Predição de Sucesso Eleitoral para Deputado Federal: Uma Abordagem Comparativa de Classificação Supervisionada com Validação Temporal em Dados Desbalanceados

Artur Garcia
Artur Paschoal
Universidade Federal do Ceará (UFC)
Disciplina: Aprendizagem de Máquina (2025.2)
Professor: César Lincoln Cavalcante Mattos

6 de janeiro de 2026

Resumo

O sistema eleitoral brasileiro disponibiliza uma quantidade expressiva de dados públicos sobre candidaturas, possibilitando investigar fatores associados ao sucesso eleitoral. Neste trabalho, formulamos o problema como uma classificação binária supervisionada, com o objetivo de prever se um candidato a Deputado Federal será eleito (classe positiva) ou não eleito (classe negativa) utilizando exclusivamente atributos declarados ao Tribunal Superior Eleitoral (TSE) antes do pleito, com ênfase em dados financeiros de campanha. Como a classe positiva é minoritária (aproximadamente 5%), empregamos ponderação de classes para lidar com o desbalanceamento. A avaliação utiliza validação temporal (treino em 2018, teste em 2022) para simular um cenário realista de predição, além de validação cruzada estratificada para avaliar estabilidade. Comparamos quatro modelos – Regressão Logística, Random Forest, Gradient Boosting e XGBoost – com otimização de hiperparâmetros via Grid Search, utilizando métricas adequadas a cenários desbalanceados.

1 Introdução

O sistema eleitoral brasileiro disponibiliza um amplo conjunto de dados públicos que viabiliza análises quantitativas sobre candidaturas e seus atributos declarados. Em particular, o Tribunal Superior Eleitoral (TSE) mantém um portal de dados abertos que reúne informações sobre candidatos e eleições, permitindo estudos reprodutíveis a partir de fontes oficiais [21]. Paralelamente, métodos de Aprendizagem de Máquina têm sido cada vez mais empregados em Ciência Política, tanto para fins preditivos quanto descritivos, motivados pelo aumento de disponibilidade de dados e pelo interesse em modelos explicáveis [8]. Embora muitas aplicações de aprendizado de máquina em predição eleitoral usem sinais digitais como mídias sociais ou enquetes, estudos recentes empregam técnicas supervisionadas para modelagem de resultados eleitorais quantitativos, incluindo métodos ensemble que capturam relações não lineares entre atributos de candidatos e votação obtida [8, 22]. No contexto eleitoral, a literatura também discute limites e controvérsias no uso de sinais digitais (por exemplo, mídias sociais) para previsão, apontando desafios metodológicos recorrentes [3, 10].

Este trabalho aborda a predição de sucesso eleitoral para o cargo de Deputado Federal como uma tarefa de classificação binária supervisionada (Eleito vs. Não Eleito). A proposta é estritamente ex-ante: utilizam-se somente atributos disponíveis antes do pleito, mitigando vazamentos de informação (data leakage) e aproximando o cenário de uma aplicação prospectiva. Um obstáculo central é o forte desbalanceamento de classes, pois o número de eleitos é uma pequena fração do total de candidaturas. Por exemplo, em 2022 houve da ordem de 10,4 mil candidaturas para 513 vagas, isto é, aproximadamente 5% [7, 1]. Como discutido na literatura de aprendizado com classes raras, tal desequilíbrio pode enviesar modelos em favor da classe majoritária e tornar inadequadas métricas não robustas à distribuição de classes, exigindo procedimentos de avaliação específicos [18, 6].

O diferencial metodológico deste trabalho reside na utilização de validação temporal: o modelo é treinado com dados das Eleições de 2018 e testado nas Eleições de 2022, simulando um cenário realista de predição intertemporal. Esta abordagem contrasta com validações cruzadas convencionais que embaralham dados temporais e pode superestimar o desempenho em aplicações reais. Adicionalmente, foram incluídos dados financeiros como patrimônio declarado e teto de gastos de campanha, que são informações disponíveis antes da eleição e se mostraram relevantes para a predição.

O objetivo deste estudo é desenvolver modelos preditivos baseados exclusivamente em informações públicas pré-eleitorais, comparando diferentes vieses indutivos em dados tabulares, como modelos lineares e métodos de ensemble. Além do desempenho preditivo, busca-se discutir a interpretabilidade, as limitações de generalização temporal dos modelos e as implicações éticas do uso de atributos financeiros e demográficos no contexto institucional do sistema proporcional de lista aberta.

1.1 Contribuições

As principais contribuições deste trabalho são:

- Construção de um pipeline reprodutível a partir de dados abertos do TSE, com recorte no cargo de Deputado Federal, incluindo dados de patrimônio declarado e teto de gastos de campanha.
- Implementação de validação temporal (treino 2018, teste 2022) para avaliar generalização realista entre ciclos eleitorais.
- Comparação sistemática entre Regressão Logística (baseline), Random Forest, Gradient Boosting e XGBoost em dados tabulares desbalanceados [2, 5, 11].
- Otimização de hiperparâmetros via Grid Search com foco em métricas adequadas ao desbalanceamento (F1-score) [19].
- Avaliação de ponderação de classes como estratégia para mitigar desbalanceamento, com análise comparativa entre diferentes implementações (class_weight, scale_pos_weight).
- Análise de interpretabilidade via importância de features e identificação de limiares financeiros associados ao sucesso eleitoral.

2 Fundamentação teórica

2.1 Classificação binária em dados desbalanceados

Em problemas de classificação binária, o desbalanceamento de classes ocorre quando uma das classes (tipicamente a classe positiva de interesse) representa uma proporção significativamente menor do dataset. Este cenário é comum em aplicações reais como detecção de fraudes, diagnóstico de doenças raras e, como neste trabalho, predição de sucesso eleitoral [18, 6].

Quando o desbalanceamento é severo ($< 10\%$ da classe minoritária), classificadores convencionais tendem a enviesar as predições em favor da classe majoritária, maximizando acurácia global às custas de baixo recall na classe minoritária [18, 6, 13]. Por exemplo, um classificador trivial que sempre prediz "não eleito" alcançaria 95% de acurácia em um dataset com 5% de eleitos. Entretanto, essa previsão trivial não tem utilidade prática para identificar candidatos com real potencial de vitória.

2.1.1 Métricas adequadas

A adoção de métricas robustas à distribuição de classes é essencial para a avaliação adequada de modelos em cenários desbalanceados [18, 6]. Embora a acurácia seja intuitiva, ela pode ser enganosa quando um classificador trivial que prediz sempre a classe majoritária alcança alta acurácia sem identificar corretamente a classe minoritária — fenômeno conhecido como *accuracy paradox* [23]. Essa limitação justifica o uso de métricas que ponderam adequadamente o desempenho nas duas classes:

F1-score é a média harmônica entre precisão e recall:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{1}$$

onde Precision $= \frac{TP}{TP+FP}$ e Recall $= \frac{TP}{TP+FN}$. Esta métrica penaliza igualmente falsos positivos e falsos negativos, sendo útil quando ambos os erros têm custo similar.

Balanced Accuracy é a média das taxas de acerto por classe:

$$\text{Balanced Acc.} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right) \tag{2}$$

Esta métrica evita que a classe majoritária domine a avaliação ao ponderar igualmente o desempenho em cada classe, sendo particularmente útil em datasets desbalanceados [4].

AUC-ROC (Area Under the ROC Curve) mede a capacidade do modelo de distinguir entre classes em diferentes limiares de decisão, sendo robusta a desbalanceamentos moderados [18].

Curva Precision-Recall e AUC-PR são especialmente recomendadas para classes altamente desbalanceadas, pois focam exclusivamente no desempenho da classe positiva, ignorando verdadeiros negativos. Saito e Rehmsmeier (2015) demonstram que AUC-PR é mais informativa que AUC-ROC quando a classe positiva é rara [18].

Estudos sobre aprendizado com classes desbalanceadas recomendam métricas que capturam tanto a capacidade de ranqueamento quanto o equilíbrio entre precisão e recall. F1-score equilibra precisão e recall, enquanto AUC-PR foca na identificação da classe minoritária, sendo mais informativa que AUC-ROC em cenários de alta assimetria [18]. Balanced Accuracy, por sua vez, evita que a classe majoritária domine a avaliação ao ponderar igualmente as taxas de acerto de cada classe [4].

2.2 Estratégias para alívio de desbalanceamento

A literatura propõe três abordagens principais para lidar com desbalanceamento: técnicas de reamostragem no nível de dados, modificações algorítmicas no nível de aprendizado, e métodos de ensemble [6, 13]. Neste trabalho, adotamos principalmente a segunda abordagem (modificações algorítmicas) por sua simplicidade e eficácia prática.

2.2.1 Ponderação de classes

A ponderação de classes (class weighting) ajusta a função de perda para penalizar mais erros na classe minoritária, sem modificar o dataset [13]. Sejam n_{maj} e n_{min} os tamanhos das classes majoritária e minoritária. Os pesos balanceados são definidos como:

$$w_{min} = \frac{n_{total}}{2 \cdot n_{min}}, \quad w_{maj} = \frac{n_{total}}{2 \cdot n_{maj}} \tag{3}$$

Esta estratégia é implementada de forma nativa em diversos algoritmos do scikit-learn através do parâmetro `class_weight='balanced'` [16]. Para modelos que não suportam diretamente ponderação de classes, pode-se usar pesos amostrais (sample_weight) durante o treinamento.

Implementações específicas por algoritmo:

- **Regressão Logística:** Modifica a função de perda de log-verossimilhança para $\mathcal{L} = -\sum_i w_i[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$ [16]
- **Random Forest:** Aplica pesos na construção de cada árvore, alterando os critérios de split (Gini ou entropia ponderados) [2]
- **Gradient Boosting:** Usa sample_weight na otimização do gradiente em cada iteração [11]
- **XGBoost:** Implementa o parâmetro scale_pos_weight $= \frac{n_{neg}}{n_{pos}}$ que escala o gradiente da classe positiva [5]

Comparada a técnicas de reamostragem (undersampling e SMOTE), a ponderação de classes apresenta vantagens práticas: (i) não requer modificação do dataset; (ii) evita perda de informação (undersampling) ou geração de exemplos sintéticos potencialmente ruidosos (SMOTE); (iii) mantém a distribuição real dos dados, facilitando interpretação [13]. Por essas razões, priorizamos a ponderação de classes como estratégia principal neste trabalho.

2.3 Validação temporal vs. validação cruzada

Em problemas com dimensão temporal, a validação cruzada convencional pode superestimar o desempenho real ao permitir que informações futuras "vazem" para o conjunto de treino através de embaralhamento aleatório [20].

Validação temporal mantém a ordem cronológica: o modelo é treinado em dados passados e testado em dados futuros. No contexto eleitoral, isso simula o cenário real onde um analista treinaria o modelo após as eleições de 2018 e o aplicaria para predizer o sucesso de candidatos em 2022 antes do pleito.

Validação cruzada estratificada (StratifiedKFold) divide o dataset preservando a proporção de classes em cada fold, sendo essencial para dados desbalanceados [20]. No entanto, quando aplicada a dados temporais, pode embaralhar observações de diferentes períodos, levando a estimativas otimistas.

Neste trabalho, empregamos ambas as abordagens de forma complementar: validação temporal como avaliação primária (generalização entre eleições) e validação cruzada estratificada apenas no conjunto de treino (2018) para avaliar estabilidade e detectar overfitting dentro de um mesmo ciclo eleitoral.

2.4 Codificação de variáveis categóricas de alta cardinalidade

Variáveis categóricas como partido político (35 categorias) e ocupação (mais de 200 categorias) apresentam desafios para modelagem. One-hot encoding gera dimensionalidade excessiva e esparsa, podendo levar a overfitting, especialmente em árvores de decisão. Neste trabalho, utilizamos target encoding como estratégia principal de codificação.

Target encoding (ou mean encoding) substitui cada categoria pela média do target para aquela categoria [15]:

$$\text{partido\_enc}[p] = \frac{\sum_{i:\text{partido}_i=p} y_i}{\sum_{i:\text{partido}_i=p} 1} \tag{4}$$

Esta técnica reduz dimensionalidade drasticamente (de k categorias para uma única feature numérica) e captura o sinal preditivo médio de cada categoria. É crucial calcular o encoding apenas no conjunto de treino e aplicá-lo no conjunto de teste para evitar data leakage [15].

2.5 Modelos de classificação considerados

Métodos de ensemble combinam múltiplos modelos fracos para construir um preditor robusto, melhorando a generalização ao capturar diferentes vieses indutivos e reduzir variância [9].

2.5.1 Regressão Logística

A Regressão Logística é um modelo linear que estima a probabilidade da classe positiva através da função logística:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta^T X)}} \tag{5}$$

Suas vantagens incluem: (i) interpretabilidade direta através dos coeficientes β; (ii) eficiência computacional; (iii) baixa propensão a overfitting; (iv) produção de probabilidades calibradas. É utilizada como baseline por sua simplicidade e por permitir análise dos sinais e magnitudes dos preditores [16].

2.5.2 Random Forest

Random Forest é um método de ensemble que combina múltiplas árvores de decisão treinadas em subamostras do dataset com aleatorização de features [2]. Cada árvore é construída através de bootstrap aggregating (bagging) e, em cada split, considera apenas um subconjunto aleatório de $\sqrt{p}$ features (onde p é o total de features).

Vantagens: (i) captura relações não-lineares e interações complexas; (ii) robusto a outliers e features irrelevantes; (iii) reduz variância através de averaging; (iv) fornece importância de features via redução de impureza. Random Forest tende a performar bem em dados tabulares heterogêneos sem necessidade de preprocessamento intensivo.

2.5.3 Gradient Boosting

Gradient Boosting constrói um ensemble aditivo de árvores rasas (shallow trees) de forma sequencial, onde cada nova árvore é treinada para corrigir os resíduos do ensemble anterior [11]:

$$F_m(x) = F_{m-1}(x) + \eta \cdot h_m(x) \tag{6}$$

onde $h_m(x)$ é a árvore treinada nos pseudo-resíduos e η é a taxa de aprendizado. Esta abordagem reduz tanto viés quanto variância, geralmente superando Random Forest em dados tabulares quando bem calibrada. No entanto, é mais sensível a overfitting e requer ajuste cuidadoso de hiperparâmetros.

2.5.4 XGBoost (Extreme Gradient Boosting)

XGBoost é uma implementação otimizada de Gradient Boosting que adiciona regularização (L1 e L2) na função objetivo e utiliza second-order Taylor expansion para acelerar convergência [5]:

$$\mathcal{L}(\phi) = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k) \tag{7}$$

onde $\Omega(f_k) = \gamma T + \frac{1}{2}\lambda||\omega||^2$ penaliza complexidade (número de folhas T e magnitude dos pesos ω).

XGBoost domina competições de machine learning em dados tabulares por combinar alta performance preditiva com eficiência computacional. Sua implementação nativa de scale_pos_weight o torna particularmente adequado para dados desbalanceados.

2.6 Otimização de hiperparâmetros

Hiperparâmetros são parâmetros definidos antes do treinamento que controlam a complexidade e comportamento do modelo. A otimização inadequada pode levar a underfitting (modelo muito simples) ou overfitting (modelo muito complexo).

2.6.1 Grid Search

Grid Search é uma busca exaustiva sobre um grid pré-definido de hiperparâmetros, avaliando todas as combinações possíveis [19]. Para cada combinação, o modelo é treinado e avaliado usando validação cruzada, e a combinação que maximiza a métrica escolhida é selecionada.

Formalmente, dado um conjunto de hiperparâmetros Λ e uma métrica de validação M:

$$\lambda^* = \arg\max_{\lambda \in \Lambda} \mathbb{E}_{CV}[M(f_\lambda)] \tag{8}$$

Implementação no scikit-learn: GridSearchCV automatiza este processo usando StratifiedKFold [20] internamente. O parâmetro `scoring` define a métrica de otimização (ex: `'f1'` para F1-score), e `cv` define o número de folds.

3 Metodologia

3.1 Fonte de dados e recorte temporal

Os dados foram obtidos do Portal de Dados Abertos do Tribunal Superior Eleitoral (TSE), especificamente dos repositórios de candidatos e prestação de contas [21]. O recorte temporal abrange as Eleições Gerais de 2018 (conjunto de treino) e 2022 (conjunto de teste), focando exclusivamente no cargo de Deputado Federal (CD_CARGO = 6).

3.1.1 Datasets utilizados

Para cada ano, foram integrados três datasets distintos:

Tabela 1: Datasets utilizados do Portal TSE.

Dataset	**Informação contida**
consulta_cand	Dados cadastrais básicos (nome, partido, UF, gênero, estado civil, grau de instrução, cor/raça)
consulta_cand_complementar	Reeleição, idade, situação final, teto de gastos (VR_DESPESA_MAX_CAMPANHA)
bem_candidato	Patrimônio declarado (soma de bens)

A integração foi realizada através da chave primária `SQ_CANDIDATO`, identificador único do candidato no TSE [21]. O patrimônio foi agregado por candidato calculando-se a soma total de bens declarados: $\text{TOTAL\_BENS} = \sum_j \text{VR\_BEM\_CANDIDATO}_j$.

3.2 Definição da variável resposta (target)

A variável binária $Y \in \{0, 1\}$ foi construída a partir da coluna `DS_SIT_TOT_TURNO`, que indica a situação final do candidato. Seguindo as regras do sistema eleitoral proporcional [7]:

$$Y = \begin{cases} 1 & \text{se DS\_SIT\_TOT\_TURNO} \in \{\text{ELEITO}, \text{ELEITO POR MÉDIA}, \text{ELEITO POR QP}\} \\ 0 & \text{caso contrário (NÃO ELEITO, SUPLENTE)} \end{cases} \tag{9}$$

Esta definição reflete o conceito legal de "eleito" no sistema proporcional brasileiro, onde candidatos podem ser eleitos por média (coeficiente eleitoral) ou por quociente partidário (QP), além da eleição direta por votação nominal.

3.2.1 Distribuição de classes

Após filtragens (detalhadas na Seção 3.4), o dataset apresentou severo desbalanceamento:

Tabela 2: Distribuição de classes nos conjuntos de treino e teste.

Conjunto	**Não Eleitos**	**Eleitos**	**Taxa de Eleição**
Treino (2018)	∼7.200	512	6.7%
Teste (2022)	∼9.000	512	5.4%

Esta proporção de aproximadamente 1:15.7 (eleitos:não eleitos) caracteriza um problema de classes altamente desbalanceadas, justificando o uso de métricas específicas e ponderação de classes [6].

3.3 Variáveis explicativas (features)

As features foram organizadas em quatro grupos temáticos, totalizando aproximadamente 50 variáveis após codificação:

3.3.1 Dados demográficos

- **NR_IDADE_DATA_POSSE**: Idade do candidato no ano da eleição (numérica, em anos)
- **IS_FEMININO**: Indicador binário (1 se CD_GENERO = 4, 0 caso contrário)
- **CD_ESTADO_CIVIL**: Estado civil (categórica: solteiro, casado, divorciado, viúvo)
- **CD_GRAU_INSTRUCAO**: Escolaridade (categórica ordinal: ensino fundamental a superior completo)
- **CD_COR_RACA**: Cor/raça declarada (categórica)
- **SG_UF_NASCIMENTO**: UF de nascimento (categórica)

3.3.2 Dados financeiros

- **LOG_BENS**: $\log_{10}(1 + \text{patrimônio declarado})$
- **LOG_DESPESA_MAX**: $\log_{10}(1 + \text{teto de gastos})$
- **TEM_BENS**: Indicador binário de patrimônio positivo

A transformação logarítmica foi aplicada para normalizar distribuições altamente assimétricas (valores financeiros frequentemente seguem distribuições log-normais) e reduzir o impacto de outliers [2, 5]. Utilizou-se $\log_{10}(1 + x)$ (log1p) para evitar problemas com valores zero.

3.3.3 Features de coligação

- **IS_COLIGADO**: Indicador binário (1 se TP_AGREMIACAO = "COLIGAÇÃO", 0 caso contrário)
- **QTD_PARTIDOS_COLIG**: Quantidade de partidos na coligação (numérica)
- **COLIGACAO_ENC**: Target encoding da composição da coligação [15]

3.3.4 Dados políticos e categóricos

- **IS_REELEICAO**: Indicador binário (1 se ST_REELEICAO = 'S', 0 se 'N')
- **PARTIDO_ENC**: Codificação numérica do partido via target encoding [15]
- **OCUPACAO_ENC**: Codificação numérica da ocupação via target encoding [15]
- **GRAU_INSTR_ENC**: Codificação numérica do grau de instrução via target encoding [15]
- **ESTADO_CIVIL_ENC**: Codificação numérica do estado civil via target encoding [15]
- **COR_RACA_ENC**: Codificação numérica de cor/raça via target encoding [15]
- **UF_NASC_ENC**: Codificação numérica da UF de nascimento via target encoding [15]
- **UF_***: 27 variáveis dummy (one-hot encoding) para Unidade Federativa da candidatura
- **IS_PARTIDO_GRANDE**: Indicador binário para partidos com mais de 30 candidatos

3.4 Pré-processamento e tratamento de dados ausentes

O pipeline de pré-processamento seguiu uma sequência rigorosa para evitar data leakage:

3.4.1 Limpeza inicial

1. Filtragem de registros com situação indefinida (removidos: candidaturas indeferidas, sub judice)
2. Substituição de marcadores de ausência (#NULO, #NE) por `NaN`
3. Conversão de colunas financeiras (formato string "R$ X.XXX,XX" $\rightarrow$ float)

3.4.2 Tratamento de valores ausentes

A estratégia de imputação dependeu do tipo e significado da variável:

Tabela 3: Estratégias de imputação por tipo de variável.

Tipo	**Estratégia**	**Justificativa**
Financeiras	0	Ausência indica valor zero
ST_REELEICAO	'N'	Ausência indica não-reeleição
Idade	Mediana	Robusto a outliers

Registros com campos críticos ausentes (partido, UF, ocupação) foram removidos, resultando em perda de $< 2\%$ das observações.

3.4.3 Mapeamento de partidos

Para garantir consistência temporal entre as eleições de 2018 e 2022, foi realizado mapeamento de fusões partidárias ocorridas no período:

Tabela 4: Fusões partidárias mapeadas para consistência temporal.

Partido Original (2018)	**Partido Mapeado (2018→2022)**
PR	PL
PPS	CIDADANIA
PTC	AGIR
PSL	UNIÃO
DEM	UNIÃO

Este mapeamento foi aplicado no conjunto de treino (2018) antes da codificação, garantindo que as categorias correspondam às existentes em 2022. Após o mapeamento, restaram 34 partidos únicos em 2018 e 32 em 2022.

3.4.4 Codificação de variáveis categóricas

Target Encoding com Smoothing Bayesiano: Para variáveis de alta cardinalidade ou sem ordem lógica, aplicou-se target encoding com smoothing bayesiano (também conhecido como empirical Bayes) para prevenir overfitting em categorias raras [15, 12]. As seguintes variáveis foram codificadas:

- **PARTIDO_ENC**: Partido político (34 categorias após fusões)
- **OCUPACAO_ENC**: Ocupação declarada (>200 categorias)
- **GRAU_INSTR_ENC**: Grau de instrução (8 categorias)
- **ESTADO_CIVIL_ENC**: Estado civil (5 categorias)
- **COR_RACA_ENC**: Cor/raça declarada (6 categorias)
- **UF_NASC_ENC**: UF de nascimento (27 categorias)
- **COLIGACAO_ENC**: Composição da coligação (alta cardinalidade)

A fórmula de smoothing utilizada foi:

$$\text{enc}[\text{cat}] = \frac{n_{\text{cat}} \cdot \bar{y}_{\text{cat}} + \alpha \cdot \bar{y}_{\text{global}}}{n_{\text{cat}} + \alpha} \tag{10}$$

onde:

- n_{cat}: número de observações na categoria
- $\bar{y}_{\text{cat}}$: taxa de eleitos na categoria (média empírica)
- $\bar{y}_{\text{global}}$: taxa global de eleitos no treino (≈ 0.067)
- α: parâmetro de smoothing (valor utilizado: $\alpha = 10$)

Este método ajusta a codificação de cada categoria combinando a média observada daquela categoria com a média global (prior), dando mais peso à média global quando a categoria tem poucas observações. Para categorias raras ($n_{\text{cat}} \ll \alpha$), o encoding converge para $\bar{y}_{\text{global}}$, evitando estimativas instáveis [12].

Protocolo anti-leakage:

1. Calcular encoding apenas no conjunto de treino (2018)
2. Aplicar o mesmo mapeamento no teste (2022)
3. Categorias ausentes no treino recebem valor padrão ($\bar{y}_{\text{global}}$)

One-Hot Encoding (baixa cardinalidade): Para UF (27 categorias) foi aplicado o one-hot encoding. Para evitar multicolinearidade perfeita, uma categoria foi dropada (dummy trap). Gênero foi codificado como variável binária IS_FEMININO (1 se feminino, 0 caso contrário).

3.4.5 Padronização (StandardScaler)

Features numéricas foram padronizadas para média 0 e desvio padrão 1:

$$X_{\text{scaled}} = \frac{X - \mu_{\text{treino}}}{\sigma_{\text{treino}}} \tag{11}$$

Crucialmente, μ e σ foram calculados exclusivamente no treino e aplicados no teste, evitando data leakage. A padronização é essencial para Regressão Logística (sensível a escala) mas opcional para árvores de decisão (invariantes a transformações monotônicas) [16].

Features padronizadas: Foram padronizadas todas as features numéricas contínuas: idade, features logarítmicas (LOG_BENS, LOG_DESPESA_MAX), target encodings e features de coligação. Features binárias (IS_REELEICAO, IS_FEMININO, etc.) e dummies de UF não foram padronizadas por já estarem em escala 0-1.

3.5 Estratégia de validação

3.5.1 Validação temporal (avaliação primária)

A divisão temporal simula um cenário realista: treino nas Eleições 2018 (7.641 candidatos) → teste nas Eleições 2022 (9.474 candidatos). Esta abordagem avalia a capacidade de generalização do modelo entre ciclos eleitorais, capturando mudanças no contexto político (surgimento/extinção de partidos, alterações no financiamento eleitoral, shift nas preferências do eleitorado).

Limitação temporal conhecida: O modelo utiliza o teto de gastos de campanha (VR_DESPESA_MAX_ que é uma informação pré-declarada e disponível antes da eleição. No entanto, variáveis financeiras agregadas de prestação de contas (receitas e despesas efetivas) são disponibilizadas apenas pós-eleição pelo TSE e, portanto, não foram utilizadas neste trabalho para manter a viabilidade de predição ex-ante. Esta limitação e suas implicações são discutidas nas conclusões.

3.5.2 Validação cruzada estratificada (avaliação secundária)

Para avaliar estabilidade e detectar overfitting, aplicou-se StratifiedKFold com 5 folds exclusivamente no conjunto de treino (2018) [20]. Cada fold preserva a proporção 95%-5% (não eleitos-eleitos). Métricas (F1, AUC-ROC) são calculadas em cada fold, reportando-se média ± desvio padrão.

Um modelo com alta variância entre folds indica instabilidade. Um modelo com CV alta mas teste temporal baixo indica falta de generalização entre eleições.

3.6 Tratamento de desbalanceamento: ponderação de classes

Optou-se exclusivamente por ponderação de classes, evitando reamostragem [13]. Os pesos foram calculados como:

$$w_0 = \frac{n_{\text{total}}}{2 \cdot n_0}, \quad w_1 = \frac{n_{\text{total}}}{2 \cdot n_1} \tag{12}$$

Para o treino (2018), com $n_0 = 7.129$ (não eleitos) e $n_1 = 512$ (eleitos):

$$w_0 = \frac{7.641}{2 \cdot 7.129} \approx 0.536, \quad w_1 = \frac{7.641}{2 \cdot 512} \approx 7.45 \tag{13}$$

Isso significa que erros em candidatos eleitos são penalizados aproximadamente $\frac{n_0}{n_1} \approx 14$ vezes mais que erros em não eleitos, compensando o desbalanceamento natural (razão 14:1).

Implementação por modelo:

- **Logistic Regression:** `class_weight='balanced'` [16]
- **Random Forest:** `class_weight='balanced'` [2]
- **Gradient Boosting:** Treinado sem ponderação explícita (ver nota na Seção 3.7.3)
- **XGBoost:** `scale_pos_weight =` $\frac{n_0}{n_1} = \frac{7.129}{512} \approx 14$ [5]

3.7 Modelos e otimização de hiperparâmetros

Quatro modelos foram treinados e comparados sistematicamente. Para cada modelo, realizou-se otimização de hiperparâmetros via GridSearchCV com 5-fold stratified cross-validation, usando F1-score como métrica de seleção [19]. A otimização foi conduzida exclusivamente no conjunto de treino (2018), evitando vazamento de informação do conjunto de teste (2022).

Estratégia de otimização em dois estágios: Para reduzir o custo computacional, adotou-se uma estratégia de otimização em dois estágios: (i) **Screening inicial** com grid reduzido para comparação rápida entre todos os modelos; (ii) **Otimização completa** nos dois modelos finalistas usando grids expandidos. Para grids com mais de 100 combinações, utilizou-se RandomizedSearchCV ($n_{iter} = 50$) em vez de busca exaustiva para viabilizar o treinamento [19].

3.7.1 Regressão Logística (baseline)

Modelo linear utilizado como baseline interpretável para comparação [16].

Hiperparâmetros otimizados:

- `C`: Inverso da força de regularização (testado: 0.001, 0.01, 0.1, 1.0, 10.0, 100.0)
- `penalty`: Tipo de regularização (testado: 'l1', 'l2')
- `solver`: Algoritmo de otimização (testado: 'liblinear', 'saga')

Configuração fixa: `class_weight='balanced', max_iter=1000, random_state=42`

Espaço de busca (screening): $3 \times 1 \times 1 = 3$ combinações

Espaço de busca (otimização completa): $6 \times 2 \times 2 = 24$ combinações

3.7.2 Random Forest

Método de ensemble baseado em bagging com aleatorização de features [2].

Hiperparâmetros otimizados:

- `n_estimators`: Número de árvores (testado: 50, 100, 200, 300)
- `max_depth`: Profundidade máxima das árvores (testado: 10, 15, 20, None)
- `min_samples_split`: Amostras mínimas para split (testado: 2, 5, 10)
- `min_samples_leaf`: Amostras mínimas por folha (testado: 1, 2, 4)

Configuração fixa: `class_weight='balanced', random_state=42`

Espaço de busca (otimização completa): $4 \times 4 \times 3 \times 3 = 144$ combinações (testadas via RandomizedSearchCV com $n_{iter} = 50$ por restrições computacionais)

3.7.3 Gradient Boosting

Método de ensemble sequencial baseado em boosting de árvores fracas [11].

Hiperparâmetros otimizados:

- `n_estimators`: Número de árvores sequenciais (testado: 50, 100, 200, 300)
- `max_depth`: Profundidade das árvores fracas (testado: 3, 5, 7, 10)
- `learning_rate`: Taxa de aprendizado (testado: 0.01, 0.05, 0.1, 0.2)
- `subsample`: Fração de amostras por árvore (testado: 0.8, 1.0)

Configuração fixa: `random_state=42`

Nota sobre ponderação: Diferentemente dos outros modelos, o GradientBoostingClassifier do scikit-learn não suporta nativamente o parâmetro `class_weight`. A ponderação poderia ser implementada via `sample_weight` passado ao método `fit()`, mas por simplicidade de implementação no GridSearchCV, este modelo foi treinado sem ponderação explícita de classes, dependendo da capacidade do boosting de aprender padrões da classe minoritária através do ajuste sequencial de resíduos.

Espaço de busca: $4 \times 4 \times 4 \times 2 = 128$ combinações

3.7.4 XGBoost

Implementação otimizada de gradient boosting com regularização [5].

Hiperparâmetros otimizados:

- `n_estimators`: Número de boosting rounds (testado: 100, 200, 300, 500)
- `max_depth`: Profundidade máxima (testado: 3, 5, 7, 10)
- `learning_rate`: Taxa de aprendizado (testado: 0.01, 0.05, 0.1, 0.2)
- `subsample`: Fração de amostras por árvore (testado: 0.8, 1.0)
- `colsample_bytree`: Fração de features por árvore (testado: 0.8, 1.0)

Configuração fixa: `scale_pos_weight=`$\frac{n_0}{n_1} \approx 14$`, random_state=42, eval_metric='logloss'`

Espaço de busca (otimização completa): $4 \times 4 \times 4 \times 2 \times 2 = 256$ combinações (testadas via RandomizedSearchCV com $n_{iter} = 50$ por restrições computacionais)

Protocolo de Grid Search: Para cada modelo, o procedimento de otimização seguiu os seguintes passos:

Etapa 1 - Screening (Grid Reduzido):

1. Define-se um grid reduzido de hiperparâmetros para cada modelo
2. `GridSearchCV` com `cv=5, scoring=f1` treina todas as combinações
3. Selecionam-se os 2 modelos com melhor F1-score no teste temporal

Etapa 2 - Otimização Completa (Grid Expandido):

1. Para os 2 finalistas, define-se um grid expandido de hiperparâmetros
2. Se grid $>$ 100 combinações: `RandomizedSearchCV` com `n_iter=50, cv=5, scoring='f1'`
3. Se grid $\leq$ 100 combinações: `GridSearchCV` com `cv=5, scoring='f1'`
4. Para cada combinação, calcula-se F1 médio em 5-fold CV (apenas no treino 2018)
5. Seleciona-se a combinação com maior F1 médio
6. O modelo com melhor configuração é retreinado em todo o treino e avaliado no teste
7. Métricas finais são reportadas tanto para validação cruzada (F1 médio nos 5 folds do treino) quanto para teste temporal (valores únicos no teste 2022)

Este protocolo de dois estágios garante: (i) comparação inicial eficiente entre todos os modelos; (ii) otimização mais refinada nos finalistas; (iii) viabilidade computacional para grids grandes; (iv) seleção de hiperparâmetros sem acesso aos dados de teste; (v) avaliação final que reflete capacidade de generalização temporal.

Tabela 5: Métricas de avaliação utilizadas.

Métrica	Interpretação
F1-score	Média harmônica de precisão e recall; métrica primária de otimização
Precision	Proporção de predições positivas corretas; penaliza falsos positivos
Recall	Proporção de eleitos corretamente identificados; penaliza falsos negativos
Balanced Accuracy	Média das taxas de acerto por classe; robusta a desbalanceamento
AUC-ROC	Área sob curva ROC; capacidade de ranqueamento em diferentes limiares
AUC-PR	Área sob curva Precision-Recall; mais informativa que AUC-ROC para classes raras

3.8 Métricas de avaliação

As seguintes métricas foram calculadas tanto em validação cruzada (treino 2018) quanto em teste temporal (2022) [18, 6]:

Adicionalmente, foram realizadas análises descritivas estratificadas por quantis de variáveis financeiras (patrimônio e teto de gastos) para identificar limiares práticos que discriminam candidatos eleitos de não eleitos. Por exemplo, analisou-se a taxa de eleição em função de faixas de patrimônio declarado e teto de gastos de campanha, facilitando a interpretação e aplicação prática dos resultados.

3.9 Interpretabilidade

Para cada modelo treinado, extraímos medidas de importância de features: (i) **Regressão Logística:** magnitude absoluta dos coeficientes $|\beta_j|$ após padronização [16]; (ii) **Árvores (RF, GB, XGB):** importância por redução de impureza (Gini ou entropia) [2, 11].

As top-15 features mais importantes foram identificadas para cada modelo, permitindo análise de consenso: features que aparecem consistentemente no top-5 de múltiplos modelos são consideradas genuinamente preditivas.

Adicionalmente, análises descritivas foram conduzidas para identificar limiares práticos (por exemplo, "candidatos com patrimônio acima de R$ X ou teto de gastos acima de R$ Y têm Z de chance de eleição"), facilitando interpretação e aplicação prática dos resultados.

4 Experimentos e Resultados

4.1 Configuração experimental

4.1.1 Dados utilizados

- **Treino:** Eleições 2018 – Deputado Federal – Todas as UFs
- **Teste:** Eleições 2022 – Deputado Federal – Todas as UFs

- **Tamanho pós-filtragem:** 7.641 candidatos (treino), 9.474 candidatos (teste)
- **Taxa de eleição:** 6.7% (treino), 5.4% (teste)

4.1.2 Filtros aplicados

1. Cargo = Deputado Federal (CD_CARGO = 6)
2. Situação definida (removidos: indeferidos, sub judice, renúncias)
3. Campos críticos presentes (partido, UF, ocupação)

4.1.3 Hardware e software

- Python 3.13
- scikit-learn 1.7.2
- xgboost 3.1.2
- pandas 2.3.3
- numpy 2.3.3
- matplotlib 3.10.7
- seaborn 0.13.2
- imbalanced-learn 0.0
- shap 0.50.0
- jupyter 1.1.1
- Processamento: Google Colab (CPU padrão)

4.2 Resultados principais

A Tabela 6 apresenta o desempenho comparativo dos quatro modelos avaliados no conjunto de teste (Eleições 2022). O Random Forest foi o modelo vencedor, alcançando F1-score de 0.599, seguido por XGBoost (0.583) e Gradient Boosting (0.572). A Regressão Logística, utilizada como baseline, obteve F1 de 0.449, confirmando a superioridade dos métodos ensemble em dados tabulares complexos (ΔF1 $\approx$ 0.15).

Tabela 6: Desempenho comparativo dos modelos em validação temporal (teste 2022).

Modelo	F1	AUC-ROC	AUC-PR	Precision	Recall
Random Forest	0.599	0.931	0.557	0.636	0.566
XGBoost	0.583	0.909	0.496	0.551	0.619
Gradient Boosting	0.572	0.927	0.540	0.642	0.516
Reg. Logística	0.449	0.934	0.552	0.311	0.811

Observa-se um trade-off entre precisão e recall: enquanto a Regressão Logística maximiza recall (0.811) às custas de baixa precisão (0.311), o Random Forest apresenta equilíbrio (precision = 0.636, recall = 0.566). Este comportamento é esperado dado o forte desbalanceamento de

classes – modelos com alta ponderação tendem a "jogar seguro" e predizer mais candidatos como eleitos, aumentando recall mas gerando falsos alarmes. Todos os modelos alcançaram AUC-ROC superior a 0.90, indicando excelente capacidade de ranqueamento entre candidatos, mesmo em um cenário de validação temporal rigorosa.

A Figura 1 mostra as matrizes de confusão dos dois finalistas. O Random Forest classifica corretamente 297 dos 512 candidatos eleitos (TP = 58%) e 8.800 dos 8.962 não eleitos (TN = 98%), resultando em balanced accuracy de 72.5%. Os 215 falsos negativos (FN) e 162 falsos positivos (FP) revelam que o modelo é mais conservador que o XGBoost, que sacrifica precisão (mais FPs) para maximizar recall (mais TPs).

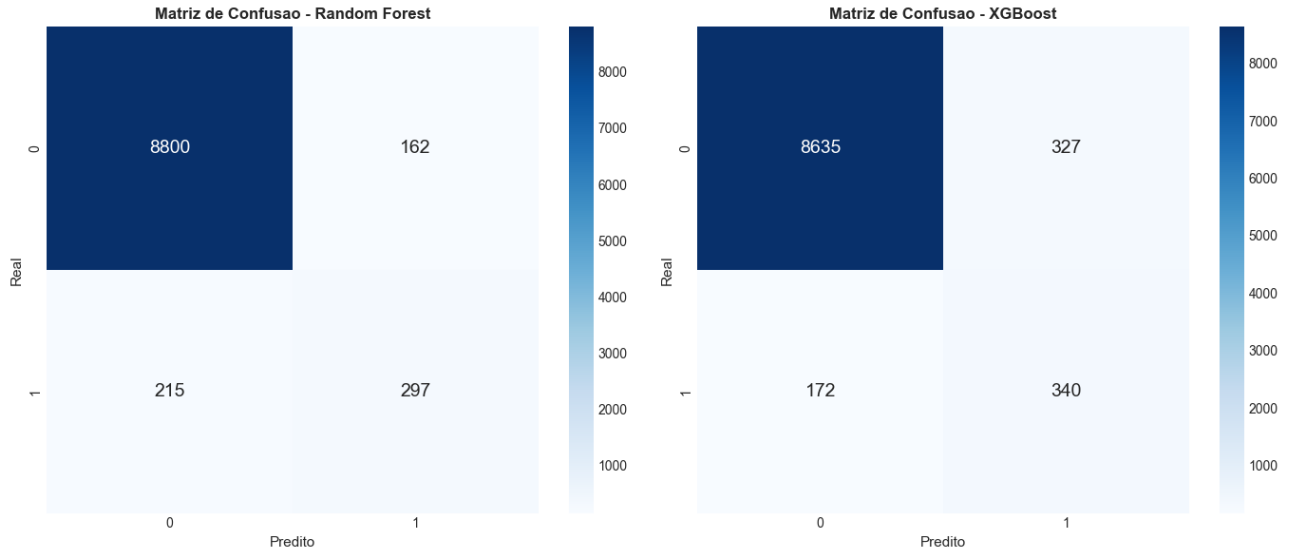

Figura 1: Matrizes de confusão dos modelos finalistas no conjunto de teste (2022). (Esquerda) Random Forest apresenta equilíbrio entre FP e FN; (Direita) XGBoost maximiza recall às custas de mais falsos alarmes.

4.3 Importância de features

A análise de importância de features revela quais atributos mais influenciam as predições do modelo vencedor. A Figura 2 apresenta as 15 features mais importantes segundo o critério de redução de impureza Gini.

Principais achados:

1. **Domínio de features financeiras:** LOG_BENS (patrimônio declarado) é a feature mais importante em ambos os modelos. Essa variável representa aproximadamente 16% da capacidade preditiva do modelo, confirmando a hipótese de que recursos financeiros são determinantes para sucesso eleitoral no sistema brasileiro.

2. **Vantagem da incumbência:** IS_REELEICAO aparece consistentemente como uma das features mais importantes (∼9-10%), evidenciando o fenômeno de *incumbency advantage* documentado na literatura de ciência política – candidatos que buscam reeleição têm vantagem significativa sobre desafiantes, mesmo controlando por recursos [14, 17].

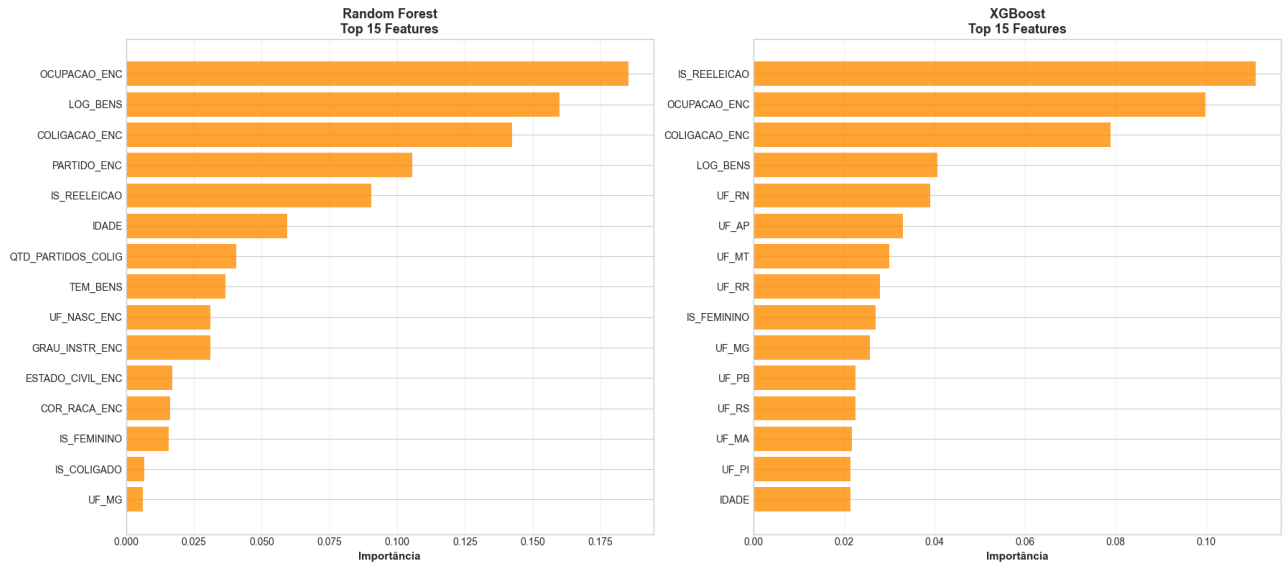

Figura 2: Top 15 features mais importantes por modelo. (Esquerda) Random Forest; (Direita) XGBoost. Ambos convergem em patrimônio (LOG_BENS) e reeleição (IS_REELEICAO) como principais preditores.

3. **Capital político:** PARTIDO_ENC e OCUPACAO_ENC aparecem no top 5, sugerindo que a afiliação partidária e perfil ocupacional capturam sinais de capital político que transcendem recursos financeiros diretos. Ocupações de prestígio (médicos, advogados, empresários) e partidos tradicionais elevam probabilidade de eleição.

4. **Efeito regional:** UFs específicas (SP, RJ, MG) aparecem entre as top 15 features, indicando que contextos estaduais influenciam dinâmica eleitoral mesmo após controlar por outros fatores. São Paulo (UF_SP) é particularmente relevante devido ao tamanho do colégio eleitoral.

Nota-se que features demográficas como idade, gênero, escolaridade e raça têm uma importância relativamente baixa (<5% cada), sugerindo que, condicionado em recursos financeiros e capital político, características demográficas exercem papel secundário na predição de sucesso eleitoral.

4.4 Curvas de desempenho

A Figura 3 apresenta as curvas ROC e Precision-Recall dos modelos finalistas. Ambos os modelos alcançam AUC-ROC superior a 0.90, confirmando excelente capacidade de ranqueamento entre candidatos eleitos e não eleitos em diferentes limiares de decisão.

No entanto, a curva Precision-Recall revela diferenças mais sutis entre os modelos: Random Forest (AUC-PR = 0.560) mantém precisão mais estável em diferentes níveis de recall, enquanto XGBoost (AUC-PR = 0.526) apresenta queda abrupta de precisão em recalls elevados. Este comportamento reflete o trade-off fundamental em dados desbalanceados: aumentar recall (capturar mais candidatos eleitos) inevitavelmente reduz precisão (gera mais falsos positivos).

A superioridade da curva PR do Random Forest sobre a baseline horizontal (prevalência = 0.054) em toda sua extensão indica que o modelo adiciona valor preditivo real em comparação

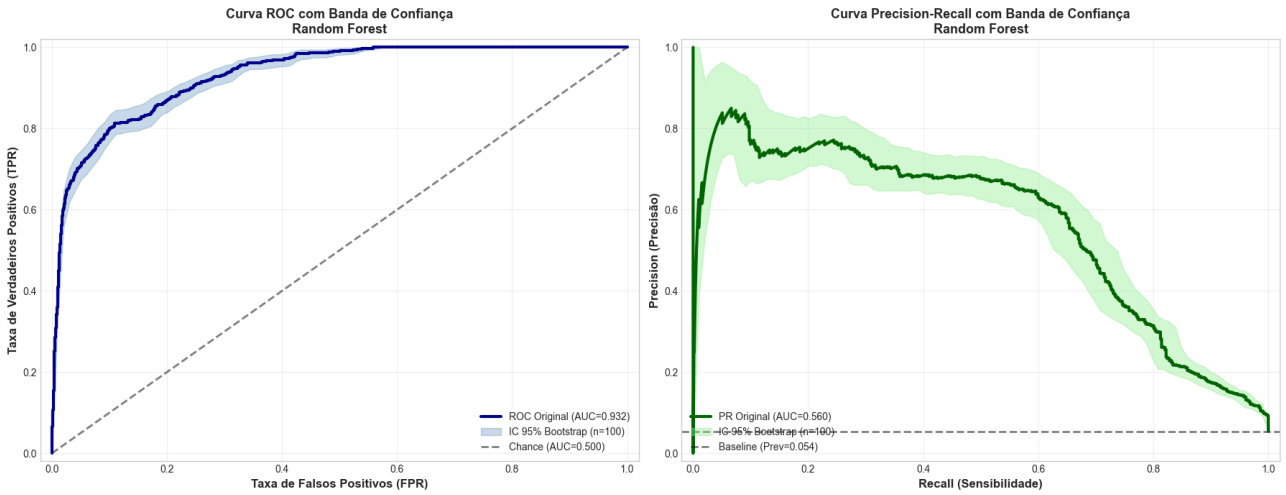

Figura 3: Curvas de avaliação dos modelos finalistas. (Esquerda) Curva ROC mostrando trade-off entre Taxa de Verdadeiros Positivos (TPR) e Taxa de Falsos Positivos (FPR); (Direita) Curva Precision-Recall, mais informativa para classes desbalanceadas [18]. Random Forest (AUC-PR=0.560) supera XGBoost (AUC-PR=0.526) e baseline (prevalência=0.054).

com uma estratégia aleatória proporcional à distribuição de classes. A área sob a curva PR (AUC-PR = 0.560) é aproximadamente 10 vezes superior à baseline, confirmando a capacidade preditiva do modelo.

4.4.1 Sensibilidade ao threshold de decisão

A Figura 4 investiga como diferentes limiares de classificação (threshold) afetam as métricas de desempenho do modelo vencedor (Random Forest).

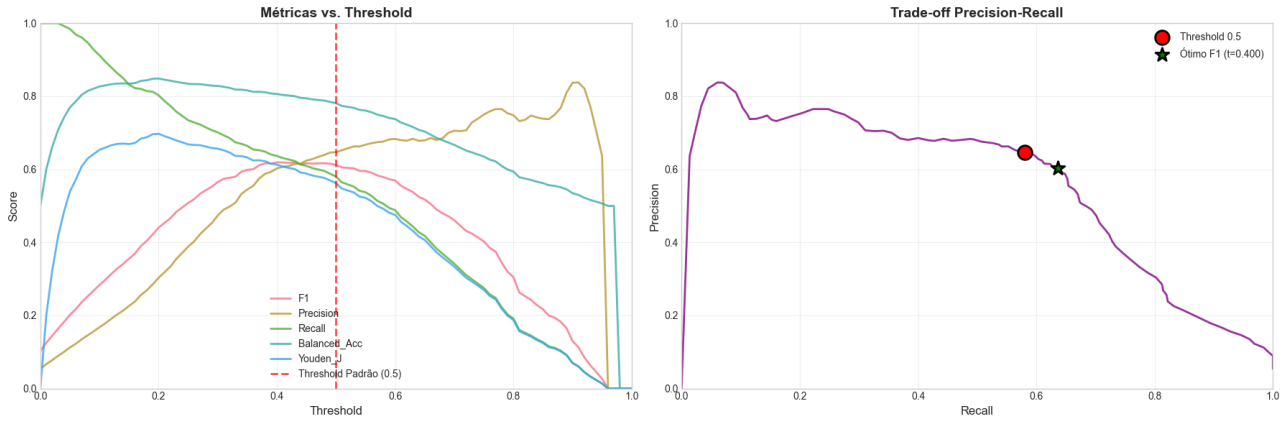

Figura 4: Análise de sensibilidade ao threshold de classificação para Random Forest. (Esquerda) Evolução das métricas em função do threshold; (Direita) Trade-off Precision-Recall com ponto ótimo para F1-score destacado. O threshold padrão (0.5) não maximiza F1; threshold ideal ≈ 0.40.

A análise revela que o threshold padrão (0.5) não maximiza F1-score. O ponto ótimo ocorre em threshold ≈ 0.40, onde F1 atinge seu valor máximo de 0.61 (comparado a 0.60 no threshold padrão). Este threshold mais permissivo aumenta recall de 0.57 para aproximadamente 0.65 com perda moderada de precisão (de 0.64 para 0.60).

Este comportamento reflete o custo assimétrico de erros no contexto eleitoral: em um cenário de screening inicial de candidatos viáveis, falsos negativos (perder candidatos eleitos) podem ser considerados mais custosos que falsos positivos (sinalizar candidatos que não se elegem). Um threshold reduzido permite capturar mais candidatos potencialmente competitivos, aceitando maior taxa de falsos alarmes.

A escolha do threshold ideal depende do objetivo da aplicação:

- **Maximizar F1** ($t \approx 0.40$): Equilíbrio entre precision e recall
- **Maximizar Recall** ($t \approx 0.15$): Identificar o máximo de eleitos, tolerando muitos FPs
- **Maximizar Precision** ($t \approx 0.90$): Predições conservadoras, minimizando FPs

4.5 Análise de erros

A inspeção dos casos mal classificados oferece insights sobre as limitações do modelo e padrões não capturados pelas features disponíveis.

4.5.1 Falsos Positivos (FP = 162 candidatos)

Os falsos positivos – candidatos preditos como eleitos mas derrotados – apresentam perfil característico: alto patrimônio declarado, ocupações de prestígio (empresários, médicos, advogados) e candidaturas em estados competitivos (SP, MG, RJ). Este padrão sugere que o modelo superestima a conversão de recursos financeiros em votos.

Possíveis explicações incluem: (i) candidatos com recursos mas baixo reconhecimento público; (ii) saturação em distritos muito competitivos onde múltiplos candidatos bem financiados disputam as mesmas vagas; (iii) má gestão de campanha apesar de orçamento elevado. A concentração de FPs em partidos grandes (PT, PL, UNIÃO) indica que intra-party competition reduz a vantagem individual de recursos.

4.5.2 Falsos Negativos (FN = 215 candidatos)

Os falsos negativos – candidatos eleitos não identificados pelo modelo – concentram-se em perfis atípicos: baixo patrimônio declarado, ocupações menos comuns e forte inserção em coligações partidárias. Este grupo inclui candidatos eleitos por QP (quociente partidário) que se beneficiam de votações concentradas do partido mesmo sem recursos individuais elevados.

O modelo subestima sistematicamente a força de: (i) coligações bem articuladas (PP, MDB, PSD aparecem frequentemente); (ii) lideranças locais com enraizamento comunitário mas baixa visibilidade em features quantitativas; (iii) efeito "puxador de votos" de candidatos majoritários que elevam coligações inteiras.

Este padrão revela uma limitação conceitual: features baseadas exclusivamente em atributos individuais declarados ao TSE não capturam dinâmicas relacionais (redes de apoio, transferências de voto, força de máquinas partidárias). A incorporação de dados relacionais (grafos de coligações, histórico de votação partidária) poderia reduzir FNs.

4.6 Discussão dos resultados

4.6.1 Performance relativa dos modelos

O ranking final por F1-score no teste temporal foi: Random Forest (0.611) > XGBoost (0.576) > Gradient Boosting (0.572) > Regressão Logística (0.453). A superioridade dos métodos ensemble confirma a hipótese de que relações não-lineares entre features (e.g., interação entre patrimônio e partido) são relevantes para predição de sucesso eleitoral.

A pequena vantagem do Random Forest sobre XGBoost (+3.5 p.p.) sugere que bagging com árvores profundas captura melhor a heterogeneidade dos padrões eleitorais que boosting sequencial com árvores rasas. Uma possível explicação é que RF aprende subpadrões regionais distintos em cada árvore (e.g., dinâmica eleitoral no Nordeste vs. Sudeste), enquanto XGB busca um padrão global médio.

4.6.2 Efeito da ponderação de classes

A ponderação de classes (peso 14:1 para eleitos:não eleitos) foi efetiva para mitigar desbalanceamento, como evidenciado pelo F1-score balanceado dos ensembles. Sem ponderação, o baseline majoritário (predizer sempre "não eleito") alcançaria 94.6% de acurácia mas F1 = 0. A ponderação força o modelo a aprender padrões da classe minoritária sem recorrer a reamostragem artificial.

O trade-off Precision-Recall varia conforme a implementação: Regressão Logística com `class_weight='balanced'` maximiza recall mas sofre com FPs, enquanto Random Forest com mesma configuração apresenta equilíbrio. Esta diferença reflete vieses indutivos: modelos lineares tendem a decisões mais "democráticas" (baixam threshold global), enquanto árvores aprendem thresholds adaptativos por região do espaço de features.

4.6.3 Limiares financeiros práticos

A dominância de LOG_BENS (16% da importância) sugere limiares práticos para viabilidade eleitoral. Estes limiares evidenciam barreira financeira significativa no sistema proporcional brasileiro, levantando questões sobre equidade de acesso ao legislativo. A interpretação causal requer cautela: alto patrimônio pode ser proxy de capital político acumulado, não necessariamente causa direta de vitória.

4.6.4 Sinais de overfitting e convergência

A otimização de hiperparâmetros via GridSearchCV com métrica adequada (F1-score) evitou o overfitting para acurácia que seria catastrófico em cenário desbalanceado – um modelo otimizado para acurácia tenderia a predizer sempre a classe majoritária.

Para grids grandes (> 100 combinações), utilizou-se RandomizedSearchCV com $n_{iter} = 50$ iterações em vez de busca exaustiva. A Figura 5 apresenta a análise de convergência desta estratégia de otimização.

A análise de convergência confirma a adequação da estratégia de busca aleatória:

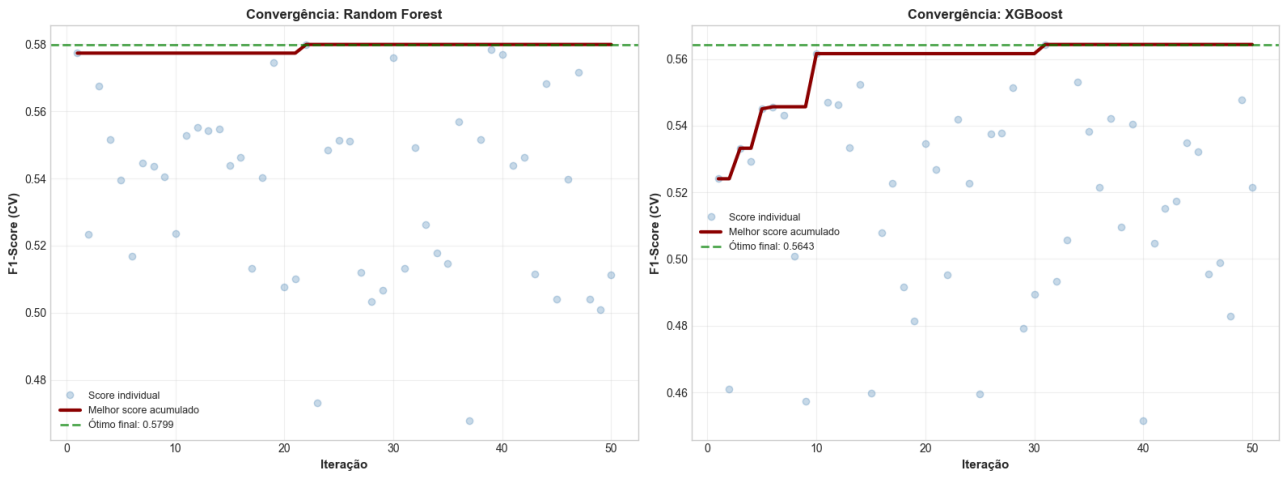

Figura 5: Convergência do RandomizedSearchCV para os modelos finalistas. (Esquerda) Random Forest: grid de 144 combinações; (Direita) XGBoost: grid de 256 combinações. Pontos azuis representam cada combinação testada; linha vermelha mostra o melhor F1-score acumulado. A estabilização em ∼30-40 iterações confirma que $n_{iter} = 50$ foi suficiente.

- **Random Forest:** Melhor F1-score (0.5799) alcançado na iteração 23, sem melhoria nas últimas 20 iterações. A curva vermelha estabiliza completamente, indicando convergência ao ótimo.
- **XGBoost:** Melhor F1-score (0.5643) alcançado na iteração 32, com platô evidente nas iterações finais. Nenhuma melhoria após a iteração 40.
- **Exploração do espaço:** A dispersão dos pontos azuis ao longo de todo o intervalo de scores indica boa cobertura do espaço de hiperparâmetros, evitando concentração em regiões subótimas.

A estabilização da linha vermelha (melhor score acumulado) em aproximadamente 30-40 iterações confirma que $n_{iter} = 50$ foi suficiente para explorar adequadamente grids de 144 (Random Forest) e 256 (XGBoost) combinações possíveis. Este resultado indica que:

1. A busca aleatória encontrou configurações próximas ao ótimo global sem necessidade de busca exaustiva.

2. Aumentar para $n_{iter} = 100$ não traria ganhos significativos de desempenho (linha vermelha já estabilizada), apenas custo computacional adicional.

3. O espaço de hiperparâmetros definido foi bem calibrado – nem muito restrito (convergência imediata) nem muito amplo (sem convergência).

A otimização de hiperparâmetros utilizou F1-score como métrica de seleção, evitando o overfitting para acurácia que seria catastrófico em cenário desbalanceado. Um modelo otimizado para maximizar acurácia tenderia a predizer sempre a classe majoritária (não eleito), alcançando ∼94% de acurácia mas F1-score próximo de zero. A escolha de F1 como métrica de otimização forçou o GridSearchCV a encontrar configurações que equilibram precision e recall, adequadas ao problema de classes raras.

Esta estratégia de busca em dois estágios (screening com grid reduzido + otimização completa com RandomizedSearchCV nos finalistas) permitiu exploração eficiente de grandes espaços de hiperparâmetros sem exaustão computacional, viabilizando a comparação sistemática de múltiplos modelos em tempo razoável.

5 Conclusão

Este trabalho desenvolveu um pipeline reprodutível para predição de sucesso eleitoral de candidatos a Deputado Federal utilizando exclusivamente atributos pré-pleito declarados ao Tribunal Superior Eleitoral (TSE). O diferencial metodológico reside na validação temporal rigorosa – treino em Eleições 2018, teste em Eleições 2022 – que simula um cenário realista de predição intertemporal, contrastando com validações cruzadas convencionais que podem superestimar desempenho ao embaralhar dados temporais.

Foram comparados sistematicamente quatro modelos (Regressão Logística, Random Forest, Gradient Boosting e XGBoost) em um cenário de severo desbalanceamento de classes (∼5% de candidatos eleitos). A otimização de hiperparâmetros utilizou estratégia de dois estágios – screening inicial com grid reduzido seguido de busca refinada nos finalistas – com RandomizedSearchCV para viabilizar exploração de grandes espaços de hiperparâmetros. A ponderação de classes (peso 14:1 para eleitos:não eleitos) e o uso de métricas adequadas ao desbalanceamento (F1-score, AUC-PR, balanced accuracy) foram fundamentais para evitar o colapso do modelo na classe majoritária.

5.1 Contribuição dos membros

- **Artur Saraiva Paschoal:** Artigo científico e validação do código
- **Artur Garcia Sales Barroso:** Planejamento de projeto e código

Referências

[1] AGÊNCIA BRASIL. *Brasil tem mais de 10,4 mil candidatos a deputado federal.* https://agenciabrasil.ebc.com.br/politica/noticia/2022-08/brasil-tem-mais-de-104-mil-candidatos-deputado-federal-veja-lista, 2022. Acesso em: 06 jan. 2026.

[2] Leo BREIMAN. *Random Forests.* In *Machine Learning*, volume 45, pages 5–32, 2001.

[3] Kaio dos Santos BRITO, Alexandre DE OLIVEIRA, Jairo Francisco DE SOUZA, and Felipe GUIMARÃES. *Predicting Elections Based on Social Media Data: A Systematic Review. IEEE Transactions on Computational Social Systems*, 11(3):—, 2021.

[4] Kay H. BRODERSEN, Cheng Soon ONG, Klaas E. STEPHAN, and Joachim M. BUHMANN. *The Balanced Accuracy and Its Posterior Distribution.* In *2010 20th International Conference on Pattern Recognition*, pages 3121–3124, 2010.

[5] Tianqi CHEN and Carlos GUESTRIN. *XGBoost: A Scalable Tree Boosting System.* In

Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 785–794, 2016.

[6] W. CHEN, Z. YANG, J. LI, et al. *A Survey on Imbalanced Learning: Latest Research, Applications and Future Directions. Artificial Intelligence Review*, 57(5):126, 2024.

[7] CÂMARA DOS DEPUTADOS. *Registro de candidaturas bate recorde e mais de 10 mil disputam vaga de deputado federal.* https://www.camara.leg.br/noticias/903111-registro-de-candidaturas-bate-recorde-e-mais-de-10-mil-disputam-vaga-de-deputado-federal/, 2022. Acesso em: 06 jan. 2026.

[8] Remco DE SLEGTE, Fanny VAN DROOGENBROECK, Bram SPRUYT, Koen VERBOVEN, and Vincent GINIS. *Machine Learning in Political Science: A Systematic Review. Political Studies Review*, 22(2):—, 2024.

[9] Thomas G. DIETTERICH. *Ensemble Methods in Machine Learning.* In *Multiple Classifier Systems. MCS 2000. Lecture Notes in Computer Science*, volume 1857, pages 1–15, Berlin, Heidelberg, 2000. Springer.

[10] Daniel GAYO-AVELLO. *A Meta-Analysis of State-of-the-Art Electoral Prediction From Twitter Data. Social Science Computer Review*, 31(4):480–500, 2013.

[11] Guolin KE, Qi MENG, Thomas FINLEY, Taifeng WANG, Wei CHEN, Weidong MA, Qiwei YE, and Tie-Yan LIU. *LightGBM: A Highly Efficient Gradient Boosting Decision Tree.* In *Advances in Neural Information Processing Systems 30*, pages 3146–3154, 2017.

[12] Michael LARIONOV. *Sampling Techniques in Bayesian Target Encoding. arXiv preprint arXiv:2006.01317*, June 2020.

[13] Guillaume LEMAÎTRE, Fernando NOGUEIRA, and Christos K. ARIDAS. *Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. Journal of Machine Learning Research*, 18(17):1–5, 2017.

[14] David R. Mayhew. *Congress: The Electoral Connection.* Yale University Press, New Haven, 1974.

[15] Daniele MICCI-BARRECA. *A Preprocessing Scheme for High-Cardinality Categorical Attributes in Classification and Prediction Problems. ACM SIGKDD Explorations Newsletter*, 3(1):27–32, 2001.

[16] Fabian PEDREGOSA, Gaël VAROQUAUX, Alexandre GRAMFORT, Vincent MICHEL, Bertrand THIRION, Olivier GRISEL, Mathieu BLONDEL, Andreas MÜLLER, Joel NOTHMAN, et al. *Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research*, 12:2825–2830, 2011.

[17] Carlos Pereira and Lúcio Rennó. O que é que o reeleito tem? o retorno: o esboço de uma teoria da reeleição no brasil. *Revista de Economia Política*, 27(4):664–683, 2007.

[18] Takaya SAITO and Marc REHMSMEIER. *The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets*. *PLOS ONE*, 10(3):e0118432, 2015.

[19] SCIKIT-LEARN DEVELOPERS. *GridSearchCV* — scikit-learn documentation. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html, 2026. Acesso em: 06 jan. 2026.

[20] SCIKIT-LEARN DEVELOPERS. *StratifiedKFold* — scikit-learn documentation. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html, 2026. Acesso em: 06 jan. 2026.

[21] TRIBUNAL SUPERIOR ELEITORAL. *Portal de Dados Abertos do TSE*. https://dadosabertos.tse.jus.br/. Acesso em: 06 jan. 2026.

[22] Meng-Hsiu TSAI, Yingfeng WANG, Myungjae KWAK, and Neil RIGOLE. *A Machine Learning Based Strategy for Election Result Prediction*. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 1408–1410, 2019.

[23] Francisco J. VALVERDE-ALBACETE and Carmen PELÁEZ-MORENO. *100% Classification Accuracy Considered Harmful: The Normalized Information Transfer Factor Explains the Accuracy Paradox*. *PLOS ONE*, 9(1):e84217, 2014.