos.keys()),
    'Media_Pontos': list(media_pontos.values())
}).sort_values(by='Media_Pontos',
ascending=False).reset_index(drop=True)

# imprimir/usar os resultados
print(df_media)
# variáveis produzidas:
# - media_pontos (dict): média de pontos por time
# - df_media (DataFrame): tabela ordenada com médias

df_media

# --- ADIÇÃO: lista para guardar a probabilidade/pontos do Cruzeiro
após combinar ---
prob_cruzeiro_simulacoes = []

for j in range(rep):
    prob_times = copy.deepcopy(prob_times_ocorreram)
    pontos_times = copy.deepcopy(pontos_times_ocorreram)
    # vetor prob [ganhar, empatar, perder]
    print("rodada:", j)

    for i in jogos_para_simulacao:
        M = list(partidas[i])[0]
        V = list(partidas[i])[1]

        p_M_ganhar, p_M_empatar, p_M_perder =
calcular_probabilidades(M, V)

        unif = np.random.uniform(size=1)
        if unif < p_M_ganhar:
            prob_times[M]['recente'] = [1, 0, 0] # Mandante ganhou
            prob_times[V]['recente'] = [0, 0, 1] # Visitante perdeu
            pontos_times[M] += 3
```

```
        elif (unif >= p_M_ganhar) & (unif < (p_M_ganhar +
p_M_empatar)) :
            prob_times[M]['recente'] = [0, 1, 0] # Mandante empatou
            prob_times[V]['recente'] = [0, 1, 0] # Visitante empatou
            pontos_times[M] += 1
            pontos_times[V] += 1
        else:
            prob_times[M]['recente'] = [0, 0, 1] # Mandante perdeu
            prob_times[V]['recente'] = [1, 0, 0] # Visitante ganhou
            pontos_times[V] += 3

        atualizar_prob(M)
        atualizar_prob(V)

    pontos_times_totais = combinar(pontos_times)

    # --- ADIÇÃO: salvar os pontos do Cruzeiro após combinar mandante
e visitante ---
    prob_cruzeiro_simulacoes.append(pontos_times_totais['cruzeiro'])

    max_pontos = max(pontos_times_totais.values())
    times_com_max_pontos = [time for time, pontos in
pontos_times_totais.items() if pontos == max_pontos]
    time_sorteado = random.choice(times_com_max_pontos)
    classificacao_times[time_sorteado] += 1

# Probabilidade final
classificacao_times_prob = {chave: valor / rep for chave, valor in
classificacao_times.items()}

classificacao_times

classificacao_times_prob
```

ANEXO D – Lucro da banca

Previsão da Classificação Final do Campeonato de 2009

```
#import requests
#from bs4 import BeautifulSoup
import pandas as pd
#from collections import Counter
#import requests
#import os
#from dotenv import load_dotenv
#from google.colab import files
#import random
#import numpy as np
from unidecode import unidecode
import numpy as np

# Dados
df_odds1 = pd.read_csv("C:\\Users\\lmcorreia\\OneDrive - Stefanini\\
Desktop\\campeonato_2013\\datasets\\odds_extraidas_2013.csv")
#df_odds2 = pd.read_csv("C:\\Users\\lmcorreia\\OneDrive - Stefanini\\
Desktop\\campeonato_2013\\datasets\\
odds_extraidas_faltantes_2013.csv")
df_jogos_datas = pd.read_csv("C:\\Users\\lmcorreia\\OneDrive -
Stefanini\\Desktop\\campeonato_2013\\datasets\\jogos_data.csv")

# Converter a coluna 'Date' para o tipo datetime com formato
dia/mês/ano
df_jogos_datas['Date'] = pd.to_datetime(df_jogos_datas['Date'],
format='%d/%m/%Y')

# Filtrar as linhas que estão no ano de 2013
df_2013 = df_jogos_datas[df_jogos_datas['Date'].dt.year == 2013]

df_2013 = df_2013.rename(columns={'Home': 'mandante',
                                  'Away': 'visitante',
                                  'Date': 'data',
                                  'HG':'gols_mandante',
                                  'AG': 'gols_visitante'})

df_2013 = df_2013[['data', 'mandante', 'visitante','gols_mandante',
'gols_visitante', 'Res']]

# Remover acentos e converter para minúsculas nas colunas 'mandante' e
'visitante'
df_odds1['mandante'] = df_odds1['mandante'].apply(lambda x:
unidecode(x.lower()))
df_odds1['visitante'] = df_odds1['visitante'].apply(lambda x:
```

```
unidecode(x.lower()))

df_2013['mandante'] = df_2013['mandante'].apply(lambda x:
unidecode(x.lower()))
df_2013['visitante'] = df_2013['visitante'].apply(lambda x:
unidecode(x.lower()))


#odds = pd.concat([df_odds1,df_odds2], ignore_index=True)
#len(odds)
#odds = odds.drop_duplicates(keep='first')
#len(odds)
#

df_completo = pd.merge(df_2013, df_odds1, on=['mandante',
'visitante'], how='left')

df_completo = df_completo.rename(columns={'mandante': 'Mandante',
                                    'visitante': 'Visitante'})

len(df_completo)

# Contar NaN por coluna
na_por_coluna = df_completo.isna().sum()
print(na_por_coluna)

from IPython.display import display
# Ajustar as configurações do pandas
pd.set_option('display.max_rows', None)  # Exibir todas as linhas
pd.set_option('display.max_columns', None)  # Exibir todas as colunas
# Exibir o DataFrame
#display(df_completo)

# Dados
df_completo = pd.read_csv("C:\\Users\\lmcorreia\\OneDrive -
Stefanini\\Desktop\\campeonato_2013\\datasets\\df_final_treino.csv")
df_completo = df_completo.rename(columns={'mandante': 'Mandante',
                                    'visitante': 'Visitante'})

#df_tratamento = df_completo.dropna()

# Ordenar pela data
df_tratamento =
df_completo.sort_values(by=['data']).reset_index(drop=True)

# Adicionar "_M" e "_V" nos nomes dos times
df_tratamento['Mandante'] = df_tratamento['Mandante'].astype(str) +
'_M'
df_tratamento['Visitante'] = df_tratamento['Visitante'].astype(str) +
'_V'
```

```
df_tratamento.head()

df_tratamento.to_csv("C:\\Users\\lmcorreia\\OneDrive - Stefanini\\
Desktop\\campeonato_2013\\datasets\\df_tratado_2013_v2.csv")

# Dados
df_tratamento = pd.read_csv("C:\\Users\\lmcorreia\\OneDrive -
Stefanini\\Desktop\\campeonato_2013\\datasets\\df_tratado_2013.csv")

len(df_tratamento)
```

Funções

```
# Funções
def atualizar_prob(time):
  h=0.9
  r=1-h
  prob_times[time]['prob'] = [(prob_times[time]['prob'][i] * h) +
(prob_times[time]['recente'][i] * r) for i in range(3)]
  if prob_times[time]['recente'] != [0, 0, 0]:
    prob_times[time]['prob'] = [x / (r+h) for x in prob_times[time]
['prob']]

def calcular_probabilidades(M, V):
    p_M_ganhar = (prob_times[M]['prob'][0] + prob_times[V]['prob'][2])
/ 2
    p_M_empatar = (prob_times[M]['prob'][1] + prob_times[V]['prob']
[1]) / 2
    p_M_perder = (prob_times[M]['prob'][2] + prob_times[V]['prob'][0])
/ 2

    total_prob = p_M_ganhar + p_M_empatar + p_M_perder
    p_M_ganhar /= total_prob
    p_M_empatar /= total_prob
    p_M_perder /= total_prob

    return p_M_ganhar, p_M_empatar, p_M_perder

def calcular_odds(p_M_ganhar, p_M_empatar, p_M_perder, infl):

   odds_M_ganhar = 1 / (p_M_ganhar + (infl * p_M_ganhar))
   odds_empatar = 1 / (p_M_empatar + (infl * p_M_empatar))
   odds_M_perder = 1 / (p_M_perder + (infl * p_M_perder))

   return odds_M_ganhar, odds_empatar, odds_M_perder

# Selecionar todos os times
times =
```

```
set(df_tratamento['Mandante']).union(set(df_tratamento['Visitante']))

# Todos os times com 0 pontos inicialmente
pontos_times = {time: 0 for time in times}

# Lista com todos os times para contagem de pontos (removendo "_M" e
"_V")
lista_cont_pontos = [s[:-2] for s in times]

# Times sem o "_M" e "_V" para classificacao final
classificacao_times = {time: 0 for time in lista_cont_pontos}

# Criar dicionario com todos os times iniciando com probabilidade 1/3
e vetor recente zerado (pq nao tiveram jogos ainda)
prob_times = {}
for index, row in df_tratamento.iterrows():
    times = [row['Mandante'], row['Visitante']]
    for time in times:
        if time not in prob_times:
            prob_times[time] = {
                'prob': [1/3, 1/3, 1/3],
                'recente': [0, 0, 0]}

# Dicionario com todas as partidas
partidas = {}
for i in range(len(df_tratamento)):
    partidas[f'jogo_{i+1}'] = [df_tratamento.loc[i, 'Mandante'],
df_tratamento.loc[i, 'Visitante']]


# Criar um DataFrame com as linhas como os times e as colunas como
p_rodada_30, ..., p_rodada_37
#rodadas = [f'p_rodada_{rodada}' for rodada in range(29, 39)]
#df_prob_classificacao_final =
pd.DataFrame(index=classificacao_times.keys(), columns=rodadas)
#df_prob_classificacao_final[:] = 0 # Preencher o DataFrame com 0
#df_prob_classificacao_final
```

Treino até a 28ª rodada

Obter as probabilidades.

```
id_partida = 290
#rep = 10000
#col = 0

# vetor de prob [ganhou, empatou, perdeu]
```

```
# Partidas que ocorreram
for i in range(id_partida):
  print(i+1)
  M = df_tratamento.loc[i, 'Mandante']
  V = df_tratamento.loc[i, 'Visitante']
  gols_mandante = df_tratamento.loc[i, 'gols_mandante']
  gols_visitante = df_tratamento.loc[i, 'gols_visitante']

  if gols_mandante > gols_visitante:
    prob_times[M]['recente'] = [1, 0, 0] # Mandante ganhou
    prob_times[V]['recente'] = [0, 0, 1] # Visitante perdeu
    pontos_times[M] += 3

  elif gols_mandante < gols_visitante:
    prob_times[M]['recente'] = [0, 0, 1] # Mandante perdeu
    prob_times[V]['recente'] = [1, 0, 0] # Visitante ganhou
    pontos_times[V] += 3

  else:
    prob_times[M]['recente'] = [0, 1, 0] # Mandante empatou
    prob_times[V]['recente'] = [0, 1, 0] # Visitante empatou
    pontos_times[M] += 1
    pontos_times[V] += 1

  atualizar_prob(M)
  atualizar_prob(V)

prob_times_ocorreram = prob_times
pontos_times_ocorreram = pontos_times

import copy
import matplotlib.pyplot as plt

# vetor de prob [ganhou, empatou, perdeu]
resultados = pd.DataFrame(columns=["j", "lucro_total"])
# Partidas que ocorreram
for j in np.arange(0.01, 0.21, 0.01):
  j=round(j, 2)
  lucro_total = 0
  valor_total = 0

  prob_times = copy.deepcopy(prob_times_ocorreram)
  pontos_times = copy.deepcopy(pontos_times_ocorreram)

  for i in range(id_partida,380):

    lucro = 0

    M = df_tratamento.loc[i, 'Mandante']
    V = df_tratamento.loc[i, 'Visitante']
```

```
    gols_mandante = df_tratamento.loc[i, 'gols_mandante']
    gols_visitante = df_tratamento.loc[i, 'gols_visitante']

    p_M_ganhar, p_M_empatar, p_M_perder = calcular_probabilidades(M,
V)

    odds_M_ganhar, odds_empatar, odds_M_perder =
calcular_odds(p_M_ganhar, p_M_empatar, p_M_perder, j)

    if gols_mandante > gols_visitante:
      prob_times[M]['recente'] = [1, 0, 0] # Mandante ganhou
      prob_times[V]['recente'] = [0, 0, 1] # Visitante perdeu
      lucro = 3 - odds_M_ganhar

    elif gols_mandante < gols_visitante:
      prob_times[M]['recente'] = [0, 0, 1] # Mandante perdeu
      prob_times[V]['recente'] = [1, 0, 0] # Visitante ganhou
      lucro = 3 - odds_M_perder

    else:
      prob_times[M]['recente'] = [0, 1, 0] # Mandante empatou
      prob_times[V]['recente'] = [0, 1, 0] # Visitante empatou
      lucro = 3 - odds_empatar

    lucro_total = lucro + lucro_total
    valor_total = 3 + valor_total

    atualizar_prob(M)
    atualizar_prob(V)
  print(j, lucro_total)

  # Salvando os valores de j e lucro_total no DataFrame
  resultados = pd.concat([resultados, pd.DataFrame({"j": [round(j,
2)], "lucro_total": [lucro_total]})], ignore_index=True)

# Exibindo o DataFrame
print(resultados)

resultados

plt.figure(figsize=(11, 6))  # largura = 12, altura = 6
# Gerando o gráfico
plt.plot(resultados['j'], resultados['lucro_total'], marker='o',
label='Observado')
# Criar vetor i de 0.01 até 0.2 (pulando de 0.01)
i = np.arange(0.01, 0.21, 0.01)

# Calcular 300 / i
linha_ref = i*300
```

```
# Adicionar a nova linha
plt.plot(i, linha_ref, color='red', linestyle='--', label='Esperado')
plt.xlabel('Percentual')
plt.ylabel('Lucro Total')
plt.title('Lucro obtido X Lucro esperado em função do percentual de
lucro esperado pela banca')
plt.grid(True)
plt.legend()  # <--- adiciona legenda automática
plt.show()

import pandas as pd
import matplotlib.pyplot as plt

# Dados fornecidos
dados = {
    'lucro_esperado': [
        0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.10,
        0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.20
    ],
    'lucro_total': [
        2.955049, 5.573137, 8.140388, 10.658269, 13.128190, 15.551509,
17.929533, 20.263518,
        22.554679, 24.804182, 27.013153, 29.182679, 31.313805,
33.407544, 35.464870,
        37.486724, 39.474017, 41.427627, 43.348403, 45.237167
    ]
}

# Criar DataFrame
df = pd.DataFrame(dados)

# Calcular proporção (lucro obtido / lucro esperado * 300)
df['lucro_esperado_valor'] = df['lucro_esperado'] * 300
df['proporcao'] = df['lucro_total'] / df['lucro_esperado_valor']

# Mostrar tabela
print(df)

# --- Gráfico ---
plt.figure(figsize=(10, 5))
plt.plot(df['lucro_esperado']*100, df['proporcao'], marker='o',
color='blue', label='Proporção Observada')
#plt.axhline(1, color='red', linestyle='--', label='Esperado
(proporção = 1)')
plt.xlabel('Percentual de lucro esperado pela banca (%)')
plt.ylabel('Proporção (Lucro obtido / Lucro esperado)')
#plt.title('Proporção do lucro obtido em relação ao lucro esperado')
plt.grid(True)
plt.legend()
```

```
plt.tight_layout()
plt.show()
```

```
import pandas as pd
import matplotlib.pyplot as plt

# Dados fornecidos
dados = {
    'lucro_esperado': [
        0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.10,
        0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.20
    ],
    'lucro_total': [
        2.955049, 5.573137, 8.140388, 10.658269, 13.128190, 15.551509,
17.929533, 20.263518,
        22.554679, 24.804182, 27.013153, 29.182679, 31.313805,
33.407544, 35.464870,
        37.486724, 39.474017, 41.427627, 43.348403, 45.237167
    ]
}

# Criar DataFrame
df = pd.DataFrame(dados)

# Calcular proporção (lucro obtido / lucro esperado * 300)
df['lucro_esperado_valor'] = df['lucro_esperado'] * 300
df['proporcao'] = df['lucro_total'] / df['lucro_esperado_valor']

# Mostrar tabela
print(df)

# --- Gráfico ---
plt.figure(figsize=(10, 5))
plt.plot(df['lucro_esperado']*100, df['proporcao'], marker='o',
color='blue', label='Proporção Observada')
#plt.axhline(1, color='red', linestyle='--', label='Esperado
(proporção = 1)')
plt.xlabel('Percentual de lucro esperado pela banca (%)')
plt.ylabel('Proporção (Lucro obtido / Lucro esperado)')
#plt.title('Proporção do lucro obtido em relação ao lucro esperado')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

ADICIONAR

- Adicionar as odds até o jogo 280 e calcular o H
- Alterar o grafico adicionando a linha vermelha com o esperado 300*p.

```python
import copy
# vetor de prob [ganhou, empatou, perdeu]

# Partidas que ocorreram
lucro_total = 0
valor_total = 0


prob_times = copy.deepcopy(prob_times_ocorreram)
pontos_times = copy.deepcopy(pontos_times_ocorreram)
for i in range(id_partida,380):
  print(i+1)

  lucro = 0

  M = df_tratamento.loc[i, 'Mandante']
  V = df_tratamento.loc[i, 'Visitante']
  gols_mandante = df_tratamento.loc[i, 'gols_mandante']
  gols_visitante = df_tratamento.loc[i, 'gols_visitante']


  p_M_ganhar, p_M_empatar, p_M_perder = calcular_probabilidades(M, V)

  odds_M_ganhar, odds_empatar, odds_M_perder =
calcular_odds(p_M_ganhar, p_M_empatar, p_M_perder, j)

  if gols_mandante > gols_visitante:
    prob_times[M]['recente'] = [1, 0, 0] # Mandante ganhou
    prob_times[V]['recente'] = [0, 0, 1] # Visitante perdeu
    lucro = 3 - odds_M_ganhar

  elif gols_mandante < gols_visitante:
    prob_times[M]['recente'] = [0, 0, 1] # Mandante perdeu
    prob_times[V]['recente'] = [1, 0, 0] # Visitante ganhou
    lucro = 3 - odds_M_perder

  else:
    prob_times[M]['recente'] = [0, 1, 0] # Mandante empatou
    prob_times[V]['recente'] = [0, 1, 0] # Visitante empatou
    lucro = 3 - odds_empatar

  lucro_total = lucro + lucro_total
  valor_total = 3 + valor_total

  atualizar_prob(M)
  atualizar_prob(V)

print(lucro_total)

import copy
# vetor de prob [ganhou, empatou, perdeu]
```

```
# Partidas que ocorreram
lucro_total = 0
valor_total = 0


prob_times = copy.deepcopy(prob_times_ocorreram)
pontos_times = copy.deepcopy(pontos_times_ocorreram)
for i in range(id_partida,380):

  lucro = 0

  M = df_tratamento.loc[i, 'Mandante']
  V = df_tratamento.loc[i, 'Visitante']
  gols_mandante = df_tratamento.loc[i, 'gols_mandante']
  gols_visitante = df_tratamento.loc[i, 'gols_visitante']


  p_M_ganhar, p_M_empatar, p_M_perder = calcular_probabilidades(M, V)

  odds_M_ganhar, odds_empatar, odds_M_perder =
calcular_odds(p_M_ganhar, p_M_empatar, p_M_perder, j)

  if gols_mandante > gols_visitante:
    prob_times[M]['recente'] = [1, 0, 0] # Mandante ganhou
    prob_times[V]['recente'] = [0, 0, 1] # Visitante perdeu
    lucro = 3 - odds_M_ganhar

  elif gols_mandante < gols_visitante:
    prob_times[M]['recente'] = [0, 0, 1] # Mandante perdeu
    prob_times[V]['recente'] = [1, 0, 0] # Visitante ganhou
    lucro = 3 - odds_M_perder

  else:
    prob_times[M]['recente'] = [0, 1, 0] # Mandante empatou
    prob_times[V]['recente'] = [0, 1, 0] # Visitante empatou
    lucro = 3 - odds_empatar

  lucro_total = lucro + lucro_total
  valor_total = 3 + valor_total

  atualizar_prob(M)
  atualizar_prob(V)

print(lucro_total)
```

Inflar probabilidades

As novas probabilidades serão p + 0,1*p (infladas em 10%).

Simulação

```
import copy

def combinar(dicionario):
    #Soma resultados do mandante e visitante
    novo_dicionario = {}

    for time_, valor in dicionario.items():
        time = time_.rsplit('_', 1)[0]
        if time in novo_dicionario:
            novo_dicionario[time] += valor
        else:
            novo_dicionario[time] = valor

    return novo_dicionario

def calcular_odds(p_M_ganhar, p_M_empatar, p_M_perder):

   odds_M_ganhar = 1 / (p_M_ganhar + (0.1 * p_M_ganhar))
   odds_empatar = 1 / (p_M_empatar + (0.1 * p_M_empatar))
   odds_M_perder = 1 / (p_M_perder + (0.1 * p_M_perder))

   return odds_M_ganhar, odds_empatar, odds_M_perder


# Simulação
jogos_para_simulacao = list(partidas.keys())[id_partida:]

lucro_total = 0

# Iterando pelas linhas
for index, row in df_tratamento.iloc[id_partida:].iterrows():

  M = row['Mandante']
  V = row['Visitante']

  print(M)
  print(V)

  p_M_ganhar, p_M_empatar, p_M_perder = calcular_probabilidades(M, V)

  odds_M_ganhar, odds_empatar, odds_M_perder =
calcular_odds(p_M_ganhar, p_M_empatar, p_M_perder)

  if row.loc['Res'] == 'H':
    lucro = 3 - odds_M_ganhar
    print("Mandante ganhou\n")

  elif row.loc['Res'] == 'A':
    lucro = 3 - odds_M_perder
```

```
    print("Mandante perdeu\n")

  else:
    lucro = 3 - odds_empatar
    print("Empate\n")

  lucro_total = lucro_total + lucro

  atualizar_prob(M)
  atualizar_prob(V)

# Calcular a probabilidade
#classificacao_times_prob = {chave: valor / rep for chave, valor in
classificacao_times.items()}
```