1. Description

This report describes the given project called *IssueAnalyzer,* the objective of the project is to perform an analysis of the issues in this repository. The code is implemented in a way that initially collects data from the provided link and then processes the collected data while answering the below questions

- How do the number of issues evolve?
- Are there any periods in which we get more issues?
- Is there anyone who reports more issues than others?
- What is the most popular category (label)?
- Classifying the issues is an essential task for any project, including Rails.

2. Methodology

This section of the report describes the methods that are performed to complete the given task. The provided code defines a Python class named IssueAnalyzer that leverages various libraries such as requests, pandas, matplotlib, seaborn, numpy, and machine learning tools from the transformers library for analyzing GitHub issues. Below is a breakdown of the methodology and functionality of each part of the IssueAnalyzer class:

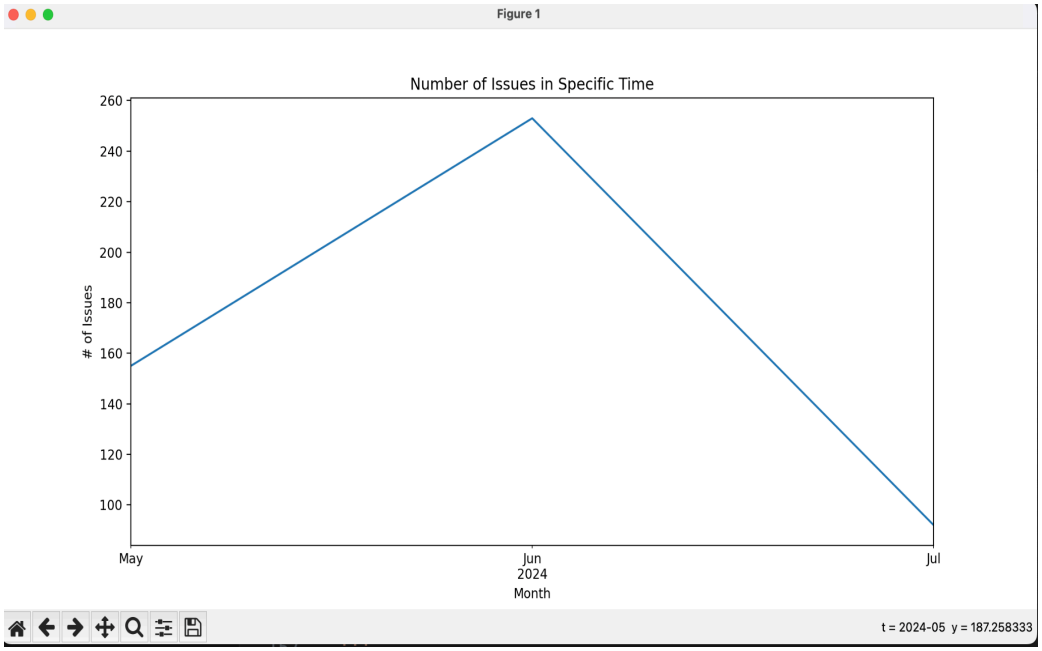
- **Initialization**

  __init__ Method: Initializes the IssueAnalyzer instance with a default URL to fetch issues from the GitHub API for a specific repository (Rails in this case).

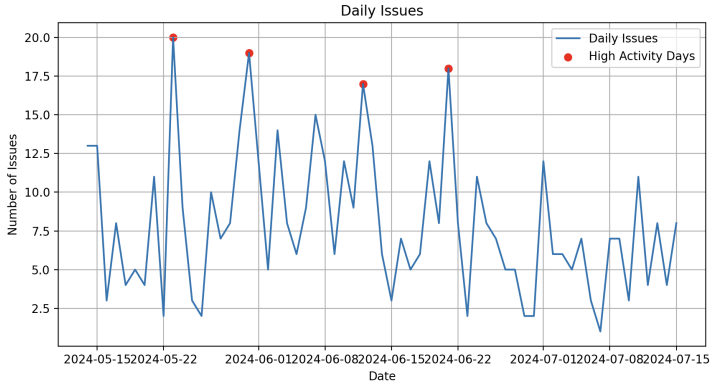
- **Data Collection**

  getIssues Method: Fetches issue data from the specified GitHub repository over multiple pages (up to 5 pages, with each page containing up to 100 issues). It handles API responses, converts the JSON data to a DataFrame, and saves or updates a local CSV file (data.csv) with the issue data. The total records collected are 500 including 27 different attributes (columns).

- **Data Analysis and Visualization**
  - **analyze Method:** Reads the issue data from the CSV, converts the created_at timestamps to pandas DateTime objects, and groups the data by month to analyze the number of issues over time. It then visualizes this data as a line plot. **June 2024 is the month found with the highest number of issues created.**



- **highestPeriods Method**: Similar to analyze but focuses on daily data. It identifies days with an exceptionally high number of issues (defined as more than two standard deviations above the mean) and visualizes these as highlighted points on a daily time series plot. **The busiest day in terms of most issues created was 22 July 2024.**

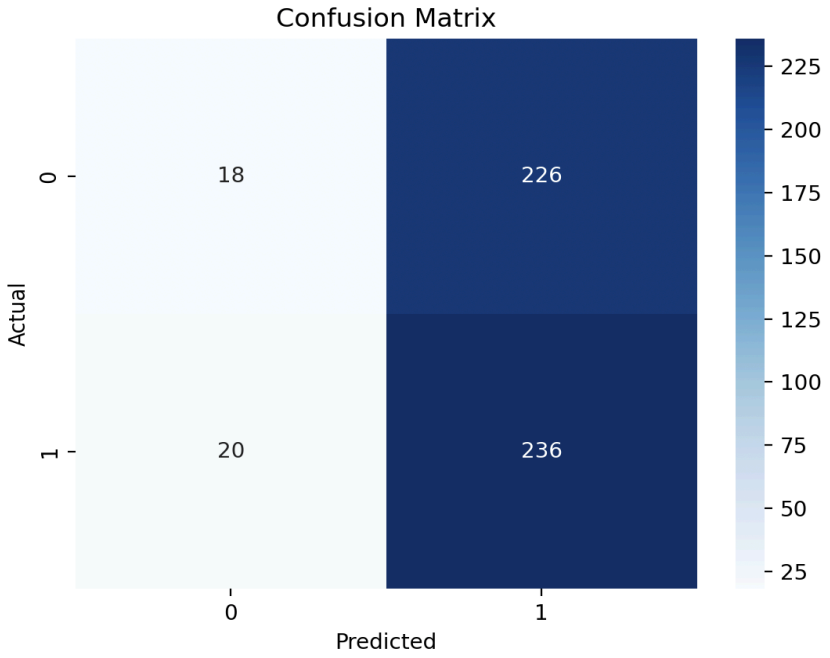


- **getActiveUser Method:** Identifies the most active user by counting occurrences of user entries in the data. As per analysis, the most active user was ‘zzak’ has this profile
- **getPopularCategory Method:** Analyzes the 'labels' associated with each issue to determine the most frequently mentioned categories, which helps in understanding the common topics or problems reported in the issues. The most active category noticed was **‘railties’** and the lowest **‘SQLite’**.

- **Issue Classification with Machine Learning**
  - **classifyIssue Method:** Uses a pre-trained BERT model from the Hugging Face transformers library to classify the textual content (body) of the issues. The classified results are saved back to the CSV. The method also handles model persistence by saving the trained model and tokenizer.
- **Evaluation of Classification**
  - **generateRandomTrueLabels Method:** Generates a random set of true labels for testing purposes (not typical for real-world model evaluation but useful for demonstrating the evaluation process in this context).
  - **evaluateResults Method:** Evaluates the classification results using metrics such as accuracy, precision, recall, and F1 score. It also displays a

confusion matrix to visualize the performance of the classification against the randomly generated true labels. Below table shows the calculated values

Metric	Measured Value
Accuracy	0.50
Recall	0.50
Precision	0.49
F-score	0.39



- **Execution**

The IssueAnalyzer class is instantiated, and its methods are called in sequence to perform the entire analysis workflow from data collection to evaluation. This sequential call simulates an end-to-end process of fetching data, analyzing it, performing machine learning classification, and evaluating the results. Moreover, I have created app.py file for flask endpoint.