UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

DIEGO BEZERRA LIRA

Detecção de bots no Twitter: Uma abordagem utilizando agrupamento

São Paulo
2022

DIEGO BEZERRA LIRA

Detecção de bots no Twitter: Uma abordagem utilizando agrupamento

Versão corrigida

Dissertação apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação.

Área de concentração: Metodologia e Técnicas da Computação

Orientador: Prof. Dr. Luciano Antonio Digiampietri

São Paulo

2022

Ficha catalográfica elaborada pela Biblioteca da Escola de Artes, Ciências e Humanidades,
com os dados inseridos pelo(a) autor(a)
Brenda Fontes Malheiros de Castro CRB 8-7012; Sandra Tokarevicz CRB 8-4936

Bezerra Lira, Diego
Detecção de bots no Twitter: Uma abordagem utilizando agrupamento / Diego Bezerra Lira; orientador, Luciano Antonio Digiampietri. -- São Paulo, 2022.
66 p: il.

Dissertacao (Mestrado em Ciencias) - Programa de Pós-Graduação em Sistemas de Informação, Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, 2022.
Versão corrigida

1. Twitter. 2. Detecção de bots. 3. Agrupamento. I. Digiampietri, Luciano Antonio, orient. II. Título.

Agradecimentos

Ao meu orientador, Prof. Dr. Luciano Digiampietri, agradeço imensamente pelo apoio ao longo de todo o período do mestrado, sem o qual esse trabalho teria sido impossível. Seu direcionamento e compreensão com as condições em que o trabalho foi realizado foram fundamentais.

Agradeço à minha família, Aline (e nossa pug Pixel), por darem todo o apoio necessário, suportando minhas longas noites e finais de semana dedicados ao projeto. Vocês foram uma grande motivação.

Agradeço ainda ao corpo de colegas da USP e professores por proporcionarem um bom ambiente ao longo de todo esse período, com várias discussões produtivas e grande aprendizado.

Esse trabalho não contou com qualquer financiamento.

Resumo

Lira, Diego Bezerra . **Detecção de bots no Twitter**: Uma abordagem utilizando agrupamento 2022. 66 f. Dissertação (Mestrado em Ciências) – Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, 2021.

O Twitter é uma importante mídia social, mas sofre com a presença constante de *bots*, perfis controlados por algoritmos. Esses *bots* podem ser malignos, se passando por usuários humanos e distorcendo o discurso da rede. Para possibilitar sua detecção, sistemas baseados em aprendizado de máquina são a solução mais comum. Entretanto, esses sistemas geralmente são limitados, pois só detectam tipos de *bots* muito próximos aos encontrados nos conjuntos de treinamento utilizados. Para remediar essa situação, a literatura explorou algumas opções, mas o uso de agrupamento para melhorar o desempenho nessa tarefa ainda é pouco difundido. O presente projeto avaliou alguns tipos de pré-processamento, utilizando um algoritmo de agrupamento, e conseguiu mostrar uma melhora nos resultados em parcela razoável dos testes. Neste trabalho também foi avaliado como os detectores mais comuns se comportam num cenário de mudança entre os dados de treino e o uso do modelo em outras bases de dados, justificando uma forma diferente de avaliar os resultados. Como conclusões, foi observado que uma avaliação em vários conjuntos de dados é necessária para um correto entendimento da performance, e os pré-processamentos avaliados são alternativas válidas que podem ser testadas para melhorar a performance dos modelos.

Palavras-chaves: Twitter. Detecção de *bots*. Agrupamento.

Abstract

Lira, Diego Bezerra. **Bot detection on Twitter**: An alternative using clustering. 2022. 66 p. Dissertation (Master of Science) – School of Arts, Sciences and Humanities, University of São Paulo, São Paulo, 2021.

Twitter is an important social media, but suffers from the constant presence of bots, profiles controlled by algorithms. These bots may be malign, passing themselves as humans and distorting general discourse in the network. To allow for their detection, systems based on machine learning are the most common solution. However, those systems are generally fragile, as they only detect the kinds of bots that are very similar to the ones found in the training set. To remedy this situation, the literature has explored a few options, but the use of clustering for optimizing performance is still uncommon. This project evaluated some types of preprocessing based on clustering algorithms, and achieved a marked improvement in a reasonable portion of the tests. We also evaluated how the most common detectors behave in a scenario of change between training data and live data, justifying a new method for evaluating results. As conclusions, it was observed that an evaluation in many datasets is necessary for a correct understanding of the performance, and the preprocessing alternatives proposed are valid techniques that might be tested to improve model performance.

Keywords: Twitter. Bots detection. Clustering.

Lista de figuras

Lista de quadros

Sumário

[1] De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

1 Introdução

Redes sociais são extremamente difundidas hoje em dia, tendo papel importante na formação de opiniões. Nestas redes é comum observarmos perfis controlados por algoritmos, definidos como *bots*.

Esses *bots* podem ter funções variadas, inclusive trazendo contribuições úteis para a comunidade. Por exemplo, *bots* que postam alertas sobre algumas situações (como alertas climáticos). Entretanto, também existem *bots* com ações potencialmente prejudiciais à sociedade, que podem realizar *spam* de produtos, *malware*, propagar notícias falsas (*fake news*) ou derivados. Cresci *et al.* (2017) apontam como uma categoria especialmente problemática os *social bots* - *bots* que tentam se passar por humanos para o resto da rede. Eles não são facilmente identificáveis por um humano, necessitando de sistemas específicos para sua detecção.

O Twitter é a principal plataforma para estudos sobre *bots* em redes sociais. Varol *et al.* (2017) estimam que entre 9% e 15% dos usuários ativos em inglês no Twitter têm comportamento semelhante a *bots*. Na literatura, os sistemas de detecção de *bots* para Twitter usam estratégias variadas, sendo comum a adoção de aprendizado de máquina com classificadores (aprendizado supervisionado).

Hoje, é de conhecimento comum a presença de *bots* no Twitter para divulgação de *fake news* e manipulação de discurso. Num exemplo recente, há uma notícia que cita que 45% das contas que publicam *fake news* sobre a pandemia de Covid-19 são operadas por *bots*[1].

Para permitir o combate a essas práticas, é essencial desenvolver sistemas de detecção de *bots*. Entretanto, observa-se uma corrida constante entre novas formas de gerar *bots* e novas formas de detecção. Echeverrià *et al.* (2018) evidenciam que os variados sistemas de detecção tendem a ter grande deterioração de desempenho numa análise *cross-domain*. Isso é especialmente verdadeiro para técnicas de aprendizado supervisionado, pois seu desempenho, via de regra, é avaliado em conjuntos de dados de *bots* que usam a mesma estratégia do treinamento (em geral não existe validação considerando diferentes conjuntos de dados). Yang *et al.* (2020) propõem uma forma de realizar essa análise, combinando diferentes conjuntos de dados para validar uma estratégia que melhore o

[1] https://computerworld.com.br/2020/05/26/cerca-de-45-das-contas-que-publicam-fake-news-sobre-pandemia-sao-bots-diz-estudo/

desempenho *cross-domain* de seu classificador. Entretanto, não foi encontrada na literatura nenhuma aplicação desse método para aprendizado não-supervisionado. Orabi *et al.* (2020) comentam que a literatura é muito concentrada em aprendizado supervisionado e encoraja novas pesquisas a explorarem alternativas não-supervisionadas.

Yang *et al.* (2020) também citam escalabilidade como empecilho à implantação de novas técnicas de detecção. A maioria da literatura usa um grande número de atributos na análise de perfis, com um processamento pouco prático para uma aplicação em larga escala.

1.1 Hipótese

Combinando métodos não-supervisionados (agrupamento) com métodos supervisionados, será possível desenvolver novas estratégias para detecção de *bots* que melhorem a detecção *cross-domain*.

Limitando os atributos usados no processo, poderemos endereçar a escalabilidade do método. Espera-se que esta limitação possa ser feita sem ocasionar grandes prejuízos ao desempenho da detecção de *bots*.

1.2 Objetivo

O objetivo do presente trabalho é propor uma nova abordagem para detecção de *bots* que apresente melhor desempenho considerando diferentes domínios (*cross-domain*) e que seja escalável.

Para isso, serão avaliadas diferentes formas de incorporar técnicas não-supervisionadas junto a técnicas supervisionadas, em diferentes conjuntos de dados.

Para verificar o desempenho da solução proposta, esta será comparada com um modelo utilizando *Random Forest*, seguindo as melhores práticas estabelecidas por Ferrara (2017).

1.3 Contribuições

Este trabalho tem por objetivo especificar, desenvolver e testar uma nova abordagem para a detecção de *bots* em redes sociais. Seu foco não é gerar um modelo específico, mas avaliar a validade da abordagem através de diferentes conjuntos de dados.

Como resultado, este trabalho disponibilizou um conjunto de dados completo para que seja possível sua reprodução, incluindo a extração do Twitter. Todos os códigos também foram disponibilizados, de forma que a avaliação *cross-domain* possa ser facilmente replicada em trabalhos futuros e este tipo de avaliação seja incentivado.[2]

Espera-se avançar o campo com novas sugestões de como detectar *bots* em sistemas reais, de forma robusta a novas estratégias para geração de *bots*, e escalável. Futuros trabalhos poderão desenvolver novas sugestões utilizando do mesmo alicerce.

[2] Todos os dados e códigos podem ser acessados em: https://github.com/dblop/bot_detector_project

2 Conceitos Básicos

Neste capítulo são apresentados alguns conceitos básicos, necessários para o entendimento do presente trabalho. Será explicada a natureza do Twitter e as principais abordagens para detecção de *bots*.

2.1 Twitter

Twitter[1] é uma rede social baseada no conceito de *microblogging*: Postagem de mensagens curtas, com as quais outros usuários podem interagir. Cada usuário tem um perfil associado e cada mensagem é popularmente conhecida como *tweet*.

Os usuários podem conectar seus perfis por meio de um relacionamento de seguidor e ter reações variadas sobre os tweets de outros usuários (por exemplo, repetir/repostar um tweet próprio ou de outro usuário, ação chamada de *retweet*).

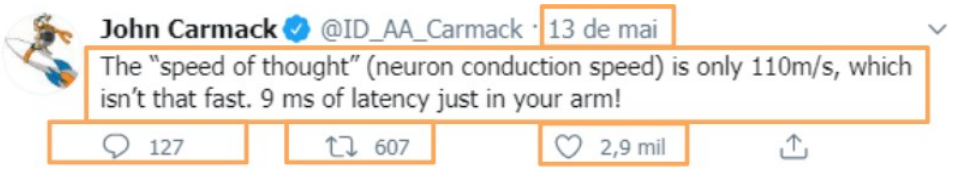
A figura 1 apresenta as informações disponibilizadas publicamente no perfil de um usuário. Cada marcação denota uma informação diferente, fornecida pela API do Twitter. Já a figura 2 ilustra um exemplo de tweet.

Figura 1 – Um perfil de Twitter

Fonte: Produzida a partir de dados extraídos do Twitter

[1] https://twitter.com/

Figura 2 – Um tweet



Fonte: Produzida a partir de dados extraídos do Twitter

2.1.1 Twitter bots

Os *bots*, usuários cujo comportamento é controlado de forma automática por algoritmos, são permitidos na plataforma Twitter.

Esses *bots* podem ter funções variadas, inclusive trazendo contribuições úteis para a comunidade. Um exemplo são *bots* que compartilham novas atividades em sites específicos ou que postam informações sobre estados de atenção relacionados a eventos climáticos, problemas no trânsito, etc. Entretanto, também existem *bots* com ações prejudiciais à sociedade, que podem realizar *spam* de produtos, *malware*, propagar *fake news*, difamar pessoas, produtos ou partidos políticos etc. Cresci *et al.* (2017) apontam como uma categoria especialmente problemática os *social bots* - *bots* que tentam se passar por humanos para o resto da rede, os quais não são facilmente identificáveis, necessitando de sistemas específicos para sua detecção.

Deve-se lembrar que os bots sociais são identificados por outros humanos ou sistemas, de forma que é mais correto falarmos em identificar comportamento semelhante a um *bot*. Não existe garantia de que a classificação esteja correta para todos os casos. A figura 3 apresenta o exemplo de um perfil anotado como *bot*.

2.2 Aprendizado de máquina

Uma definição célebre para aprendizado de máquina foi dada por Tom Mitchell (MURPHY, 2022):

> "Dizemos que um programa de computador aprende a partir da experiência E, com respeito a uma categoria de tarefas T, e medida de desempenho P, se seu desempenho nas tarefas T, medidas por P, melhora com a experiência E."

Figura 3 – Exemplo de perfil marcado como bot nos conjuntos de dados utilizados

Fonte: Produzida a partir de dados extraídos do Twitter

Os algoritmos de aprendizado de máquina dizem respeito à diversos tipos de tarefas e métricas. Focaremos aqui nos exemplos relevantes para este trabalho, que de forma geral consomem bases de dados como experiência e são também são descritos como métodos de aprendizado estatístico (JAMES *et al.*, 2013). Os métodos podem ser divididos entre (MURPHY, 2022):

- Aprendizado supervisionado: Existe uma variável Y, que gostaríamos de prever a partir de dados observados X. É análogo a encontrar $p(Y|X)$, sob uma visão probabilística.
- Aprendizado não-supervisionado: Gostaríamos de encontrar padrões a partir de dados observados X. É análogo a encontrar $p(X)$, sob uma visão probabilística.

Os dados são utilizados no processo de treinar um modelo matemático (definir o formato da equação e seus parâmetros), geralmente com o intuito de generalizar os resultados para novos conjuntos de dados. Entretanto, o erro do modelo nos dados utilizados durante o treino não é indicativo do erro em dados não utilizados no treino (JAMES

et al., 2013). Dessa forma, é importante termos outros dados separados para avaliar o desempenho.

2.3 Aprendizado supervisionado

Dada uma série de preditores $X = [X_0, X_1, ..., X_n]$ e uma variável de interesse Y, assumimos que existe um relacionamento na forma:

$$Y = f(X) + \epsilon$$

Estamos interessados na estimativa $\hat{Y} = \hat{f}(X)$, com uma medida de desempenho que minimize a diferença entre a estimativa e o valor real:

$$E(Y - \hat{Y})^2 \ = E[f(X) + \epsilon - \hat{f}(X)]^2 = [f(X) - \hat{f}(X)]^2 + Var(\epsilon)$$

Aqui $E(Y - \hat{Y})^2$ representa o quadrado do valor esperado da diferença entre o valor real e o predito, e $Var(\epsilon)$ representa a variância associada com o termo de erro ϵ. Essa variância é dita como o erro irredutível, e os algoritmos de aprendizado de máquina tentam estimar $\hat{f}(X)$ de forma a minimizar o erro $[f(X) - \hat{f}(X)]^2$.

As abordagens para definir $\hat{f}(X)$ podem ser paramétricas ou não-paramétricas. Na paramétrica, primeiro fazemos alguma suposição sobre o formato da função. Por exemplo:

$$\hat{f}(X) = \beta_0 X_0 + \beta_1 X_1 + ... + \beta_n X_n$$

Essa equação define uma função linear, cujos parâmetros $\beta = [\beta_0, \beta_1, ..., \beta_n]$ são ajustados durante o processo de aprendizagem. Numa abordagem não-paramétrica, não fazemos qualquer suposição sobre a forma de $\hat{f}(X)$, e ela também é aprendida pelo algoritmo.

Os problemas de aprendizado supervisionado podem ser divididos ainda entre os casos de regressão, em que temos uma variável $Y \in \Re$, ou classificação no caso de um conjunto finito de valores possíveis $Y = \{1, 2, ..., C\}$.

Várias técnicas diferentes foram desenvolvidas para lidar com eles, pois não existe uma forma universalmente superior para estimar $\hat{f}(X)$. Isso ocorre devido ao fato de que cada técnica faz suposições diferentes sobre os dados, cuja realidade dependerá enormemente do domínio do problema. Essa observação, conhecida como o teorema de

que não existe almoço grátis (MURPHY, 2022), nos informa que o algoritmo apropriado deve ser escolhido a partir do conhecimento de domínio ou tentativa e erro (através de técnicas de seleção).

2.3.1 Forma de aplicação para detecção de bots

Nesta abordagem, o problema de detecção de *bots* exige uma base de dados com a marcação de cada usuário como *bot* ou humano. O objetivo é usar um algoritmo classificador para conseguir relacionar alguns atributos preditivos com o atributo resposta, de forma a predizer o atributo resposta para novos exemplos.

É possível utilizar dados no nível do usuário (por exemplo, número de seguidores), bem como dados específicos de cada tweet. Um sistema que esteja classificando o usuário tipicamente usaria algumas medidas sumarizadas, como o intervalo médio entre tweets, mas também é possível classificar cada tweet e posteriormente sumarizar o resultado para o usuário.

Esse tipo de aprendizado é capaz de detectar *bots* isolados, mas seu desempenho é extremamente dependente do conjunto de dados usado para treinamento. A dificuldade em estabelecer uma resposta objetiva (como ter certeza que uma conta é um *bot*?) também interfere na comparação entre estudos, pois a geração das bases de dados não é comparável.

2.3.2 Algoritmo: Árvore de decisão

Usadas tanto para regressão quanto classificação, árvores de decisão particionam o espaço em regiões de forma recursiva. Para cada partição, um dos preditores $X = [X_0, X_1, ..., X_n]$ é selecionado, juntamente com um limiar para a partição. O conjunto de regras irá definir diversas regiões no espaço, por exemplo:

$$R_1 = \{X_1 < T_1, X_2 < T_2\}$$

$$R_2 = \{X_1 < T_1, X_2 \geq T_2\}$$

$$R_3 = \{X_1 \geq T_1, X_2 < T_2\}$$

$$R_4 = \{X_1 \geq T_1, X_2 \geq T_2\}$$

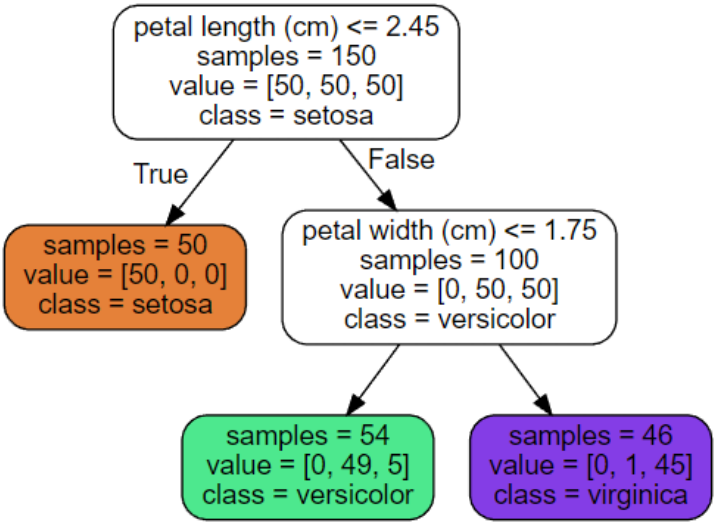
Temos aqui duas partições, que geram quatro espaços diferentes. Formalmente, considere a função indicadora:

$$\mathbf{1}_A(x) := \begin{cases} 1 & \text{se } x \in A \ , \\ 0 & \text{se } x \notin A \ . \end{cases}$$

O valor w_k atribuído ao espaço R_k será definido por:

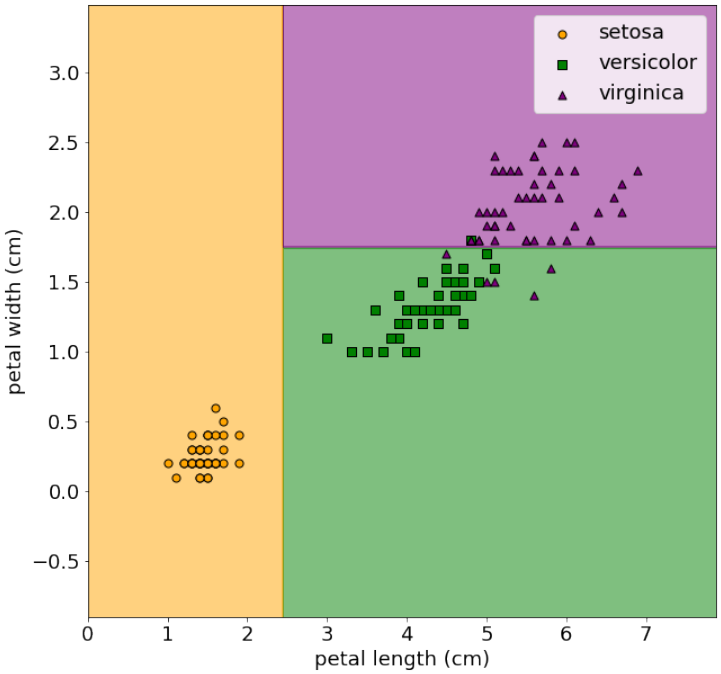
$$w_k = \frac{\sum_{n=1}^{N} y_n \mathbf{1}_{R_k}(X_n)}{\sum_{n=1}^{N} \mathbf{1}_{R_k}(X_n)}$$

Figura 4 – Exemplo de árvore de decisão criada na popular biblioteca python sklearn, para um problema de classificação. Nós brancos apresentam sua quebra no topo, seguido da quantidade total de exemplos no nó, exemplos por classe, e predição do nó. Nós finais estão coloridos por classe da sua predição. Figura original de (MURPHY, 2022).



Note que uma região é sempre selecionada, de forma que a soma jamais será 0. Podemos visualizar o algoritmo nas figuras 4 e 5. A descoberta de quais são as melhores partições e preditores consiste no treino do algoritmo da árvore de decisão. Descobrir a estrutura ótima da árvore é um problema NP-complexo, de forma que os algoritmos em uso usam técnicas gulosas para avaliar suas possibilidades (MURPHY, 2022). Uma forma de avaliação é testar, para cada quebra, todos os possíveis preditores $X = [X_0, X_1, ..., X_n]$, considerando todos os valores distintos que aparecem no conjunto de dados de treinamento como possibilidades. Geramos quebras binárias, e avaliamos qual opção leva à melhor métrica, comparando os nós filhos com o nó pai.

Figura 5 – Visualização de como o exemplo da figura 4 divide o espaço dos preditores. Figura original de (MURPHY, 2022).



Para isso definimos uma função custo que irá avaliar a quebra. No caso de regressão, considerando D exemplos no nó, temos:

$$custo(D) = \frac{1}{D}\sum_{n\in D}(y - \hat{y_n})^2$$

Em que $y = \frac{1}{D}\sum_{n\in D} y_n$, o valor médio da variável observada nos dados de treino. Enquanto que para o caso de classificação, temos para cada classe $c \in C$ possível:

$$\hat{\pi}_c = \frac{1}{D}\sum_{n\in D}\mathbf{1}_{y_n=c}$$

Que nos permite calcular o índice de Gini, nossa função custo:

$$\text{Gini} = \sum_{c=1}^{C}\hat{\pi}_c(1 - \hat{\pi}_c)$$

Calculando a melhor quebra recursivamente, chegamos eventualmente ao ponto em que apenas um exemplo fará parte de cada nó. Isso geralmente leva a uma baixo desempenho fora do conjunto de treino, e para combater isso costuma-se considerar uma quantidade mínima de exemplos nos nós finais.

Existem algumas vantagens em usar esse algoritmo. As árvores são simples de explicar e visualizar, podem lidar diretamente com dados faltantes (dado não-existente é

apenas mais uma quebra para avaliar), variáveis discretas ou contínuas, e naturalmente realizam um processo de seleção de variáveis (só quebras selecionadas fazem parte do modelo).

Suas desvantagens são um desempenho baixo na maioria dos problemas, e sua instabilidade (pequenas alterações nos dados de treino podem levar a árvores completamente diferentes).

2.3.3 Algoritmo: Random Forest

A instabilidade das árvores de decisão indica que o algoritmo sofre de alta variância (JAMES *et al.*, 2013). Dado um conjunto de observações independentes $\{Z_1, Z_2, ..., Z_n\}$, cada uma com variância σ^2, a variância do valor médio é $\bar{Z} = \sigma^2/n$, de forma que utilizar o valor médio de diversos modelos é uma forma de reduzir essa variância.

A técnica de bagging consiste em gerar B amostragens diferentes do conjunto de dados disponível, treinar um modelo diferente para cada conjunto amostrado e considerar a média (ou a classe majoritária, numa classificação) de todas as predições como a predição final:

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}_b(x)$$

Random forests vão além de bagging, pois além de amostrar os dados disponíveis por árvore, também amostram quais preditores $X = [X_0, X_1, ..., X_n]$ são considerados como candidatos em cada quebra de cada árvore. Isso diminui a similaridade entre as árvores, e leva a uma redução maior da variância do que um bagging puro (JAMES *et al.*, 2013).

2.4 Aprendizado não-supervisionado

Sem necessidade de uma marcação de resposta, o aprendizado não-supervisionado procura padrões dentro dos dados. Dado um conjunto de observações dos preditores $X = [X_0, X_1, ..., X_n]$, estamos interessados em avaliar propriedades de sua distribuição de probabilidade conjunta $p(X)$ (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). Técnicas como componentes principais tentam identificar um espaço em menos dimensões que

representa a maior densidade de informação possível, enquanto algoritmos de agrupamento tentam encontrar subgrupos dentro do espaço original.

Estamos interessados principalmente em agrupamento. Dada alguma medida de similaridade para avaliar diferentes observações, é possível dividir os dados em subgrupos. Entretanto, a validação da qualidade dos grupos é difícil, independente da técnica (MURPHY, 2022). Podemos usar critérios internos, a partir da medida de similaridade utilizada, ou comparar os grupos com alguma referência externa, se ela existir.

2.4.1 Forma de aplicação para detecção de bots

Técnicas de agrupamento são muito utilizadas para encontrar redes de *bots*, detectadas a partir de atributos muito particulares dos membros da rede. Por exemplo, um sincronismo persistente no *timestamp* de novos tweets pode ser considerado um marcador forte de uma rede.

Por definirem grupos, essas técnicas não são apropriadas para detectar *bots* isolados.

2.4.2 Algoritmo: K-Means

Dado um número de grupos K, inicialmente todas as observações são atribuídas a um grupo aleatório. Temos então duas etapas iterativas, que são executadas até o grupo atribuído a cada observação não ser mais alterado:

1. Para cada grupo, calculamos o centróide usando o valor médio dos elementos do grupo ao longo de cada dimensão.
2. Para cada observação, atualizamos o seu grupo utilizando o centróide mais próximo. A definição de distância utilizada geralmente é a euclidiana.

Formalmente, podemos dizer que o algoritmo K-Means está minimizando a medida de distância intra-grupo (distância entre elementos do mesmo grupo) (JAMES *et al.*, 2013). Isso é claro quando observamos a identidade, em que C_k são as observações no grupo K :

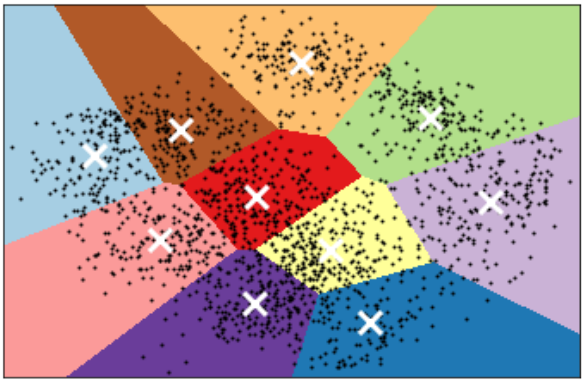
$$\frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = 2 \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})$$

Onde $\bar{x}_{kj} = \frac{1}{|C_k|} \sum_{i \in C_k} x_{ij}$. Ou seja, minimizar a distância aos centróides também minimiza a distância intra-grupo. Quando o algoritmo não alterar mais nenhuma observação, um valor ótimo local foi alcançado (JAMES *et al.*, 2013). Podemos ver na figura 6 como o algoritmo divide o espaço.

A inicialização aleatória influencia muito no resultado. Podemos executar o algoritmo diversas vezes, com inicializações diferentes, e escolher a versão com melhores métricas. Também podemos selecionar alguns pontos aleatórios diretamente como os centróides iniciais, economizando o cálculo do centróide na primeira iteração.

Outra alternativa é inicializar os centróides de forma sequencial, em que cada ponto subsequente é selecionado com uma probabilidade proporcional ao quadrado da distância do ponto ao centróide mais próximo. Essa variação é conhecida como a inicialização K-Means++ e geralmente melhora o desempenho (MURPHY, 2022).

Figura 6 – Exemplo do algoritmo K-Means dividindo o espaço em alguns grupos. Centróides em cruzes brancas. Figura original de (PEDREGOSA *et al.*, 2011).



3 Revisão de literatura

Orabi *et al.* (2020) realizaram uma extensa revisão sistemática sobre o assunto de identificação de *bots* em redes sociais, uma proposta semelhante a este trabalho. Entretanto, sua abordagem foi muito ampla. A revisão abordada neste capítulo contempla especificamente *bots* relacionados ao Twitter, detectados utilizando técnicas de aprendizado de máquina, almejando atingir conclusões mais específicas e detalhadas.

3.1 Método de pesquisa

Para ter uma visão global sobre o assunto de identificação de *bots* na plataforma Twitter, foi especificado um protocolo de revisão sistemática (KITCHENHAM, 2004). A partir dos critérios definidos no planejamento, os artigos aceitos são avaliados na etapa de condução da revisão. Os resultados extraídos são discutidos na próxima seção.

3.1.1 Planejamento

A fase de planejamento detalha qual o procedimento para a execução da revisão. São detalhados os objetivos, fontes, critérios para aceite dos artigos e quais perguntas pretende-se responder.

Objetivo

O objetivo da revisão é identificar os métodos e técnicas existentes para detecção de *bots* na plataforma Twitter. Espera-se identificar qual o atual estado da arte e lacunas relevantes.

Questões de pesquisa

A revisão busca responder às seguintes perguntas:

- Quais os métodos para detecção de *bots* no Twitter?
- Quais as dificuldades encontradas para melhorar o desempenho desses métodos?

- Quais as dificuldades em aplicar os métodos descritos para um uso prático?

Seleção de fontes:

A revisão foi realizada a partir de buscas em bibliotecas digitais, conforme critérios definidos abaixo.

Tipos de artigos: Será considerado válido qualquer artigo científico integralmente disponível nas fontes selecionadas que tenha passado por processo de revisão por pares.

Idiomas dos artigos: Inglês

Listagem de fontes:

- Elsevier (Scopus/ScienceDirect): https://www.sciencedirect.com/search
- IEEE Xplore: https://ieeexplore.ieee.org/
- ACM Digital Library: https://dl.acm.org/

Título, Resumo ou Palavras-chave: (bot AND Twitter) OR (bot AND "social network") OR ("bot detection") OR ("bot identification")

Critérios de inclusão e exclusão de trabalhos

Para avaliar a relevância dos artigos identificados pela busca inicial, serão aplicados alguns critérios para inclusão ou exclusão dos artigos.

Critérios de inclusão:

- Artigos completos disponíveis integralmente;
- Trabalhos que utilizem aprendizado supervisionado, não-supervisionado ou semi-supervisionado;
- Trabalhos que utilizem dados oriundos do Twitter;
- Trabalhos que apresentem uma abordagem para a detecção de *bots* em redes sociais;
- Trabalhos que efetivamente apliquem a abordagem proposta em um conjunto de dados.

Critérios de exclusão:

- Trabalhos publicados antes de 2016.
- Trabalhos secundários (revisões/*surveys*).

Após a leitura do resumo e aplicação dos critérios de inclusão e exclusão dos artigos retornados pelas *strings* de busca, o trabalho será selecionado se confirmada sua relevância. Serão selecionados os trabalhos que satisfaçam a todos os critérios de inclusão e nenhum dos critérios de exclusão. Havendo dúvida se um trabalho satisfaz ou não aos critérios de inclusão e exclusão, este será lido na íntegra para a verificação destes critérios. Destaca-se que o critério de exclusão referente ao período de publicação (trabalhos publicados antes de 2016 serão excluídos) foi aplicado, antes da aplicação dos demais critérios.

3.1.2 Condução

Conforme os critérios de busca, foram identificados 375 artigos para uma análise inicial. Destes, 45 artigos passaram inicialmente pelos critérios de inclusão, mas 5 foram excluídos por serem revisões. A lista completa de artigos aceitos/selecionados é apresentada no quadro 1. Os artigos aceitos foram lidos integralmente e são discutidos na próxima seção.

3.2 Resultados

Conforme pode ser observado na figura 7, os artigos estão bem distribuídos ao longo do tempo (a revisão foi realizada em meados de 2020), mostrando um interesse relativamente constante no campo de detecção de *bots* no período analisado. Pode-se dividir os trabalhos em alguns grandes grupos a partir das técnicas utilizadas na detecção. A abordagem mais comum é o uso de aprendizado supervisionado, algo já sinalizado por Orabi *et al.* (2020). Já o aprendizado não-supervisionado é usado especialmente para detectar redes de *bots* que atuam em sincronia. A seguir a revisão é resumida em uma seção com as conclusões mais relevantes para o trabalho atual, enquanto as próximas seções detalham os trabalhos analisados em cada categoria.

Quadro 1 – Artigos selecionados pelos critérios de inclusão

Referência	Inclusão	Exclusão	Aceito?
Rodríguez-Ruiz *et al.* (2020)	X		X
Price, Priisalu e Nomm (2019)	X		X
Kudugunta e Ferrara (2018)	X		X
Clark *et al.* (2016)	X		X
Orabi *et al.* (2020)	X	X	
Cresci *et al.* (2020)	X		X
Kantepe e Ganiz (2017)	X		X
Fernquist, Kaati e Schroeder (2018)	X		X
Velayutham e Tiwari (2017)	X		X
Wei e Nguyen (2019)	X		X
Bello, Heckel e Minku (2018)	X		X
Schnebly e Sengupta (2019)	X		X
Kiran *et al.* (2019)	X		X
Beskow e Carley (2018)	X		X
Alothali *et al.* (2018)	X	X	
Chavoshi, Hamooni e Mueen (2016)	X		X
Boreggah *et al.* (2018)	X		X
Minnich *et al.* (2017)	X		X
Wu *et al.* (2020)	X		X
Subrahmanian *et al.* (2016)	X	X	
Loyola-González *et al.* (2019)	X		X
Khalil, Khan e Ali (2020)	X		X
Chavoshi e Mueen (2018)	X		X
Walt e Eloff (2018)	X		X
Dorri, Abadi e Dadfarnia (2018)	X		X
Gilani, Kochmar e Crowcroft (2017)	X		X
Antoun *et al.* (2020)	X	X	
Cresci *et al.* (2018)	X		X
Khaled, El-Tazi e Mokhtar (2018)	X		X
Zhang *et al.* (2016)	X		X
Morstatter *et al.* (2016)	X		X
Mazza *et al.* (2019)	X		X
Abozinadah e Jones (2017)	X		X
Moon (2017)	X		X
Cornelissen *et al.* (2018)	X		X
Alarifi, Alsaleh e Al-Salman (2016)	X		X
Fazil e Abulaish (2017)	X		X

3.2.1 Resumo dos resultados

Entre as formas de detectar *bots*, a maioria dos sistemas utiliza técnicas de aprendizado de máquina. Alternativas incluem heurísticas em dados como a topologia da rede de

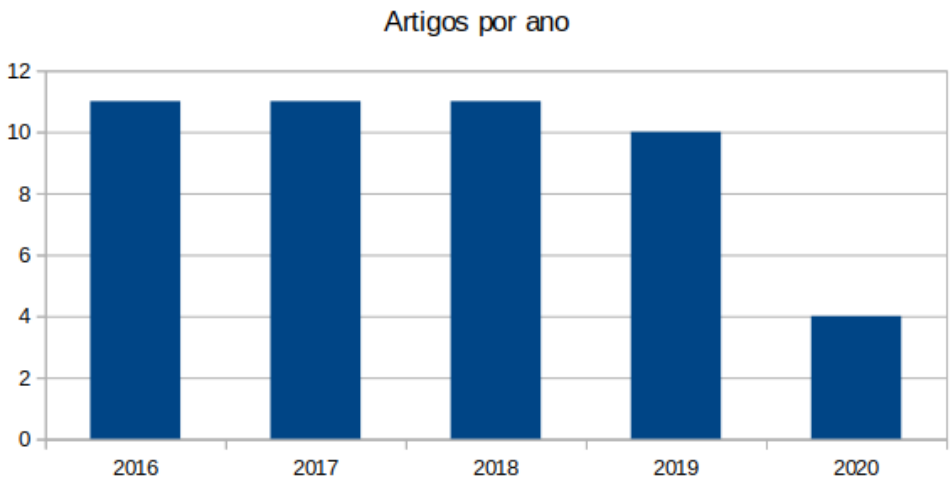
conexões entre usuários ou o tempo entre ações. A literatura é muito focada no Twitter, em parte devido a existência de uma API simples de usar, disponibilidade de conjuntos de dados públicos e a prevalência de *bots* na plataforma. Por isso, o foco do trabalho é detecção de bots no Twitter.

Sistemas de aprendizado de máquina podem ser divididos entre modelos supervisionados e não-supervisionados, e a maioria da literatura sobre *bots* foca nos supervisionados (ORABI *et al.*, 2020). Esses modelos podem usar centenas de atributos para cada predição, tanto no nível de mensagem (por exemplo, número médio de tokens por tweet) quanto no nível de usuário (por exemplo, número de amigos). Algumas categorias comuns incluem diferentes agregações (soma, média, contagem) considerando a topologia da rede, tempo em atividade, análise de sentimentos dos tweets, análise dos tokens de texto, bem como metadados de usuário (por exemplo, existência de um plano de fundo customizado).

Entre sistemas disponíveis para uso público, o Botometer (FERRARA *et al.*, 2016), desenvolvido pela Universidade de Indiana, é um exemplo que usa todas as categorias mencionadas. É um sistema que usa uma Floresta Aleatória (*Random Forest*) para predizer a probabilidade de uma conta de usuário ser um *bot*. Outros sistemas na literatura usam apenas algumas das categorias mencionadas (ALARIFI; ALSALEH; AL-SALMAN, 2016; KANTEPE; GANIZ, 2017), ou um algoritmo diferente (KHALED; EL-TAZI; MOKHTAR, 2018).

Existe uma grande dificuldade em comparar o resultado de diferentes soluções. Os conjuntos de dados não seguem um critério claro para identificar o que é um *bot*, dado que a marcação verdadeira (*ground truth*), em muitos casos, é impossível de obter. Muitos

Figura 7 – Divisão dos artigos aceitos por ano



bots são identificados manualmente. Assim, é difícil comparar o desempenho em estudos que usam conjuntos de dados diferentes. Mesmo conjuntos de dados públicos muitas vezes fornecem apenas a marcação *bot / not bot*, e o pesquisador precisa coletar os dados através da API do Twitter. Naturalmente, o resultado dessa coleta pode variar com o tempo conforme políticas internas do Twitter quanto a retenção de dados ou alterações na API.

O desempenho observado em muitos estudos é extremamente alta, mas isso é difícil de manter conforme mudamos o conjunto de dados. Para criar sistemas úteis para uma detecção genérica de *bots*, existe uma sugestão de usar avaliações *cross-domain*, ou seja, testar o modelo contra conjuntos de dados que não fizeram parte do treinamento (ECHEVERRIA *et al.*, 2018).

Quando olhamos os sistemas não-supervisionados, a maioria dos trabalhos foca em detectar grupos de usuários com comportamento semelhante a *bots*, não sendo apropriados para detectar *bots* isolados. Um grupo (ocasionalmente chamado de *botnet*) pode ser formado por indivíduos com padrões particulares, como um sincronismo temporal entre suas atividades, que é mantido por uma quantidade de tempo razoável. Existem algumas variações dessa ideia, observando por exemplo sincronismo temporal das postagens (CHAVOSHI; HAMOONI; MUEEN, 2016) ou a mesma sequência de ações (CRESCI *et al.*, 2018). Esses sistemas utilizam algoritmos de agrupamento, e os estudos encontram dimensões que permitem separar os grupos em *bot / not bot*.

Apenas alguns trabalhos lidam com detecção de anomalias (MINNICH *et al.*, 2017), podendo ser usados da mesma forma que os supervisionados para a detecção de *bots* isolados, dado que os métodos retornam um escore de anomalia por usuário.

Usar algoritmos de agrupamento como uma etapa de pré-processamento já foi apontado como uma possibilidade interessante de ser explorada (YANG *et al.*, 2020). Não existem muitos trabalhos que combinam técnicas supervisionadas e não-supervisionadas.

Também já foi observado que a maioria dos sistemas propostos na literatura tem dificuldades de escalabilidade, não sendo práticos para um monitoramento do Twitter devido aos altos custos de pré-processamento associados com a geração de atributos para os modelos (YANG *et al.*, 2020).

Dessa forma, entendemos que um sistema de uso geral para detecção de *bots* em um ambiente produtivo deve, preferencialmente, usar atributos simples e ter seu desempenho comparado em diversos conjuntos de dados.

3.2.2 Aprendizado supervisionado

Price, Priisalu e Nomm (2019) estão preocupados em analisar a fragilidade dos métodos supervisionados em relação aos dados. Através de manipulações no conjunto de treinamento, concluem que é possível diminuir de forma significativa a precisão de um detector (cerca de 5% menos de precisão para 2% de alteração na variável resposta *bot*/humano), com repercussões para possíveis agentes que poluírem conjuntos de dados.

Kudugunta e Ferrara (2018) usa uma rede neural *LSTM (Long Short-Term Memory* para prever *bots* no nível de tweet, ao invés do nível de usuário como é usual. O texto do tweet é transformado num *embedding GloVE*, que alimenta uma LSTM e, posteriormente, é combinado com atributos no nível de usuário para gerar a classificação final. Concluíram que os atributos de usuário diminuem consideravelmente o erro, quando se considera a predição no nível de usuário.

Clark *et al.* (2016) usam uma heurística para construir um classificador. A partir de medidas textuais retiradas de uma amostra de tweets do usuário, gera-se o resultado. As medidas avaliadas são a estrutura do texto (relação entre número de palavras e espaços), quantidade de URLs por tweet e uma sumarização da capacidade de introduzir vocabulário ao longo de vários tweets. O estudo mostrou a validade de se utilizar os atributos propostos.

Kantepe e Ganiz (2017) testam alguns classificadores diferentes, sendo que a melhor técnica foi *Gradient Boosted Tree*. O modelo usa informações obtidas no nível de usuário e no nível de tweet, sendo para isso utilizada a API do Twitter considerando os usuários com atividade nos *trending topics* e posteriormente sua *timeline*. Para definir a resposta nos dados coletados, os autores assumem que usuários suspensos no último mês da coleta de dados são *bots* (os usuários foram coletados ao longo dos 3 meses anteriores). Essa abordagem claramente gera um conjunto de dados falho, mas os autores escolheram essa opção dada sua simplicidade. Para gerar os atributos do modelo, pode-se destacar que alguns deles envolveram análise de sentimentos, sendo necessário o uso de um classificador de sentimento como pré-processamento para a identificação de *bots*.

Fernquist, Kaati e Schroeder (2018) utilizaram um classificador *Random Forest*, a partir de alguns conjuntos de dados disponibilizados previamente por outros grupos. Esses conjuntos não são disponibilizados de forma integral, apenas sua resposta, o que força os autores a coletarem os dados faltantes e impede uma comparação direta com outros

artigos. Os atributos utilizados são no nível de usuário e no nível de tweet, com a presença de alguns atributos temporais. Os autores então coletam os tweets referentes à eleição da Suécia com base em *hashtags*, e usam o classificador para identificar os *bots* e analisar suas tendências. Essa análise leva a crer que os *bots* usam mais URLs externas que usuários legítimos (humanos), numa proporção de 20% contra 9%.

Velayutham e Tiwari (2017) criam algumas regras simples para classificar o usuário, a partir de atributos gerados no nível de usuário e no nível de tweet. Seu conjunto de dados foi anotado manualmente. Este trabalho contribui por mostrar que é possível conseguir um desempenho razoável (precisão de 80% no teste) mesmo com poucos atributos (menos de 10). Entretanto, não explicam como chegaram nas suas regras para a classificação, tornando difícil discutir seu mérito.

Wei e Nguyen (2019) utilizam um classificador *BiLSTM*, uma proposta pouco comum. O texto do tweet é transformado em *embeddings* a partir de um modelo pré-treinado com a técnica GloVE, e isso serve de entrada para a rede neural *BiLSTM*, sem nenhum outro atributo. Seu desempenho é comparável ao estado da arte para o conjunto de dados analisado. Os autores propõem que a vantagem do seu método é um processamento mais simples, comparado à geração de até 1000 atributos encontrada em outros modelos. Como não existe nenhuma discussão sobre o tempo utilizado para a conversão em *embeddings*, é difícil avaliar a veracidade dessa afirmação. Outros modelos mencionados no artigo também relatam desempenho semelhante com um número muito menor de atributos.

Bello, Heckel e Minku (2018) estão preocupados em gerar regras a partir do comportamento de *bots*, mas sua abordagem também permite uma classificação posterior a partir das regras. Usando um conjunto de dados inicial fornecido em outro artigo, os autores propõem algumas regras que identificam um *bot*, ligando uma ação (tweet ou retweet) a uma lista de palavras-chave. As regras foram geradas inicialmente a partir de árvores de decisão, treinadas a partir de atributos sobre os tópicos do tweet. Sua principal contribuição está na geração de regras que permitem intuições sobre os *bots*.

Schnebly e Sengupta (2019) usam dados disponibilizados publicamente para gerar um classificador *Random Forest*. A partir de alguns atributos iniciais no nível de usuário são gerados atributos derivados com variadas razões entre os atributos iniciais. São usados dois conjuntos de dados e, por isso, três modelos são treinados: Primeiro treinando e testando no conjunto 1, depois treinando e testando nos conjuntos 1 e 2, e posteriormente

treinando no conjunto 1 e testando no conjunto 2. Os autores estão cientes da necessidade de se testar em conjuntos de dados diferentes do treino para chegar em métricas mais realistas para um uso produtivo.

Kiran *et al.* (2019) propõem uma rede neural para identificação de *bots*, a partir de dados no nível de usuário. Alguns atributos derivados são gerados na forma de razões. O conjunto de dados foi gerado a partir de uma busca no Twitter entre os amigos de alguns usuários iniciais, e usuários previamente marcados como *bots*. É difícil avaliar o trabalho, pois os autores não explicam o fluxo dos experimentos.

Beskow e Carley (2018) estão interessados em identificar qual nível de informação é relevante para a detecção de *bots*. O conjunto de dados utilizado é composto por alguns *bots* previamente disponibilizados publicamente, e um conjunto de usuários capturados a partir da API do Twitter e considerados humanos (os autores esperam um erro de até 8%, baseado no percentual de usuários que são *bots* apontado na literatura). A partir disso, são gerados alguns níveis de atributos, que podem ser resumidos como: nível de usuário (1), *timeline* (2), e atributos dos amigos (3). Comparando modelos *Random Forest* treinados com cada nível, chega-se na conclusão que a maior parte da informação está contida no nível 1, com um acréscimo de desempenho adicionando o nível 2, mas com pouca diferença no nível 3. Os autores também comentam a importância de validar os resultados num conjunto de dados para teste diferente do conjunto de dados usado para treino.

Boreggah *et al.* (2018) focam em um conjunto de dados em árabe. A partir de serviços árabes de compra de *bots* e anotação manual, um conjunto de dados foi construído. Os atributos utilizados foram no nível de usuário e no nível de tweet, incluindo sumarizações de atividade ao longo do tempo. A partir disso foram treinados alguns classificadores e o melhor resultado foi obtido com o algoritmo *Random Forest*, seguido de SVM.

Wu *et al.* (2020) usam uma rede neural para classificar. A contribuição deste trabalho está no fato de que a tarefa de detecção de *bots* costuma lidar com conjuntos de dados desbalanceados e existem alguns algoritmos (por exemplo, SMOTE) para balancear os dados. Neste trabalho é proposto um novo algoritmo para uma amostragem balanceada, usando uma rede generativa adversária, que apresentou um desempenho melhor nas comparações realizadas.

Loyola-González *et al.* (2019) propõem um classificador por contraste. Em resumo, várias árvores de decisão são utilizadas para extrair um conjunto de regras inicial (por exemplo, *friend count* $<$ *10 AND favorites count* $<$ *10 IS bot*), que são posteriormente

simplificadas com a retirada de duplicações. A classificação é feita com uma medida derivada do suporte para cada regra, ou seja, o percentual da base em que a conclusão é válida dadas as regras. Pode-se observar que o método proposto não ganha de classificadores tradicionais em uma análise isolada, mas tem um desempenho médio mais alto, considerando diferentes conjuntos de dados. Podemos aqui considerar a intuição que um classificador mais simples generaliza melhor, considerando uma análise *cross-domain*.

Chavoshi e Mueen (2018) enfatizam o modelo *RSC (Rest-Sleep-Comment)*, treinado para detecção de *bots* na plataforma Reddit e baseado na diferença temporal entre atividades. No Twitter, os autores avaliam que não existe grande separação no padrão temporal entre usuários humanos e *bots* no formato esperado pelo RSC. Utilizando uma medida de diferença temporal de segunda ordem e uma rede neural convolucional, os autores chegam num resultado melhor que o RSC para o Twitter. Uma contribuição foi mostrar que os *bots* apresentam menor entropia temporal que os humanos, sugerindo que modelos procurando a similaridade temporal entre usuários podem obter sucesso.

Walt e Eloff (2018) usam uma definição diferente da maioria. Ao invés de *bots* controlados por algoritmos, os autores estão interessados em contas de usuários humanos que tenham um comportamento deceptivo. Para treinar alguns classificadores, é gerada uma base com usuários sintéticos, em que as dimensões mais comumente alteradas por usuários deceptivos são modificadas de forma aleatória, mas sem alterar significativamente a média do atributo na base completa (sintéticos mais outros). Um classificador posterior tem um desempenho menor que o comumente visto na literatura, e os autores sugerem que esse tipo de usuário seja mais difícil de detectar. Entretanto, os atributos utilizados para chegar nessa conclusão dependem das dimensões alteradas. É difícil avaliar se o efeito é real ou apenas resultado da forma como os autores geraram as alterações.

Gilani, Kochmar e Crowcroft (2017) treinam um classificador *Random Forest* com atributos gerados no nível de tweet. O diferencial da abordagem é dividir o conjunto de dados em alguns grupos de popularidade, identificada pelo tamanho da rede. Dessa forma, cada grupo tem seu próprio classificador, melhor adaptado ao tipo de usuário do grupo. Isso se reflete em métricas um pouco mais altas.

Antoun *et al.* (2020) geram três classificadores usando as técnicas *Random Forest*, *AdaBoost* e *XGBoost*. Eles são posteriormente combinados numa votação para definir a resposta. A solução usa atributos no nível de tweet e no nível de usuário, chamando a

atenção para um atributo de idade da conta, que é capaz de uma precisão de 96% sendo usado de forma isolada.

Khaled, El-Tazi e Mokhtar (2018) apresentam uma técnica inovadora para classificação chamada de SVM-NN (*Support Vector Machine - Neural Network*). A combinação das duas técnicas em um modelo híbrido atingiu resultados melhores do que os métodos individualmente.

Alarifi, Alsaleh e Al-Salman (2016) testaram alguns tipos diferentes de classificadores, mas sua principal diferenciação está na construção do conjunto de dados. Os autores consideram contas de usuários humanos, híbridos ou sybils (*bots*), em vez da classificação binária mais comum. Aqui os híbridos são usuários cujos tweets foram identificados como alternando entre humanos e *bots*. Os mesmos tipos de classificadores tiveram melhor desempenho no problema com duas ou três classes.

3.2.3 Aprendizado não-supervisionado

Cresci *et al.* (2018) introduzem a técnica de *DNA-Fingerprinting*. Mapeando as ações do usuário dentro de alguns alfabetos de ações possíveis (por exemplo, dividindo ações no conjunto {tweet, retweet, resposta}, é possível gerar uma sequência de ações). Analisando a proximidade da sequência entre dois usuários, é possível identificar alguns grupos, vindo daí a separação entre humanos e *bots*. Essa proximidade é definida pela *LCS (Longest Common Subsequence)*, a maior sequência comum entre usuários. Conforme aumentamos o número de usuários comparados, os humanos têm uma tendência observada de sua LCS cair rapidamente, enquanto *bots* mantêm uma distribuição claramente distinta, com uma LCS decaindo mais lentamente. Pode-se usar isso para um aprendizado não-supervisionado, em que a derivada da LCS dado um novo usuário é usada como distância para dividir agrupamentos de usuários semelhantes. Dividindo os dados em dois agrupamentos, escolhe-se um ponto de LCS a partir do qual considera-se os usuários como um grupo de *bots*.

Em 2020, Cresci *et al.* (2020) expandem seu trabalho anterior sobre a técnica de *DNA-Fingerprinting*. Uma conclusão interessante é a importância de se considerar variáveis temporais, com os autores sugerindo incluir o tempo entre atividades como um próximo passo.

Chavoshi, Hamooni e Mueen (2016) identificam *bots* usando técnicas de agrupamento (*clustering*), a partir da correlação temporal entre atividades de um conjunto de usuários. *Dynamic time warping (DTW)* é uma técnica padrão para medir similaridade entre duas séries temporais, na forma de uma distância entre as séries. A distância apontada pela DTW é normalizada na forma de um coeficiente de *warped correlation*. O algoritmo proposto envolve procurar todos os usuários que têm atividade relacionada a alguma palavra-chave e capturar sua atividade ao longo do tempo. Os usuários são divididos em algumas categorias iniciais a partir de uma função hash, calculada com base na correlação do usuário com uma série temporal aleatória. O propósito dessa etapa é identificar um conjunto inicial de usuários suspeitos. A partir desses usuários, mais dados são coletados e o coeficiente de *warped correlation* é calculado. Para classificar um usuário como *bot* é usado um algoritmo de agrupamento hierárquico do tipo *single linkage* com um alto ponto de corte. Todos os grupos (*clusters*) formados são considerados redes de *bots*, enquanto os usuários que não foram agrupados em clusters são humanos. Grupos com amigos em comum são posteriormente agrupados em um único *cluster*. O algoritmo é validado a partir de usuários que são suspensos no Twitter nos próximos 3 meses, uma métrica que englobou 45% dos *bots* detectados de forma não-supervisionada.

Khalil, Khan e Ali (2020) testam alguns atributos e técnicas de agrupamento e seu objetivo principal é sugerir a melhor abordagem para outros pesquisadores. No teste realizado, concluíram que alguns atributos interessantes para usar como dimensão do agrupamento seriam: followers_count, friends_count, favourite_count, listed_count, retweet_count, mention_count, hashtag_count. Esses atributos estão disponíveis pela API do Twitter. Entre as técnicas k-means e dbscan, sugerem o uso de dbscan.

3.2.4 Detecção de anomalias

Rodríguez-Ruiz *et al.* (2020) consideram que os detectores supervisionados não se adaptam bem a novos conjuntos de dados e propõem utilizar *one-class classifiers*, por exemplo *Bagging-TPMiner* e *Bagging-RandomMiner*. Através de vários testes, mostram que os métodos supervisionados são superiores para os conjuntos de dados incorporados no treinamento, mas apresentam um desempenho pior que o método alternativo quando o conjunto de dados não foi disponibilizado para treinamento.

Minnich *et al.* (2017) utilizam uma estratégia de exploração e posterior detecção dos *bots* a partir de anomalias. Dado um conjunto inicial de *seed bots* e usuários comuns, a rede de primeiro nível de cada usuário é explorada e um *ensemble* de algoritmos de detecção de anomalia é usado para detectar novos *bots*. Os algoritmos são aplicados em um conjunto de atributos com nível de usuário e sumarizações de tweet, além da topologia da rede. Para avaliação do método é considerada a concordância dos usuários sinalizados como *bots* por outros algoritmos, além de uma anotação manual. Um ponto forte da solução é a seleção de atributos realizada por nível de dado (usuário, sumarização de tweet, topologia), que os autores apontam funcionar melhor que uma seleção geral.

Dorri, Abadi e Dadfarnia (2018) criam o modelo SocialBotHunter. A partir de uma extração inicial de atributos, um escore de anomalia é calculado usando *one-class SVM*. Esse escore é então ajustado a partir da rede de relacionamento do usuário, considerando que essa rede é um *Markov Random Field (MRF)* com uma função potencial dependente do escore inicial. O método vence os comparativos supervisionados, mas admite que não realiza uma comparação justa devido a incompatibilidades nas abordagens. De toda forma, é uma outra validação do uso de escores de anomalia com *one-class classifiers*.

3.2.5 Discussão

Conforme apresentado neste capítulo, os trabalhos de aprendizado supervisionado dominam a literatura. Os modelos de aprendizado não-supervisionado estão mais preocupados em identificar grupos de *bots* atuando de forma síncrona, ao invés de casos isolados. A detecção de anomalias surge como uma forma de detectar esses casos isolados, sem precisar de uma resposta como os métodos supervisionados.

Alguns trabalhos reconhecem os desafios do aprendizado supervisionado em generalizar além do seu conjunto de treinamento (SCHNEBLY; SENGUPTA, 2019; PRICE; PRIISALU; NOMM, 2019), enquanto outros reconhecem os limites práticos para *deploy* de modelos complexos (WEI; NGUYEN, 2019; BESKOW; CARLEY, 2018), mas a maioria se limita a propor classificadores sem considerar um uso real. Também observa-se pouca interação entre as abordagens, com poucos trabalhos sugerindo classificadores diferentes por grupo de usuários (GILANI; KOCHMAR; CROWCROFT, 2017), o que seria uma aplicação clara de métodos não-supervisionados.

A definição do problema também é um desafio. A maioria dos autores coletou seus próprios conjuntos de dados, com seu próprio critério do que seria um *bot*, o que dificulta comparações.

4 Materiais e métodos

O conjunto de dados utilizado neste trabalho foi extraído do repositório da Indiana University[1], em que são disponibilizados vários conjuntos de dados para o Twitter, divididos em humanos e *bots*. A primeira seção deste capítulo detalha os conjuntos utilizados.

Entretanto, a maioria dos dados disponibilizados tem apenas a identificação do usuário e sua classificação. Para obter os dados do usuário, é necessário utilizar a API do Twitter. A segunda seção deste capítulo trata desse processo.

4.1 Conjuntos de dados

Este projeto utiliza vários conjuntos de dados coletados de forma independente, e seus tamanhos finais (pós captura de dados) podem ser verificados na figura 8.

Alguns conjuntos de dados fornecidos incluem os dados de usuário que serão utilizados ao longo do projeto. Temos:

- cresci-2015: Gerado por anotação manual, divide bots (com a categorização *fake followers*) e humanos em alguns subconjuntos:
  - TFP (the fake project): 100% humanos
  - E13 (elections 2013): 100% humanos
  - INT (intertwitter): 100% *bot*
  - FSF (fastfollowerz): 100% *bot*
  - TWT (twittertechnology): 100% *bot*
- cresci-2017: Gerado por anotadores manuais do serviço CrowdFlower, faz uma distinção entre *bot* com comportamentos diferentes com as categorizações de *fake followers*, *social spambots* e *traditional spambots*, com cada subconjunto vindo de uma origem diferente. Divide-se nos subconjuntos:
  - fake followers: 100% *bot*
  - genuine accounts: 100% humanos
  - social spambots 1: 100% *bot*
  - social spambots 2: 100% *bot*
  - social spambots 3: 100% *bot*

[1] https://botometer.osome.iu.edu/bot-repository/datasets.html

- social spambots 4: 100% *bot*
- traditional spambots 1: 100% *bot*
- traditional spambots 2: 100% *bot*
- traditional spambots 3: 100% *bot*
- traditional spambots 4: 100% *bot*

Outros conjuntos de dados utilizados incluem apenas a identificação do usuário e sua classificação, de forma que a extração de informações se faz necessária. São:

- botometer-feedback-2019: Contas anotadas manualmente como parte do processo de feedback do sistema detector Botometer
- botwiki-2019: *Bots* identificados no site https://botwiki.org
- celebrity-2019: Contas autênticas de celebridades
- cresci-rtbust-2019: Contas anotadas manualmente
- cresci-stock-2018: *Bots* com a característica de atuação síncrona
- gilani-2017: Contas anotadas manualmente
- midterm-2018: Contas anotadas manualmente, com atividade em torno da eleição americana de 2018
- pronbots-2019: *Bots* que fazem propaganda de serviços pornográficos
- varol-2017: Contas anotadas manualmente a partir de um *crawling* realizado em 2016
- vendor-purchased-2019: *Bots* comprados no mercado cinza para inflar o número de seguidores

4.2 Obtenção de informações

Para o trabalho desenvolvido foram usados os atributos a nível de usuário (nenhuma informação derivada dos tweets individuais). Essa escolha considera o que foi observado na revisão de literatura: Um modelo com informações simples ainda pode atingir alto desempenho, com um baixo custo de pré-processamento dos atributos. As informações necessárias podem ser extraídas a partir de uma única chamada à API do Twitter, consultada a partir da biblioteca Python Tweepy[2]. O trecho de código a seguir exemplifica o que é necessário:

[2] https://www.tweepy.org/

```
import tweepy
auth = tweepy.AppAuthHandler(consumer_key, consumer_secret)
api = tweepy.API(auth)
user = api.get_user(user_id=user_id)
```

Em que consumer_key e consumer_secret são elementos de uma conta de desenvolvedor do Twitter, disponível para qualquer pessoa que siga o processo cadastral, e user_id representa o identificador da conta de usuário, disponibilizado nos conjuntos de dados. O objeto *user* retornado é um dicionário com as informações do usuário.

A API do Twitter sofre mudanças ao longo do tempo e é possível que extrações feitas por outros autores contenham uma quantidade diferente de campos retornados. Também é esperado que outras extrações contenham dados diferentes, refletindo mudanças no perfil de uso dos usuários ao longo do tempo.

Muitos usuários também são banidos ao longo tempo, de forma que suas informações não são mais acessíveis pela API. Assim, o conjunto de dados extraídos e usados no trabalho não engloba todos os usuários originalmente fornecidos nos conjuntos de dados, com números finais disponíveis na figura 8.

Para este projeto, consideramos apenas os campos cujas informações estavam disponíveis para todos os conjuntos de dados.

4.3 Técnicas utilizadas

Para os testes com agrupamento, foi utilizada a técnica *K-Means*. Este trabalho pretende mostrar a utilidade de técnicas de agrupamento para o problema, não existindo qualquer limitação específica ao *K-Means*. O algoritmo inicialmente selecionado meramente reflete sua difusão como uma técnica simples, comumente usada e com bons resultados.

Como algoritmo de classificação, seja como base ou após um pré-processamento com agrupamento, será utilizado *Random Forest*. A técnica é utilizada em vários detectores na literatura com bom desempenho, conforme indicado por Orabi *et al.* (2020).

Os parâmetros utilizados foram, em sua maioria, padrões da biblioteca scikit-learn, reproduzidos aqui:

- class sklearn.cluster.KMeans(n_clusters=8, *, init='k-means++', n_init=10, max_iter=300, tol=0.0001, verbose=0, random_state=None, copy_x=True, algorithm='lloyd')

- class sklearn.ensemble.RandomForestClassifier(n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None)

Os únicos parâmetros diferenciados foram random_seed=42 e os experimentos utilizaram KMeans com n_clusters=2 e n_clusters=3.

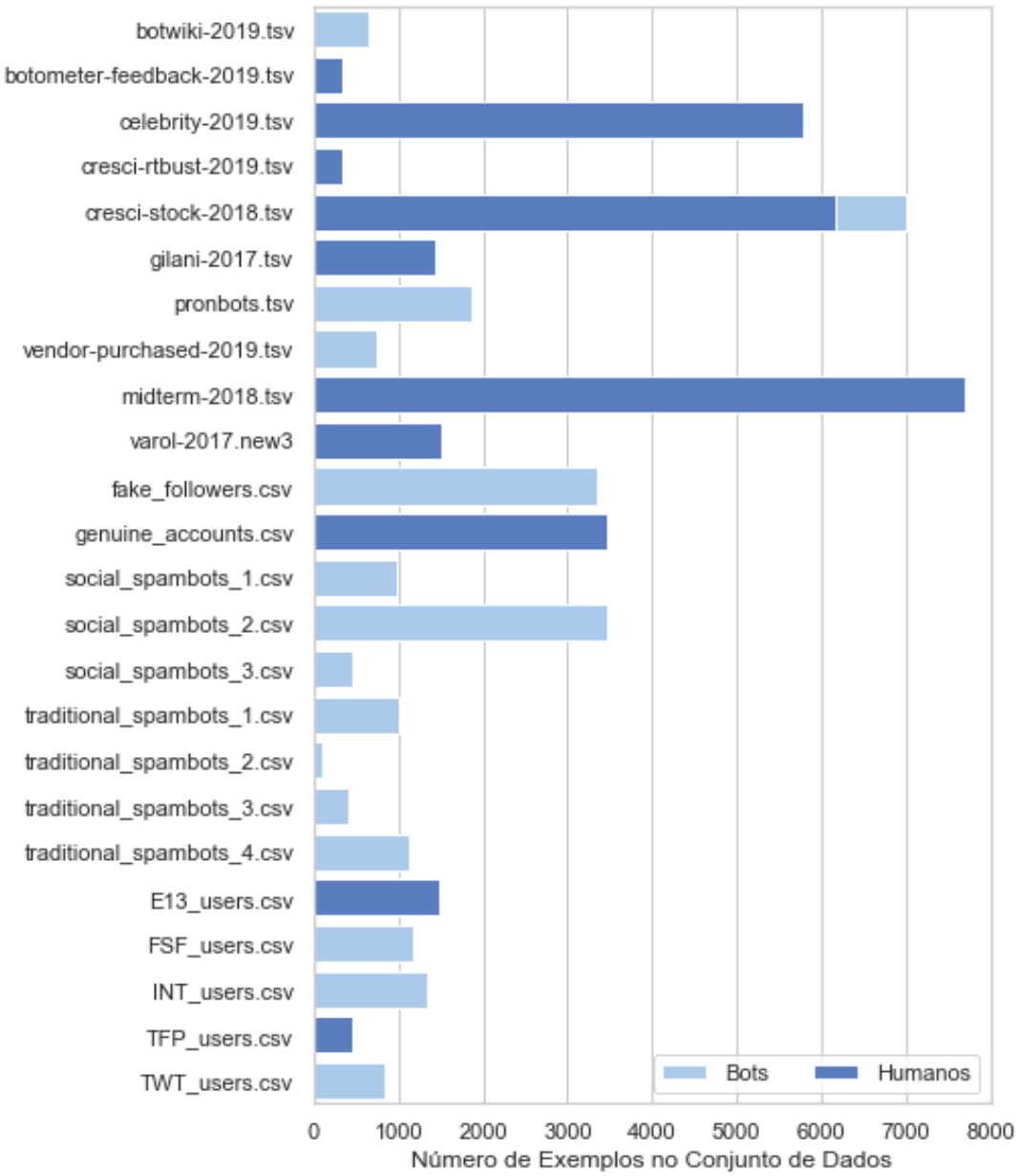


Figura 8 – Exemplos por conjunto de dados

5 Experimentos

No presente capítulo, os experimentos realizados são descritos e seus resultados são apresentados e discutidos.

5.1 Objetivos

Conforme apresentado, neste trabalho é proposta a utilização de uma etapa de pré-processamento utilizando um algoritmo de agrupamento para avaliar o efeito desta etapa no desempenho da classificação na tarefa de detecção de bots.

Devido às diferentes variáveis envolvidas na tarefa, o escopo do presente trabalho se limitou a utilizar apenas um algoritmo de agrupamento (*K-Means*) e um algoritmo de classificação (*Random Forest*), combinando diferentes conjuntos de dados para a execução dos testes. A seleção destes algoritmos, conforme já apresentado, foi realizada com base no bom desempenho destes para diferentes problemas, conforme indicado pela revisão da literatura.

Assim, não foi objetivo do presente trabalho avaliar um modelo específico, mas sim avaliar se existe uma tendência que possa ser observada através de vários conjuntos de dados diferentes. Os experimentos foram realizados diversas vezes, com diferentes conjuntos de dados sendo utilizados para cada execução.

5.2 Forma de avaliação

Para avaliar o experimento, foram considerado os conceitos *in-domain* e *out-domain*.

In-domain são conjuntos de dados que fazem parte do conjunto de treino do algoritmo de classificação. Os dados são divididos aleatoriamente entre treino (75%) e teste (25%), e as métricas observadas nesse tipo teste são entendidas como o desempenho *in-domain*.

Out-domain são os conjuntos de dados que não fazem parte do treino. Dessa forma, o conjunto de dados completo é considerado para o desempenho *out-domain*.

O cenário *out-domain* exemplifica novas distribuições de bots que um detector pode encontrar ao ser utilizado em aplicações reais. Ordinariamente, um desenvolvedor utilizaria todos os conjuntos de dados disponíveis, existindo apenas o desempenho *in-domain*.

5.3 Tipos de pré-processamento avaliados

O pré-processamento pode acontecer de três formas diferentes:

1. Grupos são gerados, cada grupo tem uma *Random Forest* treinada, e apenas o grupo com maior desempenho *in-domain* é selecionado. A *Random Forest* treinada para o melhor grupo é aplicada para detecção de todos os bots.
2. Grupos são gerados, cada grupo tem uma *Random Forest* treinada, e os dados são classificados conforme o modelo treinado para seu respectivo *cluster*. Essa pode ser entendida como a abordagem mais tradicional para uma mistura de especialistas.
3. Grupos são gerados, cada grupo tem uma Random Forest treinada, e apenas o grupo com maior desempenho *in-domain* é selecionado. Dados no *cluster* selecionado usam o modelo do *cluster*, enquanto o restante dos dados é avaliado em uma *Random Forest* treinada com dados de todos os *clusters*.

5.4 Critério para permutação das bases de dados

Foram consideradas aceitáveis para uma execução dos experimentos todas as combinações de conjuntos de dados em que a quantidade de humanos e bots usados para treino estava entre 10% e 50% do total disponível nas bases de dados, de forma que exista uma amostra razoável tanto no *in-domain* quanto no *out-domain*.

5.5 Critério para seleção das variáveis de agrupamento

Entre as variáveis coletadas, utilizamos os atributos numéricos *friends_count* e *followers_count* no algoritmo de agrupamento. O algoritmo *K-Means*, que usa distância euclidiana (JAMES *et al.*, 2013), costuma apresentar resultados mais satisfatórios com variáveis numéricas do que com variáveis categóricas ou binárias. Outros atributos numéricos pode-

riam ter sido utilizados, mas estes dois foram considerados aceitáveis e suficientes após avaliarmos suas distribuições.

Podemos observar nas figuras 9, 10, 11 e 12 que as distribuições são diferentes para cada conjunto de dados, de forma que os grupos gerados em suas combinações tendem a ser diferentes. Foi confirmado, assim, que o experimento reflete uma diferenciação entre os dados *in-domain* e out-domain, conforme esperado.

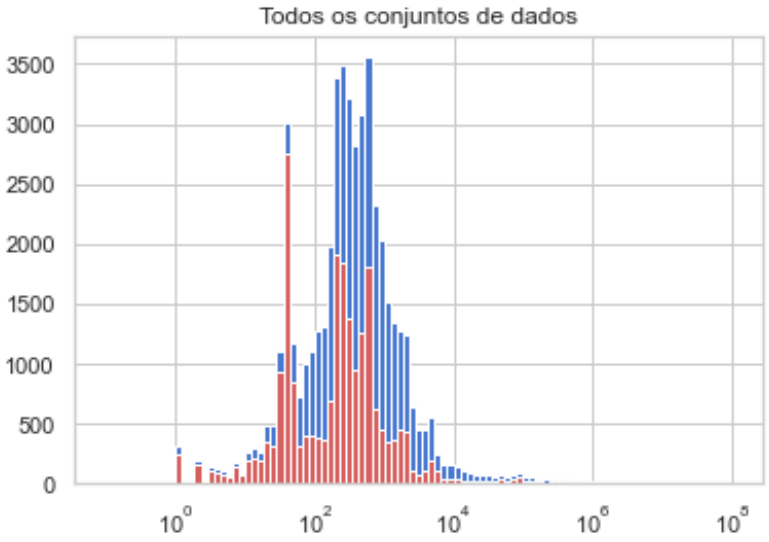


Figura 9 – Histograma do atributo numérico *friends_count*, todos os conjuntos de dados. Bots em vermelho, humanos em azul

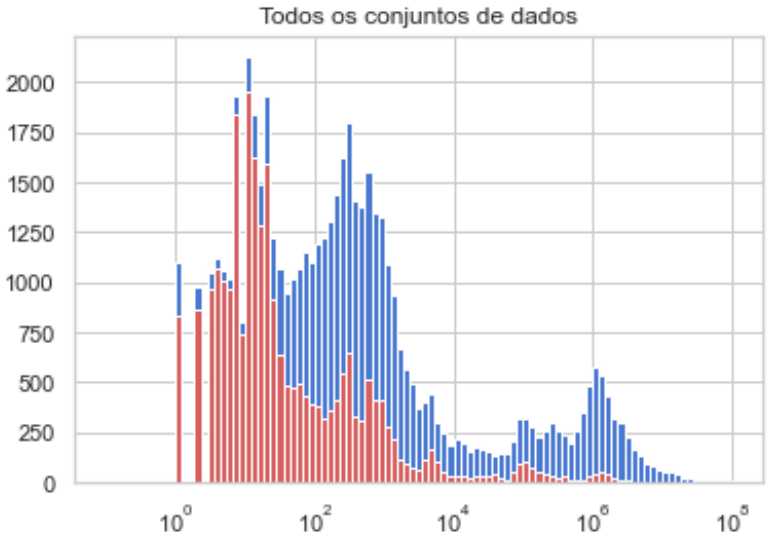


Figura 10 – Histograma do atributo numérico *followers_count*, todos os conjuntos de dados. Bots em vermelho, humanos em azul

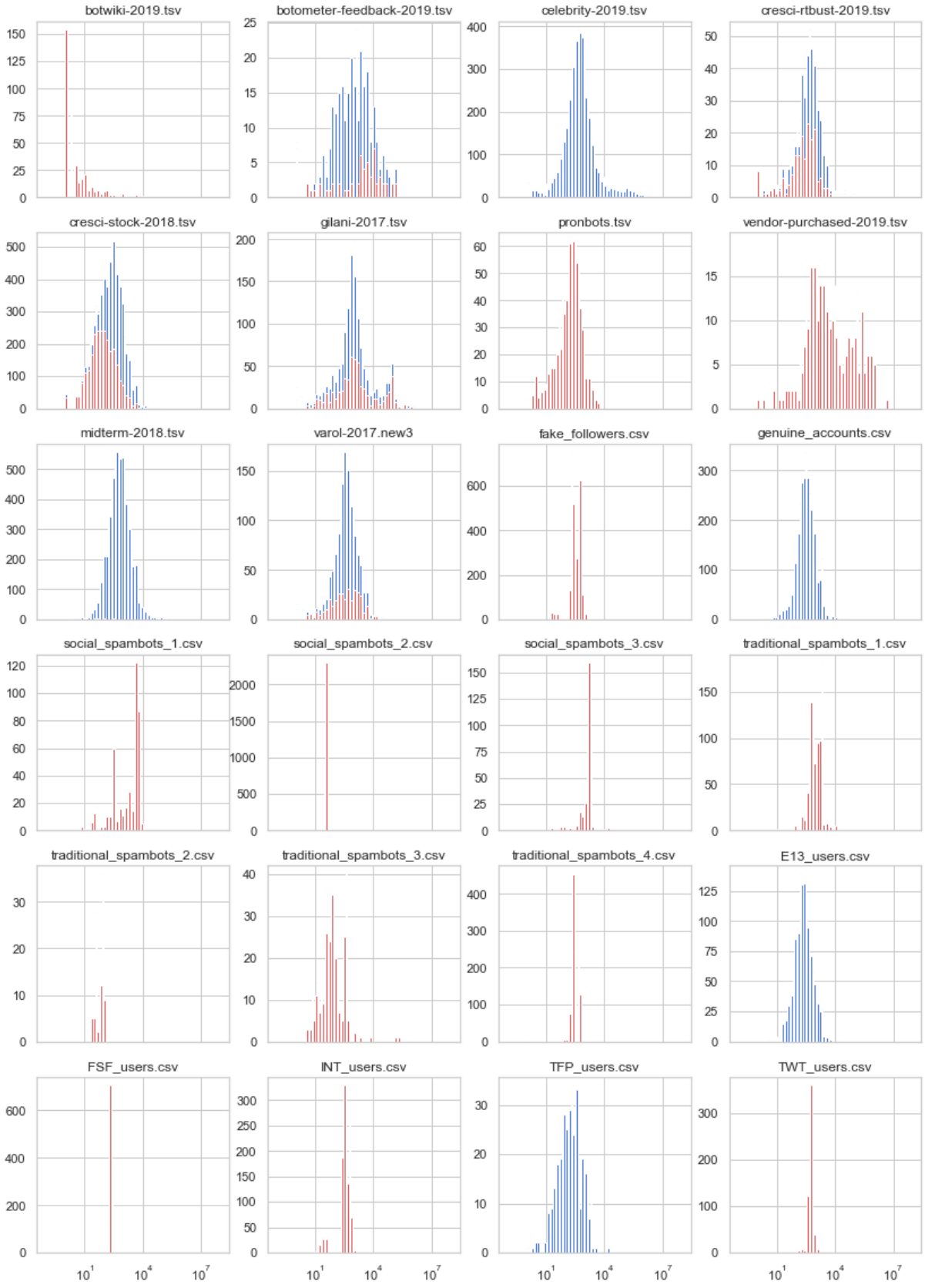


Figura 11 – Histograma do atributo numérico *friends_count*, para cada conjunto de dados. Bots em vermelho, humanos em azul

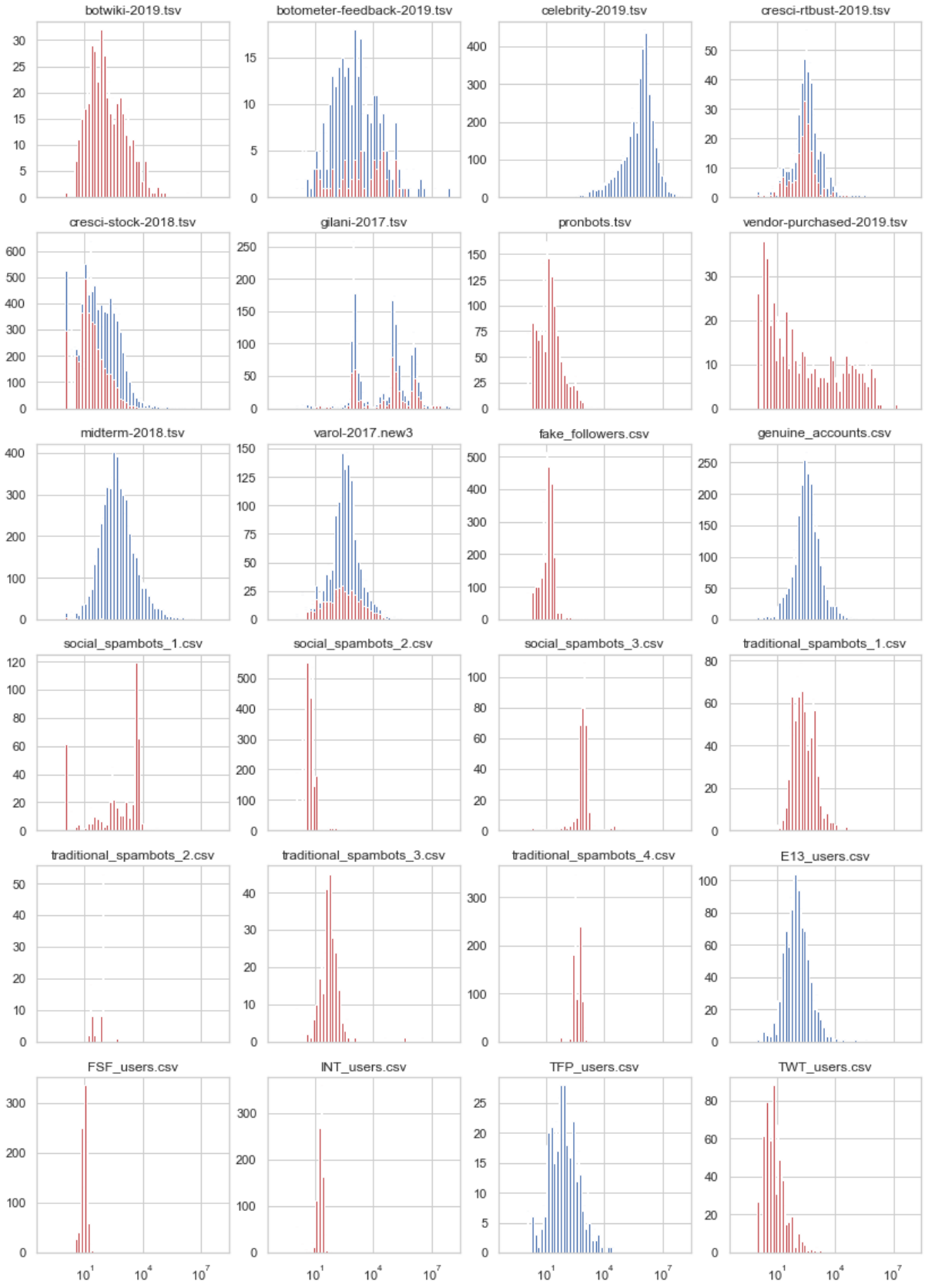


Figura 12 – Histograma do atributo numérico *followers_count*, para cada conjunto de dados. Bots em vermelho, humanos em azul

5.6 Tratamentos de dados realizados

Os atributos utilizados para a classificação, seguindo a nomenclatura da API to Twitter, foram: “statuses_count”, “followers_count”, “friends_count”, “favourites_count”, “listed_count”, “default_profile”, “geo_enabled”, “profile_use_background_image”, “verified” e “protected”. Esses atributos foram recomendados em Ferrara (2017) como suficientes para um bom desempenho de detecção.

Os atributos foram utilizados conforme são retornados diretamente pela API, com apenas o seguinte pré-processamento: Valores *NaN* foram substituídos por 0, e alguns atributos foram submetidos à transformação logarítmica $F(x) = log_{10}(x + 1)$: “statuses_count”, “followers_count”, “friends_count”, “favourites_count” e “listed_count”. Após isso, todos os atributos numéricos forma normalizados para a escala de 0 a 1, com a transformação:

$$Z_i = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

É importante enfatizar que a transformação Z_i foi aplicada usando as estatísticas dos dados de treinamento.

Para os dados *in-domain*, 25% foram selecionados aleatoriamente como teste, e 75% como conjunto de treinamento. Para o classificador *Random Forest*, foram usados os hiper-parâmetros padrões da biblioteca Sci-kit Learn.

5.7 Resultados

Todos os gráficos apresentam resultados médios, agrupados pelo percentil do número de exemplos nos experimentos. Os experimentos estão divididos nos três tipos de pré-processamento avaliados, seguindo a mesma numeração apontada anteriormente. O algoritmo KMeans foi testado com 2 ou 3 grupos, mas não houve diferença significativa. Os gráficos consolidam os resultados das duas tentativas.

Inicialmente foram considerados todos os conjuntos de dados coletados nos conjuntos Cresci como independentes, o que gera uma maior combinação de conjuntos possíveis. Entretanto, isso resulta numa combinação de milhões de modelos, o que não seria viável de se calcular. Todos os experimentos foram então feitos considerando os conjuntos de dados Cresci de forma agrupada, o que resulta em 2.034 modelos por experimento. Os valores

de uma execução parcial do conjunto desagrupado são disponibilizados, e nos permitem observar que as curvas podem variar com as combinações analisadas (Figuras 13, 14), mas o desempenho médio não sofre grande alteração (Quadro 2).

Figura 13 – *Area Under the Curve* (AUC) do modelo *baseline*, de acordo com o número de exemplos (agrupado em 50 percentis) nos conjuntos de dados *in-domain*. Conjuntos Cresci desagrupados.

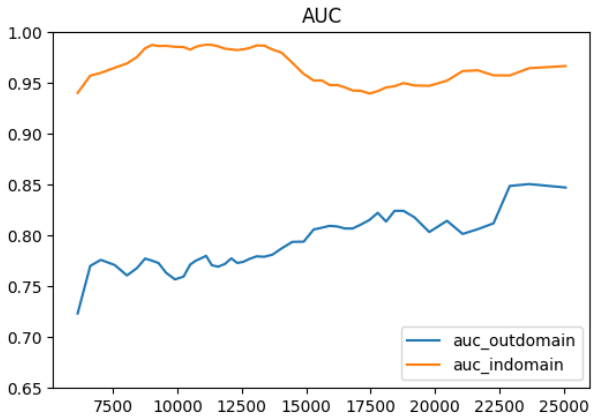
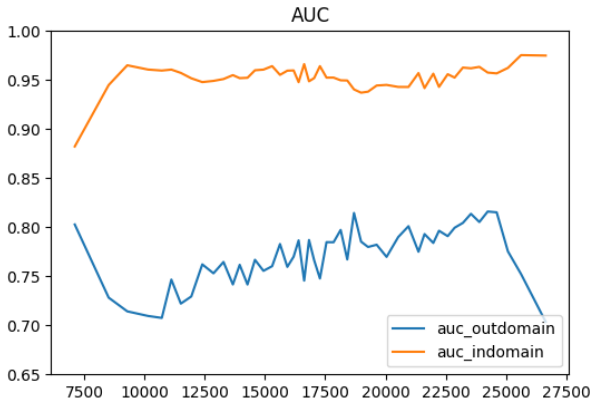


Figura 14 – *Area Under the Curve* (AUC) do modelo *baseline*, de acordo com o número de exemplos (agrupado em 50 percentis) nos conjuntos de dados *in-domain*. Conjuntos Cresci agrupados.



Quadro 2 – Métrica AUC (*Area Under the Curve*) média observada

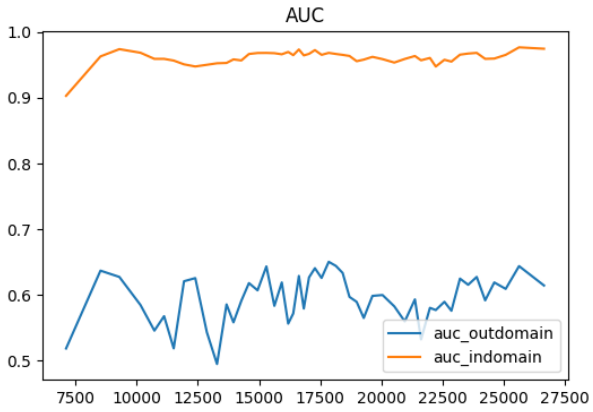
Experimento	**Valor *in-domain***	**Valor *out-domain***
Baseline (conjuntos Cresci desagrupados)	0,9671	0,7913
Baseline (conjuntos Cresci agrupados)	0,9529	0,7698
Experimento 1 (conjuntos Cresci agrupados)	0,9612	0,5948
Experimento 2 (conjuntos Cresci agrupados)	0,9508	0,7672
Experimento 3 (conjuntos Cresci agrupados)	0,9520	0,7712

Nota-se que as extremidades das figuras 13 e 14 aparentam não seguir o mesmo comportamento que o resto dos pontos. Elas representam as combinações com máximo e mínimo número de exemplos, com poucas combinações de conjuntos de dados próximos dessas quantidades. Seu formato é pouco representativo.

5.7.1 Resultados Gerais

Mesmo os menores tamanhos para conjuntos de dados usados como *in-domain* apresentam resultados por volta de 95% para a medida AUC (*Area Under the ROC Curve*). Isso corrobora a observação de que qualquer detector de bots, quando testado no mesmo conjunto de dados, tende a alcançar alto desempenho, algo já observado na literatura (ORABI *et al.*, 2020). Esse comportamento é observado em todos os experimentos, conforme visto nas figuras 15, 16 e 17.

Figura 15 – **Experimento 1**: *Area Under the Curve* (AUC) do modelo usando o melhor cluster para todas as previsões, de acordo com o número de exemplos (agrupado em 50 percentis) nos conjuntos de dados in-domain



Ao mesmo tempo, o desempenho *out-domain* é muito menor, conforme esperado de uma previsão feita num conjunto de dados seguindo uma distribuição diferente do treinamento. Também observa-se que o desempenho *out-domain* tende a crescer com o tamanho do conjunto de dados utilizado no treinamento, conforme esperado de um algoritmo de aprendizado de máquina com acesso a mais dados.

Entretanto, adicionar mais dados nem sempre tem um efeito positivo. Observa-se no gráfico detalhando os conjuntos de dados individuais (Figuras 18, 19, 20, 21, 22) que em alguns casos o desempenho aumenta quando os dados são retirados.

Figura 16 – **Experimento 2**: *Area Under the Curve* (AUC) do modelo usando a mistura de experts para todas as previsões, de acordo com o número de exemplos (agrupado em 50 percentis) nos conjuntos de dados in-domain

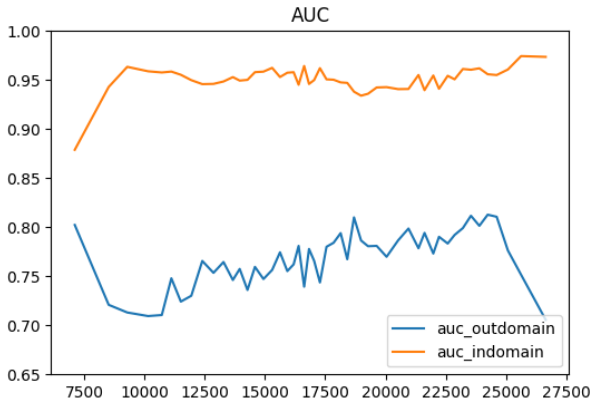
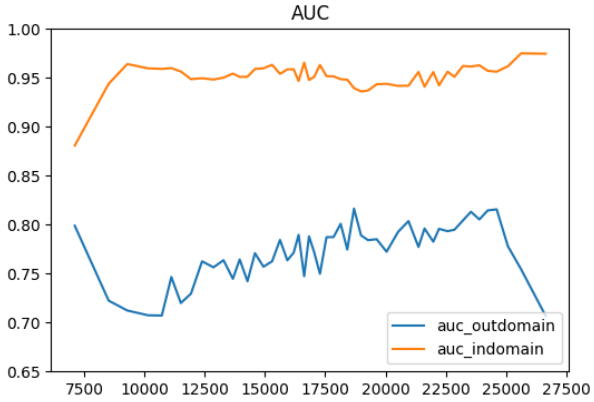


Figura 17 – **Experimento 3**: *Area Under the Curve* (AUC) do modelo usando o melhor cluster em conjunto com o baseline, de acordo com o número de exemplos (agrupado em 50 percentis) nos conjuntos de dados in-domain



Todos os experimentos não apresentaram grandes ganhos na média, mas ao observar combinações individuais de conjuntos de dados notamos que todos os experimentos apresentaram uma parcela com ganho de desempenho (Quadros 3, 4). Não existe uma relação clara entre o desempenho *in-domain* e *out-domain*, como observado nos quadros 5, 6 e 7, tornando difícil encontrar os resultados com melhor desempenho *out-domain* apenas olhando resultados *in-domain*. Separar alguns conjuntos de dados para testes *out-domain* é recomendável.

5.7.2 Experimento 1

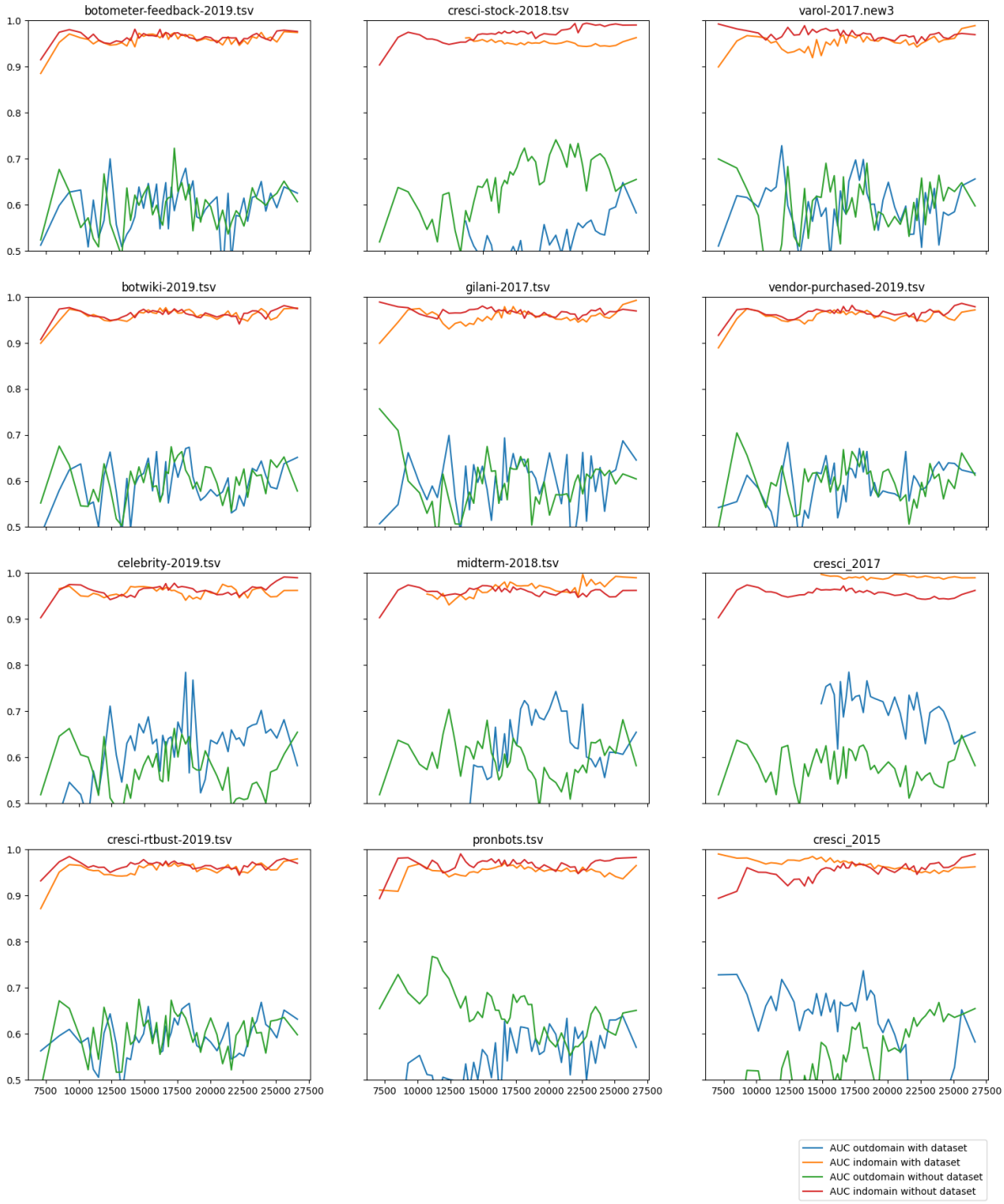
Com um valor médio de desempenho *out-domain* extremamente baixo e perda de desempenho *out-domain* em 85% dos casos, esse pré-processamento tende a não performar

Figura 18 – *Area Under the Curve* (AUC) do modelo baseline, de acordo com o número de exemplos (agrupado em 50 percentis) nos conjuntos de dados *in-domain*. Conjuntos Cresci desagrupados. Visão comparando testes com e sem cada conjunto de dados.

bem. Entretanto, 11% dos casos ainda registraram aumento de desempenho *out-domain* significativo, assim como 44% dos casos no desempenho *in-domain*.

Figura 19 – *Area Under the Curve* (AUC) do modelo baseline, de acordo com o número de exemplos (agrupado em 50 percentis) nos conjuntos de dados *in-domain*. Conjuntos Cresci agrupados. Visão comparando testes com e sem cada conjunto de dados.

Figura 20 – **Experimento 1**: *Area Under the Curve* (AUC) do modelo usando o melhor cluster para todas as previsões, de acordo com o número de exemplos (agrupado em 50 percentis) nos conjuntos de dados *in-domain*. Visão comparando testes com e sem cada conjunto de dados.



5.7.3 Experimento 2

15% dos casos tiveram melhora significativa de desempenho *out-domain*, mas o desempenho *in-domain* sofre perdas muito mais frequentes que no experimento 1, como podemos ver no quadro 4. Essas perdas tendem a ser baixas, de acordo com o quadro 2.

Figura 21 – **Experimento 2**: *Area Under the Curve* (AUC) do modelo usando a mistura de experts para todas as previsões, de acordo com o número de exemplos (agrupado em 50 percentis) nos conjuntos de dados *in-domain*. Visão comparando testes com e sem cada conjunto de dados.

Temos então um pré-processamento que frequentemente envolve perdas pequenas, potencialmente insignificantes, para uma chance de 15% de um ganho (ao menos nos conjuntos testados) de desempenho razoável no *out-domain*.

Figura 22 – **Experimento 3**: *Area Under the Curve* (AUC) do modelo usando o melhor cluster em conjunto com o baseline, de acordo com o número de exemplos (agrupado em 50 percentis) nos conjuntos de dados *in-domain*. Visão comparando testes com e sem cada conjunto de dados.

5.7.4 Experimento 3

Semelhante ao experimento 2, 13% dos casos tiveram melhora significativa de desempenho *out-domain*, mas o desempenho sofre perdas frequentes de baixa intensidade.

Quadro 3 – AUC (Area Under the Curve) do experimento menos AUC *baseline* (out-domain), proporção de experimentos observados em cada grupo

Diferença out-domain	**Experimento 1**	**Experimento 2**	**Experimento 3**
(-1,0, 0,0]	0,8510	0,6071	0,5742
(0,0, 0,001]	0,006	0,0378	0,0476
(0,001, 0,01]	0,027	0,1996	0,2438
(0,01, 1,0]	0,1150	0,1553	0,1342

Quadro 4 – AUC (Area Under the Curve) do experimento menos AUC *baseline* (in-domain), proporção de experimentos observados em cada grupo

Diferença in-domain	**Experimento 1**	**Experimento 2**	**Experimento 3**
(-1,0, 0,0]	0,3092	0,9724	0,9473
(0,0, 0,001]	0,0265	0,0255	0,0511
(0,001, 0,01]	0,2143	0,0019	0,0014
(0,01, 1,0]	0,4498	0,0000	0,0000

Quadro 5 – AUC (Area Under the Curve) do experimento menos AUC baseline, proporção de experimentos observados em cada grupo do **Experimento 1**

	Diferença em AUC out-domain			
Diferença in-domain	(-1,0, 0,0]	(0,0, 0,001]	(0,001, 0,01]	(0,01, 1,0]
(-1,0, 0,0]	0,2507	0,0039	0,0132	0,0412
(0,0, 0,001]	0,0172	0,0000	0,0004	0,0088
(0,001, 0,01]	0,1647	0,0019	0,0078	0,0398
(0,01, 1,0]	0,4183	0,0009	0,0054	0,0250

Seu desempenho médio é maior que o experimento 2, sendo o único experimento que superou a abordagem *baseline* (*out-domain*).

Esse pré-processamento também envolve trocar uma perda pequena no desempenho pela chance de um desempenho significativamente maior no *out-domain*. Adicionalmente, a perda é menor que no experimento 2.

Quadro 6 – AUC (Area Under the Curve) do experimento menos AUC baseline, proporção de experimentos observados em cada grupo do **Experimento 2**

	Diferença em AUC out-domain			
Diferença in-domain	(-1,0, 0,0]	(0,0, 0,001]	(0,001, 0,01]	(0,01, 1,0]
(-1,0, 0,0]	0,5875	0,0373	0,1941	0,1533
(0,0, 0,001]	0,0181	0,0004	0,0049	0,0019
(0,001, 0,01]	0,0014	0,0000	0,0004	0,0000
(0,01, 1,0]	0,0000	0,0000	0,0000	0,0000

Quadro 7 – AUC (Area Under the Curve) do experimento menos AUC baseline, proporção de experimentos observados em cada grupo do **Experimento 3**

	Diferença em AUC out-domain			
Diferença in-domain	(-1,0, 0,0]	(0,0, 0,001]	(0,001, 0,01]	(0,01, 1,0]
(-1,0, 0,0]	0,5462	0,0437	0,2281	0,1293
(0,0, 0,001]	0,0265	0,0039	0,0157	0,0049
(0,001, 0,01]	0,0014	0,0000	0,0000	0,0000
(0,01, 1,0]	0,0000	0,0000	0,0000	0,0000

5.7.5 Consideração sobre uso em sistemas produtivos

Grupos que estejam preocupados com o desempenho de um detector de bots num ambiente de produção (que simulamos neste experimento como o conjunto *out-domain*), mas que não queiram separar conjuntos de dados para uma avaliação *out-domain*, podem preferir usar o método do experimento 3 para tentar melhorar os resultados sem sacrificar as métricas observáveis (*in-domain*).

6 Conclusão

O uso de bots em redes sociais é uma questão de grande interesse nos últimos anos, com diversos agentes usando bots difíceis de distinguir de humanos. Esses bots são usados para propagar informações diversas, podendo manipular o discurso nas redes sociais.

Diversas abordagens de detecção de bots foram propostas, em sua grande maioria utilizando técnicas de aprendizado de máquina supervisionado. Quando observamos a literatura, a maioria dos sistemas de detecção apresenta altas métricas de desempenho. Entretanto, as métricas costumam ser coletadas apenas em uma amostra do conjunto de dados original utilizado também para treinar o detector. Pensando num uso em produção sujeito aos diferentes tipos de bots, essas métricas podem não refletir a realidade.

Este trabalho avaliou algumas formas de incorporar aprendizado não-supervisionado, na forma de três tipos de pré-processamento, utilizando os conceitos de *in-domain* e *out-domain* para simular o desempenho de um detector atuando com tipos de bots desconhecidos no momento do treinamento. Através de experimentos realizados milhares de vezes em diferentes combinações de conjuntos de dados, estabelecemos alguns padrões que podem guiar futuros sistemas.

Os resultados expostos mostram que existe uma grande diferença em desempenho quando avaliamos um sistema no mesmo conjunto de dados utilizado para o treino (*in-domain*), e quando consideramos outros conjuntos de dados (*out-domain*) que também são pertinentes para a tarefa de um detector de bots. O desempenho *in-domain* rapidamente estaciona num alto nível, enquanto o *out-domain* tende a melhorar quando aumentamos o número de exemplos no treinamento.

Novos conjuntos de dados tendem a melhorar o desempenho, conforme esperado, de um sistema de aprendizado supervisionado, mas isso não é uma certeza.

Concluímos que uma avaliação com diferentes conjuntos de dados é justificada para métricas mais realistas. Ao invés de inserir todos os dados possíveis no treinamento do detector, pesquisadores também podem investir algum tempo testando retirar conjuntos de dados, pois existe a possibilidade de uma contribuição negativa no desempenho.

Os tipos de pré-processamento avaliados geram desempenho muito próximo do original na maioria dos casos, com a possibilidade de representar um ganho de desempenho

em alguns casos. Acreditamos que são práticas que podem ser incorporadas no arsenal de testes de um pesquisador sem grande risco.

Parte dos resultados obtidos nesta pesquisa foram publicados (LIRA; XAVIER; DIGIAMPIETRI, 2021). Todos os dados e códigos dos experimentos podem ser acessados via Github[1].

Os experimentos aqui realizados podem ser facilmente estendidos com mais conjuntos de dados, mais variáveis ou diferentes técnicas de agrupamento.

Pode-se sistematizar qual a melhor forma de lidar com a avaliação *out-domain*, de forma análoga a uma validação cruzada, mas com conjuntos de dados diferentes.

Também pode-se investigar como selecionar um pré-processamento que tenha um bom efeito nas métricas e possíveis relações entre as métricas *in-domain* e *out-domain*.

[1] https://github.com/dblop/bot_detector_project

Referências[2]

ABOZINADAH, E. A.; JONES, J. H. A statistical learning approach to detect abusive twitter accounts. In: *Proceedings of the International Conference on Compute and Data Analysis*. New York, NY, USA: Association for Computing Machinery, 2017. (ICCDA '17), p. 6–13. ISBN 9781450352413. Disponível em: ⟨https://doi.org/10.1145/3093241.3093281⟩. Citado na página 27.

ALARIFI, A.; ALSALEH, M.; AL-SALMAN, A. Twitter turing test: Identifying social machines. *Information Sciences*, v. 372, p. 332 – 346, 2016. ISSN 0020-0255. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0020025516306077⟩. Citado 3 vezes nas páginas 27, 28 e 34.

ALOTHALI, E.; ZAKI, N.; MOHAMED, E. A.; ALASHWAL, H. Detecting social bots on twitter: A literature review. In: *2018 International Conference on Innovations in Information Technology (IIT)*. [S.l.: s.n.], 2018. p. 175–180. Citado na página 27.

ANTOUN, W.; BALY, F.; ACHOUR, R.; HUSSEIN, A.; HAJJ, H. State of the art models for fake news detection tasks. In: *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*. [S.l.: s.n.], 2020. p. 519–524. Citado 2 vezes nas páginas 27 e 33.

BELLO, B. S.; HECKEL, R.; MINKU, L. Reverse engineering the behaviour of twitter bots. In: *2018 Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. [S.l.: s.n.], 2018. p. 27–34. Citado 2 vezes nas páginas 27 e 31.

BESKOW, D. M.; CARLEY, K. M. Bot conversations are different: Leveraging network metrics for bot detection in twitter. In: *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. [S.l.: s.n.], 2018. p. 825–832. Citado 3 vezes nas páginas 27, 32 e 36.

BOREGGAH, B.; ALRAZOOQ, A.; AL-RAZGAN, M.; ALSHABIB, H. Analysis of arabic bot behaviors. In: *2018 21st Saudi Computer Society National Computer Conference (NCC)*. [S.l.: s.n.], 2018. p. 1–6. Citado 2 vezes nas páginas 27 e 32.

CHAVOSHI, N.; HAMOONI, H.; MUEEN, A. Debot: Twitter bot detection via warped correlation. In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. Barcelona, Spain: IEEE, 2016. p. 817–822. Citado 3 vezes nas páginas 27, 29 e 35.

CHAVOSHI, N.; MUEEN, A. Model bots, not humans on social media. In: *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. [S.l.: s.n.], 2018. p. 178–185. Citado 2 vezes nas páginas 27 e 33.

CLARK, E. M.; WILLIAMS, J. R.; JONES, C. A.; GALBRAITH, R. A.; DANFORTH, C. M.; DODDS, P. S. Sifting robotic from organic text: A natural language approach for detecting automation on twitter. *Journal of Computational Science*, v. 16, p. 1 – 7, 2016. ISSN 1877-7503. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S1877750315300363⟩. Citado 2 vezes nas páginas 27 e 30.

[2] De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

CORNELISSEN, L. A.; BARNETT, R. J.; SCHOONWINKEL, P.; EICHSTADT, B. D.; MAGODLA, H. B. A network topology approach to bot classification. In: *Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists*. New York, NY, USA: Association for Computing Machinery, 2018. (SAICSIT '18), p. 79–88. ISBN 9781450366472. Disponível em: ⟨https://doi.org/10.1145/3278681.3278692⟩. Citado na página 27.

CRESCI, S.; PIETRO, R. D.; PETROCCHI, M.; SPOGNARDI, A.; TESCONI, M. The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. In: *Proceedings of the 26th international conference on world wide web companion*. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2017. p. 963–972. Citado 2 vezes nas páginas 11 e 15.

CRESCI, S.; PIETRO, R. D.; PETROCCHI, M.; SPOGNARDI, A.; TESCONI, M. Social fingerprinting: Detection of spambot groups through dna-inspired behavioral modeling. *IEEE Transactions on Dependable and Secure Computing*, v. 15, n. 4, p. 561–576, 2018. Citado 3 vezes nas páginas 27, 29 e 34.

CRESCI, S.; PIETRO], R. D.; PETROCCHI, M.; SPOGNARDI, A.; TESCONI, M. Emergent properties, models, and laws of behavioral similarities within groups of twitter users. *Computer Communications*, v. 150, p. 47 – 61, 2020. ISSN 0140-3664. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S014036641930283X⟩. Citado 2 vezes nas páginas 27 e 34.

DORRI, A.; ABADI, M.; DADFARNIA, M. Socialbothunter: Botnet detection in twitter-like social networking services using semi-supervised collective classification. In: *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*. [S.l.: s.n.], 2018. p. 496–503. Citado 2 vezes nas páginas 27 e 36.

ECHEVERRÏà, J.; CRISTOFARO, E. D.; KOURTELLIS, N.; LEONTIADIS, I.; STRINGHINI, G.; ZHOU, S. Lobo: Evaluation of generalization deficiencies in twitter bot classifiers. In: *Proceedings of the 34th Annual Computer Security Applications Conference*. New York, NY, USA: Association for Computing Machinery, 2018. p. 137–146. Citado na página 11.

ECHEVERRIA, J.; CRISTOFARO, E. D.; KOURTELLIS, N.; LEONTIADIS, I.; STRINGHINI, G.; ZHOU, S. Lobo: Evaluation of generalization deficiencies in twitter bot classifiers. In: *Proceedings of the 34th Annual Computer Security Applications Conference*. New York, NY, USA: Association for Computing Machinery, 2018. (ACSAC '18), p. 137–146. ISBN 9781450365697. Disponível em: ⟨https://doi.org/10.1145/3274694.3274738⟩. Citado na página 29.

FAZIL, M.; ABULAISH, M. Identifying active, reactive, and inactive targets of socialbots in twitter. In: *Proceedings of the International Conference on Web Intelligence*. New York, NY, USA: Association for Computing Machinery, 2017. (WI '17), p. 573–580. ISBN 9781450349512. Disponível em: ⟨https://doi.org/10.1145/3106426.3106483⟩. Citado na página 27.

FERNQUIST, J.; KAATI, L.; SCHROEDER, R. Political bots and the swedish general election. In: *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*. [S.l.: s.n.], 2018. p. 124–129. Citado 2 vezes nas páginas 27 e 30.

FERRARA, E. Disinformation and social bot operations in the run up to the 2017 french presidential election. *First Monday*, n. 8, 2017. Citado 2 vezes nas páginas 12 e 49.

FERRARA, E.; VAROL, O.; DAVIS, C.; MENCZER, F.; FLAMMINI, A. The rise of social bots. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 59, n. 7, p. 96–104, jun. 2016. ISSN 0001-0782. Disponível em: ⟨https://doi.org/10.1145/2818717⟩. Citado na página 28.

GILANI, Z.; KOCHMAR, E.; CROWCROFT, J. Classification of twitter accounts into automated agents and human users. In: *2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. [S.l.: s.n.], 2017. p. 489–496. Citado 3 vezes nas páginas 27, 33 e 36.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009. (Springer series in statistics). ISBN 9780387848846. Disponível em: ⟨https://books.google.com.br/books?id=eBSgoAEACAAJ⟩. Citado na página 21.

JAMES, G.; WITTEN, D.; HASTIE, T.; TIBSHIRANI, R. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. Disponível em: ⟨https://faculty.marshall.usc.edu/gareth-james/ISL/⟩. Citado 6 vezes nas páginas 16, 17, 21, 22, 23 e 44.

KANTEPE, M.; GANIZ, M. C. Preprocessing framework for twitter bot detection. In: *2017 International Conference on Computer Science and Engineering (UBMK)*. Antalya, Turkey: IEEE, 2017. p. 630–634. Citado 3 vezes nas páginas 27, 28 e 30.

KHALED, S.; EL-TAZI, N.; MOKHTAR, H. M. O. Detecting fake accounts on social media. In: *2018 IEEE International Conference on Big Data (Big Data)*. Seattle, WA, USA: IEEE, 2018. p. 3672–3681. Citado 3 vezes nas páginas 27, 28 e 34.

KHALIL, H.; KHAN, M. U. S.; ALI, M. Feature selection for unsupervised bot detection. In: *2020 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*. [S.l.: s.n.], 2020. p. 1–7. Citado 2 vezes nas páginas 27 e 35.

KIRAN, K.; MANJUNATHA, C.; HARINI, T. S.; SHENOY, P. D.; VENUGOPAL, K. R. Identification of anomalous users in twitter based on user behaviour using artificial neural networks. In: *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*. [S.l.: s.n.], 2019. p. 1–5. Citado 2 vezes nas páginas 27 e 32.

KITCHENHAM, B. Procedures for Performing Systematic Literature Reviews. *Joint Technical Report, Keele University TR/SE-0401 and NICTA TR-0400011T.1*, v. 33, p. 33, 2004. Citado na página 24.

KUDUGUNTA, S.; FERRARA, E. Deep neural networks for bot detection. *Information Sciences*, v. 467, p. 312 – 322, 2018. ISSN 0020-0255. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0020025518306248⟩. Citado 2 vezes nas páginas 27 e 30.

LIRA, D. B.; XAVIER, F.; DIGIAMPIETRI, L. Combining clustering and classification algorithms for automatic bot detection: a case study on posts about covid-19. In: *SBSI 2021*. Universidade Federal de Uberlândia: [s.n.], 2021. Citado na página 61.

LOYOLA-GONZáLEZ, O.; MONROY, R.; RODRíGUEZ, J.; LóPEZ-CUEVAS, A.; MATA-SáNCHEZ, J. I. Contrast pattern-based classification for bot detection on twitter. *IEEE Access*, v. 7, p. 45800–45817, 2019. Citado 2 vezes nas páginas 27 e 32.

MAZZA, M.; CRESCI, S.; AVVENUTI, M.; QUATTROCIOCCHI, W.; TESCONI, M. Rtbust: Exploiting temporal patterns for botnet detection on twitter. In: *Proceedings of the 10th ACM Conference on Web Science*. New York, NY, USA: Association for Computing Machinery, 2019. (WebSci '19), p. 183–192. ISBN 9781450362023. Disponível em: ⟨https://doi.org/10.1145/3292522.3326015⟩. Citado na página 27.

MINNICH, A.; CHAVOSHI, N.; KOUTRA, D.; MUEEN, A. Botwalk: Efficient adaptive exploration of twitter bot networks. In: *2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. New York, NY, USA: Association for Computing Machinery, 2017. p. 467–474. Citado 3 vezes nas páginas 27, 29 e 36.

MOON, B. Identifying bots in the australian twittersphere. In: *Proceedings of the 8th International Conference on Social Media & Society*. New York, NY, USA: Association for Computing Machinery, 2017. (#SMSociety17). ISBN 9781450348478. Disponível em: ⟨https://doi.org/10.1145/3097286.3097335⟩. Citado na página 27.

MORSTATTER, F.; WU, L.; NAZER, T. H.; CARLEY, K. M.; LIU, H. A new approach to bot detection: Striking the balance between precision and recall. In: *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. [S.l.]: IEEE Press, 2016. (ASONAM '16), p. 533–540. ISBN 9781509028467. Citado na página 27.

MURPHY, K. P. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022. Disponível em: ⟨probml.ai⟩. Citado 8 vezes nas páginas 6, 15, 16, 18, 19, 20, 22 e 23.

ORABI, M.; MOUHEB, D.; AGHBARI], Z. A.; KAMEL, I. Detection of bots in social media: A systematic review. *Information Processing & Management*, v. 57, n. 4, p. 102250, 2020. ISSN 0306-4573. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0306457319313937⟩. Citado 7 vezes nas páginas 12, 24, 26, 27, 28, 40 e 51.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado 2 vezes nas páginas 6 e 23.

PRICE, K. R.; PRIISALU, J.; NOMM, S. Analysis of the impact of poisoned data within twitter classification models. *IFAC-PapersOnLine*, v. 52, n. 19, p. 175 – 180, 2019. ISSN 2405-8963. 14th IFAC Symposium on Analysis, Design, and Evaluation of Human Machine Systems HMS 2019. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S240589631932021X⟩. Citado 3 vezes nas páginas 27, 30 e 36.

RODRíGUEZ-RUIZ, J.; MATA-SáNCHEZ, J. I.; MONROY, R.; LOYOLA-GONZáLEZ, O.; LóPEZ-CUEVAS, A. A one-class classification approach for bot detection on twitter. *Computers & Security*, v. 91, p. 101715, 2020. ISSN 0167-4048. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0167404820300031⟩. Citado 2 vezes nas páginas 27 e 35.

SCHNEBLY, J.; SENGUPTA, S. Random forest twitter bot classifier. In: *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. [S.l.: s.n.], 2019. p. 0506–0512. Citado 3 vezes nas páginas 27, 31 e 36.

SUBRAHMANIAN, V. S.; AZARIA, A.; DURST, S.; KAGAN, V.; GALSTYAN, A.; LERMAN, K.; ZHU, L.; FERRARA, E.; FLAMMINI, A.; MENCZER, F. The darpa twitter bot challenge. *Computer*, v. 49, n. 6, p. 38–46, 2016. Citado na página 27.

VAROL, O.; FERRARA, E.; DAVIS, C. A.; MENCZER, F.; FLAMMINI, A. Online human-bot interactions: Detection, estimation, and characterization. In: *Eleventh international AAAI conference on web and social media*. Montréal, Québec, Canada: The AAAI Press, 2017. p. 280–289. Citado na página 11.

VELAYUTHAM, T.; TIWARI, P. K. Bot identification: Helping analysts for right data in twitter. In: *2017 3rd International Conference on Advances in Computing,Communication Automation (ICACCA) (Fall)*. [S.l.: s.n.], 2017. p. 1–5. Citado 2 vezes nas páginas 27 e 31.

WALT, E. V. D.; Eloff, J. Using machine learning to detect fake identities: Bots vs humans. *IEEE Access*, v. 6, p. 6540–6549, 2018. Citado 2 vezes nas páginas 27 e 33.

WEI, F.; NGUYEN, U. T. Twitter bot detection using bidirectional long short-term memory neural networks and word embeddings. In: *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. [S.l.: s.n.], 2019. p. 101–109. Citado 3 vezes nas páginas 27, 31 e 36.

WU, B.; LIU, L.; YANG, Y.; ZHENG, K.; WANG, X. Using improved conditional generative adversarial networks to detect social bots on twitter. *IEEE Access*, v. 8, p. 36664–36680, 2020. Citado 2 vezes nas páginas 27 e 32.

YANG, K.-C.; VAROL, O.; HUI, P.-M.; MENCZER, F. Scalable and generalizable social bot detection through data selection. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. New York, NY, USA: AAAI Press, 2020. p. 1096–1103. Citado 3 vezes nas páginas 11, 12 e 29.

ZHANG, X.; LI, Z.; ZHU, S.; LIANG, W. Detecting spam and promoting campaigns in twitter. *ACM Trans. Web*, Association for Computing Machinery, New York, NY, USA, v. 10, n. 1, fev. 2016. ISSN 1559-1131. Disponível em: ⟨https://doi.org/10.1145/2846102⟩. Citado na página 27.