Analysis of Github Issues for Rails Repository

Mithila Sivakumar
mithila.sivakumar@gmail.com

ABSTRACT

In this assignment, I analyze GitHub issues from the Rails repository to uncover trends and patterns in the project's development. I extract and examine the issues to answer questions such as which labels are most popular, which months have the highest number of issues, and who the top three contributors are. In addition, I perform a classification task to predict labels for issues using a Hugging Face model. The loss and predictions are reported to evaluate the performance of the model. I use Tableau to visualize the statistics and trends, providing a clear view of the data. All code, data and related materials are available on GitHub [1].

KEYWORDS

github, rails, statistical analysis, machine learning, tableau, python

1 INTRODUCTION

In this assignment, I have analyzed the GitHub issues from the Rails repository [3] to gain valuable insights into the trends and user activity within the project. Each issue in the repository is assigned a label, which categorizes it into different areas, such as 'actionpack', 'rails-ujs', and 'regression', providing useful context about the nature of the issue. In addition, issues are associated with various metrics, such as the number of comments, which can be used to evaluate the level of engagement and importance of the issue.

The goal of this analysis is to answer key questions regarding the Rails issue tracker, including:

(1) How does the number of issues evolve over time?
(2) Are there any periods where there is a surge in reported issues?
(3) Who are the most active contributors to the issue tracker?
(4) What is the most popular category (label) assigned to issues?

I have also explored additional questions that arose during the investigation. By collecting data on the last 500 issues, I have used visualization tools such as Tableau [4] to present the results and uncover patterns and trends that can inform the Rails community and developers involved in the project. In the following sections, I will address each of the above questions and present visualizations to illustrate the findings.

2 METHODOLOGY

I used Python as the programming language. To extract issues from GitHub, I utilized the PyGithub module [6]. After obtaining my GitHub API token, I used it to access the Rails repository. Using the get_issues method, I retrieved the last 500 issues as specified in the assignment. From each issue, I extracted data including **"Issue Number," "Title," "Posted By," "Created At," "Updated At," "Labels," "Label Count," and "Comments."** Then I used the pandas library to organize these data into a DataFrame, which made the data set easier to manipulate, preprocess, and analyze. Converting the data to a DataFrame also made it simpler to save the information as a CSV file, which is essential for later use in visualization tools like Tableau. Tableau, a widely used data analytics tool, allowed me to create multiple bar charts and dashboards to analyze trends in issues effectively.

[1]https://github.com/msivakum16/ETSassignment2

3 DATA ANALYSIS AND VISUALIZATIONS

In this section, we take a comprehensive look at the trends in the issues reported in the Rails repository.

3.1 Evolution of Issues over Time

Figure 1 illustrates the evolution of trends in the issue over time. The figure is divided into four sections, with the bar chart in the upper left corner showcasing the progression of issues reported annually from 2014 to 2024. During the initial seven years (2014–2020), the number of issues remained relatively low, averaging approximately 17 per year and never exceeding 27. However, a notable spike occurred in 2021, with the number of reported issues increasing significantly from 15 to 50.

Since 2021, the number of issues has shown a steady upward trend, culminating in a total of 159 issues reported in 2024. Several factors could explain this increase. The 2020 pandemic, which saw a shift to remote work for many, may have indirectly influenced the surge in issues the following year. Additionally, updates or changes to the Rails framework during this period might have contributed to the increase. Overall, the sharp rise in issues appears to coincide with the post-pandemic period, suggesting potential links to the unique circumstances of that time.

The remaining three graphs in Figure 1 complement the bar graph that illustrates the trend trends of the issue over the years. The second chart, which focuses on issue trends by month, highlights the months with the highest number of reported issues. Notably, the last three months of the year—October, November, and December—record the most issues, with November standing out at an impressive 72 issues over the 10-year period. This surge could be attributed to the increased use of e-commerce websites during Thanksgiving and Christmas, leading to heightened traffic and, consequently, a rise in reported issues. The third chart, depicting issue trends by quarter, reinforces the findings from the monthly analysis. It clearly shows that the fourth quarter consistently experiences the highest number of issues across the decade.

The final chart, located in the bottom right corner, explores issue trends by week. Interestingly, Week 1 ranks fourth in terms of issue volume. However, when examining the monthly trends, February, March, and August occupy the fourth position. This discrepancy underscores the heightened activity in e-commerce websites during the holiday months, as January's position in the weekly trends could be linked to its proximity to December, when holiday traffic peaks. Overall, Figure 1 offers a comprehensive overview of issue trends,

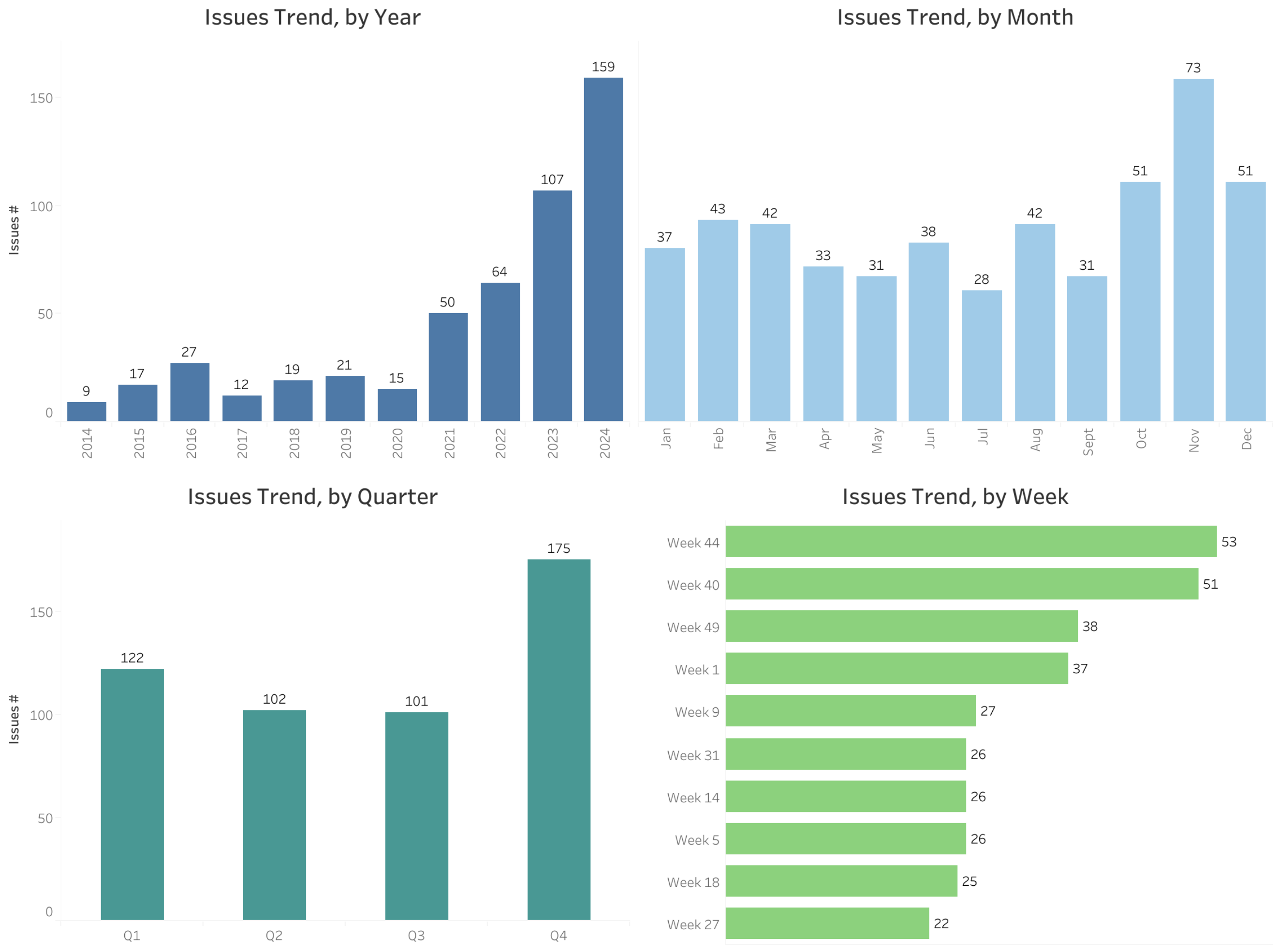

Figure 1: Tableau dashboard showing four different bar charts that depicts the trend of issues by Year, Month, Quarter and Week (left to right) respectively.

providing insights into their evolution over time and how they are influenced by seasonal and usage patterns.

3.2 Top Contributors to the Issues

Figure 2 presents four bar charts that provide insights into the top contributors to the issue tracker, their activity distribution, and the most popular labels assigned to issues. The bar chart in the top left corner highlights the leading contributors to the issue tracker. Among them, the user **dhh** stands out with the highest number of reported issues at 11, followed by **rafaelfranca** with 6 issues and **georgeclaghorn** with 4 issues. For clarity and conciseness, this chart displays only users who have reported more than three issues.

Adjacent to this chart, the second bar chart illustrates the distribution of issues reported by the top three contributors over the 10-year analysis period. It reveals that **dhh** has been consistently active since 2015, with a notable increase in issue reporting during 2023 and 2024. This sustained activity indicates a long-term engagement with the Rails project. In contrast, the other two contributors show sporadic activity. For instance, **rafaelfranca** reported two issues in 2014 and reappeared in 2024 with four issues after 10 years. This gap could be due to various factors, such as a shift away from Rails to other fields or projects before returning to Rails in 2024.

3.3 Popular Label Categories

The third bar chart in Figure 2 illustrates the most popular labels assigned to issues over the 10-year period. As evident from the chart, the top three labels are **activerecord, attached PR, and With reproduction steps**. The chart not only displays the total number of assignments for each label but also their respective percentages. For instance, **activerecord** accounts for 46.2% of all label assignments with 231 assignments over the decade. It is followed by **attached PR** with 181 assignments (36.2%) and **With reproduction steps**

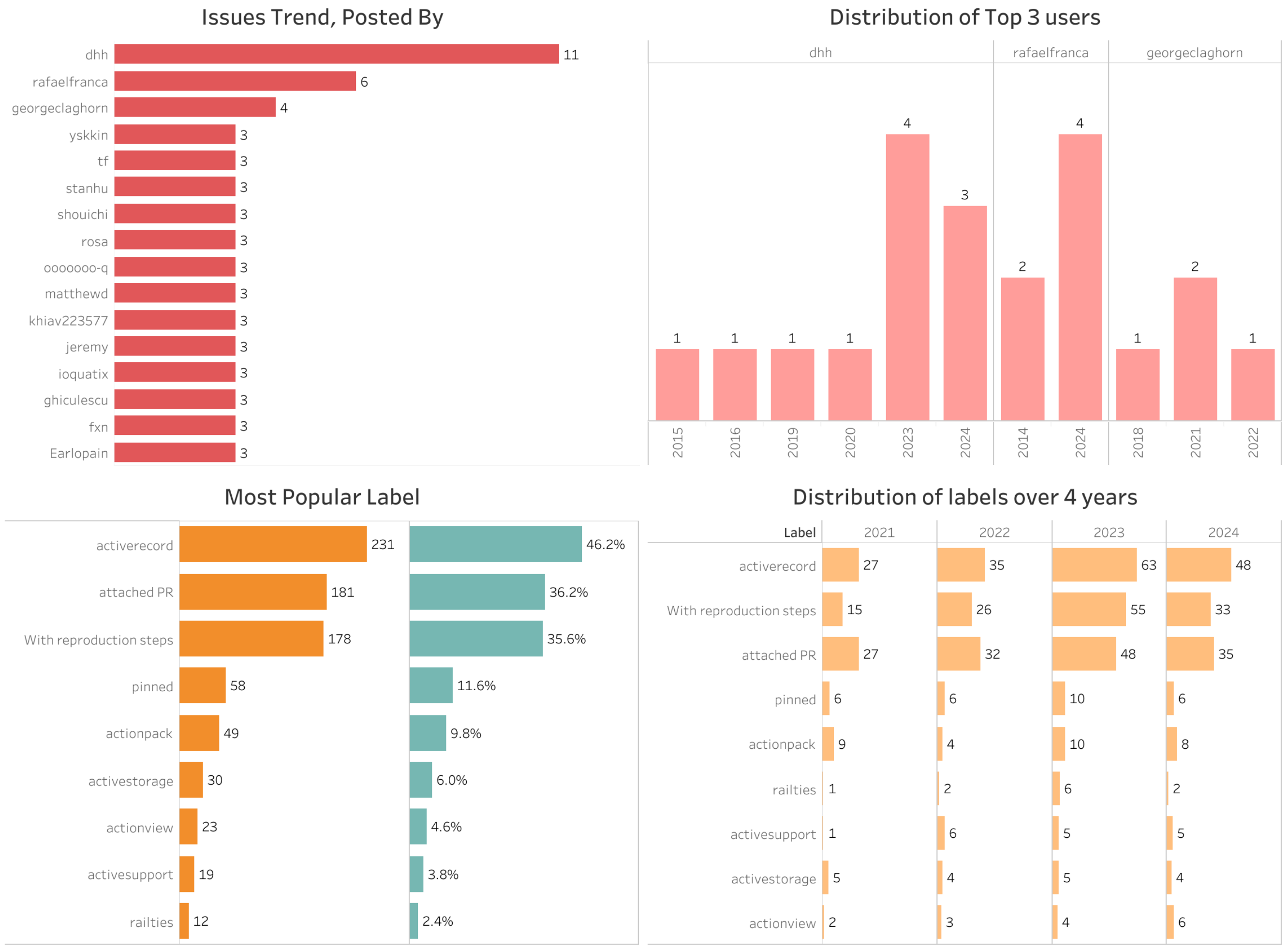

Figure 2: Tableau dashboard showing four different bar charts that showcases the top contributors of issues and the most popular labels.

with 178 assignments (35.6%). To maintain clarity, this chart has been filtered to include only labels with more than 10 assignments.

The final bar chart, located at the bottom right, focuses on the distribution of these labels during the last four years. This specific timeframe was chosen because, as discussed in the previous section, the number of reported issues significantly increased in these years. Interestingly, the chart reveals that 2023 recorded the highest number of label assignments among the top three labels, surpassing 2024, even though 2024 had the highest total number of issues. This discrepancy might suggest that new labels were introduced in 2024 or that the assignment of existing labels was deprioritized in favor of other classifications. These charts collectively shed light on the contributors' involvement and their evolving participation in the Rails project over the years.

3.4 Comments Analysis

Figure 3 provides an analysis of comment trends on issues over the 10-year period, offering insights into engagement and interaction levels. On the left side, a pie chart illustrates the distribution of issues based on the number of comments they received. Out of a total of **3152 comments**, the average number of comments per issue is approximately 6. According to the chart, **9.20%** of issues had no comments, **58.80%** had fewer than 6 comments but at least one, and **32%** of issues had more than 6 comments. This distribution suggests that most issues receive some level of attention, with only a small fraction remains uncommented. High interaction levels on GitHub issues are generally a positive indicator, as they often lead to more thorough problem-solving. Additionally, issues with more comments may be more likely to get resolved due to the increased collaboration and discussion they generate.

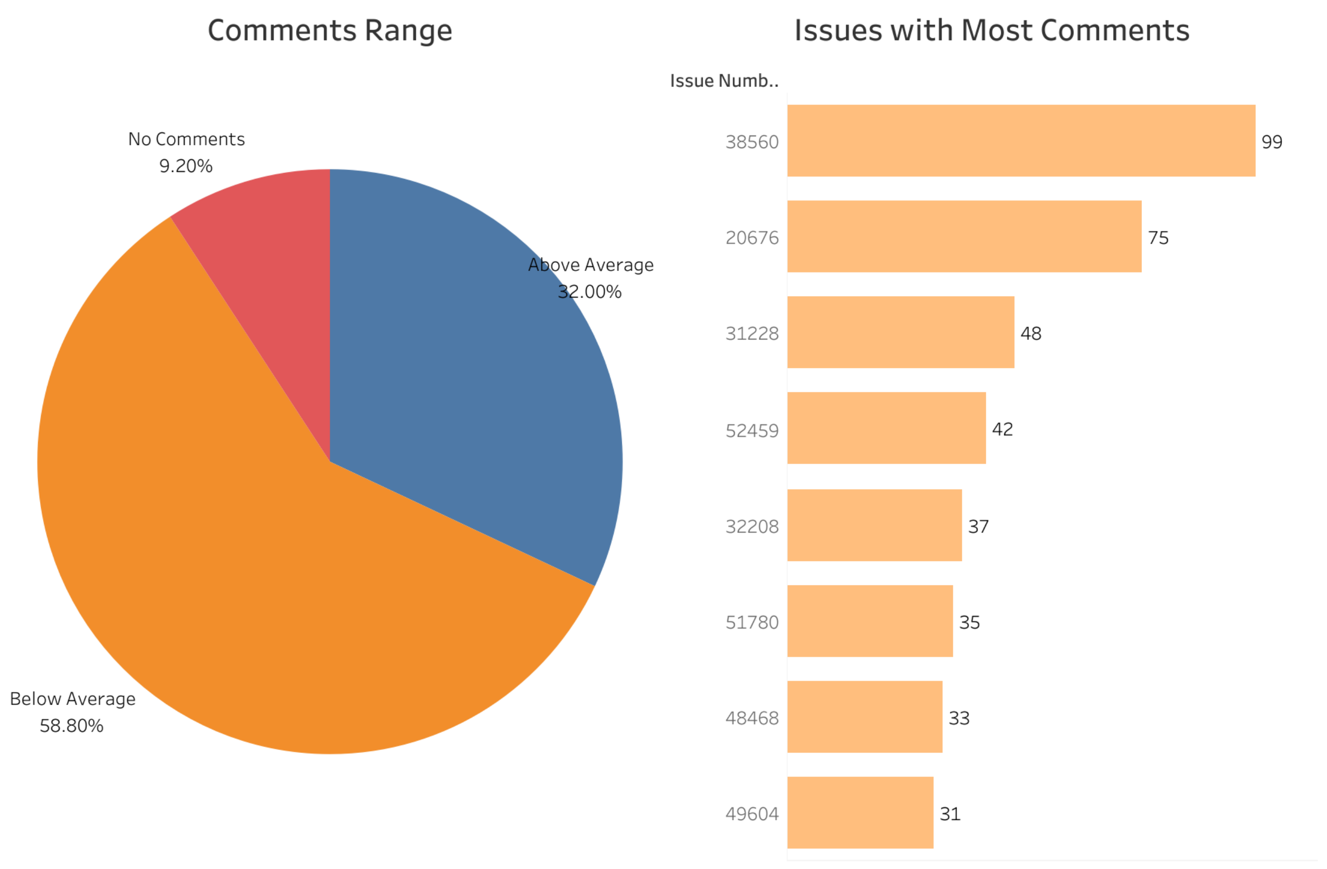

Figure 3: Tableau dashboard showing distribution of comments on the left and issue with most comments (more than 30) on the right.

On the right side of Figure 3, a bar chart highlights the issue with the highest number of comments. Issue **#38560**, titled **NSCFConstantString initialize may have been in progress in another thread when fork() was called** stands out with **99 comments**. This high level of engagement likely reflects the issue's significance or the widespread impact it had on users, making it a focal point for discussion and resolution efforts. Overall, Figure 3 offers a comprehensive view of issue engagement, emphasizing the importance of collaborative interaction in addressing and resolving reported problems effectively.

4 PREDICTIVE ANALYSIS

In this section, the focus is on automating the classification of issues for the Rails project using the DeBERTa (Decoding-enhanced BERT with disentangled attention) model from Hugging Face [2]. DeBERTa is a state-of-the-art natural language processing model known for its superior performance in understanding textual data. By utilizing the descriptions of issues as input, DeBERTa predicts the most appropriate labels, streamlining the classification process.

4.1 Model selection

DeBERTa (Decoding-enhanced BERT with Disentangled Attention) is a transformer-based language model introduced by Pengcheng He et al. [5], building upon Google's BERT and Facebook's RoBERTa. It introduces disentangled attention mechanisms and enhanced mask decoding, improving contextual representation and efficiency. Notably, it achieves superior performance using only half the training data required by RoBERTa. Hence, I have selected this model for the multi-label classification problem at hand.

4.2 Methodology

The dataset consists of 500 issues, of which 70 have no labels assigned. To address this, I first separated these 70 unlabeled issues into a distinct dataframe, as the primary task is to predict labels for these issues. The remaining 430 labeled issues were split into training and test datasets using an 65:35 ratio. This split ensures a sufficient amount of data for training while reserving some for evaluating the model's performance. I utlized the AdamW optimizer [1], with a learning rate of 6×10^{-5}. Multi-label classification presents unique challenges, as a single issue can be associated with multiple labels. Moreover, the dataset is small and highly skewed.

Some labels are prevalent across many issues, while others are rare, assigned to only one or two issues out of the entire dataset. This imbalance adds complexity to model training, as the model needs to learn to generalize across both frequent and infrequent labels.

4.3 Results

I trained the model for 15 epochs, as I observed that after this point, the training loss stabilized and showed minimal fluctuation. To avoid potential overfitting, I decided to limit the training to 15 epochs. Figures 4, 5, 6 and , 7 represent the loss, precision, recall and F-1 scores of the training and validation respectively. The training loss reached a stable and favorable value of around 0.10, which indicates good learning during the training phase. Additionally, **the precision, recall and F-1 metrics were promising, both hovering around 90% , 60% and, 71%** respectively, suggesting that the model is able to correctly identify the majority of positive instances and is relatively balanced in terms of false positives and false negatives.

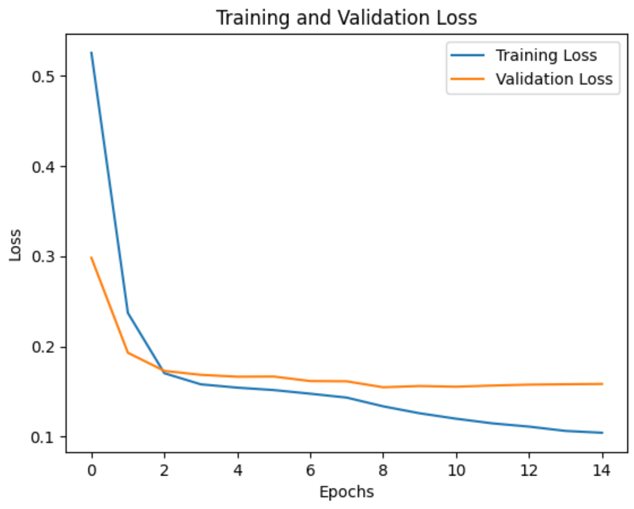

Figure 4: Line chart depicting the training and validation loss

Despite achieving strong results during training, the validation metrics indicate areas for improvement in the model's generalization to unseen data. Specifically, the validation loss is **0.1584**, with a precision of **60%**, recall of **39%**, and an F1-score of **47%**. While the precision suggests that the model is relatively effective at identifying positive instances without excessive false positives, the lower recall highlights its struggle to capture a significant portion of the actual positive instances. Consequently, the F1-score, which balances precision and recall, reflects a moderate overall performance.

This discrepancy between the training and validation performance underscores the need for enhancements to the model training process. One major contributing factor could be the limited size of the training dataset, which may not provide sufficient diversity to allow the model to generalize effectively. Furthermore, the complexity of the DeBERTa model, paired with relatively fewer training samples, likely leads to overfitting of the training data.

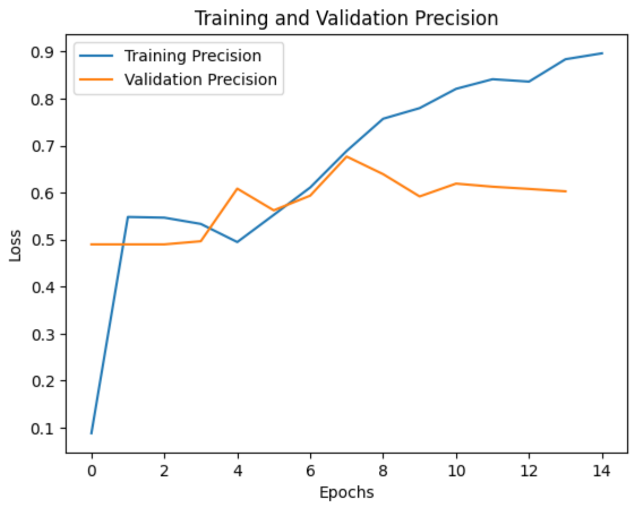

Figure 5: Line chart depicting the training and validation precision

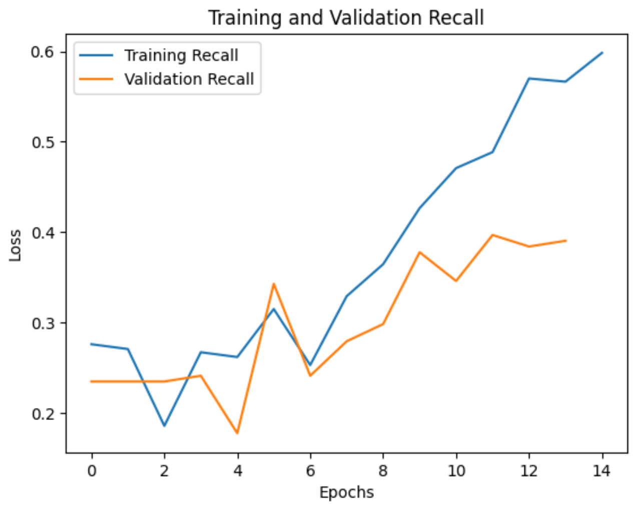

Figure 6: Line chart depicting the training and validation recall

After training, I used the model to generate predictions for a set of 70 unlabeled issues by applying a probability threshold of **50%**. While these predictions provide some insight into the model's performance, the results indicate that further improvements are necessary. Specifically, the model's performance could benefit from the addition of more labeled data for training. This would allow for further fine-tuning and help the model better capture complex patterns, thereby improving its overall accuracy.

5 CONCLUSION

In this assignment, I conducted a statistical analysis and created visualizations based on the last 500 open issues from the Rails GitHub repository. The analysis focused on trends such as the time periods

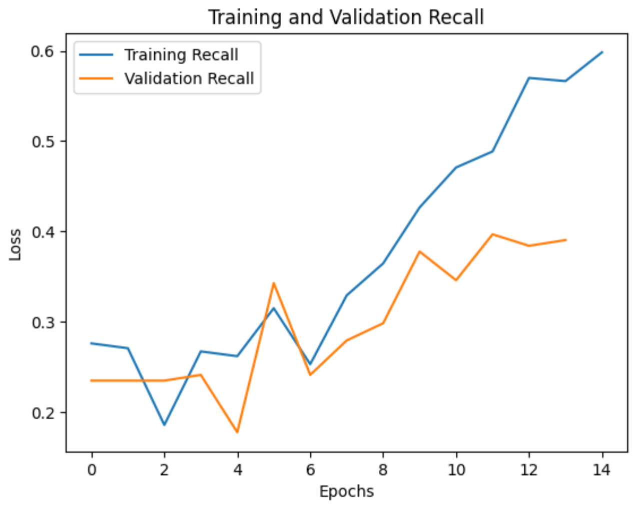

Figure 7: Line chart depicting the training and validation F-1 scores

when most issues were posted, the users who contributed the most issues, and which issues received the most comments, with corresponding visualizations to highlight these patterns. Subsequently, I tackled a classification task to classify unlabeled issues using the DeBERTa model from Hugging Face. The model achieved an impressive **90% precision and 60% recall** during training, with a **60% precision** observed during validation.

REFERENCES

[1] Adamw optimizer. https://pytorch.org/docs/stable/_modules/torch/optim/adamw.html#AdamW.

[2] Deberta. https://huggingface.co/docs/transformers/en/model_doc/deberta.

[3] Rails. https://github.com/rails/rails/issues.

[4] Steven Batt, Tara Grealis, Oskar Harmon, and Paul Tomolonis. Learning tableau: A data visualization tool. *The Journal of Economic Education*, 51(3-4):317–328, 2020.

[5] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

[6] Nafiseh Nikeghbal, Amir Hossein Kargaran, Abbas Heydarnoori, and Hinrich Schütze. Girt-data: Sampling github issue report templates. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*, pages 104–108. IEEE, 2023.