Is there a Race & Ethnicity Bias in GitHub Issue Assignment?

Aniruddhan Murali
University of Waterloo
a25murali@uwaterloo.ca

Kishanthan Thangarajah
University of Waterloo
k4thanga@uwaterloo.ca

ABSTRACT

Diversity in Open Source Software (OSS) is integral to developing inclusive software for all. **Objective:** This paper complements existing diversity research on race & ethnicity bias in OSS. We want to identify how race & ethnicity affects issues being assigned between developers. **Methodology:** A quantitative analysis was performed over five repositories to understand relationship between assigner and assignee's perceptible ethnicity. We look at how bots assign issues and the spread of issue types among White and Non-White developers. Lastly we look at how much commit experience is relevant to get issues assigned for White and Non-White developers. **Results:** We observed that assignors assign issues largely to developers of same ethnicity. Bots assign issues more to White developers than Non White developers. Bugs, Test and Documentation Issue types have more White assignees than Non-White assignees. A single commit is much more helpful in getting issues assigned for a developer of White ethnicity. **Conclusion:** There are several systemic race bias inherent in GitHub. More diversity research is required to foster healthy OSS community.

1 INTRODUCTION

Lack of diversity within Software engineering and Open-source software (OSS) is well known [3]. This bias exists for both gender and race. Women constitute only 21 percent of the software development workforce and earn lesser than their male counterparts. This number is even smaller in Open Source Software (OSS) communities. With respect to race & ethnicity, 54.2 percent of Software developers, applications and systems software are White (Non-Hispanic), making that the most common race or ethnicity followed by 32.9 percent being Asian (Non-Hispanic) according to DATA USA [8]. We aim to contribute to the growing body of research specific to race bias and focus on how issues are assigned within GitHub projects. We hope that this study can shed some light on current practices in OSS with respect to race & ethnicity and help identify new strategies to encourage a more inclusive development environment.

2 MOTIVATION

The motivation for this work is the 'self-selection' bias [16] which arises from individuals selecting themselves into a group creating abnormal or undesirable conditions in a group. This would be a self fulfilling prophecy and make the group unapproachable to certain individuals. When such undesirable conditions persists for longer periods, it affects project growth as people lose interest and motivation to participate and contribute. We desire to measure this through how people assign issues to members of the project community through the lens of race & ethnicity. In addition to this, we also want to see how automated tools, such as bots, assign or suggest issues to reviewers and identify any potential bias introduced in the process, controlling for factors such as contribution experience and types of issues.

3 BACKGROUND AND RELATED WORK

There exists various studies that have looked into diversity issues by analyzing data from GitHub repositories. Plenty of studies have focused on GitHub pull request acceptance and gender related diversity issues. Terrell et al. have looked into the acceptance of Pull Requests between men and women on GitHub [4]. They found that when women gender was known, the rejection rate was high. Brokmeier extended further to identify gender bias with pull requests statistics at individual project level [6]. He also found that there is significant bias against women within projects.

However when it comes to race & ethnicity, there are very few studies describing its impact on OSS communities. As a specific example Artificial Intelligence has a diversity crisis in that existing biased data sets or tool kits can lead to incorrect or harmful consequences for end users [9]. Bogdan Vasilescu et al. looked at perceptions of diversity and it's impact within OSS communities including ideas on team constitution [12]. Reza Nadri et al. first looked at the relationship between developers race and ethnicity and the evaluation of their contributions in OSS, specifically targeting GitHub pull requests data. They found that although there is a low number of White developers participating in OSS development, they had 6-10 % of higher chance in getting their pull request accepted than other non-white developers. Interestingly, they also found that non-white developers are most likely to get their pull requests accepted by other non-white integrators than White integrators. [11]. Similar to this Reza Nadri et al. also looked at non merged and rejected pull requests on GitHub for potential race biases. They found that developer whose race is perceived as White have the high acceptance rate by other White developer whereas developers from Hispanic and Black ethnicity have high percent of their pull requests being closed and rejected by other White developers without any reason [10]. These two studies on race related discrimination in pull requests acceptance emphasize the need for further research that should focus on different streams of GitHub repository data such as discussions, comments, issues, projects boards to identify the presence of race and ethnicity related biases

There are no research that explore how issues are handled within OSS projects hosted in GitHub. Our work is new and fundamentally different from the existing studies as we will study how issues are manually assigned systematically in the first place in GitHub. As far as we are aware this is the first of it's kind and exposes systemic biases endemic to GitHub projects.

4 RESEARCH QUESTIONS

Existing studies focus on the bias towards specific race & ethnicity when accepting or rejecting pull requests in GitHub projects. We want to extend this further to understand whether race & ethnicity related bias exists when issues are being assigned within the project collaborators. Hence it is a systemic bias within GitHub. We can set up several hypothesis to test the existence of systemic race bias. We will be analyzing the manual assignor and assignee data to find statistical correlations with race. We can also analyze what types of issues are being assigned to users of specific race & ethnicity and possibly identify any relationship with contributor experience and the types issues are being assigned. We want to see whether there are any evidence that non-trivial issues, such a documentation fixes, build fixes, etc. are being assigned largely to a specific race & ethnicity while complex or rewarding types of issues, such as new features, are controlled within group of users of specific race & ethnicity. As an extension to this study, we also want to identify how automatic issues classification and assignment tools, such as GitHub bots, perform when assigning issues among developers. We want to identify whether automatic tools or bots introduce their own bias when selecting potential assignee for an issue. Based on this discussion, below are the set of research questions we want to answer through this study.

- **RQ1:** Do White developers assign issues largely to other White developers? Do Non-White developers assign issues largely to White developers? Do bots assign issues largely to White developers?
- **RQ2:** What types of issues (Bugs, Features, Build, Test, Documentation) are assigned to different race & ethnicity?
- **RQ3:** How does issue assignment vary with commit experience across different repositories and race & ethnicity?

5 DATA GATHERING

We intend to test the biases with data derived from GitHub REST API [13] and the WebScraper [14] extension for browsers. We did not use the GHTorrent, which is popular database for querying GitHub data, since we were analyzing a few set of repositories and using the GitHub REST API and browser scraping tools were sufficient to retrieve the required data. As the first step, we identified potential projects based on popularity containing several contributors, issues, and commits across different areas of technology. We went through the GitHub repositories that are active and popular based on number of stargazers and and number of contributors.

We used the following criteria to select repositories:

- The total number of issues, which are opened and closed, should be greater than 25000.
- The total number of contributors should be greater than 1000.
- No two GitHub repositories should be of the same category and each repository should be unique with the technology being developed.

Based on the above criteria, the list of repositories filtered are given in **Table 1**. We identified five repositories for the initial study and planning to include more repositories, if the identified repositories did not have enough user data to analyze.

Project	Description	Issues	Contributors
VSCode	An IDE	114037	1440
TensorFlow	A machine learning framework	32017	3028
Kubernetes	A container management system	38750	3111
Golang	A programming language	44523	1689
Elasticsearch	A search and analytics engine	27592	1644

Table 1: List of GitHub projects.

A healthy project will be perceived as inclusive among the collaborators when everyone is treated equally. With increased adaptation and popularity, OSS projects attract various contributors from all around the world. With time, such interested developers will become contributors to the project repository by pushing commits and changes. GitHub repositories use issues to track bugs, features, improvements, tasks, etc. for project management. A GitHub issue has the life cycle starting from created to eventually getting closed. There will be many users interacting throughout the life cycle of an issue. All the issues that gets created for a particular project will be managed and distributed by its contributors to other contributors to be worked on. This is the general practice followed by GitHub projects to flush the incoming and existing issues. When issues are being worked on, there can be various users getting involved. A developer can self assign issues and work on it. But most of the time, a contributor look at the issues and selects other contributor(s) to work on it by assigning the issue. For this study, we will mainly focus on two specific users on an issue, namely assignor and assignee.

From the identified GitHub repositories, we set out to find four categories of data, both from the issues as well as from the users involved, as given below.

1. Assignor and assignee information from issues
2. Types of issues (bug, feature, test, documentation, build, etc)
3. Level of contributions of users (number of commits for a given repository for a given user)
4. Race & ethnicity information of the users

5.1 Assignor and assignee information

An assignor is the person who goes through the issues and selects a suitable person to work on it and the assignee is the person who was selected by the assignor. When an assignor assigns an issue to an assignee, there will be an event created for that issue, which captures the information of both the users involved in the event. There can be one or more such assignor-assignee events for a particular issue. We can go through all the issues in a repository of interest, and capture these events and get the information of both assignor and assignee. But this information will only reveal the GitHub username of the assignor and assignee. We have to further query GitHub user profile API to get the real name of the user using their GitHub profile information publicly available. This was done for all five repositories.

5.2 Issue labels

In GitHub issues, the Issue labels are used to classify the issues. Different projects follow different conventions for their labels. We collected all unique labels applied to issues. This information also can be queried by going through each issue. In this work we intend to use these Issue labels to classify the issue as a Bug, Feature, Test, Build, Documentation or Other. This data will help determine types of issues assigned to different race & ethnicity.

5.3 Level of contributions

We then gathered data on the level of contributions from users. The level of contributions can be considered as the amount of commits a user has contributed to a given repository. This data will allow comparison of issues assigned across race & ethnicity by controlling for commit experience to that repository.

5.4 Race & ethnicity

Once we collected the assignor-assignee and their names, the next task was to identify the race or ethnicity information of the users. NamSor [1] is a leading tool to classify name, ethnicity and gender information with good accuracy. It offers REST API support to identify race ethnicity given a name to classify. We selected this tool to identify the ethnicity of all the user names, both assignor and assignee, that we mined from all GitHub repositories.

6 METHODOLOGY

We used different tools and approaches when collecting the required data. The first step was to get the data from issues. For this we used GitHub REST API [13] and the WebScraper [14] extension for browsers. WebScraper is a browser plugin that can scrape through websites. GitHub REST API is the standard way to query data from GitHub and it offers different API endpoints for different purposes. We used repository, contributors and user API endpoints to query the data we wanted. We used WebScraper extension for Firefox, GitHub REST API and NamSor tool to extract the following information.

6.1 Issue information

Using the WebScraper Tool for Mozilla Firefox, we scraped issue URLs in an iterative manner. For each issue we record the title and the labels for that issue. Within every issue we record status trackers of the form 'X assigned Y' through regular expression matching. An issue can have multiple assignor-assignee and we capture all of them for every issue in a given repository.

The issue types in this work are Bug, Feature, Test, Build, Documentation and Other. We took unique labels applied to issues and then classified them manually into one of said issue types and the default 'Other' - when label does not belong to any Issue type. The authors cross-verified the Issue Label classification to reduce classification inaccuracy. Then a precedence order (Bug > Feature > Test > Build > Documentation > Other) was used to classify issues into one type from it's multiple labels. We then use this manually classified Labels and precedence hierarchy to categorize issues into different types. In the absence of labels, we looked at the issue title, description and comments to identify key words that describe the type of the issue. An issue can be labelled as 'Other' as well.

6.2 User information

We first processed the scraped data and identified the assignor and assignee information captured from each issue. Since this was captured using the regular expression matching, we had to further filter the data to correctly identify the assignor and assignee name. In this case, we get only the user id information so we have to further query GitHub user API to get the actual name of each user (both assignee and assignor). Similarly, we used the repository/-contributors API endpoint from GitHub to find the contributions information of users for a given repository. By this way, using both GitHub REST API and the scraped data using WebScraper, we were able to iterate through all users and the number of commits to a given repository. Further, with the GitHub REST API limitation on listing user contributions up to 500 users, we also mixed the approach of using 'git log' command locally on repositories to get the contributions for users whom we did not not fall under the 500 user limitation. We manually cloned all five repositories locally and then used the 'git log' command along with the '–author' options to get the number of commits specifically from an author.

There are various automated bots used by the project repositories we identified [17], [18], [19]. As we captured all the users involved in issue assignment, these bots were also identified as users or contributors from the same data set because they have the word 'bot' in them.

6.3 Race and ethnicity Information

We intend to use NamSor[15] to identify race ethnicity information from the real name that we scraped from previous steps. NamSor API V2 provides a set of endpoints to classify race and ethnicity information for user names given. It also includes APIs to classify gender, and country location with good accuracy. However, we focused only on APIs related to race and ethnicity classification. The "/api2/json/usRaceEthnicityBatch" API of NamSor supports batch querying by sending names (first name and last name) in a batch up to 100 names. We used this endpoint to classify the real names we gathered from scraped GitHub user data. The NamSor API classifies the names into four different race and ethnicity categories as given in **Table 2**.

Race & Ethnicity	Description
W_NL	white, non latino
HL	hispano latino
A	asian, non latino
B_NL	black, non latino

Table 2: List of race and ethnicity classification from NamSor API.

```
{
  "script": "LATIN",
  "id": "ec265264-03f5-4ba7-8579-7ad7084c",
  "firstName": "Martin",
  "lastName": "Wicke",
  "raceEthnicityAlt": "HL",
  "raceEthnicity": "W_NL",
  "score": 17.93149231369229,
  "raceEthnicitiesTop": [
    "W_NL",
    "HL",
    "B_NL",
    "A"
  ],
  "probabilityCalibrated": 0.8717343407,
  "probabilityAltCalibrated": 0.8998540998
}
```

Figure 1: NamSor API sample response

The authors decided to focus only on White developers and Non-White developers, hence the categories HL, A and B_NL categories were merged as Non-White developers, with W_NL category as White developers. Figure 1 shows an example response from NamSor RaceEthnicityBatch API for a given input name "Martin Wicke". We processed this json response and capture the race and ethnicity value from the "raceEthnicity" field. In addition to the four different race and ethnicity values from NamSor API responses, we also added a special ethnicity value, **"Bot"**, for GitHub user names found to be from automated tools, actions or bots such as [17], [18], and [19].

6.4 Identified features

Based on collecting the above information on issues, users and race, we identified a set of features we will use with analysis and to build the relationship between race & ethnicity and level of contributions. **Table 3** lists does the identified features.

Feature	Description
IssueURL	GitHub URL of the issue
Assignor	GitHub user name of the Assignor
Assignee	GitHub user name of the Assignee
Title	GitHub issue title or the issue heading
IssueType	Type of the issue (Bug, Feature, Test, Build, Docs, Other)
Commits-Assignor	Number of commits by the Assignor to the project
Commits-Assignee	Number of commits by the Assignee to the project
Ethnicity-Assignor	Identified race and ethnicity of the Assignor
Ethnicity-Assignee	Identified race and ethnicity of the Assignee

Table 3: Identified Features.

7 RESULTS

As the first step, we went through the project level issues as well as combined issues and performed some filtering. **Table 4** lists the total number of issues identified to have both assignor and assignee from each of the selected repositories. From the issues, we further filtered issues which are self assigned by the users. Finally, the total number of issues we gathered for analysis were **98084**. This table also list down the number of White and Non-White contributors identified from the filtered issues.

Project	Issue count	Issue count (exclude self assigned)	Contributors	
			W	N_W
VSCode	52449	43522	22	28
TensorFlow	53931	32637	177	576
Kubernetes	19476	15341	162	184
Golang	11367	5682	184	154
Elasticsearch	3074	902	77	25
All projects	**140297**	**98084**	**619**	**966**

Table 4: Issues having both assignor and assignee.

As we used the WebScraper to scrape issues from GitHub repositories, with some projects we found that apart from issues, pull requests also had both assignor and assignee data. A good example is the Tensorflow GitHub project where we were able to identify more than 53000 unique issues and pull requests which had both assignor and assignee event. This observation was not significant with other GitHub projects as pull requests from those repositories did not have both assignor and assignee event. Since these type of events were significantly identified with Tensorflow project, we did not filter out the pull requests from the gathered data due to the fact that it is still a valid set of data we can include here with the assignor and assignee events. The next part of the analysis to identify the race & ethnicity of assignors and assignees from each project. **Table 5** lists the total number of assignors and assignees grouped by ethnicity - White (W) and Non-White (N_W) developers.

Project	Assignors		Assignees	
	W	N_W	W	N_W
VSCode	21	25	22	27
TensorFlow	106	412	136	362
Kubernetes	55	58	158	182
Golang	146	104	85	85
Elasticsearch	51	15	69	22
All projects	**378**	**614**	**469**	**677**

Table 5: Project level assignors and assignees.

Project	Assignor's Ethnicity	Assignee				
		mean		std_dv	Hypothesis	p_value
		W	N_W			
VSCode	W	0.508	0.492	0.198	N_W < W	0.7822725239789194
	N_W	0.568	0.432	0.207	N_W < W	0.06026818038523511
	All	0.541	0.459	0.205	N_W < W	0.23741810842163397
TensorFlow	W	0.389	0.611	0.393	W < N_W	1.4812376503731383e-05 ***
	N_W	0.282	0.718	0.370	W < N_W	2.0237180133283126e-49 ***
	All	0.304	0.696	0.377	W < N_W	5.493553288911331e-52 ***
Kubernetes	W	0.545	0.455	0.250	N_W < W	0.017867941778350077 *
	N_W	0.384	0.616	0.277	W < N_W	6.674101231977313e-06 ***
	All	0.462	0.538	0.276	W < N_W	0.02193101606937441 *
Golang	W	0.908	0.092	0.197	N_W < W	4.316296595232578e-49 ***
	N_W	0.893	0.107	0.245	N_W < W	2.9968648362579848e-33 ***
	All	0.902	0.098	0.218	N_W < W	2.3410180007321714e-80 ***
Elasticsearch	W	0.650	0.350	0.319	N_W < W	4.895546780304421e-06 ***
	N_W	0.748	0.252	0.321	N_W < W	0.0005960201356252027 **
	All	0.672	0.328	0.322	N_W < W	2.2316977163047963e-08 ***
All projects	W	0.676	0.324	0.362	N_W < W	1.2824941393390544e-30 ***
	N_W	0.401	0.599	0.409	W < N_W	3.4556962445178583e-16 ***
	All	0.506	0.494	0.414	W < N_W	0.5916597143792957
	Bot	**0.737**	**0.263**	**0.192**	N_W < W	**0.0021645021645021645** *

Table 6: Relationship between assignor's and assignee's race & ethnicity. Signif. codes: 0: *; 0.001: **; 0.05: ***

Issue Type	Issue Count	mean		std_dv		Hypothesis	p_value
		W	N_W	W	N_W		
Bug	52345	0.047	0.024	0.205	0.124	N_W < W	4.7278197234308835e-06 *
Feature	17958	0.017	0.008	0.114	0.058	N_W < W	0.07707763365002812
Test	8537	0.010	0.003	0.035	0.015	N_W < W	3.5235180201405573e-14 *
Build	4638	0.003	0.002	0.020	0.018	N_W < W	0.11582492938808953
Docs	1070	0.002	0.000	0.008	0.001	N_W < W	1.767480012782269e-06 *

Table 7: Relationship between issue types and assignee's race & ethnicity. Signif. codes: 0: *

7.1 Relationship between assignor's and assignee's race & ethnicity

To answer **RQ.1**, we need to understand how issues are assigned from the perspective of assignors, their race & ethnicity and how they were assigning issues to assignees. For each assignor, we gathered data on number of issues they assign to White and Non-White developers. We normalized the data by dividing by total number of White and Non-White assignees per repository to account for skewed distributions. We performed Mann–Whitney U tests on this normalized data to check if assignors are generally assigning more issues to one ethnicity than the other. Mann Whitney Test is done for below three cases:

- Assignor is White
- Assignor is Non-White
- All Assignors

We repeated the test for all the projects separately and then we merged the data from all the projects and performed the same test. The results from these set of tests can be found from Table 6.

Statistically significant results show white developers overwhelmingly assign to white developers and non-white assign to non-white developers. At the repository level for Golang and Elasticsearch all assignors assigning to White developers was statistically significant. For Tensorflow all assignors assigning to Non White developers is statistically significant.

7.2 Automated bots as assignors

As part of **RQ.1**, we can also understand how some bots and tools automatically assign issues to users by doing a similar study when assignor is found to be a bot. We manually identified automated bots used by the selected projects and filtered their data. We then performed Mann Whitney test just as in **Section 7.1** to understand the relationship of issues assigned by bots with race & ethnicity of the assignee. This data was again normalized by total developers of same ethnicity to compare different proportions of race & ethnicity on an equal footing. The results of this test can also be observed from the last row value in **Table 6**.

Bots assigning issues to White developers more than their actual representation in collaborator community was statistically significant.

7.3 Relationship between issue types and assignee's race & ethnicity

For **RQ.2** we need to understand the different types of issue assignments from the perspective of Assignee's race & ethnicity. As explained previously, we also gathered the type of an issue for all the projects. The types of issue we captured are Bugs, Feature, Test, Build, Docs, Other. For every Issue Type in the data set, we further grouped them as assigned to White or Non-White developers. We normalized this data by dividing by total White and Non-White assignees. We performed a Mann Whitney Test for each Issue Type comparing normalized White and Non white assignees. This was done for all the projects separately and then finally, for combined data set. The results can be seen from Table 7.

Since this data is normalized (eg. White assignees per issue type divided by total white assignee) the statistically significant data shows that white developers are over represented for each Bugs, Tests and Documentation compared to actual representation in the developer community. Only for these Issue types the data was statistically significant.

7.4 Issue assignments relationship with commit experience and race & ethnicity

For **RQ.3**, regression modelling is used to analyze influence of developers perceptible ethnicity on issues assigned based on commit experience. To understand the effect of the perceptible ethnicity and commit experience (independent variables) on the issues assigned (dependant variables), we built a mixed effect regression model using python's stats model package [20]. We used Repository as random effects. We used Mixed effect Regression instead of logistic regression models because they can capture measurements within same group (repository) as a random effect.

Variable	Unique Values	Lowest Value	Highest Value	Mean	Std. Deviation
Commits	419	0	57279	176.17	1417.33
Issues	262	1	5069	76.54	352.44

Table 8: Assigned Issue and Commit experience aggregate statistics

Variable	10th Perc.	25th Perc.	50th Perc.	75th Perc.	90th Perc.
Commits	0	0	13	94	354
Issues	1	1	4	24	114

Table 9: Assigned Issue and Commit experience distribution

We split the regression modelling into all four scenarios for White/Non White Assignors/Assignees. For each of these cases we count the total issues assigned to each assignee (dependant variable) and store their commit experience (independent variable). The Commit and Assigned Issue statistics are shown in Tables 8 and 9. Then we pass this data to the regression model and obtained results in Table 10. This shows approximately the number of extra issues assigned per unit commit of an assignee.

Assignor	Assignee	Model_Coefficient	Std.Error	p_value
W	N_W	-0.002	0.002	0.341
N_W	W	0.186	0.010	0.000 *
W	W	0.169	0.009	0.000 *
N_W	N_W	-0.003	0.005	0.599

Table 10: Mixed effect regression for issues assigned based on commit experience. This can be interpreted as the number of extra issues assigned per unit commit across repositories, Signif. codes: 0: *

We found that two cases were statistically significant: when white assignors assign to white assignees (Coefficient 0.169) and Non white assignor assign to white assignees (Coefficient 0.186). The other two cases were found to not be statistically significant. This data implies that earning one extra commit will allow more issues to be assigned for White assignees compared to Non White assignees. Interestingly, people of White ethnicity receive more issues per unit commit by Non white assignors compared to White assignors.

8 DISCUSSION

RQ.1 Assignors across all repositories favor assigning issues to their own ethnicity In section 7.1, we have seen that assignors prefer overwhelmingly to assign issues to those of their own ethnicity. Our original null hypothesis that even Non-White assignors would assign to white was rejected. This may even be in line with Huntington's classification of civilizations as indicated from previous research [21]. This can be confirmed by factoring location into the data along with race information to get a more detailed picture on preferences for assigning issues. This is left for future work.

Similarly bots assign issues to White developers significantly more than their actual representation as discussed in Section 7.2. A more in depth study on source code of these bots and how they assign issues can shed light on these results and is left for future work. If bots assign issues based on past commits and the module of code to be modified it can lead to a self-fulfilling prophecy assigning issues to the same cluster of experienced developers. Automatic assignment tools need to be programmed to take into factor the demographics of the population and give more chances for developers of non White ethnicity to prove themselves through Issue assignment.

RQ.2 White developers are over-represented in assigned issues of Bugs, Tests, and Documentation Section 7.3 looked at representation of different races for different Issue Types. The conclusion from this section was that for Bugs, Test and Documentation White developers are assigned issues well over their actual representation within developer community. We are unsure of why only these categories were statistically significant for White developers

RQ.3 Single commit goes farther for a White developer than non white developer in getting issues assigned Section 7.4 found that one extra earned commit goes farther for a White assignee in getting issues assigned than for a non White assignee. These results indicate systemic bias within GitHub. This puts a stronger case for hiding social signals on GitHub to overcome these systemic biases. However such bias will still exist among developers working in the same company and who know the identity and ethnicity of their colleagues.

Non-White assignors were found to assign most issues per commit to white assignees, than any other pairing of races. This could be a side effect of perception of the information technology field being largely the expertise of developers of White ethnicity. More role models are required within the tech industry from all walks of life to change this perception of the computing field.

9 THREATS TO VALIDITY

NamSor was used for classification of data. Any inaccuracy in the classification will affect the hypothesis that are tested in the paper. The repositories chosen may also not be a good sample of all repositories on GitHub. However care was taken to choose repositories in different domains of technologies with high percentages of both White and Non-White developers. To classify issues into types we manually classified the labels for the issues. There can be a subjective error in classifying these labels. The authors cross-verified these classifications to come to a conclusion for the type of each Issue label.

10 CONCLUSION

We have found that there exists several systemic biases in the way GitHub issues are assigned in repositories. Assignors were found to assign issues mostly to assignees of same race & ethnicity. For Bugs, Documentation, Test issue types, White developers are over represented and Non-White developers are underrepresented. Bots assign issues to White developers more than their actual representation in collaborator community. Commit experience was found to be more helpful for a White assignee in getting issues assigned than for non White Assignee. The novelty of this work lies in the fact that these are systemic biases within GitHub. Our findings reinforce the need for further studies on ethnic diversity in Software Engineering for a healthy OSS community.

10.1 Replication package

The data-set we gathered for this work and the replication package can be found in https://github.com/Kishanthan/CS-846-Spring2021-Is-there-a-Race-Bias-in-GitHub-Issue-Assignment. This repository is currently private and access can be given upon request.

REFERENCES

[1] E D Canedo, Heloise Acco Tives, Madianita Bogo, Fabiano Fagundes 2019. Barriers Faced by Women in Software Development Projects (ICSE)
[2] Yi Wang, David Redmiles, 2019. Implicit Gender Biases in Professional Software Development: An Empirical Study (ICSE-SEIS)
[3] Amiangshu Bosu, Kazi Zakia Sultana, 2018. Diversity and Inclusion in Open Source Software(OSS) Projects: Where Do We Stand? (ICSE-SEIS)
[4] Josh Terrell, Andrew Kofink, Justin Middleton, Justin Middleton, Clarissa Rainear, 2016. Gender bias in open source: Pull request acceptance of women versus men (ICSE-SEIS)
[5] Garland, Stephan 2021. Examining gender bias of GitHub Pull Request comments with machine learning
[6] Brokmeier P. 2017. Project level effects of gender on contribution evaluation on GitHub. PeerJ Preprints 5:e2989v1 https://doi.org/10.7287/peerj.preprints.2989v1
[7] N. Imtiaz, J. Middleton, J. Chakraborty, N. Robson, G. Bai and E. Murphy-Hill, 2019. "Investigating the Effects of Gender Bias on GitHub" (ICSE)
[8] DATA USA https://datausa.io/profile/soc/software-developers-applications-systems-software
[9] Bram Adams,Foutse Khomh, 2020. The Diversity Crisis of Software Engineering for Artificial Intelligence (IEEE Software)
[10] Reza Nadri, Gema Rodriguez Perez, Meiyappan Nagappan, 2020. Insights into Non-Merged Pull Requests in GitHub:Is there Evidence of Bias Based on Perceptible Race?
[11] R. Nadri, G. Rodriguezperez and M. Nagappan, "On the Relationship Between the Developer’s Perceptible Race and Ethnicity and the Evaluation of Contributions in OSS," in IEEE Transactions on Software Engineering, doi: 10.1109/TSE.2021.3073773.
[12] B. Vasilescu, V. Filkov, and A. Serebrenik, “Perceptions of diversity on github: A user survey,” in 2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering.IEEE,2015, pp. 50–56
[13] GitHub REST API documentation https://docs.github.com/en/rest
[14] WebScraper https://webscraper.io/
[15] NamSor https://www.namsor.com/
[16] Self-selection bias https://en.wikipedia.org/wiki/Self-selection_bias
[17] VSCode Triage Bot https://github.com/vscode-triage-bot
[18] Tensorflowbutler https://github.com/tensorflowbutler
[19] Kubernetes CI Bot https://github.com/k8s-ci-robot
[20] https://www.statsmodels.org/stable/index.html
[21] Wei Dong, Kate Ehrlich, Michael M Macy, Michael Muller, "Embracing Cultural Diversity: Online Social Ties in Distributed Workgroups: in 2016 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering.IEEE,2015, pp. 50–56