Comparação de Algoritmos na Detecção de Bots em Redes Sociais

Gabriela Alcaide, Rodrigo Gonçalves Cardoso

[1]Escola de Artes, Ciências e Humanidades – Universidade de São Paulo (USP)
São Paulo, São Paulo - Brasil

{gabriela.alcaide,digo.gcardoso}@usp.br

Resumo. *Bots são perfis automatizados em Redes Sociais (com participação nula ou parcial de ação humana). Podem ser utilizados para diversos fins, alguns deles maliciosos, como: inibir outros usuários, propagar informações falsas e simular tendências. Este trabalho apresenta um estudo de caso sobre Bots na Rede Social Bluesky, com pelo menos uma publicação sobre as eleições municipais paulistanas de 2024. Visa-se comparar o desempenho de diferentes algoritmos e ramos de informações na resolução do problema. Percebeu-se que Random Forest apresenta os melhores resultados gerais, seguido proximamente por Naive Bayes. Combinar todos os tipos de dados demonstrou os melhores resultados.*

Palavras-chave. *Bots; Detecção; Redes Sociais; Random Forest; Naive Bayes; Regressão Logística; Árvore de Decisão.*

1. Introdução

Em seu significado mais primitivo, Bot é um sistema automatizado que simula comportamento humano, com vista a atender aos interesses de seus desenvolvedores. Ao longo dos anos, apresentaram crescimento exponencial em sua capacidade de interação com humanos [Michalski et al. 2019].

Nesse cenário de expansão, os Bots conquistaram presença nas Redes Sociais, atuando como perfis autônomos que simulam usuários humanos. Essa presença é extremamente considerável, gerando mais de 50% do tráfego na internet [Colissi et al. 2021].

Os Bots de Redes Sociais podem ser utilizados para quaisquer tarefas. O problema se encontra no fato de que, dentre elas, existem diversas de cunho negativo, como divulgação de informações falsas, simulação de tendências e coibição de outros usuários [Colissi et al. 2021].

Dado que o ambiente de Redes Sociais permite propagação de informações e impacto social intensos, especialmente pela elevada quantidade de usuários, [Léu et al. 2019], fica, então, explícita a problemática dos Bots em Redes Sociais.

Tal preocupação torna-se ainda mais latente ao observar contextos eleitorais: o comportamento dos Bots nas Redes Sociais apresenta impacto elevado na realidade política nacional, ao influenciar opiniões e conceitos dos eleitores e, dessa forma, manipular a tomada de decisão. Percebeu-se essa conjuntura, por exemplo, nas eleições presidenciais brasileiras de 2018 [Michalski et al. 2019].

Códigos em <https://github.com/Gabriela-Alcaide/Analise_de_Redes_Sociais>.

Diante desse contexto, fica clara a importância de estudar algoritmos de Detecção de Bots em Redes Sociais, a fim de entender o potencial de cada um e, assim, respaldar processos mais eficientes de cancelamento ou verificação mais atenta de contas com essas características. Objetiva-se, portanto, promover cenários mais harmônicos nas Redes, com impacto, inclusive, em situações eleitorais.

Pretende-se realizar essa análise comparativa a partir de contas da Rede Social Bluesky, as quais devem ter realizado ao menos uma publicação acerca das eleições municipais paulistanas de 2024. Vale destacar que os perfis serão considerados por completo (incluindo publicações que não versam sobre o tema supracitado, contanto que pertençam a contas selecionadas pelo critério definido).

Tem-se a hipótese de que, para qualquer abordagem utilizada, para obter o melhor resultado possível será necessário considerar informações em três ramos principais: dados das contas propriamente ditas (como tempo desde a data de criação), dados sobre o comportamento da conta na rede (frequência de publicações, por exemplo) e resposta de outros usuários à conta estudada (cita-se a quantidade de seguidores).

Além disso, acredita-se que Random Forest e Naive Bayes fornecerão os resultados mais satisfatórios, uma vez que são os mais robustos entre os algoritmos selecionados para os testes.

2. Conceitos Básicos

2.1. Árvore de Decisão

Árvores de Decisão têm, como ideia central, particionar os dados recursivamente, em uma abordagem de Dividir para Conquistar [Garcia 2003].

Essas divisões são realizadas conforme critérios definidos pelo algoritmo em cada nó. Cada partição é sempre binária, e os critérios são formados por preditores (característica considerada naquela divisão) e limiar (valor utilizado para dividir os grupos) [Lira 2022].

A definição dos critérios adotados pelos nós (e, assim, da estrutura da árvore) é feita gulosamente, com uso, por padrão, do Índice de Gini, calculado como segue:

$$\pi_c = \frac{1}{D} \sum_{n \in D} 1_{y_n = c} \qquad e \qquad Gini = \sum_{c=1}^{C} \pi_c (1 - \pi_c)$$

Onde D é a quantidade de elementos no nó e c é uma das classes do problema [Lira 2022].

Cada nó deixa de ser subdivido pelo algoritmo apenas quando todos os elementos pertencentes a ele (ou quando a maioria desses elementos) é representante da mesma classe [Garcia 2003].

As Árvores de Decisão possuem interpretação simples, visualização intuitiva e desempenho vantajoso (dada a simplicidade da técnica) [Santos et al. 2020]. Após gerar sua representação gráfica, basta analisar cada um dos ramos (do nó raiz a cada nó folha) para compreender as regras definidas pelo modelo. [Garcia 2003]

Por outro lado, sofrem grandes variações conforme os dados de treino, o que possivelmente afeta seu desempenho. [Lira 2022]

2.2. Random Forest

O algoritmo Random Forest opera gerando diferentes Árvores de Decisão, cada uma delas treinada a partir de um subconjunto do Conjunto de Treinamento [Lira 2022]. Existem diferentes formas de gerar esses subconjuntos, como a técnica de *Bootstrap Sampling*, são selecionados subconjuntos com repetição. Ressalta-se que, dada a natureza aleatória da seleção dos elementos de cada subconjunto, é possível que elementos do Conjunto de Treinamento não sejam selecionados para nenhum subconjunto [Rigatti 2017].

Todas as Árvores criadas são, então, usadas para predição. A predição final do algoritmo é dada pela classe majoritária (dada pela maioria das Árvores) predita para cada registro do conjunto de teste [Lira 2022].

O desempenho do algoritmo é dado pela média das métricas de cada uma das Árvores. Com isso, busca-se reduzir a instabilidade encontrada na Árvore de Decisão [Lira 2022]. Isso ocorre pela capacidade do algoritmo de captar regras de forma mais abrangente e relações não-lineares [Rigatti 2017].

2.3. Regressão Logística

A Regressão Logística é um modelo que busca predizer a probabilidade de dado evento da classe alvo ocorrer com base em variáveis independentes [Gonçalves et al. 2013]. Por definição, a variável alvo da predição deve ser binária [Radünz 1992].

Conforme esta probabilidade, o modelo prediz a qual classe cada elemento da amostra pertence [Gonçalves et al. 2013].

2.4. Naive Bayes

Naive Bayes tem como centro o Teorema de Bayes, calculando a probabilidade de que cada elemento e do conjunto de treino assuma a classe c, dado esse elemento: $P(c|e)$.

As probabilidades necessárias ao cálculo derivam da frequência com que cada valor aparece (em atributos categóricos) ou da densidade de probabilidade (em atributos numéricos) [Webb 2017].

O algoritmo assume que as características de cada registro (*features*) são independentes dentro de cada classe. No entanto, em muitos estudos, essa questão não é considerada e, ainda assim, o modelo tende a apresentar bons resultados [Rish 2001].

2.5. Balanceamento dos Dados

Diante de conjuntos desbalanceados de dados, os modelos de aprendizado tendem a não capturar satisfatoriamente os padrões da classe minoritária. Podem, então, aprender que a melhor solução é predizer predominantemente a classe majoritária [Prati et al. 2003].

Para evitar esse comportamento, deve-se balancear o conjunto de treinamento, de forma a equalizar a proporção das classes. Existem duas formas de fazer isso:

- Under-sampling: elimina exemplos da classe majoritária. Potencialmente, gera perda de informação;
- Over-sampling: cria exemplos para a classe minoritária, normalmente criando cópias de alguns de seus exemplos. Possivelmente, aumenta as chances de overfitting (pela possibilidade de que o modelo responda às duplicatas). [Prati et al. 2003]

3. Revisão da Literatura

[Leite et al. 2020] buscou encontrar regras para compreensão e detecção de bots, utilizando aprendizado supervisionado (Árvore de Decisão e Indução de Regras). Utilizou o conjunto de dados de Cresci et. al., desenvolvido no projeto Fake Project. O critério da Árvore de Decisão foi ganho de informação e, para a Indução de Regras, utilizaram-se os algoritmos PART, JRIP e APRIORI.

[Santos et al. 2020], por sua vez, buscou comparar algoritmos de aprendizado supervisionado para detecção de bots. Apoiou-se em um conjunto de 111.530 postagens feitas por 800 usuários do Twitter durante o segundo turno das eleições presidenciais brasileiras de 2018. Os usuários foram rotulados manualmente e balanceados para treinamento.

Para avaliação dos modelos, optou-se pela análise de suas precisões, sensibilidades e medidas-F. Os melhores resultados foram obtidos com Random Forest e Naive Bayes, mas os autores recomendam a Árvore de Decisão, uma vez que é mais simples e apresenta resultados quase tão satisfatórios.

A literatura também registra a utilização de atributos de vários tipos em conjunto (das contas, comportamentais, temporais, topológicas, gráficos e de conteúdo), aplicando-os ao algoritmo XGBoost [Owais et al. 2023]. O conjunto de dados foi obtido pela agregação de conjuntos do Twibot-22, resultando em informações sobre um milhão de usuários do Twitter. Os atributos foram selecionados conforme sua correlação com a classificação em bot ou não-bot.

[Rodríguez-Ruiz et al., 2020], por sua vez, aplica algoritmos supervisionados de one-class classification para detectar bots no Twitter, utilizando, assim, apenas exemplos de contas humanas no treinamento. O conjunto de dados foi fornecido por Cresci et al. e foram testados os principais algoritmos detectores de anomalias: BTPM, BRM e OCKRA. Considera, ainda, que há diferentes tipos de bots.

Os resultados obtidos foram altamente satisfatórios, especialmente para o BTPM, com AUC de 0,921.

[Lira 2022] aplicou aprendizado supervisionado e não-supervisionado em conjunto para detecção de bots, com principal objetivo de obter bom desempenho inter-domínio. O conjunto de dados é composto por dados do Twitter, a partir de Cresci-2015, Cresci-2017 e outros conjuntos que exigiram enriquecimento (obtenção de mais informações via API).

No pré-processamento, foi aplicado K-Means aos dados e, posteriormente, Random Forest. Assim, obtiveram-se resultados melhores que o rotineiro para testes entre domínios.

Alternativamente, [Léu et al. 2019] aplicam Random Tree e Regressão Linear para detectar bots no Twitter. O conjunto de dados de 635.957 usuários foi coletado entre 08 e 28 de outubro de 2018. Para o treinamento e validação dos modelos, foram selecionados 642 usuários aleatórios, rotulados manualmente (com 65 bots).

Os pesquisadores optaram por não balancear o conjunto de treinamento. Com Random Tree, obteve-se alta precisão, sem classificar erroneamente humanos como bots, mas sem detectar cerca de metade dos bots. Na Regressão Linear, a precisão cai, enquanto a revocação cresce. Combinando ambas as técnicas, é possível aumentar a revocação em intervalos específicos. Determinados nós e intervalos demonstraram-se altamente precisos na detecção de bots.

[Lira et al. 2021] aplicaram técnicas de agrupamento, seleção de variáveis e Árvore de Decisão para detectar bots. O conjunto de dados é composto por 500 usuários rotulados manualmente, coletados entre 16 de março e 30 de abril de 2020 e originalmente desbalanceados.

Combinando as três técnicas, notou-se uma melhoria da precisão e da medida F na detecção de bots em relação ao baseline adotado. Foram utilizados Random Forest, Random Tree e K-Means para os melhores resultados.

Também é possível aplicar PLN à detecção de bots [Santos et al. 2022], para geração de atributos que possam contribuir para o estudo. O conjunto de dados é composto por 800 usuários e 111.530 posts publicados durante o segundo turno das eleições de 2018.

Ao comparar os resultados dos algoritmos de classificação Random Forest e Naive Bayes e seleção de variável com e sem utilização de PLN, a utilização de atributos gerados por PLN juntamente com informações do usuário tornou os resultados mais satisfatórios.

[Shaabani et al. 2018] propuseram detectar bots sem considerar a estrutura da rede em si, informações de caminho de cascatas, conteúdo das publicações e informações dos usuários. O conjunto de dados incluiu cerca de nove milhões de tweets ou retweets relacionados ao grupo terrorista ISIS, entre fevereiro e maio de 2016.

Uma das técnicas implementadas foi a Threshold-Based Selection Approach, estudando a classificação dos usuários conforme uma ou mais métricas causais. A outra técnica foi Label Propagation Selection Approach, que classifica bots iterativamente, caso tenham pontuação-limite (em relação aos selecionados até então) alta o suficiente. Obteve bons resultados, com precisão de 75%.

[Braz et al. 2018] propuseram que a utilização do conteúdo original das mensagens nas postagens da rede social Twitter para detecção de bots aprimoraria as técnicas de detecção de bots. Para provar isso, foi realizado um processo utilizando DetBot Beta (Rede Neural Convolucional) e clusterização, a fim de classificar mensagens e perfis como suspeitos ou não e gerar novos atributos.

Com o conjunto de dados de 40 mil contas e 3.71 milhões de tweets, coletados entre 30 de dezembro de 2009 e 2 de agosto de 2010, o estudo obteve um resultado

positivo, sugerindo que o conteúdo original das mensagens é um fator que ajuda na detecção de bots.

4. Materiais e Métodos

Esta seção descreve o Conjunto de Dados e as resoluções do problema através de quatro algoritmos distintos: Árvore de Decisão, Random Forest, Regressão Logística e Naive Bayes.

4.1. Conjunto de Dados

O Conjunto de Dados foi desenvolvido inteiramente pelos autores dessa pesquisa, a partir de dados extraídos da API do Bluesky via script Python.

Após configurações iniciais de acesso à API, a construção da base de dados partiu de uma lista de palavras-chave, relacionadas ao primeiro turna das eleições municipais paulistanas de 2024: “eleições”, “São Paulo”, “Boulos”, “Ricardo Nunes”, “Tabata”, “Datena” e “Marçal”.

Posteriormente, foram coletados os usuários responsáveis pelas publicações obtidas. Esses usuários se tornaram as entidades efetivas da base de dados, e possuem a garantia de terem falado ao menos uma vez sobre o tema em questão.

Foram obtidas informações básicas de cada conta, com destaque para “handle” (identificador único), “displayName” (nome do perfil), “avatar” (imagem de perfil), “banner” (imagem de banner), “createdAt” (data de criação do perfil), “followersCount” (quantidade de seguidores), “followsCount” (quantas contas está seguindo) e “postsCount” (quantidade de publicações).

Além disso, obtiveram-se dados sobre todas as publicações de cada usuário, incluindo seus conteúdos e a interação de outros usuários diante delas.

Paralelamente, foi necessário obter os seguidores de cada usuário da base de dados, para construção de grafos. Derivou-se dessa informação as contas que cada usuário do conjunto segue.

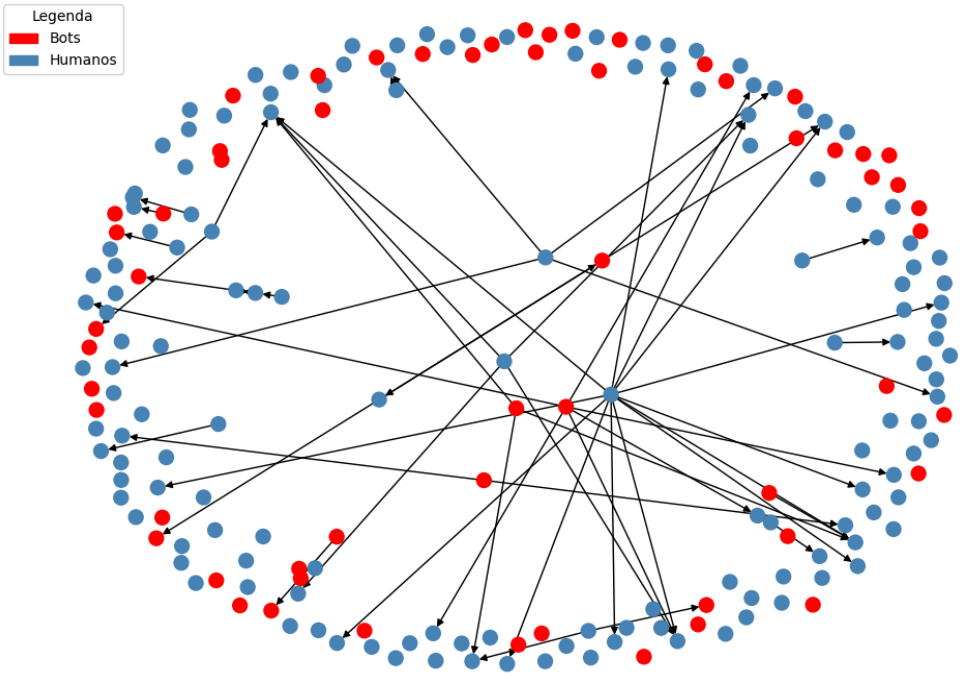
O Conjunto de Dados final contém 203 usuários do aplicativo Bluesky, ativos na data de coleta (19 de outubro de 2024), que publicaram ao menos uma vez sobre o primeiro turno das eleições municipais paulistanas de 2024. As contas foram classificadas manualmente pelos autores, resultando em 58 bots (aproximadamente 30% do total de instâncias da base) e 145 humanos (cerca de 70% do total). Vale ressaltar que não há distinção entre os diferentes tipos de bot (social, de notícia e ciborgues).

Alguns dos dados brutos obtidos da API foram diretamente transpostos para a base de dados, enquanto outros foram manipulados. Culminou-se, então, em 21 atributos para cada conta: “handle” (identificador único de cada usuário); “displayName” (nome de perfil); “imagem_perfil” (existência ou não de imagem de perfil); “imagem_banner” (existência ou não de imagem de banner); “data_criacao” (data de criação da conta); “numPosts” (quantidade de publicações até a data de extração); “numDias” (idade, em dias, até a data de extração); “posts/dia” (média de publicações por dia); “seguidores” (quantidade de seguidores); “seguindo” (quantidade

de contas sendo seguidas pela conta); “seguindo/seguidores” (relação de número de pessoas seguidas pela conta sobre o número de seguidores); “sentimento_mais_frequente” (sentimento predominante nas publicações do usuário); “porc_substantivos_proprios” (de todas as palavras das publicações do usuário, porcentagem classificada como substantivo próprio); “porc_substantivos_comuns” (porcentagem classificada como substantivo simples); “porc_verbos” (porcentagem classificada como verbo); “porc_adjetivos” (porcentagem classificada como adjetivo); “media_respostas” (média de respostas às publicações do usuário); “media_repostagens” (média de repostagens das publicações do usuário); “media_curtidas” (média de curtidas nas publicações do usuário); “media_citacoes” (média de citações nas publicações do usuário); “classificacao” (rótulo da conta, em “bot” ou “humano”).

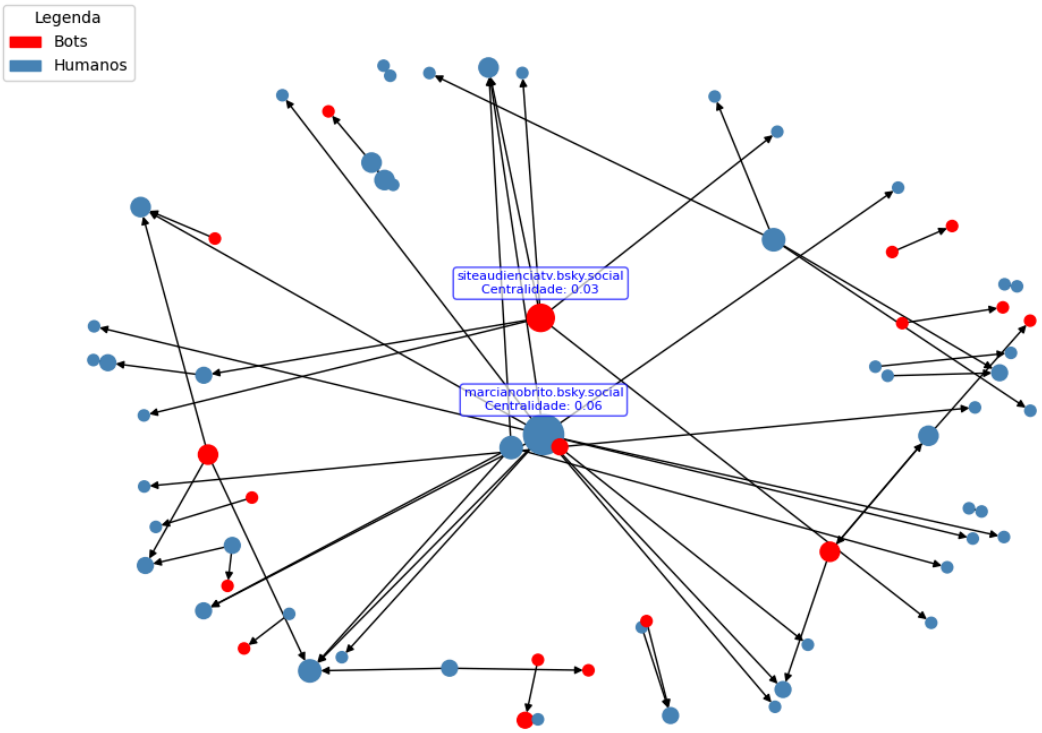
Ademais, foram criadas seis bases de dados auxiliares para teste da hipótese 1. As bases foram separadas em “atributos da conta” (“imagem_perfil”, “imagem_banner”, “numDias”), “comportamentos da conta” (“numPosts”, “posts/dia”, “seguindo”, “sentimento_mais_frequente", “porc_substantivos_proprios”, “porc_substantivos_comuns”, “porc_verbos”, “porc_adjetivos”, “media_citacoes”), “comportamentos terceiros” (“seguidores”, “media_respostas”, “media_repostagens”, “media_curtidas”) e suas respectivas combinações 2 a 2. Houve a inserção do atributo “seguindo/seguidores” na combinação de “comportamentos terceiros” com “comportamento da conta”. Ele não contava em nenhuma base auxiliar exclusiva de um ramo, por razões semânticas.

Por ser um estudo em Redes Sociais, é interessante a complementação com grafos. Para isso, obtiveram-se os seguidores de cada usuário, selecionando apenas aqueles que pertenciam à base de dados e inverteu-se o sentido e significado da relação (a partir da lógica de que, se A é seguidor de B, B é seguido por A). Assim, a aresta A → B deve ser interpretada como “A segue B”.



Foram obtidas 60 arestas (indicando existência de 60 relacionamentos de seguir dentro da sub-rede), média de 0,6 seguidas dentro da sub-rede para cada conta (grau de saída médio) e densidade do grafo em 0.001 (demonstrando que muitos relacionamentos possíveis não existem).

Posteriormente, foi desenvolvido grafo com o mesmo propósito, porém que se diferencia por atribuir tamanhos aos nós proporcionais à centralidade de grau do usuário que representam. Pôde-se observar que muitos nós deixaram de ser representados, indicando que não possuem importância à sub-rede, em questão de centralidade de grau. O humano e o bot mais relevantes nesse sentido estão destacados no grafo. Observar abaixo.



4.3. Balanceamento dos Dados

Das 203 contas pertencentes ao Conjunto de Dados, 70% é referente a humanos e apenas 30% a bots. Tem-se, portanto, um Conjunto desbalanceado, com a classe alvo do problema (“bot”) em quantidade minoritária.

Foi necessário, portanto, balancear o conjunto de treinamento, para evitar problemas em potencial, conforme explicado na seção “Conceitos Básicos”. Optou-se pela técnica de Undersampling, a fim de priorizar que não ocorresse overfitting.

Inicialmente, verificou-se que não seria possível constituir o Conjunto de Treinamento com 70% dos dados (142 registros), pois, para que ele apresentasse equilíbrio entre as classes, seria necessário haver, nele, 71 bots, o que não é viável no Conjunto de Dados disponível (58 bots) via Undersampling, uma vez que, para esse conjunto de dados, só seria possível ultrapassar esse número aumentando a quantidade de bots (Oversampling).

Por isso, partiu-se de selecionar, aleatoriamente, 70% dos bots (resultando em 41 bots) para constituir parte do Conjunto de Treinamento. Para garantir o balanceamento nesse Conjunto, foram então escolhidos, também aleatoriamente, 41 humanos para complementar o conjunto. Dessa forma, o Conjunto de Treinamento utilizado em todos os algoritmos é composto por 82 elementos, sendo 50% deles bots e 50% humanos.

O Conjunto de Treinamento foi formado pela seleção de todos os registros do Conjunto de Dados fora do Conjunto de Treinamento. Totalizou-se, nesse conjunto, 121 elementos, sendo 15% deles bots (17 bots) e 85% humanos (104 humanos).

4.4. Transformação dos Dados

Para correta aplicação dos algoritmos, percebeu-se a necessidade de algumas transformações dos dados.

A Árvore trabalha com decisões binárias em cada nó, conforme critérios no formato especificado anteriormente. Dessa forma, é necessário que os atributos do Conjunto de Dados sejam numéricos ou binários [Garcia 2003]. O mesmo ocorre para Random Forest, uma vez que aplica múltiplas Árvores de Decisão.

Em Naive Bayes, os atributos dos registros devem ser numéricos ou categóricos [Rish 2001] e, na Regressão Logística, devem ser numéricos (com variável dependente binária) [Radünz 1992].

Nesse sentido, foram operadas as seguintes transformações:

- "handle" (identificador único dos registros) foi mapeado para números inteiros únicos, utilizando a classe LabelEncoder, da biblioteca scikit-learn;
- "displayName" foi binarizada conforme a existência (1) ou não (0) de valores atribuídos a ela;
- "imagem_perfil" e "imagem_banner" já haviam sido binarizados. Essa decisão foi tomada com respaldo na Revisão de Literatura - a existência ou não desses elementos nas contas parece mais relevante do que seus conteúdos em si;
- "data_criacao" foi desconsiderado, uma vez que, dele, foi extraído o dado sobre idade das contas ("numDias"), já numérico. Dessa forma, não houve perda de informação;
- "sentimento_mais_frequente", por ser categórico, foi transformado em diferentes atributos binários (um para cada classe), com OneHotEncoder, do scikit-learn. Devido à biblioteca utilizada para extração de sentimentos das publicações, três classes são possíveis para esse atributo - "Negativo", "Neutro" e "Positivo". Entretanto, no Conjunto de Dados deste estudo, apenas duas delas estão presentes nos registros ("Negativo" e "Positivo"). Por isso, o OneHotEncoder criou dois atributos em substituição a "sentimento_mais_frequente";
- "classificacao" das contas foi binarizada, atribuindo 0 para "humano" e 1 para "bot" (ocorrência do fenômeno em estudo).

4.5. Algoritmos

Para todas as implementações de algoritmos realizadas, tanto para o teste da hipótese 1 quanto para o teste da hipótese 2, foram utilizadas classes da biblioteca scikit-learn:

- DecisionTreeClassifier para Árvores de Decisão;

- RandomForestClassifier para Random Forest;
- LogisticRegression para Regressão Logística;
- ComplementNB para Naive Bayes (existem outras classes, porém essa é a com melhor adaptação a cenários desbalanceados).

Na Árvore de Decisão, inicialmente foi testado um modelo sem definição de profundidade máxima (deixando esse ajuste a cargo do modelo) e, posteriormente, com diferentes especificações para essa definição (via parâmetro "max_depth").

Nos demais algoritmos, bastou treinar os modelos gerados pelas classes, sem variações.

5. Resultados

5.1 Ramos de Informações

Foram aplicados todos os algoritmos nas sete bases de dados criadas, com a finalidade de averiguar a hipótese de que os melhores resultados são obtidos com dados em três ramos (atributos da conta, comportamento da conta e interação da rede com a conta). O resultado pode ser observado na seguinte tabela a seguir.

Combinação de Atributos	Precisão	Recall	F1-Score	Algoritmo
Todos os dados	86%	77%	80%	Random Forest
Atributos Conta + Comportamento Conta	86%	76%	79%	Random Forest
Comportamento Conta	85%	74%	78%	Random Forest
Comportamento Conta + Comportamento Terceiros	85%	74%	78%	Random Forest
Atributos Conta + Comportamento Terceiros	73%	79%	76%	Naive Bayes
Comportamento Terceiros	72%	73%	72%	Naive Bayes
Atributos Conta	78%	55%	62%	Random Forest

O algoritmo com melhores resultados em cada base tem seu conjunto de métricas e nome escritos na tabela. Eles foram escolhidos utilizando a maior medida F1 para aqueles dados.

Dessa maneira, a hipótese não foi refutada, uma vez que o conjunto com os três ramos de informação apresentou melhores resultados. No entanto, vale ressaltar os bons

resultados considerando apenas o “Comportamento da Conta”, uma vez que geram resultados satisfatórios com menor dimensionalidade dos dados.

5.2 Melhores Algoritmos

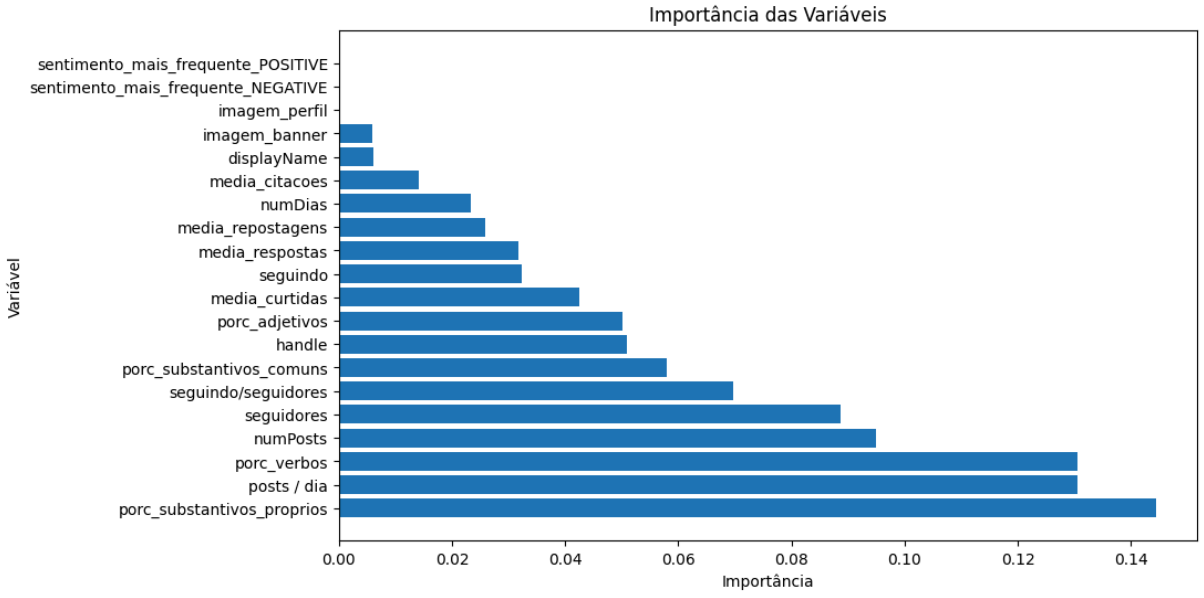
Diante do resultado apresentado anteriormente, a comparação descrita abaixo se baseia no Conjunto completo de dados (com atributos nos três ramos), uma vez que este possui as melhores métricas e as informações já foram extraídas.

Entre os algoritmos testados, Random Forest apresentou os melhores resultados, com medida F1 de 80%, precisão de 86% (contas classificadas como “bot” que, de fato, são bots) e revocação de 77% (bots identificados).

Apesar de não ter demonstrado a maior taxa de detecção de bots entre os algoritmos testados, como será apresentado adiante, a solução com Random Forest demonstrou-se mais equilibrada, pois apresenta precisão e revocação consideráveis, resultando na maior F1 entre os testes realizados.

Dessa forma, apesar de não identificar 23% dos bots, permite considerável confiança de que aqueles classificados nessa classe de fato pertencem a ela: apenas 14% das contas assim classificadas são, na verdade, humanas.

Em uma análise mais profunda do modelo gerado, constata-se que os atributos considerados mais importantes por ele são: porcentagem de substantivos próprios, posts/dia, porcentagem de verbos, número de publicações e quantidade de seguidores. Segue lista completa.



O algoritmo com o segundo melhor resultado foi Naive Bayes, com medida F1 de 76%, precisão de 83% (contas classificadas como “bot” que, de fato, são bots) e revocação de 73% (bots identificados).

Este algoritmo apresenta todas as métricas ligeiramente inferiores ao Random Forest: F1 04% menor, precisão 03% menor e revocação 04% menor. Dessa forma, classifica mais contas erroneamente como “bots” e identifica menor bots.

Dessa forma, no caso deste estudo, não há quaisquer vantagens no uso de Naive Bayes.

Posteriormente, apresenta-se o resultado da Regressão Logística, que foi consideravelmente inferior em todas as métricas analisadas: F1 de 72%, precisão de 78% (contas classificadas como "bot" que, de fato, são bots) e revocação de 69% (bots identificados).

Esse algoritmo, portanto, não mostra vantagens no caso deste estudo, pois classifica contas erroneamente como "bots" em 08% a mais dos casos e identifica 08% a menos dos bots.

Para fins de aprofundamento, considera-se, ainda assim, válido trazer algum aprofundamento sobre este resultado, uma vez que, apesar de inferior ao Random Forest, é consideravelmente satisfatório.

Ao observar os coeficientes atribuídos pelo modelo a cada atributo do Conjunto de Dados, pode-se interpretar que, para o modelo, bots tendem a possuir maior proporção de substantivos próprios em suas publicações, mais publicações, maior média de citações e mais seguidores. O sentimento predominante nas publicações aparenta ser irrelevante (devido a coeficiente nulo ou ínfimo).

Atributo	Coeficiente
porc_substantivos_proprios	0.003564
displayName	0.002477
numPosts	0.001780
media_citacoes	0.001275
seguidores	0.000043
seguindo	0.000027
sentimento_mais_frequente_POSITIVE	0.000000
imagem_perfil	-0.000149
sentimento_mais_frequente_NEGATIVE	-0.000149
porc_adjetivos	-0.000427
media_respostas	-0.000510
porc_substantivos_comuns	-0.001383
porc_verbos	-0.001903
imagem_banner	-0.002177
media_repostagens	-0.003906
seguindo/seguidores	-0.005436
numDias	-0.006277
handle	-0.007093
media_curtidas	-0.028800
posts / dia	-0.047568

As Árvores de Decisão, com diferentes alturas, apresentaram os piores resultados. Todas resultaram em medidas F1 inferiores a 38%, precisões de, no máximo, 26% (contas classificadas como "bot" que, de fato, são bots) e revocação de até 88% (bots identificados).

Aqui, vale fazer uma ressalva. A Árvore de Decisão de altura 01 apresenta a maior revocação entre todos os algoritmos testados: identifica 88% dos bots, contra 86% do Random Forest e valores menores dos demais algoritmos.

No entanto, a precisão desse algoritmo é de apenas 25%, a menor entre os algoritmos testados: das contas que classifica como “bot”, 75% não o são de fato. Dessa forma, percebe-se uma tendência do modelo a tender para predições de “bot”, o que leva à elevada revocação e à ínfima precisão.

Por isso, o algoritmo não é eficiente como um todo, o que fica nítido por sua medida F1 de apenas 38%. Sua revocação é ligeiramente superior ao do Random Forest (apenas 11% acima), enquanto sua precisão é 61% inferior à do Random Forest. Claramente, não há ganhos reais, nem bom custo-benefício.

Segue a tabela comparativa das métricas gerais de cada algoritmo.

Algoritmo	precisão	revocação	Medida F1
Random Forest	86%	77%	80%
Naive Bayes	83%	73%	76%
Regressão Logística	78%	69%	72%
Árvore altura 1	25%	88%	38%
Árvore altura 2	26%	65%	37%
Árvore altura 3	26%	59%	36%
Árvore altura 4	25%	59%	35%
Árvore altura 5	26%	53%	35%
Árvore altura 6	26%	53%	35%

6. Conclusões

A partir de 203 contas extraídas aleatoriamente da Rede Social Bluesky, o presente estudo buscou comparar diferentes abordagens na Detecção de Bots em Redes Sociais. Para isso, foram implementadas Árvores de Decisão (com variações de altura máxima), Random Forest, Regressão Logística e Naive Bayes.

O Conjunto de Dados que contava com informações das contas, de suas interações na rede e das respostas de outros usuários a elas obteve as melhores medidas. Dessa forma, confirmou-se a primeira hipótese estabelecida.

Além disso, vale ressaltar o bom desempenho da base de dados que possui apenas dados do comportamento do usuário na rede. Sendo assim, para futuros estudos, selecionar apenas esse tipo de atributo pode ser uma técnica interessante para reduzir a dimensionalidade sem perda relevante de eficiência.

Ademais, por meio da comparação das métricas geradas por cada algoritmo (em especial, medida F1, com análise crítica, também, das demais), concluiu-se que o algoritmo com melhor desempenho foi o Random Forest, seguido por Naive Bayes, com métricas próximas. Assim, está, também, verificada a segunda hipótese.

Referências

Braz, P. A. and Goldschmidt, R. R. (2018) “Redes Neurais Convolucionais na Detecção de Bots Sociais: Um Método Baseado na Clusterização de Mensagens Textuais”. In: Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais. https://sol.sbc.org.br/index.php/sbseg/article/view/4262

Garcia, S. C. (2003) “O Uso de Árvores de Decisão na Descoberta de Conhecimento na Área da Saúde”. https://lume.ufrgs.br/handle/10183/4703

Gonçalves, E. B., Gouvêa, M. A., Mantovani, D. M. N. (2013) “Análise de Risco de Crédito com Uso de Regressão Logística”. In: Revista Contemporânea de Contabilidade. v. 10, n. 20, p. 139-160. https://periodicos.ufsc.br/index.php/contabilidade/article/view/2175-8069.2013v10n20p139/25196

Leite, M. A. G. L, Guelpeli, M. V. C. and Santos, C. Q. (2020) “Um Modelo Baseado em Regras para a Detecção de bots no Twitter”. In: Brazilian Workshop on Social Network Analysis and Mining (BRASNAM). Porto Alegre: Sociedade Brasileira de Computação, 2020. p. 37-48.
https://sol.sbc.org.br/index.php/brasnam/article/view/11161

Léu, M. O., Morais, D. M. G., Xavier, F. and Digiampietri, L. A. (2019) “Detecção automática de bots em redes sociais: um estudo de caso no segundo turno das eleições presidenciais brasileiras de 2018”. In: Revista de Sistemas de Informação da FSMA. p. 31-39. https://bibliotecadigital.tse.jus.br/xmlui/bitstream/handle/bdtse/6510/2019_leu_deteccao_automatica_bots.pdf?sequence=1&isAllowed=y

Lira, D. B. (2022) “Detecção de bots no Twitter: Uma abordagem utilizando agrupamento”.https://www.teses.usp.br/teses/disponiveis/100/100131/tde-07122022-101230/publico/Texto_final_PPGSI__Diego_Lira.pdf

Lira, D. B., Xavier, F. and Digiampietri, L. A. (2021) “Combining clustering and classification algorithms for automatic bot detection: a case study on posts about COVID-19”. In: Simpósio Brasileiro de Sistemas de Informação (SBSI). https://sol.sbc.org.br/index.php/sbsi/article/view/17726

Michalski, R., Paula, L. T. (2019) “Os bots de disseminação de informação na conjuntura das campanhas presidenciais de 2018 no Brasil”. https://periodicos.ufmg.br/index.php/moci/article/view/17048/13818

Owais, M., Shoaib, M. and Waseem, M. (2023) “A Holistic Approach for Detecting Socialbots on Twitter: Integration of Diverse Features”. In: Nucleus (Islamabad). p. 201-206. http://www.thenucleuspak.org.pk/index.php/Nucleus/article/view/1312/831

Prati, R. C., Batista, G. E. A. P. A., Monard, M. C. (2003) “Uma experiência no balanceamento artificial de conjuntos de dados para aprendizado com classes desbalanceadas utilizando análise ROC”. In . Chillan: Universidade del BIO-BIO/Sociedade Chilena de Ciencia de la Computacion. https://www.researchgate.net/publication/228784062_Uma_experiencia_no_balancea

mento_artificial_de_conjuntos_de_dados_para_aprendizado_com_classes_desbalanceadas_utilizando_analise_ROC

Radünz, A. (1992) “Regressão Logística”. https://lume.ufrgs.br/bitstream/handle/10183/131354/000454706.pdf?sequence=1

Rigatti, S. J. (2017) “Random Forest”. In: Journal of Insurance Medicine. v. 47, p. 31-39. https://meridian.allenpress.com/jim/article/47/1/31/131479/Random-Forest

Rish, I. (2001) “An empirical study of the naive Bayes classifier”. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=2825733f97124013e8841b3f8a0f5bd4ee4af88a

Ruiz, J. R., Sánchez, J. I. M., Monroy, R., González, O. L.,Cuevas, A. L. (2020) “A one-class classification approach for bot detection on Twitter”. In: Computers & Security 91 101715 p. https://www.sciencedirect.com/science/article/pii/S0167404820300031

Santos, B. L., Ferreira, G. E., Ó, M. T., Braz, R. R. and Digiampietri, L. A. (2020) “Comparação de algoritmos para detecção de bots sociais nas eleições presidenciais no Brasil em 2018 utilizando características do usuário”. In: Revista Brasileira de Computação Aplicada. p. 53-54. https://www.researchgate.net/publication/367931759_Comparacao_de_algoritmos_para_deteccao_de_bots_sociais_nas_eleicoes_presidenciais_no_Brasil_em_2018_utilizando_caracteristicas_do_usuario

Santos, B. L., Ferreira, G. E., Ó, M. T., Braz, R. R. and Digiampietri, L. A. (2022) “Comparison of natural language processing techniques in social bot detection on Twitter during Brazilian presidential elections”. In: Brazilian Journal of Information Systems. https://journals-sol.sbc.org.br/index.php/isys/article/view/2225

Shaabani, E., Guo, R. and Shakarian, P. (2018) “Detecting Pathogenic Social Media Accounts without Content ou Network Structure”. https://www.researchgate.net/publication/325418382_Detecting_Pathogenic_Social_Media_Accounts_without_Content_or_Network_Structure

Webb, G. I. (2016) “Naïve Bayes”. In: Encyclopedia of Machine Learning and Data Mining.https://www.researchgate.net/profile/Geoffrey-Webb/publication/306313918_Naive_Bayes/links/5cab15724585157bd32a75b6/Naive-Bayes.pdf