UPF EDITORA
UNIVERSIDADE DE PASSO FUNDO

RBCA
ISSN 2176-6649

Revista Brasileira de Computação Aplicada, Abril, 2021

DOI: 10.5335/rbca.v13i1.9999
Vol. 13, Nº 1, pp. 53–64
Homepage: seer.upf.br/index.php/rbca/index

ARTIGO ORIGINAL

Comparação de algoritmos para detecção de *bots* sociais nas eleições presidenciais no Brasil em 2018 utilizando características do usuário

Comparison of algorithms for detecting social bots in the 2018 Brazilian presidential elections using user characteristics

Bianca Lima Santos[1], Gabriel Estavaringo Ferreira[1], Marcelo Torres do Ó[1], Rafael Rodrigues Braz[1], Luciano Antonio Digiampietri [1]

[1]Universidade de São Paulo

*bianca_lima@usp.br; estavaringo@usp.br; marcelo.torres.o@usp.br; rafael.braz@usp.br; digiampietri@usp.br

Recebido: 17/06/2020. **Revisado:** 17/09/2020. **Aceito:** 09/11/2020.

Resumo

O uso de *bots* sociais com fins políticos tem se tornado uma preocupação cada vez mais relevante e tem levantado alertas sobre o impacto nas discussões democráticas. Este trabalho apresenta um estudo de caso sobre o uso de *bots* nas discussões políticas durante o período do segundo turno das eleições brasileiras de 2018, visando construir um modelo de detecção automática para *bots* e comparando o uso de algoritmos de inteligência artificial explicáveis e não-explicáveis. Primeiramente foi construído um conjunto de dados rotulando manualmente contas entre *bots* e humanos. Então foram aplicados algoritmos de regressão linear, árvores aleatórias, Bayesiano ingênuo, *multilayer perceptron* e *random forest*. Foi identificado que mesmo algoritmos mais simples e explicáveis como árvore aleatória têm um desempenho semelhante a algoritmos mais complexos como *random forest*. Utilizando apenas características do usuário, foi possível identificar mais de 46% dos *bots*, porém todos modelos apresentaram uma precisão não maior que 52% nessa tarefa.

Palavras-Chave: Bot, classificação, redes sociais, Twitter, eleições, aprendizado de máquinas

Abstract

The use of social bots for political purposes has become an increasingly relevant concern and has raised warnings about the impact on democratic discussions. This paper presents a case study on the use of bots in political discussions during the second round of the 2018 Brazilian elections, aiming to build an automatic detection model for bots and comparing the use of explainable and non-explainable artificial intelligence algorithms. First, a dataset was built by manually labeling accounts as bots or humans. Then linear regression algorithms, random trees, naive Bayesian, multilayer perceptron, and random forest were applied. It was identified that even simple and explainable algorithms, such as random tree, perform similarly to more complex algorithms such as random forest. Using only user's characteristics, it was possible to identify more than 46% of the bots, but all models showed a precision not greater than 52% in this task.

Keywords: Bot, classification, social networks, Twitter, elections, machine learning

1 Introdução

Nas últimas décadas ocorreu um grande aumento no número de pessoas conectadas à Internet, em especial após a ascensão e popularização dos *smartphones*. Entre os diversos usos dos telefones móveis, destaca-se o uso de diferentes mídias sociais e aplicativos de mensagens.

Uma das redes sociais mais populares no mundo é o Twitter, que funciona como um *microblog*, no qual os usuários podem publicar mensagens curtas e interagir com outros usuários. A facilidade de espalhar mensagens propiciada pelo Twitter e seu grande alcance atraiu diversos desenvolvedores de *bots* sociais, que são algoritmos computacionais que simulam o comportamento humano na rede, produzindo conteúdo e interagindo com outros usuários. Estes *bots* são utilizados para divulgar notícias, propagandas ou mesmo para espalhar notícias falsas, as chamadas *fake news* (Ferrara et al., 2016).

No ano de 2018, durante as eleições presidenciais no Brasil, estima-se que uma grande quantidade de *bots* atuaram nos debates políticos no Twitter (Leu et al., 2019). Tais *bots* faziam desde propagandas políticas, realizando postagens em massa para dar mais visibilidade a *hashtags* ou assuntos específicos, até ataques a usuários que apoiavam outros candidatos, além de espalharem notícias falsas para desmoralizar determinados candidatos. Assim, a identificação desses perfis maliciosos é de suma importância para se permitir uma democracia digital.

A presente pesquisa visa a testar diferentes abordagens para a detecção de *bots* sociais, considerando características de perfis do Twitter que postaram sobre o segundo turno das eleições presidenciais no Brasil em 2018. Para tal identificação, foi empregada a abordagem de aprendizado supervisionado utilizando cinco classificadores diferentes.

O presente artigo está organizado da seguinte forma. A Seção 2 descreve alguns trabalhos correlatos, a Seção 3 apresenta os materiais e métodos utilizados para os experimentos. Já a Seção 4 apresenta os resultados obtidos. Por fim, a Seção 5 traz as considerações finais e possíveis trabalhos futuros a partir dos resultados obtidos.

2 Trabalhos Correlatos

Há diversos trabalhos na literatura sobre a identificação (ou detecção) de *bots* em diferentes contextos e utilizando diferentes abordagens, porém há uma escassez de trabalhos focando em *bots* que realizam postagens em português do Brasil.

O trabalho de Leu et al. (2019) tratou o mesmo contexto da pesquisa atual, porém focou apenas no uso de dois algoritmos explicáveis, a saber: regressão linear e árvore de decisão (utilizando o algoritmo *random tree*). Para a classificação foram utilizados atributos como idade da conta, amigos, curtidas, *tweets*, *retweets*, seguidores e *tweets* por dia. Ao longo do artigo foram citadas 13 características.

Nesse trabalho, foi utilizado um conjunto de treinamento composto de 642 instâncias rotuladas manualmente, assim como validação cruzada com dez subconjuntos *(10-fold cross-validation)*.

O trabalho de Karataş and Şahin (2017) revisa diversas técnicas de identificação de *bots* que utilizam três abordagens diferentes, sendo elas baseadas em topologia, *crowdsourcing* e aprendizado de máquina. Os autores consideram a abordagem baseada em aprendizado de máquina como a mais efetiva entre as três.

Davis et al. (2016) apresentam uma implementação de um algoritmo para a identificação de *bots* utilizando a abordagem de aprendizado de máquina. Intitulado de *BotOrNot*, utiliza *random forest* e mais de 1.000 características agrupadas em 6 grupos categóricos:

- Network: metadados sobre retweets e menções;
- User: metadados sobre linguagem utilizada, localização geográfica, quando a conta foi criada;
- Friends: metadados sobre média, mediana e outras medidas sobre o número de amigos, posts e seguidores;
- Temporal: metadados sobre padrões de tempo de geração e consumo de conteúdo, como geração, consumo e distribuição de conteúdo;
- Content: metadados sobre conteúdos de linguística analisados por processamento de linguagem natural;
- Sentimental: analisa as emoções do conteúdo de cada tweet.

Nesse trabalho, foram rotulados manualmente 15 mil *bots* e 16 mil contas de humanos e após execução do algoritmo utilizando *random forest* esse estudo obteve desempenho de 0,95 de área sob a curva ROC.

Daouadi et al. (2019) comparam classificadores de regressão logística, *bagging*, *adaboost*, *multilayer perceptron* e *random forest* para detecção de *bots* em redes sociais utilizando apenas características do perfil do usuário, destacando o desempenho das *random forest*. Além disso é utilizada a técnica *SMOTE* para reduzir o problema de desbalanceamento entre as classes.

Neste artigo foram utilizadas características como: se o usuário habilitou a marcação geográfica, se o usuário possui alguma URL atrelada ao perfil, se o usuário verificou o perfil, número de seguidores, número de perfis seguidos e número de favoritos, bem como características referentes as publicações realizadas pelo perfil como: número de favoritos, número médio de publicações realizadas e média de *hashtags* por publicação.

Alsaleh et al. (2014) construíram uma base de dados rotulando contas do Twitter entre humanos, *sybils* (*bot* malicioso) e contas híbridas por meio de e pela ajuda de voluntários. A partir dessa base trabalharam com os modelos de árvore de decisão C4.5, *random forest*, *sequential minimal optimization* (SMO) e *multilayer perceptron* para construir uma extensão de navegador de internet que classifica os perfis visitados no Twitter.

Ali Alhosseini et al. (2019) desenvolveram um modelo para detecção de *spam bots* usando *graph convolutional neural network* (GCNN), um modelo de rede neural que leva em consideração o grafo de conexões entre os perfis, seguidores e amigos.

No trabalho de Bessi and Ferrara (2016) é realizado um estudo sobre o impacto de *bots* nas discussões online relacionadas às eleições presidenciais de 2016 dos Estados Unidos. Os autores destacam que não existem grandes dificuldades para se obter um *bot* devido à grande variedade de fontes, como por exemplo tutoriais para usuários leigos, códigos fontes disponíveis e até empresas que oferecem *Bot-As-A-Service* (BaaS).

O trabalho de Balestrucci et al. (2019) aborda a identificação dos usuários ingênuos (pessoas que compartilha notícias falsas sem intenção). Uma possível vantagem de identificar tais perfis é a de encontrar mais *bots* em suas listas de amigos do que nas amizades dos demais usuários.

Para evitar que os modelos produzidos sejam super específicos ou sobreajustados (sofram do problema chamado de *overfitting*) a maioria dos trabalhos realiza validação cruzada ou a separação do conjunto de dados entre treinamento e teste (eventualmente realizando balanceamento no conjunto de treinamento, pois, tipicamente, os conjuntos tratados neste tipo de problema são desbalanceados). Por exemplo, no trabalho de Gilani et al. (2017), os autores dividiram o conjunto de dados em quatro partes, utilizando três para treinamento do classificador *random forest* e uma parte para os testes.

Como visto na literatura correlata, construir um conjunto de dados rotulado e identificar automaticamente perfis automatizados é um problema em aberto e de importância política e social, que tem sido abordado de diversas formas, com um destaque para as técnicas de aprendizado de máquina como *random forest*. Desta forma, o presente trabalho busca desenvolver uma comparação entre modelos explicáveis e não explicáveis de classificação, a fim de descobrir até que ponto os modelos não explicáveis se sobressaem.

3 Materiais e Métodos

Esta seção descreve os conjuntos de dados, modelagem do problema e estratégias utilizadas.

3.1 Conjuntos de dados

O conjunto de dados (*dataset*) utilizado foi organizado e disponibilizado por Leu et al. (2019) e é constituído pelos tweets postados durante o segundo turno das eleições presidenciais brasileiras de 2018, entre os dias 08 e 28 de Outubro de 2018 e que contém um conjunto de palavras chave específicas. Essa lista de palavras-chave contém termos relacionados às eleições presentes na lista dos *Trending Topics* do Twitter no Brasil, que é uma lista dos assuntos mais comentados, além do sobrenome dos candidatos ao segundo turno. Os tweets foram coletadas utilizando a API do Twitter por meio de um script em Python e armazenados em um banco de dados MongoDB. O desenvolvimento deste *script*, bem como a criação dessa base de dados, ocorreram fora do escopo deste artigo.

Ao todo, o conjunto de dados contém 8.083.140 tweets de um total de 1.165.498 usuários diferentes. Os usuários do conjunto de dados disponibilizados não possuíam um rótulo descrevendo se eram humanos ou *bots*.

Do conjunto original de dados foi extraída uma amostra com os 111.530 tweets de 800 perfis selecionados aleatoriamente, de modo a representar o conjunto citado anteriormente. Cada um dos perfis desta amostra foi anotado manualmente pelos autores do presente artigo como pertencentes a uma de duas classes: *bot* ou *humano*. Esse rótulo foi utilizado como classe durante os testes e validação das estratégias de classificação.

3.2 Rotulação dos dados

Os autores rotularam manualmente os dados, se dividindo em duas duplas. Cada dupla foi responsável por rotular na íntegra os 800 perfis selecionados, distinguindo entre *bots* ou *humanos*.

Para anotar os 800 perfis, cada dupla analisou perfil a perfil, utilizando informações que já existiam na base de dados e informações adicionais encontradas na própria plataforma do Twitter.

Provenientes da base de dados foram obtidas informações relacionadas aos *tweets* capturados, como os conteúdos, quantidades e intervalos entre as publicações, assim como informações relacionadas aos metadados do perfil, como a data de criação da conta, o número total de *tweets* do perfil, o *screen name* (nome de usuário), localidade do perfil e número de seguidores. Além disso, foram visitados os perfis na rede social Twitter para verificar os demais conteúdos publicados pelo usuário e então decidir se o perfil possuía um comportamento considerado de *bot* ou de humano.

Ainda que realizada de maneira subjetiva por cada dupla, o processo de rotulação foi efetuado considerando alguns parâmetros pré-estabelecidos que foram observados previamente. A Tabela 1 descreve as características que foram consideradas no processo de rotulação.

Tabela 1: Parâmetros observados durante a rotulação

Característica	Breve descrição
Diversidade de conteúdo	Variabilidade do conteúdo publicado
Mídias publicadas	Foto de perfil, foto de capa, fotos publicadas e vídeos
Complexidade de linguagem	Como o perfil escreve o conteúdo publicado
Coerência	Conteúdo publicado é adequado ao contexto e consegue manter diálogos
Outros	Outros atributos como curtidas do perfil e horário das publicações

Durante o procedimento de rotulação foram encontrados perfis ativos na data da extração dos dados mas que se encontravam inativos durante o processo de rotulação. Outra situação encontrada foi perfis bloqueados pelo Twitter e ainda contas que estavam públicas na data de extração do conjunto de dados, mas se encontravam privadas no período de rotulação, impedindo a consulta do perfil diretamente no Twitter. Para esses casos a rotulação contou exclusivamente com as informações contidas na base de dados.

Após a realização das anotações manuais por cada uma das duplas, foi realizado um procedimento de junção das duas rotulações. Essa junção foi tratada em duas etapas: para os casos rotulados em que coincidiram ambas rotulações, o rótulo foi adotado. Para os casos divergentes, considerados *humano* por uma das duplas e *bot* por ou-

tra, foi realizado um júri com todos os autores a fim de encontrar o consenso no rótulo.

Seguindo esse procedimento, todos os 800 perfis inicialmente selecionados foram rotulados, dos quais 707 foram anotados como *humanos*, o que corresponde a 88,37% do total e 93 foram anotados como *bot*, o que corresponde a 11,63% do total.

3.3 Conjunto de características

Tabela 2: Características iniciais do conjunto de dados

Característica	Breve descrição
acc_age_days	idade da conta
default_pic	se o perfil usa a foto padrão oferecida pelo twitter ou não
default_prof	se o perfil padrão foi alterado
friends	quantidade de outros perfis que o perfil segue
likes	quantidade de curtidas
tweets	número de tweets ou retweets publicados
followers	seguidores
verified	se tem conta verificada
geo_enabled	se a localização está ativa
tweets_tag	número de tweets dentro dos termos definidos
retweets_tag	número de retweets dentro dos termos definidos
tag_tweets_freq	razão tweets por retweets
tweets_per_day	razão tweets por acc age days
likes_friends_ratio	razão entre curtidas e quantidade de perfis seguindo
likes_followers_ratio	razão entre curtidas e número de seguidores e seguindo
followers_friends_ratio	razão entre seguidores e seguindo

Para realizar as classificações, apenas atributos numéricos e booleanos foram extraídos dos dados iniciais.

O conjunto de dados cedido inicialmente já contava com um conjunto de 16 características, conforme apresentado na Tabela 2.

Além das características citadas, foram produzidas 19 novas características a partir do banco de dados original, combinando características já existentes e extraindo estatísticas dos tweets do perfil presente no banco de dados. Uma dificuldade encontrada ao produzir essas características foi que o banco de dados contém apenas os tweets com as palavras-chaves selecionadas, impossibilitando a produção de novas características que considerem o comportamento do usuário fora do contexto das eleições. As características produzidas são descritas na Tabela 3.

Tabela 3: Características produzidas

Característica	Breve descrição
number_screenname_ratio	razão entre caracteres e números no *username*
screen_names_count	quantidade de *screen names* diferentes usados
screen_names_avg_len	média do número de caracteres existentes no *screen name*
mean_size_tweets	tamanho médio dos tweets publicados
active_days_tag	diferença entre a data da primeira publicação e da última
tweets_tag_per_active_interval	média de tweets diários no intervalo de dias em que o usuário esteve ativo
max_tweets_3min	máximo de tweets publicados em intervalo de 3 minutos
max_tweets_10min	máximo de tweets publicados em intervalo de 10 minutos
max_tweets_1h	máximo de tweets publicados em intervalo de 1h
max_tweets_3h	máximo de tweets publicados em intervalo de 3h
in_reply_count_tag	total de tweets do tipo resposta
truncated_count_tag	total de tweets com o texto truncado
tweets_per_day_diff	diferença entre a média de publicações diárias e a média das mesmas considerando o intervalo de dias que o usuário esteve ativo
tweets_tag_per_active_day	quantidade de tweets pela quantidade de dias que o perfil se manteve ativo dentro do período coletado
tweets_tag_per_active_day_std_deviation	desvio padrão da característica descrita acima
total_medias_tag	total de publicações que possuem mídias
total_mentions_tag	total de publicações que possuem menções
total_hashtags_tag	total de publicações que possuem *hashtags*
total_urls_tag	total de publicações que possuem links

3.4 Estratégias de Classificação

Por se tratar de um problema de classificação, foram escolhidos algoritmos supervisionados a fim de analisar a influência de cada variável nos modelos. Para escolher os classificadores foram testados vários algoritmos diferentes, utilizando o apoio do arcabouço Waikato Environment for Knowledge Analysis - WEKA (Holmes et al., 1994) e também a partir da biblioteca *scikit-learn* em Python.

Considerando que na literatura correlata, os modelos utilizando *multilayer perceptron* (MLP) e *random forest*

apresentam bom desempenho para a classificação de *bots* por características de usuário, e visando a expandir o trabalho de Leu et al. (2019), analisando o ganho ao utilizar classificadores mais complexos, foram utilizados os seguintes classificadores: regressão linear, árvores de decisão com o algoritmo *random tree*, *naive* Bayes, *multilayer perceptron* e *random forest*.

Também foi necessário adotar estratégias de seleção de atributos particulares para cada um dos classificadores. O conjunto de dados final possui 34 características, o que resultaria em 2^{34} subconjuntos diferentes de atributos, o que torna muito custoso testar todas as combinações possíveis. Utilizou-se a técnica *Correlation Feature Selection* (CFS), proposta por Hall (2000), para a seleção de subconjuntos de características. O algoritmo usa uma heurística baseada na premissa que "bons subconjuntos possuem características altamente correlacionadas com a classe, e descorrelacionadas uma com a outra" (Hall, 2000). Variações desse algoritmo foram utilizadas para cada classificador.

Para lidar com o problema de desbalanceamento entre as classes, foi testado o uso de *Synthetic Minority Oversampling Technique (SMOTE)* (Chawla et al., 2002), que visa a gerar observações sintéticas da classe minoritária (*bot*) no conjunto de treinamento, aumentando a taxa de verdadeiros positivos preditos pelo modelo, ao custo de um número maior de falsos positivos.

3.4.1 Regressão Linear

A regressão linear é uma equação utilizada para estimar o valor de uma variável, dado os valores de um conjunto de outras variáveis. Para a produção de um modelo baseado em regressão linear, é necessária a utilização de um rótulo numérico, portanto, convertemos a classe *bot* para 1.000 e a classe *humano* para −1.000.

Para realizar a seleção de atributos utilizados no modelo, foi utilizado o seletor *CorrelationAttributeEval*, que avalia o atributo medindo a sua correlação com a classe. Após a aplicação do seletor de atributos, foram efetuados alguns testes variando o conjunto, com o objetivo de maximizar o coeficiente de correlação do modelo.

Para a execução do modelo e seleção de atributos, foram utilizadas as implementações disponíveis no arcabouço WEKA.

Após a construção do modelo, a ideia é utilizar a função para atribuir uma pontuação para cada um dos usuários, que indicará uma maior ou menor chance dele ser um *bot*.

3.4.2 Árvore de Decisão

A Árvore de Decisão utilizada foi a *random tree*. O algoritmo constrói a árvore de decisão com base nos atributos que dão maior ganho de informação por nível. O atributo com maior ganho de informação é escolhido para tomar a decisão.

Para identificar os atributos mais relevantes para distinguir um *bot* de um *humano* foram utilizadas técnicas de seleção de atributos. Além do CFS, foi utilizado o seletor *InfoGainAttributeEval* o qual avalia o valor de um atributo medindo o ganho de informações que esse atributo trás em relação a classe. Após aplicação do seletor de atributos foram testadas variações do conjunto de atributos baseando-se na remoção do atributos que, segundo o seletor, davam ganho 0 ao modelo. O conjunto de atributos final é o conjunto que demonstrou o melhor desempenho nos testes.

Para a execução deste algoritmo, bem como a seleção dos atributos foram utilizadas as implementações disponíveis no WEKA.

Para o treinamento, foi utilizado um filtro de instâncias que discretiza o intervalo de atributos contínuos do conjunto de dados. A discretização foi realizada de acordo com o proposto por Fayyad and Irani (1993).

Foram realizadas simulações tanto com o conjunto de treinamento desbalanceado, proporções diferentes entre *bot* e humanos, quanto com o conjunto de treinamento balanceado, utilizando a técnica de *SMOTE* para replicar os perfis da classe *bot* e deixar as duas classes em semelhantes proporções.

3.4.3 Naive *Bayes*

O *naive* Bayes é um algoritmo que infere a probabilidade de um novo registro pertencer a uma determinada classe. Para isso, é assumido que, dada a classe, as variáveis são independentes entre si. Entretanto, frequentemente esta premissa não condiz com a realidade (Chen et al., 2020).

A implementação do algoritmo utilizada foi o *naive* Bayes do WEKA com uma função de estimação normal. Nesta implementação as probabilidades para as variáveis binárias são estimadas dividindo a frequência pelo total de ocorrências enquanto que para as variáveis contínuas a média e o desvio padrão são estimados para a construção de uma função densidade de probabilidade (FDP).

Para a seleção dos atributos foi utilizada uma implementação em Python do algoritmo *Correlation Feature Selection* com o algoritmo de busca *HillClimb* com uma taxa de adição de 75% (probabilidade de um atributo ser adicionado ao subconjunto) e um critério de parada de 50 vezes consecutivas sem expansão. O algoritmo foi executado algumas vezes e dentre os conjuntos gerados foi escolhido aquele que obteve maior precisão.

3.4.4 Multilayer Perceptron

O *multilayer perceptron* (MLP) é uma rede neural do tipo composta por componentes individuais simples, *perceptrons*, divididos em uma camada de entrada, uma ou mais camadas escondidas que fazem o processamento de sinal com funções de ativação não lineares, e a camada de saída. Além disso conta com um mecanismo de retropropagação de erro responsável pelo treinamento do modelo.

As características utilizadas para treinar o MLP foram selecionadas utilizando *Correlation Feature Selection* com *Backward Elimination* (eliminação para trás). A rede neural foi construída utilizando uma única camada escondida com 100 neurônios, função de ativação *rectified linear unit* (ReLU), taxa de aprendizado constante 0,01 e método de otimização de Gradiente Descendente Estocástico com otimizador Adam (Kingma and Ba, 2014).

3.4.5 Random Forest

Florestas aleatórias, ou *random forests* Breiman (2001) são modelos baseados em uma combinação de classificadores de árvores em que cada árvore é gerada com base em um valor aleatório produzido independente e com mesma dis-

tribuição para todas árvores. O valor predito pela floresta é definido pela moda dos valores preditos individualmente por cada árvore. No caso de classificadores binários é possível atribuir um valor numérico referente à frequência da classe positiva e trabalhar variando o limiar de classificação.

No modelo utilizado neste trabalho foram utilizadas 100 árvores por floresta. Além disso, para que cada árvore gerada seja diferente, o algoritmo utiliza uma estratégia de *bagging*, em que é gerado um novo conjunto de dados de treino para cada árvore, selecionando observações aleatórias com reposição do conjunto de dados original. Neste trabalho, estes novos conjuntos de dados foram gerados com tamanho igual ao conjunto original, e a diferença entre eles ocorre pela possibilidade de observações repetidas.

O modelo treinado utilizou a implementação disponível no WEKA. Além disso, assim como para as árvores de decisão, os atributos foram previamente discretizados.

3.5 Estratégia de avaliação

Todos os modelos foram avaliados usando validação cruzada estratificada com dez subconjuntos (10-*fold cross-validation*). Na validação cruzada (Browne, 2000) o conjunto de dados é dividido em k subconjuntos e são construídos k modelos iterativamente. A cada iteração, o modelo é treinado com $k-1$ subconjuntos e validado com o subconjunto restante, evitando *overfitting*. Ao final do processo, os modelos terão sido validados no conjunto completo de dados. A variação estratificada da validação cruzada garante que cada um dos subconjuntos possui proporções entre as classes equivalentes ao do conjunto total e é especialmente interessante para problemas de classificação com classes desbalanceadas, como no presente trabalho.

Para o problema de classificação de *bots*, a acurácia, definida como $\frac{TP+TN}{TT}$, não é uma boa medida de qualidade do modelo, já que a presença da classe *humano* se sobressai muito sobre a classe *bot*. Para o banco de dados deste trabalho, um classificador que classifique todas as observações como *humano* acertaria em 88,37% dos casos, tendo assim uma acurácia de 0,8837. Desta forma, outras medidas também foram utilizadas para avaliação, sendo elas: matriz de confusão, precisão, sensibilidade (ou revocação) e medida-F (*F-score*). Essas medidas são definidas a seguir:

$$\textit{Matriz de confusão} = \frac{TP|FN}{FP|TN}$$

$$\textit{Precisão} = \frac{TP}{TP+FP}$$

$$\textit{Sensibilidade} \text{ ou } \textit{Revocao} = \frac{TP}{TP+FN}$$

$$\textit{medida F} = \frac{2 * \textit{precisão} * \textit{sensibilidade}}{\textit{precisão} + \textit{sensibilidade}}$$

Sendo TP o número de verdadeiros positivos, ou seja, a contagem de predições positivas que o classificador acertou (*true positive*), FN refere-se a contagem de predições negativas que o classificador errou (*false negative*), FP representa a contagem de predições positivas que o classificador errou (*false positive*), TN é a contagem de predições negativas que o classificador acertou (*true negative*) e TT o total de casos da massa rotulada.

Neste trabalho a classe *bot* foi considerada a classe positiva, e a classe *humano* foi considerada a classe negativa.

Além dessas medidas, a análise da curva ROC (Fawcett, 2006), em especial a área sob a curva (*AUC*), também foi utilizada para avaliar o desempenho dos classificadores.

4 Resultados e Discussão

Nesse trabalho foram propostos modelos de classificadores utilizando regressão linear, *naive Bayes*, árvores de decisão e MLP para solucionar o problema de classificação de *bots* e *humanos* utilizando dados de perfis da rede social Twitter durante o período do segundo turno das eleições presidenciais de 2018 no Brasil. Nesta seção são apresentados e discutidos os resultados obtidos.

Inicialmente são apresentadas os resultados de cada modelo individualmente. Por fim, os resultados são discutidos

4.1 Regressão Linear

Os seguintes atributos foram selecionados de forma automática pelo modelo de regressão linear:

- geo_enabled
- number_screenname_ratio
- tweets_tag_per_active_interval
- max_tweets_3h
- total_medias_tag
- total_urls_tag

A correlação entre a classe e o resultado da regressão linear foi de 0,401 para o conjunto de treinamento, indicando uma correlação fraca.

A Eq. (1) apresenta a função de pontuação produzida pelo modelo de regressão linear. Observa-se que foram considerados cinco atributos numéricos e um atributo binário.

Os atributos com maiores pesos da regressão foram *geo_enabled = False*, significando que, se o usuário não ativou a geolocalização do perfil, sua pontuação será maior e *number_screenname_ratio*, significando que quanto maior a razão entre caracteres e números no nome de usuário, maior a pontuação do perfil. Os outros dois atributos numéricos com pesos positivos foram *tweets_tag_per_active_interval*, significando que quanto maior a média de tweets diários no intervalo de dias em que o usuário esteve ativo, maior a pontuação e *max_tweets_3h*, significando que, quanto maior o número máximo de tweets publicados em um intervalo de 3h, maior a pontuação do perfil. Por fim, os outros dois atributos numéricos possuem peso negativo, *total_medias_tag* e *total_urls_tag*, ou seja, quanto maior a quantidade de publicações que possuem mídias e *urls*, menor a pontuação do perfil.

$$
\begin{aligned}
Pontuacao = \\
119.0197 * geo\_enabled = False + \\
664.6394 * number\_screenname\_ratio + \\
14.1461 * tweets\_tag\_per\_active\_interval + \\
4.3426 * max\_tweets\_3h + \\
-5.3868 * total\_medias\_tag + \\
-1.1277 * total\_urls\_tag + \\
-1038.3402
\end{aligned} \tag{1}
$$

A pontuação obtida através da equação pode ser utilizada para a classificação dos perfis, partindo de um limiar determinado previamente. Considerando a maximização de usuários corretamente classificados para o conjunto de treinamento, definiu-se o limiar de 103,182, com uma taxa geral de acerto de 87,88%.

A Tabela 14 apresenta os resultados da classe *bot* obtidos a partir do limiar definido. A matriz de confusão correspondente é apresentada na Tabela 4.

Apesar da alta acurácia, é possível observar que os resultados obtidos utilizando esse modelo não foram satisfatórios. A acurácia de 87,88% ocorre devido ao conjunto de dados estar desbalanceado, conforme mencionado anteriormente. A área sob a curva ROC obtida por este modelo foi de 0,780.

Tabela 4: Regressão Linear: Matriz de confusão

		Predito	
		bot	humano
Real	bot	11	82
	humano	15	692

4.2 Árvore de Decisão

A seleção de atributos resultou em um conjunto de 29 características, sendo descartados os seguintes atributos:

- default_pic
- verified
- screen_names_count
- mean_size_tweets
- max_tweets_1h
- max_tweets_3h

4.2.1 *Treinamento desbalanceado*

A classificação realizada com os dados de treinamento desbalanceados foi realizada após a seleção dos atributos. O modelo classificou corretamente 96% dos humanos e 28% dos *bots*. A matriz de confusão apresentada na Tabela 5 mostra os resultados obtidos.

É possível notar a medida F, precisão, revocação e área sob a curva ROC expostos na Tabela 14. É válido destacar que as três primeiras medidas citadas obtiveram resultados médios superiores a 0,8, já a área sob a curva ROC obtida por este modelo foi de 0,676.

Tabela 5: Árvore: Matriz de confusão

		Predito	
		bot	humano
Real	bot	26	67
	humano	30	677

Foram realizadas simulações com variação da altura máxima da árvore de 1 a 5, e sem limite de altura máxima. Destaca-se que o modelo com melhor desempenho foi o modelo sem limitação de altura com os resultados destacados nas Tabelas 5 e 14.

A Fig. 1 apresenta a árvore de altura 1. A primeira característica observada foi o máximo de tweets publicados pelo perfil em um intervalo de 3 minutos, porém, apenas com esse atributo não é possível determinar se um perfil é um *bot* ou um humano.

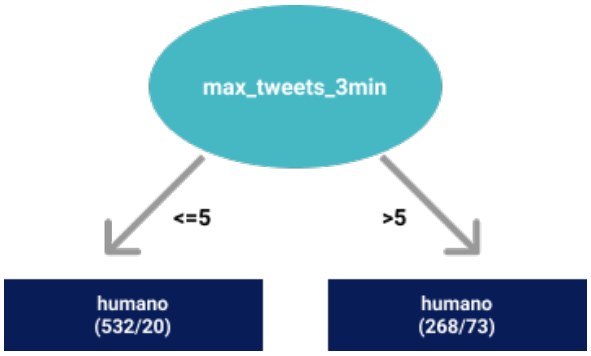

Figura 1: Árvore: Altura 1

Já na Fig. 2 com a árvore de altura 2 é possível observar como a decisão de ser um *bot* ou humano foi construída pelo algoritmo. Caso o perfil tenha 5 ou mais tweets publicados neste intervalo, ele segue para o lado esquerdo da árvore de decisão, no qual será analisada a razão entre caracteres e números presentes no *screenname* do perfil, caso essa razão seja maior do que 0,516 o perfil é classificado como *bot*.

Na Tabela 6 encontram-se as matrizes de confusão referentes ao desempenho das árvores de alturas de 1 a 5.

4.2.2 *Treinamento balanceado*

O conjunto de dados não contém quantidades equivalentes de *bots* e *humanos*. Para simular esse comportamento foi utilizada uma técnica de super montagem por minoria sintética (*SMOTE*) utilizando o mesmo conjunto de dados e a mesma técnica de discretização já apresentados no início da seção. A Tabela 7 apresenta a matriz de confusão.

Enquanto que sem o uso do *SMOTE* foi possível detectar corretamente 27,956% dos *bots*, com o uso de *SMOTE* foram identificados corretamente 40,860% dos perfis de

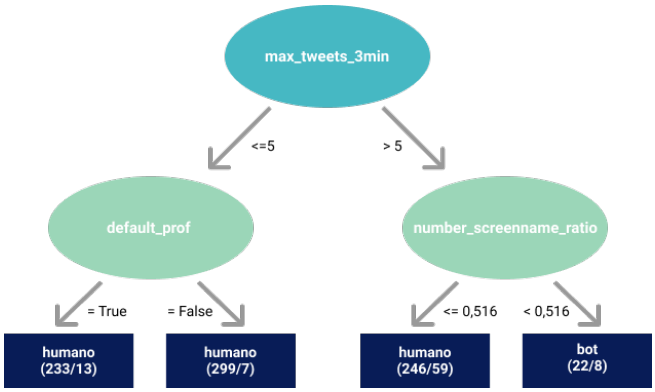

Figura 2: Árvore: Altura 2

Tabela 6: Árvore: Resultados para alturas de 1 a 5

		Predito	
	Altura	bot	humano
Real bot	1	0	93
Real humano	1	0	707
Real bot	2	4	89
Real humano	2	3	704
Real bot	3	8	85
Real humano	3	5	702
Real bot	4	8	85
Real humano	4	7	700
Real bot	5	11	82
Real humano	5	11	696

Tabela 7: Árvore com SMOTE: Matriz de confusão

		Predito	
		bot	humano
Real	bot	38	55
Real	humano	58	649

bots. Já para a classe humanos houve uma diminuição na detecção com uso de SMOTE de 3,960%.

Além de testar o modelo com altura máxima ilimitada, conforme resultados apresentados nas Tabelas 7 e 15, o modelo também foi testado com a altura da árvore de 1 a 5.

Conforme apresentado na Fig. 3, com altura 1 não é possível distinguir se o perfil pertence a classe *bot* ou humano, assim como ocorreu sem o uso de *SMOTE*.

Já com altura 2 a árvore consegue classificar perfis humanos e perfis *bots*. A Fig. 4 representa a árvore obtida. É possível observar que o algoritmo classifica como *bot* 141 perfis, sendo destes 64 falsos positivos. Sem uso de

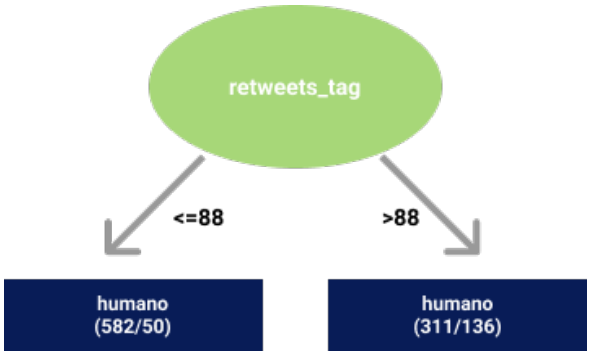

Figura 3: Árvore com SMOTE - altura 1

SMOTE com árvore da mesma altura foram classificados como *bot* 22 dos quais 8 eram falsos positivos.

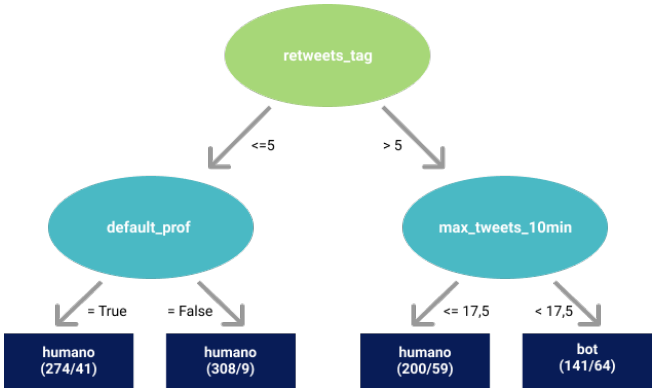

Figura 4: Árvore com SMOTE: altura 2

As matrizes de confusão das árvores de altura 1 a 5 estão explicitadas na Tabela 8.

4.3 *Naive* Bayes

O conjunto de características selecionadas foi:

- geo_enabled
- number_screenname_ratio
- tweets_tag_per_active_interval
- max_tweets_3h
- tweets_per_day_diff

As variáveis *tweets_tag_per_active_interval* e *max_tweets_3h*, que são métricas sobre a frequência de tweets durante o período de coleta de dados, apresentam uma forte correlação de 0,87%. Isso quer dizer que há duas variáveis sobre frequências de tweets, mas isso não é necessariamente algo negativo, pois aumentar o peso sobre a frequência pode aumentar o desempenho do algoritmo.

4.3.1 *Treinamento desbalanceado*

Para este modelo, a matriz de confusão e a área sob a curva ROC e os resultados por classe estão listados nas Tabelas 9

Tabela 8: Árvore com SMOTE: Resultados para alturas de 1 a 5

			Predito	
	Altura		bot	humano
Real	1	bot	3	90
		humano	7	700
	2	bot	10	83
		humano	29	678
	3	bot	16	77
		humano	25	682
	4	bot	20	73
		humano	39	668
	5	bot	25	68
		humano	55	652

e 14, respectivamente. Comparando aos demais modelos, os resultados são bons, uma vez que a precisão para a classe *bot* foi de 0,413 e a revocação 0,462, ou seja, o modelo detecta quase 50% dos *bots* e entre os que ele aponta como *bot*, cerca de 41% são de fato.

Para as variáveis contínuas, a média e o desvio padrão estimados para a classe *bot* foram maiores do que para a classe humano. Isso implica que usuários com valores maiores para estas variáveis possuem maiores chances de serem *bots*. Também é possível concluir que, para o modelo, humanos possuem um comportamento menos frequente e mais uniforme.

Tabela 9: Naive Bayes sem SMOTE: Matriz de confusão

		Predito	
		bot	humano
Real	bot	43	50
	humano	61	646

4.3.2 *Treinamento balanceado*

Para este modelo a área sob a curva ROC foi 0,810, praticamente igual ao anterior. Entretanto ao se analisar a matriz de confusão descrita na Tabela 10 e os resultados mostrados na Tabela 15, é possível ver que a revocação de *bots* aumentou sob o custo de aumentar o número de falsos positivos.

As variáveis contínuas apresentaram um leve aumento na média e redução da variância, mantendo as FDPs praticamente iguais. Entretanto, houve uma alteração mais significativa para a variável *booleana geo_enabled*. Treinando com o conjunto desbalanceado, para a classe *bot*, as probabilidades eram 0,768 para verdadeiro e 0,232 para falso. Com o conjunto balanceado os valores passaram a ser 0,878 para verdadeiro e 0,122 para falso.

Tabela 10: Naive Bayes com SMOTE: Matriz de confusão

		Predito	
		bot	humano
Real	bot	45	48
	humano	76	631

4.4 Multilayer Perceptron

O conjunto de características utilizado, resultante do processo de *Correlation Feature Selection* foi:

- acc_age_days
- tweets_per_day
- geo_enabled
- tweets_tag
- number_screenname_ratio
- tweets_tag_per_active_interval
- max_tweets_3min
- max_tweets_10min
- max_tweets_1h
- in_reply_count_tag
- tweets_per_day_diff
- total_mentions_tag

Os resultados da execução do MLP no conjunto de dados desbalanceado são apresentados nas Tabelas 11 e 14. O classificador obteve uma área sob a curva ROC de 0,65. Ainda que significativos, o MLP não se sobressaiu sobre outros modelos nos resultados, apresentando baixa precisão e revocação para a classe *bot*.

Tabela 11: MLP: matriz de confusão

		Predito	
		bot	humano
Real	bot	30	63
	humano	70	637

Aplicando a técnica *SMOTE* para balancear o treinamento, foi possível aumentar significativamente a taxa de revocação de *bots*, ao custo de um número excessivamente alto de falsos positivos. A área sob a curva obtida foi de 0,76 e demais resultados podem ser vistos nas Tabelas 12 e 15. Foi possível detectar 70% dos *bots*, no entanto com uma precisão de apenas 0,23, o que torna uma segunda análise necessária para poder rotular efetivamente um perfil como *bot*.

4.5 Random Forest

Para treinamento do modelo de *random forest* foi utilizado o mesmo conjunto de características que para o MLP.

Tabela 12: MLP com SMOTE: matriz de confusão

		Predito	
		bot	humano
Real	bot	65	28
	humano	218	489

Os resultados da execução do modelo *random forest* (RF) são apresentados nas Tabelas 13 e 14. O classificador obteve uma área sob a curva ROC de 0,804. O modelo se mostrou bem mais preciso que outros modelos, como o MLP, com uma precisão de 0,511 para a classe *bot*. Além disso, ainda que a revocação para *bots* seja baixa, o modelo apresenta uma área sob a curva ROC mais alta que outros modelos, e a revocação pode ser aumentada mudando o limiar do modelo. Aplicar uma técnica de *oversampling* como o *SMOTE* não trouxe melhorias ao modelo em termos de acurácia.

Tabela 13: RF: matriz de confusão

		Predito	
		bot	humano
Real	bot	23	70
	humano	22	685

4.6 Resultados gerais

Os resultados apresentados anteriormente foram sumarizados e estão organizados nas Tabelas 14 e 15.

5 Conclusões e Trabalhos Futuros

O presente trabalho propôs e apresentou um estudo de caso para classificação de perfis de usuários da rede social Twitter usando dados extraídos durante o segundo turno das eleições presidenciais de 2018 no Brasil. Foram gerados diversos modelos fazendo uso de cinco algoritmos diferentes. Cada modelo utilizou um conjunto distinto de variáveis, entretanto a maioria dos modelos utilizou variáveis relacionadas à frequência de postagem dos usuários.

O modelo de *random forest* foi capaz de detectar a classe *bot* com uma precisão maior que os demais modelos, mas com uma revocação baixa. Em comparação, o algoritmo *Naive Bayes* conseguiu uma revocação maior, mas obteve uma precisão menor. Em geral, mesmo um algoritmo avançado como o *random forest* não foi significantemente superior ao modelo de árvore aleatória, sendo apenas 5% mais preciso. Nesse caso, é vantajoso o uso de árvore de decisão, pois, por ser um algoritmo explicável, é possível visualizar facilmente quais foram as decisões tomadas para a classificação.

Os modelo gerados utilizando o algoritmo MLP foram os que conseguiram detectar o maior número de *bots*, chegando a 70% de revocação quando combinado com a técnica *SMOTE*. No entanto, a precisão nessa detecção é muito baixa, gerando um número muito grande de falsos positivos. Já a regressão linear apesar de obter uma área sob a curva de 0,78, apresentou revocação e medida F inferiores aos demais algoritmos utilizados.

Conforme esperado, o uso da da técnica SMOTE para o balanceamento do conjunto de treinamento resultou no aumento da revocação da classe minoritária (isto é, dos *bots*). A precisão na classificação tendeu a ser reduzida pois, apesar do aumento dos verdadeiros positivos (*bots* classificados corretamente) houve um aumento ainda maior dos falsos positivos (humanos classificados como *bots*). As variações nas métricas área sobre a curva ROC e medida F foram, em geral, neutras, pois o treinamento com um conjunto balanceado produziu um maior equilíbrio entre precisão e revocação, porém com uma redução da precisão. Conforme esperado, a acurácia geral da solução foi reduzida.

Considera-se que os resultados com o uso da técnica SMOTE foram satisfatórios, destacando que o uso ou não do balanceamento deve considerar o objetivo principal da aplicação de detecção de *bots*: se o objetivo é maximizar a acurácia, provavelmente o balanceamento não deverá ser considerado. Por outro lado, se o objetivo é maximizar a revocação (por exemplo, para uma etapa posterior de confirmação se um usuário classificado como *bot* é ou não um *bot*), o uso de técnicas de balanceamento deve ser considerado.

Como trabalhos futuros, pretende-se explorar as demais classes de características, como as apresentadas por Davis et al. (2016): baseadas nos amigos, baseadas na rede, temporais, conteúdo das postagens e sentimentos, uma vez que o presente trabalho teve foco principalmente nas características do usuário. Para isso, pretende-se trabalhar com novos conjuntos de dados com contextos políticos atuais, como a propagação de notícias falsas (*fake news*), contando com o conjunto completo de publicações do usuário em determinado período, além de informações sobre a topologia da rede.

Outra abordagem interessante utilizando o mesmo conjunto de dados é fazer uso de processamento de linguagem natural (PLN) para analisar, além das características gerais do perfil e dos metadados, os conteúdos dos textos publicados e com isso enriquecer os modelos apresentados neste artigo.

Referências

Ali Alhosseini, S., Bin Tareaf, R., Najafi, P. and Meinel, C. (2019). Detect me if you can: Spam bot detection using inductive representation learning, *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, Association for Computing Machinery, New York, NY, USA, p. 148–153. https://doi.org/10.1145/3308560.3316504.

Alsaleh, M., Alarifi, A., Al-Salman, A. M., Alfayez, M. and Almuhaysin, A. (2014). Tsd: Detecting sybil accounts in twitter, *2014 13th International Conference on Machine Learning and Applications*, pp. 463–469. https://doi.org/10.1109/ICMLA.2014.81.

Balestrucci, A., De Nicola, R., Inverso, O. and Trubiani, C. (2019). Identification of credulous users on twitter,

Tabela 14: Resultados por classe - sem SMOTE

Algoritmo	Classe	Precisão	Revocação	Medida F	ROC
Regressão Linear	bot	0,423	0,118	0,184	0,780
Árvore Aleatória	humano	0,910	0,958	0,933	0,676
	bot	0,464	0,280	0,349	0,676
	média ponderada	0,858	0,879	0,865	0,676
Naive Bayes	humano	0,928	0,914	0,921	0,811
	bot	0,413	0,462	0,437	0,811
	média ponderada	0,868	0,861	0,865	0,811
Multilayer Perceptron	humano	0,910	0,901	0,905	0,650
	bot	0,300	0,323	0,311	0,650
	média ponderada	0,839	0,834	0,836	0,650
Random Forest	humano	0,907	0,969	0,937	0,804
	bot	0,511	0,247	0,333	0,804
	média ponderada	0,861	0,885	0,867	0,804

Tabela 15: Resultados por classe - com SMOTE

Algoritmo	Classe	Precisão	Revocação	Medida F	ROC
Árvore Aleatória	humano	0,922	0,918	0,920	0,692
	bot	0,396	0,409	0,402	0,692
	média ponderada	0,861	0,859	0,860	0,692
Naive Bayes	humano	0,929	0,893	0,911	0,810
	bot	0,372	0,484	0,421	0,810
	média ponderada	0,865	0,845	0,854	0,810
Multilayer Perceptron	humano	0,946	0,692	0,799	0,760
	bot	0,230	0,699	0,346	0,760
	média ponderada	0,863	0,693	0,746	0,760

Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC '19, Association for Computing Machinery, New York, NY, USA, p. 2096–2103. https://doi.org/10.1145/3297280.3297486.

Bessi, A. and Ferrara, E. (2016). Social bots distort the 2016 u.s. presidential election online discussion, *First Monday* **21**(11). https://doi.org/10.5210/fm.v21i11.7090.

Breiman, L. (2001). Random forests, *Machine Learning* **45**(1): 5–32. https://doi.org/10.1023/A:1010933404324.

Browne, M. W. (2000). Cross-validation methods, *Journal of Mathematical Psychology* **44**(1): 108 – 132. https://doi.org/10.1006/jmps.1999.1279.

Chawla, N. V., Bowyer, K. W., Hall, L. O. and Kegelmeyer, W. P. (2002). Smote: Synthetic minority oversampling technique, *Journal of Artificial Intelligence Research* **16**: 321–357. http://dx.doi.org/10.1613/jair.953.

Chen, S., Webb, G. I., Liu, L. and Ma, X. (2020). A novel selective naïve bayes algorithm, *Knowledge-Based Systems* **192**: 105361. https://doi.org/10.1016/j.knosys.2019.105361.

Daouadi, K., Rebaï, R. and Amous, I. (2019). *Bot Detection on Online Social Networks Using Deep Forest*, Springer, pp. 307–315. https://doi.org/10.1007/978-3-030-19810-7_30.

Davis, C. A., Varol, O., Ferrara, E., Flammini, A. and Menczer, F. (2016). Botornot: A system to evaluate social bots, *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW '16 Companion, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, p. 273–274. https://doi.org/10.1145/2872518.2889302.

Fawcett, T. (2006). An introduction to roc analysis, *Pattern Recognition Letters* **27**(8): 861 – 874. https://doi.org/10.1016/j.patrec.2005.10.010.

Fayyad, U. M. and Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning, *IJCAI*. https://doi.org/10.1007/978-3-642-40897-7_11.

Ferrara, E., Varol, O., Davis, C., Menczer, F. and Flammini, A. (2016). The rise of social bots, *Commun. ACM* **59**(7): 96–104. https://doi.org/10.1145/2818717.

Gilani, Z., Kochmar, E. and Crowcroft, J. (2017). Classification of twitter accounts into automated agents and human users, *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, ASONAM '17, Association for Computing Machinery, New York, NY, USA, p. 489–496. https://doi.org/10.1145/3110025.3110091.

Hall, M. (2000). Correlation-based feature selection for discrete and numeric class machine learning, *Proceedings of the 17th international conference on machine learning (ICML-2000)*, pp. 359–366.

Holmes, G., Donkin, A. and Witten, I. H. (1994). Weka: A machine learning workbench. http://dx.doi.org/10.1109/ANZIIS.1994.396988.

Karataş, A. and Şahin, S. (2017). A review on social bot detection techniques and research directions, *ISCTurkey 10th International Information Security and Cryptology Conference*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. Disponível em https://arxiv.org/abs/1412.6980.

Leu, M. D. 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, *Revista de Sistemas de Informação da FSMA*, p. 31–39. Disponível em http://www.fsma.edu.br/si/edicao24/Download_FSMA_SI_2019_2_Principal_5.html.