COMPILER
Vol. 14, No. 1, May 2025, pp.33-42
ISSN: 2252-2403 (Print), ISSN: 2549-2403 (Online)
Aaccredited Fourth Grade by Kemenristekdikti, Decree No:200/M/KPT/2020
DOI: 10.28989/compiler.v14i1.2979

Software Defects Predictions using SQL Complexity and Naïve Bayes

Made Agus Putra Subali[1,*], I Gusti Rai Agung Sugiartha[2], I Made Budi Adnyana[3]
I Putu Aditya Putra[4], Made Dai Subawa[5]
[1-5]Institut Teknologi dan Bisnis STIKOM Bali, Indonesia

Article Info

Article history:

Received May 4, 2025
Accepted May 30, 2025
Published June 13, 2025

Keywords:

Software Defects Predictions
SQL Complexity
Naïve Bayes
NASA MDP Datasets
SQL Commands

ABSTRACT

Software defects result in unreliable software, therefore predicting software defects is an effort to produce quality software. In this study, we used the naïve bayes method because it has the appropriate characteristics of the data used. The data used include NASA MDP datasets and datasets from the calculation of the SQL complexity method on eight software modules. The use of two datasets was carried out because in the NASA MDP datasets there were no attributes that paid attention to the use of SQL commands, therefore in the datasets from the eight software modules the SQL complexity attribute was included which paid attention to the level of complexity of the use of SQL commands in each module. The prediction results of this study were evaluated by considering the values of accuracy, precision, recall, and f-measure. Based on these results, the accuracy results of CM1 were 88%, PC2 was 97%, and KC3 was 78%, meanwhile, for the data of the eight software modules used, an accuracy result of 67% was obtained.

Corresponding Author:

Made Agus Putra Subali,
Institut Teknologi dan Bisnis STIKOM Bali, Indonesia,
Jl. Raya Puputan No.86, Dangin Puri Klod, Denpasar, Bali 80234.
Email: *madeagusputrasubali@gmail.com

1. INTRODUCTION

Software defects or defects in software are errors or bugs that occur in software [1]. The types of defects that often occur in software are defects in program code, user interface, documentation, performance, and security [2]. Currently, software has a high size and level of complexity [3]. Defects or defects in software make the software unreliable [4], therefore being able to predict defects in software is an effort to produce quality software [5].

Previous research related to software defects was conducted using various approaches, including research using the RARM method, which has advantages in finding data combinations so that prediction results can be more accurate, but has disadvantages if the combination opportunities for the items being searched require more time because they require various combinations, so that the level of accuracy provided can be lower [6]. Other research using the SVM method, has advantages in a better level of accuracy for data characteristics that have two classes, but has disadvantages in longer processing times because it requires defining hyperplanes and kernels so that items can be predicted [7]. In research using the NN method, has advantages that can be applied to more varied data characteristics but has disadvantages in the complexity of applying the method [8]. In previous studies, the naïve Bayes method was used to classify the probability of stunting rates with an accuracy level of up to 75% [9]. Several variations of the naïve Bayes method in conducting sentiment analysis on datasets obtained from the Indonesian language Twitter social media showed good results [10]. In other studies, the naïve Bayes method optimization was carried out using the genetic algorithm and the bagging method showed an increase in accuracy of 4.57% in the case of credit risk analysis [11]. Based on the results of previous studies, in this study, we used the naïve Bayes method because the naïve bayes method is very appropriate when applied to the characteristics of the data used, namely data with two classes and the naïve bayes method is easier and simpler to implement with more accurate accuracy [12].

The data used in this study consists of two types of data, namely: (1) NASA MDP (Metrics Data Program) software defect datasets on CM1, PC2, and KC3 projects, and (2) datasets collected independently

from eight software modules. The use of two datasets was carried out because in the NASA MDP dataset, no attribute paid attention to the use of SQL commands, therefore in the datasets from the eight software modules collected, the SQL complexity attribute was included which paid attention to the level of complexity of the use of SQL commands in each module [13]. Contributions to this study include the use of the SQL complexity attribute in predicting software defects using the naïve Bayes method.

2. RESEARCH METHOD

Figure 1 shows the proposed method of the research carried out, starting from data collection, preprocessing, classification using the naïve Bayes method, and evaluating the results.

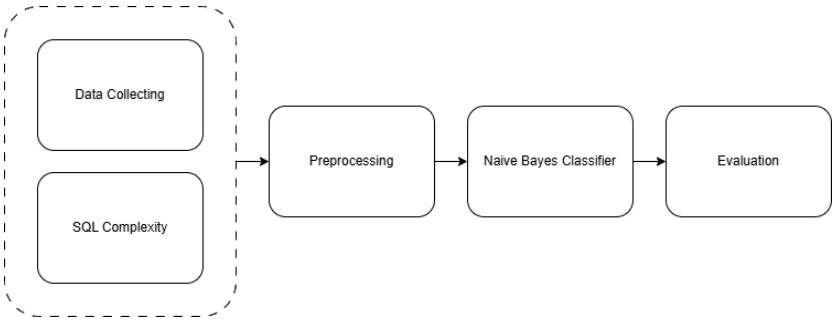

Figure 1. Proposed Method

2.1. Data Collecting

The data used were obtained from two different types of data, the first is data from NASA MDP software defect datasets and the second is data collected independently from eight software modules. Table 1 shows the characteristics of the NASA MDP data used in this study, including projects CM1, PC2 and KC3. The selection of these three projects was based on the percentage of defects they had, CM1 (12.20%) on average, PC2 (1.01%) at minimum, and KC3 (18%) at maximum [14].

Table 2 shows the characteristics and attributes used for the second data collected independently, if you pay attention to the types of attributes used, they include the results of sloc, cc, and SQL complexity calculations from the eight modules and there is a defect label on each module. The use of the SQL complexity method is carried out to add attributes that pay attention to the use of SQL commands in each module [13].

The SQL complexity method is a method for measuring software complexity that takes into account the use of SQL commands or queries. This method consists of five stages: (1) analysis of SQL commands in program modules, (2) modelling of SQL commands, (3) weighting of SQL commands, (4) calculation of SQL complexity values, and (5) analysis of complexity calculation results [13]. Figure 2 shows the flow of the SQL complexity method stages.

Table 1. NASA MDP Software Defect Datasets

No.	Project	Description	Total Module	Defect Module
1	CM1	Spacecraft instrument	344	42
2	PC2	Dynamic simulator for attitude control system	1585	16
3	KC3	Storage management for ground data	200	36

Table 2. Characteristics and Attributes of the Eight Modules

No.	Module Name	SLOC	CC	SQL Complexity	Defect Label
1	*Menjawab Kuis*	170	18	3,35	Yes
2	*Update Biodata*	120	14	1,2	None
3	*Posting Soal*	158	17	3,65	None
4	*Menilai Kuis*	180	19	10,55	Yes
5	*Input Kelas*	121	14	2,25	None
6	*Input Siswa*	134	14	2,85	None
7	*Input Guru*	161	19	3,45	None
8	*Input Mapel*	154	16	3,65	None

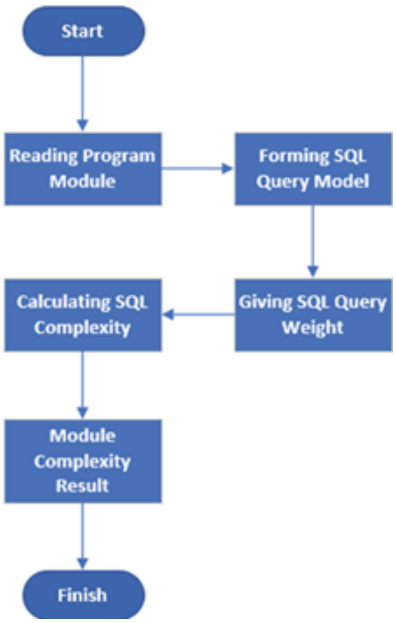

Figure 2. SQL Complexity Method [13]

The calculation of SQL complexity value in the fourth stage is calculated using equation (1).

$$SC = \sum_{i=0}^{n} x_i \cdot w_i \quad (1)$$

Description:
n is the total sql query attribute.
x_i is the number of SQL query attribute i.
w_i is the weight of SQL query attribute i.

The calculation process of the SQL complexity method starts from the reading program module stage in the SQL query whose complexity is measured, for example, the SQL query to update data as seen in Figure 3. After the SQL query is determined, the process continues to the second stage of forming the SQL query model, at this stage the value of each query attribute is obtained, including: output variable, input variables, nested queries, join tables, and the number of tables to better understand the value of each query attribute, visualization can be done using a tree diagram, as seen in Figure 4. The next stage is giving SQL query weight to each attribute according to the value rules in Table 3. After the attribute values and weights are obtained, the next step is the SQL complexity calculation process using equation (1) with the results that can be seen in Table 4. The final stage is carried out to evaluate the results of the SQL complexity calculation using the value rules in Table 5.

```
Query = "UPDATE " +
        "T1 " +
        "SET " +
        "V1 = '" + N1 + "', " +
        "V2 = '" + N2 + "', " +
        "V3 = '" + N3 + "', " +
        "V4 = '" + N4 + "', " +
        "V5 = '" + N5 + "', " +
        "V6 = '" + N6 + "', " +
        "V7 = '" + N7 + "', " +
        "V8 = '" + N8 + "', " +
        "V9 = '" + N9 + "', " +
        "V10 = '" + N10 + "', " +
        "V11 = '" + N11 + "', " +
        "V12 = '" + N12 + "', " +
        "V13 = '" + N13 + "' " +
        "WHERE " +
        "V14 = '" + N14 + "'";
```

Figure 3. Query Update Data [13]

Figure 4. Query Model Update Data [13]

Table 3. SQL Query Attribute Values

No.	Query Attribute *x*	Weight *w*
1	Variable Output	0,10
2	Variable Input	0,15
3	Nested Query	0,20
4	Join Table	0,25
5	Number of Table	0,30

Table 4. SQL Complexity Measurement Update Data

No.	Query Attribute *x*	Weight *w*	x_i	$x_i \cdot w_i$
1	Variable Output	0,10	0	0
2	Variable Input	0,15	14	2,10
3	Nested Query	0,20	0	0
4	Join Table	0,25	0	0
5	Number of Table	0,30	1	0,30
			SQL Complexity	**2,40**

Table 5. Complexity Rating

No.	Score	Rating
1	1-4	Very Low
2	5-10	Low
3	11-20	Normal
4	21-40	High
5	41-50	Very High
6	>50	Extra High

2.2. Preprocessing

After the data is obtained, the next stage is to preprocess the data with the aim that the research data used can be relied on so that the analysis results can be more accurate and effective [15]. The data preprocessing stages carried out include: data cleaning and data normalization. Figure 5 is the process flow of the data preprocessing stages carried out, if observed at the data cleaning stage, incomplete and inconsistent data are cleaned, while at the data normalization stage, data is limited to a certain range [16]. Data preprocessing is only carried out on NASA MDP datasets because there is incomplete, inconsistent, and non-normal data [17].

Figure 5. Data Preprocessing

2.3. Naïve Bayes Classifier

After the data is obtained and the data preprocessing is carried out, the classification process is carried out using the naïve bayes method for each data used. The initial process is carried out by breaking the data into two types of sizes, namely 70% training data and 30% testing data, initializing the gaussian naïve bayes model, training the model with training data, making predictions using the previously trained model, and finally the evaluation process is carried out.

The naïve bayes method is a simple, fast, accurate, and reliable supervised learning method on datasets that have two-class characteristics [12]. The naïve bayes method is widely applied based on several of its simple properties [18]. The naïve bayes method predicts data based on the probability P of the x attribute of each class y of the data [19]. Equation (2) is how the naïve bayes method predicts data based on probability.

$$P(y_k|x_1, x_2, \dots x_a) \quad (2)$$

Description:
P is the probability.
y_k is each class.
x_a is the attribute.

In equation (3) is the naïve bayes calculation, namely the probability of the appearance of attribute x_a in class category y_k multiplied by the probability $P(y_k)$ then divided by the probability of the appearance of attribute $P(x_a)$.

$$P(y_k|x_a) = \frac{P(y_k)P(x_a|y_k)}{P(x_a)} \quad (3)$$

2.4. Evaluation

At the evaluation stage, a confusion matrix is used to present the truth of a prediction [20]. Figure 6 shows a confusion matrix table.

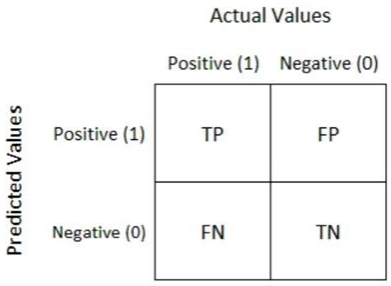

Figure 6. Confusion Matrix Table

Description:
True Positive (TP) is predicted positive and the result is correct.
False Positive (FP) is predicted positive and the result is incorrect.
False Negative (FN) is predicted negative and the result is incorrect.
True Negative (TN) is predicted negative and the result is correct.

The classification accuracy results are calculated using equations (4)-(7), where each value in the accuracy calculation is obtained from the confusion matrix model.

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \quad (4)$$

$$Precision = \frac{TP}{TP+FP} \quad (5)$$

$$Recall = \frac{TP}{TP+FN} \quad (6)$$

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision+Recall} \quad (7)$$

3. RESULTS AND ANALYSIS

3.1. Data Collecting

The datasets used in this study consist of two types of data, namely NASA MDP software defect datasets and eight independently collected software modules.

1) NASA MDP Software Defect

- CM1 can be viewed at https://intip.in/cmone
- PC2 can be seen at https://intip.in/pctwo
- KC3 can be seen at https://intip.in/kcthree

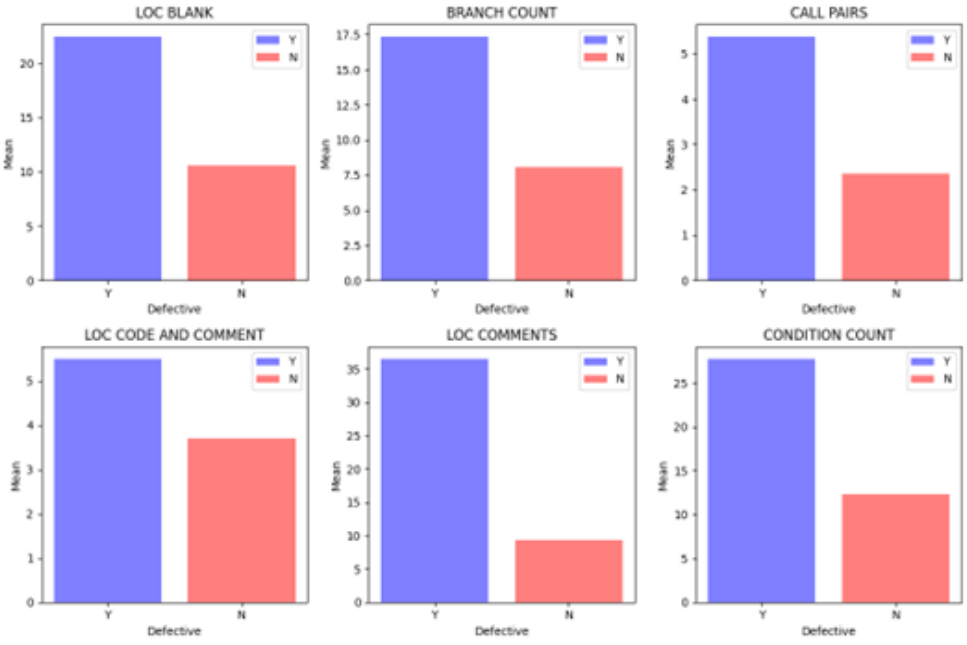

Figure 7. Six Sample Attribute and Label Information on CM1

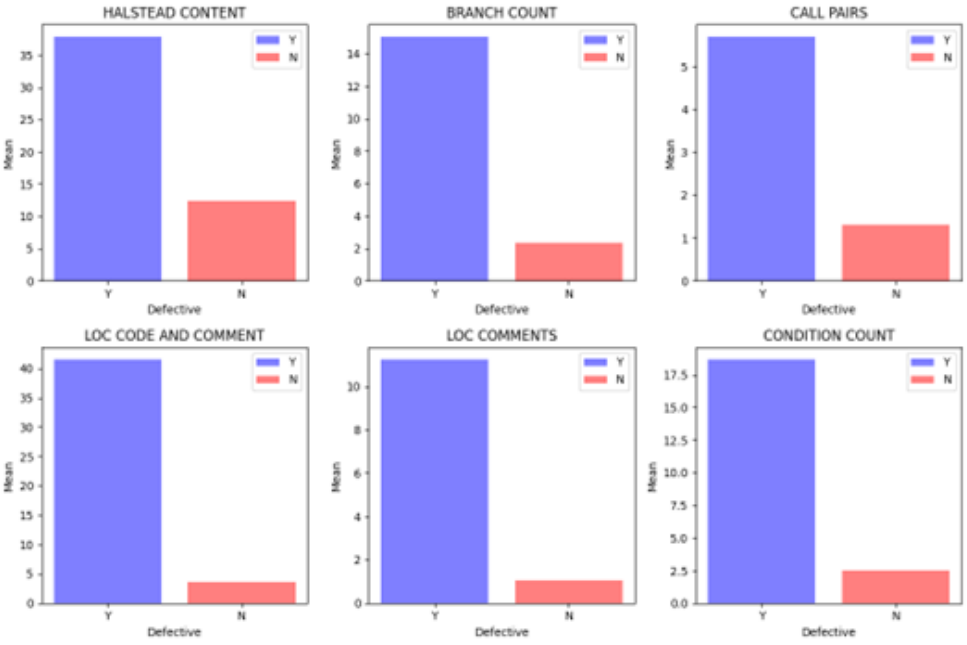

Figure 8. Six Sample Attribute and Label Information on PC2

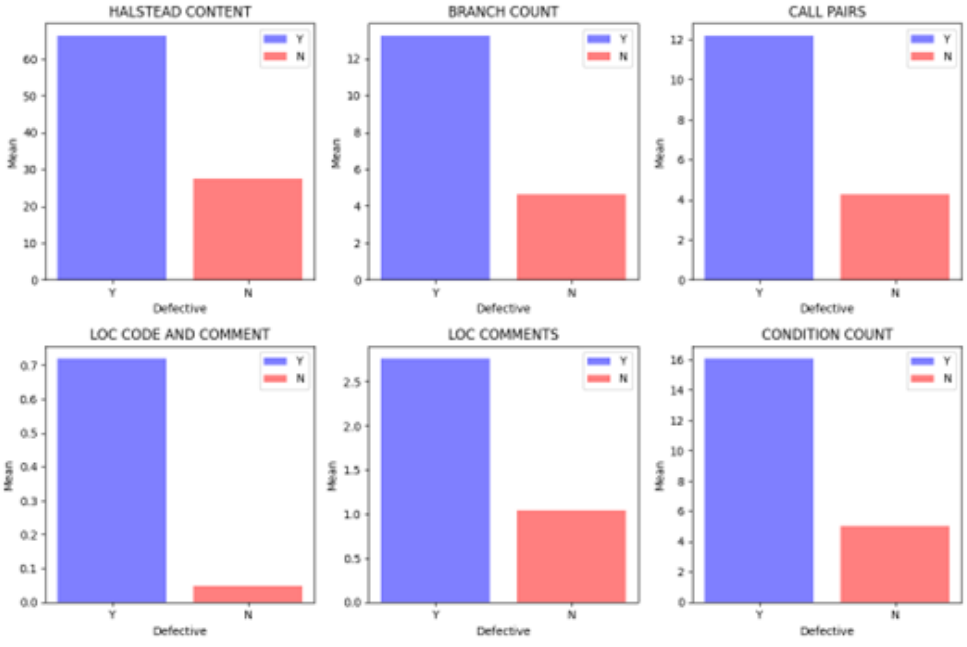

Figure 9. Six Sample Attribute and Label Information on KC3

Based on the data obtained, the information provided in Figure 7 to Figure 9 can be seen in several attribute samples used, the average attribute value in each software module has a tendency for defects to occur at higher values compared to non-defects with a smaller average number.

2) Eight Software Modules

Table 6 is the data from the eight software modules used, there are three attributes used in each module, namely: sloc, cc, SQL complexity. If observed each module does not have a defective label, therefore it is given label as seen in Table 2. The assignment of the defect label "true" to the "menjawab kuis" and "menilai kuis" modules is based on the implementation phase of these modules, which involves a higher level of complexity and more dynamic feature changes. This makes these two modules more prone to defects compared to the other modules.

Table 6. Data SLOC, CC, SQL Complexity

No.	Module Name	File Category	Filename	SLOC	CC	SQLC
1	*Menjawab Kuis*	*File Model*	*Answer_model.php*	38	3	1,75
			Question_model.php	31	2	1,60
		File View	*exam.php*	39	4	0
			finish.php	13	2	0
		File Controller	*Exam.php*	49	7	0
				170	18	3,35
2	*Update Biodata*	*File Model*	*Member_model.php*	41	4	1,20
		File View	*profile.php*	25	3	0
		File Controller	*Member.php*	54	7	0
				120	14	1,2
3	*Posting Soal*	*File Model*	*Category_model.php*	36	5	2,00
			Question_model.php	31	2	1,65
		File View	*room/question.add.php*	22	2	0
		File Controller	*Question.php*	69	8	0
				158	17	3,65
4	*Menilai Kuis*	*File Model*	*Answer_model.php*	101	9	10,55
		File View	*room/answer.php*	29	2	0
		File Controller	*Answer.php*	50	8	0
				180	19	10,55
5	*Input Kelas*	*File Model*	*Category_model.php*	18	2	0,75
			Class_model.php	34	3	1,50
		File View	*panel/category.add.php*	22	2	0
		File Controller	*Category.php*	47	7	0
				121	14	2,25
6	*Input Siswa*	*File Model*	*Class_model.php*	34	3	1,50
			Member_model.php	25	2	1,35
		File View	*panel/user.add.php*	22	2	0
		File Controller	*User.php*	53	7	0
				134	14	2,85
7	*Input Guru*	*File Model*	*Class_model.php*	34	3	1,50

			Teach_model.php	19	2	0,90
			Teacher_model.php	21	2	1,05
		File View	*panel/employee.add.php*	25	2	0
		File Controller	*Employee.php*	62	10	0
				161	19	3,45
8	*Input Mapel*	*File Model*	*Categgory_model.php*	36	5	2,00
			Question_model.php	31	2	1,65
		File View	*panel.question.add.php*	22	2	0
		File Controller	*Question.php*	65	7	0
				154	16	3,65

3.2. Preprocessing

In NASA MDP software defect datasets, a preprocessing stage is carried out because the data cannot be applied directly to the classification process. The preprocessing stage is carried out in two stages, namely data cleaning and normalization, the following are the details of the process at each stage.

1) Cleaning Data

- Convert arff to csv format.
- There are a number of missing values in the datasets (?) replaced with 0.
- Adjusting data attributes and labels.
- Duplicate value removal.

2) Normalization

At the normalization stage, the min-max method is used to obtain a value range of 0-1 with the aim that all attribute values in the dataset have a uniform scale, which is important for several machine learning and data analysis methods.

3.3. Naïve Bayes Classifier

Based on the dataset that has been obtained, then the classification process is carried out using the naïve bayes method. In the three datasets CMI, PC2, and KC 3, the data is divided into the proportion of data allocated as testing and training data. In this case, 30% of the total data will be used as testing data, while 70% will be training data.

Based on these results, the accuracy of CM1 is 88%, PC2 is 97%, and KC3 is 78%, meanwhile, for the data of the eight software modules used, an accuracy of 67% was achieved. Based on the accuracy obtained, it can be concluded that the developed model has a varying level of accuracy for each class or category tested. Accuracy is an important metric in evaluating model performance, but other aspects such as precision, recall, and f-measure need to be considered to get a more comprehensive picture of the model's performance in predicting each class accurately.

3.4. Evaluation

To obtain a more detailed picture of the proposed model, an evaluation process was carried out that measures the values of precision, recall, and f-measure. Figure 10 to Figure 13 is the evaluation process that has been carried out. Based on the results obtained from the data used, the proposed method provides accuracy for the NASA MDP software defect dataset, as follows: CM 1 of 88%, PC2 of 97%, and KC3 of 78%. Given that the NASA MDP dataset does not have attributes that pay attention to the use of SQL commands, the data of the eight software modules that directly pay attention to the use of SQL commands are used with the use of the SQL complexity attribute which shows how complex the SQL command is. The provision of defect labels is used for the classification process using the naïve Bayes method, while the results given on the data of the eight software modules produce an accuracy of 67%.

```
Accuracy: 0.881578947368421
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.94      0.93       137
           1       0.38      0.33      0.36        15

    accuracy                           0.88       152
   macro avg       0.66      0.64      0.65       152
weighted avg       0.87      0.88      0.88       152
```

Figure 10. CM1 Evaluation Results

Accuracy: 0.9761478831246273
Classification Report:

	precision	recall	f1-score	support
0	1.00	0.98	0.99	1670
1	0.00	0.00	0.00	7
accuracy			0.98	1677
macro avg	0.50	0.49	0.49	1677
weighted avg	0.99	0.98	0.98	1677

Figure 11. PC2 Evaluation Results

Accuracy: 0.7898550724637681
Classification Report:

	precision	recall	f1-score	support
0	0.90	0.85	0.87	117
1	0.36	0.48	0.41	21
accuracy			0.79	138
macro avg	0.63	0.66	0.64	138
weighted avg	0.82	0.79	0.80	138

Figure 12. KC3 Evaluation Results

Classification Report:

	precision	recall	f1-score	support
No Defect	1.00	0.67	0.80	2
Defect	0.00	0.00	0.00	1
accuracy			0.67	3
macro avg	0.50	0.33	0.40	3
weighted avg	0.67	0.67	0.67	3

Figure 13. Evaluation Results of the Eight Software Modules

4. CONCLUSION

In this research, the following results have been obtained: (1) the research datasets used include NASA MDP software defects in projects CM1, PC2, KC3 and data from eight software modules obtained from calculations using the sloc, cc, and SQL complexity methods, (2) pre-processing stages have been carried out before classification calculations, including data cleaning and normalization, and (3) the results of the classification process were obtained with an accuracy of CM1 of 88%, PC2 of 97%, and KC3 of 78%, meanwhile, for the data of the eight software modules used, an accuracy of 67% was achieved.

ACKNOWLEDGEMENTS

Researchers would like to thank Institut Teknologi dan Bisnis STIKOM Bali who has provided support for this research.

REFERENCES

[1] K. Okumoto, "Early Software Defect Prediction: Right-Shifting Software Effort Data into a Defect Curve," in *2022 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2022, pp. 43–48.

[2] J. Xu, J. Ai, and T. Shi, "Software Defect Prediction for Specific Defect Types based on Augmented Code Graph Representation," in *2021 8th International Conference on Dependable Systems and Their Applications (DSA)*, 2021, pp. 669–678.

[3] L. Qiao, X. Li, Q. Umer, and P. Guo, "Deep learning based software defect prediction," *Neurocomputing*, vol. 385, pp. 100–110, 2020.

[4] R. S. Wahono and N. Suryana, "Combining particle swarm optimization based feature selection and bagging technique for software defect prediction," *Int. J. Softw. Eng. Its Appl.*, vol. 7, no. 5, pp. 153–166, 2013.

[5] S. Patil and B. Ravindran, "Predicting software defect type using concept-based classification," *Empir. Softw. Eng.*, vol. 25, no. 2, pp. 1341–1378, 2020.

[6] B. Dhanalaxmi, G. A. Naidu, and K. Anuradha, "Defect Classification using Relational Association Rule Mining Based on Fuzzy Classifier along with Modified Artificial Bee Colony Algorithm," *Int. J. Appl. Eng. Res.*, vol. 12, no. 11, pp. 2879–2886, 2017.

[7] B. Shuai, H. Li, M. Li, Q. Zhang, and C. Tang, "Software Defect Prediction Using Dynamic Support Vector Machine," in *2013 Ninth International Conference on Computational Intelligence and Security*, 2013, pp. 260–263.

[8] J. Zheng, "Cost-sensitive boosting neural networks for software defect prediction," *Expert Syst. Appl.*, vol. 37, no. 6, pp. 4537–4543, 2010.

[9] Endah Ratna Arumi, Sumarno Adi Subrata, and Anisa Rahmawati, "Implementation of Naïve bayes Method for Predictor Prevalence Level for Malnutrition Toddlers in Magelang City," *J. RESTI Rekayasa Sist. Dan Teknol. Inf.*, vol. 7, no. 2, Mar. 2023, doi: 10.29207/resti.v7i2.4438.

[10] Najirah Umar and M. Adnan Nur, "Application of Naïve Bayes Algorithm Variations On Indonesian General Analysis Dataset for Sentiment Analysis," *J. RESTI Rekayasa Sist. Dan Teknol. Inf.*, vol. 6, no. 4, Aug. 2022, doi: 10.29207/resti.v6i4.4179.

[11] Agung Nugroho and Yoga Religia, "Analisis Optimasi Algoritma Klasifikasi Naive Bayes menggunakan Genetic Algorithm dan Bagging," *J. RESTI Rekayasa Sist. Dan Teknol. Inf.*, vol. 5, no. 3, Jun. 2021, doi: 10.29207/resti.v5i3.3067.

[12] F. M. Tua and W. D. Sunindyo, "Software Defect Prediction Using Software Metrics with Naïve Bayes and Rule Mining Association Methods," in *2019 5th International Conference on Science and Technology (ICST)*, 2019, pp. 1–5.

[13] M. A. P. Subali and S. Rochimah, "A new model for measuring the complexity of SQL commands," in *International Conference on Information Technology and Electrical Engineering (ICITEE 2018)*, Bali, 2018, pp. 1–5.

[14] Y. Shen, S. Hu, S. Cai, and M. Chen, "Software Defect Prediction based on Bayesian Optimization Random Forest," in *2022 9th International Conference on Dependable Systems and Their Applications (DSA)*, 2022, pp. 1012–1013.

[15] Khadijah, A. Adorada, P. W. Wirawan, and K. Kurniawan, "The Comparison of Feature Selection Methods in Software Defect Prediction," in *2020 4th International Conference on Informatics and Computational Sciences (ICICoS)*, 2020, pp. 1–6.

[16] E. A. Felix and S. P. Lee, "Integrated Approach to Software Defect Prediction," *IEEE Access*, vol. 5, pp. 21524–21547, 2017.

[17] T. F. Husin, M. R. Pribadi, and Yohannes, "Implementation of LSSVM in Classification of Software Defect Prediction Data with Feature Selection," in *2022 9th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 2022, pp. 126–131.

[18] I. B. G. W. Putra, M. Sudarma, and I. N. S. Kumara, "Klasifikasi Teks Bahasa Bali dengan Metode Supervised Learning Naive Bayes Classifier," *Teknol. Elektro*, vol. 15, no. 2, pp. 81–86, 2016.

[19] Y. D. PRAMUDITA, S. S. PUTRO, and N. MAKHMUD, "Klasifikasi Berita Olahraga menggunakan Metode Naive Bayes dengan Enhanced Confix Stripping Stemmer," *J. Teknol. Inf. Dan Ilmu Komput. JTIIK*, vol. 5, no. 3, pp. 269–276, 2018.

[20] J. Li, P. He, J. Zhu, and M. R. Lyu, "Software Defect Prediction via Convolutional Neural Network," in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, 2017, pp. 318–328.