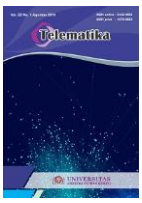

Available online at :
http://ejournal.amikompurwokerto.ac.id/index.php/telematika/

Telematika

Accreditated SINTA "2" Kemenristek/BRIN, No. 85/M/KPT/2020

Comparative Analysis of Distance Metrics in KNN and SMOTE Algorithms for Software Defect Prediction

Khusnul Rahmi Maulidha[1], Mohammad Reza Faisal [2,] Setyo Wahyu Saputro [3], Friska Abadi [4], Dodon Turianto Nugrahadi [5], Puput Dani Prasetyo Adi [6], Hariyady Hariyady [7,8]

[1,2,3,4,5] Department of Computer Science, Faculty of Mathematics and Natural Science
[6] Research Center for Telecommunication
[7] Faculty of Computing and Informatics
[8] Department of Informatics, Faculty of Engineering
[1,2,3,4,5] Lambung Mangkurat University, Banjarbaru, Indonesia
[6] National Research and Innovation Agency, Jakarta, Indonesia
[7] Universiti Malaysia Sabah, 88400 Kota Kinabalu, Malaysia
[8] Universitas Muhammadiyah Malang, Malang, Indonesia
E-mail:khusnulrahmimaulidha@gmail.com[1], reza.faisal@ulm.ac.id[2], setyo.saputro@ulm.ac.id[3], friska.abadi@ulm.ac.id[4], dodonturianto@ulm.ac.id[5], pupu008@brin.go.id[6], hariyady_di21@iluv.ums.edu.my[7], hariyady@umm.ac.id[8]

ARTICLE INFO

History of the article:
Received October 27, 2024
Revised September 23, 2024
Accepted February 26, 2025

Keywords:
Software Defect Prediction
SMOTE
KNN Algorithm
Distance Metrics

Correspondence:
E-mail: reza.faisal@ulm.ac.id

ABSTRACT

As the complexity and scale of projects increase, new challenges arise related to handling software defects. One solution uses machine learning-based software defect prediction techniques, such as the K-Nearest Neighbors (KNN) algorithm. However, KNN's performance can be hindered by the majority vote mechanism and the distance/similarity metric choice, especially when applied to imbalanced datasets. This research compares the effectiveness of Euclidean, Hamming, Cosine, and Canberra distance metrics on KNN performance, both before and after the application of SMOTE (Synthetic Minority Over-sampling Technique). Results show significant improvements in the AUC and F-1 measure values across various datasets after the SMOTE application. Following the SMOTE application, Euclidean distance produced an AUC of 0.7752 and an F1 of 0.7311 for the EQ dataset. With Canberra distance and SMOTE, the JDT dataset produced an AUC of 0.7707 and an F-1 of 0.6342. The LC dataset improved to 0.6752 and 0.3733 in tandem with the ML dataset, which climbed to 0.6845 and 0.4261 with Canberra distance. Lastly, after using SMOTE, the PDE dataset improved to 0.6580 and 0.3957 with Canberra distance. The findings confirm that SMOTE, combined with suitable distance metrics, significantly boosts KNN's prediction accuracy, with a P-value of 0.0001.

INTRODUCTION

As projects increase in complexity and size, new software quality challenges arise, including issues related to the handling of software errors or defects. Testing and review are the traditional methods used to find bugs or defects in the software. However, these activities can be time-consuming (Giray et al., 2023). One method for dealing with this problem is the use of software fault or defect prediction tools. Finding possible faults or defects that might arise throughout the software development process is the aim of software defect prediction, thereby enabling quick fixes and more effective risk management (Jin,

2021). Mehta & Patnaik (2021) as well as Reddivari & Raman (2019) outlined how precise software defect prediction may lower expenses, streamline the testing process, and enhance software quality.

A method often used for software defect prediction is machine learning classification. One of the benefits of machine learning algorithms is their capacity to gradually enhance their classification methods while remaining dynamic (Mahesh et al., 2020). KNN is an example of a machine learning classification algorithm. The very simple concept of KNN is its advantage. The KNN algorithm works by comparing the new data to the data that is most similar to or closest to the data in the training data (Uddin et al., 2022). The distance metric is crucial in deciding the final classification result because the performance of KNN depends on the distance/similarity measure utilized (Alfeilat et al., 2023).

Mushtaq et al., (2020) investigated the performance of the KNN algorithm with and without L_1based selection and Chi-square base selection using a number of distance metrics, including Euclidian, Minkowski, Manhattan, Hamming, Canberra, Cosine, Correlation, and Chebyshev, in order to classify on the WBC and MDBC datasets. According to the results, the Manhattan distance function with K = 1 in the WBC dataset used chi-square feature selection to obtain a 99.42% accuracy and a 1.00 AUC value. Following the use of the Chisquare feature selection technique, the Canberra and Manhattan distance functions achieved the best result of 98.62% with a K value of 7 or 8 for the WDBC dataset. The research by Hidayati & Hermawan (2021) compared the performance of Euclidean and Manhattan distance measurements for categorizing students' graduation, and obtained the highest accuracy value of 85.28% in both distance metrics. According to a research by (Tsalera et al., 2020), which compared the use of various distance metrics, including Euclidian, Cosine, and Chebyshev, to monitor, profile, and classify urban environmental noise using sound characteristics, the Cosine distance had the best prediction rate of 85%. Cosine and Chi-square distance metrics combined with feature selection produced the best results in a research by Rehman et al., (2021) that compared the performance of various distance metrics, including Euclidean, Canberra, Chebyshev, Minkoswki, Cosine, Correlation, and Manhattan. The research by (Nayak et al, 2022) additionally examined how well the KNN algorithm classified stars using a variety of distance metrics, including Euclidean, Manhattan, Chebyshev, Cosine, and Hamming, showing that Cosine distance obtained the highest accuracy value of 84.72%. Additionally, research by (Iqbal et al., 2019) using the NASA dataset and research by (Kumar et al., 2022) using the PROMISE dataset repository have applied the KNN algorithm to software defect prediction.

Moreover, KNN also applies the majority rule to its classification. Sun & Chen (2021) dan Suyanto et al., (2022) explain that when KNN is used to classify test data, the class that is the majority in a neighborhood of k values is used as the classification result. This majority-based classification method worsens the performance of KNN when applied to data with imbalance problems. Classification models applied to data that have unbalanced problems tend to neglect the minor classes and make better predictions about the major classes (Chakrabortyet al., 2021), (Kumar et al., 2021). Many approaches, such as an algorithm-level approach, a data-level approach, or a hybrid strategy that combines the two, can be used to solve the imbalance problem (Zhao et al., 2021). Several resampling strategies are used in the data-level method to balance the distribution of classes (Huang et al., 2020). One example is the Synthetic Minority Over-sample Technique (SMOTE). Research by Prasetya & Abdurakhman (2023) and research by Pertiwi et al, (2020) have shown that the application of SMOTE helps to solve the issue of dataset imbalance and enhances the KNN algorithm's classification performance.

Numerous datasets are frequently utilized in the domain of software defect prediction research, including but not limited to NASA MDP, PROMISE, among others. The NASA MDP dataset encompasses software development projects undertaken by NASA since the year 1990. Conversely, the PROMISE dataset has been derived from various sources, including contributions from the industry and NASA since the year 2005. In the year 2010, the AEEEM datasets were compiled from open-source projects, specifically Eclipse JDT, PDE, Equinox, Lucene, and Mylyn, which represent authentic industrial projects. The scholarly investigation utilizing this dataset is predicated on its potential to identify software defects across diverse industries. The research employing AEEEM datasets is documented in the work of (Malhotra et al., 2021). This particular study utilized the Mylyn dataset from AEEEM, applying the SMOTE technique alongside SVM and Naive Bayes algorithms, yielding an AUC performance of 0.73 and 0.70, respectively. Recent investigations utilizing the AEEEM dataset are chronicled in the research conducted by (Bala et al., 2024). This study executed software defect prediction employing four distinct machine learning methodologies, specifically KNN, Random Forest, SVM, and Linear Regression (LR), without any modifications, resulting in average F1 score classification performances of 0.542, 0.6, 0.582, and 0.526, respectively. The KNN algorithm, while utilizing the Euclidean distance calculation technique, shows minimal performance differences compared to SVM and LR models.

In light of the aforementioned background and the prevailing state of the art, we embarked on research that concentrated exclusively on the implementation of KNN algorithms utilizing distance calculation methodologies distinct from Euclidean techniques, such as Hamming, Cosine, and Canberra. Furthermore, this research incorporates an array of varying parameter values for k, specifically 1, 3, 5, 7, and 9. Additionally, data balancing techniques employing SMOTE methodologies are also implemented. The objective of this research is to ascertain an optimal amalgamation of techniques and parameters to enhance the efficacy of software defect prediction models.

RESEARCH METHODS

The flow of the research methods can be seen in Figure 1.

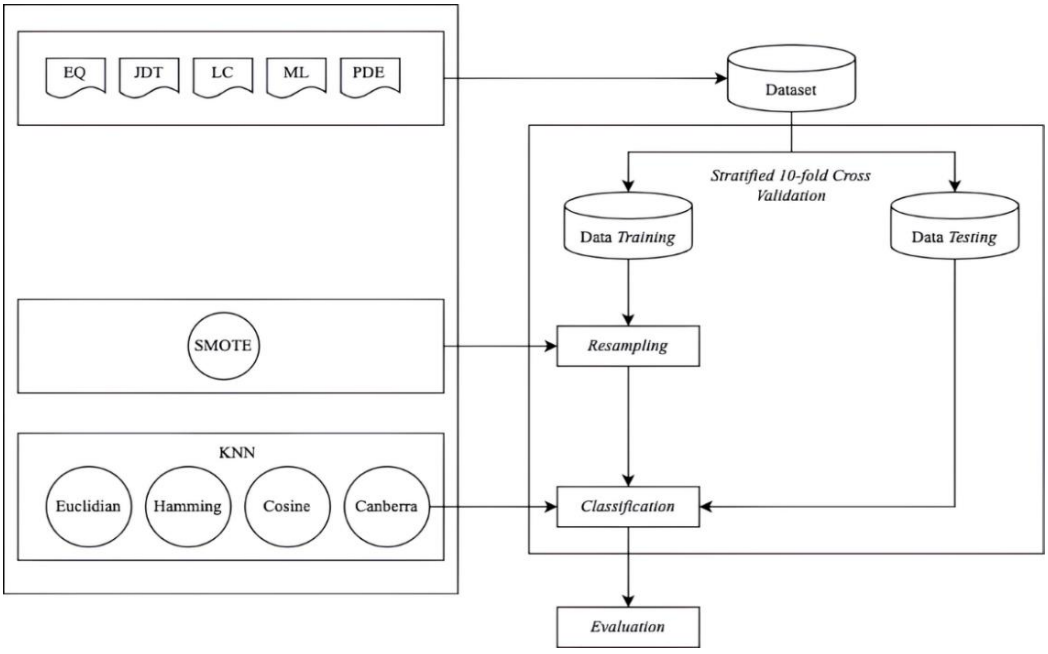

Figure 1. Research Flow

1. Dataset Collection

This study makes use of the AEEEM dataset, which was gathered by (D'Ambros 2010). The AEEEM dataset consists of five software system models and metrics: Equinox Framework (EQ), Mylyn (ML), Eclipse PDE UI (PDE), Eclipse JDT Core (JDT), and Apache Lucene (LC). The AEEEM dataset has also been used for software defect prediction, such as in research by (Zhao 2021) and research by Duy-An Ha et al., (2019). This dataset has two classes, clean and buggy. Table 1 displays the AEEEM dataset specifications.

Table 1. AEEEM dataset specifications

Dataset	*Number of Metrics*	*Number of Data*	*Number of Buggy Data*	*Number of Clean Data*
EQ	61	324	129	195
JDT	61	997	206	791
LC	61	691	64	627
ML	61	1844	245	1617
PDE	61	1497	209	1288

2. Dataset Distribution

Next, the dataset is split into testing and training subsets. Testing data is used to evaluate a model's performance, whereas training data is used to train classification algorithms. The stratified 10-fold cross-validation technique is used to partition the dataset. There are ten partitions in the dataset, where the remaining partitions serve as training data and the first partition serves as test data. This process is repeated 10 times, with each partition alternating as the test data. The stratified method is useful when the data has a significant imbalance between classes because it ensures that the distribution of every partition maintains the balance of each partition's distribution while reflecting the dataset's overall class distribution (Suyanto et al., 2022). Table 2 displays the AEEEM dataset distribution results.

Table 2. AEEEM dataset distribution

Dataset	*Training*	*Testing*
EQ	291-291	32-33
JDT	897-898	99-100
LC	621-622	69-70
ML	1675-1676	186-187
PDE	1347-1348	149-150

3. *Resampling* Dataset

The imbalance issue with the AEEEM dataset is resolved by resampling the data using the Synthetic Minority Over-sampling Technique (SMOTE). To stop significant data in the majority class from being erased, the SMOTE technique is utilized (Dudjak & Martinović, 2020). To ensure that the total of the two classes is equal, SMOTE balances the data by adding new data to the minor class of the artificially generated data (Kaope & Pristyanto, 2023). Only training data is used for this resampling method. According to Vandewiele et al., (2021), applying oversampling techniques to the entire dataset can lead to biased models and overoptimistic estimation performance because the model already has knowledge of the test data. Table 3 displays the outcomes of resampling the AEEEM dataset using the SMOTE method.

Table 3. Resampling results of AEEEM dataset

Dataset	*Clean (0)*	*Buggy (1)*
EQ	175-176	175-176
JDT	711-712	711-712
LC	564-565	564-565
ML	1455-1456	1455-1456
PDE	1159-1160	1159-1160

4. KNN Classification Algorithm

The majority rule and the distance/similarity metric determine how well KNN performs in determining the final classification result. The KNN algorithm works by finding the nearest neighbours of the data to be classified based on a distance metric. Then, KNN determines the final classification of the new data according to its k nearest neighbours majority category. In order to determine how many nearest neighbours will be taken into account for the final classification result, the KNN algorithm uses a parameter called the k value. According to Taunk et al., (2019), one technique to figure out the value of k is to run the classification multiple times with various values to see which value produces the best outcomes.

In this study, four distinct distance metrics are employed with k values of 1, 3, 5, 7, and 9, namely: Euclidean, Hamming, Cosine, and Canberra. The equations (1), (2), (3), and (4) show the distance metric equation that was applied..

Euclidian:
$$ED(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (1)$$

Hamming:
$$HamD(x,y) = \sum_{i=1}^{n} 1_{x_i \neq y_i} \quad (2)$$

Cosine:
$$CosD(x,y) = 1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \quad (3)$$

Canberra:
$$CanD(x,y) = \sum_{i=1}^{n} \frac{|x_i - y_i|}{|x_i| + |y_i|} \quad (4)$$

5. Evaluation

The performance value of each software failure prediction method is calculated with AUC as the first indicator and F-1 measure as the second indicator. The AUC value theoretically lies between 0 and 1. An accurate prediction should have a large AUC, the predictor performs well in the sensitivity and specificity dimensions with values above 0.5 to closer to 1.0. (Bowers & Zhou, 2019). Meanwhile, the F-1 measure is an evaluation metric that combines the combining the harmonic mean of recall and precision into a single number to represent the model's performance (Javed et al., 2024). The F-1 measure value of a classification model can be calculated using equation (5).

$$2 \times \frac{(precision \times recall)}{precision + recall} \quad (5)$$

In addition to using the F-1 and AUC measure values, this study also used a statistical test called the T-test to compare the average classification group with the use of SMOTE and without the use of SMOTE. Equation (6) is used to obtain the T-Test value.

$$T = \frac{mean_1 - mean_2}{std\sqrt{n}} \quad (6)$$

RESULTS AND DISCUSSION

1. Result

After going through the stages of data collection and data distribution, the dataset then went through the classification stage twice: without resampling using SMOTE, and with resampling using SMOTE. The classification process was performed using the KNN algorithm with four different distance metrics: Euclidean, Hamming, Cosine, and Canberra, where the k values used include: 1, 3, 5, 7, and 9. The classification results for the tables 4, 5, 6, and 7 display each distance metric that was employed.

Table 4. Average AUC and F-1 Measure Euclidean distance metric

	No-SMOTE		*SMOTE*	
Dataset	*Mean AUC*	*Mean F-1 Measure*	*Mean AUC*	*Mean F-1 Measure*
EQ	0.6962	0.6314	0.7212	0.6746
JDT	0.7228	0.5768	0.7400	0.5557
LC	0.5320	0.1052	0.6336	0.2646
ML	0.5722	0.2475	0.6429	0.3313
PDE	0.5382	0.1568	0.6384	0.3355

Table 5. Average AUC and F-1 Measure Hamming distance metric

	No-SMOTE		*SMOTE*	
Dataset	*Mean AUC*	*Mean F-1 Measure*	*Mean AUC*	*Mean F-1 Measure*
EQ	0.6376	0.4785	0.6357	0.4719
JDT	0.5741	0.2527	0.5811	0.2746
LC	0.5158	0.0557	0.5158	0.0557
ML	0.5700	0.2337	0.5832	0.2750
PDE	0.5312	0.1174	0.5330	0.1265

Table 6. Average AUC and F-1 Measure Cosine distance metric

	No-SMOTE		*SMOTE*	
Dataset	*Mean AUC*	*Mean F-1 Measure*	*Mean AUC*	*Mean F-1 Measure*
EQ	0.6728	0.6029	0.7020	0.6572
JDT	0.6644	0.4705	0.7264	0.5239
LC	0.5475	0.1443	0.6540	0.2743
ML	0.5826	0.2681	0.6605	0.3459
PDE	0.5592	0.2169	0.6380	0.3360

Table 7. Average AUC and F-1 Measure Canberra distance metric

	No-SMOTE		*SMOTE*	
Dataset	*Mean AUC*	*Mean F-1 Measure*	*Mean AUC*	*Mean F-1 Measure*
EQ	0.6796	0.5984	0.6941	0.6241
JDT	0.7094	0.5568	0.7521	0.6063
LC	0.5902	0.2650	0.6618	0.3373
ML	0.6255	0.3642	0.6730	0.4095
PDE	0.5765	0.2615	0.6359	0.3648

2. Discussion

The application of SMOTE generally improves the AUC and F1-measure values on almost all datasets and distance metrics, as shown in Table 4, Table 5, Table 6, and Table 7. A significant increase in AUC values is seen in datasets such as LC and ML, where the AUC values without SMOTE were previously very low. After applying SMOTE, the prediction model with the Euclidean distance metric on the average AUC value for the entire dataset was 0.6677 and an F-1 measure of 0.4642, which increased to 0.6999 and 0.4901 respectively. The values of the AUC and the F-1 measure of the prediction model using the Euclidean distance metric at each value of k for every data set is displayed in Figure 2.

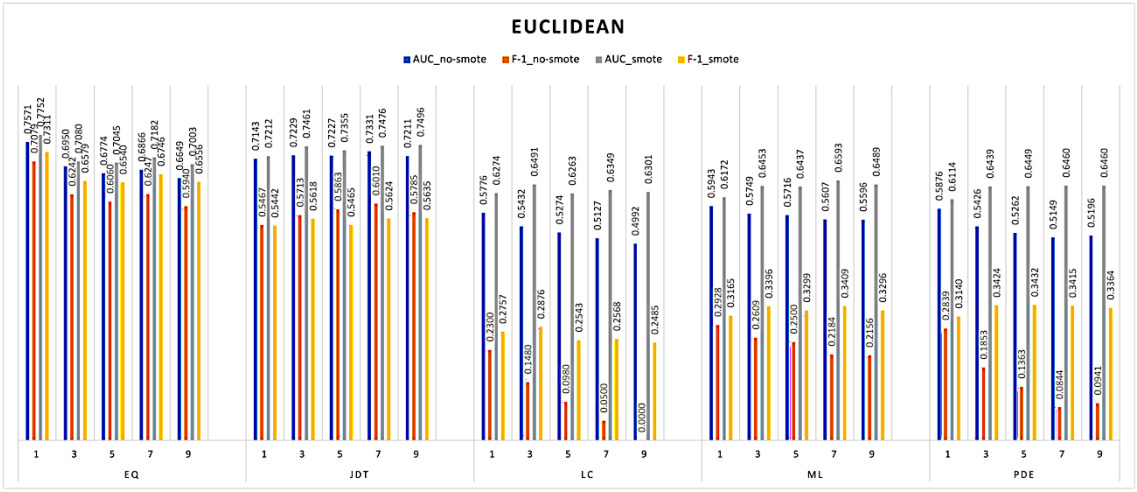

Figure 2. KNN classification results with Euclidean distance metric

After applying SMOTE, the prediction model with the Hamming distance metric on the average AUC value for the entire dataset was 0.5464 and an F-1 measure of 0.1426 which increased to 0.6163 and 0.2330 respectively. Figure 3 displays the results of the F-1 measure and the AUC of the prediction model using the Hamming distance metric for each value of k for each data set.

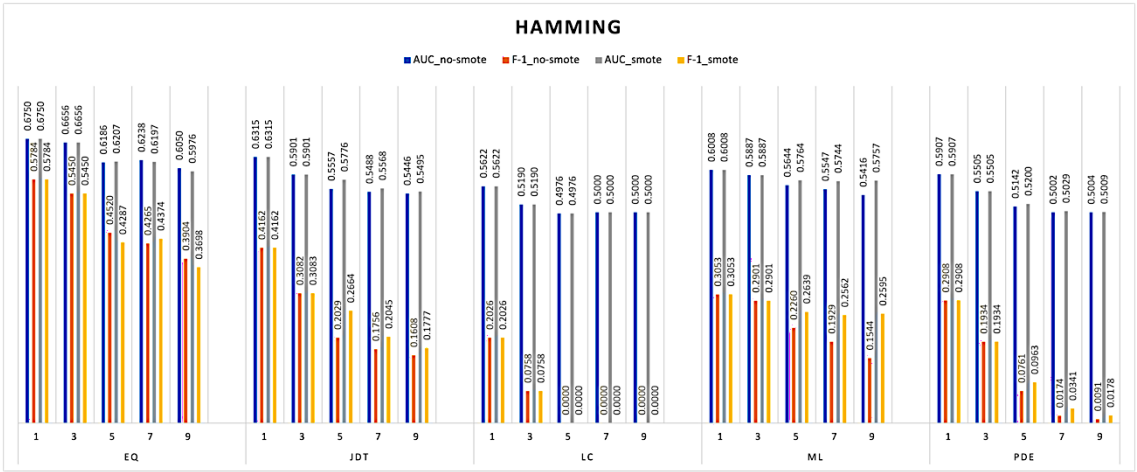

Figure 3. KNN classification results with Hamming distance metric

After applying SMOTE, the prediction model with the Cosine distance metric on the whole dataset obtained an average AUC value of 0.5876 and an F-1 measure of 0.2784 which increased to 0.6399 and 0.3404 respectively. Figure 4 displays the results of the F-1 measure and the AUC of the prediction model using the Cosine distance metric at each value of k for each data set.

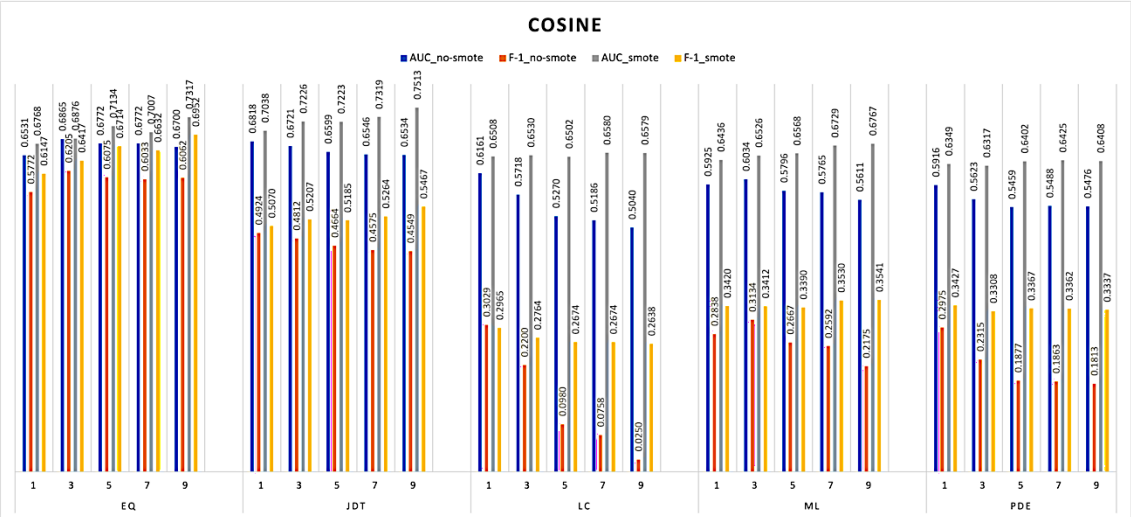

Figure 4. KNN classification results with Cosine distance metric

After applying SMOTE, the prediction model with the Canberra distance metric on the average AUC value for the entire dataset was 0.5513 and F-1 Measure of 0.1881 which increased to 0.6113 and 0.2907 respectively. Figure 5 shows the values of the F-1 measure and the AUC of the prediction model using the Canberra distance metric at each value of k for each set of data.

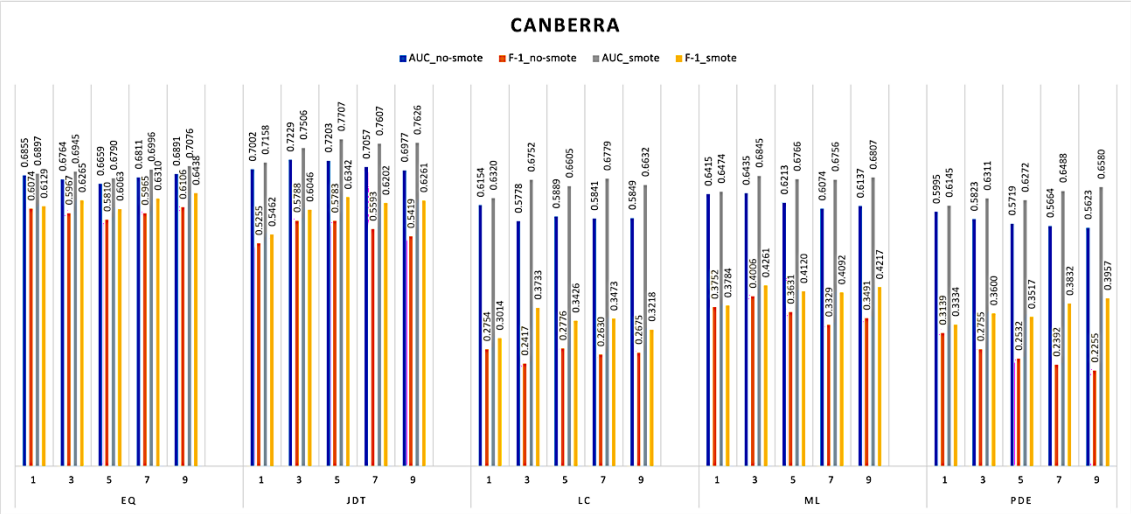

Figure 5. KNN classification results with Canberra distance metric

With reference to Figures 2, 3, 4, and 5, Table 8 shows the comparison of the best AUC and F-1 Measure results on KNN classification for each dataset without the application of SMOTE and with the application of SMOTE.

Table 8. Best AUC and F-1 Measure results for each dataset

	No-SMOTE			*SMOTE*		
Dataset	*Distance Metric*	*AUC*	*F-1 Measure*	*Distance Metric*	*AUC*	*F-1 Measure*
EQ	Euclidean	0.7571	0.7079	Euclidean	0.7752	0.7311
JDT	Euclidean	0.7331	0.5713	Canberra	0.7707	0.6342
LC	Cosine	0.6161	0.3029	Canberra	0.6779	0.3473
ML	Canberra	0.6435	0.4006	Canberra	0.6845	0.4261
PDE	Cosine	0.5916	0.2975	Canberra	0.6580	0.3957

Furthermore, in order to determine whether the application of SMOTE in this study is beneficial and provides a significant difference, a statistical method, To check for significance, one uses the T-test,

specifically. The paired t-test type was employed in this study because it contrasts a model's performance on the same dataset before and after SMOTE was applied. Table 9 displays the significance test findings.

Table 9. T-Test Result

	AUC		*F-1 Measure*	
Group	*No-SMOTE*	*SMOTE*	*No-SMOTE*	*SMOTE*
Mean	0.604892	0.651132	0.330213	0.392227
SD	0.068017	0.064977	0.193559	0.173092
SEM	0.006802	0.006498	0.019356	0.017309
N	100	100	100	100
P-Value	0.0001		0.0001	

The statistically significant mean differences in AUC from 0.604892 to 0.651132 and F-1 Measure from 0.330213 to 0.392227, with a very small P value of 0.0001, indicate that these results are no coincidence. In addition, the non-zero confidence intervals of -0.046240 for the AUC and -0.062014 for the F-1 measure further support the evidence that there is a real difference between the two groups. The implementation of SMOTE successfully improved the average of either the AUC or F-1 Measure metrics used in this analysis. The results were highly statistically significant, demonstrating the obvious benefits of the use the SMOTE technique to address the data imbalance.

The analysis presented herein juxtaposes the optimal performance outcomes of the current research with two prominent studies in the field. The initial comparison entails an evaluation of the findings from this investigation against those documented in the study conducted by (Bala et al., 2024), which is illustrated in Table 10. The performance of the K-Nearest Neighbors (KNN) algorithm demonstrates superiority solely in the context of the EQ and JDT projects. Conversely, in the datasets pertaining to the LC, ML, and PDE projects, both KNN and the Synthetic Minority Over-sampling Technique (SMOTE) algorithms exhibit notable limitations. This deficiency may stem from the pronounced data imbalance ratios present in these three datasets, coupled with the concentration of minority data within a singular region of the data space, thereby hindering the efficacy of SMOTE in achieving data balance. Consequently, this limitation adversely impacts the overall performance of the algorithm.

Table 10. Performance comparison with (Bala et al., 2024).

Dataset	(Bala et al., 2024)		KNN + SMOTE	
	Method	F1 Measure	Method	F1 Measure
EQ	RF	0.57	Euclidean	**0.7311**
JDT	RF	0.62	Canberra	**0.6342**
LC	SVM	**0.59**	Canberra	0.3473
ML	RF	**0.61**	Canberra	0.4261
PDE	SVM	**0.63**	Canberra	0.3957

Table 11 presents a comparative analysis with the findings of Malhotra et al. (2021), which employed Support Vector Machine (SVM) and Naïve Bayes (NB) algorithms alongside data balancing techniques utilizing SMOTE. It is pertinent to note that this particular study focused exclusively on a subset of the AEEEM dataset, specifically the ML portion. This performance evaluation reveals that KNN, utilizing Canberra distance calculation methodologies, can perform effectively when contrasted with SVM and Naïve Bayes algorithms that do not incorporate data balancing. However, following the application of SMOTE for data balancing, the model developed in the research by (Malhotra et al., 2021) demonstrated superior performance outcomes.

Table 11. Performance comparison with (Malhotra et al., 2021).

Dataset	(Malhotra et al., 2021)		Our methods	
	Method	AUC	Method	AUC
ML	SVM	0.52	KNN Canberra	**0.64**
	SVM + SMOTE	**0.73**	KNN Canberra + SMOTE	0.68
	NB	0.57	KNN Canberra	**0.64**
	NB + SMOTE	**0.70**	KNN Canberra + SMOTE	0.68

CONCLUSIONS AND RECOMMENDATIONS

This study concludes that the application of SMOTE can generally improve the performance of KNN classification for nearly every distance metric examined. The best results were displayed by the Canberra distance metric, consistently improving AUC and F-1 measure values on all datasets after SMOTE implementation, followed by Euclidean, Cosine, and Hamming.

Despite the fact that the SMOTE application in this study can improve the KNN algorithm's classification performance, the resulting AUC and F-1 measure values are still relatively low. Therefore, some recommendations for future research can be made, including using other resampling techniques to discuss the AEEEM dataset's imbalance problem and the application of other distance metrics or combinations of distance metrics with KNN.

REFERENCES

Alfeilat, H. A. A., Hassanat, A. B. A., Lasassmeh, O., Tarawneh, A. S., Alhasanat, M. B., Salman, H. S. E., & Prasath, V. B. S. (2024). Effects of Distance Measure Choice on K-Nearest Neighbor Classifier Performance: A Review | Big Data. Retrieved July 26, 2024, from https://www.liebertpub.com/doi/abs/10.1089/big.2018.0175.

Bala, Y. Z., Samat, P. A., Sharif, K. Y., & Manshor, N. (2024). The influence of machine learning on the predictive performance of cross-project defect prediction: empirical analysis. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 22(4), 830–837.

Bowers, A. J., & Zhou, X. (2019). Receiver Operating Characteristic (ROC) Area Under the Curve (AUC): A Diagnostic Measure for Evaluating the Accuracy of Predictors of Education Outcomes. *Journal of Education for Students Placed at Risk (JESPAR)*, *24*(1), 20–46. Routledge.

Chakraborty, J., Majumder, S., & Menzies, T. (2021). Bias in machine learning software: Why? how? what to do? *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2021 (pp. 429–440). New York, NY, USA: Association for Computing Machinery. Retrieved August 24, 2024, from https://dl.acm.org/doi/10.1145/3468264.3468537

D'Ambros, M., Lanza, M., & Robbes, R. (2010, May 2). (PDF) An extensive comparison of bug prediction approaches. Retrieved July 26, 2024, from https://www.researchgate.net/publication/221657038_An_extensive_comparison_of_bug_prediction_approaches

Dudjak, M., & Martinović, G. (2020). In-Depth Performance Analysis of SMOTE-Based Oversampling Algorithms in Binary Classification. *International journal of electrical and computer engineering systems*, *11*(1), 13–23. Sveučilišta Josipa Jurja Strossmayera u Osijeku, Elektrotehnički fakultet.

Duy-An Ha, Chen, T.-H., & Yuan, S.-M. (2019). Unsupervised methods for Software Defect Prediction. *Proceedings of the 10th International Symposium on Information and Communication Technology*, SoICT '19 (pp. 49–55). New York, NY, USA: Association for Computing Machinery. Retrieved August 24, 2024, from https://doi.org/10.1145/3368926.3369711

Giray, G., Bennin, K. E., Köksal, Ö., Babur, Ö., & Tekinerdogan, B. (2023). On the use of deep learning in software defect prediction. *Journal of Systems and Software*, *195*, 111537.

Hidayati, N., & Hermawan, A. (2021). K-Nearest Neighbor (K-NN) algorithm with Euclidean and Manhattan in classification of student graduation. *Journal of Engineering and Applied Technology*, *2*(2). Retrieved August 24, 2024, from https://journal.uny.ac.id/index.php/jeatech/article/view/42777

Huang, C., Li, Y., Loy, C. C., & Tang, X. (2020). Deep Imbalanced Learning for Face Recognition and Attribute Prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(11), 2781–2794. Presented at the IEEE Transactions on Pattern Analysis and Machine Intelligence.

Iqbal, A., Aftab, S., Ali, U., Nawaz, Z., Sana, L., Ahmad, M., & Husen, A. (2019). Performance Analysis of Machine Learning Techniques on Software Defect Prediction using NASA Datasets. *International Journal of Advanced Computer Science and Applications*, *10*, 300–308.

Javed, K., Shengbing, R., Asim, M., & Wani, M. A. (2024, April 10). Cross-Project Defect Prediction Based on Domain Adaptation and LSTM Optimization. Retrieved July 26, 2024, from https://www.mdpi.com/1999-4893/17/5/175

Jin, C. (2021). Cross-project software defect prediction based on domain adaptation learning and optimization. *Expert Systems with Applications*, *171*, 114637.

Kaope, C., & Pristyanto, Y. (2023). The Effect of Class Imbalance Handling on Datasets Toward Classification Algorithm Performance. *MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, *22*(2), 227–238.

Kumar, P., Bhatnagar, R., Gaur, K., & Bhatnagar, A. (2021). Classification of Imbalanced Data:Review of Methods and Applications. *IOP Conference Series: Materials Science and Engineering*, *1099*(1), 012077. IOP Publishing.

Kumar, P. S., Nayak, J., & Behera, H. S. (2022). Model-based Software Defect Prediction from Software Quality Characterized Code Features by using Stacking Ensemble Learning. *Journal of Engineering Science and Technology Review*, *15*(2), 137–155.

Malhotra, R., Agrawal, V., Pal, V., & Agarwal, T. (2021). Support vector based oversampling technique for handling class imbalance in software defect prediction. *2021 11th International Conference on Cloud Computing, Data Science \& Engineering (Confluence)*, 1078–1083.

Mahesh Kumar Thota, Francis H Shajin, & P. Rajesh. (2020). Survey on software defect prediction techniques. *International Journal of Applied Science and Engineering*, *17*(4).

Mehta, S., & Patnaik, K. S. (2021). Improved prediction of software defects using ensemble machine learning techniques. *Neural Computing and Applications*, *33*(16), 10551–10562.

Mushtaq, Z., Yaqub, A., Sani, S., & Khalid, A. (2020). Effective K-nearest neighbor classifications for Wisconsin breast cancer data sets. *Journal of the Chinese Institute of Engineers*, *43*(1), 80–92. Taylor & Francis.

Nayak, S., Bhat, M., Reddy, N. V. S., & Rao, B. A. (2022). Study of distance metrics on k—Nearest neighbor algorithm for star categorization. *Journal of Physics: Conference Series*, *2161*(1), 012004. IOP Publishing.

Pertiwi, A. G., Bachtiar, N., Kusumaningrum, R., Waspada, I., & Wibowo, A. (2020). Comparison of performance of k-nearest neighbor algorithm using smote and k-nearest neighbor algorithm without smote in diagnosis of diabetes disease in balanced data. *Journal of Physics: Conference Series*, *1524*(1), 012048. IOP Publishing.

Prasetya, J., & Abdurakhman, A. (2023). COMPARISON OF SMOTE RANDOM FOREST AND SMOTE K-NEAREST NEIGHBORS CLASSIFICATION ANALYSIS ON IMBALANCED DATA. *MEDIA STATISTIKA*, *15*(2), 198–208. Department of Statistics, Faculty of Science and Mathematics, Universitas Diponegoro.

Prusty, S., Patnaik, S., & Dash, S. K. (2022, August 19). Frontiers | SKCV: Stratified K-fold cross-validation on ML classifiers for predicting cervical cancer. Retrieved July 26, 2024, from https://www.frontiersin.org/journals/nanotechnology/articles/10.3389/fnano.2022.972421/full

Reddivari, S., & Raman, J. (2019). Software Quality Prediction: An Investigation Based on Machine Learning. *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)* (pp. 115–122). Presented at the 2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI), Los Angeles, CA, USA: IEEE. Retrieved August 24, 2024, from https://ieeexplore.ieee.org/document/8843447/

Rehman, H. A. U., Chyi-Yeu Lin, & Zohaib Mushtaq. (2021). Effective K-Nearest Neighbor Algorithms Performance Analysis of Thyroid Disease. *Journal of the Chinese Institute of Engineers*, *44*(1), 77–87.

Sun, B., & Chen, H. (2021). A Survey of k Nearest Neighbor Algorithms for Solving the Class Imbalanced Problem -. Retrieved July 26, 2024, from https://onlinelibrary.wiley.com/doi/10.1155/2021/5520990

Suyanto, S., Yunanto, P. E., Wahyuningrum, T., & Khomsah, S. (2022). A multi-voter multi-commission nearest neighbor classifier. *Journal of King Saud University—Computer and Information Sciences*, *34*(8, Part B), 6292–6302.

Taunk, K., De, S., Verma, S., & Swetapadma, A. (2019). A Brief Review of Nearest Neighbor Algorithm for Learning and Classification. *2019 International Conference on Intelligent Computing and Control Systems (ICCS)* (pp. 1255–1260). Presented at the 2019 International Conference on Intelligent Computing and Control Systems (ICCS). Retrieved July 26, 2024, from https://ieeexplore.ieee.org/document/9065747

Tsalera, E., Papadakis, A., & Samarakou, M. (2020). Monitoring, profiling and classification of urban environmental noise using sound characteristics and the KNN algorithm. *Energy Reports*, Technologies and Materials for Renewable Energy, Environment and Sustainability, *6*, 223–230.

Uddin, S., Haque, I., Lu, H., Moni, M. A., & Gide, E. (2022). Comparative performance analysis of K-nearest neighbour (KNN) algorithm and its different variants for disease prediction. *Scientific Reports*, *12*(1), 6256. Nature Publishing Group.

Vandewiele, G., Dehaene, I., Kovács, G., Sterckx, L., Janssens, O., Ongenae, F., De Backere, F., et al. (2021). Overly optimistic prediction results on imbalanced data: A case study of flaws and benefits when applying over-sampling. *Artificial Intelligence in Medicine*, *111*, 101987.

Zhao, T., Zhang, X., & Wang, S. (2021). GraphSMOTE: Imbalanced Node Classification on Graphs with Graph Neural Networks. *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21 (pp. 833–841). New York, NY, USA: Association for Computing Machinery. Retrieved August 24, 2024, from https://dl.acm.org/doi/10.1145/3437963.3441720

Zhao, Y., Zhu, Y., Yu, Q., & Chen, X. (2021). Cross-Project Defect Prediction Method Based on Manifold Feature Transformation. *Future Internet*, *13*(8), 216. Multidisciplinary Digital Publishing Institute.