CS680 Project Report: Finding the correlation between pull request features and contributors Gender

Amirreza Shamsolhodaei AMIRREZA.SHAMSOLHODAEI@UWATERLOO.CA
Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada

1. Introduction

Gender fairness is a significant issue that deserves attention in the software development industry, particularly in open-source software (OSS) projects, where women are underrepresented and face various challenges and biases. These projects result from the collaborative work of developers with diverse backgrounds, and the quality of their contributions should be the sole criteria for assessing their value to the project. However, studies have demonstrated a correlation between diversity factors, such as gender, and the acceptance or rejection of these contributions.

Recent research has suggested that gender discrimination exists in the software development industry, with women experiencing lower acceptance rates for their code contributions and receiving fewer comments and feedback on their work than their male counterparts. While this is a serious issue that must be addressed, it is also important to consider that some assumptions may be based on incomplete or inaccurate data.

Added to this is the massive under-representation of women in software development, according to a global software developer survey in 2022 by Statista website, women only account for a little over 5 percent of respondents. Gender imbalance in the tech industry can have negative consequences for innovation, diversity, and productivity, as well as for the well-being and career advancement of women.

By analyzing a dataset of more than 3.5 million pull requests with over 100 different features from GitHub, and examining the correlation between pull request features and contributor gender, we can gain a more nuanced understanding of the factors that influence gender diversity in OSS projects. Pull requests are a common way of contributing code to open-source projects, where developers can propose changes and receive feedback from other developers. Pull requests can reveal various features of software development, such as code quality, collaboration, communication, and feedback. This research can help us identify which features are more likely to lead to predicting whether a male or female contributed to the pull request process and create strategies that increase diversity and gender fairness in software development teams. Overall, this study can contribute to the ongoing conversation surrounding gender fairness in software development and provide valuable insights for creating more equitable and inclusive work environments.

1.1 The problem and why it is important

Gender fairness in software development is important for several reasons. Firstly, it promotes equal opportunities for all developers, regardless of gender. It ensures that contributions are assessed based on merit and not on the gender of the contributor. This not only benefits individual contributors but also leads to better software development outcomes as it allows for a wider range of perspectives and experiences to be brought to the table.

Secondly, gender fairness promotes inclusivity in the workplace. When developers feel that they are valued and respected, they are more likely to feel motivated and engaged in their work. This can lead

to higher job satisfaction, productivity, and retention rates. Additionally, a more diverse workforce can help create a culture of innovation and creativity as people from different backgrounds bring unique ideas and approaches to problem-solving.

Finally, gender fairness is crucial for ethical and social justice reasons. Discrimination in any form, including gender-based discrimination, is a violation of basic human rights. By creating a more equitable and inclusive software development industry, we can help create a society where everyone has an equal opportunity to succeed, regardless of their gender or any other characteristic.

1.2 Background and related works

Open Source Software (OSS) projects result from the collaborative work of developers with diverse backgrounds. Despite the fact that the quality of their contributions should be the sole criteria for assessing their value to the project, as it is discussed in the work by Nadri et al. (2022). studies have demonstrated a correlation between diversity factors, such as gender, and the acceptance or rejection of these contributions. Vasilescu et al. (2015a) work suggests that diversity in race and gender is beneficial for the software development industry and studies show that team diversity positively correlates with team output in a positive way. Terrell et al. (2016) highlighted the different acceptance rates of men and women in OSS community in detail and discovered some noteworthy outcomes. They found that gender bias could work both ways, with biases against men occurring in certain situations. Additionally, they identified that factors such as experience and the tendency of women to tackle immediate project issues could contribute to the gender bias observed in OSS

An example of the impact of gender bias is explored by Padala et al. (2022). They found that gender biases in tools such as GitHub's pull request review system and Stack Overflow's Q&A forum can lead to negative experiences for female newcomers, including increased scrutiny of their contributions and lower acceptance rates for their code.

Looking at the state of AI and its increasing prevalence in our daily lives, Mehrabi et al. (2021) suggests it is important to consider fairness in the design and engineering of artificial intelligence systems. The use of AI in sensitive environments where crucial decisions are made necessitates ensuring that such decisions do not exhibit discriminatory behavior towards specific groups or populations.

Generally, by looking at these studies, we can understand that by identifying these factors and implementing strategies to promote greater gender fairness in OSS projects, we can create more welcoming and inclusive environments for all developers, regardless of their gender.

2. Dataset

The dataset that was used for this project was provided by Zhang et al. (2020), This is a rich dataset with over 3.5 million pull requests and 118 features, such as contributor gender, contributor's experience, lines of codes changed, and a number of commits. they have used this dataset in their work Zhang et al. (2023) for the purpose of finding a correlation between the features discussed and the decision that was made for the pull request by the integrator of the pull request.

For this project, certain preparation needed to take place as my use case for this dataset was different. the preparation of the dataset involved a series of steps to ensure the quality and reliability of the data. These steps included removing incomplete data, reducing the number of features, and filling in missing values. These measures were taken to ensure that the analysis accurately reflects the correlation between pull request features and contributor gender in OSS projects.

The dataset for this study consisted of 3.5 million pull requests with 118 features. To ensure the reliability of the results, pull requests with no gender information were removed, resulting in a total of 2,653,541 pull requests being used for analysis. Of these, 90.16% were contributed by male

contributors, and only 9.84% were contributed by female contributors. This significant imbalance in gender distribution highlights the need for investigating gender disparities in OSS projects.

To prepare the data for analysis, categorical data was encoded into numerical data to better model them. To reduce the number of features, multiple techniques were applied. First, highly correlated features were identified using Pearson correlation, and one of the two features was removed. Features that the contributor had no control over were dropped and only those that the contributor had a role in were kept. Examples of these features include the number of lines of code changed, whether the contributor had any previous pull requests, their affiliation, and their nationality.

After conducting feature selection, 42 features were retained out of the initial 118 features. However, 13 of these features had missing values. For example, the "ci_exists" feature had 2,203 pull requests with missing values. To address this issue, the context of each feature and the number of missing values were carefully analyzed to determine the most appropriate approach to filling in these values. For most features, the distribution of the existing values was examined, and either the mean or median of the other values in the feature was used to fill in the missing values. In some cases, the context of the feature was also taken into consideration, and either 0 or the mean was assigned to the feature. Visualizations, such as figures, were used to help determine the distribution and appropriate values for filling in missing data. Overall, a careful and thorough approach was taken to ensure that the resulting dataset was as accurate and complete as possible.

3. Methods Used

For this project, I used machine learning techniques that are commonly used for classification tasks and are well-suited for analyzing high-dimensional datasets like the one in this project. Additionally, using a variety of techniques can help ensure that the results are robust and not dependent on any one particular model.

Support Vector Machine (SVM) is a machine learning algorithm that is commonly used for classification problems. I chose to use LinearSVC, which is a variant of SVM that is optimized for linear problems and has faster runtime. SVM works by creating a hyperplane that separates the data points into different classes. The algorithm then tries to maximize the margin between the hyperplane and the closest points to it from both classes. SVM is known for its ability to handle high-dimensional data, which is suitable for our dataset which has many features.

Logistic Regression is another classification algorithm that I used. It is a type of regression analysis that is used to predict binary outcomes. Logistic Regression works by estimating the probability of an event occurring, which is transformed into a binary outcome using a decision boundary. Logistic regression is a simple yet powerful algorithm that is easy to interpret, which makes it a good choice for this task and can help identify the most important features for predicting a contributor's gender.

Decision Trees are a type of algorithm that is used for both classification and regression problems. Decision Trees work by splitting the data into smaller subsets based on the values of the features. The algorithm then creates a tree structure that represents the different possible outcomes of the decision-making process. Decision Trees are easy to understand and interpret, which makes them a good choice for feature selection tasks.

Random Forest is an ensemble algorithm that combines multiple decision trees to improve the performance of the model. Random Forest works by creating multiple decision trees on random subsets of the data and features. The algorithm then combines the results of these trees to make a final prediction. Random Forest is known for its ability to handle noisy data and high-dimensional data, which is suitable for our dataset.

Dense Neural Network is a type of artificial neural network where all the nodes in each layer are connected to all the nodes in the previous and next layers. The network consists of an input layer,

one or more hidden layers, and an output layer. The activation function used in this project was ReLU (rectified linear unit) for the hidden layers and sigmoid for the output layer. The optimizer used was Adam, which is a variant of stochastic gradient descent. in this case, a dense neural network can help identify the most important features for predicting a contributor's gender.

4. Empirical Evaluation

In this section, I will provide the results of each model and look at the most important features based on each model. as well as compare metrics such as performance and complexity.
An important point to note is that there can be differences in feature importance between different models because each model uses a different algorithm and approach to make predictions. Different models may have different criteria for selecting and weighing features and may be more or less sensitive to certain features depending on the algorithm used.

The metrics that I used are a measure of how well a model performs overall, but they do not necessarily indicate how important each individual feature is in making predictions. A model with higher accuracy may not necessarily have more accurate or important feature importance measures than a model with lower accuracy. Therefore, it is important to examine the feature importance measures for each model individually, rather than relying solely on accuracy as an indicator of important features.

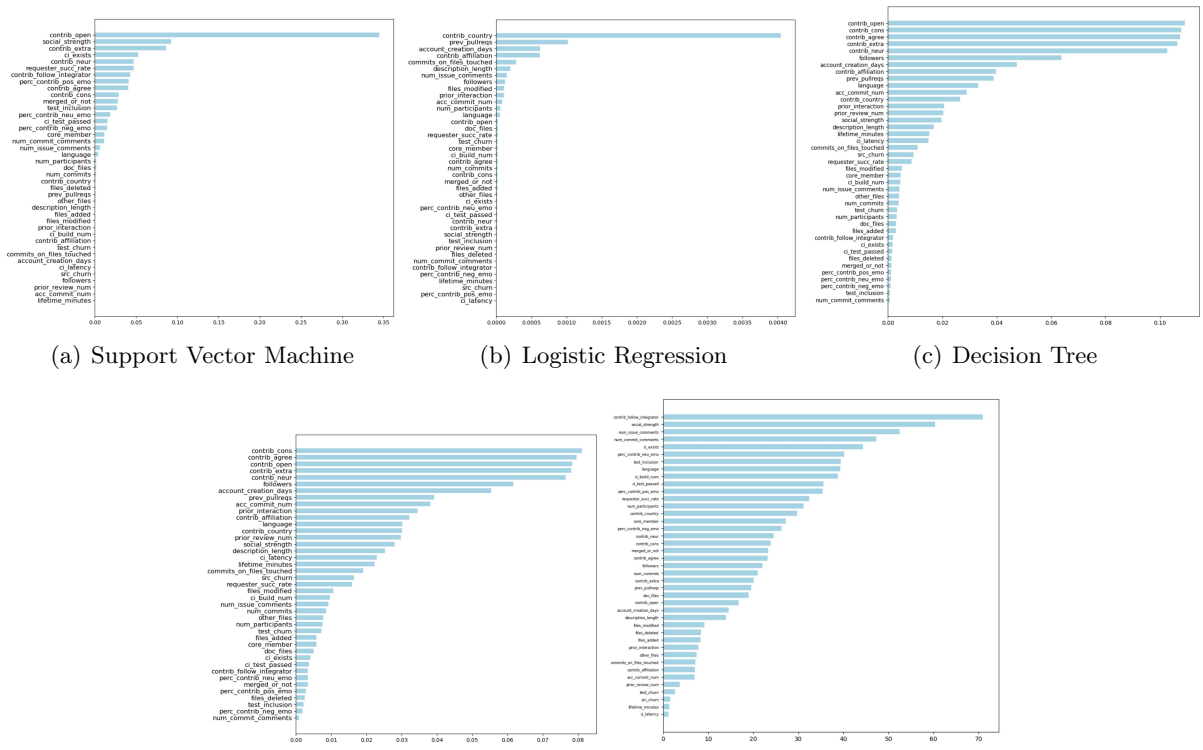
4.1 Results and Most Important Features

for the five different machine learning techniques that I implemented, here are the results, figure 1 shows the most important features according to each technique.
A. **Support Vector Machine:** In SVM, the algorithm tries to find the hyperplane that best separates the data into different classes. This hyperplane is defined by a set of weights (coefficients) assigned to each feature. therefore, The higher the absolute value of the coefficient, the more important that feature is in determining the predicted outcome.
With this in mind, I got the coefficient of each feature, and the top five important features are:

1. **contrib_open:** Openness of contributor based on big five personality traits, based on the tool developed by Iyer et al. (2021) based on previous comments on commits and pull requests.
2. **social_strength:** The fraction of repository team members who interacted with a developer over the team size in the last three months.
3. **contrib_extra:** Extraversion of contributor based on big five personality traits based on previous comments on commits and pull requests.
4. **ci_exists:** Whether a pull request uses continuous integration tools, based on the work by Vasilescu et al. (2015b).
5. **contrib_neur:** Neuroticism of contributor based on big five personality traits based on previous comments on commits and pull requests.

B. **Logistic Regression:** In logistic regression, a set of coefficients is used to model the relationship between the input variables (features) and the output variable (target). The coefficients are estimated through an optimization algorithm that aims to minimize the error between the predicted and actual target values. A higher coefficient means that the corresponding feature has a stronger impact on the prediction, making it more important.
With this in mind, I got the coefficient of each feature, and the top five important features are:

1. **contrib_country:** Contributor's Nationality.
2. **prev_pullreqs:** Number of previous pull requests of the contributor.
3. **account_creation_days:** The number of days that the contributor's account has been created.
4. **contrib_affiliation:** The company which the contributor's affiliated with.



(a) Support Vector Machine (b) Logistic Regression (c) Decision Tree

(d) Random Forest (e) Neural Network

Figure 1: Ranking of Important Features

5. **commits_on_files_touched:** Number of commits made on files that were touched (modified, added, or deleted) in the pull request.

C. **Decision Tree:** In the decision tree, the feature_importances_ is a built-in attribute in the scikit-learn decision tree method. This attribute calculates the importance of each feature in predicting the required feature. The higher the value of feature_importances_, the more important that feature is in the decision tree model. In other words, it means that the feature has a stronger relationship with the target feature and contributes more to the decision-making process in the model.

With this in mind, using the feature_importance_ attribute, the top five important features are:

1. **contrib_open:** Openness of contributor based on big five personality traits, based on previous comments on commits and pull requests.
2. **contrib_cons:** Conscientiousness of contributor based on big five personality traits, based on previous comments on commits and pull requests.
3. **contrib_agree:** Agreeableness of contributor based on big five personality traits, based on previous comments on commits and pull requests.
4. **contrib_extra:** Extraversion of contributor based on big five personality traits based on previous comments on commits and pull requests.
5. **contrib_neur:** Neuroticism of contributor based on big five personality traits based on previous comments on commits and pull requests.

D. **Random Forest:** In random forest, the feature_importance_ attribute in Scikitlearn represents the relative importance of each feature in the prediction task. It is a weighted average of the feature importance values computed for each decision tree in the forest. The feature importance values are computed as the total reduction of the impurity (e.g., Gini impurity or entropy) achieved by each

feature in all the decision trees of the forest. As usual, a higher feature_importance_ value for a feature in random forest indicates that the feature is more important in the prediction task.
With this in mind, using the feature_importance_ attribute, the top five important features are as below, it is noticeable that they are almost identical to the decision tree technique:

1. **contrib_cons:** Conscientiousness of contributor based on big five personality traits, based on previous comments on commits and pull requests.
2. **contrib_agree:** Agreeableness of contributor based on big five personality traits, based on previous comments on commits and pull requests.
3. **contrib_open:** Openness of contributor based on big five personality traits, based on previous comments on commits and pull requests.
4. **contrib_extra:** Extraversion of contributor based on big five personality traits based on previous comments on commits and pull requests.
5. **contrib_neur:** Neuroticism of contributor based on big five personality traits based on previous comments on commits and pull requests.

E. **Dense Neural Network:** To identify the most important features in the dataset, I created several neural network models using PyTorch torch.nn. I selected PyTorch due to its usage during the course and its efficient use of GPU. As the dataset was highly unbalanced, I tried using both SMOTE and random over sampler from the imblearn library to determine their impact on model performance. This was done because the imbalance of the dataset could have resulted in a biased model.
During the experiments, the performance of the model was evaluated based on its accuracy and F1 score. After testing various neural network architectures and hyperparameters, the model without using any oversampling techniques achieved the best performance which could be due to various reasons, One possible reason is that oversampling methods such as SMOTE and random oversampler may create synthetic samples that do not accurately represent the original data distribution, leading to overfitting or underfitting. Another reason could be that oversampling techniques may increase the complexity of the dataset, making it harder for the neural network to generalize to unseen data. Additionally, oversampling techniques may also increase the risk of creating noisy data and redundant information, which may negatively impact the performance of the model.

To identify the most important feature, the weight assigned to each feature in the neural network model was analyzed. In neural networks, the weight assigned to each feature determines its importance in predicting the output.

Here, I will share the results of the most important features from the best-performing neural network model without using oversampling techniques:

1. **contrib_followintegrator:** Whether the contributor follows the integrator of the pull request.
2. **social_strength:** The fraction of team members who interacted with a developer over the team size in the last three months.
3. **num_issue_comments:** Number of comments in the Issue
5. **num_commit_comments:** Number of comments in the commit
5. **ci_exists:** Whether a pull request uses continuous integration tools.

4.2 Evaluation

4.2.1 Performance

I used Scikit Learn accuracy_score, precision_score, recall_score, f1_score for the performance metrics. Table 1 shows how different models performed according to these metrics.
The accuracy score is a commonly used metric for evaluating the overall performance of a classification model, the precision score is a metric that evaluates the proportion of true positive predictions among

Model	Accuracy	Precision	Recall	F1 Score
Support Vector Machine	0.8970	0.8319	0.8970	0.8551
Logistic Regression	0.9025	0.9120	0.9025	0.8572
Decision Tree	0.9691	0.9695	0.9692	0.9693
Random Forest	0.9776	0.9781	0.9776	0.9763
Neural Network (no oversampling)	0.9057	0.910	0.9057	0.8651
Neural Network (with SMOTE)	0.6744	0.8505	0.6744	0.7382
Neural Network (with Random Oversampling)	0.7089	0.8553	0.7089	0.7634

Table 1: Performance of Different Models

all positive predictions made by the model, the recall score is a metric that evaluates the proportion of true positive predictions among all actual positive instances in the dataset and the F1 score is a metric that combines both precision and recall scores to provide an overall measure of the model's performance. By comparing these metrics, it could be determined which model performed the best overall.

From the results, we can see that both Decision Tree and Random Forest models had high accuracy, precision, recall, and F1 score. The neural network model without oversampling techniques also had good performance with an accuracy of 0.9057 and an F1 score of 0.8651. However, the models that utilized oversampling techniques (SMOTE and Random Oversampling) did not perform as well as the others, with significantly lower accuracy and F1 score. SVM and Logistic Regression had a good performance but were slightly outperformed by the tree-based models. Overall, we can conclude that Decision Tree and Random Forest models may be good choices for this dataset.

This result suggests that based on performance, random forest, and decision trees might be better suited for this task. Regarding feature importance, decision trees, and random forest are known for their ability to perform feature selection and have an easily interpretable feature importance metric. Therefore, their feature-importance results are generally considered more credible than other models, which is showcased here.

4.2.2 Complexity

Here, I will compare time and space complexity and the runtime of each method in my local computer. note that neural network models were done on Colab environment as my laptop's GPU could not handle PyTorch and training the model.

Logistic Regression: This is a linear model and relatively simple to compute, which explains its short runtime of 2 minutes. It has a time complexity in big O notation of $O(nd^2 + nd^3)$ where n is the number of samples and d is the number of features and space complexity of $O(p)$ where p is the number of features.

Decision Tree: This model builds a tree-based structure to make predictions, which is more complex than logistic regression, but still relatively straightforward to compute. It took 2 minutes to run, similar to logistic regression. It has a time complexity in big O notation of $O(ndlog(n))$ where n is the number of samples and d is the number of features and space complexity of $O(nd)$ where n is the number of samples and d is the depth of the tree.

Support Vector Machine (SVM): This model tries to find a hyperplane that maximally separates the classes, which can be computationally intensive. Therefore, it took 14 minutes to run, longer than logistic regression and decision tree. It has a time complexity in big O notation of $O(n^2d + n^3d)$ where n is the number of samples and d is the number of features and space complexity of $O(n)$ where n is the number of samples.

Random Forest: This model uses a collection of decision trees and combines their predictions, which can be computationally expensive as well. It took 12 minutes to run, similar to SVM. It has a time complexity in big O notation of $O(mndlog(n))$ where m is the number of trees in the forest, n is the number of samples, and d is the number of features and space complexity of $O(nm)$ where n is the number of samples and m is the number of trees in the forest.

Neural Network: This model is the most complex among the models featured, as it involves training multiple layers of interconnected neurons. It took around 20 minutes to run, the longest among the models. It has a time complexity in big O notation of $O(kn^3)$ where k is the number of layers in the network and n is the number of nodes in each layer and space complexity of $O(nd)$ where n is the number of samples and d is the number of hidden layers in the network.

5. Future Work

This work only touched upon the issue of gender fairness and gender imbalance in software development and OSS environments, there is a lot of other work that needs to be done to encapsulate this problem. In this work I only looked at the features that correlate and influence the prediction of the contributor's gender, other studies could look at the impact of other factors such as age, and education level as well as how this imbalance and unfairness needs to be dealt with.
Another way for improvement and future work would be to fine-tune the models that were created. Due to time constraints, especially regarding neural network models, although I tried different models (more than the three mentioned here) with different numbers of layers and nodes and different optimizers, the models leave a lot to be desired and by tuning the hyperparameters, trying different models with different structures, different optimizers, and changes as such, much-improved models could be offered. added to this is the fact that the dataset could be managed and filtered in other ways which would be more efficient and provide better results.

6. Conclusion

In conclusion, the evaluation of different machine learning techniques to showcase the most important features in predicting contributors' gender in open-source software projects has been presented. Among the evaluated techniques, Random Forest achieved the highest accuracy, precision, recall, and F1 score. In terms of complexity, logistic regression and decision tree were the fastest to run, while SVM and neural network models took significantly longer. When considering the space complexity, all the models had a similar performance since they did not require significant memory usage.

While this study provides insights into the performance of different machine learning techniques, it is important to note that further research is needed to evaluate the generalizability of these findings across different datasets and open-source projects. Moreover, future work can explore the correlation between features and contributors' gender in greater detail to identify new features that could improve the accuracy of the models.

To answer the question of which technique is the best, it can be argued that the Random Forest model achieved the highest performance in predicting contributors' gender. However, it is important to note that the choice of technique depends on the specific requirements and characteristics of the dataset, and no single technique is universally good enough to declare the problem solved. Therefore, the selection of the best technique depends on the trade-off between accuracy, complexity, and other factors that are specific to the project at hand.

References

Decision Tree. URL https://scikit-learn/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

Rahul N. Iyer, S. Alex Yun, Meiyappan Nagappan, and Jesse Hoey. Effects of personality traits on pull request acceptance. *IEEE Transactions on Software Engineering*, 47(11):2632–2643, Nov 2021. ISSN 0098-5589, 1939-3520, 2326-3881. doi: 10.1109/TSE.2019.2960357. URL https://ieeexplore.ieee.org/document/8935389/.

LinearSVC. URL https://scikit-learn/stable/modules/generated/sklearn.svm.LinearSVC.html.

Logistic Regression. URL https://scikit-learn/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys*, 54(6):1–35, jul 2021. doi: 10.1145/3457607. URL https://doi.org/10.1145%2F3457607.

Reza Nadri, Gema Rodriguez-Perez, and Meiyappan Nagappan. On the relationship between the developer's perceptible race and ethnicity and the evaluation of contributions in OSS. *IEEE Transactions on Software Engineering*, 48(8):2955–2968, aug 2022. doi: 10.1109/tse.2021.3073773. URL https://doi.org/10.1109%2Ftse.2021.3073773.

Hema Susmita Padala, Christopher Mendez, Felipe Fronchetti, Igor Steinmacher, Zoe Steine-Hanson, Claudia Hilderbrand, Amber Horvath, Charles Hill, Logan Simpson, Margaret Burnett, Marco Gerosa, and Anita Sarma. How gender-biased tools shape newcomer experiences in oss projects. *IEEE Transactions on Software Engineering*, 48(1):241–259, Jan 2022. ISSN 0098-5589, 1939-3520, 2326-3881. doi: 10.1109/TSE.2020.2984173. URL https://ieeexplore.ieee.org/document/9055190/.

Random Forest. URL https://scikit-learn/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

random over sampler. URL https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.RandomOverSampler.html.

SMOTE. URL https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html.

Statista. URL https://www.statista.com/statistics/1126823/worldwide-developer-gender/.

Josh Terrell, Andrew Kofink, Justin Middleton, Clarissa Rainear, Emerson Murphy-Hill, and Chris Parnin. Gender bias in open source: Pull request acceptance of women versus men. feb 2016. doi: 10.7287/peerj.preprints.1733v1. URL https://doi.org/10.7287%2Fpeerj.preprints.1733v1.

torch.nn. URL https://pytorch.org/docs/stable/nn.html.

Bogdan Vasilescu, Daryl Posnett, Baishakhi Ray, Mark G.J. van den Brand, Alexander Serebrenik, Premkumar Devanbu, and Vladimir Filkov. Gender and tenure diversity in GitHub teams. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, apr 2015a. doi: 10.1145/2702123.2702549. URL https://doi.org/10.1145%2F2702123.2702549.

Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. Quality and productivity outcomes relating to continuous integration in github. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, page 805–816, Bergamo Italy, Aug 2015b. ACM. ISBN 9781450336758. doi: 10.1145/2786805.2786850. URL https://dl.acm.org/doi/10.1145/2786805.2786850.

Xunhui Zhang, Ayushi Rastogi, and Yue Yu. On the shoulders of giants. In *Proceedings of the 17th International Conference on Mining Software Repositories*. ACM, jun 2020. doi: 10.1145/3379597.3387489. URL https://doi.org/10.1145%2F3379597.3387489.

Xunhui Zhang, Yue Yu, Georgios Gousios, and Ayushi Rastogi. Pull request decisions explained: An empirical overview. *IEEE Transactions on Software Engineering*, 49(2):849–871, feb 2023. doi: 10.1109/tse.2022.3165056. URL https://doi.org/10.1109%2Ftse.2022.3165056.