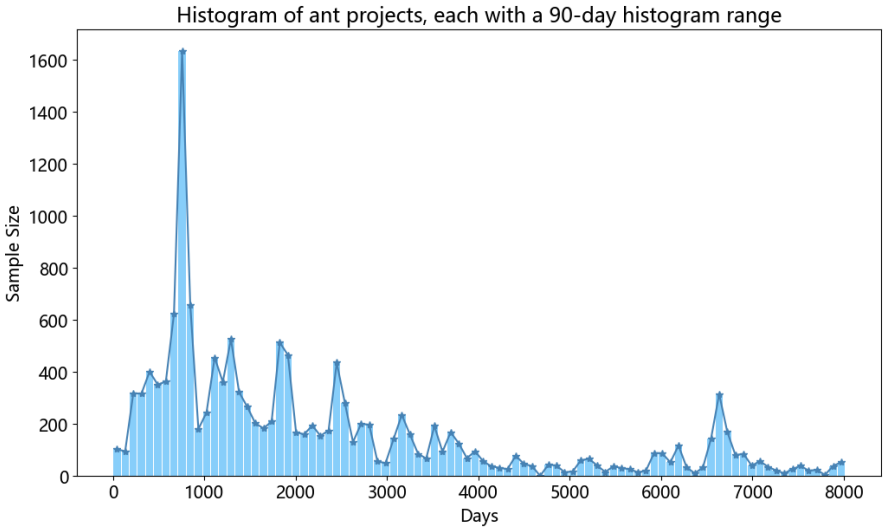

FIGURE 1 Histogram of commit activity over the life of the project

1 | APPLICATION SCENARIO DISCUSSION

(1) Project Activity Level The datasets used in this paper are from open-source projects on GitHub. In real-world development projects, the activity level of a project can vary with the project's progress. For example, when new feature requirements arise, project development becomes more active, and the frequency of commits increases. However, the project may become less active without new requirements, entering a quiet period with reduced commit frequency. As shown in Figure 1, taking the ant project as an example, it is evident from the histogram that the number of commits in the ant project fluctuates over days, exhibiting clear peaks and valleys. Peaks indicate that the project enters an active phase during that period, while valleys suggest that the project goes into a quiet phase.

The effectiveness of the TWAO method is closely related to the activity level in project commits. When a project follows a relatively regular development cycle and has development and testing teams maintaining and updating the system periodically, the TWAO method can fully leverage its advantages. It uses a time-aware mechanism to balance the distribution of defective and non-defective categories to improve prediction performance. In turn, this better assists the team in task allocation, optimizes resource utilization, and enhances overall development efficiency.

In some cases, such as when a project has an irregular development cycle, experiences extended periods of inactivity or has a relatively complex development process, the applicability of the TWAO method may be limited. This paper used the same parameter configuration for different projects. When dealing with projects with irregular development cycles, adjust the parameters according to the specific characteristics of that project to achieve better results.

(2) Frequency of Model Iterations

In the research on JIT-SDP[1 2 3], researchers have used sample quantities as a basis for dividing data, such as using a fixed proportion of the dataset as the training set to predict the remaining samples. Considering that in practical development processes, testing personnel may update the defect prediction model when a sufficient number of commits have been made, this approach also has practical significance. In other words, model updates are more frequent when a project is active, whereas when the project is inactive, model updates occur at a slower pace.

This paper discusses the impact of the frequency of model iterations on the TWAO method. Figure 2 shows the relationship between different proportions of the dataset used for training and model performance for all projects. The x-axis represents the ratio of the number of samples used in each iteration to the total number of samples, ranging from $1/10$ to $1/2$, and the y-axis represents the predictive performance on the 4 metrics. Since the size of project datasets is fixed, the more significant the proportion of the dataset used in each iteration, the more data is input for each model iteration, resulting in fewer iterations. Conversely, smaller proportions lead to fewer data inputs and more iterations. During each model iteration, the TWAO method assigns appropriate time weights. Experimental results indicate that model performance increases as the training sample ratio in each iteration decreases. It implies that more frequent model iterations lead to higher model performance under the

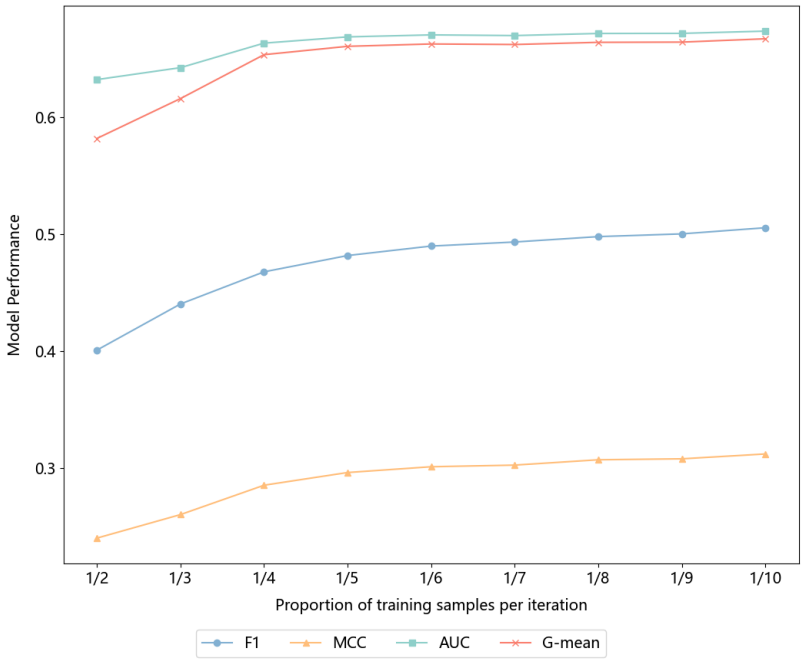

FIGURE 2 The relationship between iteration frequency and model performance

same conditions of dataset duration and the total number of samples. Therefore, this demonstrates that the TWAO method is more suitable for projects that require more frequent model updates.

Based on the two aspects discussed above, it is evident that the TWAO method performs better in scenarios where the development cycle is relatively regular and the system can be maintained and updated periodically. Increasing the frequency of model iterations can lead to improvements in prediction performance. Therefore, this paper recommends selecting projects with higher commitment activity in realistic software development environments. When testing resources allow, updating defect prediction models on time can enhance the model's awareness of time, resulting in better prediction performance.

References

1. Hoang T, Dam HK, Kamei Y, Lo D, Ubayashi N. DeepJIT: an end-to-end deep learning framework for just-in-time defect prediction. In: IEEE. ; 2019: 34–45.

2. Huang Q, Xia X, Lo D. Revisiting supervised and unsupervised models for effort-aware just-in-time defect prediction. *Empirical Software Engineering* 2019; 24: 2823–2862.

3. Kamei Y, Fukushima T, McIntosh S, Yamashita K, Ubayashi N, Hassan AE. Studying just-in-time defect prediction using cross-project models. *Empirical Software Engineering* 2016; 21: 2072–2106.