# BotHunter: An Approach to Detect Software Bots in GitHub

Ahmad Abdellatif
Concordia University
Montreal, Canada
ahmad.abdellatif@concordia.ca

Mairieli Wessel
Delft University of Technology
Delft, Netherlands
m.wessel@tudelft.nl

Igor Steinmacher
Universidade Tecnológica Federal do Paraná
Campo Mourão, Brazil
igorfs@utfpr.edu.br

Marco A. Gerosa
Northern Arizona University
Flagstaff, USA
marco.gerosa@nau.edu

Emad Shihab
Concordia University
Montreal, Canada
emad.shihab@concordia.ca

## ABSTRACT

Bots have become popular in software projects as they play critical roles, from running tests to fixing bugs/vulnerabilities. However, the large number of software bots adds extra effort to practitioners and researchers to distinguish human accounts from bot accounts to avoid bias in data-driven studies. Researchers developed several approaches to identify bots at specific activity levels (issue/pull request or commit), considering a single repository and disregarding features that showed to be effective in other domains. To address this gap, we propose using a machine learning-based approach to identify the bot accounts regardless of their activity level. We selected and extracted 19 features related to the account's profile information, activities, and comment similarity. Then, we evaluated the performance of five machine learning classifiers using a dataset that has more than 5,000 GitHub accounts. Our results show that the Random Forest classifier performs the best, with an F1-score of 92.4% and AUC of 98.7%. Furthermore, the account profile information (e.g., account login) contains the most relevant features to identify the account type. Finally, we compare the performance of our Random Forest classifier to the state-of-the-art approaches, and our results show that our model outperforms the state-of-the-art techniques in identifying the account type regardless of their activity level.

**ACM Reference Format:**
Ahmad Abdellatif, Mairieli Wessel, Igor Steinmacher, Marco A. Gerosa, and Emad Shihab. 2022. BotHunter: An Approach to Detect Software Bots in GitHub. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Software Engineering (SE) bots are a game-changer in software development where they serve as a conduit that connects software developers and services [44, 46] and provide additional value on top of services' basic capabilities [29]. Bots assist practitioners in automating tedious and repetitive tasks in software projects. For example, practitioners use bots to fix bugs [34], refactor source code [50], recommend tools [10], and update out-of-date dependencies [33]. Automating tedious tasks allows the project maintainers to focus on the core and critical tasks related to the project. Interestingly, bots produce more activities (e.g., commits or issues/pull requests) than humans in some software projects [17] and change how developers communicate and deal with pull requests [47].

Due to their ability to save resources and increase software development velocity [2, 3, 44, 46], bots are becoming more prevalent in the SE environments [46] and getting progressively harder to identify, biasing data-driven research and impacting practitioners [24]. Therefore, the research community needs to be able to isolate bots and their activities in empirical studies [16, 28, 30, 41]. For example, Dey et al. [16] find that bots create the majority of issues and pull requests in the examined repositories, which might bias their results and conclusions. Developers are also looking for ways to identify bots on GitHub and ask questions on Q&A websites [26, 52].

Although several approaches have been proposed to detect bots on social coding platforms [17, 19, 24, 25], they are limited to detecting bots in specific development activities. For example, Dey et al.'s approach [17] detects software bots that commit code, while Golzadeha et al.'s approach [24] detects bots that comment on issues and pull requests. In other words, none of these approaches is generalized to detect bots that commit code and operate on issues and pull requests. Consequently, practitioners need to run more than one approach to identifying all bots in their repositories.

In this paper, we present an approach to detect software bots in multiple activity levels (i.e., commit and issue bots) on GitHub. Our approach leverages different account activities that occur in multiple repositories to train a machine learning classifier to identify the account type (i.e., human or bot). More specifically, we extract 19 features that can be grouped into three dimensions: (1) profile information related to GitHub user (e.g., account login), (2) account activity related to different events performed by the user (e.g., the total number of activities), and (3) text similarity, which is related to the text similarity on issue comments and commit messages. We extract the 19 features from 5,107 GitHub accounts. Finally, we train five machine learning classifiers to identify the account type. To evaluate the effectiveness of the techniques, we conduct a study to answer the following research questions:


*Conference'17, July 2017, Washington, DC, USA*

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

- **RQ1: Which classifier performs best at detecting bots?** We built five machine learning classifiers: Logistic Regression, K-Nearest Neighbors, Decision Tree, Random Forest, and Support-Vector Machine, using 19 features, such as account login and median response time. Then, we evaluated the classifiers' performance on a dataset containing human and bot (issue and commit bots) accounts. The results show that the Random Forest classifier accurately predicts the account type with an F1-score of 92.4% and AUC of 98.3%.
- **RQ2: What features are the best indicators of bot accounts?** We studied the most relevant features that contribute towards the account type prediction to understand the characteristics that differentiate bots from humans on GitHub. The features related to the account profile (e.g., account login, account name) are the best indicators of the account type on GitHub. In addition, the 'similarity of the issue/PR comments' and 'median activity per day' features are important factors in identifying the account type.
- **RQ3: How effective is our approach compared to the state-of-the-art?** We compared the best classifier's performance from RQ1 (i.e., Random Forest) with the performance of the state-of-the-art approaches. The results show that the Random Forest classifier outperforms the state-of-the-art approaches in identifying the account type on GitHub.

Based on our results, we provide a set of implications for researchers, practitioners, and social coding platforms. To help practitioners identify the user account type on GitHub, we develop a tool using our approach, called BotHunter, and make it publicly available [5]. Finally, we share all of our datasets [4] to accelerate future research in the area.

**Paper Organization.** The remainder of the paper is organized as follows. Section 2 presents the work related to our study. We detail our approach to identify the account type and the study setup in Section 3. Section 4 presents the results. We discuss our findings in Section 5 and their implications in Section 6. Section 7 discusses the threats to validity, and Section 8 concludes the paper.

## 2 RELATED WORK

In this section, we discuss the work most related to our study. We divide the prior work into two main areas: (i) work related to detecting social media bots and (ii) work that identifies bots on GitHub.

### 2.1 Identifying Social Media Bots

Some studies focus on identifying software bots on social media [15, 20, 32, 36, 37, 40]. For example, Santia et al. [40] used text and timestamp related features to detect bots on Facebook. Efthimion et al. [20] focused on Twitter bots and used features related to user profile (e.g., has less than 30 followers), text analysis (e.g., Levenshtein distance between user's tweets is less than 30), and account activity (e.g., Tweeting time) to train supervised machine learning algorithms. Polignano et al. [36] trained a convolutional neural network on the tweet content to identify whether a Twitter account type is a bot or human. Davis et al. [15] trained a Random Forest model using features related to network (e.g., centrality), account (e.g., creation time), friends (e.g., number of followers), time (e.g., inter-tweet time distribution), tweet content, and sentiment. The authors evaluated their model on 31,000 Twitter accounts and achieved an AUC of 0.95. These studies show the importance of using the user account information (e.g., number of followers) and activities to predict the account type. Moreover, they inspired us to include such features to detect bots in social coding platforms.

### 2.2 GitHub Bots

Software Engineering bots assist developers in their daily tasks such as generating bug fixes [12, 42, 45], answering developers' technical questions [2, 39], and performing code refactoring [51]. Recently, researchers proposed approaches that work on the comment level to identify whether a comment is generated by a human or bot [13, 23]. For example, Cassee et al. [13] evaluated three machine learning classifiers to identify the pull-request and issue comments posted by bots on GitHub. The work closest to ours is the work that proposes approaches to identify the type (bot or human) at the account level on social coding platform (e.g., GitHub) [17, 23–25]. For example, Golzadeh et al. [25] proposed an approach that identifies the account type based on the text similarity between the comments issued by the account. This is based on the idea that bots perform repetitive tasks, which yield to generate repetitive comments. Also, Golzadeh et al. [24] proposed BoDeGHa, a machine learning-based approach to detect software bots that comment on issues and pull requests on GitHub based on comments-related features (e.g., repetitive comment patterns). The authors evaluated the proposed approach on 5,000 GitHub accounts. The results show that BoDeGHa achieved an F1 score of 0.98. On the other hand, Dey et al. [17] developed BIMAN to identify bots that commit code using commits meta-data (e.g., files modified by the commit) and achieved an AUC-ROC of 0.9.

The previous studies focused on detecting bots that operate on a single activity level (i.e., issues/pull requests or commits) on the GitHub repository using certain types of features (e.g., text-based features). Our work differs and complements the prior work in different ways. First, our approach identifies bots regardless of their activity level. Second, having an overview of various types of account activities (e.g., create a pull request and commit a code), rather than a single type of activity, helps to better identify the account type. Third, we leverage a rich and comprehensive set of features (e.g., profile information features) to identify the user account type on GitHub. Finally, our approach provides a consistent account type across all repositories, enabling us to build a benchmark of the bot accounts on GitHub, which can be used to advance future research in the field.

## 3 STUDY SETUP

The main goal of our study is to detect bots regardless of their activity levels (e.g., commit bot) on GitHub. To achieve this goal, we propose the use of machine learning techniques. Therefore, we start by examining the different users' activities on GitHub. More specifically, we inspect the code commits, issues, and pull requests activities performed by a user in other repositories to identify the features that could reveal the user account type. Then, we extract the selected features from different types (e.g., human) of GitHub accounts used in similar work and use them to evaluate our machine

**Table 1: An overview of features used to identify account type in GitHub.**

| Dimensions | Features | Definition |
|---|---|---|
| **Profile Information** | Account login | The primary identification of an account. |
| | Account name | The name of an account on GitHub. |
| | Account bio | The short bio description of an account. |
| | Number of followings | The total number of users that an account is following. |
| | Number of followers | The total number of users is following the account. |
| | Account tag | Used to tag GitHub applications as "*bot*". |
| **Account Activity** | Total number of repository activities | Number of activities performed by an account on at the repository level (e.g., repository fork). |
| | Total number of issue activities | Number of issue activities an account has performed (e.g., close an issue). |
| | Total number of pull request activities | Number of pull request activities performed by an account (e.g., assign a reviewer). |
| | Total number of commit activities | Number of commit activities performed by an account (e.g., push code). |
| | Unique repository activities | Different repository activities performed by the account. |
| | Unique issue activities | Unique issue activities performed by an account. |
| | Unique pull request activities | Distinct pull request activities an account has performed. |
| | Unique commit activities | Unique commit activities an account has executed. |
| | Median activity per day | Median number of activities performed per day. |
| | Median response time | Median response time to the earliest event in issue or pull request. |
| **Text Similarity** | Issue/Pull request comments | Issue and pull request comments similarity created by an account. |
| | Preceding comments | Text similarity of comments that precede bot events (e.g., @dependabot merge) on issues and pull requests. |
| | Commit messages | Commit messages similarity made by an account. |

learning classifiers. In this section, we describe the features used to classify the account type, present our dataset, and detail the machine learning classifiers and evaluation process.

## 3.1 Feature Selection

To generate a comprehensive behavior profile for a given GitHub account, we need to have a large collection of features that captures different aspects of user characteristics and activities. By covering various features, we reduce bias and expand the quantity and types of bots that we can identify. To achieve this, first, we randomly examine 20 popular software (i.e., non-educational) projects on GitHub [4] in terms of the number of stars. More specifically, the first two authors independently inspected each of the accounts' profile and their activity history (e.g., pull request comments and commits meta-data) in those repositories looking for features that help to reveal the account type. We found 23 bot accounts [4] in the examined repositories. Finally, the first two authors discussed the features until they reached a consensus. At the end of this process, we selected 19 features grouped into three dimensions: profile information, account activities, and text similarity. Table 1 presents an overview of the dimensions and their associated features. Next, we discuss those dimensions and their features in more detail.

**Profile information.** Every user on GitHub has an account profile that contains 1) personal information such as name, login, and a short bio; and 2) social information, such as the followings and followers accounts by other GitHub users. We hypothesize that such information might contribute to distinguishing bot from human accounts. We considered six features in this dimension:

*Account login:* The account login (i.e., username) is the primary identifier of an account on GitHub. Figure 1 shows the profile information of a bot account, called 'WhiteSource Renovate', on GitHub. As shown in the figure, the account login of the 'WhiteSource Renovate' is *renovate-bot*. During the manual inspection of the GitHub profiles for the examined accounts, we noticed that the bot accounts are likely to contain the 'bot' substring in their logins. In particular, 48% of the bot accounts have 'bot' in their logins such as *traviscibot*, *dependabot*, and *pyup-bot*.

*Account name:* User accounts might also provide a profile name to identify themselves (e.g., 'WhiteSource Renovate' in Figure 1) thereby complementing their profile information. Similar to the account login, bot accounts tend to have the 'bot' substring in their profile name. Among the inspected accounts, we find 40% of them have 'bot' in their names (e.g., *CamperBot* and *pyup.io bot*).

*Account bio:* Users on GitHub can write a short bio on their profiles to describe their interests and skills. For example, the 'WhiteSource Renovate' account describes itself as "A bot to keep the dependencies updated in the PRs" in its bio, as shown in Figure 1. Also, the 'crowndin bot' account describes itself as "*Just a robot*" in its bio. We notice that 35% of the bot accounts explicitly declare themselves bots in their bios.

*Number of followings:* Users follow other accounts on GitHub to receive activity updates, learn from them, and socializing [9]. We believe that these social activities are more related to humans than bots. In other words, human accounts are more likely to follow other account activities to stay aware of their most recent activities on GitHub. Therefore, we identify the number of following accounts as an indicator of human social activity. We find that human accounts have on median eight followings. On the other hand, bot accounts are unlikely to have followings. The number of followings of the inspected bot accounts ranges from 0 to 1, with a median of 0 followings. For example, the 'WhiteSource Renovate' bot is not following any account (i.e., the number of followings is 0) on GitHub, as shown in Figure 1.

*Number of followers:* In addition to the number of followings, we also identify the total number of followers of an account as another indicator of human social activity. Among the examined accounts, human accounts have on median 36 followers, while the bot accounts have on median one follower. However, there are nine bots that have a high number of followers because they are well-known bots in the community (e.g. *Dependabot*, *Greenkeeper*, *renovate-bot*) or highly active on large projects (e.g. *Camperbot*) which increased their popularity.

*Account tag:* Bot developers can build their bots and share them as GitHub Apps to be easily integrated with the repositories. Developers usually tag the app as a 'bot' on the GitHub platform. Thus, we examine whether an account is used by a GitHub App[1] and found that 43% of the bot accounts were tagged as 'bot' from the examined accounts.

**Account Activity.** Some bots cannot be captured using the profile information features only since these bots do not have the specified features, e.g., bots that do not have the word 'bot' in their bios. Thus, we explore the features related to the activities performed by bot accounts on GitHub. More specifically, we examine the type and number of account activities in this dimension. Features that characterize the activities of an account shed light on the amount of effort and time the human or bot puts in when accomplishing tasks. Therefore, those features might contribute toward revealing the account type on GitHub. In the following, we detail the activities-related features.

*Total number of activities:* Bots and humans are likely to have different activity patterns since the bot tasks are strict, and their actions are previously defined. Therefore, as a proxy for activity level, we calculate the total number of account activities, grouping them into four different groups of activities: 1) activities related to repositories (total number of repository activities) such as creating a new repository, fork an existing repository, or watch a repository to be notified in case there are updates; 2) activities related to issues (total number of issue activities), including opening, closing, labeling, and posting comments on issues; 3) activities related to pull requests (total number of pull request activities), including opening, closing, labeling, and posting review comments on pull requests; and 4) activities related to commits (total number of commit activities) such as pushing one or more commits to a repository.

*Unique activities:* Users perform different events related to an activity on GitHub. For example, users can comment on an issue and assign a developer to fix it. Both events are related to the issue activities. In contrast to human accounts, most bots on GitHub perform a single type of activity (e.g., measure the code coverage) [46]. Thus, we consider the unique number of activities performed by an account to indicate the account activity level. Therefore, we compute the number of unique activities for an account, grouping them into four different groups: 1) repository (e.g., creating branches); 2) issue (e.g., label an issue); 3) pull request (e.g., assign a reviewer to the pull request); and 4) commit (e.g., push a commit).

*Median activity per day:* Bots efficiently automate repetitive tasks, saving development cost, time, and effort [44]. During our manual analysis, we noticed that bots are more likely to have a higher daily activity level than humans. This is expected because there are limits to how rapidly and often humans can perform tasks. For example, *openapi-sdkautomation*, *renovate-bot*, and *codecov-io* bots perform on median 300 activities per day. Therefore, we compute an account's median number of daily activities (e.g., comments on issues).

[1]https://docs.github.com/en/developers/apps/about-apps

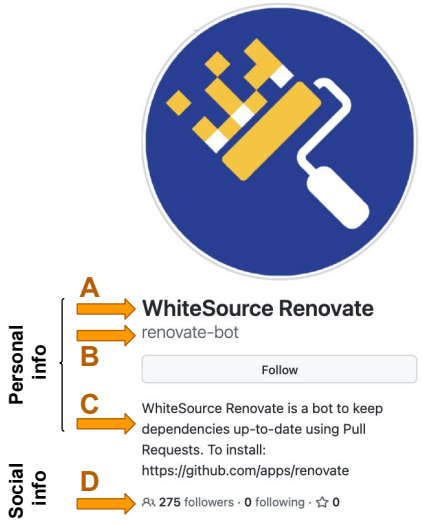


**Figure 1: Example of *renovate-bot* profile on GitHub.**

*Median response time:* In addition to the high level of activity per day, we also notice that bots are likely to respond quickly to an event compared to human accounts. This is because bots are tools that constantly monitor the repository, and they are triggered based on a predefined action (e.g., issue or pull request creation). Therefore, we consider the earliest event of an issue or pull request timeline to indicate the account type. In particular, we compute the median time an account takes to respond to an event where an issue or pull request are created, regardless of whether the account is mentioned or not.

**Text similarity.** Since bots automate tedious tasks on GitHub [23, 46], we expect bots' comments to have a higher text similarity compared to humans. Some bots post comments after performing an action to indicate whether they act on something or report the action results (e.g., display the test code coverage). For example, *CamperBot* adds the following description "This PR was opened auto-magically by Crowdin" when it creates a new pull request on the repository. Next, we describe the three features related to the text similarity.

*Issue/Pull request comments similarity:* Features quantifying the repetition patterns on the accounts' comments have been shown to be effective in identifying bots that work on issues and pull requests [24, 25]. To measure the similarity between comments made by an account, we rely on the cosine similarity metric (i.e., $similarity(A,B) = \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$), as used by prior work [2, 6, 53]. To compute the cosine similarity for an account on GitHub, we first compute the text similarity between all pairs of issue and pull request comments made by the account on each repository. Then, we calculate the median of comment similarities across the different repositories the accounts have contributed, as the predefined bot's

**Table 2: The GitHub accounts distribution in the oracle.**

| Account Type | Count (%) | Repositories |
|---|---|---|
| **Humans** | 4,428 (86.7%) | 3,489 |
| **Issue/PR bots** | 455 (9%) | 444 |
| **Commit bots** | 224 (4.3%) | 256,775 |

response might be different across different repositories. We use Scikit-learn, which is a Python machine learning library [35], to calculate the cosine similarity of the comments.

*Preceding comment similarity:* Some bots that perform activities on pull requests and issues are triggered by specific comment patterns, commands, tags, or combinations. For example, a user could close a pull request by adding a comment for the *dependabot* such as "*@dependabot close*", which triggers *dependabot* to close the pull request. We hypothesize that these trigger comments help identify the bots as users add similar comments to ask the bot to act. Similar to the *Issue/Pull request comments similarity* feature, for each repository, we measure the similarity between comments that occurred before the targeted account action (e.g., close a PR) using the cosine similarity. Then, we compute the median of comments similarities across the different repositories.

*Commit messages similarity:* Similar to the *Issue/Pull request comments similarity*, code commit bots tend to have predefined commit messages. One example is the *nextcloud-bot* that has the following commit message pattern: "*[tx-robot] updated from transifex*". We perform the same process as in the *Issue/Pull request comments similarity* feature on the commit messages to calculate the median of the commit messages similarity for the target account.

## 3.2 Dataset

To evaluate our approach in identifying different types of bots (i.e., commit and/or issue bots) in GitHub, we select two representative datasets composed of bots that perform issue/pull request [24] and commits [17] related activities. Golzadeh et al. [24] randomly selected 136K GitHub repositories of 37 software package registries (e.g., PyPI). Then, they excluded the accounts with less than ten comments, resulting in 79,342 GitHub accounts. Finally, the authors randomly selected 5,082 accounts to manually annotate them based on the issues/pull requests comments. The final dataset contains 5,000 GitHub accounts, where 4,473 are human accounts and 527 are bot accounts.

Dey et al. [17] shared a dataset of bots and their commits used to evaluate their approach (BIMAN). Their dataset is composed of 461 bot accounts with 13,762,430 commits. While this dataset contains commit meta-data, account names, and e-mail addresses, it lacks the GitHub account logins. Our approach requires the accounts' login to collect their activities on GitHub, as discussed in Section 3.1. Therefore, we extended the dataset by leveraging the GitHub API to retrieve the logins for these accounts based on the provided e-mail addresses. We found the GitHub logins for 300 bot accounts.

For both datasets (issue/pull request and commit), we filter out inactive accounts (those with no activities since 2017). We were left with 5,107 GitHub accounts, which composed our oracle. Table 2 presents the distribution of the GitHub account types and the number of repositories that these accounts have contributed to. Most accounts (86.7%) in our oracle are of human accounts, while bots compose 13.3% distributed across issue/pull requests and commit bots. Moreover, we notice that bots contribute to more software projects than humans in our dataset. This evidence the need for a general approach to detect bots by leveraging the activities from different repositories to provide a consistent classification of the account type on GitHub rather than using the activities in a single repository which might lead to incorrect and inconsistent classification.

## 3.3 Performance Evaluation

To perform our predictions, we leverage five machine learning classifiers: Logistic Regression, K-Nearest Neighbors, Decision Tree, Random Forest, and Support-Vector Machine. These classifiers have different assumptions about the analyzed data and can deal with overfitting [11]. Moreover, they have been commonly used by software engineering work [1, 8, 17, 24, 27]. To measure the performance of the classifiers, we compute the precision, recall, and F1 score. In our study, precision is the percentage of correctly classified accounts type relative to the total number of classified accounts (i.e., $Precision = \frac{TP}{TP+FP}$). Recall is the percentage of the correctly classified account type to the total number of accounts in the oracle (i.e., $Recall = \frac{TP}{TP+FN}$). We then combine the precision and recall using the F1-score (i.e., $F1\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$) to have an overall performance for each classifier. As discussed in Section 3.2, our data is highly skewed since the majority of accounts are human accounts. Therefore, we exploit Area Under the ROC Curve (AUC) to measure the ability of the classifiers to discriminate between account types. The AUC is computed by measuring the area under the curve that plots the true positive rate against the false-positive rate while varying the threshold used to determine whether an account is human or bot. The value of AUC ranges between 0-1, and a larger value of AUC indicates better classification performance.

# 4 RESULTS

In this section, we present the results of our study. We describe our motivation, methodology to answer the question, and results for each research question.

## 4.1 RQ1: Which classifier performs best at detecting bots?

**Motivation:** Many Open Source Software (OSS) projects on GitHub use software bots to automate tedious tasks [46]. This implies that software practitioners should consider their presence while maintaining their projects or mining software repositories data (e.g., exclude bots [16, 30, 41]). The goal of this research question is to evaluate the classifiers described in Section 3.3 to determine which classifier leads to the best results in identifying the account type. This helps the SE practitioners to have an accurate prediction of the account type that acts in their software project so they can manage those accounts based on their types.

**Approach:** To evaluate the performance of the classifiers, first, we extract the features for all GitHub accounts (5,107 accounts) in our oracle using the GitHub API. Next, we check the highly correlated features to avoid the multicollinearity problem checking

**Table 3: Performance of the classifiers.**

| Classifier | Precision (%) | Recall (%) | F1-score (%) | AUC (%) |
|---|---|---|---|---|
| **Logistic Regression** | 95.3 | 84.2 | 89.4 | 97.6 |
| **K-Nearest Neighbors** | 84.1 | 60.1 | 70.1 | 89.4 |
| **Decision Tree** | 88.4 | 86.6 | 87.5 | 93.0 |
| **Random Forest** | **95.7** | **89.4** | **92.4** | **98.3** |
| **Support-Vector Machine** | 90.5 | 82.8 | 86.5 | 92.4 |

whether two independent variables have a linear relationship. More specifically, we perform pairwise Pearson correlation between the features and remove those correlated with over > 70% to any other feature. The only features that are highly correlated are "unique pull-request activities" and "unique repository activity". Then, we use the collected data to evaluate the classifiers' performance in predicting the account type. In particular, we use stratified 10-folds-cross-validation. We randomly divide our oracle into ten equal folds where each fold maintains a consistent distribution of account types as the one in the oracle. Next, we train the classifier on nine folds and use the remaining fold to evaluate the classifier's performance. This process is repeated ten times for each classifier, and the average performance of these runs presents the overall performance for that classifier. It is important to note that we use the same folds to train and test all classifiers. Moreover, we use the CountVectorizer (word count) to encode the features used to train the machine learning models. In our work, we implement all models using the default parameter settings on top of scikit-learn [35].

**Results:** Table 3 presents the precision, recall, F1-score, and AUC values for all classifiers. Overall, the classifiers achieve high performance (F1-score > 80% and AUC > 89%) in identifying the account type on GitHub. Random Forest stands out as the best classifier, achieving the highest F1-score (92.4%) and AUC (98.3%).

After evaluating the performance of our approach on different types of bots (e.g., bots working at the commit level), we set out to examine the performance of our approach in identifying bots that operate at each activity level (i.e., commits or issues/PRs). This analysis provides a snapshot of our approach performance when it is used by practitioners that want to study a specific type of bot (e.g., commit bots). Therefore, we split the bots that operate on different activity levels into two different datasets: issue_bots and commit_bots datasets. We add all accounts of type human that exist in the oracle to both datasets. Then, we apply the stratified 10-folds-cross-validation process to evaluate all classifiers discussed in Section 3.3 on each dataset.

Table 4 presents the results of all classifiers evaluated on issue_bots and commit_bots datasets. The Random Forest classifier outperforms other classifiers with F1-score 91.7% and AUC 98.7% in detecting the issue/PR bots in the issue_bots dataset. Also, Random Forest achieves the best performance in identifying the commit bots in the commit_bots dataset with an F1-score 89.7%. Logistic Regression achieves the best performance in terms of AUC (97.3%).

Our results show that using profile information (e.g., account bio) and account activity features (e.g., median activity per day) beside the text similarity features (e.g., commit message similarity) collected from different repositories is effective to identify the account type (i.e., human or bot) of a GitHub user.

> ***The Random Forest classifier performs best in detecting bots, with an F1-score 92.4%. The Random Forest model outperforms other classifiers in detecting bots on commits or issues bots with F1-score 91.7% and 89.7%, respectively.***

## 4.2 RQ2: What features are the best indicators of bot accounts?

**Motivation:** Given that Random Forest achieves a high performance (92.4%) in identifying the account type, we want to better understand the most important features that contribute to the prediction. By understanding these influential factors, we may further study the characteristics that differentiate bots from humans on social coding platforms. Also, these features can be used to manually identify bots on GitHub.

**Approach:** We use the permutation feature importance technique [7] to measure the most influential features in the Random Forest classifier. This technique randomly permutes the values of one feature while maintaining the values of the remaining features. This is done to dissociate between the permuted feature and the target variable. Finally, we evaluate the classifier's performance using the test set and compare its performance to the performance before the permutation (baseline). The permuted feature is considered important if the classifier's performance decreases significantly compared to the baseline's performance. This process is applied to all features discussed in Section 3.1. We use *permutation_importance* function in the Scikit-learn to compute the feature importance values in the Random Forest. To statistically compare the important feature distributions between bots and humans, we use the non-parametric Mann-Whitney-Wilcoxon test [49]. In this context, the null hypothesis ($H_0$) is that the feature distribution of bots and humans are the same, and the alternative hypothesis ($H_1$) is that these distributions differ. Moreover, we also use Cliff's Delta [38] to quantify the difference between these groups of observations beyond $p$-value interpretation. We use Romano et al. [38] guides for interpreting the effect size ($|\delta|$) using the following thresholds: $|\delta| < 0.147$ 'negligible', $|\delta| < 0.33$ 'small', $|\delta| < 0.474$ 'medium', otherwise 'large'.

**Results:** Table 5 presents the distributions of the five most important features of the features discussed in Section 3.1. The features are sorted by their importance level, from most to least important. We notice that the most important features are related to the account profile information. On the other hand, we find that the unique repository activities, unique commit activities, commit message text similarity features are not helpful for bot detection. In other words, these features do not provide significant contributions to the prediction. Thus, they can be removed from the classifier.

Upon closer examination of the most important feature (*account name*) distributions for both humans and bots, we find that while 42.2% of bot accounts names (287 accounts) contain the term 'bot', only 0.13% of the human accounts name (6 accounts) contain it. Moreover, we find a statistically significant difference ($p$-value= 0)

**Table 4: Performance of the classifiers on issue_bots and commit_bots.**

| | Classifier | Precision (%) | Recall (%) | F1-score (%) | AUC (%) |
|---|---|---|---|---|---|
| issue_bots | **Logistic Regression** | 94.6 | 83.3 | 88.6 | 98.0 |
| | **K-Nearest Neighbors** | 81.0 | 51.0 | 62.6 | 86.0 |
| | **Decision Tree** | 88.0 | 85.3 | 86.6 | 93.7 |
| | **Random Forest** | **95.3** | **88.4** | **91.7** | **98.7** |
| | **Support-Vector Machine** | 88.2 | 80.6 | 84.2 | 91.5 |
| commit_bots | **Logistic Regression** | **92.3** | 79.9 | 85.7 | **97.3** |
| | **K-Nearest Neighbors** | 80.5 | 54.0 | 64.6 | 87.9 |
| | **Decision Tree** | 81.0 | 80.4 | 80.7 | 89.5 |
| | **Random Forest** | 92.1 | **87.5** | **89.7** | 95.9 |
| | **Support-Vector Machine** | 92.0 | 87.0 | 89.4 | 94.3 |

between these distributions with a medium effect size ($\delta = 0.421$). The second most important feature, according to our analysis, is the *account login*. Comparing the *account login* distributions of bots and humans, the results show that 71% of bot accounts login (482 accounts) contain the term 'bot', and only 1% of the human accounts name (42 accounts) contain it. In addition to that, the statistical analysis also shows that the difference between these distributions is statistically significant ($p$-value= 0) with a large effect size ($\delta = 0.699$). Also related to the account profile information, the *number of followers* is within the most relevant features to determine the account type. Compared to humans, bots tend to have fewer followers: the median number of followers for bot accounts is 0, while human accounts have seven followers on median. These differences in the *number of followers* between the humans and bots distributions are statistically significant with a large effect size ($p$-value= $8.87^{218}$, $\delta = -0.74$).

Besides the profile information, the *Issue/PR comments similarity* feature also contributes to distinguishing bot accounts from human accounts due to the high similarity within bots' comments. Our statistical analysis shows that the distributions are different ($p$-value= 0.005, $\delta = 0.064$). For human accounts, Table 5 shows that the mean *Issue/PR comments similarity* is 11%. On the other hand, the mean *Issue/PR comments similarity* of the bot accounts is 34%. Further, 25% (3rd quartile) of bot accounts have more than 89% of text similarity on pull requests and issues. The account activity feature is also relevant for distinguishing the account type. The distribution of the *median activity per day* in Table 5 shows that bots are more active than humans, where bots and humans have on average 18.01 and 2.27 *median activity per day*, respectively. The statistical analysis shows that the difference between these distributions is statistically significant ($p$-value= $1.5^{-11}$) with a negligible effect size ($\delta = 0.138$).

***Account profile information features (account login and name) are the best indicators to identify the account type. Comment similarity in issues and pull requests is another important indicator that contributes to correctly predicting the account type on GitHub.***

## 4.3 RQ3: How effective is our approach compared to the state-of-the-art?

**Motivation:** In RQ1, we assessed the performance of our approach and found that the Random Forest classifier achieves the best results in identifying bot accounts with an F1 score of 92.4%. In this RQ, we set out to examine the performance of the state-of-the-art approaches (BoDeGHa [24] and BIMAN [17]) in detecting bots regardless of their types (i.e., commit or issue bots). This analysis allows us to put our results into perspective and compares the performance of our approach against the state-of-the-art approaches. In addition, it will give us an idea about the practicality of our approach to detecting different types of bots.

**Approach:** To evaluate the performance of BoDeGHa on our dataset, we employed the tool implemented by BoDeGHa authors [22], which takes the repository name as an input and prints the list of accounts and their types (human or bots). Therefore, we provide the list of repositories in our dataset discussed in Section 3.2 as an input and store the tool's output. We ran the tool using the default parameters [24]. For some repositories, the tool might return accounts that do not exist in the oracle (i.e., we do not have their ground truth). These accounts might have contributed to the repository after the creation of the oracle. Therefore, we only consider the accounts that exist in our oracle. Unsurprisingly, we find 610 cases where BoDeGHa returns inconsistent account types of the same account. For example, the *ualbertalib-bot* account in ualbertalib/NEOSDiscovery repository is considered human, but in ualbertalib/jupiter repository is considered a bot. This happens because BoDeGHa analyzes the account activities (e.g., pull request comments) in a single repository to perform the classifications. In other words, the account activities differ from one repository to another, which leads to different classifications of the same account. To mitigate this issue, we use the majority vote by taking the account type returned by most of the repositories. For example, in the case of *ualbertalib-bot*, we consider it as a bot because BoDeGHa classified the account as a bot in two repositories and a human in one repository. It is important to note that we find 15 accounts where the number of votes on their type is equal. We consider those cases as false negative because the user of BoDeGHa would not be able to identify their types without a manual examination of these accounts.

**Table 5: The distribution of the most important features.**

| Feature | Account Type | Min. | Q25% | Median | Mean | Q75% | Max. |
|---|---|---|---|---|---|---|---|
| **Account name** | **Bots** | 0 | 0 | 0 | 0.42 | 1 | 1 |
| | **Humans** | 0 | 0 | 0 | 0.001 | 0 | 1 |
| **Account login** | **Bots** | 0 | 0 | 1 | 0.70 | 1 | 1 |
| | **Humans** | 0 | 0 | 0 | 0.01 | 0 | 1 |
| **Number of followers** | **Bots** | 0 | 0 | 0 | 1.47 | 0 | 444 |
| | **Humans** | 0 | 1 | 7 | 31.11 | 27 | 3470 |
| **Issue/PR comments similarity** | **Bots** | 0 | 0 | 0 | 0.34 | 0.89 | 1 |
| | **Humans** | 0 | 0 | 0.09 | 0.11 | 0.18 | 1 |
| **Median activity per day** | **Bots** | 0 | 0 | 0 | 18.01 | 6 | 300 |
| | **Humans** | 0 | 0 | 0 | 2.27 | 1 | 78 |

To evaluate BIMAN's performance, we run its tool [18] on all accounts that exist in our datasets. BIMAN requires the users' commits meta-data such as commit message, filenames, and timestamp to make the prediction. Therefore, we leverage World of Code (WoC) [31] to collect the required data for all users in our dataset. WoC is a collection of software development activities for open-source projects. The account name and email combination serves as the account's primary ID on the WoC dataset. We use the account login on GitHub to retrieve the name and email address associated with that account using the GitHub API. We find the information for 2,133 accounts (2,061 humans and 72 bots). Then, we fetch the collected information into WoC to retrieve the commits' meta-data associated with these accounts. WoC has only the information for 1,938 accounts (1,898 humans and 40 bots), totaling more than 6 million commits. Finally, we provide the WoC output as an input for BIMAN to identify the type for these accounts. Finally, we compare BoDeGHa and BIMAN's performance to the classifier's performance that achieved the best results in RQ1 (i.e., Random Forest). It is important to note that we evaluate the performance of another Random Forest classifier using the same dataset (1,938 accounts) used to assess BIMAN's performance to compare the Random Forest's against BIMAN's performances.

**Table 6: Comparison of our approach against the state-of-art approaches.**

| Approach | Precision (%) | Recall (%) | F1-score (%) |
|---|---|---|---|
| **BoDeGHa** | 77.5 | 5.9 | 11 |
| **Random Forest** | **95.7** | **89.4** | **92.4** |
| **BIMAN*** | 83.6 | 99.4 | 90.8 |
| **Random Forest*** | **100** | **100** | **100** |

* The results are based on 1,938 GitHub accounts.

**Results:** Table 6 presents the F1-score for the state-of-the-art approaches against the Random Forest classifiers when identifying the account type in our dataset. We observe that the Random Forest classifier outperforms BoDeGHa and BIMAN with an F1-score higher than 92.4%. Surprisingly, BoDeGHa achieves a low F1-score (12.1%) in identifying the account type. To better understand the reason for BoDeGHa's low performance, we examined the results and found that BoDeGHa tool has a limitation that requires at least ten comments for each account in the repository. Maybe this is due to the reason that BoDeGHa achieves the best performance when the examined account has at least ten comments [24]. This serves as another reason for leveraging the global activities of an account on GitHub (i.e., activities in all repositories) rather than a single repository. This is especially useful in case the practitioners examine repositories with low activity. On the other hand, the Random Forest classifier outperforms BIMAN with F1-score 100% when it is evaluated on the same dataset (1,938 accounts). The reason behind the Random Forest's high performance is that most of the bot accounts have the word 'bot' in the account login and name. These two features (account login and name) are the two most important features, as discussed in RQ2. For the bot accounts that do not have the word 'bot' in the profile information, they either have no (or a low number of) followers or the median activity per day is 0. Our findings highlight the importance of using features other than the text-similarity features to identify the account type in GitHub.

***Our Random Forest classifier outperforms the state-of-the-art approaches in identifying bots that operate at the commit or issue/PR level with an F1-score higher than 92.4%.***

## 5 DISCUSSION

To have deeper insights into the reasons that led our approach to misclassify the account types in RQ1, the first two authors examined the RQ1 results. We found five main reasons that yield to account type misclassification:

**No activity:** One dimension that our approach depends on to classify the account type is the activities performed by an account. We found 29 bots misclassified as humans. Those misclassified accounts had been inactive for the last three months before we conducted this study. To retrieve the account activities on GitHub, we leveraged GitHub API as discussed in Section 3.2. We reiterate that the GitHub API returns the latest 300 activities that occurred in the last three months only. Due to this limitation, our approach effectively identifies the type of active accounts. Even with this limitation, our

approach outperforms the state-of-art approaches in identifying the account type as shown in the results of RQ3.

**Has word 'Bot' in their login or bio:** We found that our approach misclassified 17 human accounts as bots. Upon closer examination, those accounts have the word 'bot' in their logins (e.g., *brokenbot*). This is expected because the 'Account login' feature is the most important feature used by the Random Forest model to predict the account type, as discussed in RQ2.

**Different activity type:** There are 14 misclassified accounts with no text (e.g., issue comment) associated with these accounts. Interestingly, they are active accounts but without comments related to these activities. For example, *osrfbuild* bot opens a pull request without providing any text/comment to describe the pull request. Among the 14 misclassified accounts, there are 12 bots accounts classified as humans because they perform different types of activities in the repository. The *osrfbuild* bot performs three distinct activities on GitHub related to the repository, pull requests, and commit activities. Therefore, our approach misclassified it as a human. On the other hand, the remaining two accounts of type human are misclassified as bots. Both accounts perform a single type of task on GitHub (e.g., create a repository). That is because most activities of these two accounts are either on private repositories or the accounts have a low activity rate (i.e., not too active). This suggests that future detection approaches should not entirely depend on account comments to identify the account type. Moreover, they need to consider that a bot can perform various tasks on the repository in the future, although there are currently few bots who can do multiple tasks [46].

**Mixed accounts:** We found 14 misclassified bot accounts with a high number of followings (on median 87.5 followings). We deep-dived into the results and observed that these accounts are mixed account types. In other words, these are human accounts and the account owner allows bots to perform tasks using these accounts. In fact, in some accounts, we have difficulty identifying their types. This is also observed by the authors of BoDeGHa while manually annotating the accounts to create the ground-truth [24]. New techniques are necessary to identify those accounts on social coding platforms (e.g., GitHub). There are early-stage studies focusing on identifying the mixed accounts at the comment level (e.g., PR comment) [13, 23]. To the best of our knowledge, no approach identifies the mixed accounts at the account level. Thus, we encourage the research community to advance this area by studying the mixed accounts and their characteristics, proposing account levels techniques to identify those accounts. We plan in the future to evaluate our approach performance in identifying the mixed accounts besides the full human and bot accounts.

# 6 IMPLICATIONS

In the following, we discuss implications of our approach and results for researchers, practitioners, and social coding platforms.

## 6.1 Implications for Researchers

Although using bots on the social coding platform has several advantages (e.g., decreases time to merge pull requests [47]), they add extra effort on the researchers' side to filter them out, especially those who depend on the software project historical data. In fact, some empirical studies acknowledged that they applied pre- and post-processing techniques to filter out bots [16, 30, 41]. Our approach can help researchers to identify the bot presence regardless of their activities in order to handle them differently based on the researcher's intention.

Considering that bots on GitHub have different behaviours [46], such as posting comments on issues or creating new pull requests, identifying commit or issue bots opens new research directions related to their effects on developers. For example, researchers can investigate the impact of bots that create new pull requests on the developers' behaviors and decision making.

As discussed in Section 5, some bots can perform more than one task in the project. Therefore, we encourage the research community to explore how practitioners perceive multitasks bots versus several bots that perform the same tasks in the repository [14].

## 6.2 Implications for Practitioners

Some software practitioners are worried about bots' intrusive behaviors on social coding platforms. For example, bots change pull requests content or even spam repositories with unsolicited comments on the GitHub issues [48]. Using our approach, repository owners and projects maintainers can monitor the contributors (e.g., bots that impersonate humans) and take action once they identify undesirable bot behaviors on a repository.

Prior work shows that newcomers feel intimidated by bots responding to them on pull requests [48], which adds to the large number of barriers already faced by these contributors [43]. This might decrease the success of their contributions since newcomers get discouraged by the bot comments. It is crucial for newcomers to be aware of the presence of bots and know beforehand whether they are communicating with a human or bot. Using our approach helps the newcomers to identify bots in their project and react accordingly to the comments based on its author account type.

## 6.3 Implications for Social Coding Platforms

Because of the growing use of bots for collaborative development activities [21], a proliferation of bots to automate software development tasks on social coding platforms is expected. Consequently, we encourage social coding platforms to leverage our approach to distinguish human contributors from bot contributors in a repository by tagging the bot accounts on the platform. This enables the platforms to summarize different bot activities to keep the repository owners and maintainers aware of these activities. For example, the summary report can include the interaction of bots and humans, the frequency of bots' activities, and their types (e.g., commit code) for each bot. This report might help the project maintainers to identify and monitor bots that operate on the repository and have not been approved by them (i.e., external bots). This would facilitate the management of bots on the repository and avoid wasting maintainers' time filtering non-humans' content.

## 6.4 BotHunter Tool

As we showed in RQ1, our approach effectively identifies the account type on GitHub. To put our approach into the hands of practitioners and researchers, we implemented it as a tool called BotHunter and made it publicly available [5]. BotHunter can yield more

accurate and consistent results in classifying the GitHub accounts as human or bot, requiring a single tool for multiple activity types.

BotHunter takes two inputs, the GitHub API key and GitHub account login, for the account to be examined. Then, it leverages the GitHub API to extract features for the specified account. Finally, it uses a Random Forest model to classify the accounts. We trained this Random Forest classifier on all GitHub accounts in our dataset.

# 7 THREATS TO VALIDITY

In this section, we discuss the threats to our study's internal and external validity.

## 7.1 Internal validity

Internal validity concerns confounding factors that could have influenced our results. To extract the features for the accounts in our dataset, we depended on the GitHub API to collect those features. However, the GitHub API returns the top 300 activities in the last three months. Including activities that span more than three months might change our results. Nevertheless, our approach achieves high performance (F1-measure 92.4%) on predicting the accounts type with this limitation.

Another threat to internal validity is that we constructed our oracle using datasets of previous studies [17, 24], which may not be curated correctly. However, both datasets have been curated by at least two annotators (authors of the previous work) with a high level of agreement between the annotators. Moreover, during our examination of the misclassified cases by our approach in Section 5, we did not find any mislabeled account type (e.g., bot labeled as human), which makes us confident in the used datasets.

## 7.2 External validity

External validity concerns the generalization of our findings. In our study, we assessed the performance of our approach using 5,107 GitHub accounts collected from prior work. Hence, our results may not generalize to other datasets or social coding platforms (e.g., BitBucket). However, we believe that our dataset is comprehensive and contains human, commit, and issue bots. Future work can evaluate our approach on a larger dataset that composes more accounts on different social coding platforms (e.g., BitBucket). Another threat is that we used stratified 10-folds-cross-validation to evaluate the classifiers, and we do not have a separate test set, which might affect the generalizability of the results. We plan to evaluate the classifiers on test sets that have not been used in this work.

# 8 CONCLUSION

In this paper, we propose an approach that identifies bots in GitHub regardless of their activity levels (issues or commits), considers multiple projects, and includes features disregarded in previous work. We extracted 19 features related to the account's profile information (e.g., account login), activities (e.g., median response time), and text similarity (e.g., commit messages similarity) for 5,107 GitHub accounts. Then, we evaluated the performance of five machine learning classifiers, namely, Logistic Regression, K-Nearest Neighbors, Decision Tree, Random Forest, and Support-Vector Machine, in identifying the account type (i.e., human or bot). The Random Forest classifier achieved the best results among all the classifiers with an F1-score of 92.4% and AUC of 98.7%. Also, our results show that account login and name are the most important features used by the Random Forest classifier to reveal the account type. Finally, our Random Forest classifier outperforms the state-of-the-art approaches in detecting bots regardless of their types.

In the future, we plan to evaluate our approach on social coding platforms other than GitHub (e.g., BitBucket, GitLab) and larger datasets. Moreover, we will evaluate more sophisticated models (e.g., neural networks and transformers) on the task of identifying software bots on GitHub. We plan (and encourage others) to propose and evaluate approaches to identify the mixed accounts on social coding platforms. Finally, we will use our approach to conduct a large-scale study to understand the amount of bots adopted in software projects and their impact on software development (e.g., issue fixing time). By making BotHunter available as a tool we expect to empower practitioners and researchers who want to classify GitHub user accounts as bots and humans.

# ACKNOWLEDGMENT

This work is partially supported by the National Science Foundation under Grant numbers 1815503 and 1900903, and CNPq grant #313067/2020-1.

# REFERENCES

[1] R. Abdalkareem, S. Mujahid, and E. Shihab. 2020. A Machine Learning Approach to Improve the Detection of CI Skip Commits. *IEEE Transactions on Software Engineering* (2020), 1–1. https://doi.org/10.1109/TSE.2020.2967380

[2] Ahmad Abdellatif, Khaled Badran, and Emad Shihab. 2020. MSRBot: Using Bots to Answer Questions from Software Repositories. *Empirical Software Engineering (EMSE)* 25 (2020), 1834–1863. Issue 3.

[3] Ahmad Abdellatif, Diego Elias Costa, Khaled Badran, Rabe Abdelkareem, and Emad Shihab. 2020. Challenges in Chatbot Development: A Study of Stack Overflow Posts. In *Proceedings of the 17th International Conference on Mining Software Repositories (MSR'20)*. To Appear.

[4] Ahmad Abdellatif, Mairieli Wessel, Igor Steinmacher, Marco A. Gerosa, and Emad Shihab. 2022. BotHunter: An Approach to Detect Software Bots in GitHub. https://doi.org/10.5281/zenodo.6395211

[5] Ahmad Abdellatif, Mairieli Wessel, Igor Steinmacher, Marco A. Gerosa, and Emad Shihab. 2022. BotHunter tool. https://github.com/ahmad-abdellatif/BotHunter. (Accessed on 03/29/2022).

[6] Karan Aggarwal, Tanner Rutgers, Finbarr Timbers, Abram Hindle, Russ Greiner, and Eleni Stroulia. 2015. Detecting duplicate bug reports with software engineering domain knowledge. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. 211–220. https://doi.org/10.1109/SANER.2015.7081831

[7] André Altmann, Laura Toloşi, Oliver Sander, and Thomas Lengauer. 2010. Permutation importance: a corrected feature importance measure. *Bioinformatics* 26, 10 (04 2010), 1340–1347. https://doi.org/10.1093/bioinformatics/btq134 arXiv:https://academic.oup.com/bioinformatics/article-pdf/26/10/1340/16892402/btq134.pdf

[8] L. Bao, Z. Xing, X. Xia, D. Lo, and S. Li. 2017. Who Will Leave the Company?: A Large-Scale Industry Study of Developer Turnover by Mining Monthly Work Report. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. 170–181. https://doi.org/10.1109/MSR.2017.58

[9] Kelly Blincoe, Jyoti Sheoran, Sean Goggins, Eva Petakovic, and Daniela Damian. 2016. Understanding the popular users: Following, affiliation influence and leadership on GitHub. *Information and Software Technology* 70 (2016), 30–39. https://doi.org/10.1016/j.infsof.2015.10.002

[10] Chris Brown and Chris Parnin. 2019. Sorry to Bother You: Designing Bots for Effective Recommendations. In *Proceedings of the 1st International Workshop on Bots in Software Engineering (BotSE)*.

[11] Rich Caruana and Alexandru Niculescu-Mizil. 2006. An Empirical Comparison of Supervised Learning Algorithms. In *Proceedings of the 23rd International Conference on Machine Learning* (Pittsburgh, Pennsylvania, USA) *(ICML '06)*. Association for Computing Machinery, New York, NY, USA, 161–168. https://doi.org/10.1145/1143844.1143865

[12] Antonio Carvalho, Welder Luz, Diego Marcilio, Rodrigo Bonifácio, Gustavo Pinto, and Edna Dias Canedo. 2020. C-3PR: A Bot for Fixing Static Analysis

Violations via Pull Requests. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 161–171. https://doi.org/10.1109/SANER48275.2020.9054842

[13] Nathan Cassee, Christos Kitsanelis, Eleni Constantinou, and Alexander Serebrenik. 2021. Human, bot or both? A study on the capabilities of classification models on mixed accounts. In *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 654–658. https://doi.org/10.1109/ICSME52107.2021.00075

[14] Ana Paula Chaves and Marco Aurelio Gerosa. 2018. Single or multiple conversational agents? An interactional coherence comparison. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–13.

[15] Clayton Allen Davis, Onur Varol, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. 2016. Botornot: A system to evaluate social bots. In *Proceedings of the 25th international conference companion on world wide web*. 273–274.

[16] Tapajit Dey, Yuxing Ma, and Audris Mockus. 2019. Patterns of Effort Contribution and Demand and User Classification Based on Participation Patterns in NPM Ecosystem. In *Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering* (Recife, Brazil) *(PROMISE'19)*. Association for Computing Machinery, New York, NY, USA, 36–45. https://doi.org/10.1145/3345629.3345634

[17] Tapajit Dey, Sara Mousavi, Eduardo Ponce, Tanner Fry, Bogdan Vasilescu, Anna Filippova, and Audris Mockus. 2020. Detecting and Characterizing Bots That Commit Code. In *Proceedings of the 17th International Conference on Mining Software Repositories* (Seoul, Republic of Korea) *(MSR '20)*. Association for Computing Machinery, New York, NY, USA, 209–219. https://doi.org/10.1145/3379597.3387478

[18] Tapajit Dey, Sara Mousavi, Eduardo Ponce, Tanner Fry, Bogdan Vasilescu, Anna Filippova, and Audris Mockus. 2022. BIMAN: Bot Identification by commit Message, commit Association, and author Name. https://github.com/ssc-oscar/BIMAN_bot_detection. (Accessed on 01/05/2022).

[19] Tapajit Dey, Bogdan Vasilescu, and Audris Mockus. 2020. An exploratory study of bot commits. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. 61–65.

[20] Phillip George Efthimion, Scott Payne, and Nicholas Proferes. 2018. Supervised machine learning bot detection techniques to identify social twitter bots. *SMU Data Science Review* 1, 2 (2018), 5.

[21] Linda Erlenhov, Francisco Gomes de Oliveira Neto, Riccardo Scandariato, and Philipp Leitner. 2019. Current and Future Bots in Software Development. In *Proceedings of the 1st International Workshop on Bots in Software Engineering* (Montreal, Quebec, Canada) *(BotSE '19)*. IEEE Press, Piscataway, NJ, USA, 7–11. https://doi.org/10.1109/BotSE.2019.00009

[22] Mehdi Golzadeh. 2021. BoDeGHa: A python tool to predict the identity type in github activities (Human,Bot). https://github.com/mehdigolzadeh/BoDeGHa. (Accessed on 04/22/2021).

[23] Mehdi Golzadeh, Alexandre Decan, Eleni Constantinou, and Tom Mens. 2021. Identifying bot activity in GitHub pull request and issue comments. In *Identifying bot activity in GitHub pull request and issue comments (3rd International Workshop on Bots in Software Engineering)*.

[24] Mehdi Golzadeh, Alexandre Decan, Damien Legay, and Tom Mens. 2021. A ground-truth dataset and classification model for detecting bots in GitHub issue and PR comments. *Journal of Systems and Software* 175 (2021), 110911. https://doi.org/10.1016/j.jss.2021.110911

[25] Mehdi Golzadeh, Damien Legay, Alexandre Decan, and Tom Mens. 2020. Bot or not? Detecting bots in GitHub pull request activity based on comment similarity. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. 31–35.

[26] James Hardwick. 2020. GitActor.user should be type Actor, not User. Cannot distinguish between bots/users otherwise. - GitHub Ecosystem - GitHub Support Community. https://github.community/t/gitactor-user-should-be-type-actor-not-user-cannot-distinguish-between-bots-users-otherwise/14559. (Accessed on 04/13/2021).

[27] Y. Kamei, E. Shihab, B. Adams, A. E. Hassan, A. Mockus, A. Sinha, and N. Ubayashi. 2013. A large-scale empirical study of just-in-time quality assurance. *IEEE Transactions on Software Engineering* 39, 6 (2013), 757–773. https://doi.org/10.1109/TSE.2012.70

[28] SayedHassan Khatoonabadi, Diego Elias Costa, Rabe Abdalkareem, and Emad Shihab. 2021. On Wasted Contributions: Understanding the Dynamics of Contributor-Abandoned Pull Requests. arXiv:2110.15447 [cs.SE]

[29] Carlene Lebeuf, Alexey Zagalsky, Matthieu Foucault, and Margaret-Anne Storey. 2019. Defining and Classifying Software Bots: A Faceted Taxonomy. In *Proceedings of the 1st International Workshop on Bots in Software Engineering* (Montreal, Quebec, Canada) *(BotSE '19)*. IEEE Press, Piscataway, NJ, USA, 1–6. https://doi.org/10.1109/BotSE.2019.00008

[30] J. Lipcak and B. Rossi. 2018. A Large-Scale Study on Source Code Reviewer Recommendation. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 378–387. https://doi.org/10.1109/SEAA.2018.00068

[31] Y. Ma, C. Bogart, S. Amreen, R. Zaretzki, and A. Mockus. 2019. World of Code: An Infrastructure for Mining the Universe of Open Source VCS Data. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. 143–154. https://doi.org/10.1109/MSR.2019.00031

[32] Amanda Minnich, Nikan Chavoshi, Danai Koutra, and Abdullah Mueen. 2017. BotWalk: Efficient adaptive exploration of Twitter bot networks. In *Proceedings of the 2017 IEEE/ACM international conference on advances in social networks analysis and mining 2017*. 467–474.

[33] Samim Mirhosseini and Chris Parnin. 2017. Can Automated Pull Requests Encourage Software Developers to Upgrade Out-of-date Dependencies?. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering* (Urbana-Champaign, IL, USA) *(ASE 2017)*. IEEE Press, Piscataway, NJ, USA, 84–94. http://dl.acm.org/citation.cfm?id=3155562.3155577

[34] Martin Monperrus. 2019. Explainable Software Bot Contributions: Case Study of Automated Bug Fixes. In *Proceedings of the 1st International Workshop on Bots in Software Engineering* (Montreal, Quebec, Canada) *(BotSE '19)*. IEEE Press, Piscataway, NJ, USA, 12–15. https://doi.org/10.1109/BotSE.2019.00010

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[36] Marco Polignano, Marco Giuseppe de Pinto, Pasquale Lops, and Giovanni Semeraro. 2019. Identification Of Bot Accounts In Twitter Using 2D CNNs On User-generated Contents.. In *CLEF (Working Notes)*.

[37] Jorge Rodríguez-Ruiz, Javier Israel Mata-Sánchez, Raul Monroy, Octavio Loyola-Gonzalez, and Armando López-Cuevas. 2020. A one-class classification approach for bot detection on twitter. *Computers & Security* 91 (2020), 101715.

[38] Jeanine Romano, Jeffrey D Kromrey, Jesse Coraggio, and Jeff Skowronek. 2006. Appropriate statistics for ordinal level data: Should we really be using t-test and Cohen'sd for evaluating group differences on the NSSE and other surveys. In *annual meeting of the Florida Association of Institutional Research*. 1–33.

[39] Ricardo Romero, Esteban Parra, and Sonia Haiduc. 2020. Experiences Building an Answer Bot for Gitter. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops* (Seoul, Republic of Korea) *(ICSEW'20)*. Association for Computing Machinery, New York, NY, USA, 66–70. https://doi.org/10.1145/3387940.3391505

[40] Giovanni C. Santia, Munif Ishad Mujib, and Jake Ryland Williams. 2019. Detecting Social Bots on Facebook in an Information Veracity Context. *Proceedings of the International AAAI Conference on Web and Social Media* 13, 01 (Jul. 2019), 463–472. https://ojs.aaai.org/index.php/ICWSM/article/view/3244

[41] F. Sarker, B. Vasilescu, K. Blincoe, and V. Filkov. 2019. Socio-Technical Work-Rate Increase Associates With Changes in Work Patterns in Online Projects. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. 936–947. https://doi.org/10.1109/ICSE.2019.00099

[42] Dragos Serban, Bart Golsteijn, Ralph Holdorp, and Alexander Serebrenik. 2021. SAW-BOT: Proposing Fixes for Static Analysis Warnings with GitHub Suggestions. In *2021 IEEE/ACM Third International Workshop on Bots in Software Engineering (BotSE)*. 26–30. https://doi.org/10.1109/BotSE52550.2021.00013

[43] Igor Steinmacher, Tayana Conte, Marco Aurélio Gerosa, and David Redmiles. 2015. Social barriers faced by newcomers placing their first contribution in open source software projects. In *Proceedings of the 18th ACM conference on Computer supported cooperative work & social computing*. 1379–1392.

[44] Margaret-Anne Storey and Alexey Zagalsky. 2016. Disrupting Developer Productivity One Bot at a Time *(FSE 2016)*. Association for Computing Machinery, New York, NY, USA, 928–931. https://doi.org/10.1145/2950290.2983989

[45] Simon Urli, Zhongxing Yu, Lionel Seinturier, and Martin Monperrus. 2018. How to Design a Program Repair Bot? Insights from the Repairnator Project. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice* (Gothenburg, Sweden) *(ICSE-SEIP '18)*. Association for Computing Machinery, New York, NY, USA, 95–104. https://doi.org/10.1145/3183519.3183540

[46] Mairieli Wessel, Bruno Mendes de Souza, Igor Steinmacher, Igor S. Wiese, Ivanilton Polato, Ana Paula Chaves, and Marco A. Gerosa. 2018. The Power of Bots: Characterizing and Understanding Bots in OSS Projects. *Proc. ACM Hum.-Comput. Interact.* 2, CSCW, Article 182 (Nov. 2018), 19 pages. https://doi.org/10.1145/3274451

[47] Mairieli Wessel, Alexander Serebrenik, Igot Wiese, Igor Steinmacher, and Marco A. Gerosa. 2020. Effects of Adopting Code Review Bots on Pull Requests to OSS Projects. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 1–11. https://doi.org/10.1109/ICSME46990.2020.00011

[48] Mairieli Wessel, Igor Wiese, Igor Steinmacher, and Marco A. Gerosa. 2021. Don't Disturb Me: Challenges of Interacting with Software Bots on Open Source Software Projects. *Procidings of ACM Human-Computer Interaction* CSCW (2021).

[49] Daniel S Wilks. 2011. *Statistical methods in the atmospheric sciences*. Vol. 100. Academic press.

[50] Marvin Wyrich and Justus Bogner. 2019. Towards an Autonomous Bot for Automatic Source Code Refactoring. In *Proceedings of the 1st International Workshop on Bots in Software Engineering* (Montreal, Quebec, Canada) *(BotSE '19)*. IEEE Press, Piscataway, NJ, USA, 24–28. https://doi.org/10.1109/BotSE.2019.00015

[51] Marvin Wyrich and Justus Bogner. 2019. Towards an Autonomous Bot for Automatic Source Code Refactoring. In *Proceedings of the 1st International Workshop on Bots in Software Engineering* (Montreal, Quebec, Canada) *(BotSE '19)*. IEEE Press, 24–28. https://doi.org/10.1109/BotSE.2019.00015

[52] Yang Zhang. 2018. Can we use Github API to determine if a Github user is a bot? - Stack Overflow. https://stackoverflow.com/questions/52731464/can-we-use-github-api-to-determine-if-a-github-user-is-a-bot. (Accessed on 04/13/2021).

[53] Yun Zhang, David Lo, Xin Xia, and Jian-Ling Sun. 2015. Multi-Factor Duplicate Question Detection in Stack Overflow. *Journal of Computer Science and Technology* 30, 5 (2015), 981–997. https://doi.org/10.1007/s11390-015-1576-4