Optimizing Python Code Metrics Feature Reduction Through Meta-Analysis and Swarm Intelligence

Marina Ivanova[a], Zamira Kholmatova[b] and Nikolay Pavlenko[c]
Innopolis University, Innopolis, Universitetskaya str., 1 bldg, Russia
{*m.ivanova, z.kholmatova, n.pavlenko*}*@innopolis.ru*

Keywords: Code Metrics, Feature Reduction, Meta-Analysis, Artificial Bee Colony.

Abstract: Feature selection plays an important role in reducing the complexity of datasets while preserving the integrity of data for analysis and predictive tasks. This study tackles this problem in the context of optimizing metrics for Python source code quality assessment. We propose a combination of meta-analysis with the Modified Discrete Artificial Bee Colony (MDisABC) algorithm to identify an optimal subset of metrics for evaluating code repositories. A systematic preprocessing step using correlation-based thresholds (0.7, 0.8, 0.9) through random-effects meta-analysis effectively reduces redundancy while retaining relevant metrics. The MDisABC algorithm is then employed to minimize Sammon error, ensuring the preservation of structural properties in the reduced feature space. Our results demonstrate significant error reductions, faster convergence, and consistent identification of key metrics that are critical for assessing code quality. This work highlights the utility of integrating meta-analysis and nature-inspired algorithms for feature selection and establishes a foundation for scalable, accurate, and interpretable models in software quality assessment. Future research could expand this methodology to other programming languages and explore alternative algorithms or cost functions for more comprehensive evaluations. All the relevant code can be found on our GitHub repository*.

1 INTRODUCTION

Software metrics play a crucial role in project evaluations, planning, risk estimation, and quality assurance (Jabborov et al., 2023). However, the variety of the software metrics causes a number of issues related to computation complexity, time, and resources needed, and the possibility of diverting excessive attention to irrelevant metrics. Therefore, the ability to choose a minimal set of relevant metrics that could precisely represent and evaluate the source code of the product could help to efficiently estimate the quality of the software (Bugayenko et al., 2024). This can be used for determining the choice of libraries, tools, and dependencies during situations like development planning, product contests, and tenders. An optimized method for swiftly and efficiently estimating software product quality can be integrated into CI/CD pipelines, enabling real-time quality tracking and predictive insights into project success by monitoring quality improvements and ensuring compliance with predefined requirements and standards.

Addressing the challenge of feature selection in software metrics requires balancing competing objectives: reducing dimensionality while maintaining accuracy and computational feasibility. Traditional methods often fall short when faced with the high-dimensional, non-linear, and heterogeneous nature of source code datasets. Nature-inspired optimization algorithms, such as Genetic Algorithms (GAs), Particle Swarm Optimization (PSO), and Artificial Bee Colony (ABC) algorithms, offer a promising solution to these challenges. These algorithms mimic biological or physical processes to efficiently explore vast solution spaces and are particularly well-suited to the combinatorial nature of feature selection problems. For example, ABC algorithms, inspired by the foraging behavior of bees, excel at optimizing binary solutions like selecting subsets of metrics.

While nature-inspired algorithms can effectively identify optimal subsets of features, their success is highly dependent on the quality of the input data. This is where meta-analysis plays a critical role. By systematically aggregating and analyzing metric correlations across multiple datasets, meta-analysis provides an efficient framework for identifying redundant met-

[a] https://orcid.org/0009-0002-6841-3736
[b] https://orcid.org/0000-0003-1688-1183
[c] https://orcid.org/0009-0008-0066-5252
*https://github.com/Daru1914/ABCFeatureSelector

Ivanova, M., Kholmatova, Z. and Pavlenko, N.
Optimizing Python Code Metrics Feature Reduction Through Meta-Analysis and Swarm Intelligence.
DOI: 10.5220/0013377200003896

In *Proceedings of the 13th International Conference on Model-Based Software and Systems Engineering (MODELSWARD 2025)*, pages 338-345
ISBN: 978-989-758-729-0; ISSN: 2184-4348

rics and reducing dataset dimensionality before optimization. This preprocessing step not only enhances the performance of optimization algorithms but also ensures that the selected features capture the most informative and independent aspects of source code quality.

Despite their potential, the combined use of meta-analysis and nature-inspired algorithms for feature selection in software metrics remains underexplored. To bridge this gap, this study aims to integrate these approaches, leveraging meta-analysis for data-driven dimensionality reduction and employing a Modified Discrete Artificial Bee Colony (MDisABC) algorithm for optimal feature subset selection. The ultimate objective is to develop a scalable and accurate model for assessing Python code repositories, enabling efficient quality assurance and comparative analysis.

The remainder of this paper is organized as follows. In Section 2, we provide a detailed review of existing literature on feature selection methods, with a focus on nature-inspired algorithms and their application to software metrics. Section 3 outlines the methodology employed in this study, including dataset collection, preprocessing steps using meta-analysis, and the implementation of the Modified Discrete Artificial Bee Colony algorithm. Section 4 presents the experimental results and evaluates the performance of the proposed approach. A discussion of the findings, their implications, and potential limitations is provided in Section 5. Finally, Section 6 concludes the paper by summarizing the key contributions and suggesting directions for future research.

2 LITERATURE REVIEW

Feature selection is the process of choosing a subset of features of the dataset for a model. It is a task that has been explored by researchers for decades, with many approaches emerging as a result. Recently, more than a hundred meta-heuristics from the field of nature-inspired algorithms were developed to solve this task (Sharma and Kaur, 2021)(Abu Khurma et al., 2022). Many of the nature-inspired algorithms are similar in performance to the genetic algorithm (GA) (Holland, 1975)(Goldberg, 1988)(Michalewicz and Schoenauer, 1996), which is one of the most often-used algorithms for feature selection (Colaco et al., 2019). In (Meiri and Zahavi, 2006) the authors performed a comparison between Simulated Annealing (SA)(Pincus, 1970)(Khachaturyan et al., 1979)(Kirkpatrick et al., 1983) and Stepwise Regression (SWR) (Efroymson, 1960) for feature selection. They proved that SA is more stable in handling complex data, while SWR performed comparably in simpler datasets. SWR was faster but more sensitive to parameter choices. In (Peng et al., 2018a), the authors used Ant Colony Optimization (ACO) algorithm, termed FACO, for feature selection. They enhanced the pheromone updating mechanism and designed a fitness function to prevent the algorithm from falling into local optima. FACO was then compared to existing algorithms. FACO outperformed the existing ACO-based algorithms by converging faster, achieving higher classification accuracy, and reducing false positive rates. The algorithm also improved classification performance across different classifiers, particularly KNN and SVM (Peng et al., 2018b). In (Schiezaro and Pedrini, 2013), researchers explored a binary version of the Artificial Bee Colony (ABC) algorithm for feature selection, which operates using forward search strategies. They applied this method to multiple UCI datasets and compared its performance to PSO, ACO, and GA. The proposed ABC method consistently reduced the number of selected features while maintaining or improving classification accuracy compared to other methods and outperformed other nature-inspired algorithms in terms of feature reduction and accuracy on most datasets.

In (Hancer et al., 2015) authors introduce the Modified Discrete ABC (MDisABC) improvement of the binary ABC algorithm for feature reduction, and compare it to other algorithms, such as DisABC (Kashan et al., 2012), AMABC (Pampará and Engelbrecht, 2011), and MRABC (Akay and Karaboga, 2012). MDisABC achieved better accuracy and classification performance after feature optimization in the latter two cases and generalized better to large datasets compared to the first one, proving it is more effective in feature selection problems.

Insights from feature selection algorithms can be enhanced by a prior meta-analysis (Borenstein et al., 2021), which identifies consistent patterns across multiple repositories. By aggregating correlation data using fixed-effects and random-effects models, meta-analysis reveals significant relationships that persist across different repositories, offering robust criteria for identifying redundant metrics. This systematic approach allows for the removal of highly correlated features while ensuring that important variability across repositories is considered, ultimately improving the efficiency and accuracy of the feature selection process. Additionally, this step reduces the dimensionality of the dataset, enabling optimization algorithms to operate more effectively on a refined set of features.

In the end, our findings identified the lack of research in the optimization of feature selection in soft-

ware development, in particular, source code and its quality properties assessment.

3 METHODOLOGY

3.1 Task

The primary objective of this study is to determine an optimal subset of metrics that quantify or evaluate properties of individual methods (or functions) within a software codebase while minimizing the dimensionality of the data. Specifically, starting with a dataset represented as an $n \times m$ matrix of real values (n - number of methods and m - number of metrics), the goal is to reduce the number of metrics m as much as possible while preserving the pairwise distances between norms, measured using the ℓ_2 norm.

To achieve this, the task involves:

1. **Preprocessing and Meta-Analysis** ensured that only the most independent and informative metrics were retained for further analysis, thereby reducing computational overhead and improving the efficiency of the optimization algorithm.
2. **Optimization Algorithm Selection and Implementation.** For feature selection, we employed a MDisABC algorithm with the Sammon error as an objective function. By minimizing this error, the algorithm ensures that the reduced dataset retains the structural properties of the original.
3. **Application of Feature Optimization.** The MDisABC algorithm was applied separately to each repository, considering the unique characteristics of codebases (e.g., differences in structure and complexity). This ensured that the selected subsets of metrics were optimally tailored to each dataset while maintaining consistency in evaluation criteria.
4. **Performance Evaluation and Validation:** The effectiveness of the selected metric subsets was evaluated by calculating the average and best Sammon error for each repository. Additionally, we compared results across different levels of dimensionality reduction achieved through meta-analysis.

3.2 Dataset Collection

In order to collect Python source code metrics, we selected repositories for the dataset using PyGithub (pyg, 2024). PyGithub is a Python library, aiming to provide optimized access to the GitHub API (git, 2024; Kalliamvakou et al., 2015) from a Python script, overcoming the complexity and maintainability issues of handling HTTP API requests and responses. Through PyGitHub we selected code repositories, written in Python, that satisfy the quality rules that we set:

- the repository is not archived, open access and public;
- the repository is not a dataset;
- the main language of the repository code is Python;
- the repository contains at least 10 commits, 10 stars, and 10 forks;
- the repository has at least 500000 lines of code.

We selected 100 repositories, corresponding to these criteria, in descending order, starting from the largest amount of stars.

In the research, we focused on the methods metrics of the Python source code, as the most frequent and meaningful unit in Python. In order to extract and calculate the metrics, we used the SourceMeter tool (sou, 2024), as it offers a good variety of metrics and presents well-organized datasets, and Radon (rad, 2024), as it covers complexity metrics of the code, such as Cyclomatic Complexity and Halstead metrics. We computed metrics for each repository using the tools of our choice. The final datasets contains 35 method metrics, described in our Github repository. Further, we split the largest datasets to increase the performance and decrease the memory requirements of the algorithm. Finally, we rescaled the metrics to the $[0, 1]$ range in order to allow Sammon error as the cost function to have some meaning.

3.3 Meta-Analysis for Correlation-Based Dimensionality Reduction

Some of the metrics extracted by SourceMeter failed to gather any results, leading to the existence of several zero columns across all parts of our dataset, as well as a few nulls. We handled the missing values by replacing them with zeros where necessary and ensured only metrics with non-zero variance were considered further.

For each repository, we computed pairwise Pearson's correlation coefficients (Eq. 1) between all metrics. Pearson's correlation is suitable for meta-analysis as it quantifies the strength and direction of linear relationships. These coefficients were later converted to Fisher's z-scores for stability:

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2 \cdot \sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \quad (1)$$

Given the heterogeneity inherent in our dataset, we relied on the random-effects model of meta-analysis to identify metrics with consistently high correlations and statistical significance across repositories as candidates for removal.

High-correlation thresholds (0.7, 0.8, 0.9) were applied to filter out redundant metrics, creating different sub-datasets retaining only the most informative and independent metrics to reduce the computational burden and compare the results of the application of the feature selection algorithm on them and on the original dataset.

3.4 Modified Discrete Artificial Bee Colony (MDisABC)

The ABC is a Swarm Intelligence algorithm primarily used for solving optimization problems. In this study, we employed a MDisABC algorithm tailored for the feature selection task (Schiezaro and Pedrini, 2013) (Hancer et al., 2015). This approach is inspired by the behavior of foraging bees, where solutions (food sources) are explored, exploited, and refined collaboratively by employed, onlooker, and scout bees.

1. **Initialization:** Create initial solutions as binary vectors, each representing a subset of features. To constrain the number of features, each binary vector includes exactly m active bits (1s), corresponding to selected features.
2. **Fitness Evaluation:** Calculate the Sammon error for each feature subset defined by the binary vectors and store it as the fitness of the solution.
3. **Exploitation Stage:** Generate a new neighbor solution for each food source through mutation and crossover using the MDisABC algorithm:
   (a) Select three random neighbors X_{r1}, X_{r2}, X_{r3} for the current food source X.
   (b) Compute the Jaccard dissimilarity score (Eq. 2) between X_{r2} and X_{r3}. The Jaccard metric is particularly suited for binary representations, as it measures the proportion of differing elements between two subsets relative to their total unique elements. Scale this dissimilarity by a factor ϕ to set a target diversity for the mutant solution:
   $$J(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|} \quad (2)$$
   (c) Solve an optimization problem to minimize the difference between the target dissimilarity and the mutant's dissimilarity with X_{r1}.
   (d) Generate the final neighbor solution through crossover, probabilistically combining elements from the parent and mutant solutions.
4. **Neighbor Evaluation:** Calculate the Sammon error for the newly generated neighbor solution and compare it with the original food source:
   - If the neighbor's fitness is better, it replaces the original solution.
   - Otherwise, increment the exploitation count (LIMIT) for the original food source. If the LIMIT exceeds the MAX_LIMIT parameter, the food source is replaced with a new random solution (exploration stage).
5. **Best Solution Update:** After processing all food sources, update the current global best solution.
6. **Termination:** Repeat steps 2-5 until the maximum number of iterations is reached or another termination condition is satisfied.

3.5 Sammon Error

In our feature selection task, we use the Sammon Error as the fitness function when evaluating the solution to preserve the structure of data when projecting it from a high-dimensional space into a lower-dimensional space. The Sammon error measures the difference between pairwise distances in the original high-dimensional space and the reduced space:

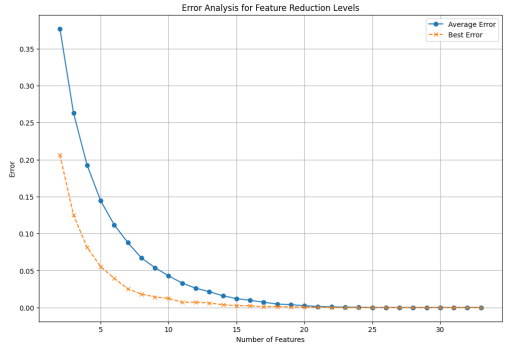
$$E = \frac{1}{\sum_{i<j} d_{ij}} \sum_{i<j} \frac{(d_{ij} - d'_{ij})^2}{d_{ij}} \quad (3)$$

where d_{ij}represents the Euclidean distance between points i and j in the high-dimensional space, and d'_{ij} is the distance in the lower-dimensional space.

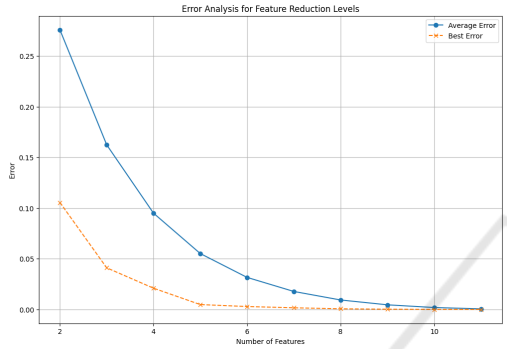
4 RESULTS

In Fig. 1 we have depicted the progression of average error reduction as the number of features in the dataset is reduced to are incrementally increased. The average error for each feature count was calculated across all repositories by aggregating the errors for a given feature count and dividing them by the total number of repositories.

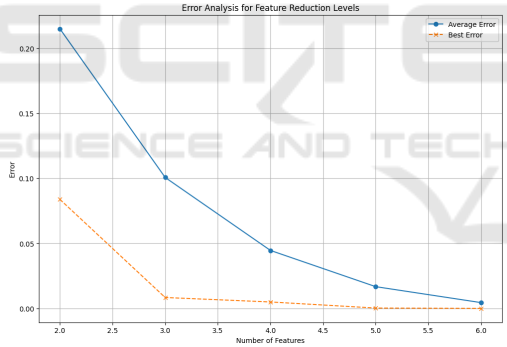
A consistent trend is observed: adding more features initially reduces the error significantly, but the rate of improvement slows beyond a certain number of features (10 in 1a, 6 in 1b, 4 in 1c). Since the case 1d removes most of the metrics from the analysis, the reduction only happens to 2, 3, and 4 metrics, and the effect cannot be readily observed.



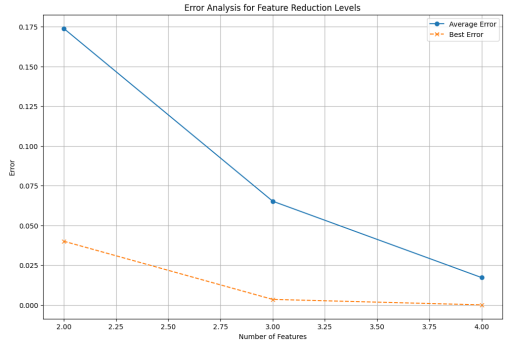
(a) Error Reduction without Meta-Analysis.



(b) Error Reduction with 0.9 Threshold.



(c) Error Reduction with 0.8 Threshold.



(d) Error Reduction with 0.7 Threshold.

Figure 1: Comparison of Error Reductions Across Different Meta-Analysis Thresholds.

In all cases except the one without meta-analysis, the best error reduction happens significantly faster than average, indicating a better convergence rate to the best solution.

In Fig. 2 we can observe the metrics that appear most often in all the subsets of reduced features. In the first dataset (2a), features like HDIF, NUMPAR, NOI, Anti Pattern, HPV, and Complexity Metric Rules dominate, with frequencies significantly higher than others. These features represent cognitive complexity, code interaction, and structural design properties, reflecting their critical role in evaluating maintainability, readability, and functional clarity. However, a long tail of less frequently selected features suggests redundancy and potential noise in the dataset.

After applying a meta-analysis threshold of 0.9 (2b), the focus on the top metrics remains, with only NOS increasing in importance. NOS (Number of Statements) is a simple size metric that quantifies the functionality of code but lacks the depth to assess complexity, interaction, or structural design compared to the other mentioned metrics.

At a stricter 0.8 threshold (2c), the dominance of NUMPAR, NOI, and Anti Pattern persists, but metrics like HDIF and HPV are no longer in the dataset, leading to an increase in the importance of purely numerical metrics like LLOC and CLOC.

With the most aggressive 0.7 reductions (2d), the remaining metrics are selected almost as equally often, suggesting that among the subsets NUMPAR, NOI, Complexity Metric Rules, CLOC, LLOC, and Coupling Metric Rules none is significantly more important than the other.

By analyzing the subset proportions across the four scenarios in Table 1, we can conclude the following:

- **Scenario 1 (No Meta-Analysis):** Without prior meta-analysis reduction, subsets with higher proportions (e.g., $k = 31/35$: `(Anti Pattern, CD, CLOC, Complexity Metric Rules, ...)` at 52.9%) dominate. The lack of dimensionality reduction results in multiple overlapping subsets (e.g., $k = 30/35$: `(Anti Pattern, CD, CLOC, Complexity Metric Rules, ...)` at 23.5%), indicating potential redundancy and complexity in feature selection. The presence of subsets like $k = 2/35$: `(Anti Pattern, HDIF)` at 17.6% suggests that smaller subsets play a limited role compared to larger, redundant subsets.
- **Scenario 2 (Threshold = 0.9):** Applying a 0.9 threshold reduces redundancy, with smaller subsets achieving higher proportions. For instance, $k = 11/13$: `(Anti Pattern, CLOC,`

Table 1: Comparison of Subset Proportions Across Different Scenarios.

Scenario	k	Subset	Proportion
1	31/35	Anti Pattern, CD, CLOC, Complexity Metric Rules, ...	0.529
1	30/35	Anti Pattern, CD, CLOC, Complexity Metric Rules, ...	0.235
1	30/35	Anti Pattern, CD, CLOC, Complexity Metric Rules, ...	0.235
1	2/35	Anti Pattern, HDIF	0.176
2	11/13	Anti Pattern, CLOC, Complexity Metric Rules, ...	0.424
2	10/13	Anti Pattern, CLOC, Complexity Metric Rules, ...	0.251
2	9/13	Anti Pattern, CLOC, Complexity Metric Rules, ...	0.172
2	2/13	Anti Pattern, HDIF	0.136
3	6/8	Anti Pattern, CLOC, Complexity Metric Rules, ...	0.259
3	5/8	Anti Pattern, CLOC, Complexity Metric Rules, ...	0.151
3	3/8	Anti Pattern, NOI, NUMPAR	0.129
3	5/8	Anti Pattern, CLOC, LLOC, NOI, NUMPAR	0.122
4	2/6	Complexity Metric Rules, NOI	0.137
4	3/6	Complexity Metric Rules, NOI, NUMPAR	0.129
4	3/6	LLOC, NOI, NUMPAR	0.115
4	4/6	CLOC, Complexity Metric Rules, NOI, NUMPAR	0.107

Scenario Deciphering:
[1] Results without any prior meta-analysis reduction.
[2] Results with a meta-analysis threshold of 0.9 applied.
[3] Results with a meta-analysis threshold of 0.8 applied.
[4] Results with a meta-analysis threshold of 0.7 applied.

`Complexity Metric Rules, ...)` achieves 42.4%, while subsets like $k = 10/13$ and $k = 9/13$ maintain proportions of 25.1% and 17.2%, respectively. Even smaller subsets, such as $k = 2/13$: `(Anti Pattern, HDIF)` at 13.6%, show a balance between reducing redundancy and maintaining predictive capacity.

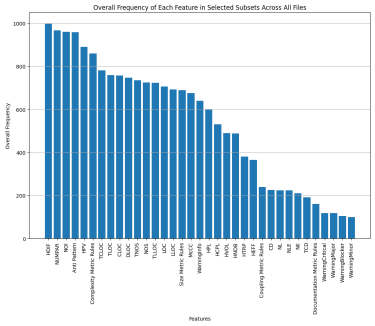
- **Scenario 3 (Threshold = 0.8):** A stricter threshold further narrows the focus, with subsets like $k = 6/8$: `(Anti Pattern, CLOC, Complexity Metric Rules, ...)` achieving 25.9%. Smaller subsets such as $k = 3/8$: `(Anti Pattern, NOI, NUMPAR)` (12.9%) and $k = 5/8$: `(Anti Pattern, CLOC, LLOC, NOI, NUMPAR)` (12.2%) gain prominence. These results suggest reduced redundancy but also indicate a shift towards smaller and more diverse subsets.
- **Scenario 4 (Threshold = 0.7):** At the strictest threshold, even smaller subsets dominate, but with reduced proportions. For example, $k = 2/6$: `(Complexity Metric Rules, NOI)` achieves 13.7%, and $k = 3/6$: `(Complexity Metric Rules, NOI, NUMPAR)` achieves 12.9%. The consistent inclusion of subsets like `(CLOC, LLOC)` and `(CLOC, LLOC, NUMPAR)` across scenarios highlights their critical importance, even in highly reduced datasets.

To conclude, the results highlight the shifting importance of specific metrics and subsets across different meta-analysis thresholds. Features such as Anti Pattern, Complexity Metric Rules, and CLOC emerge as consistently important across scenarios, particularly in smaller subsets under stricter thresholds, underscoring their relevance in predictive tasks. However, lower thresholds like 0.7 result in substantially reduced proportions, indicating a trade-off between aggressive dimensionality reduction and consistency in feature importance. While Scenario 1 (no meta-analysis) demonstrates higher proportions in larger subsets, the lack of redundancy reduction limits its practicality. Scenarios 2 (0.9 threshold) and 3 (0.8 threshold) achieve a better balance, with Scenario 2 providing more focused subsets and Scenario 3 maintaining diversity while reducing redundancy, making them the most promising approaches for practical applications depending on the desired balance between compactness and feature diversity.

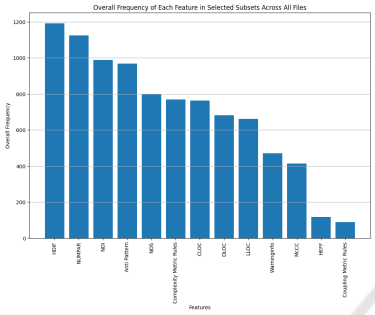
Additionally, the computational intensity of working with the full dataset highlights the practical benefits of meta-analysis-based dimensionality reduction. For instance, applying a threshold of 0.9 to remove highly correlated metrics reduced the dataset size significantly and sped up computations by approximately 10 times compared to the full dataset.

5 DISCUSSION

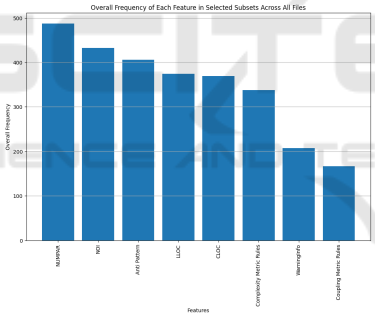
The results of this study demonstrate the utility of combining meta-analysis with feature selection algorithms to enhance the efficiency and accuracy of source code quality assessment. By systematically applying correlation-based thresholds through random-effects meta-analysis, we were able to identify and remove redundant metrics, resulting in



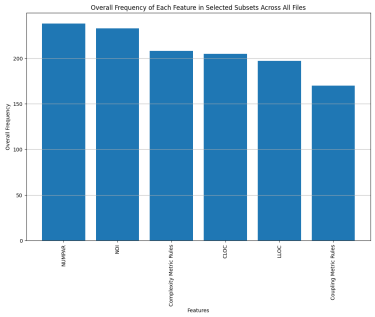
(a) Metric Selection Frequency without Meta-Analysis.



(b) Metric Selection Frequency with 0.9 Threshold.



(c) Metric Selection Frequency with 0.8 Threshold.



(d) Metric Selection Frequency with 0.7 Threshold.

Figure 2: Comparison of Metric Selection Frequencies Across Different Meta-Analysis Thresholds.

smaller, more focused subsets. This reduction in dimensionality improved the performance of the ABC algorithm, as evident from the faster convergence to lower errors and more consistent feature subsets across repositories.

The consistent selection of metrics, such as NUMPAR, NOI, Complexity Metric Rules, CLOC, and LLOC, underscores their indispensability for quality assessment across all scenarios. Metrics like HDIF and HPV played smaller but still relevant roles in reduced feature sets, indicating potential dataset-specific contributions.

The MDisABC algorithm's ability to optimize subsets effectively, even in reduced datasets, confirms its robustness as a feature selection method for binary solutions. However, the reliance on hyperparameters such as MAX_LIMIT and iteration count highlights the algorithm's sensitivity to tuning, which could be explored in future work.

An important limitation of the current study is the exclusive use of the Sammon error as the cost function. While the Sammon error is effective for preserving the structure of high-dimensional data in reduced feature sets, it may not fully capture other important aspects of the datasets or the quality of the selected features.

6 CONCLUSIONS

This study addresses the gap in feature selection optimization for software source code assessment by integrating meta-analysis with nature-inspired algorithms. The proposed methodology balances dimensionality reduction and predictive accuracy, achieving significant improvements in error reduction and feature subset selection.

Key findings include:

1. Metrics such as NUMPAR, NOI, Complexity Metric Rules, CLOC, and LLOC are consistently important across all scenarios, indicating their strong predictive relevance for Python code quality assessment.
2. Applying some meta-analysis threshold is necessary in order to find an optimal balance between reducing redundancy and maintaining accuracy.
3. The MDisABC algorithm demonstrates robust performance, effectively leveraging reduced feature sets to minimize Sammon error while exploring and exploiting the solution space.

In this work, we highlight the potential of combining theoretical frameworks such as meta-analysis with

practical optimization algorithms to address the problem of feature selection. Moreover, the proposed technique can account for metrics deemed mandatory according to the regulatory or domain-specific constraints by considering them as fixed inputs while selection process. For future work, we propose the following directions, that could extend our approach:

1. Algorithm Tuning - investigation of the MDis-ABC sensitivity to different hyperparameters values;
2. Alternative Cost Functions - incorporation of other metrics alternatives instead of Sammon error, to evaluate feature subsets;
3. Generalization - evaluation of the scalability and adaptability of this method for other programming languages or domains in order to further validate its efficiency and extend its applicability.

REFERENCES

Github rest api. https://docs.github.com/en/rest?apiVersion=2022-11-28. Accessed Nov. 27, 2024. [Online].

Pygithub. https://pygithub.readthedocs.io/en/stable/. Accessed Nov. 27, 2024. [Online].

Radon. https://pypi.org/project/radon/. Accessed Nov. 27, 2024. [Online].

Sourcemeter. https://sourcemeter.com/. Accessed Nov. 27, 2024. [Online].

Abu Khurma, R., Aljarah, I., Sharieh, A., Abd Elaziz, M., Damaševičius, R., and Krilavičius, T. (2022). A review of the modification strategies of the nature inspired algorithms for feature selection problem. *Mathematics*, 10(3):464.

Akay, B. and Karaboga, D. (2012). A modified artificial bee colony algorithm for real-parameter optimization. *Information sciences*, 192:120–142.

Borenstein, M., Hedges, L. V., Higgins, J. P., and Rothstein, H. R. (2021). *Introduction to meta-analysis*. John Wiley & Sons.

Bugayenko, Y., Kholmatova, Z., Kruglov, A., Pedrycz, W., and Succi, G. (2024). Selecting optimal software code descriptors—the case of java. *PLOS ONE*, 19(11):1–23.

Colaco, S., Kumar, S., Tamang, A., and Biju, V. G. (2019). A review on feature selection algorithms. *Emerging Research in Computing, Information, Communication and Applications: ERCICA 2018, Volume 2*, pages 133–153.

Efroymson, M. A. (1960). "multiple regression analysis.

Goldberg, D. E. (1988). Genetic algorithms in search optimization and machine learning.

Hancer, E., Xue, B., Karaboga, D., and Zhang, M. (2015). A binary abc algorithm based on advanced similarity scheme for feature selection. *Applied Soft Computing*, 36:334–348.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press.

Jabborov, A., Kharlamova, A., Kholmatova, Z., Kruglov, A., Kruglov, V., and Succi, G. (2023). Taxonomy of quality assessment for intelligent software systems: A systematic literature review. *IEEE Access*, 11:130491–130507.

Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D., and Damian, D. (2015). The promises and perils of mining github (extended version). *Empirical Software Engineering*.

Kashan, M. H., Nahavandi, N., and Kashan, A. H. (2012). Disabc: a new artificial bee colony algorithm for binary optimization. *Applied Soft Computing*, 12(1):342–352.

Khachaturyan, A., Semenovskaya, S., and Vainstein, B. (1979). Statistical-thermodynamic approach to determination of structure amplitude phases. *Sov. Phys. Crystallography*, 24(5):519–524.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Meiri, R. and Zahavi, J. (2006). Using simulated annealing to optimize the feature selection problem in marketing applications. *European journal of operational research*, 171(3):842–858.

Michalewicz, Z. and Schoenauer, M. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary computation*, 4(1):1–32.

Pampará, G. and Engelbrecht, A. P. (2011). Binary artificial bee colony optimization. In *2011 IEEE Symposium on Swarm Intelligence*, pages 1–8. IEEE.

Peng, H., Ying, C., Tan, S., Hu, B., and Sun, Z. (2018a). An improved feature selection algorithm based on ant colony optimization. *Ieee Access*, 6:69203–69209.

Peng, H., Ying, C., Tan, S., Hu, B., and Sun, Z. (2018b). An improved feature selection algorithm based on ant colony optimization. *Ieee Access*, 6:69208.

Pincus, M. (1970). Letter to the editor—a monte carlo method for the approximate solution of certain types of constrained optimization problems. *Operations Research*, 18(6):1225–1228.

Schiezaro, M. and Pedrini, H. (2013). Data feature selection based on artificial bee colony algorithm. *EURASIP Journal on Image and Video processing*, 2013:1–8.

Sharma, M. and Kaur, P. (2021). A comprehensive analysis of nature-inspired meta-heuristic techniques for feature selection problem. *Archives of Computational Methods in Engineering*, 28:1103–1127.