AI-Powered Developer Performance Analytics Dashboard

Objective

To develop a Streamlit-based dashboard that provides insights into developer performance using data from an open-source GitHub repository. The system should focus on collecting and analyzing GitHub data, calculating performance metrics, and implementing a natural language interface for querying these metrics.

Methodology

- ✓ Data such as issues, pull requests, and commits were collected using the GitHub API.
- ✓ The data which was collected from the data collection module was loaded into Pandas DataFrames using helper functions. So based on this data, performance metrics were calculated for the repository.
- ✓ An interactive dashboard was created using the Plotly visualization library integrated within a Streamlit application.
- ✓ Cohere's advanced API processed user queries by providing relevant results based on the repository's data.

Project planning

Used the Trello project management tool to have control over my project progress.

Trello link: ***https://trello.com/b/UvfeIOST/pro-dev-performance-dashboard***

GitHub Repository

https://github.com/JoshikRaj/AI-Powered-Developer-Performance-Analytics-Dashboard.git

Key Findings

- ✓ Performed Data collection by getting the URLs **from multiple repositories**.
- ✓ The system effectively fetched all the data about the given repository URL.
- ✓ Calculated metrics ***(Commit Frequency, Total Commits, PR Merge Rate, Issue Resolution Time (days), PR Review Time (days), Code Churn Rate, Average Commit Size, Open-to-Closed Issues Ratio)*** were visualized clearly and represented in the dashboard.
- ✓ NLP – Module successfully responded to the requests/questions raised by the user - Trained the calculated metrics to be integrated with the LLMs to retrieve details from the requested repositories.
- ✓ Used Cohere's advanced API for **fast retrieval of complex queries**.
- ✓ Used pickle for data storage as it can be more advantageous than CSV file. This pickle file can be converted to a CSV file later using dataframe.
- ✓ Whenever the user needs to analyze the successive URL, the data from the previous URLs is overwritten instead of creating multiple files.
- ✓ **Error handling – Test cases are checked and handled**, unit testing and functionality testing with exception handling is done.

Recommendations (TO DO)

1. Need to expand the data size by accessing high-complexity repositories like ultralytics.
2. Need to use more secure methods for storing the API keys.
3. Need to process multiple URLs for the last three modules.

Conclusion:

This AI-powered Developer Performance Dashboard provides project managers and team leads with actionable insights to track and optimize developer productivity. By visualizing key metrics like PR merge rates, issue resolution times, and code churn, it empowers teams to make data-driven decisions. This tool seamlessly integrates with real-world scenarios, fostering continuous improvement and team efficiency.