An Attribute Selection For Severity Level Determination According To The Support Vector Machine Classification Result

Ghaluh Indah Permata Sari
Department of Informatic Engineering
Nusantara PGRI University
Kediri, Indonesia
ghaluh_gips@yahoo.com

Daniel Oranova Siahaan
Department of Informatic Engineering
Technology Sepuluh Nopember Institute
Surabaya, Indonesia
daniel@if.its.ac.id

***Abstract*— Determination of bug severity level is needed in fixing bug. Actually, in bug-tracking system, there is around 14 attributes used for defining a bug. But, all this time we do not know which attributes are highly influential for this.**

In this research, a new model of severity type classification using Infogain method for Bugzilla is proposed. As for the classsification process, we use Support Vector Machine, because this method is suitable in handling a massive data records. In this research, 8 bug attributes and 17.746 record of bug reports are involved.

From the result of the experiment, we recommend five attributes which can be used effectively in classifying the severity types with a minimal value of infogain 0,33 which is *component, qa_contact, summary, cc_list* and *product*. The combination of those 5 attributes resulting in 99,83% accuracy of severity types classification.

Keywords- Bug Tracking System; Severity Level Classification; TF-IDF; Infogain; SVM.

I. INTRODUCTION

Bug-tracking system is an application to improve servicing of customer satisfaction. Bugzilla, one of the bug-tracking system, is for tracking bug in mozilla's products. It had 14 attributes: summary, status, severity, resolution, assigned_to, product, component, priority, cclist_accesible, version, op_sys, reporter_accessible, qa_contact [1]. From those attributes, severity was considered important for bug report [1] because the completion of fixing bug was based on the type of bug severity. Moreover, user interpretation of the type of bug severity was must be precise. However, it was still determined by perception and estimation. Therefore, there were reccommendations of the type of bug severity to make it easier, So it would help developers to fix bugs [3].

Based on the background above, Menzies & Marcus did research on the classification of the type of bug severity. The research project called SEVERIS (Severity Issue Assessment), where the bug-tracking system which was used was a commercial robotic satellite NASA Independent Verification and Validation (NASA IV &V) [3]. SEVERIS has done giving the process of recommendations by making classification rule using the Rule Learning method. Experimental data were only taken from the comments attribute where its type data is String, so it was necessary to find the weight of term prior before it was classified [3]. The process of search terms was done by calculating tf * idf, then it was calculated by Infogain to get the effective terms for the classification process [3]. SEVERIS took two data sets from the database Pits, the total number of data is the 3875 bug [3]. The drawback of this method was for a small number of data sets [3].

In different studies, Lucas D. Panjer stated his research that the other attributes in addition to comments also affected the process of determining the age of a bug. He also stated that the attributes on the bug report was divided into two: the influential attributes and non-influential attribute on determining the age of bugs [2].

Based on the background of the two previous studies, our study is trying to make improvements in the classification process of the research conducted by Menzies & Marcus, with considering the other attributes in addition to comments (in Bugzilla called the Summary attribute) as it was done on Panjer's research, with the aim that the method type of bug severity classification can also be done for open source bug tracking system like Bugzilla to produce a high degree of accuracy where Bugzilla attribute has the number of data sets larger than the SEVERIS data sets. It was more than 17,000 bugs.

Classification method proposed for our study is the Support Vector Machine (SVM). SVM is often known as a binary classification, but it can also be used for the classification that has a lot of class. That SVM method is known as Multiclass Support Vector Machine (SVM Multiclass). This method was introduced by Chih - Wei Hsu & Chih - Jen Lin [4]. Due to the number of classes is more than two (multiclass), then made an approach to solve these problems. There are two approaches that are often done for multiclass SVM: to combine all the data in a problem; to build a multiclass classifier. Our study will use the second approach because it does not require the completion of the complex optimization and high computing, so it make easier to implement multiclass SVM [4]. Apart from that, SVM was chosen because of the way the classification is more accurate than 3 on the other classification methods with a large feature space [4]. From those advantages, SVM is considered capable for classification of the type of severity in the open-source bug tracking system like Bugzilla.

From the research experiment that was conducted by Ghaluh, Daniel, and Umi [5]. SVM proved appropriate for the classification of the type of severity in the open source bug tracking system like Bugzilla, where the large number of data sets that are used is 17,746 bugs. Apart from that, SVM is also suitable for multi-attribute classification. This is indicated by the results of the classification which reached 99.83%.

II. METHODS

This section describes the process steps in this study. It is began with preprocessing: tokenizing, filtering, and stemming. After that, performing weighting process with tf*idf, and then, ranking with infogain. The last process is doing classification.

Classification process is not only used to classify bugs according its class (severity) but is also used to test the significant attributes that influence it. The process steps in this study is shown in Figure 2.

A. Preprocessing

Preprocessing aims to search the terms which can represent the contents of a document. Thus, the analysis of connectedness between documents can be done. Preprocessing in our study consisted of three stages: tokenizing, filtering, and stemming. Figure 1 shows the flow chart of preprocessing.

Figure 1. Preprocessing stages.

Tokenizing stage is the stage of cutting the input string based on each word which arranges the input string. Filtering stage is the stage to take the essential terms of the results of tokenizing, at this stage, it used a list of stop words (removing words that are less important) or word list (saving the important word) that obtained from the website www.briandunning.com/cf/936. The last stage text mining process is stemming, searching the root word of each word of filtering results [6].

B. Term Frequency*Inverse Document Frequency (TF*IDF)

At this stage, each document is associated as a vector with the number elements a lot of as the successful terms recognized from the extraction stage of the documents produced in the previous text mining. The vector consists of terms which will then be calculated based on tf * idf method. Tf * idf method is a method of weighting which is the integration between the term frequency (tf) and inverse document frequency (idf) [7]. The formula shown in Equation (1).

$$w(t,d) = tf(t,d)*log2(N/nt) \quad (1)$$

The symbol w(t,d) is the weight of the term t in document d, while tf (t,d) is the term frequency in a document (tf), where N is the total number of documents used for the calculation of idf. The nt is the number of documents that contain the value of t. The function of this method is to seek representation value from each of the terms of a collection of documents. Then it will set up a matrix of terms with documents, the type of attribute severity that contains all summary according to the its severity type.

C. Infogain

Infogain method was first introduced by Thomas J. McCabe. This method aims to measure the level of complexity of an attribute by looking for the best terms that most facilitate the target concept [8]. How it works is to do a ranking on the data that have been previously weighted. It is calculated by the formula in Equation (2).

$$\textbf{Gain } (S, A) = \textbf{Entropy } (S) - \sum_{v \in ValuesA} \frac{|S_v|}{|S|} \textbf{Entropy } (S_v) \quad (2)$$

Where A = attributes, V = possible value for atribut A, Values (A) = the set of possible values for atribut A, | Sv | = number of samples of the value of v, | S | = total number of data samples, Entropy (Sv) = Entropy for samples that have the value v. Entropy in this case is a parameter used to measure the level of diversity (heterogeneity) of the data set. The more heterogeneous the data, the higher the value of entropy. The Entropy is performed by the formula in Equation (3).

$$\textbf{Entropy } (S) = \sum_{i}^{c} -p_i log_2 p_i \quad (3)$$

Where c = number of values in the atibut target (number of classification classes), whereas pi = number of samples proportion (opportunities) for class i.

In a previous study conducted by Menzies & Marcus [3], infogain is used to determine the most informative terms, by way of re-order the existing term. This is done because the term is considered as an attribute [3]. So Menzies & Marcus used infogain to find a term trend towards a type of severity. Slightly different from our study, where each of severity type has attributes, and in these attributes contain terms. So in our study, infogain used not only looking for a tendency toward an attribute of severity type, but also seek the trend term toward attribute, before seeking a tendency of attribute toward severity type.

D. Make a Term Code

The process is conducted during forming matrices for SVM input. It is to recognize term in classification process easily. The encoding term will support classification process due to the formed code is unique. So, it can easily

distinguish the term of each of its severity. The first step is making the code form each of terms. The encoding uses 2 simple rules: the sequence number of term; the sequence number of attribute for term from non-summary attribute result. The first rule is based on the sequence of sorting term, while the second is made by the sequence: (1) Product, (2) Component, (3) Priority, (4) Version, (5) Op_sys, (6) Qa_contact, (7) Cc_list. The term from summary attribute result is just uses the sequence number of each term without uses additional code at its behind. The second step is matching the term according to the sequence of original form before the term is separated (the sequence is accroding to the bugzilla report).

E. Classification

The classification process in this study used SVM Multiclass One Against All with Gaussian kernel. The input is matrixes of encoded weighting process. One Against All method will build k number of binary SVM, where k is the number of classes [4]. SVM-i is trained with all samples in the class-i with positive label and all other samples with negative label if given l training data $(x_i, y_i),...,(x_l, y_l)$, where $x_i \in \Re^n$, $i = 1,...l$ and $y_i \in (l,...,k)$ is the lass of x_i, then the SVM – i will solve the problem in Equation (4).

$$min_{w^i b^i \xi^i} \frac{1}{2} (w^i)^T w^i + C \sum_{j=1}^{l} \xi_j^i,$$
$$(w^i)^T \Phi(x_j) + b^i \geq 1 - \xi_j^i, \text{ jika } y_j = i,$$
$$(w^i)^T \Phi(x_j) + b^i \leq -1 + \xi_j^i, \text{ jika } y_j \neq i,$$
$$\xi_j^i \geq 0, j = 1, ..., l. \quad (4)$$

Where the data x_i is mapped into the higher dimensional space using the function Φ and C as the penalty parameter.

Minimizin g $\frac{1}{2}(w^i)^t w^i$ means maximizing $\frac{2}{|w|^2}$ or the margin beetwen the two groups of data. When the data are not separated by linear, then there is the penalty of C $\sum_{j=1}^{l} \xi_j^i$ which can reduce the amount of training error. The idea of SVM is to balance the regulation $\frac{1}{2}(w^i)^t w^i$ and training error.

After finished minimization problem, then there are k decision functions shown in Equation (5).

$$f^1(x) = (w^1)\, x + b^1, ..., f^k(x) = (w^k)\, x + b^k \quad (5)$$

In this study, x data class will be determined based on the highest decision function value.The functions for searching minimization solution has been provided in the function of quadratic programming that will be implemented with a monqp function in matlab.

Classification process is done in two processes: training and testing process. In the training process, hyperplane variable of each classifier will be recorded and be used as a classifier in the training process. If a class of training process same with the one of testing process, then the recognition is correct. Its result is a matrix of weight term that corresponding to the index value of the biggest decision function of testing process.

The testing process is intended to prove the counting infogain result in the important attribute selection. It is conducted with two testing scenarios that will be described one by one with the diagram. Each of scenarios is conducted with varying the number of attributes. The first scenario is done by varying the number of atrributes in the testing process, while in the training process still used 8 attributes. The second scenario is done by varying the number of attributes in the training and testing process. So, the number of attributes of training and testing process are same. Each of scenarios is conducted in 3 steps: measuring accuracy of summary attribute; measuring accuracy of qa_contact, component, product, cc_list, and summary attribute; measuring accuracy of version, op_sys, and priority attribute.

III. EXPERIMENTS

This explain the processes of determination of Important Attributes. This process is done with two test scenarios. Each of scenarios is made by performing a variation on the number of used attributes. Scenario 1 is done by varying the number of attributes in the testing process, while in the training process still using 8 attributes. Scenario 2 is done by varying the number of attributes in the process of training and testing process. So the number of attributes of the process of training and testing in scenario 2 is the same. Each of scenarios is performed in 3 steps: measurement accuracy for only summary attribute; measurement accuracy for attributes with a high infogain value (qa_contact, component, product, cc_list and summary); measurement accuracy for the attribute with a a low infogain value (version, op_sys, and priority). In scenario 1 and scenario 2, the numbers of data used for training are 16,146 bugs, while for the testing are 1,600 bugs. The number of term per severity for training are 666 terms, while for the testing process are 200 terms. Process of scenario 1 is shown in Figure 3, while the process of Scenario 2 is shown in Figure 4.

IV. EXPERIMENTAL RESULT

From testing of important attributes with performing scenario 1 and scenario 2, the conclusion drawn that the results can be seen in Table 1.

TABLE I. TESTING RESULT WITH SCENARIO 1 AND SCENARIO 2

Testing	Scenario 1	Scenario 2
Step 1	16,97	16,67
Step 2	16,97	98
Step 3	16,97	Bad Mouve

. The testing result of step 1, 2, and 3 in scenario 1 is equal to 16.97%, this is because the number of input attributes of the training process and testing process is not the same. So the matching process of data classifier with testing input did not have many partners. This caused a lot of bad mouve conditions, the lack of a proper support vector in the testing process.

The testing result of scenario 2 looks more varied, this is because the number of input attributes in the process of training and testing are the same. So, the matching process between the test data with the classifier has a lot of couples. It Seen from Table 1, that the test results in step 1 yields an accuracy rate of 16.67%. This shows that, if only the summary attributes are used in the classification process, is not sufficient to obtain high accuracy for classification of severity type.

The testing result of step 2 in scenario 2 shows the accuracy rate of 98%. This result indicates that the five attributes that have the top 5 value infogain: summary, component, product, qa_contact, cc_list can very significantly affect the accuracy of classification result.

The last discussion on the result of step 3 in scenario 2, where the testing of step 3 did not get the level of accuracy. This also was occured in the process of training. In the testing step 3 was dominated by bad mouve condition since the beginning of the testing process. From the result in Table 1 can be concluded that the three attributes that have the three lowest value infogain: version, op_sys, priority did not significant effect on the classification process.

From the discussion of the results of the scenario method that has been done, one conclusion can be drawn again that in our study, in order to improve the accuracy of classification results on the type of severity bugzilla, the lowest limit infogain value of an attribute is 0.33. Apart from that, if there are attributes that have values below 0.33 infogain, can be expressed as an attribute that is not effective to increase the accuracy of classification results on the type of severity bugzilla.

In our study, the lower limit infogain value of the attribute effectively determined after the testing is done, this is because there is no specific benchmark to determine how large the lower limit infogain value of an attribute that can be declared effective to improve the accuracy of classification results.

V. CONCLUSIONS

The purpose of the series of studies conducted is to choose a significant attribute for the classification process. For that, it was performed a series of experiments that have been implemented and documented in Methods, which generate a sequence of important attributes according infogain results and verification results by the classification according to the order of the results of these infogain. From those the verification process, it can be concluded as following below:

1. The number of terms did not have significant effect on the level of accuracy, but the unique terms can raise the level of classification accuracy. The more unique terms as data classification, the higher the level of accuracy of the classification process. moreover, the unique terms can help reducing the time required in the classification process. This leads to more easily find a classifier which divides each of class because machine learning only find a single term in each class.
2. The process to determine the terms on the bug report attributes are in 2 ways: tf * idf and counting chance occurrence. This is because the data of each attribute is different. Tf * idf is used for the data type in the summary attribute. The calculation probabilty of occurrence is used for the data type on the attributes of component, product, qa_contact, summary, op_sys, vesion, priority.

Attributes that can help improve the accuracy for the classification process of severity type are qa_contact with a value of 0.97, component with a value of 0.91, summary with a value of 0.47, and product with a value of 0.33. This was showed by the results of the classification accuracy of four attributes by 98%.

REFERENCES

[1] Vlasceanu, Ion Valentin., Christian Bac. 2009. " Study Concerning The Bug Tracking Applications ".

[2] Panjer, Lucas D. 2007. " Predicting Eclipse Bug Lifetimes ". IEEE 29th International Conference on Software Engineering Workshops, 0-7695-2830-9/07.

[3] Menzies, Tim., Andrian Marcus. 2008. "Automated Severity Assessment of Software Defect Reports". IEEE, 346-355.

[4] Hsu, Chih-Wei., Chih-Jen Lin. 2002. "A Comparison Methods for Multi-class Support Vector Machines". IEEE Transactions on Neural Networks, Vol. 13, No.2, 415-425.

[5] Sari, Ghaluh I. P., Daniel O. S., and Umi L. Y. 2011. "Klasifikasi Bug Untuk Menentukan Tingkat Severity Menggunakan Support Vector Machine". Jurnal Teknik Informatika, Vol. 2.

[6] Mooney, Raymond J., Loriene Roy. 1999. "Content-Based Book Recommending Using Learning for Text Categorization". The SIGIR-99 Workshop on Recommender Systems : Algorithms and Evaluation. Berkley.

[7] Yang, Yiming, Jaime G. Carbonell, Rulf D. Brown, Brian T. Achibald, Xin Liu. 1999. " Learning Approaches For Detecting and Tracking News Events ". IEEE Intelligent Systems, Language Technologies Institute, Carbeige Mellon University.

[8] McCabe, J. Thomas. 1976. " A Complexity Measure ". IEEE Transactions on Software Engineering, Vol. SE – 2, No. 4.

[9] http://www.bugzilla.mozilla.org

[10] http://tartarus.org/~martin/PorterStemmer

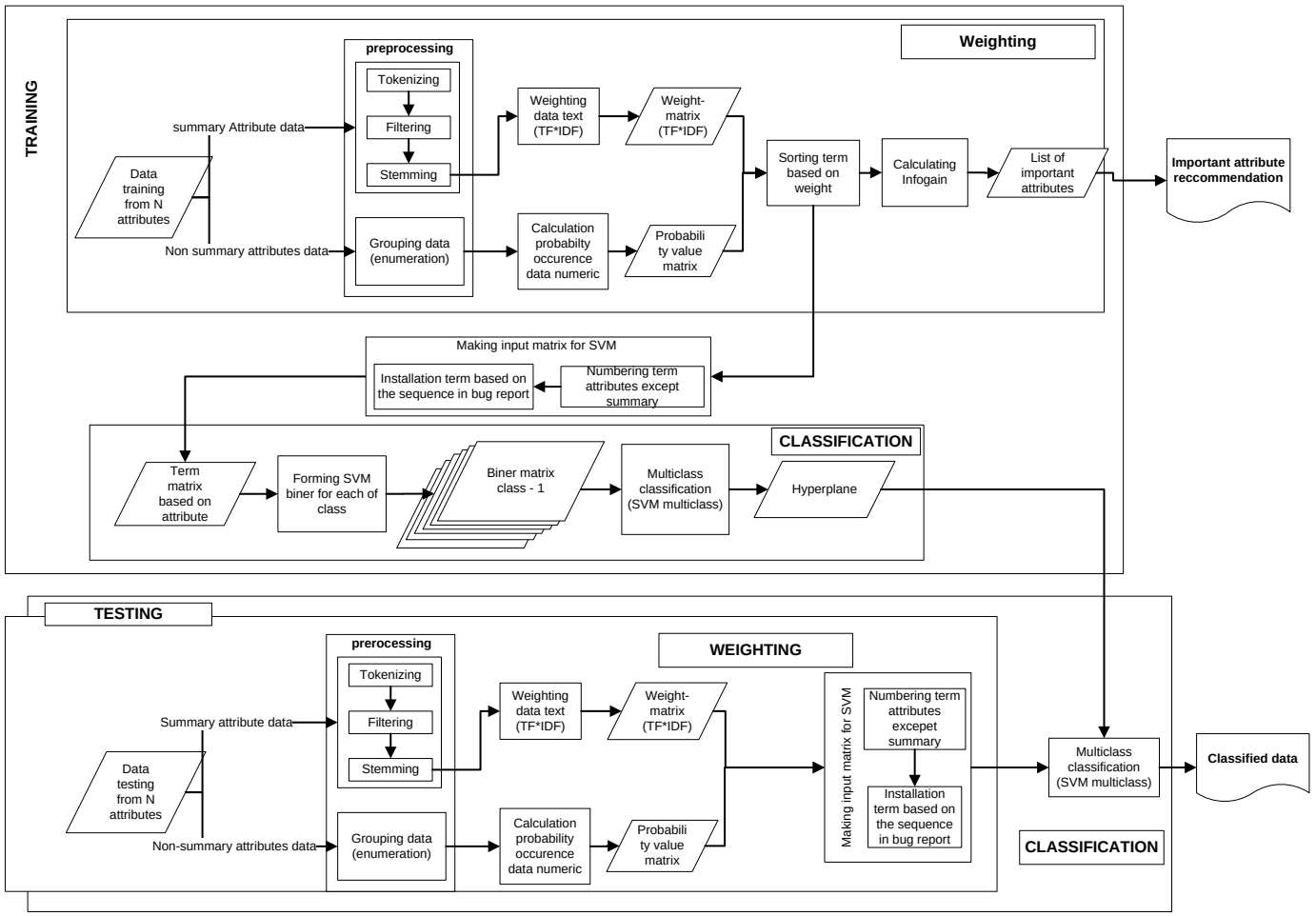

Figure 2. Design of an attribute selection for severity classification.

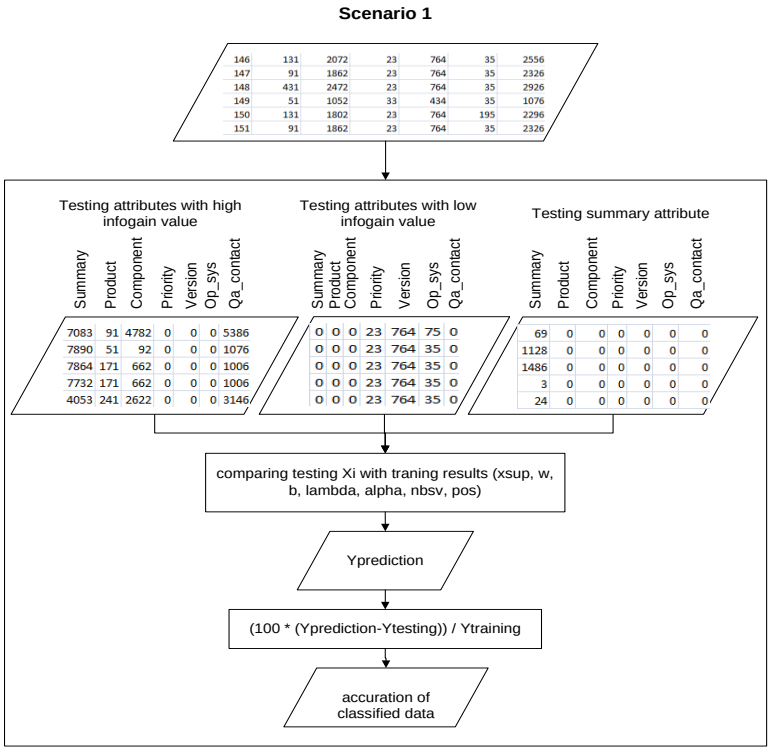

Figure 3. Testing process scenario 1.

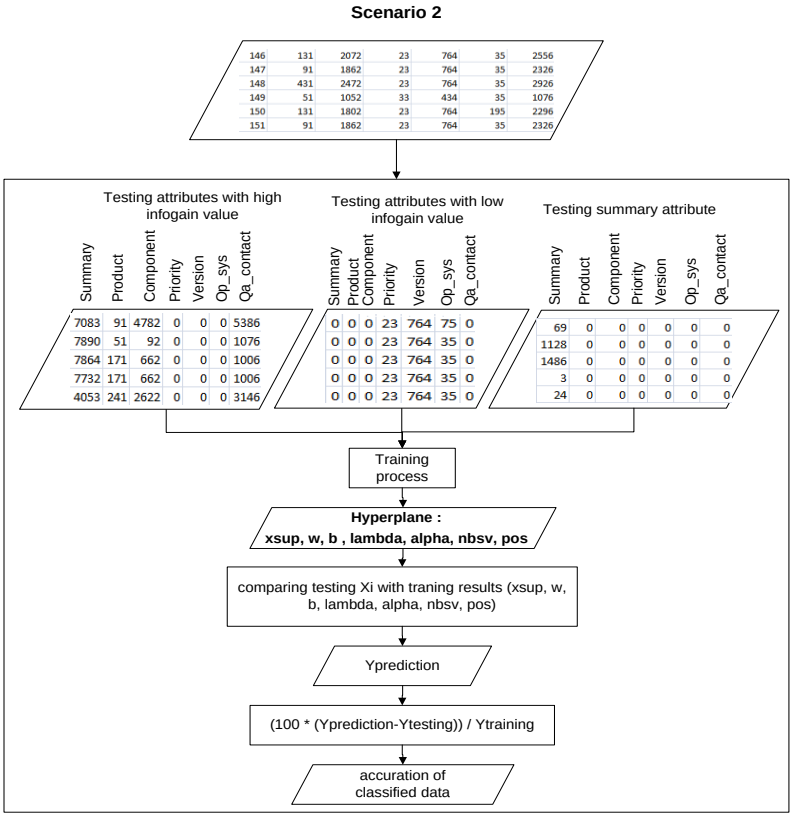

Figure 4. Testing process scenario 2.