Journal of Software Engineering Research and Development, 2025, 13:3, doi: 10.5753/jserd.2025.4697

Feature selection in an interactive search-based PLA design approach

Caio Vieira Arasaki [**State University of Maringa (UEM)** | ***arasakivieiracaio@gmail.com***]
Lucas Wolschick [**State University of Maringa (UEM)** | ***ra123658@uem.br***]
Willian Marques Freire [**State University of Maringa (UEM)** | ***willianmarquesfreire@gmail.com***]
Aline Maria Malachini Miotto Amaral [**State University of Maringa (UEM)** | ***ammmamaral@uem.br***]

Abstract The Product Line Architecture (PLA) is one of the most important artifacts of a Software Product Line (SPL). PLA design can be formulated as an interactive optimization problem with many conflicting factors. Incorporating Decision Makers' (DM) preferences during the search process may help the algorithms find more adequate solutions for their profiles. Interactive approaches allow the DM to evaluate solutions, guiding the optimization according to their preferences. However, this brings up human fatigue problems caused by excessive interactions and solutions to evaluate. A common strategy to prevent this problem is limiting the number of interactions and solutions the DM evaluates. Machine Learning (ML) models were also used to learn how to evaluate solutions according to the DM profile and replace them after some interactions. Feature selection performs an essential task as non-relevant and/or redundant features used to train the ML model can reduce the accuracy and comprehensibility of the hypotheses induced by ML algorithms. This study aims to enhance the usage of an ML model in an interactive search-based PLA design approach by addressing two critical challenges: mitigating decision-maker fatigue through feature selection and improving computational efficiency, particularly during testing phases. We applied four selectors, and through results, We managed to reduce 30% of the features and 25% of the time spent on testing, achieving an accuracy of 99%.

Keywords: *Interactive search-based Software Engineering, Machine Learning, Feature Selection*

1 Introduction

Search-Based Software Engineering (SBSE) approaches apply search techniques to generate a set of solutions that solve complex problems of Software Engineering (SE) (Harman and Jones, 2001) based on parameters predefined by Decision Makers (DMs)[1]. These parameters are defined when the DM interacts with the approach, and this interaction may occur in three different moments, namely: *a priori*, before the beginning of the search process; *interactive*, during the process; and *a posteriori*, at the end of the process.

The most common moments of interaction are *a priori* and *a posteriori* (Amal et al., 2014; Mkaouer et al., 2013). However, knowing the DM preferences during the search process (*interactive*) may help the search algorithms to find solutions more adequate to the DM's profile.

Ferreira et al. (Ferreira FN et al., 2016) and Simons et al. (Simons et al., 2015) indicate that the obtained solutions in SBSE approaches are, in some cases, rejected by the DMs because many aspects of the problem cannot be mathematically modeled through objective functions. Therefore, it is crucial to incorporate DM preferences to improve results, considering both computational demands and DM fatigue, ensuring that the solution process is efficient and sustainable. Some researchers have explored incorporating DM preferences into the process performed by search algorithms (Ferreira et al., 2017; Ramirez et al., 2018). For this purpose, the researchers have proposed interactive processes to allow the inclusion of DM preferences during the search process.

[1]In this work, the software architect is referred to as *Decision Maker* (DM).

Interactive SBSE approaches, typically supported by evolutionary algorithms (Vikhar, 2016) (such as Genetic Algorithms), involve many interactions and produce numerous solutions for evaluation. However, due to human fatigue, it becomes impractical for DMs to evaluate all the generated solutions in each interaction. To address this, it is essential to implement strategies that prevent DM fatigue and reassure the audience about the practicality of the process.

In the context of Product Line Architecture (PLA) design, an interactive optimization approach based on the MOA4PLA (Multiobjective Optimization Approach for PLA) (Colanzi et al., 2014), was developed in Bindewald et al. (2019); Freire et al. (2019); Bindewld et al. (2020); Kuviatkovski et al. (2022); Rosa et al. (2022); Freire et al. (2020, 2022). To prevent DM exhaustion, this approach uses: (i) a *clustering algorithm* and (ii) a *Machine Learning (ML) model*. During the search process, similar solutions are generated, so evaluating all these solutions may be unnecessary and require great effort from the DM. Thus, through a *clustering algorithm*, the solutions are categorized according to their similarities, and the DM can evaluate only one solution per category. Besides, the greater the number of interactions with the DM, the better the search process results, but the more fatigued the DM can become. Thus, to reduce the excessive number of interactions, a *ML model* was developed to learn the DM's profile during their interaction, which then assumes their role in the remaining interactions of the search process.

Training is one of the most important steps in an ML solution. This step depends on data in terms of both quantity and quality. Specifically in the context of the PLA design approach presented above, during the interactions with the DM, the ML model, named OPLA-ALM, was trained with a dataset composed of **features** obtained from the evaluated PLA solutions. This dataset assumes the DM role until the end of the search process.

Feature selection plays an important role in the ML process since it represents a central problem in ML and is frequently applied as a data pre-processing step. Its objective is to choose a subset from the original features that describe a dataset according to important criteria by removing irrelevant and redundant features, as they may decrease data quality and reduce the comprehensibility of hypotheses induced by machine learning algorithms (Liu and Motoda, 2012).

This work also aims to reduce the computational time, improving DM interactions in the ML model (OPLA-ALM) within the interactive MOA4PLA approach by identifying a smaller, optimized set of features that maintains model performance. We did not apply this process to the *clustering algorithm* mentioned before since the clustering model already has only the features strictly necessary to its task (Freire et al., 2022). However, for the *ML model*, there is evidence that irrelevant features exist (Kuviatkovski et al., 2022), and thus validation is required. For this purpose, a quantitative experiment was carried out using four different feature selectors, and the results demonstrated that it is possible to reduce the dataset used to train and test the OPLA-ALM model while maintaining the accuracy of the results at 99%.

The feature selection reduces processing time when the number of features is minimized (K.Sutha and Tamilselvi, 2015). Nevertheless, the goal of the feature selectors is to support the decision of which features are irrelevant and redundant for the ML model. The time is reduced when these features are removed from the dataset, as it reduces the dimensionality of the data, consequently reducing the computational effort of training and testing the ML algorithm. Keeping this in mind, we focused on minimizing the number of features without harming the accuracy of the OPLA-ALM.

With the subsets constructed and evaluated with the performance metrics, we measured the training and testing times spent by the classification algorithm on the original set and the set with the selected features. The results showed that there was no significant reduction in training times for each DM, but we observed a considerable difference in the testing times spent. The analysis indicated that the feature selection not only maintained the accuracy of the OPLA-ALM but also improved its computational efficiency, especially during testing phases.

These time savings are of utmost importance for PLA design, as it is an iterative process where long iteration times represent a major hurdle for the software architects involved. Time and performance gains also tend to be amplified as the datasets get larger, such as with larger PLAs, which makes feature selection even more attractive in an industry context. Thereby, the main contribution of this work is a well-defined ML model for an interactive search-based PLA design approach. In summary, this study tackles computational inefficiency and DM fatigue in interactive PLA design by optimizing feature selection for the OPLA-ALM model, aiming to reduce computational costs while maintaining minimal yet effective interactions.

This paper is organized as follows: Section 2 presents the main concepts involved in this work. Section 3 describes the experiment design defined to carry out feature selection. Section 4 presents the obtained results and the threats to validity. The related works are discussed in Section 5. Section 6 concludes the paper and points out future works.

2 Interactive Search-Based PLA Design

Software Product Line (SPL) is an approach that employs reuse techniques for a software product family in a specific domain. Aiming at reducing cost and increasing productivity, this approach allows the creation and delivery of products that are variants of an initial product, inheriting common features and receiving new properties (Pohl et al., 2005).

A Product Line Architecture (PLA) defines a common design for all products derived from an SPL, including mandatory and PLA variable features. In this context, these features are user-visible aspects or characteristics of the domain (Kang et al., 1990). Figure 1 shows an excerpt of the PLA for the SPL Arcade Game Maker (AGM) (SEI, 2009). The AGM is a PLA used by a fictional game development company to develop its games, which are part of the same SPL. In this PLA design, there are five variabilities detailed in UML [2] notes attached to the classes that realize the variabilities. Variation points, alternative variants, and optional and mandatory PLA variable features are shown by the stereotypes $variationPoint$, $alternative\_OR$, $optional$, and $mandatory$, respectively, according to the SMarty approach (OliveiraJr et al., 2010). PLA variable features, such as ranking, movement, play, and save, are also tagged with stereotypes. Each architectural component is represented by a package. Attributes, operations, GUI components, and some details of variabilities were omitted to improve readability.

PLA design is a people and time-intensive task that involves multiple and often mutually incompatible quality objectives (Contieri et al., 2011). Previous work has established that this task can be automated using Search-Based Software Engineering (SBSE) techniques, which use search algorithms and quality metrics to optimize PLAs (Colanzi et al., 2014).

The MOA4PLA approach applies multi-objective evolutionary algorithms (MOEAs) to resolve the problem of search-based PLA design (Colanzi et al., 2014). This approach receives as input a class diagram containing common and variable elements of a given SPL and optimizes it so the resulting architectures (PLAs) can be evaluated and improved according to quantitative software metrics that measure software quality aspects.

However, in some cases, the DMs can reject the obtained solutions because many aspects of the problem cannot be

[2] UML stands for Unified Modeling Language, which is a standardized visual language used to design and model software systems. It provides a set of diagrams and notations that can be used to represent various aspects of a software system, such as its structure, behavior, and interactions with external systems (Booch et al., 1998).

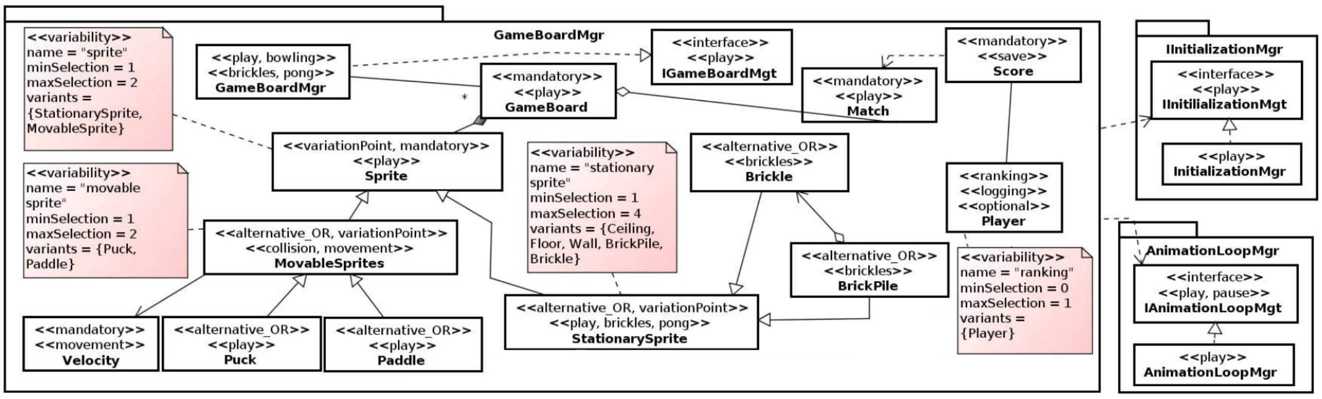


Figure 1. Excerpt of the original PLA for AGM

mathematically modeled through objective functions (Ferreira et al., 2017) (Simons et al., 2015). An objective function, also known as fitness function or evaluation function, is a measure used to determine the quality or suitability of a potential solution or candidate solution within a search space (Russell et al., 2016).

In this context, MOA4PLA also allows the DMs to guide the search process according to their preferences by interactively evaluating its intermediary solutions. This evaluation occurs by assigning a score to solutions ranging from one to five (Likert, 1932). During the evaluation, the DM can also freeze architectural elements [3] in the solutions that it deems good enough, they are not modified further in the search process.

Figure 2 presents the main steps of interactively freezing architectural elements in PLA design optimization. In summary, during the search process, at some points, the search pauses, and the DM opens optimized solutions, allowing them to freeze elements according to their preferences. As mentioned before, the frozen elements remain changeless in the whole search, being presented to the DM in the resulting solutions exactly as they have been frozen. After the evaluation, the DM closes the solutions and the search proceeds.

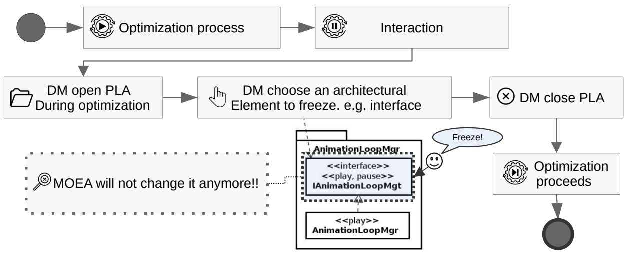


Figure 2. Architectural elements freezing

Designing PLA is complex, so evaluating too many solutions can cause cognitive stress. Nevertheless, the MOA4PLA provides techniques for dealing with human fatigue. This approach implements a machine learning (ML) model based on DM choices (**OPLA-ALM**) (Bindewld et al., 2020), which, after a configurable number of interactions, replaces the DM by evaluating the following solutions based on the DM's preferences. Since this model aims to freeze elements and evaluate the PLA solutions, all the architectural elements from the solutions that contain frozen elements, evaluated during the interactions with the DM, are used in its training. The ML implementation currently uses thirteen different features, so the OPLA-ALM can learn which elements to freeze according to DM preferences.

Unlike the PLA design feature, in machine learning, a feature refers to a measurable property or characteristic of the data used to make predictions or classifications (Bishop, 2006). Features can be thought of as the input variables or independent variables in a machine learning model.

Some of the features used in this work are the objective function (Verdecia et al., 2017) values obtained from the solutions during the search process. The features that compose the OPLA-ALM training and test dataset are presented below.

- **Feature modularization (*FM*)** provides an indicator for the PLA design based on the sum of the metrics for feature-driven cohesion, feature scattering, and feature interlacing over architectural elements.
- **Class coupling (*ACLASS*)** is calculated by the number of architectural elements that depend on other classes of the design, added to the number of elements on which each class depends.
- **Cohesion (*COE*)** evaluates the PLA design in terms of the internal relationships of the classes of the PLA design.
- **Element identifier (*ElementId*)** is the attribute that represents the unique name of the element.
- **Element type (*ElementType*)** indicates the type of the element, which can be a class, interface, attribute, or method.
- **Number of classes (*NumClasses*)** is the amount of classes inside the element. Generally, it is different from 0 only for packages.
- **Number of interfaces (*NumInterfaces*)** is the number of interfaces inside the element. Similarly to *NumClasses*, it differs from 0 only for packages.
- **Number of attributes (*NumAttributes*)** measures the number of attributes of a class and is set only when the element is a class.
- **Number of methods (*NumMethods*)** counts the number of methods in an element and is different from 0

[3] Architectural elements are components from the software architecture, e.g., attributes, methods, classes, and interfaces. These elements are the building blocks that make up the diagram's structure, allowing objects to be created from it.

only in classes and interfaces.

- **Feature modularization applied to element (*FMAppliedToElement*)** is the FM objective function value, but calculated specifically to the architectural element.
- **Class coupling applied to element (*ACLASSAppliedToElement*)** is the *ACLASS* objective function value, but specifically to the architectural element.
- **Cohesion applied to element (*COEAppliedToElement*)** is the *COE* objective function value, but for the architectural element.
- **Whether the architecture contains frozen elements (*HasFreezing*)** is also used because the training dataset does not only contain frozen architectural elements since the ML model needs to learn which elements should not be frozen too.

3 Experiment Design

This section presents the description of the experiment design, in which we **aim to** reduce the number of features of the OPLA-ALM, **with the purpose of** validating which features improve the ML model, **from the point of view of** researchers, **in the context of** interactive SBSE.

To achieve this goal, we execute a quantitative experiment using feature selectors to answer the following research question (RQ): **Is it possible to reduce the number of features used in OPLA-ALM without harming the performance of the model?**. From this question, the following hypotheses were formulated:

- **Hypotheses for RQ$_{Exp01}$:**
  - **H0$_{RQ}$:** It is not possible to reduce the number of features used in OPLA-ALM without harming the performance of the model
  - **H1$_{RQ}$:** Reducing the number of features used in OPLA-ALM improves computational efficiency and reduces decision-maker fatigue without compromising model performance

Subsections 3.1 and 3.2 provide a comprehensive overview of the methodological foundations that support this work. Considering the experiment proposed, Subsection 3.4 presents information about the dataset used, and Subsection 3.5 describes in detail the steps carried out to conduct it.

3.1 Feature Selection in Machine Learning

According to Mjolsness et al. (Mjolsness and Decoste, 2001), machine learning is the study of algorithms capable of learning to improve the performance of a task based on their own previous experience. From sample data, known as "training data". ML algorithms can build mathematical models that can make predictions or decisions without being explicitly programmed for these purposes. In evaluating OPLA-ALM, the features used are the thirteen ones listed in Section 2.

A large number of features, however, does not necessarily improve the efficiency of the training process nor the final resulting algorithm, as features can be correlated and thus may add no new information to the training process, taking up computation time, or they can actively hamper the learning process, slowing down the learning rate substantially. Therefore, reducing the number of features to the minimum is reasonable while still ensuring satisfactory training results. The process of reducing the number of features is called *feature selection*.

3.2 Feature Selection with Weka

In this work, we used the Weka machine learning data mining toolset (Witten and Frank, 2005) to select and test features for the OPLA-ALM. Weka offers feature evaluators, or selectors, which score features based on metrics calculated over their corresponding dataset; we used these evaluators to score the features described in Section 2 to assemble feature subsets with the best-scored features. In this work, we used the most used selectors, presented as follows:

- **Correlation-based Selector with best-first search (CFS-BestFirst)** (Hall, 1998) is a selector which, given a set of features, evaluates different possible subsets of this set to find a subset whose features are highly correlated with the class [4] and with low intercorrelation. It is paired with the Best-First Search algorithm, which uses greedy hillclimbing with backtracking facilities (Hall and M., 2022) to find the preferred features subset. The features in the subset resulting from CFS-BestFirst are marked as selected; for our experiments, we attributed the numerical value one to selected features and zero to unmarked features (because a feature is either selected or not selected with CFS-BestFirst, we also refer to it as a binary selector in this paper).
- **Correlation** is a numerical selector that measures the Pearson correlation coefficient (Hall, 1998) between a feature and its class; returns a numerical value between zero and one.
- **Information gain (InfoGain)** is a numerical selector that returns the information gain metric (Quinlan, 2014) for the feature about the class, defined by equation 1, where $H(\text{Class})$ is the entropy of the class and $H(\text{Class}|\text{Feature})$ is the entropy of the class given the feature's value; returns a numerical value between zero and one.

$$\text{InfoGain(Class, Feature)} = H(\text{Class}) - H(\text{Class}|\text{Feature}) \tag{1}$$

- **Gain ratio (GainRatio)** is a numerical selector which computes, through equation 2, the ratio between the information gain rate metric (defined above) concerning the class (Quinlan, 2014). Where $H(\text{Feature})$ is the entropy of the feature's value, This selector returns a numerical value between zero and one.

$$\text{GainRatio(Class, Feature)} = \frac{H(\text{Class}) - H(\text{Class}|\text{Feature})}{H(\text{Feature})} \tag{2}$$

[4]In machine learning, a class refers to a group or category that a data point or example belongs to (Bishop, 2006). The classes are typically the target or output variables the model aims to predict based on the input features.

Based on the values returned by these four selectors, we also define a metric named AV for each feature, which is an unweighted average of the averages of each selector's score for that feature across all DMs. This metric gives each feature a single score, which can be used to identify whether the selectors tended to favor it or not. Equation 3 defines this metric, where each of the overlined terms in the sum represents the average score that the feature received by the selector across all DMs.

$$AV = \frac{\overline{\text{CFS-BestFirst}} + \overline{\text{Correlation}} + \overline{\text{InfoGain}} + \overline{\text{GainRatio}}}{4} \tag{3}$$

To measure the effectiveness of a feature subset, we submit it to Weka's K* classification algorithm (Cleary and Trigg, 1995) and measure its success using statistical metrics that measure the correctness of its answers. We chose the K* classification algorithm because it obtained the best performance in previous work (Kuviatkovski et al., 2022).

In binary classification, instances are usually divided into four groups post-classification: true positives (TP), the instances that were labeled as positive and are marked as positive by the classifier; true negatives (TN), the instances that were labeled as negative and are marked as negative by the classifier; false positives (FP), instances classified as positive but which were supposed to be negative; and false negatives (FN), instances which were classified as negative but whose actual label is positive. When calculating performance statistics, the groups are usually represented as the number of elements in each set, and in multi-class classification contexts, these metrics are generally redefined in terms of being classified in the correct class or not. The metrics we used are the following ones.

- **Precision** is the ratio of instances correctly classified as positive (true positives, or TP) and the total number of instances marked as positive (true positives plus false positives, or $TP + FP$. Also called positive predictive value (PPV), it is calculated as $PPV = \frac{TP}{TP+FP}$. Values closer to 1 are better.
- **Accuracy** is the ratio of how many instances were correctly classified and all instances in a dataset (true positives, true negatives, false positives, and false negatives, or $TP + TN + FP + FN$). It is calculated as $ACC = \frac{TP+TN}{TP+TN+FP+FN}$. Values closer to 1 are better.
- **Recall** is similar to Precision, in which it is the ratio of instances correctly classified as positive (true positives) and the total number of instances labeled as positive (true positives plus false negatives, or $TP + FN$), in contrast with all instances labeled as positive. It is also known as true positive rate (TPR) and is calculated as $TPR = \frac{TP}{TP+FN}$. Values closer to 1 are better.
- **F1 score** is the harmonic mean of precision and recall. It is calculated as $F1 = 2 \cdot (\frac{1}{PPV} + \frac{1}{TPR})^{-1} = \frac{2TP}{2TP+FP+FN}$. Values closer to 1 are better.

In this work, we evaluated the time performance of ML models by using two metrics described below Bishop (2006). These metrics help us analyze the models in both the training and testing phases. By analyzing these metrics, we can ensure that the model performs well in terms of accuracy and operates within acceptable computational limits. This is crucial for interactive scenarios where timely responses are needed.

- **Elapsed time** is the total time elapsed from beginning to completing testing or training the model in seconds. It includes the CPU time dedicated to the operation and any waiting time or other resources that may be involved.
- **User CPU time millis** is the CPU time consumed exclusively by the process being executed as training or testing the model in milliseconds. It does not include waiting time or time devoted to other processes running on the system.
- **Dataset Size** is the number of lines in the CSV (comma-separated values) file used for training the ML algorithms. Each line contains information such as objective function values of the parts of the solutions (architectural elements) evaluated by the DM during optimization. So, the file contains lines, and each line contains numeric values separated by a comma. Each value represents a feature of the data.

3.3 Average Value Metric

To guide the feature selection process, the AV (Average Value) metric was introduced to aggregate feature scores from multiple selectors (Correlation, InfoGain, GainRatio, and CFS-BestFirst). The AV metric is calculated as the unweighted average of the scores assigned to each feature by the four selectors, providing a balanced view of feature relevance. This metric was chosen because it captures the strengths of each selector and mitigates their weaknesses.

To validate the effectiveness of the AV metric, we conducted a statistical comparison between the AV scores and the performance of the feature subsets they helped select. Specifically, we computed the correlation between AV scores and key performance metrics of the ML model, including accuracy, precision, recall, and F1-score. The results showed a strong positive correlation ($r > 0.7$) between high AV scores and high model performance, confirming that the AV metric reliably identifies relevant features.

Empirical testing further validated the AV metric's utility. Feature subsets selected using the highest AV scores consistently maintained model accuracy at 99%, while reducing computational time by 25%. For example, subsets B_d and A_7, identified based on AV scores, demonstrated no loss in performance compared to the original feature set, supporting the metric's practical value in maintaining model quality with fewer features.

The AV metric balanced feature selection by leveraging multiple evaluative perspectives. It enabled the identification of irrelevant and redundant features that did not contribute to model performance, guiding the selection of optimized subsets that reduced computational effort. This balanced approach is precious in interactive PLA design, where computational efficiency and DM engagement are critical.

In summary, the AV metric is a practical tool for aggregating selector insights and a validated component of our feature selection strategy that directly supports the study's objectives of reducing computational time and mitigating DM fatigue.

3.4 Dataset

To conduct the experiment defined in this work, we used a dataset resulting from experiments already conducted in (Kuviatkovski et al., 2022). This set contains data from eleven DMs interacting with the MOA4PLA approach while searching and using the AGM PLA (presented partially in Figure 1). This dataset is available in the complementary material (Arasaki et al., 2023).

Although the AGM PLA was originally designed for pedagogical purposes, it provides a well-defined and representative structure for evaluating feature selection in SPLs. The modular nature of AGM facilitates the testing of variability and commonality management techniques. While AGM is a controlled setting, future work will explore its application to larger and more complex PLAs to confirm the scalability and generalizability of the findings.

3.5 Experiment Steps

Figure 3 presents the steps defined to carry out the proposed feature selection experiment.

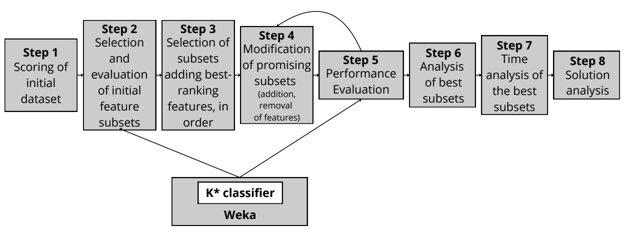

Step 1 Scoring of initial dataset
Step 2 Selection and evaluation of initial feature subsets
Step 3 Selection of subsets adding best-ranking features, in order
Step 4 Modification of promising subsets (addition, removal of features)
Step 5 Performance Evaluation
Step 6 Analysis of best subsets
Step 7 Time analysis of the best subsets
Step 8 Solution analysis
K* classifier
Weka

Figure 3. Experiment steps

We evaluated feature subsets from the thirteen original OPLA-ALM features in this experiment using Weka's feature selectors and its K* classification algorithm implementation. We also compared the results with the Multilayer Perceptron (MLP) (Rumelhart et al., 1986) algorithm since it was used in previous works with the same goal (Bindewald et al., 2019) (Bindewld et al., 2020). We used the dataset described in the last subsection and the DMs' decisions for the training process. Our evaluation process (Figure 3) was iterative and procedural, with later steps depending on the outcome of earlier steps. In general, our process can be summarized in the following steps.

Step 1. We executed the four feature selectors described in subsection 3.2 for all features on the dataset and plotted the resulting values in two graphs, one for each selector type — numerical and binary. We also calculated the AV metric for all features according to the earlier formulae. Hereafter, we scored and ranked the features according to this metric and then separately analyzed the graphs to determine the promising features.

Step 2. Considering the results of Step One, we built subsets of features by adding features with relevant results for the selectors. The assembly of these subsets will be presented in the next section. We selected and evaluated the features in this subset using the Weka K* classification algorithm and measured the performance of the subsets using the statistical metrics of accuracy, precision, recall, and the F1 score, all relative to the baseline results obtained by the entire thirteen feature set (also calculated in this step).

Besides, we tested the subset of features recommended by CFS-BestFirst, as this algorithm already executes a search over the feature space and returns the best subset, considering subsets containing features with high correlation with the class and low intercorrelation. This subset was also evaluated after implementing the K* classification algorithm to measure accuracy, precision, recall, and F1 score.

Step 3. We also tried to build subsets based on the sorting induced by AV. We started with the feature with the greatest AV and then incrementally added the remaining twelve features in descending order of AV. Afterward, we conducted one test for each addition. All built subsets were evaluated with the K* classification algorithm.

Step 4. Considering these subsets, we determined which of the two approaches had better results considering the performance metrics mentioned above. We then modified the best subset by adding each of the features not in the subset to it one at a time; we also tried removing one of the features included in the subset one at a time.

Step 5. We evaluated the subsets built in step three with the K* classification algorithm. From the most promising subset obtained from this evaluation, we tried modifying it by adding one more feature from the features not included, as done in step three and repeated the K* classification.

Step 6. We analyzed the resulting performance metrics and determined the smallest feature subset with equal or better performance than the original feature set.

Step 7. We checked the difference in the Elapsed time and User CPU time millis time metrics of the best subsets compared to the original feature set. Also, we checked the size of the datasets used for training the models.

Step 8. We analyzed the solutions generated by the optimization process using the selected features. This was needed to ensure that the selected features led to high-quality solutions aligned with the DM's preferences. By evaluating the solutions, it was possible to verify if the reduced feature set maintained or even enhanced the quality of the PLA designs, confirming (or not) that irrelevant and redundant features had been effectively removed.

3.6 Step 8 - Solutions Analysis Protocol

Figure 4 details the steps needed to analyze the solutions. These steps are described below.

Firstly, as presented in Steps 1 and 2 of Figure 4, we ran the optimization process with the selected features and all the features to perform solutions analysis. The optimization was performed interactively, as defined in (Freire et al., 2020). So, we ran the optimization algorithm with the interaction enabled during the process. This makes the optimization algorithm show the optimized solutions to the DM evaluation after evolving it during the optimization process. Our first configuration made in the algorithm was the interval for showing optimized solutions during the process. We defined that they would be presented after every three generations of the algorithm-optimizing solutions. This definition was based on

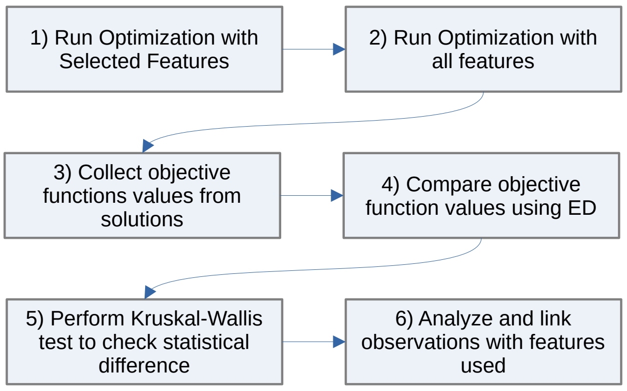

1) Run Optimization with Selected Features
2) Run Optimization with all features
3) Collect objective functions values from solutions
4) Compare objective function values using ED
5) Perform Kruskal-Wallis test to check statistical difference
6) Analyze and link observations with features used

Figure 4. Solution Analysis Steps

previous works such as (Bindewld et al., 2020), so the solutions can be evolved before the DM analyzes them. In addition, the algorithm configurations were also based on this work, such as 200 individuals and 3000 iterations for the optimization of the PLA.

As mentioned in Section 2, the DMs assign a score between 1 and 5 to the solutions in each evaluation. Score 1 indicates that the solution is the worst for the DM, and 5 indicates that the solution is the best from their point of view. This score impacts the evolution of solutions in the following way: solutions with greater scores have more chance to survive and be present in results; on the other hand, solutions with lower scores have less chance to survive the optimization process. In the solution analysis, we defined a pattern of choosing solutions to evaluate and assigning the score. We prioritize solutions with the best feature modularization, which organizes the features to maximize their individual contributions to the overall performance. So, in every interaction, we chose four solutions with the best value for the FM objective function and assigned a score of 4 for the solutions. We did not assign the score 5 since, as presented by (Freire et al., 2020), the algorithm makes the solution immutable during the optimization. Our goal was to allow the algorithm to improve the solution.

About the freezing strategy mentioned in Section 2, in which we choose parts (architectural elements) from the optimized solutions and mark them as frozen to maintain immutable during the optimization, we also defined a strategy for choosing elements. Figure 5 shows the element class named `Velocity` in the way we searched for freezing. In the evaluated PLA alternatives, we froze this class with four attributes and methods related to the PLA feature movement. We choose to freeze classes like that since it has only one feature, which has good feature modularization from the perspective of the FM objective function.

After the optimization process finished, the objective function values obtained from the optimized solutions were collected (Step 3). We compared the values of these objective functions to check if reducing the features impacted the quality of solutions (Step 4). The comparison was made using the Euclidian Distance (ED) (Bishop, 2006) quality indicator. In our analysis, we utilized the ED to compare the objective function values of different solutions. The ED measures the true straight-line distance between two points in Euclidean space. In the context of our study, it is calculated as

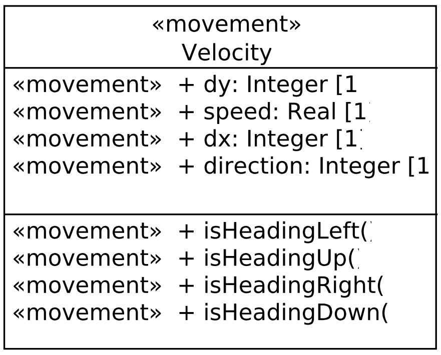

«movement»
Velocity
«movement» + dy: Integer [1
«movement» + speed: Real [1
«movement» + dx: Integer [1
«movement» + direction: Integer [1
«movement» + isHeadingLeft(
«movement» + isHeadingUp(
«movement» + isHeadingRight(
«movement» + isHeadingDown(

Figure 5. Example of class frozen Velocity

the square root of the sum of the squared differences between corresponding objective function values of two solutions. Mathematically, for two solutions $\mathbf{A} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{B} = (b_1, b_2, \ldots, b_n)$, the Euclidean distance $d(\mathbf{A}, \mathbf{B})$ is defined as:

$$d(\mathbf{A}, \mathbf{B}) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \tag{4}$$

This indicator was employed to quantify the difference between solutions in the multi-objective optimization space (Step 5). By measuring the ED, we could objectively assess how similar or different the solutions are based on their performance across multiple objective functions. This approach allowed us to compare and rank solutions, ensuring that our selected features led to optimal solutions.

We used the non-parametric method Kruskal-Wallis (Kruskal and Wallis, 1952) for testing whether samples originate from the same distribution. It compares two or more independent samples of equal or different sample sizes. A significant Kruskal–Wallis test indicates that at least one sample stochastically dominates one other sample. The test does not identify where this stochastic dominance occurs or how many groups of stochastic dominance are obtained, but it checks if the results obtained by using fewer features were better than those obtained by using all the features.

Finally, in Step 6, we opened each solution and analyzed the organization of the elements, linking the observations we made with the features used. The objective here was to understand the impact of the selected features in the resulting solutions.

4 Results and Discussion

This section presents and discusses the results obtained in the feature selection experiment carried out in this work. All the data produced during the experiment, including the results of the application of the feature selectors and the data resulting from the tests in the subsets built, are available in the complementary material (Arasaki et al., 2023).

4.1 Obtaining Initial Scores

Considering the purpose of Step 1 (see Section 3.5), we initially executed a trial with all features and all selectors with the K* classification algorithm to obtain a performance baseline for comparison in further tests.

After that, for all DM profiles, we evaluated each feature with each selector using Weka. These evaluations were performed for the eleven different DM profiles (as defined in the Dataset described in Section 3.4). We also calculated the averages for each feature selector value across all DMs. From this initial group of results, we observed that:

- *HasFreezing* was identified as irrelevant to the ML model for all DMs, as shown in the last row in Table 1, where the feature received a score of zero for all selectors. Thus, we did not consider *HasFreezing* in further tests as it had no impact on the ML model performance.
- The CFS-BestFirst selector consistently returned zero for all DMs for features *COE*, *NumInterfaces*, *ACLASSAppliedTE*, *COEAppliedTE* and *HasFreezing*, indicating that these features may be excluded according to its selection criteria, as shown in the first column of Table 1, where these features received an average of zero for CFS-BestFirst. These features were stored for use in future tests.
- The *ElementId* feature was also consistently selected by CFS-BestFirst through all DMs, which is to be expected as it uniquely identifies the element in the architecture. One of the trials described below, which removed *ElementId* from the feature set, had noticeably worse performance than the baseline results.

Table 1 presents the initial averages for each (selector, feature) pair across all DMs; the first column indicates the evaluated feature's name, and subsequent columns indicate the averages for each selector.

Table 1. Feature selector results

Feature	CFS-BestFirst	Correlation	GainRatio	InfoGain
ElementId	1	0.097092	0.085219	0.226587
NumMethods	0.545454	0.070508	0.058436	0.111993
NumAttributes	0.454545	0.127445	0.066700	0.065458
FM	0.363636	0.046475	0.010260	0.011960
ACLASS	0.272727	0.052192	0.010455	0.012001
COE	0	0.050411	0.005681	0.004666
ElementType	0.0909090	0.153760	0.043705	0.050113
NumClasses	0.181818	0.071195	0.053878	0.027310
FMAppliedTE	0.727272	0.951155	0.068165	0.111882
COEAppliedTE	0	0.097227	0.045167	0.025768
ACLASSAppliedTE	0	0.074368	0.049038	0.032523
NumInterfaces	0	0.085672	0.039382	0.020383
HasFreezing	0	0	0	0

4.2 Analysis of Scored Features

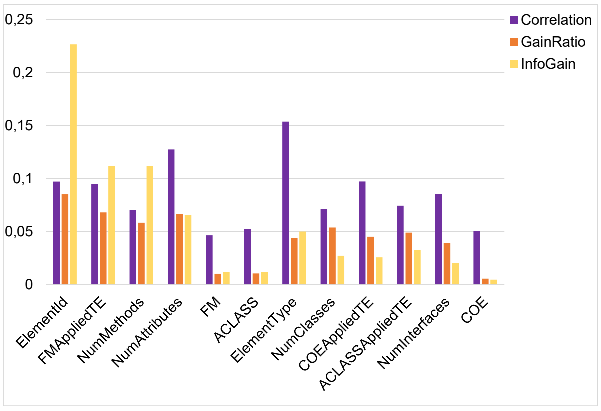
After computing all selector averages, we calculated the metric AV for each feature. Then, the best and worst features were ranked according to these scores, which guided the definition of the next set of tests.

As an example, we observed that *COE* achieved the following average scores by selector: 0 for CFS-BestFirst (never being selected), 0.0504 for Correlation, 0.0057 for GainRatio and 0.0047 for InfoGain. The average of those scores provides an AV of approximately 0.0152. *ElementId* was the most relevant feature to the ML model, achieving an AV of approximately 0.352.

Table 2. AV metric for all features in the dataset

Feature	AV
ElementId	0.3522246818
FMAppliedTE	0.2506091136
NumMethods	0.19659825
NumAttributes	0.1785376136
FM	0.1080834773
ACLASS	0.08684429545
ElementType	0.08462220455
NumClasses	0.08355061364
COEAppliedTE	0.04204081818
ACLASSAppliedTE	0.03898245455
NumInterfaces	0.03635972727
COE	0.01518990909
HasFreezing	0

Table 2 presents the features sorted according to their relevance based on the AV values. Besides calculating AV, we constructed graphs considering each (selector, feature) pair to observe any special effects or trends. As we have two types of selectors, numerical and binary, we separated each selector type into different graphs so we could analyze them separately. These graphs are shown in Figures 6 and 7.



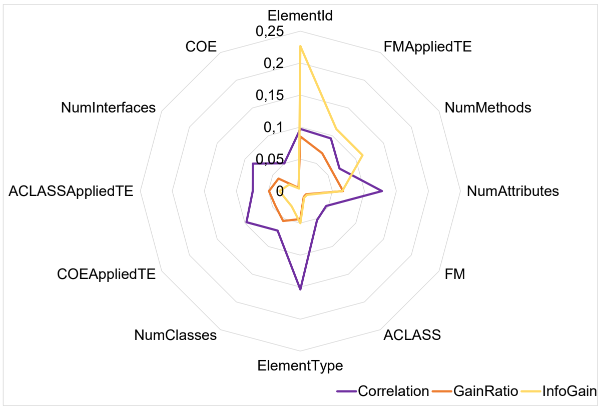


Figure 6. Numeric Selectors

The numerical selectors graph (Figure 6) shows that the features *ElementId* (InfoGain Avg. = 0.2266), *ElementType*

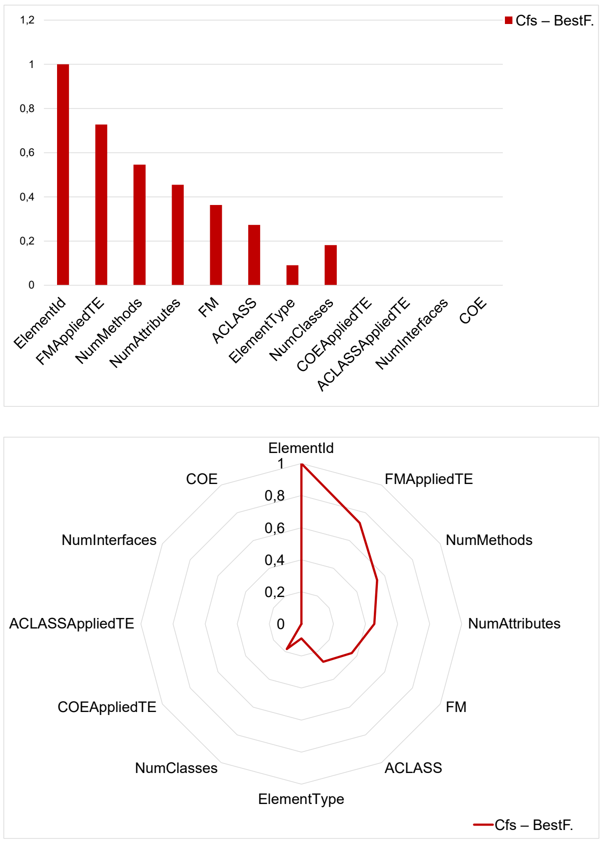


Figure 7. Binary Selector

(Correlation Avg. = 0.1538), *NumAttributes* (Correlation Avg. = 0.1275) and *FMAppliedTE* (InfoGain Avg. = 0.1119 and Correlation Avg. = 0.0951) performed better in certain selectors. This occurrence indicates a possible relevance of these features and they were considered in the assembly of the possible subsets with relevant features to the ML model.

When observing the binary selector CFS-BestFirst graph (Figure 7), we noticed that the features *ElementId* (Average = 1.0000), *FMAppliedTE* (Average = 0.7273), *NumMethods* (Average = 0.5455) and *NumAttributes* (Average = 0.4545) performed better than other features. This occurrence also indicates a possible relevance of these features and was similarly considered in the assembly of the subsets.

4.3 Building Feature Subsets for Evaluation

This section has assembled some possible subsets with relevant features to the ML model. All these subsets were tested by running the K* classification algorithm, and the performance of each subset was measured using the metrics Precision, Accuracy, Recall, and F1-Score.

Based on the results presented in the previous sections, we built an initial subset with the most relevant features named A. This subset was composed of the following features: *ElementType*, as it was well-positioned in the numerical feature selectors; *NumMethods* and *ElementId*, as both ranked were well in binary feature selector; and lastly *FMAppliedTE* and *NumAttributes*, which were highlighted in both analyses.

Considering the original subset, all features except those presented in A were sorted by their AV score, as: *FM* (0.1081), *ACLASS* (0.0868), *NumClasses* (0.0835), *COEAppliedTE* (0.0420), *ACLASSAppliedTE* (0.0390), *NumInterfaces* (0.0364) and *COE* (0.0152).

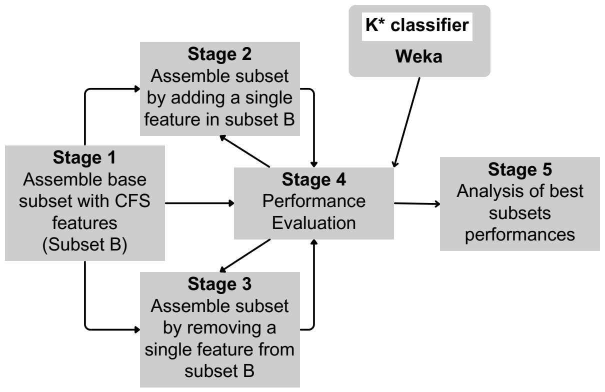
Then, an ensemble of subsets was incrementally prepared, using these features, thereby: $A_1 = A \cup \{FM\}$, $A_2 = A_1 \cup \{ACLASS\}$, and so forth, until subset $A_7 = A_6 \cup \{COE\}$ (which includes all features except *HasFreezing*). These subsets and their respective values for Precision, Accuracy, Recall, and F1-Score are presented in Table 3.

Table 3. AV-based subset percentage results

Subset	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)
A	-2,760	-1.872	-1.295	-1.606
A_1	-1.197	-0.810	-0.541	-0.677
A_2	-0.289	-0.36	-0.063	-0.150
A_3	-0.289	-0.236	-0.063	-0.150
A_4	-0.289	-0.236	-0.063	-0.150
A_5	-0.290	-0.236	-0.063	-0.150
A_6	-0.288	-0.235	-0.063	-0.149
A_7	0	0	0	0

Analyzing the performance of the subsets built (Table 3), A_7 attained the best performance presenting metrics of Precision, Accuracy, Recall, and F1-Score similar to the original subset. However, as the subset A_7 is composed of all features, except *HasFreezing*, which was classified as irrelevant in all feature selector algorithms, we decided to try another approach aiming to reduce the number of features without harming the model classification performance.

Figure 8 presents the approach to exploring the intercorrelation factor between the features considered in the CFS-BestFirst selector. It is worth mentioning that from this point on, we used the CFS-BestFirst results as the basis for further analysis since this selector presented the most relevant features to the OPLA-ALM. Table 1 shows that the other selectors highlighted the same features as relevant to the ML model. Nevertheless, they considered other features important to the model, and there was a divergence between which features should be selected. Thus, they did not ascertain which features are the most important to OPLA-ALM as CFS-BestFirst did.



K* classifier
Weka
Stage 2
Assemble subset by adding a single feature in subset B
Stage 1
Assemble base subset with CFS features (Subset B)
Stage 4
Performance Evaluation
Stage 5
Analysis of best subsets performances
Stage 3
Assemble subset by removing a single feature from subset B

Figure 8. Assembly of subsets by CFS-Best First Selector

Considering the stages presented in Figure 8, the process which guided the construction of the subsets evaluated was: in **Stage 1** we created the subset B containing features that received a score different from zero from the CFS-BestFirst selector, {*ElementId, NumMethods, NumAttributes, FM, ACLASS, ElementType, NumClasses, FMAppliedTE*} (Table 1). In **Stage 2**, we prepared more subsets by adding to B the non-selected features one by one. **Stage 3** was similar to **Stage 2**, but instead of adding features, we removed features one by one. In **Stage 4**, we computed the Accuracy, Precision, Recall, and F1-Score of the subsets built-in **Stages 2 and 3**. Finally, in **Stage 5**, we analyzed the performance of all subsets resulting from **Stage 4**, aiming to define the best subset for the OPLA-ALM model.

The subsets tested in Stage 2 (see Figure 8) were $B_a = B \cup \{COEAppliedTE\}$, $B_b = B \cup \{ACLASSAppliedTE\}$, and so forth, up to $B_d = B \cup \{COE\}$. The results from these subsets are displayed in Table 4. This table presents the assembled subsets and their respective performance for each metric collected from the OPLA-ALM.

Table 4. Subset percentage results from CFS-BestFirst analysis, adding features

Subset	Accuracy(%)	Precision(%)	Recall(%)	F1-Score(%)
B	-0.289	-0.236	-0.063	-0.150
B_a	-0.289	-0.236	-0.063	-0.150
B_b	-0.289	-0.236	-0.063	-0.150
B_c	-0.281	-0.222	-0.068	-0.145
B_d	0	0	0	0

By using the subset B and its combinations with features not selected previously by CFS-BestFirst, we observed small differences in Accuracy (-0.289%), Precision (-0.236%), Recall (-0.063%) and F-Score (-0.150%) when comparing with the subset that contains all the features. The results were similar to the test executed with all features except *COE*. We also observed that B_d (the subset containing all selected features plus *COE*) was that which presented the best overall result among all conducted tests, with values of Accuracy, Precision, and Recall metrics similar to the test using all the features.

In Stage 3 (Figure 8), we carried out trials containing all features selected by CFS-BestFirst but excluding one of them per assembled subset. These tests did not obtain better results than those described above, but we noticed that removing *ElementId* reduced the performance of all metrics since it is the most relevant feature in OPLA-ALM. The *ElementId* feature uniquely identifies the element in the architecture, and removing it results in a great loss of information for the OPLA-ALM.

The subsets we prepared by removing attributes from B were $C_a = B - \{ElementId\}$, $C_b = B - \{FMAppliedToElement\}$, $C_c = B - \{NumMethods\}$, and so forth, up until subset C_h. The results from these tests are presented in Table 5.

Table 5. Subset percentage results from CFS-BestFirst analysis, removing features

Subset	Acc(%)	Prec(%)	Rec(%)	F1(%)
C_a	-2.818	-2.667	-0.714	-1.704
C_b	-0.289	-0.236	-0.063	-0.150
C_c	-0.289	-0.236	-0.063	-0.150
C_d	-0.290	-0.237	-0.063	-0.150
C_e	-1.311	-1.141	-0.359	-0.754
C_f	-1.197	-0.810	-0.541	-0.677
C_g	-0.289	-0.236	-0.063	-0.150
C_h	-0.289	-0.236	-0.063	-0.150

After many assembled subsets with the most varied combinations of features, we found that the configuration comprising the features selected by CFS-BestFirst (Table 1) plus *COE* (subset B_d) was the one which best performed so far. We obtained results that were roughly equal in performance to the subset that contains all the features.

Since we aimed to minimize the number of features without harming the performance of OPLA-ALM, we concluded that it is not worth considering the other subsets, as they have more features that worsened the overall execution time in comparison with this subset, {*ElementId, NumMethods, NumAttributes, FM, ACLASS, ElementType, NumClasses, FMAppliedTE, COE*}.

4.4 Time Analysis

This section presents a performance analysis based on the time spent training and testing the model using the features from the original set and the best subsets (subsets B_d and B_c). This analysis focuses on verifying the impact of removing the four features on the time necessary for training and testing the model. The time values presented in this section were measured by the Elapsed Time and User CPU time millis metrics, abbreviated to ETT and UTMT, respectively.

Firstly, we calculated and collected ETT and UTMT metrics values on subsets using Weka tool. We measured the training and testing times spent on the original set, the subset B_d (the best found), and the subset B_c (second best subset evaluated). From this point, we will refer to the model trained using the original set as Model A. Similarly, the models using B_d and B_c subsets will be referred to as Model B and C, respectively.

As presented in Figure 9, there is no significant difference in the time for training the models A, B, and C. We observed that the times spent on training them were similar because we used the same dataset but with a different number of features. Another observation is that the longest time spent training the model, even when using all the features (Model A), is 31 milliseconds. Thus, despite removing four features, ETT and UTMT metrics cannot show much difference between the models due to the low time spent training (even when using Model A, which contains all the features).

The graph in Figure 10 presents the measured time values. As mentioned in Section 3.4, eleven datasets were used in experiments. Axis-X summarizes the metrics calculated for each dataset obtained from experiments for each DM, and Axis-Y presents the time values. We tagged the datasets with the DM identifications. So, Dataset DM01 contains the data from DM 01, Dataset DM02 from DM 02, and so on.

According to testing times presented in Figure 10, all the models (A, B, and C), when using datasets DM04 and DM08, show a huge difference compared to the other datasets. The models using dataset DM04 have the best testing times, considering the values of 0.02 and 22.81 in ETT and UTMT, respectively. The models using dataset DM08 presented the worst testing time, with 1.18 and 1,035.47 in time ETT and UTMT, respectively. This gap is due to the size of the two datasets, as the dataset DM08 is the largest file and is approximately seven times larger than the dataset DM04. As mentioned in Section 3, the dataset size is the number of lines in

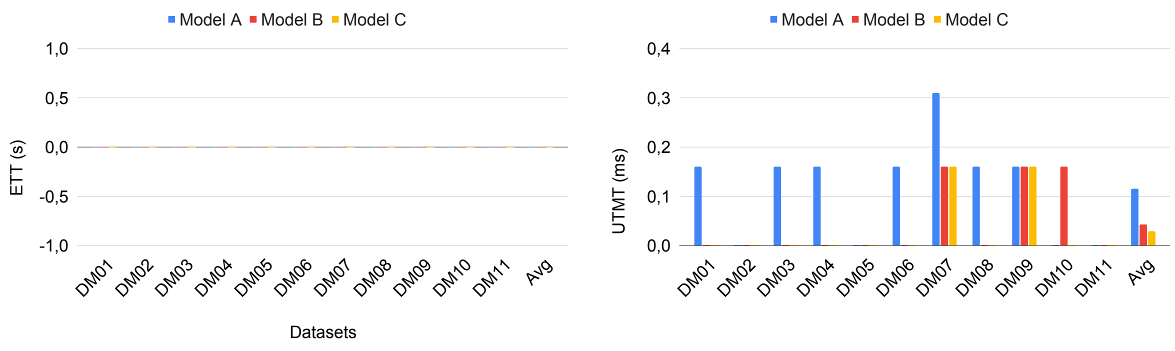


Figure 9. Training Times

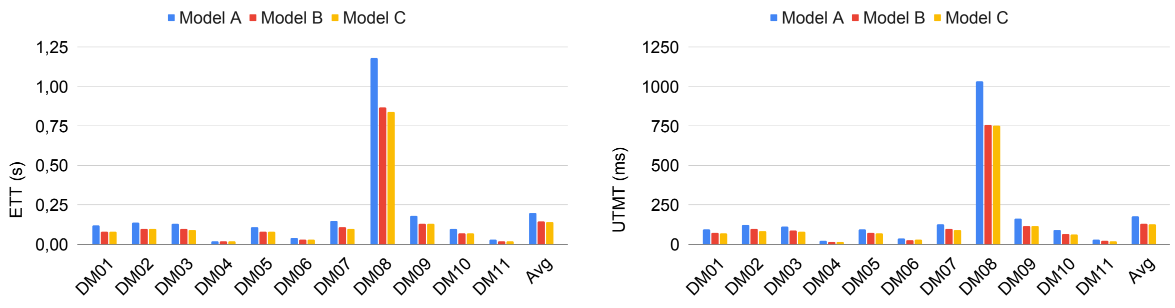


Figure 10. Testing Times

the file used for training the models. Each line contains features such as *ElementId* and *ElementType* from parts of the optimized solutions evaluated by DMW during the optimization process. Although the dataset size is seven times larger, the testing times are not seven times longer. So, we can conclude that the growth in testing times in relation to the size of datasets is not linear.

To complement the analysis, we built Table 6 that compares Model A's (with all features) and Model B's testing time (with only the best features). In this table, we present the percentage of reduction of testing time of Model B in relation to Model A. Figure 11 presents another view of the values.

Table 6. Time Reduction in Percentage for Model B

Dataset	Testing	
	ETT(%)	UTMT(%)
DM01	33.33	21.71
DM02	28.57	21.52
DM03	23.07	24.59
DM04	0	28.75
DM05	27.27	23.48
DM06	25	27.31
DM07	26.66	22.09
DM08	26.27	26.87
DM09	27.77	29.04
DM10	30	26.63
DM11	33.33	30.37
Average	25.57	25.67

Observing this table, it is important to highlight that the

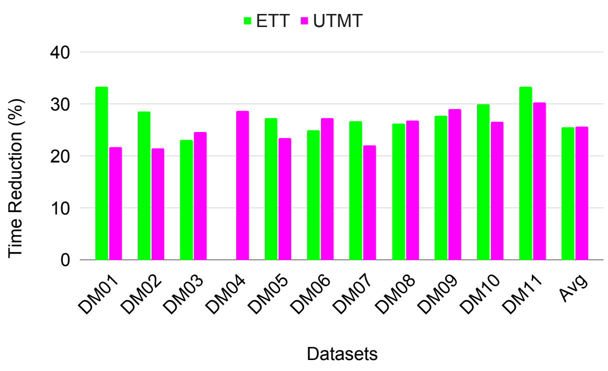


Figure 11. Time Reduction Model B

models using dataset DM04 do not show a reduction in testing time when considering only the ETT metric. Also, ETT obtained an absolutely zero difference between Model B's and Model A's testing time metrics. This occurrence is not a significant indication, as the dataset size is so small that the testing time cannot be reduced.

Unlike the absolute time graphs in Figure 10, the reduction graph of Model B's in relation to Model A's, presented in Figure 11, shows close values between the models using the different datasets, except for DM04, which presents zero difference for the ETT metric. This event indicates that models B and C (with fewer features) needed less time for testing.

It is noted that the greatest difference does not occur in dataset DM08 as expected since it was the one that spent the most time testing. Dataset DM08 presents approximately 26% reduction, but the greatest difference between Model B and Model A, in terms of ETT, is approximately 33%

occurred in dataset DM01 and dataset DM11. Considering the column with the UTMT metric values in Table 6, we obtain the largest reduction value in any dataset that is not dataset DM08, which again presents approximately 26% reduction. The largest reduction occurs in dataset DM11, which achieves a 30%

Table 6 also shows that, despite reaching peaks of up to 33% reduction, the average reduction when considering the eleven datasets that use the features of Model B is 25% for ETT and UTMT. This value supports the validity of the feature removal process carried out that originated the subset B_d, as when we eliminated 30% of the features, we noticed that datasets presented close reductions, resulting in an average reduction of 25% in time metrics.

Similarly, as was done for Model B, Table 7 presents the percentage of reduction of testing time of Model C in relation to Model A, and the graph presented in Figure 12 presents the values from another view.

Table 7. Time Reduction in Percentage for Model C

Dataset	**Testing**	
	ETT(%)	**UTMT(%)**
DM01	33.3	27.22
DM02	28.57	31.28
DM03	30.76	28.28
DM04	0	32.88
DM05	27.27	26.11
DM06	25	25.7
DM07	33.33	29.3
DM08	28.81	27.11
DM09	27.77	28.66
DM10	30	30.47
DM11	33.33	33.33
Average	27.1	29.11

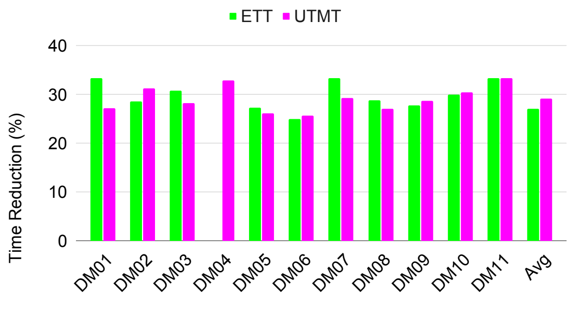


Figure 12. Time Reduction Model C

In this table, we first observed the zero value for ETT in dataset DM04. This happened because the time needed for testing was very short. Once again, dataset DM08 did not present the highest reduction value as expected, returning approximately 28% reduction when considering the ETT metric. The largest reduction presented by Model C in ETT is 33%, occurring in datasets DM01, DM08, and DM12. When just observing at the UTMT, something similar happens. DM08 presents a testing time reduction of 27%, but the largest testing time reduction appears in the dataset DM04 and dataset DM11 with approximately 33%.

The table also shows that, despite reaching peaks of up to 33% reduction, the average reduction when considering the eleven datasets that use Model C is 27% for ETT and 29% for UTMT. These values support the validity of the feature removal process that originated the subset B_c, as when we eliminated 30% of the features, we noticed that datasets presented close reductions resulting in an average decrease of 27% and 29% in time metrics.

To validate the process used in feature selections, performance tests and measurements of the time spent with the Multilayer Perceptron classifier with 500 epochs were carried out. This classifier was chosen because it had already been used in Bindewald et al. (2019) and Bindewld et al. (2020). From this moment on, we will use the acronym MLP to refer to the Multilayer Perceptron classifier.

Before analyzing the times performed with the MLP, the performance metrics of the Bd and Bc subsets [5] performed with the MLP were extracted and compared with the metrics of the original feature set also performed with the MLP. All metrics from both models performed with each subset, Bd and Bc, showed no worsening in performance concerning the original set.

Again, we calculated and collected ETT and UTMT metric values into subsets using the Weka tool. We measured the training and testing times spent on the original set, the B_d subset (the best found), and the B_c subset (the second best-evaluated subset) with the MLP classifier. From then on, we will refer to the model trained using the original set as Model M1. Likewise, models using subsets B_d and B_c will be called Model M2 and M3, respectively.

The graph in Figure 13 presents the measured time values for training times in the same way as the models using K*.

As presented in Figure 14, there is no significant difference in the time for testing the models M1, M2, and M3. Another observation is that the longest time spent testing the model, even when using all the features (Model M1), is 0.47 milliseconds. Thus, despite removing four features, ETT and UTMT metrics cannot show much difference between the models due to the low time spent testing (even when using Model A, which contains all the features).

To complement the analysis, we built Table 8 that compares Model M1's (with all features) and Model M2's testing time (with only the best features). In this table, we present the percentage of reduction of testing time of Model M2 in relation to Model M1.

Observing this table, it is important to highlight that all models present a reduction in training time for both the ETT and UTMT metrics. The average reduction for ETT was approximately 53% and for UTMT it was 51%. These values presented reinforce the feature reduction method proposed and applied in Model M2.

Similarly, as was done for Model M2, Table 9 presents the percentage of reduction of testing time of Model M3 in relation to Model M1.

In this table, we first observe that again all models show a reduction in training time for the ETT and UTMT metrics. The average reduction for ETT was approximately 52% and for UTMT it was 50%. These values presented also reinforce

[5] Bd and Bc subsets were the best ones obtained in previous tests.

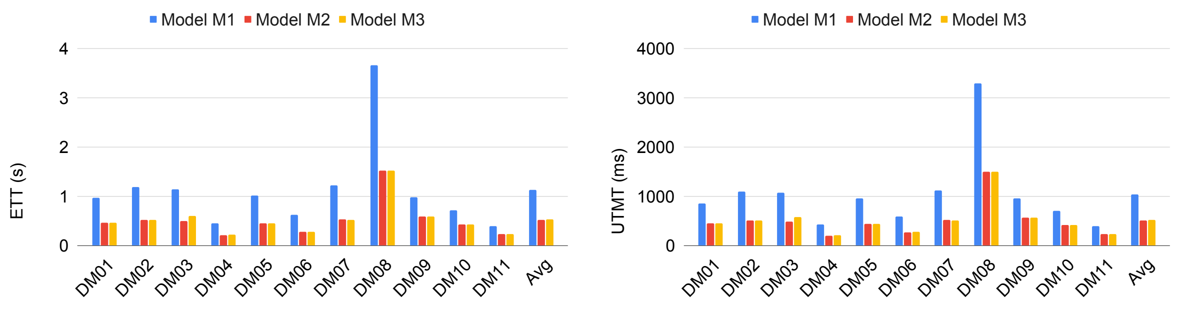


Figure 13. Training Times with MLP

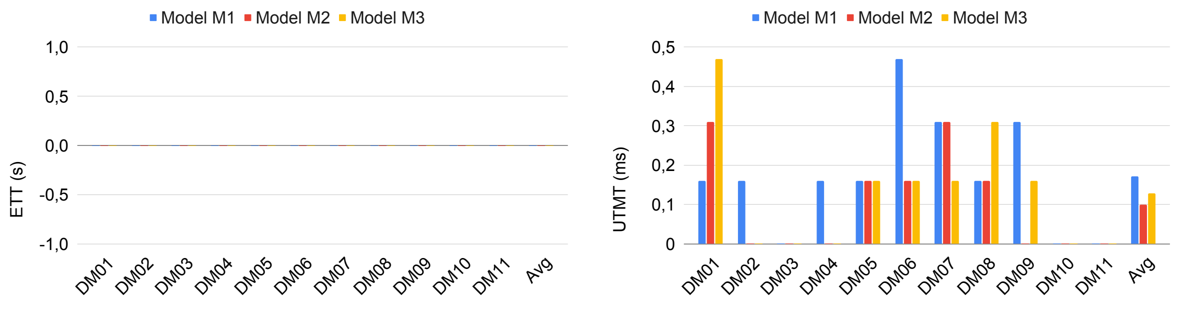


Figure 14. Testing Times with MLP

Table 8. Time Reduction in Percentage for Model M2 with MLP

Dataset	**Training**	
	ETT(%)	**UTMT(%)**
DM01	51.55	46.78
DM02	56.30	53.80
DM03	56.14	54.95
DM04	54.35	53.87
DM05	55.88	54.29
DM06	55.56	53.74
DM07	56.91	54.58
DM08	58.31	54.58
DM09	39.80	40.11
DM10	40.28	40.36
DM11	40.00	39.77
Average	53.67	51.20

Table 9. Time Reduction in Percentage for Model M3 with MLP

Dataset	**Training**	
	ETT(%)	**UTMT(%)**
DM01	51.55	47.20
DM02	56.30	53.59
DM03	47.37	46.45
DM04	52.17	50.64
DM05	54.90	53.55
DM06	55.56	53.55
DM07	57.72	54.34
DM08	58.31	54.55
DM09	39.80	40.27
DM10	40.28	40.32
DM11	40.00	40.09
Average	52.78	50.29

the features reduction method presented in this work applied to Model M3.

4.5 Solutions Analysis

To measure the quality of the solutions obtained at the end of the optimization process, we used a set of objective functions, namely PLA Feature Modularization (FM), Class Coupling (ACLASS), and Cohesion (COE). Previous works have demonstrated the importance of these metrics for maintaining a high-quality PLA design.

We compared the quality of the solutions generated using the model with all features versus the model using only the selected features (30% fewer features). The comparison was made based on the ED between the objective function values of the solutions. The ED measures how different two solutions are in the multi-objective optimization space.

To perform this comparison, we collected the objective function values and calculated the ED of the solutions to the ideal solution. The ideal solution represents a hypothetical point in the multi-objective optimization space where all objective function values are optimal. This means that the ideal solution achieves the best possible value for each objective. Calculating the ED to the ideal solution allows us to quantify how close each solution is to the optimal performance across all objectives, providing a clear metric to assess the effectiveness of the feature selection.

Table 10 presents the best (minimal), median, and average ED for the solutions obtained using all features and the selected features. The results obtained from the feature selection process are not just theoretical improvements; they have practical implications for the PLA design, particularly when applied to a PLA like the AGM.

Table 10. ED for resulting solutions

	Selected Features	All Features
Min	709	667
Median	726	720
Average	724	709

The results show that the solutions generated with the selected features were slightly less optimal than those generated with all features. However, according to the Kruskal-Wallis test, there was no statistical difference ($p - value > 0.05$) between the two sets of solutions.

To complement the metric comparison, we analyzed the organization of the architectural elements in the PLA solutions. One notable observation was that the number of solutions generated at the end of the optimization was affected by the reduction of features. On average, the optimization algorithm with fewer features generated 40% more solutions. This increase in the number of solutions suggests that the algorithm explored a broader solution space with fewer features.

This percentage was calculated by the equation `1 - (3 / 5)`. To obtain these values, the average of the number of solutions generated in the executions was calculated. The average of solutions generated by the algorithm using all features was three solutions. On the other hand, the algorithm using fewer features obtained five solutions on average: two more solutions.

We also examined specific elements in the PLA alternatives. For instance, in all the solutions obtained, the element `Velocity` remained consistent with four attributes and four methods related to the movement feature, regardless of the feature set used. This consistency indicates that the reduced feature set did not compromise the essential characteristics of the solutions.

As mentioned in (Bishop, 2006), overfitting is problematic when using ML algorithms because it leads to models that perform well on training data but poorly on unseen data. Overfitting occurs when a model learns the noise and details in the training data to the extent that it negatively impacts the model's performance on new data. Figure 15 presents another observation we made, which gives some insights into the features' impact on the overfitting of the model.

The first part of this figure shows the pattern of the AnimationLoop elements generated when using all the features. The second part shows the elements obtained using only the selected features. As can be observed, using all the features overfitted the model in separate PLA features since the AnimationLoop elements have only one method related to the pause of the animation. In the second example, the elements involve the pause and the play animation. As can be observed, using all the features overfitted the model in separate PLA features since the AnimationLoop elements have only one method related to the pause of the animation. In the second example, the elements involve the pause and the play animation.

These findings indicate that feature selection effectively reduces the number of features without negatively impacting the quality of the solutions. The selected features prevented overfitting and maintained a high level of solution quality. In future work, we intend to conduct experiments with more DMs on a larger scale and with different PLA designs to validate these results' generalizations and check the solutions' diversity.

4.6 Summary of Findings

While carrying out the experiment described in this section, we identified some promising feature subsets that were smaller than the thirteen features originally used for OPLA-ALM.

Considering the evaluated features, *ElementId*, *FMAppliedTE*, *NumAttributes*, *NumMethods* and *ElementType* are the most relevant features because they have the best overall scores according to the feature selectors. In contrast, *HasFreezing*, *FM*, *NumInterfaces*, *ACLASS*, *COEAppliedTE* and *COE* were considered irrelevant according to the feature selectors, although it is worth noting that *COE* presented the best result in this study when used in conjunction with other features selected by CFS-BestFirst (Table 1).

In this context, and based on the tests performed in this work, our findings indicate that the feature set B_d presented the best performance. This set is composed of the following features: *FM*, *ACLASS*, *ElementId*, *ElementType*, *NumClasses*, *NumAttributes*, *NumMethods*, *FMAppliedTE* and *COE*; reducing the number of features in the original set, from thirteen to nine features (a 30% reduction).

We carried out tests to measure and compare the time spent training and testing the model on the subsets B_d and B_c (the best subsets) in relation to the original set. Although the B_d subset is the best in accuracy, the B_c subset presented a slightly faster average performance. Even slower than subset B_c, subset B_d showed an average time reduction of 25% of the time needed to test the model, indicating that the removal of features really had a positive impact on model performance.

It is also worth mentioning that while we could extract useful information and data from all selectors, the CFS-BestFirst selector should be highlighted as it guides the feature selection process toward the best results.

Table 11 summarizes the final performance results for all tested feature subsets compared to the original that used all attributes. The first column lists the different subsets tested. The subsequent columns provide the percentage changes in Accuracy (Acc), Precision (Prec), Recall (Rec), and F1-Score (F1) relative to the initial trial. Negative values indicate a decrease in performance, while 0 means no change.

The most notable insights from this table are the results for subsets B_d and A_7. Subset B_d shows no change (0%) across all performance metrics, indicating that it maintains the same high performance as the original feature set while reducing the number of features by 30%. This confirms the efficiency of the feature selection process in maintaining model quality with fewer features. On the other hand, subset A_7 also maintains 0% change in all metrics, suggesting it is another strong candidate for an optimized feature set. In contrast, subsets such as A and C_a show significant adverse changes in all metrics, indicating that these feature sets decrease model performance. Highlighting these results helps to emphasize

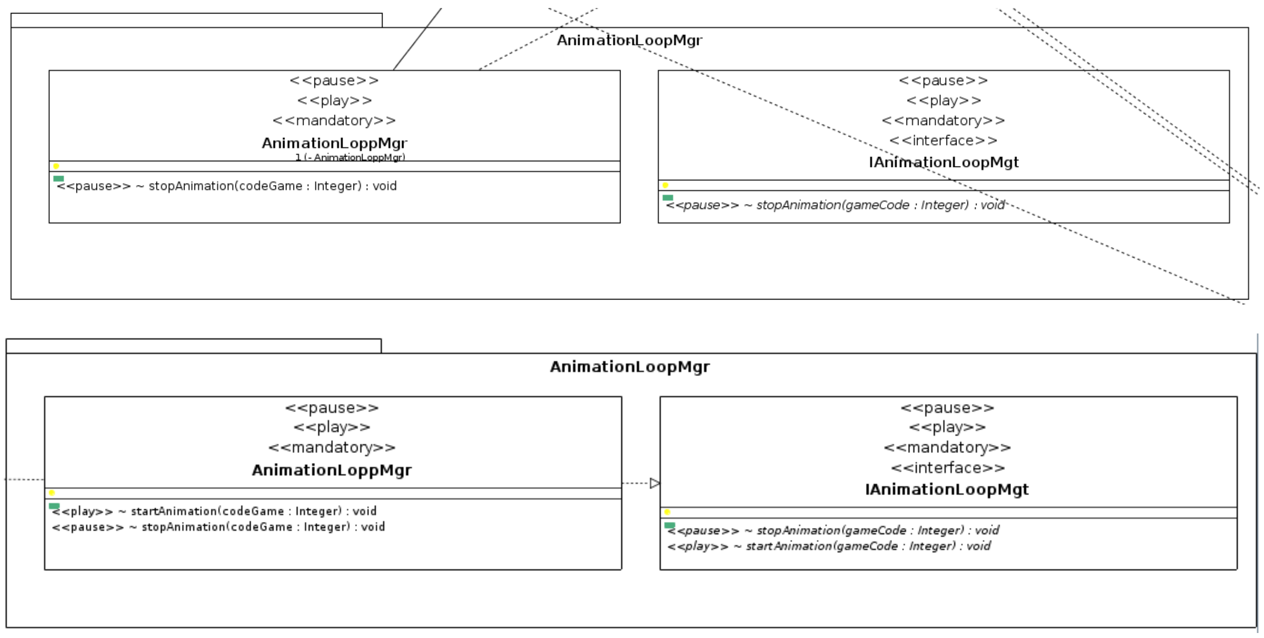

AnimationLoopMgr
<<pause>>
<<play>>
<<mandatory>>
AnimationLoppMgr
<<pause>> ~ stopAnimation(codeGame : Integer) : void
<<interface>>
IAnimationLoopMgt
<<pause>> ~ stopAnimation(gameCode : Integer) : void
<<play>> ~ startAnimation(codeGame : Integer) : void
<<play>> ~ startAnimation(gameCode : Integer) : void

Figure 15. Example of AnimationLoop by using all features vs selected features

the effectiveness of the selected feature subsets and supports the rationale for their use in optimizing the mode.

Table 11. All subset percentage results

Subset	Acc(%)	Prec(%)	Rec(%)	F1(%)
A	-2.760	-1.872	-1.295	-1.606
A_1	-1.197	-0.810	-0.541	-0.677
A_2	-0.289	-0.36	-0.063	-0.150
A_3	-0.289	-0.236	-0.063	-0.150
A_4	-0.289	-0.236	-0.063	-0.150
A_5	-0.290	-0.236	-0.063	-0.150
A_6	-0.288	-0.235	-0.063	-0.149
A_7	0	0	0	0
B	-0.289	-0.236	-0.063	-0.150
B_a	-0.289	-0.236	-0.063	-0.150
B_b	-0.289	-0.236	-0.063	-0.150
B_c	-0.281	-0.222	-0.068	-0.145
B_d	0	0	0	0
C_a	-2.818	-2.667	-0.714	-1.704
C_b	-0.289	-0.236	-0.063	-0.150
C_c	-0.289	-0.236	-0.063	-0.150
C_d	-0.290	-0.237	-0.063	-0.150
C_e	-1.311	-1.141	-0.359	-0.754
C_f	-1.197	-0.810	-0.541	-0.677
C_g	-0.289	-0.236	-0.063	-0.150
C_h	-0.289	-0.236	-0.063	-0.150

Although this study focused on K* classification due to its demonstrated high performance in prior work, we acknowledge the need for broader validation. Future studies will compare our approach with other ML models, including Decision Trees, SVMs, and Neural Networks. Such comparisons will help validate the robustness of our approach and potentially identify scenarios where alternative models may outperform or complement K*.

Reducing DM fatigue is critical for maintaining engagement and efficiency in the iterative PLA design process. Our approach automates interactions and limits the number of evaluations required, significantly lowering the cognitive load on DMs. This improvement is expected to enhance usability and support better decision-making by preventing the adverse effects of fatigue. Future work will include qualitative assessments from DMs to further understand the impact of our approach on user experience.

The observed 25% reduction in computational time directly enhances the practicality of our approach in real-world applications. This time savings is significant in industry settings where software architects often work under tight deadlines. Our approach supports faster decision-making processes by reducing the time needed for model training and evaluation. It improves the overall efficiency of PLA design, making it a valuable tool for software development teams.

The computational gains from feature selection are particularly relevant as datasets grow in size and complexity. By reducing the number of features, we observed a consistent 25% reduction in testing time while maintaining 99% accuracy, ensuring the model's efficiency and scalability for larger SPL scenarios. Furthermore, fewer features streamline the testing phase, which is critical in interactive PLA design processes where rapid interactions are required. Reducing the feature set simplifies the decision-making process for DMs by focusing on the most impactful variables. Additionally, feature selection ensures scalability as the size of SPLs and datasets grows in industrial applications.

Feature selection not only improves computational efficiency but also reduces the cognitive load on decision-makers by minimizing the complexity of their evaluations. By focusing on the most relevant features, the OPLA-ALM reduces the model effort required to assess solutions during the search process.

4.7 Answering RQ

By using the feature selectors, we obtained a subset of nine features (B_d: *FM*, *ACLASS*, *ElementId*, *ElementType*, *Num-*

Classes, *NumAttributes*, *NumMethods*, *FMAppliedTE* and *COE*), that is 30% smaller than the initial subset, which originally contained thirteen features. Additionally, the resulting ML model presented an accuracy of 99%, similar to previous works. Analyzing the time performance, the selected subset showed an average reduction of 25% in testing time, reducing the computational effort. The solution analysis confirmed that the quality of the generated solutions was maintained, with no significant differences in the objective function values compared to using all features. Thus, the answer is positive, and $H1_{RQ}$ was accepted.

4.8 Threats to validity

This section presents the threats to validity related to this work and how they were mitigated.

Internal Validity. We consider that there is space for further improvement in this work, as we only evaluated a few of all possible feature subsets in this study. We tested seventeen possible configurations out of a total of $2^{13} = 8\,192$ subsets[6], considering that the entire feature set had thirteen elements in it. However, we also consider that our results already represent a substantial improvement over the current feature set, in terms of both a lower quantity of features and slightly better performance.

The second threat is the diversity of data for training the OPLA-ALM. We mitigated this threat by using a dataset used in several works (Kuviatkovski et al., 2022) (Freire et al., 2019) (Freire et al., 2020) (Freire et al., 2022), which includes data obtained from DMs with different profiles.

External Validity. The sample size can be a threat to external validity because we used only the AGM PLA in the experiment. Thus, it is not possible to generalize the experiment results for other PLAs. However, the sample allowed the generation of assumptions and conclusions valid to PLAs of other sizes and domains.

The AGM PLA dataset was chosen for its complexity and representativeness and serves as a benchmark for evaluating feature selection in PLAs. Future work will involve applying our approach to different datasets and domains to assess its generalization.

Construct Validity. The definition of the parameter values adopted in the experiment can be considered a threat to validity. This threat was controlled using values adopted in related works (Bindewald et al., 2019; Bindewld et al., 2020; Kuviatkovski et al., 2022), which included the calibration of the ML algorithm for the same context addressed in this work.

Conclusion Validity. We minimize the main threats to the conclusion validity identified during the experiment using feature selectors and metrics commonly used for the validation of the results.

[6]The number of possible subsets was derived from the formula for the total number of subsets from a given set. If a set S has k values in it, then there exist 2^k subsets of S. In combinatorial terms, it can be seen as a product of k different choices with two possibilities each — whether each element is included in the subset or not.

5 Related works

This section presents the works we used to guide us in the feature selection process. Sunita and Aurora (Sunita Beniwal and Arora, 2012) present many ways of classifying and selecting features widely used in data mining. We highlight the fact that the work was not an experimental study, so it did not present ways of evaluating the performance of ML models with selected features.

Sutha and Tamilselvi (K.Sutha and Tamilselvi, 2015) list the advantages and disadvantages of approximately 12 feature selection algorithms with an emphasis on efficiency. Since there are many feature selectors available, this study was important to support us in choosing which selectors to use.

Similarly, Ramaswami and Bhaskaran (M. Ramaswami and Bhaskaran, 2009) compare the performance of six feature selection algorithms to find the best method. They pointed out that the performance of each algorithm can be measured using different classification models. The results claim a reduction in computational cost and construction cost in the training and classification stages of the ML model when using the minimum number of features obtained through the feature selectors.

A feature selection process using four feature selectors is presented in (Gnanambal S et al., 2018). This feature selection is based on assembling subsets, one for each selector, where each subset is composed of the features best ranked by one of the feature selectors. In contrast, the subset assembly presented in the work considers the performance of the features in the four selectors in aggregate.

6 Conclusion

Feature selection is essential in ML since it can improve the model's performance obtained through the training process, reducing training time and computational cost. Also, the feature selection can prevent overfitting and identify smaller sets of features that are the most influential variables, making it easier to interpret and explain the model's decision-making process.

This work aimed to select features for an ML model used in interactive PLA design. This model, named OPLA-ALM, prevents the human fatigue problems caused by excessive interactions with the DM during the search process. In summary, OPLA-ALM learns how DMs evaluate PLAs during interaction in the search process. After achieving the required expertise in evaluating PLAs according to DMs' preferences, it replaces them in this task.

A quantitative experiment was performed using four different feature selectors. The results were promising since we could reduce from thirteen to nine (-30%) features for training and testing the OPLA-ALM model. This reduction also resulted in an average reduction of 25% in testing times performed to validate the model. Furthermore, the accuracy of the OPLA-ALM remained at 99%, indicating a well-defined ML model.

While modest, the 25% reduction in testing time achieved in this study has significant implications for interactive PLA design processes, where time savings accumulate across multiple evaluations. Additionally, as feature selection reduces data dimensionality, it lays a foundation for applying the OPLA-ALM model to larger, more complex datasets in industrial settings, ensuring scalability and broader applicability.

For future work, we intend to extend the application of feature selection to other ML models available in the SBSE context, broadening the scope of the analysis. The subsets of features presented in this work were manually assembled and evaluated. Moving forward, we plan to automate the subset generation process and their evaluation using feature selectors and ML metrics to streamline and enhance the reproducibility of the methodology.

Although the AGM PLA was originally designed for pedagogical purposes, it offers a well-defined and representative structure for evaluating feature selection in SPLs. Its modular nature facilitates testing variability and commonality management techniques. However, to confirm the scalability and generalizability of our findings, future studies will apply our feature selection approach to larger and more complex PLAs. This exploration will ensure that the observed gains in computational efficiency and the reduced feature sets' utility extend to more diverse and industrially relevant scenarios.

7 acknowledgements

The authors acknowledge CAPES (Process 88887.901819/2023-00) and CNPq (Process 428994/2018-0)- for the financial support provided to this research.

References

Amal, B., Kessentini, M., Bechikh, S., Dea, J., and Said, L. B. (2014). On the use of machine learning and search-based software engineering for ill-defined fitness function: a case study on software refactoring. In *International Symposium on Search-based Software Engineering (SSBSE)*, pages 31–45.

Arasaki, C., Wolshick, L., Freire, W. M., and Amaral, A. M. M. M. (2023). Feature selection in an interactive search-based pla design approach. https://zenodo.org/records/12735234.

Bindewald, C. V., Freire, W. M., Amaral, A. M. M. M., and Colanzi, T. E. (2019). Towards the support of user preferences in search-based product line architecture design: an exploratory study. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering (SBES)*, pages 387–396.

Bindewld, C. V., Freitas, W. M., Anaral, A. M. M. M., and Colanzi, T. E. (2020). Supporting user preferences in search-based product line architecture design using machine learning. In *XIV Brazilian Symposium on Software Components, Architectures, and Reuse*. SBC.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Booch, G., Rumbaugh, J., and Jacobson, I. (1998). The unified modeling language user guide. *Addison-Wesley Professional*.

Cleary, J. G. and Trigg, L. E. (1995). K*: An instance-based learner using an entropic distance measure. *Machine Learning*, 21(1-2):61–81.

Colanzi, T. E., Vergilio, S. R., Gimenes, I. M. S., and Oizumi, W. N. (2014). A search-based approach for software product line design. In *18th International Software Product Line Conference*, pages 237–241.

Contieri, A. C., Correia, G. G., Colanzi, T. E., Gimenes, I. M., OliveiraJr, E. A., Ferrari, S., Masiero, P. C., and Garcia, A. F. (2011). Extending UML components to develop software product-line architectures: Lessons learned. In *European Conference on Software Architecture*, pages 130–138.

Ferreira, T. N., Vergilio, S. R., and de Souza, J. T. (2017). Incorporating user preferences in search-based software engineering: A systematic mapping study. *Information and Software Technology*, 90:55–69.

Ferreira FN, Araújo, A. A., Neto, A. D. B., and de Souza, J. T. (2016). Incorporating user preferences in ant colony optimization for the next release problem. *Applied Software Computing*, 49:1283–1296.

Freire, W., Rosa, C., Amaral, A., and Colanzi, T. (2022). Validating an interactive ranking operator for NSGA-II to support the optimization of software engineering problems. In *Proceedings of the XXXVI Brazilian Symposium on Software Engineering (SBES)*, pages 337–346.

Freire, W. M., Bindewald, C. V., Amaral, A. M. M., and Colanzi, T. E. (2019). Supporting decision makers in search-based product line architecture design using clustering. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 139–148.

Freire, W. M., Massago, M., Zavadski, A. C., Amaral, A. M. M. M., and Colanzi, T. E. (2020). OPLA-Tool v2.0: a tool for product line architecture design optimization. In *34th Brazilian Symposium on Software Engineering (SBES)*.

Gnanambal S, M, T., V.T, M., and V, G. (2018). Classification algorithms with attribute selection: an evaluation study using weka. *International Journal of Advanced Networking and Applications*, 09(06):3640–3644.

Hall, M. and M., G. (2022). Bestfirst — weka. https://weka.sourceforge.io/doc.dev/weka /attributeSelection/BestFirst.html. Accessed in April 2023.

Hall, M. A. (1998). *Correlation-based Feature Subset Selection for Machine Learning*. PhD thesis, University of Waikato, Hamilton, New Zealand.

Harman, M. and Jones, B. F. (2001). Search-based software engineering. *Information and Soft. Technology*, 43(14):833–839.

Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., and Peterson, A. S. (1990). Feature-oriented domain analysis (foda) feasibility study. Technical report, DTIC Document.

Kruskal, W. H. and Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260):583–621.

K.Sutha and Tamilselvi, D. J. (2015). A review of feature selection algorithms for data mining techniques. *International Journal on Computer Science and Engineering*, 7(6).

Kuviatkovski, F. H., Freire, W. M., Amaral, A. M., Colanzi, T. E., and Feltrim, V. D. (2022). Evaluating machine learning algorithms in representing decision makers in search-based pla. In *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*, pages 68–75. IEEE.

Likert, R. (1932). A technique for the measurement of attitudes. *Archives of psychology*.

Liu, H. and Motoda, H. (2012). *Feature selection for knowledge discovery and data mining*, volume 454. Springer Science & Business Media.

M. Ramaswami and Bhaskaran, R. (2009). A study on feature selection techniques in educational data mining. *JOURNAL OF COMPUTING*, 1(1).

Mjolsness, E. and Decoste, D. (2001). Machine learning for science: State of the art and future prospects. *Science (New York, N.Y.)*, 293:2051–5.

Mkaouer, M. W., Kessentini, M., Slim, B., and Tauritz, D. R. (2013). Preference-based multi-objective software modelling. In *Proceedings of the 1st International Workshop on Combining Modelling and Search-Based Software Engineering*, pages 61–66.

OliveiraJr, E., Gimenes, I. M. S., and Maldonado, J. C. (2010). Systematic Management of Variability in UML-based Software Product Lines. *Journal of Universal Computer Science*, 16:2374–2393.

Pohl, K., Böckle, G., and van Der Linden, F. J. (2005). *Software product line engineering: foundations, principles and techniques*. Springer.

Quinlan, J. R. (2014). *C4.5: programs for machine learning*. Elsevier.

Ramirez, A., Romero, J. R., and Simons, C. (2018). A systematic review of interaction in search-based software engineering. *IEEE Transactions on Software Engineering*.

Rosa, C. T., Freire, W. M., Amaral, A. M. M. M., and Colanzi, T. E. (2022). Towards an interactive ranking operator for NSGA-II. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO)*, pages 794–797.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. In *Nature*, volume 323, pages 533–536. Springer.

Russell, S., Norvig, P., and Davis, E. (2016). *Artificial Intelligence: A Modern Approach*. Pearson.

SEI (2009). Software Engineering Institute - the Arcade Game Maker pedagogical product line. https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=485941. Accessed in 2018 August.

Simons, C., Singer, J., and White, D. R. (2015). Search-based refactoring: Metrics are not enough. In Barros, M. and Labiche, Y., editors, *Search-Based Software Engineering*, pages 47–61.

Sunita Beniwal and Arora, J. (2012). Classification and feature selection techniques in data mining. *International Journal of Engineering Research & Technology*, 1(6).

Verdecia, Y. D., Colanzi, T. E., Vergilio, S. R., and Santos, M. C. (2017). An enhanced evaluation model for search-based product line architecture design. In *20th Iberoamerican Conference on Software Engineering (CIbSE)*, pages 155–168, San Jose, Costa Rica. CIbSE.

Vikhar, P. A. (2016). Evolutionary algorithms: A critical review and its future prospects. *International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, pages 261–265.

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, 2nd edition.