Analyzing GitHub Issues in the Rails Project

Authored by :
Borhaneddine **Hamadou**

January, 31th 2024

1 Introduction

The Rails project is a popular open-source web application framework that has been widely adopted by developers around the world. As with any open-source project, effective issue management is crucial for maintaining a robust development ecosystem. In this document, we embark on a comprehensive analysis of GitHub issues within the context of the Rails project. Through our explorations, we aim to unravel the intricate interplay of labels, temporal dynamics, and contributors within the Rails project, laying the groundwork for further insights. Our study seeks to address various questions related to issue management, including label-related, temporal dynamics, and user-related inquiries.

2 Problem statement

This study aims to address various questions related to issue management, seeking to gain valuable insights from the analyzed findings. To streamline our inquiries, we have categorized them into three distinct groups : *label-related*, *temporal dynamics*, and *user-related* questions.

For *label-related* questions, we focus on three primary inquiries :

- What is the closing rate of issues categorized with no labels, a single label, or multiple labels ?
- What is the predominant labeling category for issues in the Rails project ?
- Are certain labels more frequently assigned than others ?
- Do specific labels commonly occur together in issue assignments ?

In the realm of *temporal dynamics* questions, our interest lies in exploring and answering two key queries :

- How does the distribution of issue closure times vary among those with no labels, a single label, or multiple labels ?
- What patterns emerge in the behavior of issues over time, specifically examining their evolution across days of the week and throughout different hours of the day ?

Furthermore, we delve into user engagement, investigating whether certain users contribute more significantly to issue creation than others.

Lastly, a pivotal aspect of this work involves classifying issues based on their labels derived from their descriptions. To accomplish this, we employed fine-tuning and evaluation processes using the renowned BERT model tailored to our specific set of Rails project issues.

3 Working methodology

The methodology employed in this study consists of five distinct stages :

1. ***Extraction of issue information :*** This initial stage involves the extraction of rails project issues informations from github.
2. ***Pre-process dataset :*** Following the extraction, the dataset undergoes a pre-processing phase. This involves cleaning and organizing the data to ensure its readiness for subsequent analysis.
3. ***General overview :*** The third stage provides a broad and initial overview of the dataset, enabling a preliminary understanding of its composition, characteristics, and any noticeable patterns.
4. ***Deeper exploratory analysis :*** Building upon the general overview, this stage delves deeper into the dataset, employing advanced analytical techniques to specifically address the predefined problematic questions.
5. ***Classification model evaluation :*** In the final stage, the study assesses the performance of the classification model. This involves fine-tuning and evaluating the BERT model specifically tailored for classifying issues based on their labels derived from descriptions.

The diagram in Figure 1 succinctly captures the essence of the five stages comprising the employed methodology.

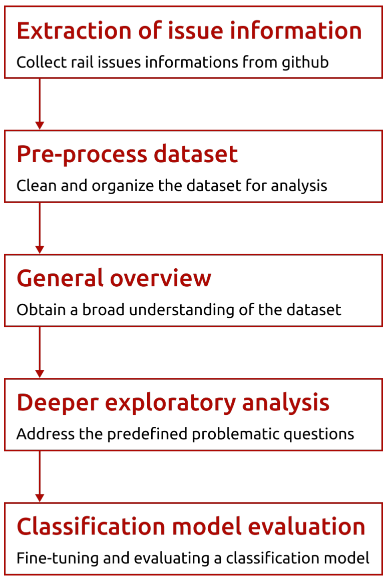
Extraction of issue information
Collect rail issues informations from github
Pre-process dataset
Clean and organize the dataset for analysis
General overview
Obtain a broad understanding of the dataset
Deeper exploratory analysis
Address the predefined problematic questions
Classification model evaluation
Fine-tuning and evaluating a classification model

FIGURE 1 : Sequential stages of the employed methodology

4 Extraction of issue information

The first stage of our methodology involves a systematic process to obtain the necessary data for our study. We used a methodical approach to issue information extraction, composed of the following steps :

1. **Extracting issues using gitHub API :** Utilize the gitHub API's issues endpoint to retrieve the most recent 500 issues, encompassing both open and closed issues.

2. **Selecting relevant information :** Extract the following key details from the retrieved issues :
    - *number :* The issue number, serving as a unique identifier.
    - *'title' :* The title of the issue.
    - *'state' :* Indicates whether the issue is open or closed.
    - *'created at' :* The date of issue creation.
    - *'closed at' :* The date of issue closure (if applicable).
    - *'user' :* The user responsible for creating the issue.
    - *'labels' :* An array of labels associated with the issue (if any).
    - *'description' :* The body of the issue.
3. **Creation of Pandas dataframe :** Organize the extracted information into a structured format by creating a Pandas dataframe.

5 Pre-process dataset

The second stage of our methodology focuses on preparing and pre-processing the dataset to ensure it is optimized for subsequent in-depth analysis. Each step serves a specific purpose in enhancing the dataset's usability and extracting relevant insights.

1. **Handle labels :**
    - **Purpose :** Simplify label representation for ease of analysis.
    - **Description :** In the labels column, retain only the label names from each label object in the array. This step streamlines the label information, making it more accessible and manageable for subsequent analysis.
2. **Convert 'created at' and 'closed at' columns to datetime format :**
    - **Purpose :** Enhance the dataset's time-related analysis capabilities.
    - **Description :** Convert the *'created at'* and *'closed at'* columns to datetime format. This conversion ensures a standardized time representation, facilitating accurate and comprehensive time-based analyses.
3. **Extract additional timestamp features :**
    - **Purpose :** Enable detailed time-based analysis.
    - **Description :** Extract additional features, including the day of the week, month, and year, from *'created at'* column. These extracted features enhance the dataset's granularity, allowing for more nuanced temporal insights.
4. **Text preprocessing for descriptions :**
    - **Purpose :** Refine text data for improved modeling accuracy.
    - **Description :**

  - Remove comments from the description (text between '<!–' and '–>').
  - Eliminate markdown formatting symbols like headings, images, links ('\r', '\n*', '\#').
  - Remove stop-words (e.g., 'a', 'the', 'that') to focus on more meaningful content.
  - Perform stemming on words to simplify them and obtain their roots. This step aids in reducing complexity and ensuring essential information is retained for subsequent modeling.

5. **Save processed dataset :**
   - **Purpose :** Preserve the pre-processed dataset for future analyses.
   - **Description :** Save the pre-processed dataset, now refined and enhanced, as a (.csv) file. This step ensures that the optimized dataset is readily available for subsequent stages of the study.

6 General overview

This stage provides a snapshot of the dataset's key characteristics :

1. **Dataset overview :**
   - The dataset, compiled as of January 30, 2024, at 15 :44 pm, provides valuable insights into the Rails project. Given the dynamic nature of the project, continuous issue creation is anticipated.
   - The dataset comprises 500 rows and 11 columns, detailing 376 closed issues (75.2%) and 124 open issues.
   - 9 within the dataset lack accompanying descriptions.
2. **Label-related observations :**
   - 87 issues lack any labels.
   - 101 issues have multiple labels, underscoring the complexity of topics in the Rails project.
   - 20 different label values, showcasing the diverse nature of topics.
   - Maximum number of labels on one issue is 13.
3. **Temporal observations :**
   - Issues cover a 40-day period, from December 21, 2023, to January 30, 2024.
4. **User-related observations :**
   - Impressively, 196 distinct users actively contributed to issue creation, emphasizing the collaborative spirit of the Rails community.

7 Deeper exploratory analysis

Continuing our exploration, this stage delves into the analytics to address the predefined problematic questions.

7.1 Close rate of issues for different labels

To better understand the impact of labels on issue closure, we categorize issues into three types : no-label, single-label, and multi-label.

The table 1 illustrates the close rates of issues categorized by the number of labels, including issues with no labels, single labels, and multiple labels.

Issue Type	**Number of Issues**	**Close Rate**
No-label	87	81.61%
Single-label	312	73.72%
Multi-labels	101	74.26%

TABLE 1 : Close rate of issues for different labels

No-label issues display a high close rate of 81.61%, indicating effective resolution despite lacking specific labels. This could suggest that many issues, although not categorized, are promptly addressed.

Single-label issues show a close rate of 73.72%, indicating slightly lower closure efficiency compared to no-label issues. This may imply that issues with a single label might exhibit varied complexities, influencing their resolution time.

Multi-label issues demonstrate a close rate of 74.26%, similar to single-label issues. This finding suggests that, despite potential complexities associated with multiple labels, the close rate remains comparable.

Q1 : What is the closing rate of issues categorized with no labels, a single label, or multiple labels ?

A1 : The analysis of the closing rates for issues categorized by labels reveals insightful patterns within the Rails project. The notably high close rate for no-label issues prompts considerations about the efficiency of issue resolution without explicit categorization.

7.2 Issue Distribution According to the Number of Labels

The figure 2 illustrates the number of issues based on the number of labels assigned. Issues with a single label dominate, exceeding 300, while those with no label or two labels surpass the count of issues with three labels or more. This distribution suggests that while a single label is most commonly used, issues with no label and two labels are also prevalent. In contrast, issues with more than three labels are infrequent.

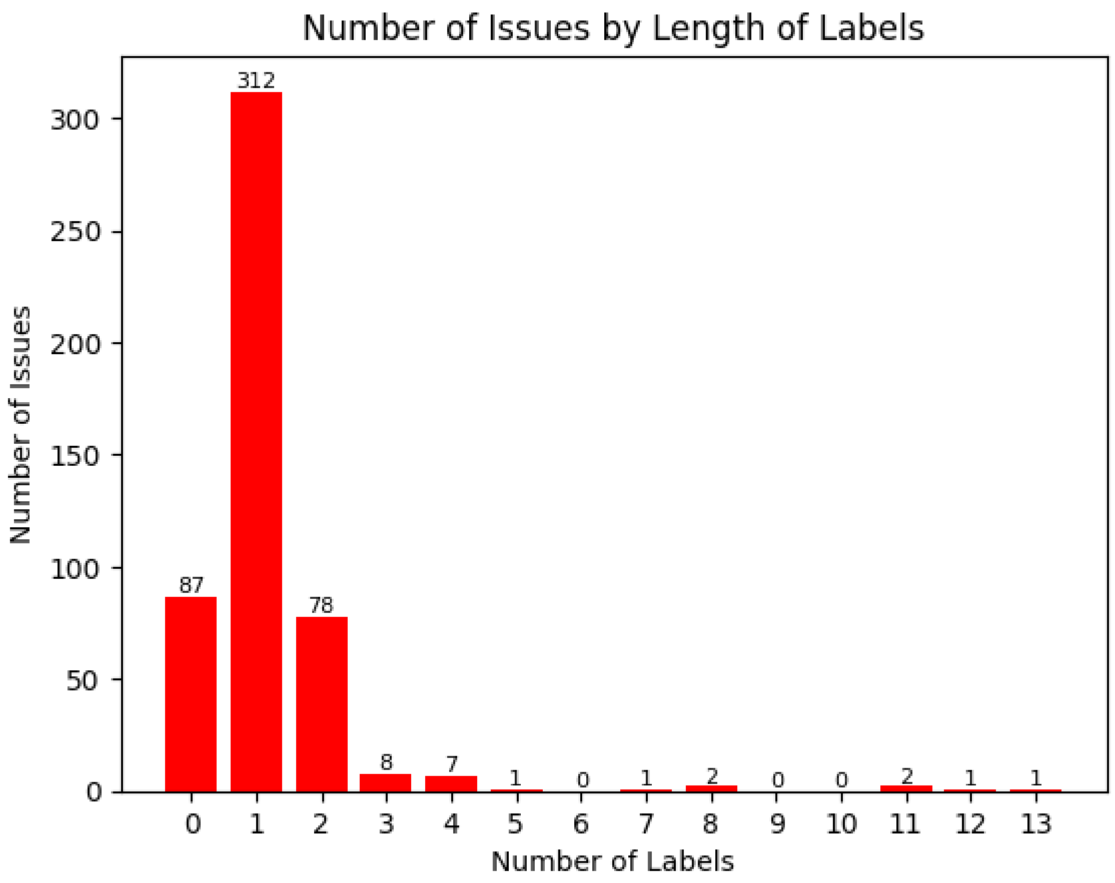

FIGURE 2 : Number of issues by length of labels

Q2 : What is the predominant labeling category for issues in the Rails project ?

A2 : The distribution of issues according to the number of labels highlights the prevalent use of a single label for categorization. Additionally, the frequency of issues with no label or two labels suggests flexibility and diversity in the labeling approach. Conversely, issues with more than three labels are relatively rare, indicating a less common and more intricate categorization practice within the Rails project.

7.3 Label frequencies

Illustrated in Figure 3, the frequencies of the 20 used labels provide a visual representation of the diverse categorizations employed within the Rails project.

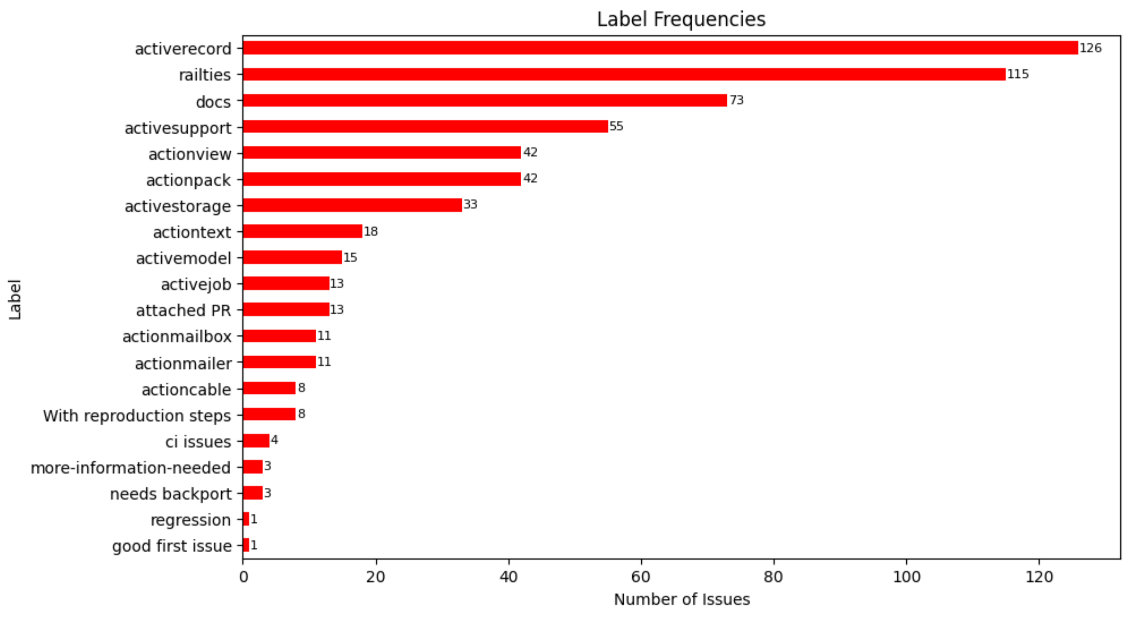

FIGURE 3 : Frequencies of the 20 used labels in Rails project

- **activerecord and railties :** Predominant labels, indicating a significant focus on enhancing and extending the core components of the Rails framework.
- **Docs Label :** Significantly present, showcasing a commitment to maintaining comprehensive documentation, crucial for the Rails community.
- **activesupport, actionview, actionpack :** Demonstrate sustained attention to fundamental aspects of Rails development.
- **Diverse Labels (email handling, background jobs, real-time communication, file management) :** Emphasize the breadth of functionalities supported by Rails.
- **With reproduction steps and ci issues :** Indicate a meticulous approach to issue reporting and testing, contributing to the overall reliability of the framework.
- **Backporting changes and addressing regressions :** Reflect a commitment to supporting various Rails versions.
- **good first issue :** Represents an inclusive approach, welcoming new contributors to the Rails community.

The label frequencies offer insights into the multifaceted nature of Rails development. They showcase a balance between core improvements, documentation, collaboration, and quality assurance.

Q3 : Are certain labels more frequently assigned than others ?

A3 : Yes, certain labels, such as activerecord, railties, and docs, demonstrate higher frequencies, signifying their significance in the Rails project. The predominance of activerecord and railties indicates significant focus on enhancing and extending the core components of the Rails framework.

7.4 Labels co-occurrence analysis

Now, the objective is to observe if, in multi-label issues, there are some labels that are usually used together. Figure 4 illustrates label co-occurrence percentages.

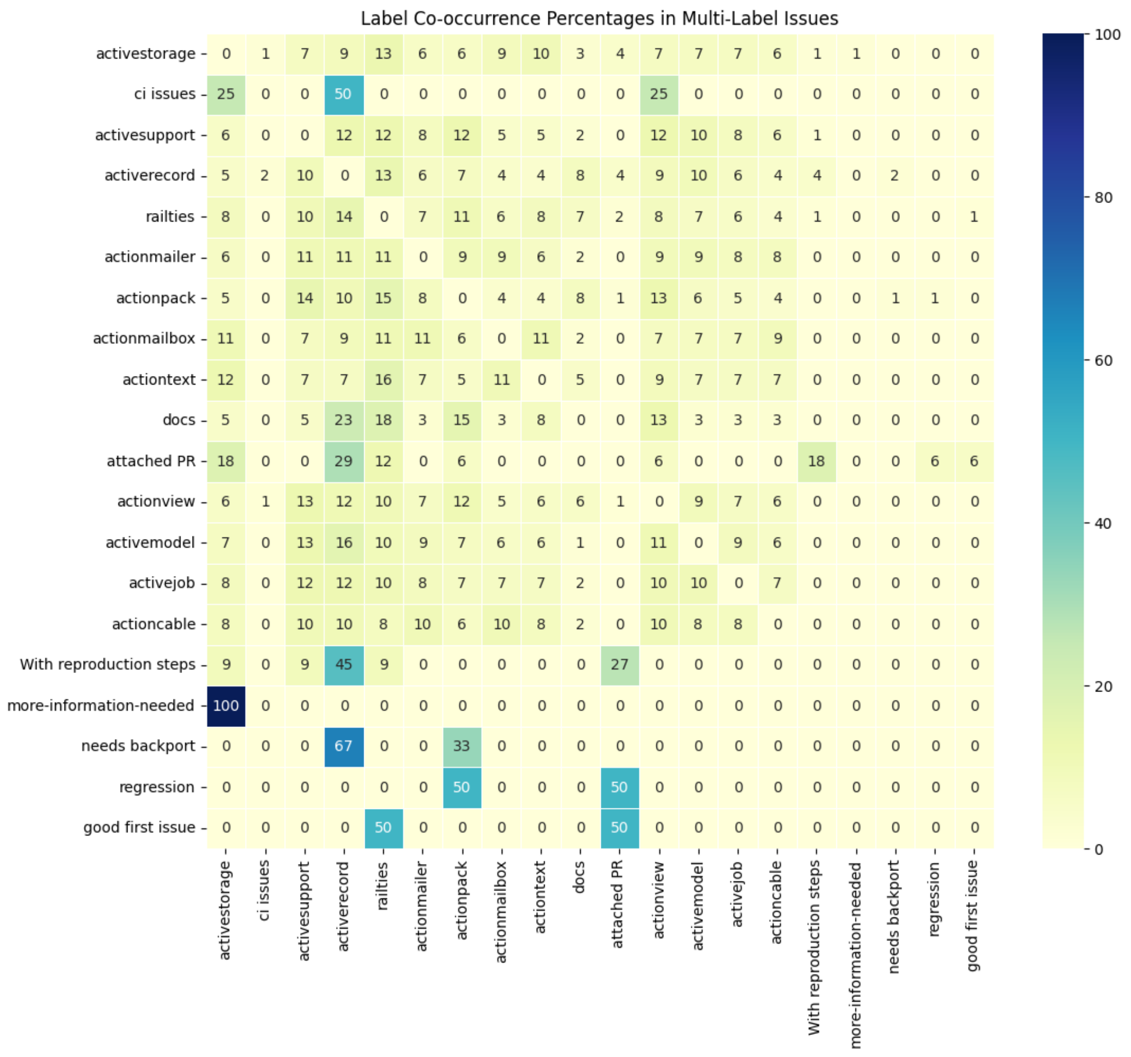

FIGURE 4 : Label co-occurence persentage in multi-label issues

- **'docs' label :** Around 70% of occurrences are coupled with other prominent labels, including 'activerecord', 'railties', 'actionpack', and 'actionview'. This substantial association indicates the collaborative nature of documentation efforts with core components of the framework.
- **'More information needed' label :** Appears 3 times and is consistently associated solely with the 'activestorage' label. This indicates that contributors often require additional information or details specifically for issues related to 'activestorage'.
- **'activerecord' and 'railties' :** Emerge as the most frequently appearing labels, with 126 and 115 occurrences, respectively. The co-occurrence analysis reveals a strong relationship, appearing together in more than 13% of their occurrences, suggesting a significant association between these two labels. This association implies that these components are central to ongoing development efforts.

- **'activesupport' label :** Notably related to 'activerecord' and 'railties', co-occurring in more than 12% of its occurrences with each, which is the highest co-occurrence percentage with any other label, indicating a substantial connection.

Q4 : Do specific labels commonly occur together in issue assignments ?

A4 : Yes, the label co-occurrence analysis unveils patterns of consistent associations between certain labels, reflecting collaborative efforts, dependencies, and thematic connections within the Rails project.

7.5 Close time distribution of issues

On this part, we focus on studying the average time to close for no-label, single-label, and multi-label issues. Figure 5 illustrates the time-to-close distribution for different label groups.

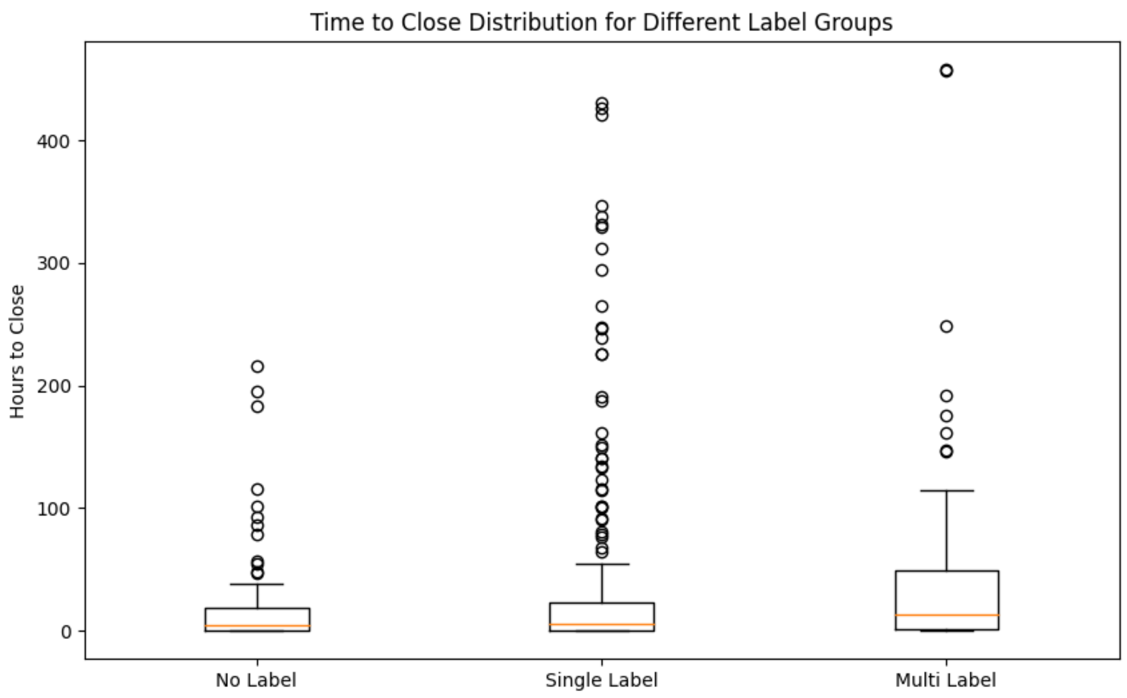

FIGURE 5 : Time-to-close distribution for different label groups

- **No-label issues :** Clearly stand out as the fastest to close, affirming a swift resolution process for issues without specific labels.
- **Single-label issues :** Exhibit a slightly longer time to close compared to no-label issues, indicating a moderate issue management performance.
- **Multi-label Issues :** Show a lower issue management performance, as they tend to take more time to close than both no-label and single-label issues.

Q5 : How does the distribution of issue closure times vary among those with no labels, a single label, or multiple labels ?

A5 : The analysis affirms that issues with no labels tend to close more quickly, followed by single-label issues with a moderate closure time. In contrast, multi-label issues show a lower issue management performance, taking more time to close. These findings align with the observed closing rates for different label categories, reinforcing the connection between labeling, closure efficiency, and overall issue resolution dynamics within the Rails project.

7.6 Studying the behaviour of issues over time

To comprehensively understand the dynamics of issue creation within the Rails project, we conduct an in-depth analysis of the behavior of issues over time.

7.6.1 Daily analysis

The analysis of daily issue creation patterns over a 40-day period provides valuable insights into the ebb and flow of activity within the Rails project. From the data presented on table2, on average, approximately 12 new issues are created daily, with moderate variability observed. Some days exhibit higher or lower activity levels, and instances of days with no new issues suggest potential periods of reduced activity, such as weekends. Quartile analysis provides distribution details, with most days experiencing 11 or fewer new issues, and 75% of days having 16 or fewer issues.

Count	41
Mean	12.20
Standard deviation	6.30
Min value	0
25% (Q1)	8
50% (Q2)	11
75% (Q3)	16
Max value	27

TABLE 2 : Descriptive statistics of daily issue creation

Illustrated in figure6, a discernible pattern is the increase and subsequent decrease in activity on weekends (red points), reflecting a standard workweek schedule. An exception is a spike in issue creation on December 31st, a weekend (Sunday), suggesting heightened activity due to year-end practices. This anomaly can be attributed to developers addressing pending issues, conducting year-end reviews, and performing bug fixes to ensure project stability.

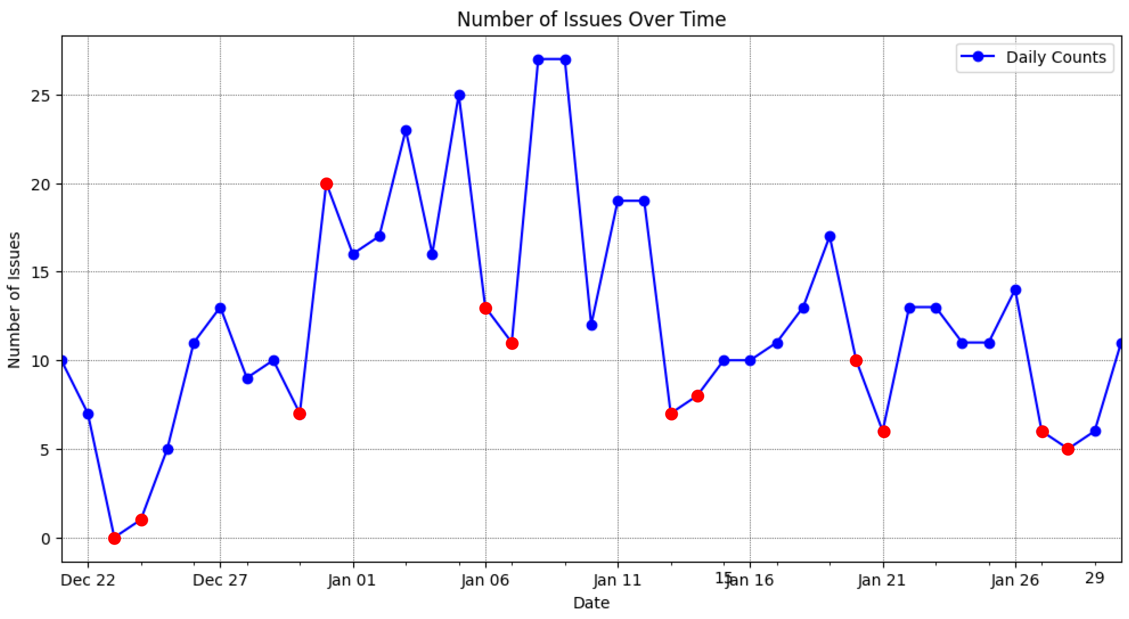

FIGURE 6 : Daily created issues over time

In the graph presented in figure7, a notable pattern emerges, depicting a consistent upward trend from late December to early January, followed by a decline. This trend aligns with heightened activity or increased contributions during the year-end and new year period. Factors contributing to this surge may include more available time during holidays, and new goals or initiatives at the start of the year.

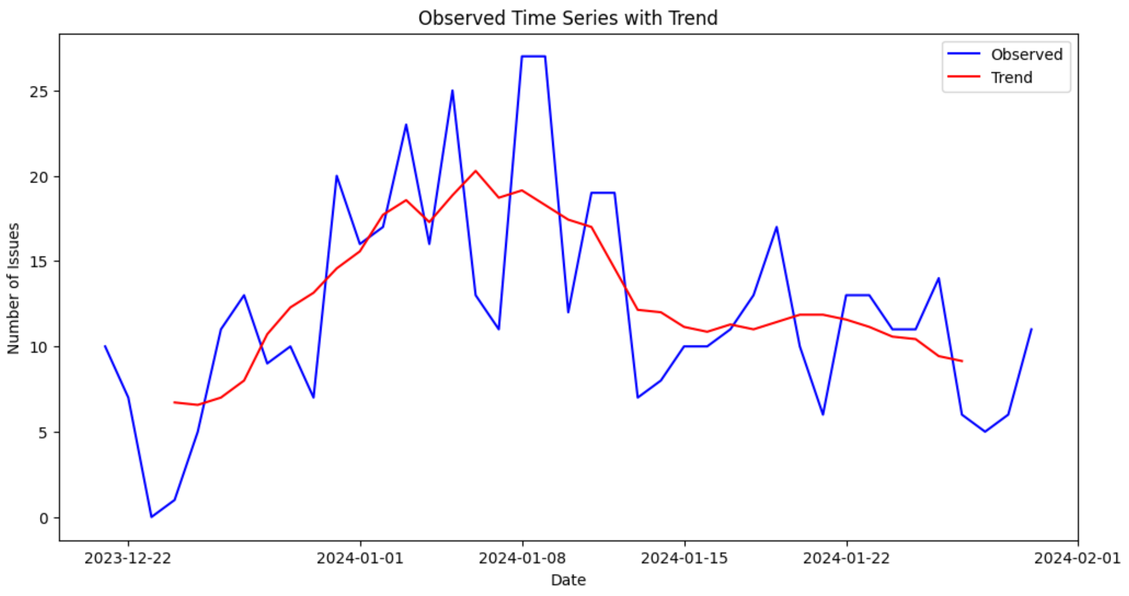

FIGURE 7 : Daily created issues with the trend over time

7.6.2 Hourly analysis

The distribution of issues created during different 4-hour periods throughout the day reveals intriguing patterns.

As presented in figure8, the period between 16 :00 and 20 :00 (4 :00 PM to 8 :00 PM) stands out with the highest percentage, constituting 22% of total issues. This indicates increased project engagement and contributions during late afternoon and early evening hours. The second most active period is from 12 :00 to 16 :00 (noon to 4 :00 PM), representing 21.6% of issues. Additionally, the periods from 20 :00 to 00 :00 (8 :00 PM to midnight) and 08 :00 to 12 :00 (8 :00 AM to noon) both contribute significantly, with 20.6% and 12.8%, respectively. The hours between 00 :00 and 08 :00 show comparatively lower activity. These findings suggest that contributors are more active during the later parts of the day and early evening.

Approximately 65% of issues are created during the second half of the day (from noon to midnight). This dominance in the later hours may be attributed to contributors finding it more convenient to engage during leisure hours after regular working hours, allowing for dedicated time and focus.

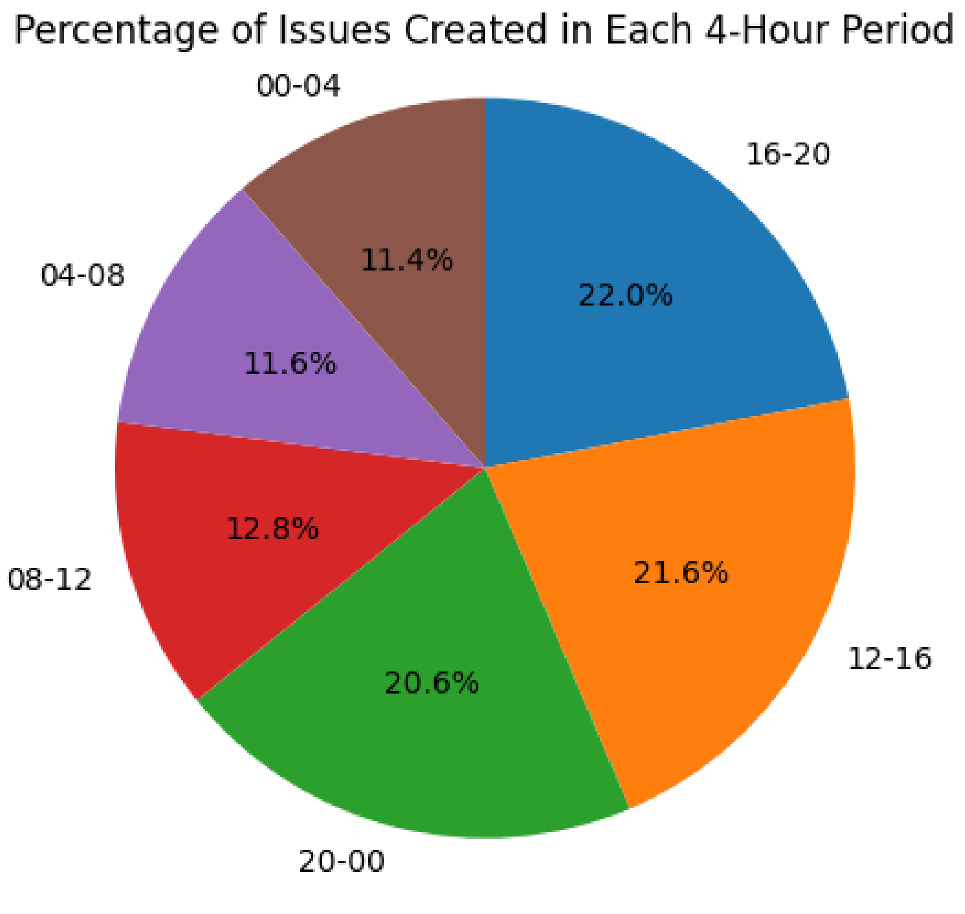

FIGURE 8 : Percentage of issues created in each 4-hour period

Q6 : What patterns emerge in the behavior of issues over time, specifically examining their evolution across days of the week and throughout different hours of the day ?

A6 : The analysis of daily patterns underscores notable variations influenced by workweek schedules, with reduced activity on weekends. Exceptional spikes, particularly observed during significant periods like the year-end, indicate heightened project engagement and efforts to address pending issues. Additionally, the hourly analysis reveals contributors' preferences for increased activity during the late afternoon and early evening, suggesting a concentrated effort during these hours, potentially after regular working hours.

8 User engagement

The chart in the figure9 illustrates users who have reported more than 10 issues, totaling 11 users falling within the range of 10 to 31 reported issues. Notably, *'dhh'* and *'skipkayhil'* stand out with 31 reported issues each, while an additional 3 users fall within the range of 21 to 29 issues (depicted by the blue bars). The remaining users reported between 10 and 13 issues each. Remarkably, these 11 users collectively contributed to 196 reported issues, constituting approximately 40% of the total. The remaining 304 issues were distributed among 185 users, resulting in an average of 1.6 issues per user. This concentration of reported issues among a smaller group indicate a core group of actively involved and experienced users within the community.

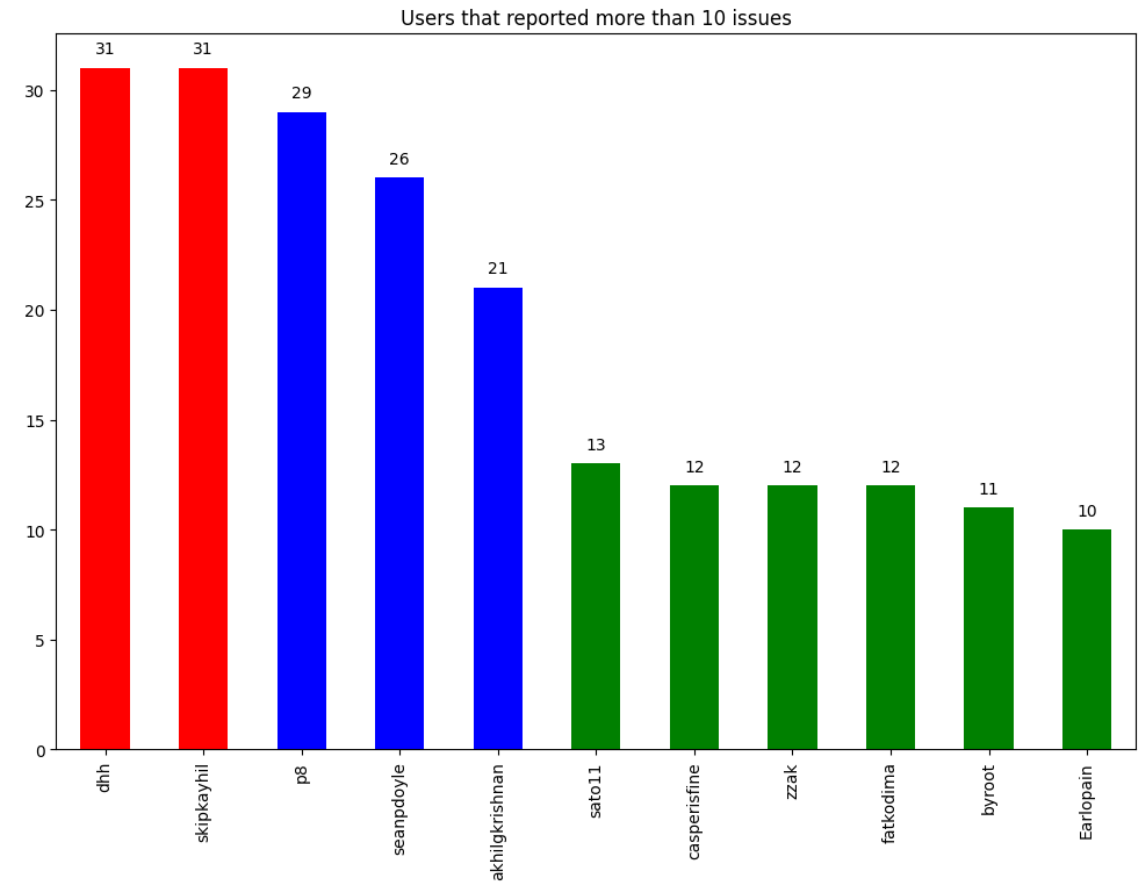

FIGURE 9 : Users that reported more than 10 issues

Q7 : Are certain users contributing more significantly to issue creation than others ?

A7 : Yes, the analysis indicates a concentration of contributions among a smaller group of 11 users, suggesting their significant role in issue reporting and community engagement. *'dhh'* and *'skipkayhil'* emerge as notable contributors, highlighting the importance of experienced and actively involved users in shaping the Rails project's development and issue resolution.

9 Classification model evaluation

In this final section, we present the steps of development of Rails issues classifier by fine-tuning and evaluating a BERT model specifically tailored for classifying Rails project issues based on their labels derived from descriptions. The dataset used for this task is limited to 500 rows, and certain filters are applied, such as removing issues without descriptions and those with no or multiple labels. The objective here is to predict the label based on the issue description.

9.1 Modeling methodology

1. **Data Splitting :** The dataset is split into training and testing sets. The split is 80% training and 20% testing.
2. **BERT Model Setup :**
   - BERT tokenizer (BertTokenizer) and a pre-trained BERT model for sequence classification (BertForSequenceClassification) are loaded from the Hugging Face Transformers library.
   - The model is initialized with weights pre-trained on a large corpus.
3. **Label Mapping :** Unique labels are identified, and a mapping from labels to numerical values is created.
4. **Training :** The model is trained for 10 epochs using stochastic gradient descent. Cross-entropy loss is calculated for backpropagation, updating the model's parameters.

9.2 Evaluation

For information, following the filtering process, the dataset now comprises 312 issues, encompassing 15 distinct labels (indexed from 0 to 14). Within this subset, 249 issues were allocated for training purposes, while the remaining 63 were reserved for testing.

The classifier is evaluated on the testing dataset. Predictions are made, and accuracy as well as a detailed classification metrics are computed :

- **Precision :**
  - Precision is the ratio of correctly predicted positive instances to the total predicted positives. It measures the accuracy of the positive predictions made by the model.

$$\text{Precision(label i)} = \frac{\text{True Positives of label i}}{\text{True Positives of label i + False Positives of label i}}$$

  - A high precision indicates that when the model predicts a positive class, it is likely to be correct.

- **Recall :**
  - Recall, or true positive rate, is the ratio of correctly predicted positive instances to the total actual positives. It measures the model's ability to capture all relevant instances.

$$\text{Recall(label i)} = \frac{\text{True Positives of label i}}{\text{True Positives of label(i) + False Negatives of label i}}$$

  - High recall indicates that the model is effective in capturing most of the positive instances.

- **Accuracy :**
  - Accuracy is the ratio of correctly predicted instances to the total instances. It measures the overall correctness of the model's predictions.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

  - High accuracy indicates the overall effectiveness of the model across all classes.

9.3 Results analysis

The evaluation results of the classifier on the test set are presented in Table 3 :

label(mapped)	**Precision**	**Recall**	**Support**
0	0.00	0.00	1
1	0.12	0.20	2
2	0.00	0.00	1
3	0.00	0.00	2
4	0.00	0.00	1
7	0.00	0.00	1
8	0.54	0.64	18
9	0.00	0.00	4
10	0.56	0.61	15
11	0.67	0.52	14
14	1.00	0.40	4
Accuracy		0.51	63

TABLE 3 : Evaluation results of the classifier on test set

- The overall accuracy of the model on the testing set is 50.79%. It demonstrates moderate overall accuracy.
- The model struggles with classes having limited support (low number of instances). Some classes exhibit good predictive performance (e.g., classes 1, 8, 10, and 11), while others lack accurate predictions.
- The observed accuracy of the model is commendable, considering the specific circumstances of having an extremely restricted training dataset comprising only 249 data points and 15 distinct classes. It is evident that employing a more extensive and diverse dataset for training purposes would contribute to a notable enhancement in the model's accuracy.

10 Conclusion

In conclusion, the analysis of GitHub issues within the Rails project provides valuable insights into the project's issue management dynamics. The study highlights the importance of effective labeling, issue closure, and user engagement in maintaining a robust development ecosystem. The study successfully addresses all the questions posed, shedding light on the predominant labeling category, label co-occurrences, issue closure time distributions, and user engagement patterns.

Additionally, the study presents the development and evaluation of a BERT model specifically tailored for classifying Rails project issues based on their labels derived from descriptions. The classifier achieved an accuracy of 51% on the test set, with precision and recall scores varying across different labels. While the results indicate room for improvement, the development of this classifier provides a foundation for further studies in automating issue categorization and management within the Rails project.