Ballarin, M.; Marcén, AC.; Pelechano Ferragud, V.; Cetina, C. (2021). On the influence of model fragment properties on a machine learning-based approach for feature location. Information and Software Technology. 129:1-19. https://doi.org/10.1016/j.infsof.2020.106430

On the influence of Model Fragment Properties on a Machine Learning-based approach for Feature Location

Manuel Ballarín[a], Ana C. Marcén[a,b], Vicente Pelechano[b], Carlos Cetina[a]

[a] *SVIT Research Group, Universidad San Jorge, Autovía A-23 Zaragoza-Huesca Km.299 50830, Zaragoza, Spain*

[b] *Centro de Investigación en Métodos de Producción de Software, Universitat Politècnica de València Camino de Vera, s/n 46022, Valencia, Spain*

Abstract

Context: Leveraging machine learning techniques to address feature location on models has been gaining attention. Machine learning techniques empower software product companies to take advantage of the knowledge and the experience to improve the performance of the feature location process. Most of the machine learning-based works for feature location on models report the machine learning techniques and the tuning parameters in detail. However, these works focus on the size and the distribution of the data sets, neglecting the properties of their contents.

Objective: In this paper, we analyze the influence of three model fragment properties (density, multiplicity, and dispersion) on a machine learning-based approach for feature location.

Method: The analysis of these properties is based on an industrial case provided by CAF, a worldwide provider of railway solutions. The test cases were evaluated through a machine learning technique that uses different subsets of a knowledge base to learn how to locate unknown features.

Results: Results show that the density and dispersion properties have a direct impact on the results. In our case study, the model fragments with extra-small density values achieve results with up to 43% more precision, 41% more recall, 42% more F-measure, and 0.53 more Matthews Correlation Coefficient (MCC) than the model fragments with other density values. On the other hand, the model fragments with extra-small and small dispersion values achieve results with up to 53% more precision, 52% more recall, 52% more F-measure, and 0.57 more MCC than the model fragments with other dispersion values.

Conclusions: The analysis of the results shows that both density and dispersion properties significantly influence the results. These results can serve not only to improve the reports by means of the model fragment properties, but also to be able to compare machine learning-based feature location approaches fairly improving the feature location results.

Email addresses: mballarin@usj.es (Manuel Ballarín), acmarcen@usj.es (Ana C. Marcén), pele@dsic.upv.es (Vicente Pelechano), ccetina@usj.es (Carlos Cetina)

Keywords:
Model Fragment Location, Feature Location, Machine Learning, Learning to Rank

1. Introduction

Feature location is the task of finding the features that implement a specific, user-observable functionality in a software system [1]. It plays a key role in the initiation of Software Product Lines when products already exist. In the case of feature location in models, the goal is to identify the model fragment that best realizes a specific feature. Therefore, a model fragment contains the elements of the product model that make the software functionality described in a feature possible.

The emerging interest in leveraging machine learning to address the challenge of feature location has been growing rapidly [2, 3, 4, 5, 6, 7]. In these works, machine learning techniques take advantage of the knowledge and the experience that have been generated in software companies for years in order to perform the feature location process. In feature location on models, the model fragments that have been manually retrieved by the engineers and modellers for years are collected in knowledge bases. Then, all or some of the retrieved model fragments can be exploited by machine learning techniques in order to learn how to locate unknown features.

Figure 1 shows an example of a retrieved model fragment, which can be found in a knowledge base for feature location on models. The retrieved model fragment is stored in the knowledge base with a feature description and a score, which is called a triplet of the knowledge base. The feature describes (using natural language) the functionality searched for. The model fragment contains the model elements retrieved for the feature description. In the example, the model fragment, which is highlighted in gray, consists of a set of three model elements (*Pantograph 2*, *Circuit Breaker 2*, and the relation between them). These three elements belong to the product model, which has more elements that are not included in the retrieved model fragment, such as *Pantograph 1* or *Converter 1*. The score determines the similarity between the feature description and the model fragment. In this example, a score equal to 4/4 means that the retrieved model fragment contains all of the model elements associated to the described feature. In contrast, a score equal to 0/4 means that none of the elements of the retrieved model fragments are related to the described feature.

The whole or part of a knowledge base is usually exploited in machine learning in order to learn how to classify. Specifically, the sets of data, which are selected from knowledge bases for learning, are usually called training datasets. In feature location on models, where the knowledge bases are composed of triplets, a training dataset contains the triplets of the knowledge base, which are used to learn how to locate unknown features. To report machine learning-based works for feature location on models, researchers describe the machine learning techniques, the tuning parameters, and the training datasets. The description

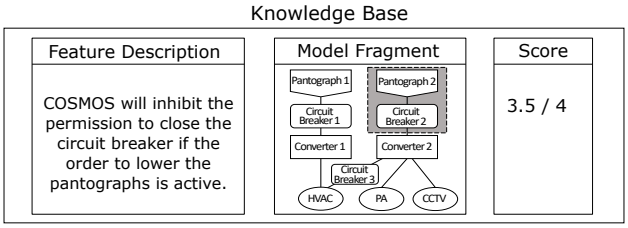


Figure 1: Example of a triplet of the knowledge base for feature location on models.

of training datasets is based on their number of triplets (size) and the distribution of the triplets according to their scores (score distribution). However, the specific properties of the model fragments are neglected even though the model fragments form part of the triplets in the training datasets as the scores do.

In this paper, we analyze the influence of three model fragment properties [8]: density, multiplicity, and dispersion. Specifically, the influence of these properties is obtained taking into account the results of a machine learning-based approach for feature location (FLiM-ML). Among the machine learning techniques supported by this approach, Rankboost is notable for its good results in other fragment location problems [9]. For this reason, Rankboost was the machine learning technique selected for the approach to evaluate an industrial case study. The industrial case study was provided by CAF[1], a worldwide provider of railway solutions, and contains 268 test cases and 7,500 triplets in the knowledge base. The test cases were evaluated through the FLiM-ML approach using different training datasets. Specifically, the subsets of the knowledge base, which were used as training datasets, were selected taking into account the different density, multiplicity, or dispersion values of the model fragments in knowledge base.

Our results show that the multiplicity property does not have an impact on the feature location results. However, the density and dispersion properties have a direct impact on the feature location results. With regard to density, the model fragments with extra-small density values (between 0% and 25%) achieve results with up to 43% more precision, 41% more recall, 42% more F-measure, and 0.53 more MCC than the model fragments with other density values. In fact, the improvement of the results in our evaluation is progressive. The lower the density values, the better the results are.

With regard to dispersion, the model fragments with extra-small (between 0 and 0.25) and small (between 0.25 and 0.50) dispersion values achieve results with up to 53% more precision, 52% more recall, 52% more F-measure, and 0.57 more MCC than the model fragments with other dispersion values. Although the relation between the results is not as clear as in the case of the density property, there may be a relation between the connected elements in a model fragment and the facility to locate the model fragment. In our case study, it seems that the

[1] `www.caf.net/en`

stronger the connection between the elements of a model fragment, the easier it is to find that model fragment. However, future research about dispersion can help to clarify this relation.

The statistical analysis of the results shows that both density and dispersion properties significantly influence the results. Taking into account these results, reporting the model fragment properties may be relevant to enable the replication of research works, that locate features applying a machine learning technique. In fact, to replicate the best results of this work, we should report density and dispersion properties. Moreover, these results also provide evidence that model fragment properties may be relevant to be able to compare machine learning-based approaches fairly and improve their feature location results.

On the one hand, if we wanted to compare two machine learning-based approaches for feature location in models fairly, we should take into account that both approaches may need model fragments with different property values. For example, a deep learning technique usually needs a training dataset with a greater number of triplets than a learning to rank algorithm, although both are machine learning techniques. Likewise, an approach based on deep learning may need model fragments with different density values than an approach based on a learning to rank algorithm. Therefore, in order to obtain the best performance of the compared approaches, we must select the best training dataset for each approach. For example, in our work, the best results were obtained using a training dataset, where the triplets contain model fragments with density values between 0% and 25%. These results were achieved by a learning to rank algorithm (i.e., Rankboost), but other machine learning techniques could require a different training dataset to obtain the best results.

On the other hand, the evaluation results show that by taking into account specific values for one property, we can improve the feature location results. For example, taking into account only extra-small density values, we can surpass the results obtained using other density values. Likewise, taking into account extra-small and small dispersion values, we can surpass the results obtained using other dispersion values. Therefore, we can improve the feature location results of our case study using a training dataset that contains model fragments with extra-small density or extra-small and small dispersion. In summary, in machine learning-based feature location on models, the results of this work can serve not only to improve the reports by means of the model fragment properties, but also can serve to compare machine learning-based feature location approaches fairly and, naturally, to improve the feature location results.

The remainder of this paper is structured as follows: Section 2 provides background on our case study. Section 3 highlights the properties of training datasets, including the model fragment properties. Section 4 details the means used to evaluate our work, the results of the evaluation, and the statistical significance of the obtained results. Section 5 discusses the performed experiment and the obtained results. Section 6 describes the threats to the validity of our work. Section 7 introduces the existing works that are related to our work. Finally, Section 8 concludes the paper.

2. Background

This section presents the Train Control and Management Language (TCML), which is used to formalize the products manufactured by our industrial partner CAF. Their trains can be found all over the world and in different forms (regular trains, subway, light rail, monorail, etc.). We held monthly meetings with their domain experts, who are software engineers with at least 10 years of experience in software development for train control. In this work, the TCML will be used throughout the rest of the paper to present a running example and to describe the case study for the evaluation. Moreover, in this section, we also present an illustration of an equipment-focused simplified subset of the Domain Specific Language (DSL). We also present the Common Variability Language (CVL) [10], which is the language used to formalize the model fragments. Finally, we present the feature location problem for our industrial partner CAF.

The Train Control and Management Language (TCML) has the expressiveness required to describe both the interaction between the main pieces of equipment installed in a train unit and the non-functional aspects related to regulation. A train unit is furnished with multiple pieces of equipment. Some examples are the traction equipment, the compressors that feed the brakes, the pantograph that harvests power from the overhead wires, and the circuit breaker that isolates or connects the electrical circuits of the train. The control software of the train unit is in charge of making all of the equipment cooperate in order to achieve the train functionality, while guaranteeing compliance with the specific regulations of each country.

For the sake of understandability and legibility, and due to intellectual property rights concerns, we present an equipment-focused simplified subset of the DSL. The top left of Figure 2 depicts one example, taken from a real-world train. It presents a converter assistance scenario where two separate pantographs (High Voltage Equipment) collect energy from the overhead wires and send it to their respective circuit breakers (Contactors), which in turn send it to their independent Voltage Converters. The converters then power their assigned Consumer Equipment: the HVAC on the left (the train's air conditioning system), and the PA (public address system) and CCTV (television system) on the right.

To formalize the model fragments used in the rest of our work, we use the Common Variability Language (CVL) [10], which expresses variability among models in terms of Model Fragments such as Placement Fragments (variation points) and Replacement Fragments (variants). The materialization of product models is performed by means of Fragment Substitutions between a Base Model (Placements) and a Model Library (Replacements). In the top right of Figure 2, the element highlighted in gray conform an example of model fragment, including one circuit breaker that connects Converter 2 to Consumer Equipment assigned to Converter 1. This model fragment is the realization of the "converter assistance" feature, which allows the passing of current from one converter to equipment assigned to its peer for coverage in case of overload or failure of the first converter. A simple example of model fragment manipulation in Train Control and Management Language (TCML) can be found at:

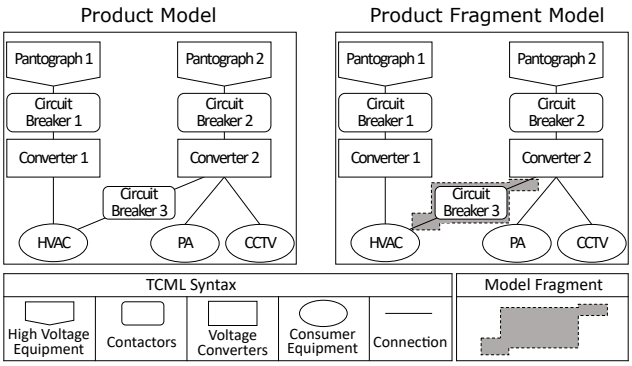


Figure 2: Example of the TCML model, model fragment, and triplet of the knowledge base.

youtube.com/watch?v=Ypcl2evEQB8

2.1. Feature Location

For feature location, the engineers in the CAF company manually search for the model fragment that best realizes a feature in a product model. To do this, the company provides feature descriptions, and the engineers have to locate the model elements (fragment) that are described in each feature.

Although the example of Figure 2 makes the manual retrieval process seem easy, it is important to remember that the figure just shows an example and the real conditions differ from that. Basically, product models are larger and more complex than the product model in the figure. Moreover, the models are created and maintained over long periods of time by different software engineers, and the engineers in charge of feature location often lack knowledge of the entirety of the product details. Under these conditions, feature location consumes great amounts of time and effort, without guaranteeing good results.

In addition to the engineers' experience, the dimensions of the data and the collaboration of several engineers are also relevant factors for time and effort costs. Suppose we ask a group of 19 domain experts to manually retrieve the model fragments that correspond to the 121 feature provided by our industrial partner. Taking into account that the data comprises a family of product models with 23 models of 1200 model elements, at least 27,600 model elements should be evaluated. Moreover, since each model element has about 15 properties, about 414,000 properties should be considered. Assuming that a domain expert only needs one second to consider a property of a model element, the domain expert would need 4.79 days to manually locate each query. For 121 features and 19 domain experts, it would take 30.17 years to obtain the results [11].

These numbers demonstrate that an approach that automatically retrieves model fragments is greatly needed [12]. Moreover, the quality of the retrieved model fragment depends on the experience of the engineer, the complexity of the description, the complexity of the product model, and the time needed. Therefore, the model fragments are stored with the feature description and a

score value. Sometimes, the manual searches can be unsuccessful or incomplete. However, a domain expert assigns scores to indicate how good the model fragments are. Then, the model fragment with the score and the correspondent feature description are stored.

3. Properties of the training datasets

In machine learning, training datasets contain the information used to learn how to classify. The success of a machine learning technique depends largely on the information available. If the information in a training dataset is not enough, the technique cannot properly learn how to perform the classification. Therefore, the completeness and heterogeneity of a training dataset can either facilitate or limit learning. For this reason, both the size and the score distribution of training datasets have been widely discussed in the literature:

- **Size** is related to the amount of information in a training dataset and can greatly impact classification accuracy. As a consequence, the analysis of the size has been a major point of interest in research, generally finding a positive relationship between the size of the dataset and the classification accuracy for a wide range of classifiers [13, 14, 15]. In feature location on models, the size is the number of triplets in the training dataset.

- **Score distribution** is related to amount of information for each different value. It is commonly called class distribution, but this name could be confused in the context of this work. In the model domain, the term "class" is related to class diagrams. In contrast, in machine learning, class refers to the different values that can be predicted for a machine learning technique. To avoid misunderstandings, in this article, the concept of class is replaced by score. In feature location on models, the score distribution is the percentage of triplets in the training dataset for each score range.

  The score distribution is balanced when there is a similar number of triplets in the training dataset for each score range. It is conventional wisdom that classifiers tend to perform worse when the training dataset is imbalanced. A classifier trained from an imbalanced training set generally predicts with greater frequency the scores that are most often repeated in the dataset. This classification behaviour is unacceptable when the minority scores in the dataset are the scores of primary interest. For example, in our case, the primary interest is to find model fragments with high scores (between 3 and 4). Predicting that a model fragment has a score equal to 1 when its real score is 2 may not be important. However, predicting that a model fragment has a score equal to 1 when its real score is equal to 4 could be unacceptable. The cost of making the second error (misclassification cost) would be higher than the cost of making the first error. For this reason, most works, such as this work, use balanced training datasets. However, some research works [16, 17, 18] have been tackling

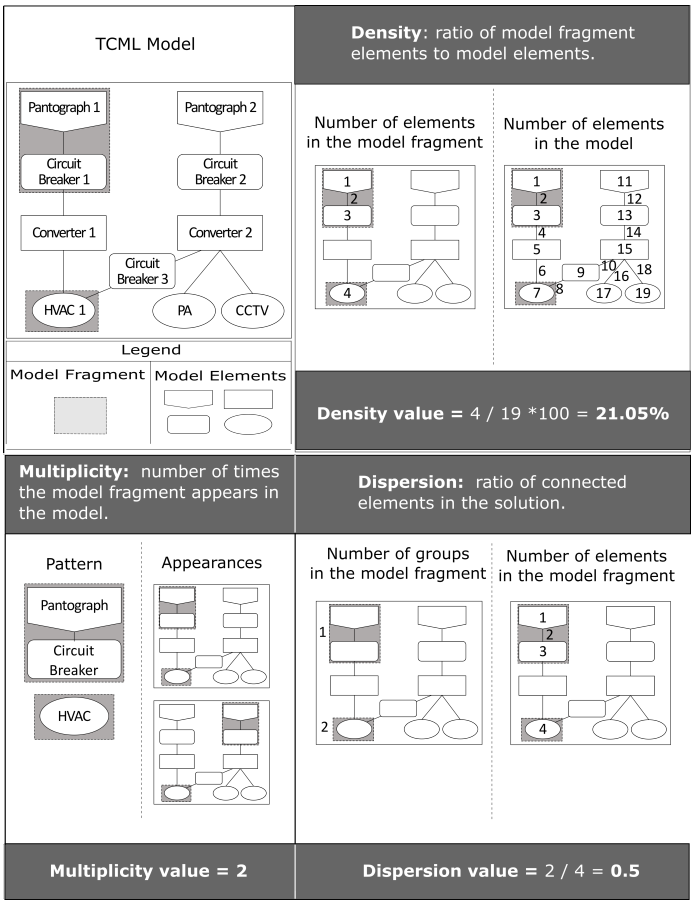


Figure 3: Example of a model fragment with its values for the properties: density, multiplicity, and dispersion.

the problems about learning from datasets with imbalanced distributions where the costs of misclassifying examples is non-uniform.

We consider that these properties (i.e., size and score distribution) must be used to report our training datasets. Nevertheless, we also think that the nature of the problem may require other more specific properties in order to provide all the necessary information for the replication of the experiments. For example, the properties identified in [8] to report the search space and the solutions in feature location problems are specifically based on models. In fact, among the identified properties, three of them (density, multiplicity, and dispersion) focus on properly reporting model fragments.

- **Density** measures the percentage of model elements that are present in a model fragment. Density is computed as the ratio of model fragment

elements to model elements. Figure 3 shows an example of how to calculate the density of a model fragment. In this example, the model fragment contains four elements (the *Pantograph 1*, the relation between *Pantograph 1* and *Circuit Breaker 1*, the *Circuit Breaker 1*, and the *HVAC*), and the model has 19 elements in total. Therefore, the density value is equal to 21.05%, which means that the model fragment is composed of 21.05% of the model elements.

- **Multiplicity** measures the number of times that the model fragment appears in the model. Figure 3 shows an example of how to calculate the multiplicity of a model fragment. In this example, the model fragment contains four elements: a pantograph that is connected to a circuit breaker, the relation to connect the pantograph and the circuit breaker, the circuit breaker connected to the pantograph, and a HVAC. Taking into account these elements, the multiplicity value is equal to 2, which means that there are two possible model fragments that contain these elements with their connections.

- **Dispersion** measures the ratio of connected elements in the model fragment. Specifically, a model fragment is composed of model elements, but these elements may or may not be connected in the model. The dispersion values are between 0 and 1, where 0 means that none of the model elements in the model fragment are connected with each other and 1 means that all the model elements in the model fragment are connected with each other. Figure 3 shows an example of how to calculate the dispersion of a model fragment. In this example, the model fragment contains four elements: the *Pantograph 1*, the relation between *Pantograph 1* and *Circuit Breaker 1*, the *Circuit Breaker 1*, and the *HVAC*. The first three elements are connected, so they compose the first group. The last element is not connected to the other elements so this element composes the second group. Since the dispersion is computed as the ratio between the number of groups and the number of elements, the dispersion value is equal to 0.5, which means that the model fragment has a medium dispersion.

Based on these properties, three domain experts discussed whether some of these three properties should be used to select a balanced training dataset from the knowledge base. The first expert believed that none of the properties would have an impact on the results. The second expert believed that density and dispersion would have an impact on the results, so the knowledge base should be sampled balancing these properties. The third one believed that the three properties would have an impact on the results, so the knowledge base should be sampled balancing the three properties. In fact, the decision about how to select a balanced training dataset from the knowledge base become complex and expensive. First, taking into account the domain experts' opinions, the sampling of the knowledge base is not obvious. Second, the more balanced the properties must be, the lower the number of triplets that will be leveraged from the knowledge base.

Let us create a training dataset, whose triplets are balanced taking into account the score. Therefore, the training dataset will contain a similar number of triplets for each score range (between 0 and 1, between 1 and 2, between 2 and 3, and between 3 and 4). However, the knowledge base only contains 1,500 triplets with scores between 2 and 3. In this case, to be balanced, the training dataset would have a size of around 6,000 triplets (4 score ranges x 1,500 triplets), even if the size of knowledge base is greater. Suppose that we decide to create a second training dataset, which not only balances the scores but also the density. The training dataset will contain a similar number of triplets for each score range and a similar number of triplets for each density range (between 0% and 25%, between 25% and 50%, between 50% and 75%, and between 75% and 100%). However, the knowledge base only contains 1,800 triplets with scores between 2 and 3 and only 100 of these triplets have density values between 75% and 100%. In this case, to be balanced, this second training dataset would have a size around of 1,600 triplets (4 scores ranges x 4 density ranges x 100 triplets). Therefore, the second training dataset discards 4,400 triples, which are leveraged in the first training dataset. In contrast, the second training dataset is balanced taking into account both the scores and the density.

Our industrial partner has been developing firmware since 1995, and 7,500 triplets have been documented in their knowledge base. However, if the knowledge base were sampled to get a training dataset balanced taking into account only one of the properties and the scores, the resulting training dataset would contain approximately 1,600 triplets. Therefore, we would be discarding at least 5,900 triplets. Also, a training dataset that was balanced taking into account all of the properties would contain a few hundred triplets, so most of the triplets of the knowledge base would be discarded.

The different beliefs of the domain experts indicate that density, multiplicity, and dispersion in the model fragments of the knowledge base may have an impact on the results, but the problem is complex enough to require an experiment to reach an agreement. First, given the time and effort required to manually retrieve each triplet, before sampling the knowledge base to obtain a balanced training dataset, an experiment can help us to determine if balancing any of the properties makes sense. Second, the experience can also help us to decide how to sample the knowledge base in order to take advantage of the effort and time invested. In other words, the experiment could provide information about which model fragments are the most relevant for learning in order to select the greatest number of triplets from the knowledge base. These two facts motivated us to propose the research questions and the experiment that are defined in Section 4.

4. Evaluation

Since the model fragments of the training datasets have different properties (density, multiplicity, and dispersion), the goal of this paper is to research the influence of these properties on the results of a machine learning based approach for feature location. To do this, this paper provides answers to several research

questions. The first three research questions focus on determining which properties influence the results:

RQ1: Do the density values of the model fragments in the training datasets influence the results?

RQ2: Do the multiplicity values of the model fragments in the training datasets influence the results?

RQ3: Do the dispersion values of the model fragments in the knowledge base influence the results?

The following two research questions focus on determining how the properties influence the results. Therefore, taking into account the previous research questions, this paper provides answers to the following questions for each property that influences the results:

RQ4: What values of a property should be covered by the model fragments in a training dataset in order to obtain the best results for model fragment location?

RQ5: If the solution found in the test cases have different property values than the model fragments in the training dataset, is it possible to obtain good results?

This section presents the evaluation that was performed to answer the RQs. It includes a description of the experimental setup, a description of the case study, the implementation details, the results obtained, and the statistical analysis performed.

4.1. Experimental Setup

The goal of this experiment is to provide answers to the RQs. To do this, the experiment was addressed by means of the feature location in models based on the (FLiM-ML) machine learning approach. However, instead of using the whole knowledge base for learning, we selected several subsets of the knowledge base covering different values of each property. These subsets are the training datasets for learning. Therefore, the approach used each training dataset to train a classifier, and each classifier was used to evaluate all of the test cases. Finally, the results of the approach were analyzed to provide the required answers. Figure 4 shows an overview of the process.

The left part of Figure 4 shows the inputs. Most of these inputs are extracted from the documentation provided by our industrial partner. The first input that is provided by our industrial partner is the **approved features**. The approved features are the best model fragments for the features in the test cases. Each approved feature is the correct solution of a test case. In other words, each approved feature is the result that we expect to obtain through the approach for a test case. This helps us to compare the results obtained by the approach

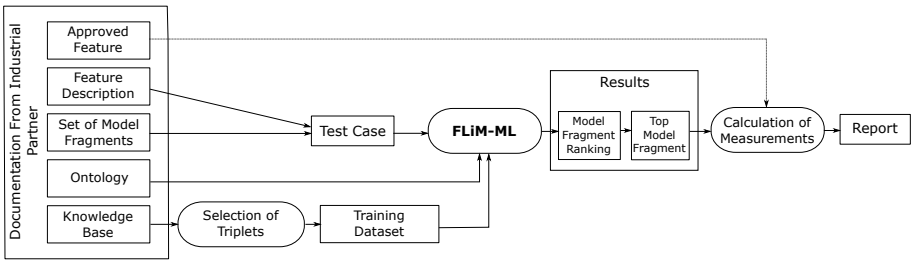


Figure 4: Experimental Setup.

with the correct solutions of the test cases in order to determine how good the obtained results are.

The following two inputs are the **feature descriptions** and the **sets of model fragments**. Each feature description defines a feature whose model fragment we want to find (in natural language). Each set of model fragments is composed of different model fragments, including the one that we want to find. Specifically, each feature description and the correspondent set of model fragments compose a test case. Each test case is evaluated through the FLiM-ML approach in order to find the best model fragment for the described feature among the model fragments in this set.

The last two inputs are the **ontology** and the **knowledge base**. Both inputs are necessary so that the approach can be used to evaluate the test cases. On the one hand, the ontology contains the main concepts, properties, and relations of a domain. Specifically, the ontology is used by the approach to characterize and encode the model fragments and feature descriptions into a format that is understandable for the machine learning techniques.

On the other hand, the knowledge base contains triplets with different property values (i.e., different scores, density values, multiplicity values, and dispersion values). Based on their property values, we select triplets to provide different training datasets from the knowledge base. To do this, we use purpose sampling, where items are selected based on their usefulness for achieving the study's objective [19]. Therefore, we select the triplets according to the three properties in order to answer the research questions. Specifically, each training dataset covers different values of the property that is being analyzed.

In order to answer RQ1, we select the triplets that have a density value between 0% and 25%, between 25% and 50%, between 50% and 75%, and between 75% and 100%. Therefore, we provide a training dataset whose triplets have extra-small density values (Density-XS), a training dataset with small density values (Density-S), a training dataset with medium density values (Density-M), and a training dataset with large density values (Density-L), respectively. Moreover, we provide another more training dataset that covers all density values (Density-A). Specifically, each quarter of the triplets in this dataset was selected according to the density values of the triplets in the knowledge base: between 0% and 25%, between 25% and 50%, between 50% and 75%, and be-

tween 75% and 100%.

In order to answer RQ2, we select the triplets that have a multiplicity value equal to 1 or greater to 1. The cutoff at 1 was selected taking into account the definition of multiplicity. A model fragment can be single (=1) or be repeated (¿1) in the product model. Therefore, we provide a training dataset whose triplets have unique model fragments for the product model (Multiplicity=1) and a training dataset whose triplets have non-unique model fragments for the product model (Multiplicity>1). Moreover, we provide another training dataset that covers all multiplicity values (Multiplicity-A). Specifically, half of triplets in this dataset have a multiplicity value equal to 1 and half of the triplets in this dataset have a multiplicity value greater than 1.

Likewise, in order to answer RQ3, we select the triplets that have a dispersion value between 0 and 0.25, between 0.25 and 0.50, between 0.50 and 0.75, and between 0.75 and 1. Therefore, we provide a training dataset of the knowledge base whose triplets have extra-small dispersion values (Dispersion-XS), a training dataset with small dispersion values (Dispersion-S), a training dataset with medium dispersion values (Dispersion-M), and a training dataset with large dispersion values (Dispersion-L), respectively. Moreover, we provide another training dataset that covers all dispersion values (Dispersion-A). Specifically, each quarter of the triplets in this dataset was selected according to the dispersion values of the triplets in the knowledge base: between 0 and 0.25, between 0.25 and 0.50, between 0.50 and 0.75, and between 0.75 and 1.

In summary, from the knowledge base, we select triplets to compose 13 different training datasets, where 5 training datasets are related to density values for RQ1, 3 training datasets are related to multiplicity values for RQ2, and 5 training datasets are related to dispersion values for RQ3. In our evaluation, each training dataset is used by the FLiM-ML approach to evaluate all of the test cases. Each combination of a training dataset and a test case provides a ranking of model fragments as output. The top fragment of this ranking is the best realization of the feature found by means of the FLiM-ML approach. This top model fragment (obtained solution) is compared with the correspondent approved feature (correct solution) in order to determine how good the obtained solution is. This comparison is performed by means of a confusion matrix.

A confusion matrix is a table that is often used to describe the performance of a classification model (in this case, the performance of each training dataset) on a set of test data (the test cases) for which the true values are known (from the approved features). In our case, each model fragment in the test cases obtains a score in the feature location process. Since the granularity is at the level of model elements, the presence or absence of each model element is considered as a classification. The confusion matrix distinguishes between the predicted values and the real values, classifying them into four categories:

- True Positive (TP): values that are predicted as true (in the obtained solution) and are true in the real scenario (in the correct solution).
- False Positive (FP): values that are predicted as true (in the obtained solution) but are false in the real scenario (in the correct solution).

- True Negative (TN): values that are predicted as false (in the obtained solution) and are false in the real scenario (in the correct solution).
- False Negative (FN): values that are predicted as false (in the obtained solution) but are true in the real scenario (in the correct solution).

Then, some performance measurements are derived from the values in the confusion matrix. Specifically, we create a report that includes four performance measurements (recall, precision, the F-measure, and the Matthews Correlation Coefficient) for each combination of a training dataset and a test case.

Recall measures the proportion of elements of the correct solution that are correctly retrieved by the obtained solution and is defined as follows:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Precision measures the proportion of elements from the obtained solution that are correct according to the correct solution and is defined as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

The F-measure corresponds to the harmonic mean of precision and recall and is defined as follows:

$$\text{F} - \text{measure} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}}$$

However, none of these previous measures take into account negative examples (TN). The Matthews Correlation Coefficient (MCC) is a correlation coefficient between the observed and predicted binary classifications that takes into account all of the observed values (TP, TN, FP, FN) and is defined as follows:

$$\text{MCC} = \frac{\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}$$

Recall values can range between 0% (i.e., no single model element from the correct solution is present in the obtained solution) and 100% (i.e., all of the model elements from the correct solution are present in the obtained solution). Precision values can range between 0% (i.e., no single model element from the obtained solution is present in the correct solution) and 100% (i.e., all of the model elements from the obtained solution are present in the correct solution). A value of 100% precision and 100% recall implies that both the obtained solution and the correct solution are the same. MCC values can range between -1 (i.e., there is no correlation between the obtained solution and the correct solution) to 1 (i.e., the obtained solution is perfect). Moreover, a MCC value of 0 corresponds to a random prediction.

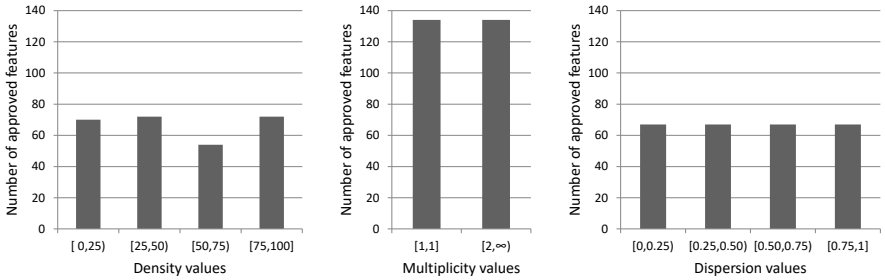


Figure 5: Distribution of the approved features for the different density, multiplicity, and dispersion values.

4.2. Case Study

The case study where we applied feature location was provided by our industrial partner CAF, and the domain is described in the background of this work. The case study includes the ontology, 268 test cases, and a knowledge base with 7,500 triplets. The ontology contains a total of 54 elements consisting of concepts and relations.

The 268 test cases form part of the learning material that the CAF company has to train new engineers. Each test case is composed of a feature description and a set of model fragments. The feature description is defined using natural language and contains about 25 words. The set of model fragments has about 100 model fragments with different values of density, multiplicity, and dispersion. Therefore, among the 100 model fragments, the engineer has to find the model fragment that best realizes the described feature. Similarly, the target of the approach is to evaluate these 100 model fragments in order to decide which one is the best realization of the described feature. Figure 5 shows the distribution of these model fragments regarding the density, multiplicity, and dispersion properties. Moreover, since these model fragments are the correct solutions, they always have a score equal to 4, so the score is not considered in the plots.

The knowledge base has about 7,500 triplets, but it does not contain the same number of triplets for each property value. For example, in the knowledge base, the number of triplets with small density values is greater than the number of triplets with large density values. The training datasets were created by selecting different triplets from this knowledge base. All of the training datasets are balanced according to the scores. This means that each training dataset contains the same number of triplets with scores between 0 and 1, between 1 and 2, between 2 and 3, and between 3 and 4. However, in the knowledge base, there are only 400 triplets with a low score (between 0 and 1) and an extra-small dispersion value (between 0 and 0.25). Therefore, the training dataset Dispersion-XS can contain a maximum of 1600 triplets with extra-small dispersion values, where 400 triplets have scores between 0 and 1, between 1 and 2, between 2 and 3, and between 3 and 4. Since the maximum number of triplets for this dataset is 1,600, we consider that all the training datasets contain 1,600

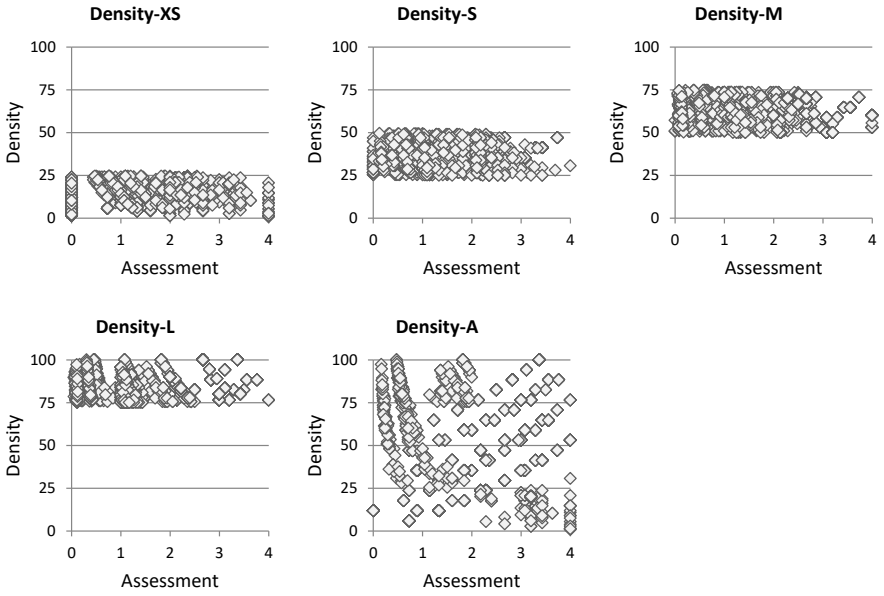


Figure 6: Distribution of the training datasets based on density (Density-XS, Density-S, Density-M, Density-L, and Density-A) according to the scores and density values of their triplets.

triplets of the knowledge base to be able to compare the results fairly.

The training datasets were created taking into account the research questions. For RQ1, five training datasets were created based on density: Density-XS, Density-S, Density-M, Density-L, and Density-A. Figure 6 shows the distribution of these datasets according to the scores and the density values of their triplets.

Then, for RQ2, three training datasets were created based on multiplicity: Multiplicity=1, Multiplicity>1, and Multiplicity-A. Figure 7 shows the distribution of these training datasets according to the scores and the multiplicity values of their triplets.

Finally, for RQ3, five training datasets were created based on dispersion: Dispersion-XS, Dispersion-S, Dispersion-M, Dispersion-L, and Dispersion-A. Figure 8 shows the distribution of these datasets according to the scores and the dispersion values of their triplets.

For each training dataset, we evaluated all of the test cases. Moreover, each combination of training dataset and test case was run 30 times. As suggested by [20], given the stochastic nature of the FLiM-ML approach, several repetitions are needed to obtain reliable results. Finally, the results (obtained solutions) were compared to the approved features (correct solutions), which are the solutions for each test case. Therefore, the approved features correspond to 268 model fragments, one for each test case.

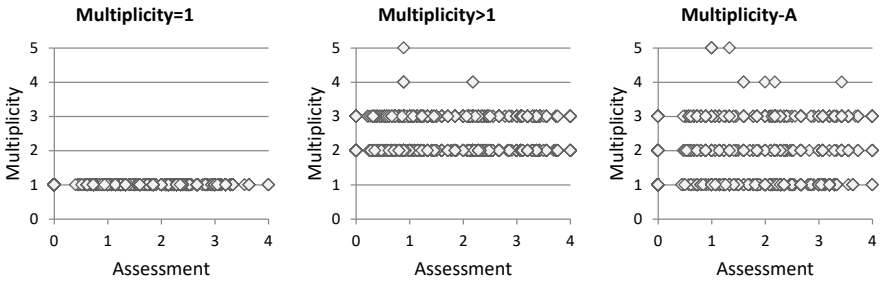


Figure 7: Distribution of the training datasets based on multiplicity (Multiplicity=1, Multiplicity>1, Multiplicity-A) according to the scores and multiplicity values of their triplets

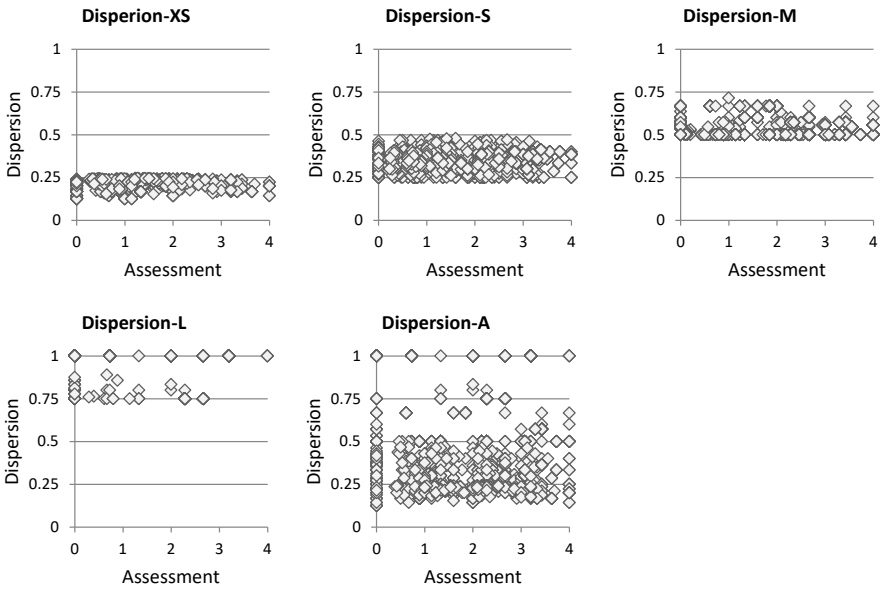


Figure 8: Distribution of the training datasets based on dispersion (Dispersion-XS, Dispersion-S, Dispersion-M, Dispersion-L, and Dispersion-A) according to the scores and dispersion values of their triplets

Table 1: Kolmogorov–Smirnov test with the Lilliefors correction for all of the training datasets.

	Score		Density		Multiplicity		Dispersion	
	D	*p-Value*	D	*p-Value*	D	*p-Value*	D	*p-Value*
Density-XS	0.103	$< 2.20 \times 10^{-16}$	0.128	$< 2.20 \times 10^{-16}$	0.259	$< 2.20 \times 10^{-16}$	0.259	$< 2.20 \times 10^{-16}$
Density-S	0.109	$< 2.20 \times 10^{-16}$	0.111	$< 2.20 \times 10^{-16}$	0.278	$< 2.20 \times 10^{-16}$	0.144	$< 2.20 \times 10^{-16}$
Density-M	0.092	$< 2.20 \times 10^{-16}$	0.098	$< 2.20 \times 10^{-16}$	0.286	$< 2.20 \times 10^{-16}$	0.150	$< 2.20 \times 10^{-16}$
Density-L	0.156	$< 2.20 \times 10^{-16}$	0.129	$< 2.20 \times 10^{-16}$	0.286	$< 2.20 \times 10^{-16}$	0.128	$< 2.20 \times 10^{-16}$
Density-A	0.101	$< 2.20 \times 10^{-16}$	0.096	$< 2.20 \times 10^{-16}$	0.261	$< 2.20 \times 10^{-16}$	0.152	$< 2.20 \times 10^{-16}$
Multiplicity=1	0.109	$< 2.20 \times 10^{-16}$	0.164	$< 2.20 \times 10^{-16}$	*	*	0.265	$< 2.20 \times 10^{-16}$
Multiplicity>1	0.099	$< 2.20 \times 10^{-16}$	0.088	$< 2.20 \times 10^{-16}$	0.334	$< 2.20 \times 10^{-16}$	0.119	$< 2.20 \times 10^{-16}$
Multiplicity-A	0.095	$< 2.20 \times 10^{-16}$	0.183	$< 2.20 \times 10^{-16}$	0.314	$< 2.20 \times 10^{-16}$	0.217	$< 2.20 \times 10^{-16}$
Dispersion-XS	0.106	$< 2.20 \times 10^{-16}$	0.101	$< 2.20 \times 10^{-16}$	0.262	$< 2.20 \times 10^{-16}$	0.204	$< 2.20 \times 10^{-16}$
Dispersion-S	0.103	$< 2.20 \times 10^{-16}$	0.115	$< 2.20 \times 10^{-16}$	0.256	$< 2.20 \times 10^{-16}$	0.105	$< 2.20 \times 10^{-16}$
Dispersion-M	0.097	$< 2.20 \times 10^{-16}$	0.161	$< 2.20 \times 10^{-16}$	0.258	$< 2.20 \times 10^{-16}$	0.394	$< 2.20 \times 10^{-16}$
Dispersion-L	0.251	$< 2.20 \times 10^{-16}$	0.248	$< 2.20 \times 10^{-16}$	0.475	$< 2.20 \times 10^{-16}$	0.467	$< 2.20 \times 10^{-16}$
Dispersion-A	0.108	$< 2.20 \times 10^{-16}$	0.160	$< 2.20 \times 10^{-16}$	0.259	$< 2.20 \times 10^{-16}$	0.209	$< 2.20 \times 10^{-16}$

* The standard deviation is zero

4.2.1. Strength of Correlations

The training datasets are selected to analyze the influence of a specific property. However, all of the training datasets are composed of triplets that have all of the properties: score, density, multiplicity, and dispersion. Therefore, it is worth knowing whether there is a correlation between the specific property that is being analyzed through a training dataset and the rest of the properties.

Since the training datasets have 1600 triplets, we use the Kolmogorov–Smirnov test with the Lilliefors correction to study the normality of each property in each training dataset. Table 1 shows the values of the test. All of the properties have *p-values* under 0.05 in all of the test cases, which means that none of the properties have a normal distribution. Moreover, the histograms in the appendix show the distributions of the properties for all of the training datasets. These plots show that the scores always have a uniform distribution, which is desirable in order to be able to obtain balanced training datasets for learning.

Since the properties do not have normal distributions, we used a non-parametric test to measure the correlation. Specifically, the Spearman correlation test was not only applied to measure the correlation but also to verify the strength of the correlations.

The first five training datasets (Density-XS, Density-S, Density-M, Density-L, and Density-A) are selected to analyze the influence of the density property. Therefore, we first analyze whether there is any correlation between the density property and the rest of the properties in these training datasets. Table 2 shows the Spearman correlation coefficients, which indicate the strength of the correlation between the density property and one of the other properties in these training datasets. A correlation coefficient of zero indicates that there is no relation, and a correlation coefficient of -1 or $+1$ indicates a perfect

Table 2: Spearman's correlation coefficients between the density and the rest of the properties (Score, Multiplicity, and Dispersion) in density-based training datasets: Density-XS, Density-S, Density-M, Density-L, and Density-A.

	Score	Multiplicity	Dispersion
Density-XS	0.204	0.099	-0.090
Density-S	0.047	0.115	0.022
Density-M	0.072	0.162	0.049
Density-L	-0.054	-0.489	-0.454
Density-A	0.083	0.338	-0.450

Table 3: Spearman's correlation coefficients between the multiplicity and the rest of the properties (Score, Density, and Dispersion) in multiplicty-based training datasets: Multipliciyty=1, Multiplicity>1, and Multiplicity-A.

	Score	Density	Dispersion
Multiplicity=1	*	*	*
Multiplicity>1	9.921×10^{-5}	-0.004	-0.016
Multiplicity-A	0.013	0.247	-0.347

* The standard deviation is zero

relationship between them. Most correlation coefficients in Table 2 indicate that there is a weak relation or there is no relation between the density and the other properties. However, there is a moderate relation between the density and the multiplicity and a moderate relation between the density and the dispersion in the last two training datasets, Density-L and Density-A.

The following three training datasets (Multipliciyty=1, Multiplicity>1, and Multiplicity-A) are selected to analyze the influence of the multiplicity property. Therefore, we first analyze whether there is any correlation between the multiplicity property and the rest of the properties in these training datasets. Table 3 shows the Spearman correlation coefficients, which indicate the strength of the correlation between the multiplicity property and one of the other properties in these training datasets. Most correlation coefficients in Table 3 indicate that there is no relation between the density and the other properties. However, there is a weak relation between the multiplicity and the density and a weak relation between the multiplicity and the dispersion in the last training dataset, Multiplicity-A.

The following last five training datasets (Dispersion-XS, Dispersion-S, Dispersion-M, Dispersion-L, and Dispersion-A) are selected to analyze the influence of the dispersion property. Therefore, we first analyze if there is any correlation between the dispersion property and the rest of the properties in these training datasets. Table 4 shows the Spearman correlation coefficients, which indicates the strength of the correlation between the dispersion property and one of the other properties in these training datasets. Most correlation coefficients in Table 4 indicate that there is a weak relation or there is no relation between the dispersion and the other properties. However, there is a moderate relation between

Table 4: Spearman's correlation coefficients between the dispersion and the rest of the properties (Score, Density, and Multiplicity) in dispersion-based training datasets: Dispersion-XS, Dispersion-S, Dispersion-M, Dispersion-L, and Dispersion-A.

	Score	Density	Multiplicity
Dispersion-XS	0.054	0.392	0.205
Dispersion-S	0.203	0.094	0.161
Dispersion-M	0.072	0.048	0.162
Dispersion-L	-0.053	-0.454	-0.489
Dispersion-A	0.053	-0.358	-0.213

the dispersion and the multiplicity in the Dispersion-L training dataset.

According to the correlation coefficients, there are a few moderate relations. However, in most of the cases, there are no relations or the relations are weak. In fact, none of the moderate relations correspond to the training datasets that achieve the best results in our evaluation.

4.3. FLiM-ML Approach.

The FLiM-ML [5] is a machine learning-based approach for feature location on models. In this work, the FLiM-ML approach was used to evaluate the different training datasets that serve to answer the RQs of this work. Figure 9 shows an overview of the FLiM-ML approach, whose objective is to provide a ranking of model fragments. The top model fragment in the ranking is the best realization found for a feature description. To do this, the approach has two phases: training and testing.

In the training phase, a classifier is trained to learn how well each model fragment realizes a specific feature description. To do this, the input consists of an ontology and a training dataset. In our case, instead of using the whole knowledge base, we used the training datasets that are described in the experimental setup and the case study.

The training phase consists of four steps:

1. Encoding: The ontology is used to encode the triplets of a training dataset into feature vectors, as described in [21]. Moreover, since both feature descriptions and model fragments are based on natural language, the terms used in the ontology do not always align well with the terms in the feature description and with the terms in the model fragments. For this reason, before encoding, the feature descriptions and the model fragments are processed by a combination of NLP techniques defined in [22], which consists of tokenizing, lowercasing, removal of duplicate keywords, syntactical analysis, lemmatization, and stopword removal.

2. Feature selection: A mask is applied to select only the most relevant characteristics in the feature vectors. To do this, a set of masks is generated and evolved by means of an evolutionary algorithm as in [21]. As a result, the top mask in the ranking, which is obtained from the evolutionary algorithm, indicates the selected characteristics.

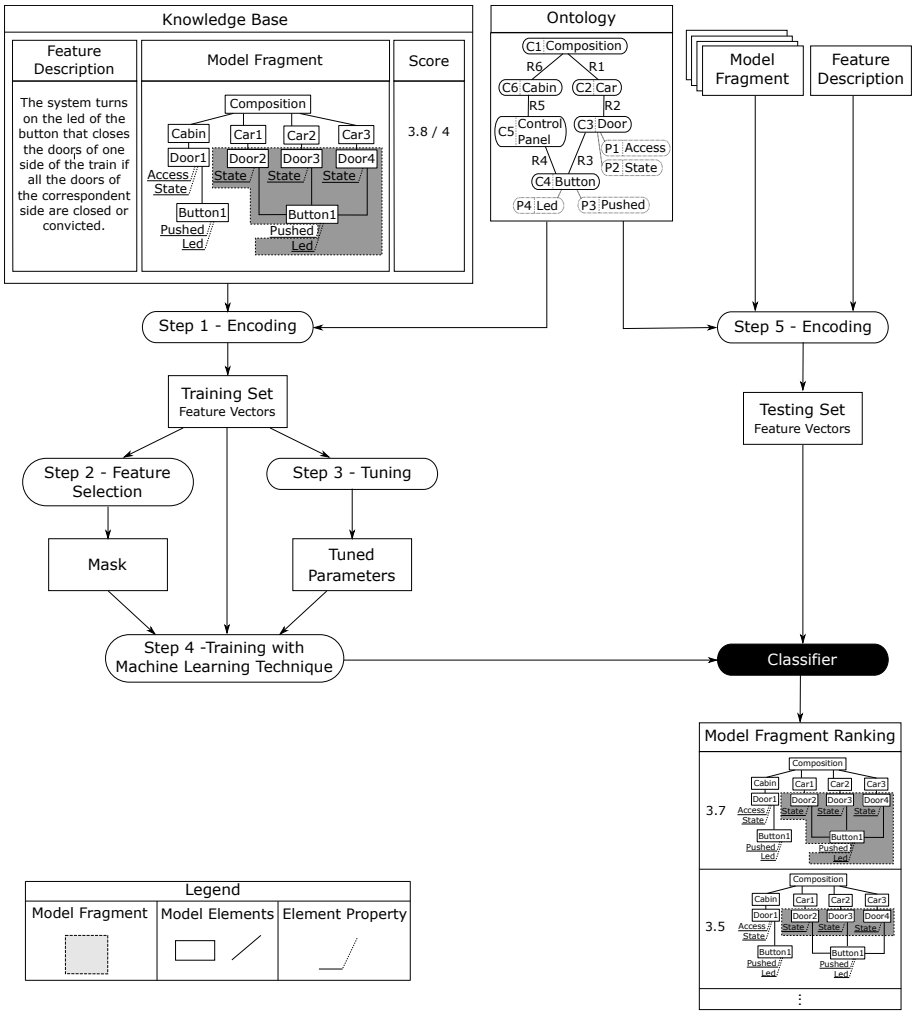


Figure 9: Overview of the FLiM-ML approach.

3. Tuning: This determines what parameters must be used to obtain the best performance of the machine learning technique. The parameter tuning in the FLiM-ML approach is based on grid search. First, a grid search is built to determine the values of the parameters, which depend on the machine learning technique selected. Then, the FLiM-ML approach uniformly samples each of the parameters in their range and evaluates all of the combinations of the sampled values.

4. Training with machine learning technique: The feature vectors are used to train the classifier, which learns a rule-set through the comparison of the feature vectors [23]. However, before using this classifier in the testing phase, it is worth analyzing the performance of the classifier through cross-validation. Cross-validation is a statistical method of evaluating and comparing machine learning techniques by dividing data into two segments: one used to train a classifier, and the other used to validate the classifier [24]. Moreover, to reduce variability, multiple rounds of cross-validation are performed using different partitions, and the results are averaged over the rounds [25].

   The results of the cross-validation provide the performance of the classifier. If this performance is not considered suitable, it is necessary to perform another training iteration. In this iteration, some artifacts of the training phase (e.g., the encoding, the ontology, the training dataset, or the machine learning technique) must be modified in order to improve the classifier. Otherwise, if the performance is considered suitable, the classifier obtains the go-ahead, so the classifier trained with the training dataset is used in the testing phase. Once the classifier has been generated, the training phase does not have to be repeated again. The same classifier is used to evaluate all of the test cases for the same training dataset in the testing phase. Therefore, the classifier is considered as both an artifact (output of Step 4 in the training phase) and a step (responsible for testing the test cases in the testing phase). For this reason, Figure 9 shows the classifier in a black, rounded rectangle to point out its double meaning.

In the testing phase, the classifier is used to rank the set of model fragments according to a feature description described in natural language. To do this, the input consists of the set of model fragments, the feature description, and an ontology. Each model fragment realizes the feature description to a greater or lesser extent.

The testing phase consists of two steps:

5. Encoding: The ontology is used to encode each model fragment and the feature description. To be fair, both the characteristics of the encoding and the ontology must be same for the the training phase and the testing phase. As a result, each model fragment and the feature description are encoded as a feature vector.

6. Classifier: Each feature vector is then tested by the classifier, which uses the learned rule-set to assign a score to each one of them. This score is a

numerical value that is equal to or greater than 0. The higher the score, the closer the model fragment to the feature description. Therefore, this model fragment would be a better realization of the feature description than any other model fragment with a lower score. Taking into account the scores, the model fragments can be ordered in a ranking where the top positions are occupied by the model fragments with the highest relevance to the feature description. This ranking is the final result of the FLiM-ML approach.

4.4. Implementation details

We used the Eclipse Modeling Framework to manipulate the models and CVL to manage the model fragments. With regard to the machine learning technique, FLiM-ML can support several machine learning techniques, such as Rankboost.

RankBoost belongs to the family of Learning to Rank, whose algorithms are specifically designed to perform ranking tasks. Moreover, Rankboost is well known for its efficiency and effectiveness in different domains [26, 27]. In fact, among the machine learning techniques supported by FLiM-ML, Rankboost is notable for its good results in other fragment location problems [9]. For these reasons, Rankboost was selected as the machine learning technique for the approach in this work.

RankBoost was implemented using the RankLib library [28] and tuned using a grid search. Specifically, Rankboost has two parameters that have to be tuned to improve its performance. The first one is related to the number of iterations to search in each dimension. The second one is related to the metric, which is used to evaluate the test data. The values tuned for our evaluation correspond to *iteration* = 200 and *metric* equal to ERR10, respectively.

4.5. Results

In Table 5, we outline the mean results for the test cases regarding the training datasets (Density-XS, Density-S, Density-M, Density-L, and Density-A) that answer RQ1. Each column shows the Precision, Recall, F-measure, and MCC obtained through each dataset.

In response to RQ1, there are clear differences between the results obtained for the different training datasets. A Table 5 shows, Density-XS achieves the best results for all of the performance indicators, providing a mean precision value of 89.74%, a recall value of 82.26%, a F-measure value of 84.16%, and a MCC value of 0.81. In contrast, Density-L presents the worst results for all of the indicators, providing a mean precision value of 46.74%, a recall value of 41.22%, a F-measure value of 42.71%, and a MCC value of 0.28. These results indicate that the density values of the model fragments in the training dataset influence the results. However, the statistical analysis is the final step to claim that the results of the five training datasets are significantly different and assesses the magnitude of the improvement from Density-XS.

Therefore, since RQ1 was answered affirmatively, we needed to answer RQ4 and RQ5 for the density property. **In response to RQ4**, Table 5 shows that

Table 5: Mean Values and Standard Deviations for Precision, Recall, F-Measure, and MCC for the test cases regarding the training datasets (Density-XS, Density-S, Density-M, Density-L, and Density-A) that answer RQ1.

	Precision	Recall	F-measure	MCC
Density-XS	89.74 ± 25.46	82.26 ± 32.11	84.16 ± 29.50	0.81
Density-S	60.73 ± 43.47	58.62 ± 45.21	58.41 ± 44.07	0.47
Density-M	59.17 ± 44.34	57.22 ± 45.32	57.47 ± 44.44	0.48
Density-L	46.74 ± 40.57	41.22 ± 41.35	42.71 ± 40.85	0.28
Density-A	82.06 ± 32.84	74.84 ± 37.50	76.35 ± 35.47	0.72

Density-XS achieves the best results for all of the performance indicators. Moreover, there is at least a difference of 7% between the recall, the precision, and the F-measure obtained for Density-XS and the rest of the training datasets. For the MCC, the difference between Density-XS and the rest of the training datasets is at least equal to 0.09. These results indicate that a machine learning-based approach like FLiM-ML can obtain different results depending on the density values of the model fragments in the training dataset. In fact, for FLiM-ML and the case study presented in this work, the training dataset should cover the extra-small density values in order to obtain the best results.

To answer RQ5, Table 6 shows the mean results for the test cases grouped by the density values of their approved features. Each column shows the Precision, Recall, F-measure, and MCC obtained through each training dataset. Figure 5 shows the number of approved features according to their density values.

In response to RQ5, as Table 6 shows, Density-XS achieves the best results for all of the performance indicators regardless of the test cases. Density-XS obtained the best results for the test cases whose approved features have density values between 0 and 25, providing a mean precision value of 87.85%, a recall value of 86.83%, a F-measure value of 86.80%, and a MCC value of 0.86. This same training dataset obtained the best results for the test cases whose approved features have density values between 25 and 50, providing a mean precision value of 96.30%, a recall value of 90.74%, a F-measure value of 92.10%, and a MCC value of 0.92. This same training dataset also obtained the best results for the test cases whose approved features have density values between 50 and 75, providing a mean precision value of 75.93%, a recall value of 63.37%, a F-measure value of 68.16%, and a MCC value of 0.55. Finally, the same training dataset obtained the best results for the test cases whose approved features have density values between 75 and 100, providing a mean precision value of 95.37%, a recall value of 83.49%, a F-measure value of 85.66%, and a MCC value of 0.83. Therefore, although the approved features for the test cases have different density values than the model fragments in the training dataset, the obtained results can be even better than the results obtained using model fragments with similar density values.

Table 6: Mean Values and Standard Deviations for Precision, Recall, F-Measure, and MCC for the test cases grouped by the density values of their approved features regarding the training datasets: Density-XS, Density-S, Density-M, Density-L, and Density-A.

Density values in test cases	Training Dataset	Precision	Recall	F-measure	MCC
[0,25)	Density-XS	87.85 ± 27.18	86.83 ± 30.40	86.80 ± 29.08	0.86
	Density-S	59.40 ± 45.88	59.44 ± 47.76	59.11 ± 46.87	0.58
	Density-M	55.76 ± 48.52	56.86 ± 49.00	56.24 ± 48.71	0.55
	Density-L	41.86 ± 44.02	39.75 ± 44.85	40.08 ± 44.55	0.38
	Density-A	85.21 ± 30.43	84.03 ± 32.82	83.87 ± 31.94	0.83
[25,50)	Density-XS	96.30 ± 17.52	90.74 ± 25.16	92.10 ± 21.94	0.92
	Density-S	61.33 ± 39.08	60.62 ± 41.52	59.06 ± 39.57	0.49
	Density-M	45.92 ± 43.43	44.44 ± 44.78	43.99 ± 42.95	0.35
	Density-L	41.96 ± 39.69	36.48 ± 39.83	37.79 ± 39.31	0.28
	Density-A	76.61 ± 40.63	73.33 ± 41.57	73.72 ± 40.59	0.73
[50,75)	Density-XS	75.93 ± 37.71	63.37 ± 35.24	68.16 ± 35.40	0.55
	Density-S	45.85 ± 45.07	44.59 ± 44.44	44.70 ± 44.17	0.20
	Density-M	51.57 ± 43.13	45.06 ± 40.06	47.09 ± 40.33	0.27
	Density-L	42.86 ± 39.96	36.61 ± 35.70	38.64 ± 36.57	0.14
	Density-A	72.84 ± 35.66	59.67 ± 35.88	63.70 ± 34.48	0.49
[75,100]	Density-XS	95.37 ± 11.27	83.49 ± 32.46	85.66 ± 27.67	0.83
	Density-S	72.57 ± 41.26	66.36 ± 45.33	67.36 ± 43.84	0.56
	Density-M	81.44 ± 33.09	79.48 ± 37.48	79.94 ± 35.80	0.70
	Density-L	59.18 ± 36.38	50.84 ± 42.47	53.25 ± 40.57	0.28
	Density-A	91.36 ± 19.08	78.80 ± 35.72	81.15 ± 31.59	0.77

Table 7: Mean Values and Standard Deviations for Precision, Recall, F-Measure, and MCC for the test cases regarding the training datasets (Multiplicity=1, Multiplicity>1, and Multiplicity-A) that answer RQ2.

	Precision	Recall	F-measure	MCC
Multiplicity=1	91.13 ± 23.44	89.91 ± 26.10	89.86 ± 24.83	0.89
Multiplicity>1	90.26 ± 24.77	87.94 ± 28.26	88.16 ± 26.56	0.88
Multiplicity-A	92.14 ± 21.99	89.97 ± 26.12	90-06 ± 24.25	0.90

Table 8: Mean Values and Standard Deviations for Precision, Recall, F-Measure, and MCC for the test cases regarding the training datasets (Dispersion-XS, Dispersion-S, Dispersion-M, Dispersion-L, and Dispersion-A) that answer RQ3.

	Precision	Recall	F-measure	MCC
Dispersion-XS	86.84 ± 30.38	86.96 ± 26.29	85.63 ± 28.72	0.85
Dispersion-S	86.79 ± 30.24	87.21 ± 26.56	85.75 ± 28.83	0.85
Dispersion-M	33.61 ± 32.31	34.61 ± 33.88	32.72 ± 32.48	0.28
Dispersion-L	66.58 ± 39.97	67.20 ± 35.98	65.86 ± 37.77	0.64
Dispersion-A	73.73 ± 37.16	78.77 ± 29.27	73.80 ± 33.89	0.73

In Table 7, we outline the mean results for the test cases for the training dataset (Multiplicity=1, Multiplicity>1, and Multiplicity-A) that answer RQ2. Each column shows the Precision, Recall, F-measure, and MCC obtained through each dataset.

In response to RQ2, the results obtained for all the training datasets are very similar. Therefore, Table 7 does not have enough evidence to say with certainty that multiplicity influences the results. For this reason, RQ2 is answered taking into account the statistical analysis.

In Table 8, we outline the mean results for the test cases for the training datasets (Dispersion-XS, Dispersion-S, Dispersion-M, Dispersion-L, and Dispersion-A) that answer RQ3. Each column shows the Precision, Recall, F-measure, and MCC obtained through each dataset.

In response to RQ3, there are clear differences between the results obtained for the different training datasets. A Table 5 shows, Dispersion-XS and Dispersion-S achieve the best results: Dispersion-XS attains 86.84% precision, 86.96% recall, 85.63% F-measure, and 0.85 MCC; and Dispersion-S attains 86.79% precision, 87.21% recall, 85.75% F-measure, and 0.85 MCC. In contrast, Dispersion-M presents the worst results in all of the indicators, providing a mean precision value of 33.61%, a recall value of 34.61%, a F-measure value of 32.72%, and a MCC value of 0.28. These results indicate that the dispersion of the model fragments in the training dataset influence the results. However, the statistical analysis is the final step to claim that the results of the five training datasets are significantly different and assesses the magnitude of the

improvement from Dispersion-XS or Dispersion-S.

Therefore, since RQ3 was answered affirmatively, we needed to answer RQ4 and RQ5 for the dispersion property. **In response to RQ4**, Table 8 shows that Dispersion-XS and Dispersion-S achieve the best results for all of the performance indicators. Moreover, the difference between the recall, the precision, and the F-measure obtained for these two training datasets and the rest of the training datasets is greater than 12%. For MCC, the difference between these two training datasets and the rest of the training datasets is at least equal to 0.12. To all appearances, these results indicate that a machine learning-based approach like FLiM-ML can obtain different results depending on the dispersion values of the model fragments in the training dataset. In fact, for FLiM-ML and the case study presented in this work, the training dataset should cover extra-small dispersion values or small dispersion values in order to obtain the best results. However, a statistical analysis can determine if whether or not there are significant differences between the results of FLiM-ML for extra-small dispersion values and the results for small dispersion values.

To answer RQ5, Table 9 shows the mean results for the test cases grouped by the dispersion values of the approved features. Each column shows the Precision, Recall, F-measure, and MCC obtained through each training dataset. Figure 5 shows the number of approved features according to their dispersion values.

In response to RQ5, as Table 9 shows, four training datasets (Dispersion-XS, Dispersion-S, Dispersion-L, and Dispersion-A) achieve similar results for the test cases where the approved features have dispersion values between 0 and 0.50. Three training datasets (Dispersion-XS, Dispersion-S, and Dispersion-A) achieve similar results for test cases where the approved features have dispersion values between 0.75 and 1 or between 0.50 and 0.75. Finally, for the test cases where the approved features have dispersion values between 0.75 and 1, two training datasets (Dispersion-XS and Dispersion-S) achieve similar results. Therefore, for the dispersion property, there are more than one training datasets that obtain the best results for test cases where the approved features have different dispersion values than the model fragments in the training datasets.

4.6. Statistical Analysis

A statistical test must be run to assess whether there is enough empirical evidence to claim that there is a difference between two approaches (e.g., A is better than B). To achieve this, two hypotheses are defined: the null hypothesis H_0, and the alternative hypothesis H_1. The null hypothesis H_0 is typically defined to state that there is no difference between the approaches, whereas the alternative hypothesis H_1 states that the training datasets differ. In such a case, a statistical test aims to verify whether the null hypothesis H_0 should be rejected.

Statistical tests provide a probability value, *p-Value*. The *p-Value* obtains values between 0 and 1. The lower the *p-Value* of a test, the more likely that the null hypothesis is false. It is accepted by the research community that a *p-Value* under 0.05 is statistically significant [29], and so the hypothesis H_0 can be considered false.

Table 9: Mean Values and Standard Deviations for Precision, Recall, F-Measure, and MCC for the test cases grouped by the dispersion values of their approved features regarding the training datasets: Dispersion-XS, Dispersion-S, Dispersion-M, Dispersion-L, and Dispersion-A.

Dispersion values in test cases	Training Dataset	Precision	Recall	F-measure	MCC
[0,0.25)	Dispersion-XS	100.00 ± 0.00	88.56 ± 7.79	93.76 ± 4.25	0.93
	Dispersion-S	100.00 ± 0.00	89.55 ± 8.12	94.30 ± 4.43	0.94
	Dispersion-M	30.27 ± 12.57	17.41 ± 11.76	21.75 ± 11.37	0.14
	Dispersion-L	100.00 ± 0.00	88.56 ± 7.79	93.76 ± 4.25	0.93
	Dispersion-A	100.00 ± 0.00	89.30 ± 8.05	94.17 ± 4.39	0.94
[0.25,0.50)	Dispersion-XS	95.24 ± 15.82	93.37 ± 20.20	93.68 ± 18.84	0.93
	Dispersion-S	95.14 ± 15.07	92.87 ± 21.60	93.43 ± 19.58	0.92
	Dispersion-M	54.04 ± 44.00	51.14 ± 45.02	51.79 ± 44.75	0.46
	Dispersion-L	77.11 ± 36.41	73.51 ± 40.10	74.69 ± 38.64	0.71
	Dispersion-A	94.89 ± 15.98	92.44 ± 21.65	93.02 ± 19.96	0.92
[0.50,0.75)	Dispersion-XS	73.96 ± 42.87	74.13 ± 42.36	73.82 ± 42.44	0.73
	Dispersion-S	74.10 ± 42.54	74.63 ± 42.49	74.14 ± 42.27	0.73
	Dispersion-M	23.66 ± 37.42	27.36 ± 36.55	24.88 ± 36.71	0.21
	Dispersion-L	58.21 ± 49.50	56.72 ± 49.06	57.16 ± 49.04	0.54
	Dispersion-A	74.23 ± 42.73	75.12 ± 42.82	74.14 ± 42.44	0.72
[0.75,1]	Dispersion-XS	78.17 ± 34.87	91.79 ± 18.78	81.28 ± 30.45	0.82
	Dispersion-S	77.92 ± 35.13	91.79 ± 18.78	81.13 ± 30.63	0.81
	Dispersion-M	26.46 ± 13.39	42.54 ± 21.91	32.47 ± 16.40	0.30
	Dispersion-L	31.00 ± 8.22	50.00 ± 15.19	37.81 ± 9.60	0.37
	Dispersion-A	25.81 ± 9.30	58.21 ± 20.72	33.86 ± 8.85	0.35

Table 10: Quade test statistic and *p-Values*.

		Precision	Recall	F-Measure	MCC
Density	p-Value	$< 2.20 \times 10^{-16}$	$< 2.20 \times 10^{-16}$	$< 2.20 \times 10^{-16}$	$< 2.20 \times 10^{-16}$
	Statistic	79.79	63.62	71.41	65.50
Multiplicity	p-Value	0.41	0.15	0.35	0.51
	Statistic	0.87	1.86	1.06	0.67
Dispersion	p-Value	$< 2.20 \times 10^{-16}$	$< 2.20 \times 10^{-16}$	$< 2.20 \times 10^{-16}$	$< 2.20 \times 10^{-16}$
	Statistic	118.7	126.6	111.89	117.41

The test carried out depends on the properties of the data. Since our data does not follow a normal distribution in general, our analysis required the use of non-parametric techniques. There are several tests for analyzing this kind of data; however, the Quade test is the most powerful one when working with real data [30]. In addition, according to Conover [31], the Quade test is the one that has shown the best results for a low number of datasets (no more than 4 or 5 datasets).

Table 10 shows the Quade test statistic and *p-Values* for precision, recall, F-measure, and MCC. In the case of density and dispersion, the *p-Values* are smaller than 0.05, so we could reject the null hypothesis. In contrast, in the case of multiplicity, the *p-Values* are not smaller than 0.05, so we could not reject the null hypothesis. Consequently, we can state that there are significant differences among the results of the five training datasets for density and among the results of the five training datasets for dispersion. However, there are no significant differences among the results of the three training datasets for multiplicity. Therefore, in response to RQ1, the statistical analysis determines that density influences the results. In response to RQ2, the statistical analysis determines that multiplicity does not influence the results. In response to RQ3, the statistical analysis determines that dispersion influences the results.

Nevertheless, with the Quade test, we cannot answer the following question: *Which of the training datasets regarding density gives the best performance?* and *Which of the training datasets regarding dispersion gives the best performance* In this case, the performance of each training dataset should be individually compared with all of the other alternatives. In order to do this, we performed an additional post hoc analysis. This kind of analysis performs a pair-wise comparison among the results of each training dataset, determining whether statistically significant differences exist among the results of a specific pair of datasets.

Table 11 shows the *p-Values* of Holm's post hoc analysis for each specific pair of training datasets according to the RQs. Most of the *p-Values* shown in this table are smaller than 0.05, so these comparisons have significant differences for all of the performance measurements. However, Table 11 shows that there are no significant differences between Density-S and Density-M, between Dispersion-XS and Dispersion-S, and between Dispersion-L and Dispersion-A.

Table 11: Holm's Post Hoc *p-Values*.

		Precision	Recall	F-Measure	MCC
Density	Density-XS vs Density-S	$< 2.0 \times 10^{-16}$	5.8×10^{-11}	7.6×10^{-14}	1.4×10^{-13}
	Density-XS vs Density-M	$< 2.0 \times 10^{-16}$	1.4×10^{-12}	1.0×10^{-14}	4.9×10^{-15}
	Density-XS vs Density-L	$< 2.0 \times 10^{-16}$	$< 2.0 \times 10^{-16}$	$< 2.0 \times 10^{-16}$	$< 2.0 \times 10^{-16}$
	Density-XS vs Density-A	1.7×10^{-3}	9.7×10^{-3}	7.1×10^{3}	3.9×10^{-3}
	Density-S vs Density-M	0.37	0.5	0.64	1.0
	Density-S vs Density-L	1.4×10^{-5}	4.9×10^{-8}	6.3×10^{-7}	9.8×10^{-7}
	Density-S vs Density-A	4.7×10^{-12}	6.4×10^{-8}	8.1×10^{-10}	3.5×10^{-10}
	Density-M vs Density-L	1.1×10^{-7}	2.2×10^{-9}	9.8×10^{-9}	1.0×10^{-7}
	Density-M vs Density-A	$< 2.0 \times 10^{-16}$	1.6×10^{-11}	2.0×10^{-13}	5.4×10^{-16}
	Density-L vs Density-A	$< 2.0 \times 10^{-16}$	$< 2.0 \times 10^{-16}$	$< 2.0 \times 10^{-16}$	$< 2.0 \times 10^{-16}$
Dispersion	Dispersion-XS vs Dispersion-S	0.99	0.49	0.61	0.6
	Dispersion-XS vs Dispersion-M	$< 2.0 \times 10^{-16}$	$< 2.0 \times 10^{-16}$	$< 2.0 \times 10^{-16}$	$< 2.0 \times 10^{-16}$
	Dispersion-XS vs Dispersion-L	2.4×10^{-10}	1.6×10^{-13}	6.2×10^{-10}	2.8×10^{-10}
	Dispersion-XS vs Dispersion-A	1.0×10^{-11}	1.1×10^{-6}	1.2×10^{-8}	$< 2.7 \times 10^{-9}$
	Dispersion-S vs Dispersion-M	$< 2.0 \times 10^{-16}$	$< 2.0 \times 10^{-16}$	$< 2.0 \times 10^{-16}$	$< 2.0 \times 10^{-16}$
	Dispersion-S vs Dispersion-L	8.7×10^{-11}	3.1×10^{-14}	7.2×10^{-11}	2.1×10^{-11}
	Dispersion-S vs Dispersion-A	2.3×10^{-10}	1.4×10^{-8}	9.7×10^{-10}	4.4×10^{-10}
	Dispersion-M vs Dispersion-L	$< 2.0 \times 10^{-16}$	$< 2.0 \times 10^{-16}$	$< 2.0 \times 10^{-16}$	$< 2.0 \times 10^{-16}$
	Dispersion-M vs Dispersion-A	$< 2.0 \times 10^{-16}$	$< 2.0 \times 10^{-16}$	$< 2.0 \times 10^{-16}$	$< 2.0 \times 10^{-16}$
	Dispersion-L vs Dispersion-A	0.56	2.7×10^{-5}	0.38	0.068

4.7. Effect Size

Statistically significant differences can be obtained even if they are so small as to be of no practical value [29]. It is then important to assess whether a training dataset is statistically better than another and to assess the magnitude of the improvement. *Effect size* measures are needed to analyze this.

For a non-parametric effect size measure, we used Vargha and Delaney's $\hat{A}_{12}$ [32]. $\hat{A}_{12}$ measures the probability that running one training dataset yields higher values than running another training dataset. If the two training datasets are equivalent, then $\hat{A}_{12}$ will be 0.5.

For example, $\hat{A}_{12} = 0.7$ means that we would obtain better results in 70% of the runs with the first of the pair of training datasets that have been compared, and $\hat{A}_{12} = 0.3$ means that we would obtain better results in 70% of the runs with the second of the pair of training datasets that have been compared. Thus, we have an $\hat{A}_{12}$ value for every pair of training datasets.

Table 12 shows the values of the effect size statistics between for each pair of training datasets. **In response to RQ4 regarding density**, Density-XS obtains the best results, and Table 12 shows the superiority of this dataset in comparison to the others. Specifically, the $\hat{A}_{12}$ measures indicates that Density-XS would obtain better results than Density-S in 68% of the runs for precision, in 64% of the runs for recall, in 66% of the runs for F-measure, and in 66% of the runs for MCC. Density-XS would obtain better results than Density-M in 69% of the runs for precision, in 65% of the runs for recall, in 67% of the runs for F-measure, and in 67% of the runs for MCC. Density-XS would obtain better results than Density-L in 77% of the runs for precision, in 76% of the

Table 12: $\hat{A}_{12}$ statistic for each pair of training datasets.

		Precision	Recall	F-Measure	MCC
Density	Density-XS vs Density-S	0.68	0.64	0.66	0.66
	Density-XS vs Density-M	0.69	0.65	0.67	0.67
	Density-XS vs Density-L	0.77	0.76	0.76	0.76
	Density-XS vs Density-A	0.56	0.55	0.55	0.55
	Density-S vs Density-M	0.51	0.52	0.51	0.50
	Density-S vs Density-L	0.58	0.59	0.58	0.59
	Density-S vs Density-A	0.37	0.40	0.39	0.38
	Density-M vs Density-L	0.57	0.57	0.57	0.59
	Density-M vs Density-A	0.36	0.39	0.38	0.37
	Density-L vs Density-A	0.28	0.29	0.29	0.28
Dispersion	Dispersion-XS vs Dispersion-S	0.50	0.50	0.50	0.50
	Dispersion-XS vs Dispersion-M	0.84	0.84	0.84	0.84
	Dispersion-XS vs Dispersion-L	0.63	0.67	0.65	0.65
	Dispersion-XS vs Dispersion-A	0.60	0.59	0.60	0.60
	Dispersion-S vs Dispersion-M	0.84	0.84	0.84	0.84
	Dispersion-S vs Dispersion-L	0.63	0.67	0.65	0.66
	Dispersion-S vs Dispersion-A	0.59	0.60	0.60	0.60
	Dispersion-M vs Dispersion-L	0.28	0.27	0.27	0.26
	Dispersion-M vs Dispersion-A	0.24	0.19	0.22	0.20
	Dispersion-L vs Dispersion-A	0.46	0.41	0.45	0.44

runs for recall, in 76% of the runs for F-measure, and in 56% of the runs for MCC. Density-XS would obtain better results than Density-S in at least 55% of the runs for precision, in 55% of the runs for recall, in at least 66% of the runs for F-measure, and in at least 55% of the runs for MCC.

Table 12 shows the values of the effect size statistics for each pair of training datasets. **In response to RQ4 regarding dispersion**, Dispersion-XS and Dispersion-S obtain the best results. Moreover, according to the post hoc analysis, there are no significant differences between these datasets. Table 12 shows that they are equivalent because all of the $\hat{A}_{12}$ values for all of the indicators are equal to 0.5. The table also shows the superiority of these training datasets in comparison to the others. Specifically, the $\hat{A}_{12}$ measures indicates that Dispersion-XS and Dispersion-S would obtain better results than Dispersion-M in 84% of the runs for precision, in 84% of the runs for recall, in 84% of the runs for F-measure, and in 84% of the runs for MCC. Dispersion-XS and Dispersion-S would obtain better results than Dispersion-L in 63% of the runs for precision, in 67% of the runs for recall, in 65% of the runs for F-measure, and in at least 65% of the runs for MCC. Dispersion-XS and Dispersion-S would obtain better results than Dispersion-A in at least 59% of the runs for precision, in at least 59% of the runs for recall, in 60% of the runs for F-measure, and in

60% of the runs for MCC.

5. Discussion

This section discusses the results obtained in the evaluation through a machine learning-based approach for feature location on models. These results correspond to a single case study, which is real and industrial. Specifically, the obtained results highlight that Density-XS achieves better results than any other training datasets regarding density. There is no significant difference between the training datasets regarding multiplicity. Dispersion-XS and Dispersion-S achieve better results than the other training datasets regarding dispersion, and there is no significant difference between Dispersion-XS and Dispersion-S.

The first aspect of the results that can be discussed is the fact that none of the model fragment properties has to be balanced in order to obtain the best feature location results. The density requires extra-small values, the multiplicity does not matter, and the dispersion requires extra-small or small values. This fact can be surprising since it is conventional wisdom that an imbalanced dataset tends to have worse results. Therefore, we could hope that the training dataset with the balanced model fragment properties (Density-A, Multiplicity-A, and Dispersion-A) will obtain the best results. When this research started, three domain experts discussed whether some of three model fragment properties should be used to balance the training dataset. None of the domain experts considered sampling the knowledge base using model fragments that only have some specific property values because this means that the resulting training dataset is imbalanced for that property. However, taking into account the results, none of the three properties has to be balanced for our case study. Although this is a surprising result, it is indeed positive and beneficial for our case study, where the Ranboost algorithm can leverage a greater number many triplets from the knowledge.

The more balanced the properties must be, the lower the number of triplets that will be leveraged from the knowledge base. For example, if we decide to create a training dataset that balances the scores and the density values, the training dataset will contain a similar number of triplets for each score range (between 0 and 1, between 1 and 2, between 2 and 3, and between 3 and 4) and for each density range (between 0% and 25%, between 25% and 50%, between 50% and 75%, and between 75% and 100%). However, the knowledge base only contains 1,800 triplets with scores between 2 and 3, and only 100 of these triplets have density values between 75% and 100%. In this case, to be balanced, this training dataset will have a size around 1,600 triplets (4 scores ranges x 4 density ranges x 100 triplets). Therefore, 5,900 triplets will be discarded to obtain this balanced training dataset. If we do not have to balance the training dataset for the density values, we will be able to exploit a greater number of triplets from knowledge base.

Since none of the three properties has to be balanced for our case, we only have to balance the scores so that the sampling is affordable. Nevertheless, if several properties had to be balanced, we would have to decide whether the

improvement in the results compensates the cost of discarding a great number of triplets.

The second aspect of the results that can be discussed is the relation among the training datasets.

With regard to density, the model fragments with extra-small density values achieve results with up to 43% more precision, 41% more recall, 42% more F-measure, and 0.53 more MCC than the model fragments with other density values. However, the improvement in the results is progressive. The lower the density, the better the results are. Density-L obtains the worst result. Density-M obtains better results than Density-L. Density-S obtains better results than Density-M. Finally, Density-XS obtains better results than Density-S. In the case of Density A, since the dataset contains model fragments with all of the density values, it obtains better results than Density-L but worse results than Density XS. Table 5 and Table 6 show this progressive improvement in the results.

Note that this progressive improvement is very logical. With extra-small model fragments, we can compose a bigger model fragment by joining several extra-small model fragments like pieces of a puzzle. Regardless of the density of the model fragment that we try to find (small, medium, or large), the machine learning classifier can locate each extra-small model fragment that composes this small, medium, or large model fragment. Therefore, the training dataset with extra-small model fragments has enough information to locate any model fragment regardless of its density value. Likewise, the small model fragments can be joined to compose medium and large model fragments. In this case, the machine learning classifier may have problems locating extra-small model fragments because it is not possible to know how to divide small model fragments into extra-small model fragments. For this reason, the training dataset with small model fragments (Density-S) does not obtain better results than the training dataset with extra-small density model fragments (Density-XS). The same behaviour is shown for the results of the Density-M and Density-L training datasets.

With regard to multiplicity, the three different training datasets have no significant differences. This may be due to the definition of the property. The multiplicity property relates a model fragment and a model, determining how many times the model fragment appears in the model. In contrast, both the density property and the dispersion property involve not only the model fragment and model, but also the elements of the model fragment. The density property determines the relation between the number of elements in the model fragment with number of elements in the model. The dispersion determines the number of elements in the model fragment that are connected with each other. Therefore, since the multiplicity property does not involve the content (elements) of the model fragment, the multiplicity property may not help to differentiate a model fragment from another one. For this reason, this property may not be significant.

With regard to dispersion, the model fragments with extra-small and small dispersion values achieve results with up to 53% more precision, 52% more

recall, 52% more F-measure, and 0.57 more MCC than the model fragments with other dispersion values. Although the relation between the results is not as clear as in the case of the density property, it seems that the stronger the connection between the elements of a model fragment the easier it is to find that model fragment. In fact, four different training datasets (Dispersion-XS, Dispersion-S, Dispersion-L, and Dispersion-A) achieve good results when the approved features are strongly connected, so the approved features have values between 0 and 0.25. The same behavior is shown for the rest of the results: the less connected that the elements of the desired model fragment are, the better the results for the training datasets that contain small or extra-small dispersion values. Therefore, it may be a relation between the connected elements in a model fragment and the facility to locate the model fragment. However, future research about dispersion can help to clarify this relation.

6. Threats to Validity

In this section, we use the classification of threats to validity of [33] to acknowledge the limitations of our experiment.

Construct validity: This aspect of validity reflects the extent to which the operational measures that are studied represent what the researchers have in mind. To minimize this risk, our evaluation is performed using four measures: precision, recall, F-measure, and MCC. These measures are widely accepted in the software engineering research community.

Internal Validity: This aspect of validity is of concern when causal relations are examined. There is a risk that the factor being investigated may be affected by other neglected factors. To reduce this threat, the knowledge base of our case study is big enough to be sampled into training datasets for all of the properties: density, multiplicity, and dispersion. Therefore, the training datasets are not affected negatively because they are imbalanced. Moreover, all of the training datasets contain a similar number of triplets (about 1600), so that all of the training datasets are compared taking into account the same conditions. Furthermore, RankBoost tends to overfit when the training dataset is not large enough and there are many encoding characteristics [34]. However, this threat has been reduced because the number of triplets in our training dataset is not small in comparison to the training sets in other works [9]. Moreover, our approach uses only 54 encoding characteristics, which is a small number in machine learning applications [35, 36].

External Validity: This aspect of validity is concerned with to what extent it is possible to generalize the findings and to what extent the findings are of relevance for other cases. We reduced this threat by using standards that are frequently leveraged to specify all kinds of different software (e.g., MOF). Moreover, our experiment does not rely on the specific conditions

of our domain feature descriptions and models. However, the evaluation is limited to a single case study and a single machine learning technique. Therefore, the experiment and its results should be replicated in other domains and using other machine learning techniques before assuring their generalization.

Reliability: This aspect is concerned with to what extent the data and the analysis are dependent on the specific researchers. To reduce this threat, most of the inputs were provided by our industrial partner and the analyzed properties were selected from [8]. With regard to the inputs, the only inputs that were not provided by our industrial partner were the training datasets. These training datasets were specifically created by selecting triplets of the knowledge base provided by our industrial partner. Moreover, to prevent the researchers from influencing the results by looking for a specific outcome, all of the test cases were evaluated for all of the training datasets; none of the test cases were removed for any reason whatsoever. With regard to the analyzed properties, the evaluation is limited to three properties. These properties were proposed in [8] precisely because the researchers were not considering any property to report model fragments. However, we acknowledge that there could be other properties that may influence the results (e.g., the properties of the feature description), so the results of our evaluation are limited by these properties.

7. Related Work

Table 13 summarizes the related works that report their training datasets in order to be able to replicate the results they have obtained in their experiments. Specifically, Table 13 presents the related works that are not only related to feature location, but also to bug location and traceability links recovery, where model fragment location is fundamental. In Table 13, we highlight which properties (size, score distribution, density, multiplicity, dispersion) are taken into account as well as showing which software artifacts are considered in each work (source code or models). The $\surd$ indicates that a specific property is considered. The X indicates that a specific property is not considered. The - indicates that a specific property does not exist for this artifact. Furthermore, the last column of the table shows which works are evaluated in an industrial scenario.

Some existing works focus on machine learning for feature location. For instance, Corley et al. [2] explore the use of deep learning applied to feature location by the usage of document vectors. The authors in [37] propose a research method comprised of two experiments to evaluate five information retrieval methods targeting the extraction of feature-code trace links. Binkley et al. [6] further illustrate the benefits of using the learning to rank technique in both feature location and traceability by applying learning to rank algorithms to improve several feature models for software maintenance. Marcén et al. [5] presented a feature location approach that targets model fragments as the feature realization artifacts using learning to rank. All of these works propose new

Table 13: Overview of related work regarding the industrial evaluation and the properties of the knowledge base: size, score distribution (SD), density (DE), multiplicity (MU), and dispersion (DI).

	Related Works	Software Artifact	Properties					Industrial Evaluation
			Size	SD	DE	MU	DI	
Feature Location	[2]	Source code	√	X	-	-	-	X
	[37]	Source code	X	X	-	-	-	X
	[6]	Source code	√	X	X	X	X	X
	[5]	Models	√	√	X	X	X	√
Bug Location	[3]	Source code	√	X	-	-	-	X
	[4]	Source code	X	X	-	-	-	X
	[38]	Source code	√	X	-	-	-	X
	[39]	Source code	√	√	-	-	-	X
Traceability Links Recovery	[6]	Source code	√	X	X	X	X	X
	[40]	Source code and Models	√	√	X	X	X	X
	[7]	Source code and Models	√	√	X	X	X	X
	[41]	Models	√	X	X	X	X	X
	[42]	Models	√	X	X	X	X	X
	[21]	Models	√	√	X	X	X	√
Our work		Models	√	√	√	√	√	√

approaches that use machine learning technique for feature location, reporting algorithms, and how to tune them. Our work focuses on reporting the importance of the model fragment properties when we are training the classifier from a training dataset with model fragments.

Several research studies apply machine learning approaches to bug location. For instance, Tien-Duy et al. [3] focus on learning to rank through feature vectors that are based on likely invariants. Ye et al. [4] propose a learning to rank approach for information retrieval (IR) based bug localization using features extracted from textual bug reports and source code files. In [38], eight learning to rank techniques in bug localization are compared. The features are selected from previous hybrid bug localization studies, and the feature weight values in learning to rank techniques are learned from historical bug data and source code information. Zhao et al. [39] defined three effort-aware metrics, which are all based on lines of code, to examine the actual performance of learning to rank bug localization and claimed that the learning to rank method is similar or even worse than the standard vector support machine (VSM). All of these work focus on locating bugs by machine learning at the code level in contrast to our work, which focuses on the model level.

Recent years have seen an increasing interest in traceability-based machine

learning approaches. Binkley et al. [6] further illustrate the benefits of using the learning to rank technique in the context of traceability by applying learning to rank algorithms to improve several feature models for software maintenance. In [40], the authors used machine learning classifiers to estimate the number of valid links remaining in a set of candidate links returned by IR techniques. Sherba et al. [41] proposed an approach, TraceM, based on a technique from open-hypermedia and information integration. TraceM manages traceability links between requirements and architecture. TraceM enables the creation, maintenance, and viewing of traceability relationships in tools that software professionals use on a daily basis. In [42], the author proposes a tool for managing system requirements, system architectures, and the traceability between them. The tool involves an underlying information model that captures the key concepts and relationships of requirements engineering and architecture design. More recently, in [21], the authors propose an evolutionary ontological encoding approach to enable machine learning techniques to be used to perform Software Engineering tasks in models. Furthermore, Mills et al. [7] propose TRAIL, a technique for automating traceability maintenance by considering traceability link recovery as a binary classification problem. They address this problem using machine learning algorithms trained on historical traceability information. In contrast, our work has been evaluated in an industrial domain with software product line engineers suggesting feature properties to be considered during classifier training.

In summary, our work differs from the previous ones in four aspects: 1) we do not propose a new approach, but rather we focus on assessing the impact of feature properties in the training of the classifier; 2) we evaluate our work in an industrial domain; 3) we focus our efforts in a model-based approach; and 4) we are the only ones who have carefully evaluated the impact of feature properties during classifier training on the quality of the results.

8. Conclusions

Feature location on models involves specific properties, which may not be relevant in other domains. Although most of the works report the machine learning techniques, the tuning parameters, and the size and score distribution of the training datasets, they do not discuss the model fragment properties. In models, the training datasets contain model fragments, whose properties (density, multiplicity, and dispersion) may or may not influence the feature location results. Therefore, since model fragments form part of the training datasets, they should be properly reported in case they have an impact on the feature location results.

In this paper, we have analyzed the influence of three model fragment properties: density, multiplicity and dispersion. After the evaluation in an industrial case study, our results show that the density and dispersion properties significantly influence the results. In contrast, the multiplicity property does not influence the results regardless of the multiplicity values of the model fragments. According to the results, when the training dataset of our case study is

reported, we would have to describe its model fragments regarding the density and dispersion properties. Likewise, works on machine learning-based feature location on models should also analyze the influence of model fragment properties on their case studies not only to properly report but also to be able to compare the approaches fairly and thus improve the feature location results of their case studies.

The promising results of this work lead to interesting research questions for the future, such as the following: *Can the same results be obtained through another machine learning-based approach?*; *Do we need other training datasets for a different machine learning technique?*; *Is an analysis of the model fragment properties worth doing for any case study?*; or *Are there more properties in the content of a training dataset that can influence the results?* Therefore, in order to answer some of these research questions and to test the external validity of the results of this work, the next steps are clear: 1) other machine learning techniques must be applied; and 2) other case studies must be tested in other domains.

Acknowledgements

This work has been partially supported by the Ministry of Economy and Competitiveness (MINECO) through the Spanish National R+D+i Plan and ERDF funds under the Project ALPS (RTI2018-096411-B-I00). We also thank the ITEA3 15010 REVaMP2 Project and ACIF/2018/171.

References

References

[1] J. Font, L. Arcega, Ø. Haugen, C. Cetina, Achieving Feature Location in Families of Models through the use of Search-Based Software Engineering, IEEE Transactions on Evolutionary Computation (2017).

[2] C. S. Corley, K. Damevski, N. A. Kraft, Exploring the Use of Deep Learning for Feature Location, in: 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE, Washington, DC, USA, 2015, pp. 556–560.

[3] T.-D. B. Le, D. Lo, C. Le Goues, L. Grunske, A learning-to-rank based fault localization approach using likely invariants, in: Proceedings of the 25th International Symposium on Software Testing and Analysis, ISSTA 2016, ACM, New York, NY, USA, 2016, pp. 177–188.

[4] X. Ye, R. Bunescu, C. Liu, Learning to rank relevant files for bug reports using domain knowledge, in: Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2014, ACM, New York, NY, USA, 2014, pp. 689–699.

[5] A. C. Marcén, J. Font, O. Pastor, C. Cetina, Towards feature location in models through a learning to rank approach, in: Proceedings of the 21st International Systems and Software Product Line Conference - Volume B, SPLC '17, ACM, New York, NY, USA, 2017, pp. 57–64.

[6] D. Binkley, D. Lawrie, Learning to rank improves ir in se, in: 2014 IEEE International Conference on Software Maintenance and Evolution, IEEE, Washington, DC, USA, 2014, pp. 441–445.

[7] C. Mills, J. Escobar-Avila, S. Haiduc, Automatic Traceability Maintenance via Machine Learning Classification, in: 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE, Washington, DC, USA, 2018, pp. 369–380.

[8] M. Ballarín, A. C. Marcén, V. Pelechano, C. Cetina, Measures to Report the Location Problem of Model Fragment Location, in: Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2018, Copenhagen, Denmark, October 14-19, 2018, ACM, New York, NY, USA, 2018, pp. 189–199.

[9] A. C. Marcén, R. Lapeña, Ó. Pastor, C. Cetina, Traceability link recovery between requirements and models using an evolutionary algorithm guided by a learning to rank algorithm: Train control and management case, Journal of Systems and Software 163 (2020) 110519.

[10] Ø. Haugen, B. Møller-Pedersen, J. Oldevik, G. K. Olsen, A. Svendsen, Adding Standardized Variability to Domain Specific Languages, in: 2008 12th International Software Product Line Conference, IEEE, Washington, DC, USA, 2008, pp. 139–148.

[11] F. Pérez, J. Font, L. Arcega, C. Cetina, Collaborative feature location in models through automatic query expansion, Automated Software Engineering 26 (1) (2019) 161–202.

[12] F. Pérez, R. Lapeña, J. Font, C. Cetina, Fragment Retrieval on Models for Model Maintenance: Applying a Multi-objective Perspective to an Industrial Case Study, Information and Software Technology 103 (2018) 188–201.

[13] X. Zhuang, B. Engel, D. Lozano-Garcia, R. Fernandez, C. Johannsen, Optimization of Training Data Required for Neuro-classification, Remote Sensing 15 (16) (1994) 3271–3277.

[14] G. M. Foody, A. Mathur, A Relative Evaluation of Multiclass Image Classification by Support Vector Machines, IEEE Transactions on geoscience and remote sensing 42 (6) (2004) 1335–1343.

[15] G. M. Foody, A. Mathur, C. Sanchez-Hernandez, D. S. Boyd, Training Set Size Requirements for the Classification of a Specific Class, Remote Sensing of Environment 104 (1) (2006) 1–14.

[16] G. M. Weiss, F. Provost, The Effect of Class Distribution on Classifier Learning, Tech. Rep. (2001).

[17] G. M. Weiss, F. Provost, Learning when Training Data are Costly: The Effect of Class Distribution on Tree Induction, Journal of artificial intelligence research 19 (2003) 315–354.

[18] M. Buda, A. Maki, M. A. Mazurowski, A Systematic Study of the Class Imbalance Problem in Convolutional Neural Networks, Neural Networks 106 (2018) 249–259.

[19] S. Baltes, P. Ralph, Sampling in software engineering research: A critical review and guidelines, arXiv preprint arXiv:2002.07764 (2020).

[20] A. Arcuri, G. Fraser, Parameter Tuning or Default Values? An Empirical Investigation in Search-Based Software Engineering, Empirical Software Engineering 18 (3) (2013) 594–623.

[21] A. C. Marcén, F. Pérez, C. Cetina, Ontological Evolutionary Encoding to Bridge Machine Learning and Conceptual Models: Approach and Industrial Evaluation, in: International Conference on Conceptual Modeling, Springer, Cham, Switzerland, 2017, pp. 491–505.

[22] R. Lapeña, J. Font, Ó. Pastor, C. Cetina, Analyzing the Impact of Natural Language Processing over Feature Location in Models, ACM SIGPLAN Notices 52 (12) (2017) 63–76.

[23] A. Shabtai, R. Moskovitch, Y. Elovici, C. Glezer, Detection of Malicious Code by Applying Machine Learning Classifiers on Static Features: A State-of-the-Art Survey, information security technical report 14 (1) (2009) 16–29.

[24] P. Refaeilzadeh, L. Tang, H. Liu, Cross-Validation, in: Encyclopedia of database systems, Springer, 2009, pp. 532–538.

[25] Q. Song, Z. Jia, M. Shepperd, S. Ying, J. Liu, A General Software Defect-Proneness Prediction Framework, IEEE Transactions on Software Engineering 37 (3) (2011) 356–370.

[26] S. D. Canuto, F. M. Belém, J. M. Almeida, M. A. Gonçalves, A Comparative Study of Learning-to-Rank Techniques for Tag Recommendation, Journal of Information and Data Management 4 (3) (2013) 453.

[27] Z. Cao, Y. Tian, T.-D. B. Le, D. Lo, Rule-Based Specification Mining Leveraging Learning to Rank, Automated Software Engineering 25 (3) (2018) 1–30.

[28] V. Dang, The Lemur Project - Wiki - RankLib, `http://sourceforge.net/p/lemur/wiki/RankLib/`, [Online; accessed April-2017] (2013).

[29] A. Arcuri, L. Briand, A Hitchhiker's Guide to Statistical Tests for Assessing Randomized Algorithms in Software Engineering, Software Testing, Verification and Reliability 24 (3) (2014) 219–250.

[30] S. García, A. Fernández, J. Luengo, F. Herrera, Advanced Nonparametric Tests for Multiple Comparisons in the Design of Experiments in Computational Intelligence and Data Mining: Experimental Analysis of Power, Information Sciences 180 (10) (2010) 2044–2064.

[31] W. Conover, Practical nonparametric statistics, Wiley series in probability and statistics: Applied probability and statistics, Wiley, New Jersey, NJ, USA, 1999.

[32] A. Vargha, H. D. Delaney, A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and Wong, Journal of Educational and Behavioral Statistics 25 (2) (2000) 101–132.

[33] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering, Springer Science & Business Media, 2012.

[34] L. Wolf, I. Martin, Robust Boosting for Learning from Few Examples, in: Computer Vision and Pattern Recognition, Vol. 1, IEEE, 2005, pp. 359–364.

[35] Z.-H. Zhou, J. Feng, Deep Forest: Towards an Alternative to Deep Neural Networks, arXiv preprint arXiv:1702.08835 (2017).

[36] J. Wang, P. Zhao, S. C. Hoi, R. Jin, Online Feature Selection and its Applications, IEEE Transactions on Knowledge and Data Engineering 26 (3) (2014) 698–710.

[37] T. Vale, E. Santana de Almeida, Experimenting with information retrieval methods in the recovery of feature-code spl traces, Empirical Software Engineering (11 2018). doi:10.1007/s10664-018-9652-3.

[38] Z. Shi, J. Keung, K. Bennin, X. Zhang, Comparing learning to rank techniques in hybrid bug localization, Applied Soft Computing 62 (11 2017).

[39] F. Zhao, Y. Tang, Y. Yang, H. Lu, Y. Zhou, B. Xu, Is learning-to-rank cost-effective in recommending relevant files for bug localization?, in: 2015 IEEE International Conference on Software Quality, Reliability and Security, 2015, pp. 298–303.

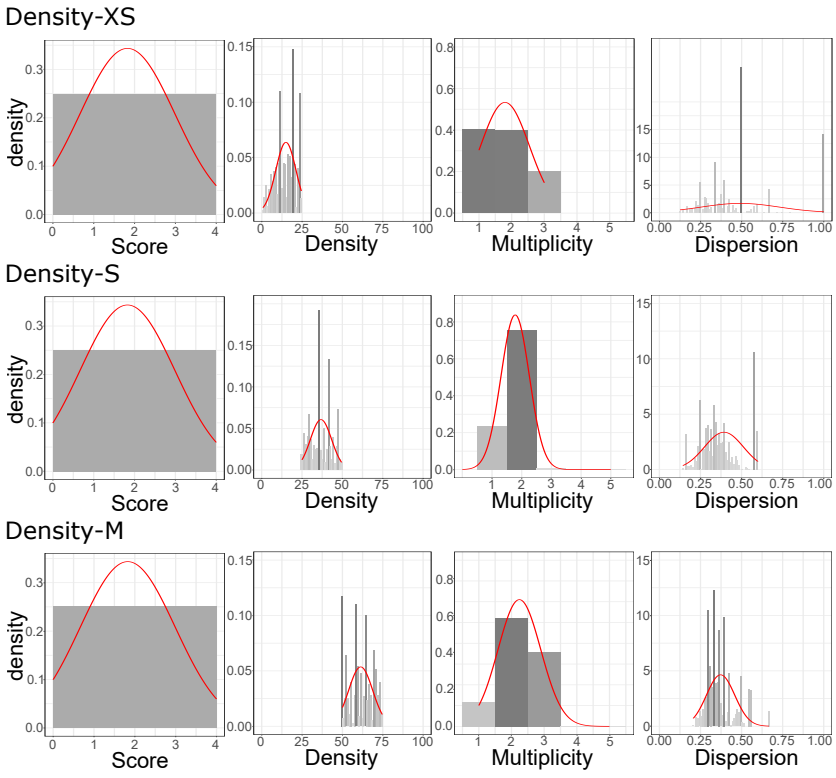
[40] D. Falessi, M. Di Penta, G. Canfora, G. Cantone, Estimating the Number of Remaining Links in Traceability Recovery, Empirical Software Engineering 22 (3) (2017) 996–1027.

[41] S. A. Sherba, K. M. Anderson, A framework for managing traceability relationships between requirements and architectures., in: STRAW, 2003, pp. 150–156.

[42] J. Han, TRAM: A tool for requirements and architecture management, Proceedings - 24th Australasian Computer Science Conference, ACSC 2001 (2001) 60–68doi:10.1109/ACSC.2001.906624.

Appendix A. Histograms for training datasets

Figure A.10: Histograms of the properties (Score, Density, Multiplicity, and Dispersion) in the training datasets (Density-XS, Density-S, and Density-M) related to RQ1.



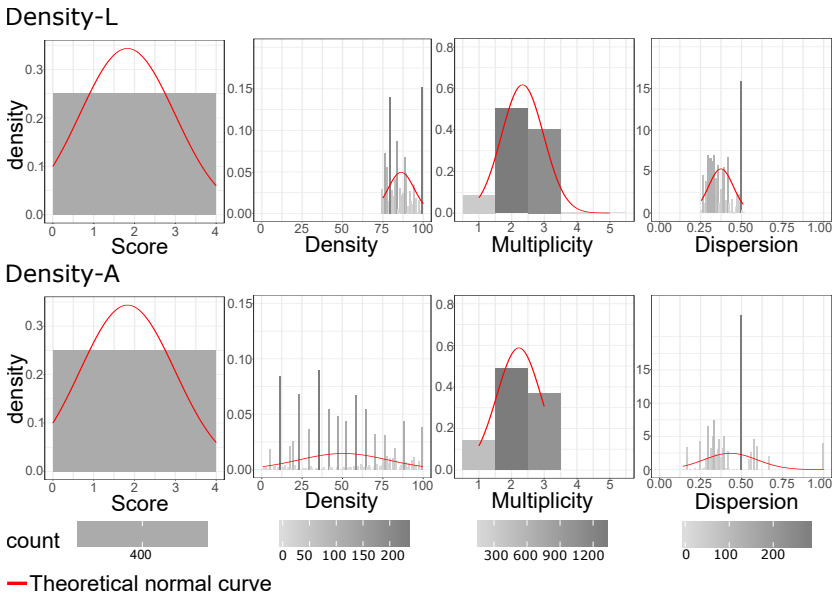


Figure A.11: Histograms of the properties (Score, Density, Multiplicity, and Dispersion) in the training datasets (Multiplicty=1, Multiplicity>1, and Multiplicity-A) related to RQ2.

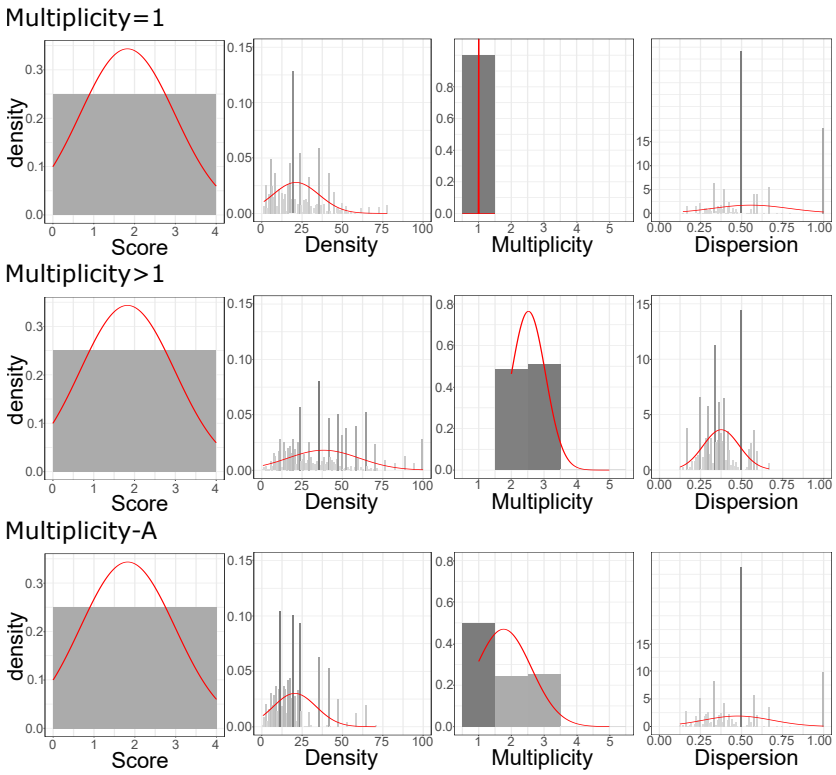


Figure A.12: Histograms of the properties (Score, Density, Multiplicity, and Dispersion) in the training datasets (Dispersion-XS, Dispersion-S, Dispersion-M, Dispersion-L, and Dispersion-A) related to RQ3.