Mining Graph Patterns in Software Development from Code Repositories

Alexander MacDonald (CS), Ethan Vaz Falcao (DS & Math), Jakob Simmons (CS), Nur Fateemah (CS)
Advisors: Prof. Fabricio Murai (CS & DS), Prof. Frank Zou (Math & DS)

Can **maintainability** and **repository status** of code repositories be predicted from *their developers' commit activity*?

Objectives

1. Collect data on repositories using pre-existing tools and APIs and establish useful features from the data for training machine learning models.
2. Establish if/what network graphs are good predictors of maintainability and activity status.
3. Use machine learning and Graph Neural Networks to study the relationship between the various features of a repository and evaluate performance.

Prediction Tasks

Task 1 (Regression): Predict **Code Complexity Metrics**:
- Trained on Mean Square Error
- Used Mean Absolute Percentage Error (MAPE) to determine effectiveness

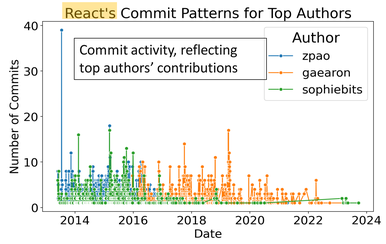
Task 2 (Classification): Predict **Repository Status** (Dead/ Alive)
- Trained and evaluated on Log Loss

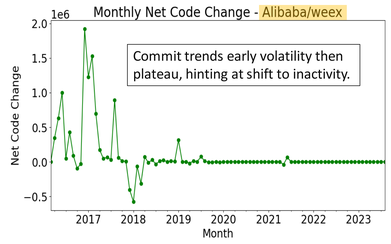
Methodology

1. Extracted comprehensive file-level metrics using Lizard, a repository feature analysis tool and pydriller.
2. Developed a data collection tool for metrics, commits, and issues from top GitHub repos.
3. Used Python library *NetworkX* to create 3 types of graphs: Repository Structure, Co-commited files, and User- file commute graphs
4. Generated Graph Encodings using a Graph Autoencoder implemented in DGL
5. Experimented with different models: Random Forest, k-NN, Lasso Regression, XGBoost, and Graph Neural Networks
6. Evaluation metrics: Mean squared error, R^2 score, and MAPE for regression, Log loss for classification.

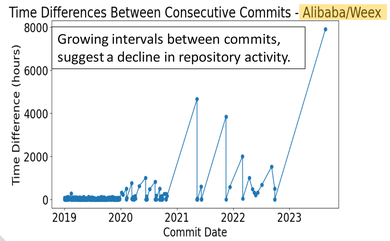
Exploratory Data Analysis

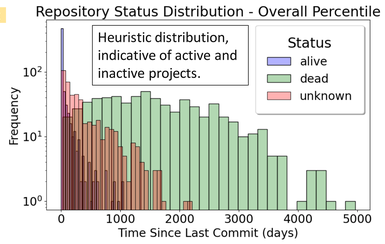
Analysis of **top authors' contributions** and **trends in commit volume & timing**, showing insights into collaborative dynamics and project activity.



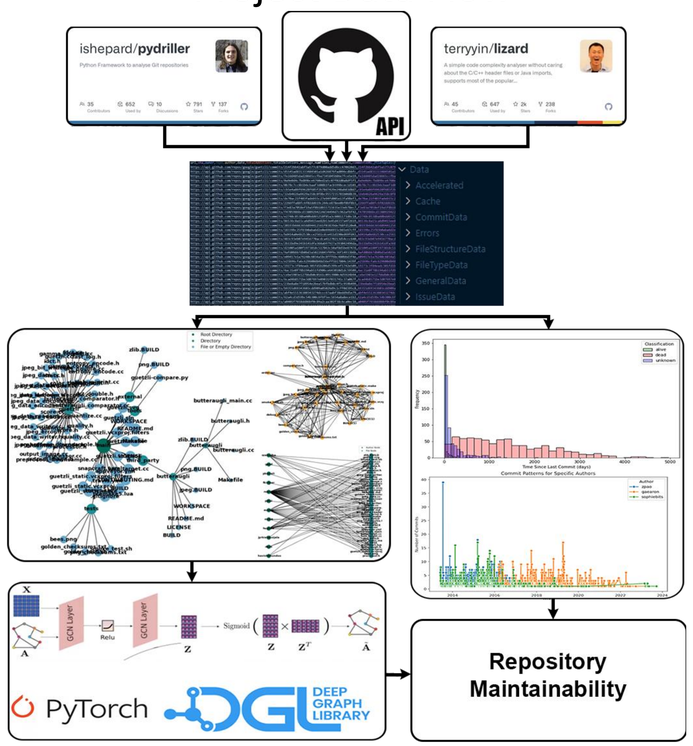


Study of project's coding momentum, and repository vitality (through a heuristic for development activity).

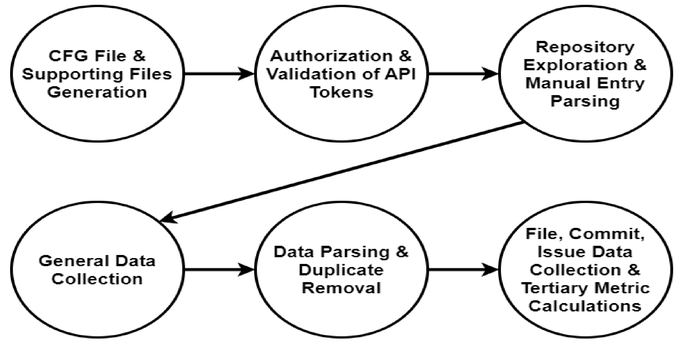




Project Data Flow



Data Collection Tool Results



We collected over 7 GB of text data from GitHub's API over a month of executing this tool.

Additional Features: API token limit handling, save states, ARGV commands, error logging & handling.

Results and Conclusions

Prediction Task 1 – Estimating Maintainability Index

Mean Absolute Percentage Error (lower = better)

Model	Repo Features	Graph Embeddings	Both
Random Forest	**10.56**	11.39	10.62
KNN	11.17	11.28	11.17
Lasso	11.40	11.40	11.40
XG Boost	11.06	**11.12**	**11.37**

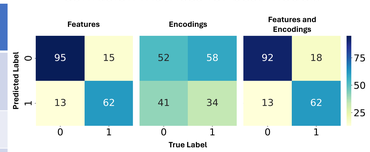
Predictions based on
- Repository features only: led to best results (with Random Forest);
- Graph encodings only: resulted in slightly higher MAPE; and
- Both types of features: resulted in slightly worse results.

Prediction Task 2 – Classifying Repos as Dead or Alive

Log Loss of trained models (lower = better)

Model	Repo Features	Graph Embeddings	Both
Random Forest	4.68	16.37	4.68
KNN	8.57	15.98	8.75
Lasso	16.36	16.36	16.36
XG Boost	**1.98**	**7.76**	**2.07**

Confusion Matrices for best models



Similar observations as before, but Graph Encodings hurt performance more.

Conclusion

Based on our results, one of two outcomes are possible
1. Relation between graph encodings and target variables is weak; OR
2. It is too complex to be captured with the models that we used

Future work could include: training more complex models for the prediction tasks, changing the architecture of the graph autoencoder, or using more data to train the models.