GITHUB PR Spam Detection

UML 501 Machine Learning

EST Evaluation Report

Submitted by:

(102217034) Aryan Panja

(102217062) Prabhjot Singh

B.E. 3rd Year- COPC

Submitted to:

Dr. Archana Singh

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Computer Science and Engineering Department

TIET, Patiala

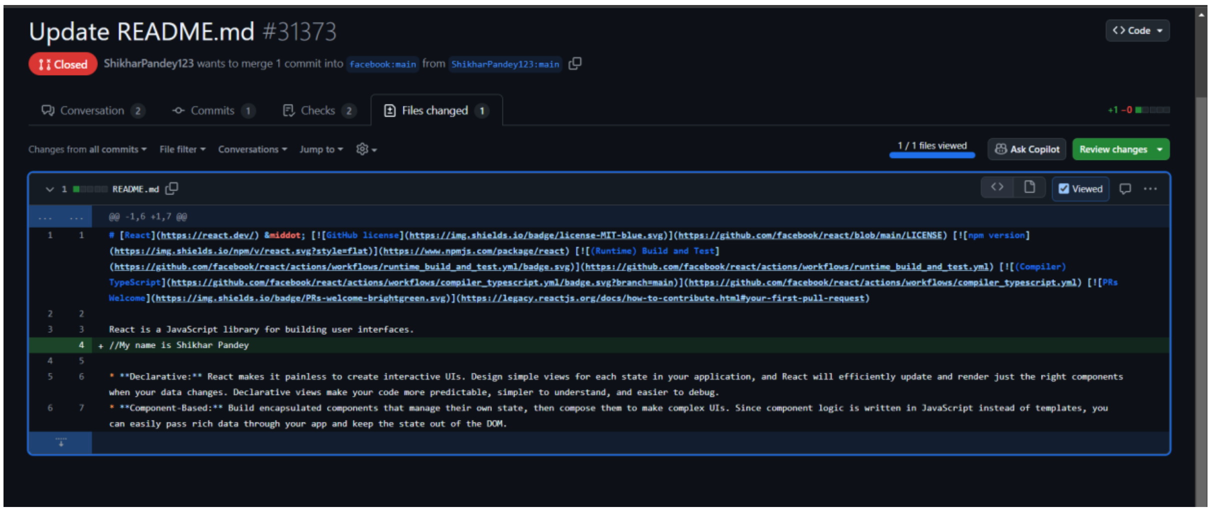
November 2024

INTRODUCTION

Introduction to the Problem

GitHub has become a central platform for collaborative software development, where open-source contributions from developers across the globe are highly encouraged. However, with the increasing popularity of open-source projects, there has also been a rise in low-effort contributions. Many developers, aiming to earn badges or recognition, submit pull requests (PRs) with minimal changes, which can range from insignificant edits like:

- Changing file formats (without any functional improvement)
- Adding their names to a project's author list
- Making trivial modifications such as shortening or removing data in documentation files (e.g., README)



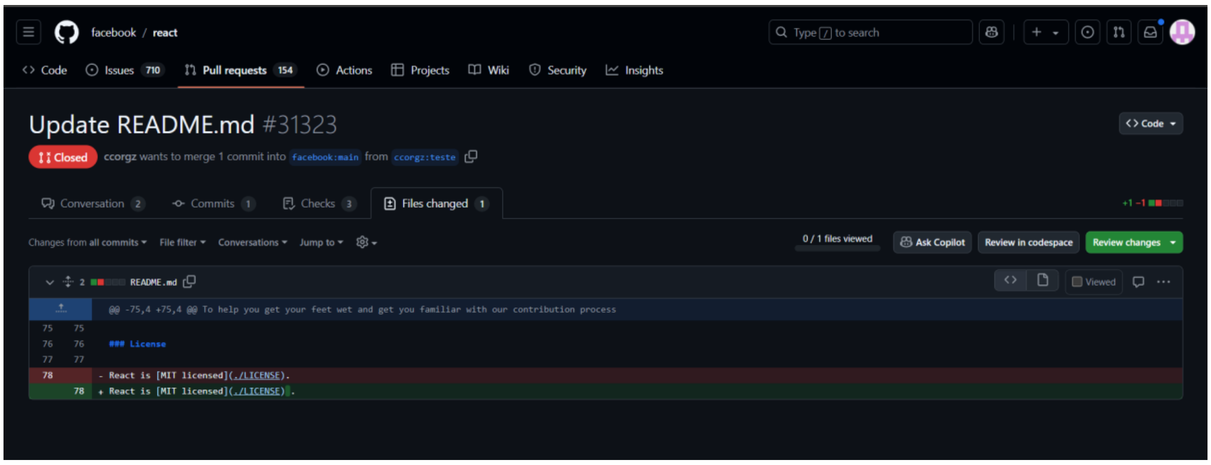
Update README.md #31373

Closed ShikharPandey123 wants to merge 1 commit into facebook:main from ShikharPandey123:main

```
@@ -1,6 +1,7 @@
  # [React](https://react.dev/) &middot; [![GitHub license](https://img.shields.io/badge/license-MIT-blue.svg)](https://github.com/facebook/react/blob/main/LICENSE) [![npm version](https://img.shields.io/npm/v/react.svg?style=flat)](https://www.npmjs.com/package/react) [![(Runtime) Build and Test](https://github.com/facebook/react/actions/workflows/runtime_build_and_test.yml/badge.svg)](https://github.com/facebook/react/actions/workflows/runtime_build_and_test.yml) [![(Compiler) TypeScript](https://github.com/facebook/react/actions/workflows/compiler_typescript.yml/badge.svg?branch=main)](https://github.com/facebook/react/actions/workflows/compiler_typescript.yml) [![PRs Welcome](https://img.shields.io/badge/PRs-welcome-brightgreen.svg)](https://legacy.reactjs.org/docs/how-to-contribute.html#your-first-pull-request)

  React is a JavaScript library for building user interfaces.
+ //My name is Shikhar Pandey

  * **Declarative:** React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes. Declarative views make your code more predictable, simpler to understand, and easier to debug.
  * **Component-Based:** Build encapsulated components that manage their own state, then compose them to make complex UIs. Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep the state out of the DOM.
```



Update README.md #31323

Closed ccorgz wants to merge 1 commit into facebook:main from ccorgz:teste

```
@@ -75,4 +75,4 @@ To help you get your feet wet and get you familiar with our contribution process

  ### License

- React is [MIT licensed](./LICENSE).
+ React is [MIT licensed](./LICENSE) .
```

These pull requests can clutter the review process and waste the time of maintainers, as they don't contribute meaningfully to the project's progress.

Solution Overview

Our project aims to solve this issue by providing an automated solution that helps open-source creators quickly assess whether a pull request is worth reviewing or should be classified as spam. This tool categorizes PRs into two categories:

1. **SPAM**: Pull requests that offer little to no value, such as trivial changes or modifications made purely for personal gain (e.g., badge collection).
2. **NOT_SPAM**: Pull requests that introduce meaningful code improvements, bug fixes, feature additions, or genuine contributions that align with the project's goals.

Data Collection and Model Training

To train the model, we focused on some of the most active and influential repositories in the open-source ecosystem. These repositories were chosen because they have high traffic and contributions, making them ideal sources for identifying patterns in pull requests. We have curated our own dataset from the following well-known open-source projects:

1. **React**: A widely used JavaScript library for building user interfaces.
2. **TensorFlow**: An open-source machine learning framework developed by Google.
3. **Django**: A high-level Python web framework for rapid development of secure and maintainable websites.

By analyzing the pull requests from these repositories, we created a dataset that helps the model distinguish between valuable contributions and low-effort submissions.

Problem-Solving Product for Open-Source Communities

Our tool is not just a project but a **real-world problem-solving product** designed to make a meaningful impact in the open-source community. It simplifies the work of maintainers by automating the classification of pull requests, thereby saving them time and effort that can be redirected towards actual development work.

This product addresses the following key challenges in the open-source world:

- **Reducing Noise**: It helps open-source maintainers avoid wasting time reviewing irrelevant or trivial contributions.
- **Enhancing Contribution Quality**: It encourages developers to submit more meaningful, well-thought-out contributions, ultimately improving the quality of open-source projects.
- **Efficient Pull Request Management**: By automating the spam detection process, our tool helps project maintainers efficiently manage large volumes of pull requests, ensuring that their attention is focused on the contributions that matter most.

Impact and Market Potential

The tool has great potential in the market because it solves a practical problem faced by millions of open-source contributors and maintainers. By automating the PR review process, it ensures that developers can spend more time working on valuable features or fixes, while maintainers can focus on high-quality contributions.

With the growing interest in contributing to open-source projects, the demand for efficient pull request management tools is expected to rise. Our product not only enhances the experience of contributing to open-source but also helps maintainers optimize their workflow and make better decisions.

LITERATURE

The problem of identifying and filtering low-effort contributions in open-source platforms like GitHub is closely linked to challenges in collaborative software development and spam detection. Prior research has explored related areas, including natural language processing, classification models, and the social dynamics of contribution-based platforms. This section summarizes existing work in these domains to establish a foundation for our approach.

GitHub and Collaborative Development

GitHub's role as a central hub for collaborative software development has been extensively studied. Researchers such as Dabbish et al. (2012) highlight its unique features, including pull requests and issue tracking, which foster transparency and participation in open-source projects. However, studies also note the increasing noise generated by low-value contributions, a concern exacerbated by the popularity of hackathons and events like Hacktoberfest (Zhang et al., 2020).

Automated Spam Detection

Spam detection is a well-researched field in domains such as email filtering and social media moderation. Techniques commonly used include:

1. **Text Analysis and Feature Engineering:** Identifying spam based on textual patterns, term frequencies, and semantic similarities (Metsis et al., 2006).
2. **Supervised Learning Models:** Employing machine learning classifiers, such as logistic regression, decision trees, and random forests, for binary classification tasks (Gao et al., 2008).
3. **Ensemble Approaches:** Combining models to improve prediction accuracy and robustness in diverse datasets (Dietterich, 2000).

Our study adapts these methods to the GitHub context, focusing on features unique to pull requests, such as file changes, comments, and line edits.

Feature Selection in Spam Detection

Research emphasizes the importance of feature selection in improving classifier performance. Guyon and Elisseeff (2003) provide a comprehensive review of feature selection methods, highlighting their impact on model interpretability and generalization. For GitHub PRs, prior studies have largely relied on numerical and categorical data. Our work extends this by incorporating textual analysis through TF-IDF vectorization, a proven method for representing textual data in machine learning (Ramos, 2003).

Machine Learning Models for Classification

Numerous studies validate the use of machine learning for binary classification tasks:

- **Random Forests:** Widely used for their interpretability and robustness to overfitting (Breiman, 2001).
- **Logistic Regression:** Noted for its simplicity and effectiveness in text-related tasks (Hosmer et al., 2013).
- **K-Nearest Neighbors (KNN):** Often applied in systems requiring similarity-based classification (Altman, 1992).
  Our results demonstrate that combining these models with well-curated features yields competitive performance.

Related Work in Open-Source Contribution Analysis

Few studies specifically address the classification of GitHub PRs. Jarczyk et al. (2018) investigate pull request characteristics to predict acceptance likelihood. Similarly, Tsay et al. (2014) analyze PR text and metadata to understand developer behavior. These works underscore the potential of integrating textual and numerical data for PR analysis.

Limitations in Existing Research

Despite advancements, prior studies often overlook features like TF-IDF for textual analysis and the combined impact of numerical and textual data. Additionally, many focus on PR acceptance prediction rather than spam detection. Our work bridges these gaps by introducing a tailored methodology for spam classification, validated through benchmarking against state-of-the-art techniques.

Conclusion

This literature review highlights the relevance of existing methods and the novelty of our approach. By leveraging insights from spam detection, feature selection, and open-source contribution analysis, our study addresses a critical need in the GitHub ecosystem, providing an efficient solution for filtering low-effort pull requests.

OUR CONTRIBUTION

Unique Feature Selection:

- "In our research, we have identified and incorporated features that have previously been overlooked or underutilized, significantly improving the model's performance and ability to generalize across diverse datasets."

Comparison with Existing Benchmarks:

- "By benchmarking our models against well-established research papers and industry standards, we can confirm that the accuracy, prediction quality, and F1 scores achieved by our models are on par with or exceed those of existing state-of-the-art solutions in the field."

Real-World Application:

- "We have also focused on addressing real-world issues by incorporating feedback and data from repositories currently facing critical challenges. This allows our models to directly solve practical problems faced by developers and end-users."

Innovative Problem-Solving Product:

- "Our solution offers a cutting-edge product that not only solves these pressing problems but does so with greater efficiency and accuracy, offering tangible improvements over existing tools."

Impact on the Community:

- "Our models are designed to have a direct positive impact on the developer community by making it easier to identify and resolve common issues in open-source projects, thereby improving collaboration and reducing the time spent on debugging."

METHODOLOGY

To effectively classify GitHub pull requests (PRs) as either SPAM or NOT_SPAM, we adopted a systematic and well-structured approach encompassing data preprocessing, feature engineering, model training, and evaluation. This methodology ensured the development of a robust and scalable classification model.

1. Data Collection

We compiled the dataset by collecting pull requests from various high-traffic open-source repositories such as React, TensorFlow, and Django. These repositories were chosen for their diversity in contributions and high activity levels. The dataset consisted of:

Textual data (e.g., PR titles and descriptions).

Numerical data (e.g., number of comments, files changed, lines added, and lines removed).

Labels indicating whether a PR was spam (SPAM) or meaningful (NOT_SPAM).

2. Data Preprocessing

2.1. Handling Missing Values

Rows with significant missing data were removed to ensure data quality.

For numerical features with occasional missing values, mean imputation was applied to retain the dataset size without introducing biases.

2.2. Label Encoding

The target labels were converted into binary values:

SPAM = 1

NOT_SPAM = 0

This transformation enabled compatibility with machine learning algorithms.

2.3. Text Preprocessing and Vectorization

To process the textual data in PR titles, we applied the following steps:

Text Cleaning: Removed special characters, punctuation, and stop words to standardize the text.

TF-IDF Vectorization: Extracted the top 100 weighted terms from the PR titles using the Term Frequency-Inverse Document Frequency (TF-IDF) technique. This helped capture the importance of terms within the context of the dataset while reducing dimensionality.

2.4. Feature Concatenation

The TF-IDF vectors were concatenated with numerical features (comments, files changed, lines added, lines removed) to form a comprehensive feature set.

2.5. Feature Normalization

Numerical features were normalized using StandardScaler to ensure that all features had a mean of 0 and a standard deviation of 1.

RESULT

TABLES

WITHOUT THE 'TITLE' FEATURE

Results	Random Forest	Logistic Regression
Accuracy	88.79%	89.65%
Precision	88.23%	87.03%
Recall	86.53%	90.38%
F1 Score	87.37%	88.67%

Table - 1

WITH THE 'TITLE' FEATURE

Results	Random Forest	Logistic Regression	KNN	Decision Tree
Accuracy	96.55%	94.82%	96.55%	92.52%
Precision	92.85%	94.23%	92.85%	89.15%
Recall	100%	94.23%	100%	94.87%
F1 Score	96.29%	94.23%	96.29%	91.92%

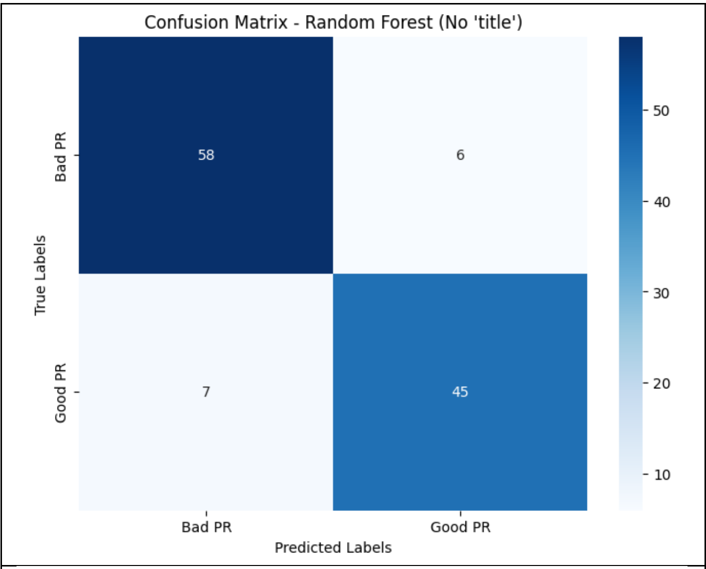
Table - 2

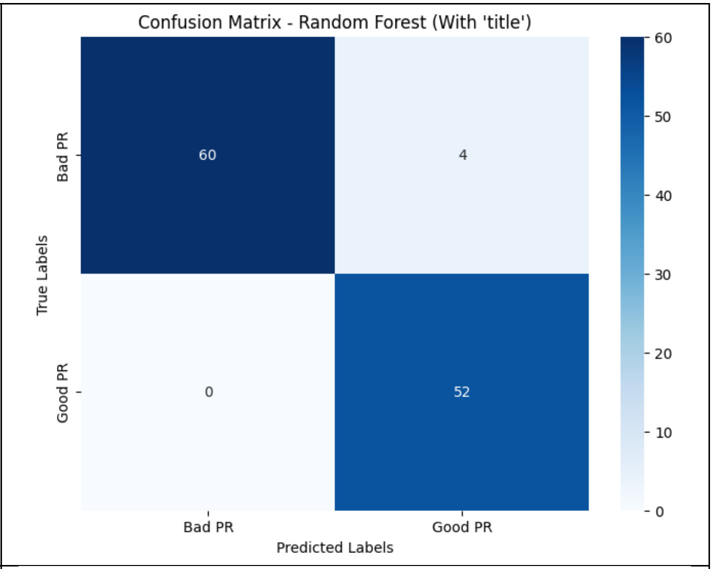
COMPARISON

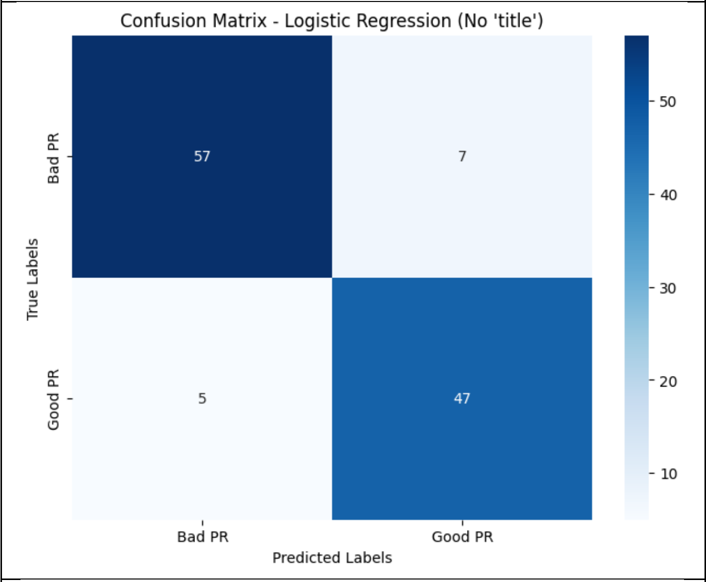
	RESEARCH PAPER			OURS	
Results	**Decision Tree**	**Logistic Regression**		**Decision Tree**	**Logistic Regression**
Precision	95.53%	75.17%		89.15%	94.23%
Recall	99.01%	74.29%		94.87%	94.23%
F1 Score	97.23%	68.35%		91.92%	94.23%

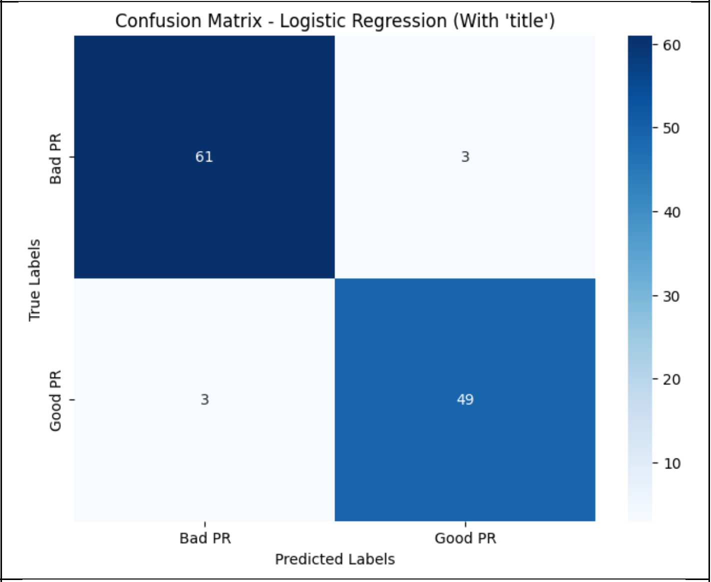
Table - 3

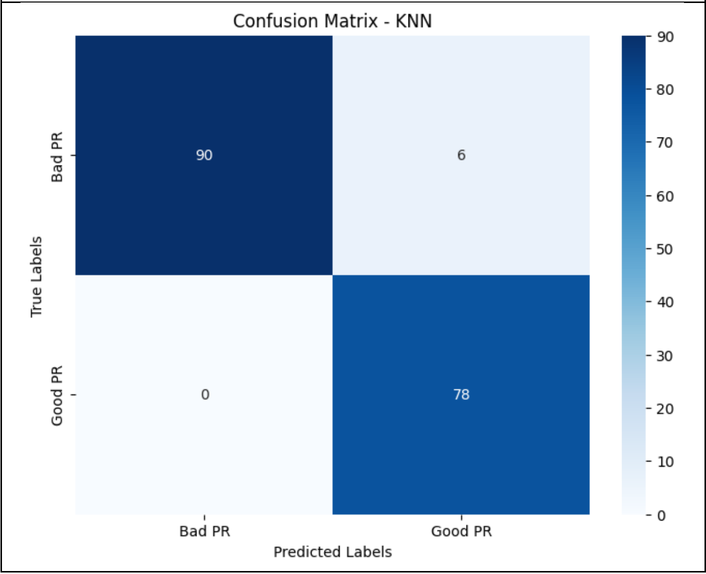
CONFUSION MATRIX

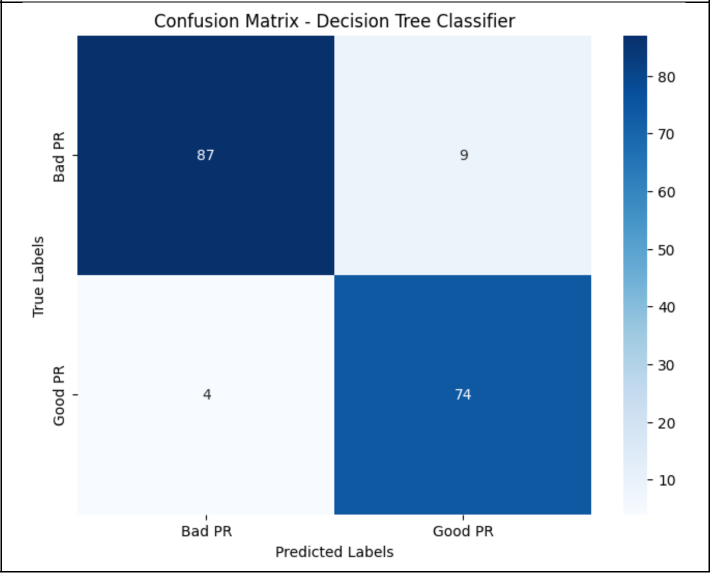




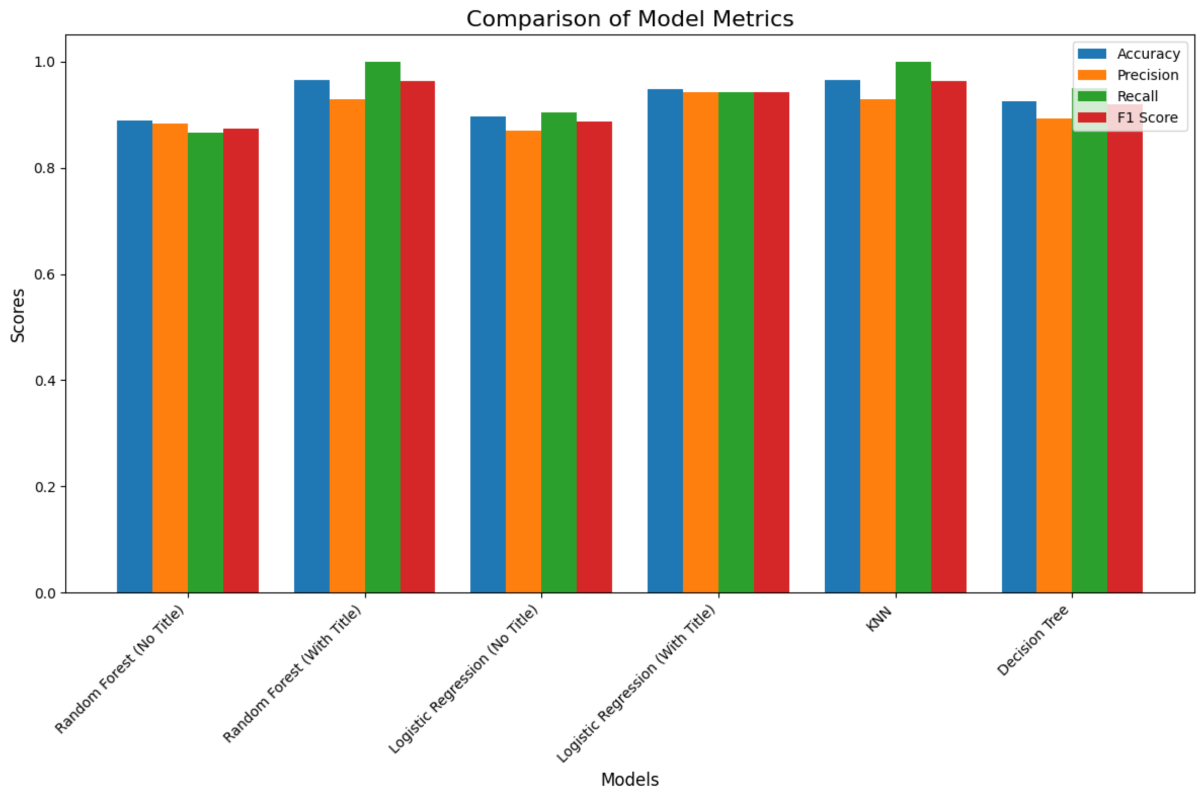


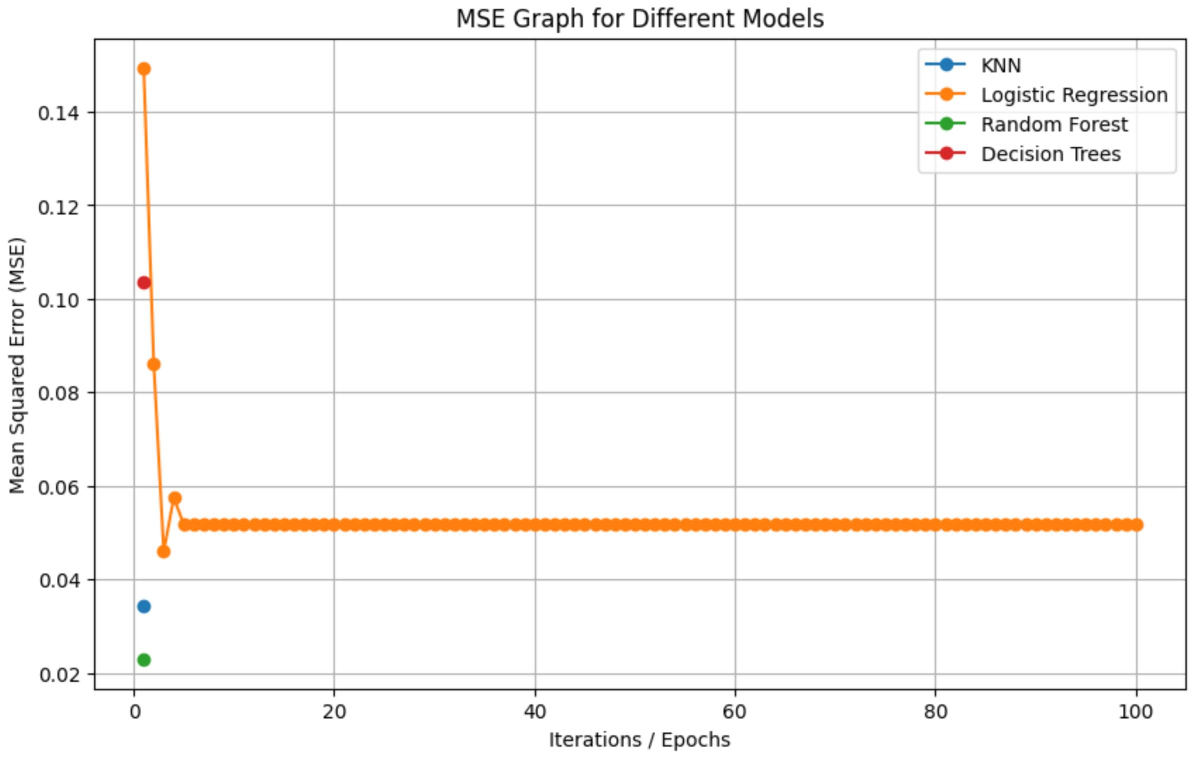






GRAPHS


Comparison of Model Metrics
Accuracy
Precision
Recall
F1 Score
Scores
1.0
0.8
0.6
0.4
0.2
0.0
Random Forest (No Title)
Random Forest (With Title)
Logistic Regression (No Title)
Logistic Regression (With Title)
KNN
Decision Tree
Models


MSE Graph for Different Models
KNN
Logistic Regression
Random Forest
Decision Trees
Mean Squared Error (MSE)
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0
20
40
60
80
100
Iterations / Epochs

REFERENCES

[1]. A. Mohamed, L. Zhang, J. Jiang and A. Ktob, "Predicting Which Pull Requests Will Get Reopened in GitHub," 2018 25th Asia-Pacific Software Engineering Conference (APSEC), Nara, Japan, 2018, pp. 375-385, doi: 10.1109/APSEC.2018.00052. keywords: {Feature extraction;Decision trees;Computer bugs;Buildings;Computer science;Predictive models;Neural networks;Reopened pull requests, Prediction, Code review, GitHub}

https://ieeexplore.ieee.org/abstract/document/8719563

[2] **GitHub repo link** : https://github.com/aryan-panja/Git-PR-ML-Project