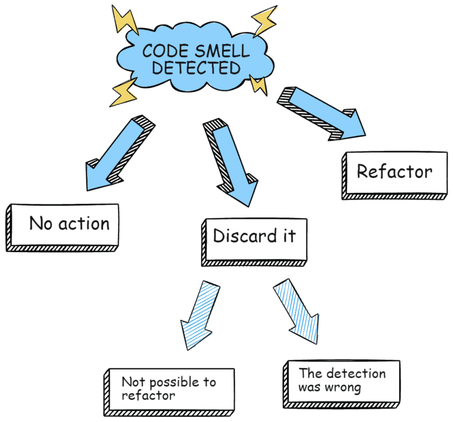
Enhancing Software Quality: A Machine Learning Approach to Python Code Smell Detection and Refactoring

Jannatul Ferdoshi (20301193), Kazi Zunayed Quader Knobo (20241020), Shabab Abdullah(20301005), Mohammed Sharraf Uddin (20241018)

Supervisor: Md. Aquib Azmain

Abstract

Python's popularity has surged due to its versatility in software and data science projects, leveraging features like classes, lambda functions, and list comprehension. However, this flexibility poses a risk of code smells, impacting software quality. While Java has extensive research on code smells, Python lacks automated solutions. Our paper addresses this gap by creating a dataset for machine learning-based code smell detection and automated refactoring in Python, contributing crucial advancements in an emerging field.



Research Objectives

Our study conducts the following research objectives

- **D**esign a Python dataset that would allow machine learning algorithms to effectively recognize python code smells
- **D**etect Code Smells: Large Class, Long Method, Long Message Chain, Long Parameter List, Long Lambda Function.
- **R**efactor desired code smells.
- **T**rack code smell improvement percentage post-refactoring.

Related Works

- **PyNose:** PyNose, a novel Python tool by Golubev et al., fills a gap in test smell detection. Integrated as a PyCharm plugin, it detects and customizes 17 Python-specific test smells with impressive precision rates in evaluations using curated datasets.[2]
- **FaultBuster:** FaultBuster, an automatic code smell refactoring toolset, by Nagy et al., facilitates continuous refactoring with a framework, IDE plugins, and a Java Swing client, resolving 11,000 code-related issues across 5 million lines of code. [3]
- **Pysmell:** Chen et al. filled the gap in automated Python code smell detection with Pysmell, achieving a 98% precision and 100% mean recall in their evaluation.[1]

The works on PyNose, FaultBuster, and Pysmell collectively underscore the evolving strategies for detecting and refactoring code smells.

Workflow

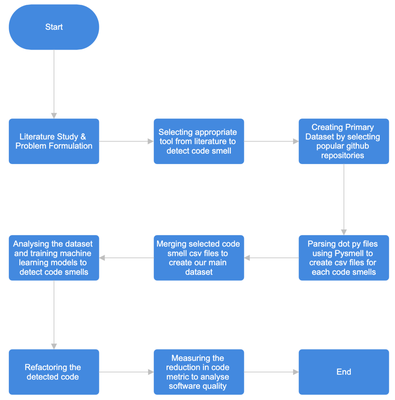


Figure 1. Flow Chart

Dataset

We chose 50 popular GitHub projects, like Keras and Django, for our primary dataset. Using Pysmell, we created a secondary dataset with ten code smells from 33 projects. The final "code smell dataset" merged five code smells, ensuring balance between smelly and non-smelly instances. This dataset underwent detailed analysis, preprocessing, and model evaluation in our research.

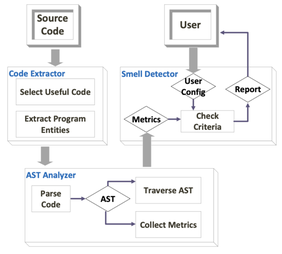


Figure 2. Architecture of Pysmell

Methodology

The optimal architecture materialized with two hidden layers, the first comprising twelve nodes and the second six nodes.

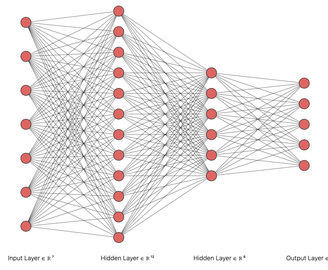


Figure 3. Architecture of The Neural Network

Preliminary Analysis

In order to understand the relationship between the input features and the output class label, we have analyzed the data. This inspection of data was graphically inspected with the aid of python libraries like 'seaborn' and 'matplotlib'.

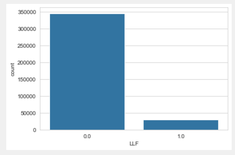


Figure 4. LLF

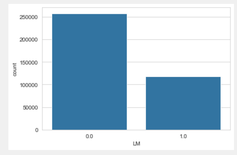


Figure 5. LM

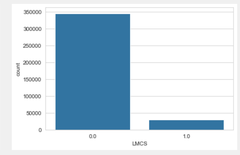


Figure 6. LMCS

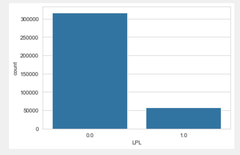


Figure 7. LPL

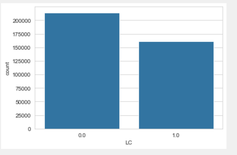


Figure 8. LC

Results

The performance of the ensemble techniques is outlined below in the table:

Ensemble Models	Accuracy	Precision	F1-Score	Recall	Hamming Loss
AdaBoost	69%	96%	83%	73%	0.06
XgBoost	100%	100%	100%	100%	0.00000798
Random Forest	100%	100%	100%	100%	0.00000533

Table 1. Performance metrics for Ensemble Learning Models

From the above table we can evaluate that XgBoost and Random Forest were absolute best in detecting code smells with 100% accuracy, precision, F1-score and Recall.

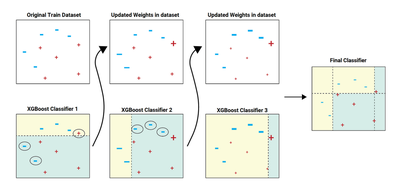


Figure 9. XgBoost

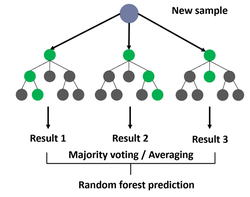


Figure 10. Random Forest

References

[1] Zhifei Chen, Lin Chen, Wanwangying Ma, and Baowen Xu.
Detecting code smells in python programs.
In *2016 International Conference on Software Analysis, Testing and Evolution (SATE)*, pages 18 – 23, 2016.

[2] Tongjie Wang, Yaroslav Golubev, Oleg Smirnov, Jiawei Li, Timofey Bryksin, and Iftekhar Ahmed.
Pynose: A test smell detector for python.
In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2021.

[3] Gábor Szo˝ke, Csaba Nagy, Lajos Jeno˝ Fülöp, Rudolf Ferenc, and Tibor Gyimóthy.
Faultbuster: An automatic code smell refactoring toolset.
In *2015 IEEE 15th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, pages 253 – 258, 2015.