pISSN 2597-4602
eISSN 2597-4637

The Effect of Resampling on Classifier Performance: an Empirical Study

Utomo Pujianto [a,1,*], Muhammad Iqbal Akbar [a,2], Niendhitta Tamia Lassela [a,3], Deni Sutaji [b,4]

[a] *Department of Electrical Engineering, Universitas Negeri Malang, Jl. Semarang No. 5, Malang 65145, Indonesia*
[b] *Bilgisayar Bilimleri (Computer Science), Gazi Üniversitesi Emniyet, Milas Sk. No:30, 06560 Yenimahalle/Ankara, Turkey*
[1] *utomo.pujianto.ft@um.ac.id*; [2] iqbal.akbar.ft@um.ac.id; [3] niendhittatamia.1605356@students.um.ac.id;*
[4] *deni.sutaji@gazi.edu.tr*
** corresponding author*

ARTICLE INFO

Article history:
Received 2 January 2021
Revised 25 January 2021
Accepted 4 June 2022
Published online 7 November 2022

Keywords:
Classifier Performance
Resampling Techniques
Paired T-Test

ABSTRACT

An imbalanced class on a dataset is a common classification problem. The effect of using imbalanced class datasets can cause a decrease in the performance of the classifier. Resampling is one of the solutions to this problem. This study used 100 datasets from 3 websites: UCI Machine Learning, Kaggle, and OpenML. Each dataset will go through 3 processing stages: the resampling process, the classification process, and the significance testing process between performance evaluation values of the combination of classifier and the resampling using paired t-test. The resampling used in the process is Random Undersampling, Random Oversampling, and SMOTE. The classifier used in the classification process is Naïve Bayes Classifier, Decision Tree, and Neural Network. The classification results in accuracy, precision, recall, and f-measure values are tested using paired t-tests to determine the significance of the classifier's performance from datasets that were not resampled and those that had applied the resampling. The paired t-test is also used to find a combination between the classifier and the resampling that gives significant results. This study obtained two results. The first result is that resampling on imbalanced class datasets can substantially affect the classifier's performance more than the classifier's performance from datasets that are not applied the resampling technique. The second result is that combining the Neural Network Algorithm without the resampling provides significance based on the accuracy value. Combining the Neural Network Algorithm with the SMOTE technique provides significant performance based on the amount of precision, recall, and f-measure.

I. Introduction

Classification is one of the activities in data mining that aims to group data into a class. In general, a dataset contains two or more class labels. However, most data in a dataset have an unbalanced amount of data between classes. That means one of the classes in the dataset has more data than another class which is called the majority class. The impact of existence of a majority class creates a minority class [1]. The minority class is a class that has less amount of data than the majority class. The occurrence of a majority class and a minority class in the dataset is called class imbalance in the data. When performing the classification process using a dataset with a class imbalance, the majority class will dominate the occurrence of the majority class label so that the result is a decrease in the performance of the classification algorithm [2]. One of the solutions to overcome the class imbalance in the dataset is using a data-level approach, such as resampling and synthesizing data [3]. The purpose of resampling is to support the recognition of minority data to make it more recognizable by the algorithm by adjusting the distribution of minority and majority classes. The class balance in the dataset can be obtained by eliminating the majority class and adding the minority class.

Several examples of methods are included in the resampling technique, such as random undersampling, random oversampling, and SMOTE [4]. Undersampling is a resampling by reducing the data in the majority class. This technique is effective in overcoming class imbalance because a lot of the majority class data is ignored so that the dataset becomes more balanced, and the data training process becomes faster [5]. Oversampling is a resampling technique by adding data to the minority class. Oversampling can add necessary information to minority classes and prevent misclassification [6]. This study will use random undersampling, random oversampling, and SMOTE as resampling techniques to overcome class imbalances in the used dataset.

Related research on resampling showed that random undersampling significantly improves the classification performance of imbalanced classes in Medicare Big Data [7]. The results obtained from the research were random undersampling, which got an AUC score of 97%. Related research on using the random oversampling and SMOOTE technique outperformed the other resampling [8]. Majority-to-Minority Resampling (MMR), a hybrid approach to pick switched instances, adaptively selects potential instances from the majority class to enhance the minority class, showing that the result of the proposed approach outperforms several strong baselines across standard metrics for imbalanced data [9]. Similarity Oversampling and Undersampling Preprocessing (SOUP), which resamples tough cases, outperforms specialist preprocessing methods for multi-imbalanced issues and competes with the most famous decomposition ensembles on natural and artificial datasets [10]. Borderline, Random Over Sampler, SMOTE, SMOTE-ENN, SVM-SMOTE, and SMOTE-Tomek handle imbalanced data and predict pupil success using two datasets using machine learning models like Random Forest, K-Nearest-Neighbor, Artificial Neural Network, XG-boost, Support Vector Machine (Radial Basis Function), Decision Tree, Logistic Regression, and Naïve Bayes [11]. SVM-SMOTE outperforms other resampling methods in the Friedman statistical relevance test. Random Forest was best after SVM-SMOTE resampling.

This study's motivation and new contribution lie in evaluating the resampling algorithm with three different classifiers: Naïve Bayes Classifier, Decision Tree, and Backpropagation Neural Network (BPNN) on 100 public datasets. While the resampling algorithm has been widely used in the literature, its effectiveness in improving classification performance on imbalanced datasets is still an open question, particularly when combined with different classifiers. Furthermore, this study goes beyond simply applying the resampling algorithm and evaluates its impact on classification metrics, including accuracy, precision, recall, and f-measure, which are particularly relevant in resampling applications on classifiers where false negatives and false positives can have significant consequences.

Overall, this study aims to find the significance of resampling on the classifier's performance between the resampled dataset and the classifier performance of the dataset without resampling. To achieve that goal, an empirical study was carried out on 100 datasets by comparing the results of the classifier performance from the dataset without resampling. Each dataset applied resampling techniques: random undersampling, oversampling, and SMOTE. Then the dataset without resampling and the dataset that has been resampled are classified using three different classifiers. The classification metrics values were tested using paired t-tests to find the significance between resampling and combining the classifier with resampling that can provide the most significant results. These findings can inform the development of more effective and reliable machine-learning models for resampling applications on classifier performance. In Section II of this article, the methodologies that were utilized in this research are explained. The findings are presented in Section III, with a discussion of those results and a comparison of relevant models. The last part, the conclusion, can be found in Section IV.

II. Methods

A. Data Collection

This study used 100 datasets obtained online through 3 websites: UCI Machine Learning, Kaggle, and OpenML. Each dataset has a numeric input and a binary class. Most datasets have more than ten attributes and have fewer than 1000 instances. The fields used in the Dataset can be seen in Figure 1.

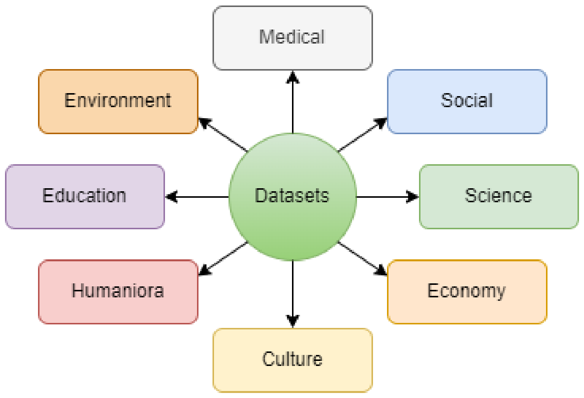


Fig. 1 The fields used in the Dataset

B. *Data Preprocessing*

Two preprocessing steps are carried out in this phase, imputing missing values and resampling. Impute missing values aims to replace the missing values in the dataset with new sample values. In this study, the method used to create new samples is the K-Nearest Neighbor (K-NN) with a k-neighbor of 10 and a Manhattan distance. This method will look for cases that are similar to issues where there are missing values. The two cases are identical if each attribute of the two cases is close together. If a similar case has been found, the attribute with missing values will be filled with the value from the attribute value with a similar case. The K-NN algorithm can provide more robust and more sensitive predictions of missing values [12]. A k-neighbor of 10 to overcome missing values can minimize the error rate when doing classification [13]. Manhattan distance can give ba better results than the other kind of distance (Euclidean Distance, Correlation Distance, and Cosine Distance) [14].

Resampling aims to resample each dataset using random undersampling, random oversampling, and SMOTE with a ratio of 100%. Random undersampling is a technique that removes some randomly selected data to decrease the majority [15]. Random oversampling is an oversampling technique by duplicating randomly selected data to increase the minority [16]. SMOTE is an oversampling technique that creates new synthetic data from some of the closest selected data using the K-NN [17]. SMOTE, with a ratio of 100%, is of a kind feel the payment ratio in the smote where the process of making new samples is carried out until the minority has the same number as the majority.

C. *Data Classification*

Three classifiers used to classify this study are Gaussian Naïve Bayes, Decision Tree with C4.5 Algorithm, and BPNN. Gaussian Naïve Bayes is a kind of Naïve Bayes Classifier that uses the gaussian normal distribution to calculate the probability of each attribute. Naïve Bayes Classifier is a classifier based on Bayes' Theorem with an assumption of independence among features [18]. The normal distribution formula can be shown in (1).

$$(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (1)$$

Where $f(x)$ is the normal distribution of each attribute in each class, σ is the standard deviation for each attribute in each class, μ is the mean value of each attribute in each class, and x is the sample value of each attribute. Here is the pseudocode of Gaussian Naïve Bayes.

```
Input: Training dataset D = {(x1, y1), (x2, y2), ..., (xn, yn)}, where
xi is a feature vector and yi is the corresponding class label.
Output: A trained Gaussian Naive Bayes classifier.
Start
{
1. Calculate the prior probability of each class P(y) as the frequency
   of each class in the training dataset.
2. For each feature i, calculate the mean and standard deviation for each
   class j:
    •  Mean μij = mean(xi|yi=j)
    •  Standard deviation σij = std(xi|yi=j)
3. For a new sample x:
    a. For each class j:
     i. Calculate the likelihood P(x|y=j) using a Gaussian probability
        density function with mean μij and standard deviation σij for each
        feature i.
    ii.Calculate the posterior probability P(y=j|x) using Bayes' theorem:
        P(y=j|x) = P(x|y=j) * P(y=j) / P(x).
    b. Choose the class with the highest posterior probability as the
        predicted class for x.
}
End
```

The Decision Tree Classifier is one classifier that makes branching conditions based on specific attribute values that are done until the branching process cannot be done. There are three parts to the Decision Tree, they are the root node is the main attribute that influences determining class [19], the branch node is the attribute that is selected next after the root node [20], and the leaf node is the class label of each branch that is passed [21]. So that the decision tree structure is similar to a tree structure. This study used the C4.5 Algorithm to determine the branch. The C4.5 Algorithm is a development of the ID3 method, which can provide better accuracy than the ID3 method [22]. The formula to calculate the Gain Ratio can be shown as in (2) to (6).

$$Gain\ Ratio(A) = \frac{Gain(A)}{SplitInfo_A(D)} \tag{2}$$

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|Dj|}{|D|} log_2(\frac{|Dj|}{|D|}) \tag{3}$$

$$Gain(A) = Info(D) - Info_A(D) \tag{4}$$

$$Info_A(D) = \sum_{j=1}^{v} \frac{|Dj|}{|D|} \times Info(D_j) \tag{5}$$

$$Info(D) = -\sum_{i=1}^{m} p_i log_2\,(p_i) \tag{6}$$

Where $Gain\ Ratio(A)$ is the gain ratio value for each split point, $Gain(A)$ is the information gain value for each split point, $SplitInfo_A(D)$ is the split info value for each split point, $Info(D)$ is the overall Entropy value in the dataset, $Info_A(D)$ is the entropy value for each split point, Dj is the number of events at each split point, D is the total number of events at each split point, v is the number of times class label type, A is the split point value and p_i is the probability value for each class. This is the pseudocode for Decision Tree with C4.5 Algorithm.

```
Input: Training dataset D = {(x1, y1), (x2, y2), ..., (xn, yn)}, where
xi is a feature vector and yi is the corresponding class label.
Output: A trained decision tree classifier.
Start
{
1. If all samples in D belong to the same class y, then return a leaf
   node with class y.
2. If the set of features F is empty, then return a leaf node with the
   majority class in D.
3. Calculate the information gain ratio for each feature i in F:
    •  Calculate the entropy H(D) of the current dataset D.
    •  For each possible value v of feature i, calculate the entropy
       H(D|Xi=v) of the subset of D with Xi=v.
    •  Calculate the information gain IGi = H(D) - Σ P(Xi=v) * H(D|Xi=v),
       where P(Xi=v) is the proportion of samples with Xi=v in D.
    •  Calculate the split information Si = - Σ P(Xi=v) * log2(P(Xi=v)).
    •  Calculate the information gain ratio IG_ratioi = IGi / Si.
4. Choose the feature i with the highest information gain ratio IG_ratioi
   as the splitting feature.
5. Create a decision tree node with feature i and its possible values as
   children.
6. For each child node j of the current node:
    •  Let Dj be the subset of D with Xi=j.
    •  If Dj is empty, create a leaf node with the majority class in D.
    •  Otherwise, recursively build the subtree rooted at node j using
       Dj and the remaining features in F - {i}.
7. Return the root node of the decision tree.
}
End
```

Neural Network (NN) is one of the classification algorithms in which the classification is similar to the workings of the human nervous system, the existence of a collection of interconnected neurons used to perform complex learning repeatedly [23]. A NN contains a collection of inputs and outputs connected by a weighted line. The weights are adjusted during the learning phase to help the network on NN make correct class predictions from the input. NN are suitable for applications that require complex learning because NN takes a long time to carry out repeated learning [24] and adjust to empirically determined parameters and network designs [25]. In the NN model, nodes with added weight are on each path, so the NN can learn to handle the wrong datasets. NN conducts several training in one case, so the NN Algorithm has a relatively small error rate and high accuracy [26]. The prediction results are influenced by determining the learning rate value, the target error, the amount of training data used, and the initial weight [27]. The NN model has three parts: the input layer, the hidden layer, and the output layer. The input layer is a layer that contains a collection of input nodes where the input node has the input attribute values. The hidden layer is a layer after the input layer, which contains a collection of hidden nodes where the hidden node contains the input set values that have been processed with weight and bias values. The weight value is a value that states an input priority. The bias value is a constant value contained in each hidden node. The output layer is a layer that contains a collection of output nodes where the output node contains values that have been processed from several hidden nodes and become predictive values.

In this study, the learning method in the NN Algorithm is the Backpropagation method. The Backpropagation method is one of the learning methods in the NN Algorithm, where the learning method adjusts the weights repeatedly in each tuple by changing the weights carried out backward from the output layer to the hidden layer. The purpose of this adjustment is to minimize the Mean Squared Error (MSE). If the MSE is low, the predicted class with the actual class has similarities. There are two processes in the Backpropagation Neural Network (BPNN). They are feedforward which is the process of calculating each value in the input attribute carried out forward from the input layer to the output layer, and backward is the process of calculating the error value between the value on the output layer and the target value. The error value adjusts the weight until the error value has a smaller number than the target error [28]. The following steps in the BPNN can be shown as in (7).

$$I_j = \sum_i w_{ij} O_i + w\theta_j \theta_j \tag{7}$$

Equation (8) is used to calculate the new input value for each unit in the hidden layer and the output layer where I_j is the new input value for each unit in the hidden layer and output layer, w_{ij} is the weight value from unit i in the previous layer to unit j, O_i is the output value from the previous layer. If the calculation is done for the first time, then the value of O_i is the value of the input layer, $w\theta_j$ is the bias weight for each unit, and θ is the bias value for each unit. Then calculate the new output in each unit in the hidden layer and the output layer using the formula as in (8).

$$O_j = \frac{1}{1+e^{-I_j}} \tag{8}$$

Where O_j is the output value in unit j and I_j is the input value in j-unit. Next, calculate the error value used as a stop condition using the MSE calculation formula as in (9).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(X_i - S_i)^2 \tag{9}$$

Where X_i is the output value in the dataset, S_i is the output value calculated in the previous layer, and n is the class number. To perform the backward pass process, the first thing to do is to calculate the total error against the weight of each unit using the formula as in (10).

$$\frac{ET}{w_{ij}} = \frac{ET}{O_j} \times \frac{O_j}{I_j} \times \frac{I_j}{w_{ij}} \tag{10}$$

Where $\frac{ET}{O_j}$ _i is the backward pass value of the total error against each unit in the hidden layer and output layer, $\frac{O_j}{I_j}$ is the backward pass value of the output of each unit on the output layer, the hidden layer is the input of each unit in the output layer and the hidden layer, and $\frac{I_j}{w_{ij}}$ is the backward pass value of the input from each unit in the output layer and the hidden layer against the weight connected to each unit. Then update the weight using the formula as in (11).

$$w_{ij}new = w_{ij}old - (l \times \frac{ET}{w_{ij}}) \tag{11}$$

Where $w_{ij}new$ is the weight value of the new unit ij , $w_{ij}old$ is the weight value of the old ij unit, $\frac{ET}{w_{ij}}$ is the total error value for the weight in each unit, and l is the learning rate value. This is the pseudocode for the BPNN.

```
Input: Training dataset D = {(x1, y1), (x2, y2), …, (xn, yn)}, where xi
is a feature vector and yi is the corresponding class label.
Output: A trained BPNN classifier.
Start
{
1. Initialize the weights and biases of the neural network randomly.
2. For each training sample (x, y) in D, do the following steps:
   a. Forward pass:
    i.   Calculate the output y' of the neural network for input x by
         applying the weights and biases to each neuron using the
         activation function.
    ii.  Calculate the error δ for each neuron in the output layer as δj
         = y'j(1-y'j)(yj-y'j), where yj is the desired output for neuron
         j.
    iii. Calculate the error δ for each neuron in the hidden layers using
         the chain rule: δj = yj(1-yj)Σ wjk δk, where wjk is the weight
         from neuron k to neuron j, and δk is the error for neuron k.
   b. Backward pass:
    i.   Update the weights and biases of the neural network using the
         error δ and the learning rate α as follows:
       • For each weight wjk from neuron k to neuron j: wjk = wjk + αδjyk
       • For each bias bj of neuron j: bj = bj + αδj
3. Repeat step 2 for a fixed number of epochs or until the error on the
   validation set stops improving.
4. Return the trained BPNN classifier.
}
End
```

D. Evaluation

The first evaluation is using the classification matrix. The matrix is performance evaluation metrics calculated from the confusion matrix. A confusion matrix is a table with as many row and column dimensions as the number of classes in the dataset used to analyze the performance of the classification algorithm. The confusion matrix is used as an evaluation of how good the quality of classifier performance. The confusion matrix has four components: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). TP is the number of positive class data that is classified correctly. TN is the number of negative class data that is classified correctly. FP is the number of negative class data that is incorrectly predicted as a positive class. FN is the number of positive class data incorrectly predicted as negative [29]. The four values are used to find the algorithm performance evaluation value: accuracy, precision, recall, and f-measure. The formula to calculate the accuracy, precision, recall, and f-measure as in (12) to (15) [30].

$$Accuracy = \left(\frac{TP+TN}{TP+TN+FP+FN}\right) \times 100\% \tag{12}$$

$$Precision = \left(\frac{TP}{TP+FP}\right) \times 100\% \tag{13}$$

$$Recall = \left(\frac{TP}{TP+FN}\right) \times 100\% \tag{14}$$

$$F - Measure = \left(\frac{2 \times precision \times recall}{precision+recall}\right) \times 100\% \tag{15}$$

The second evaluation used the paired t-test used scipy, a python language-based library, in this study. To perform a paired t-test, namely by calculating the t value between the values compared using the formula can be shown as in (16) [31].

$$t = \frac{\bar{d}}{S_d} \times \sqrt{n} \tag{16}$$

Where t is the t-statistic value used to determine the significance between 2 values, d is the difference value of 2 samples, S_d is the value of standard deviation, $\bar{d}$ is the mean value of the difference between 2 samples, and n is the number of instances. In the paired t-test, there are two hypotheses, are H_0 which means that there is no significant difference between the two values being compared and H_1 there is a significant difference between the two values being compared. To determine whether H_0 and H_1 are accepted, an alpha value is required. The alpha value used in this study is 5%. If the t-statistic value is less than 5%, then H_0 is rejected and $H_1$is accepted. If the t-statistic value is more than 5%, then H_0 is accepted and $H_1$is rejected.

III. Results and Discussion

There are two results obtained from this study. The first result is the performance evaluation of each classifier combination with resamplings, such as accuracy, precision, recall, and f-measure. The second result is paired t-test results from resampling in general and a combination of the classifier with resampling based on accuracy, precision, recall, and f-measure values. The performance evaluation results of each classifier combination with resampling techniques are shown in Figure 2 to Figure 4.

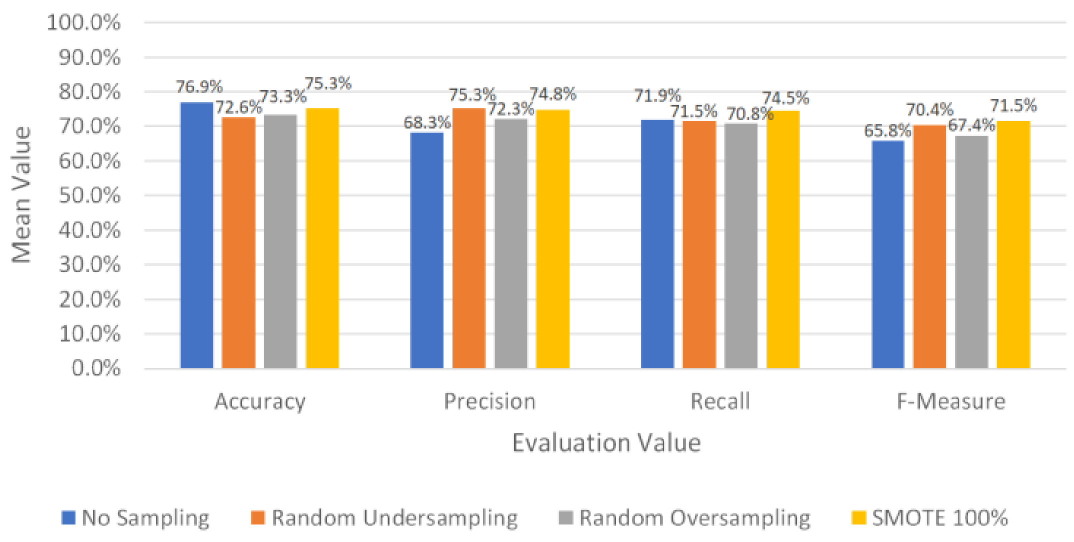


Fig. 2. Evaluation of Gaussian Naïve Bayes with each resampling technique for each type of evaluation value is based on the mean

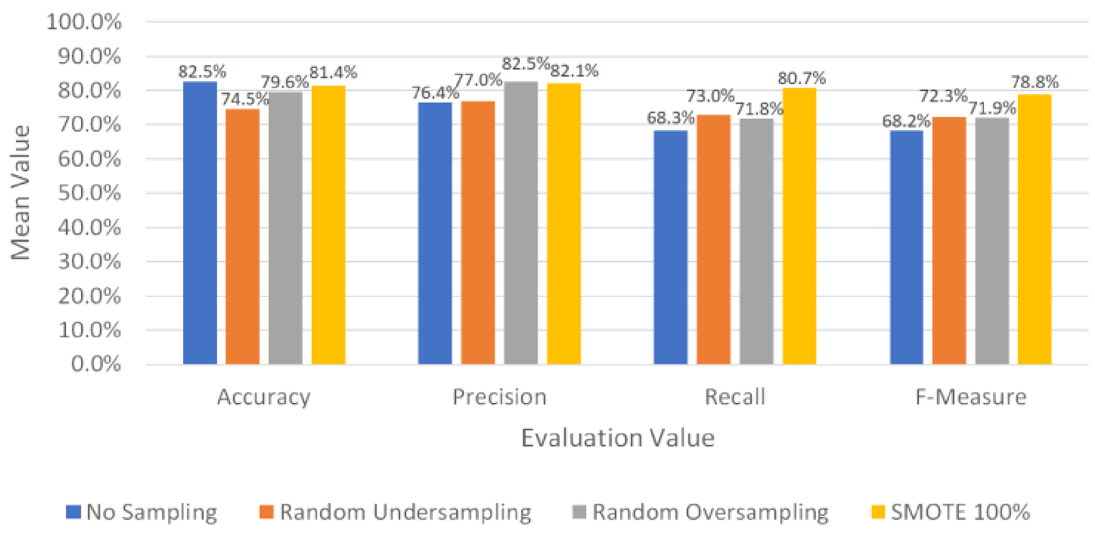


Fig. 3. Evaluation of Decision Tree with C4.5 Algorithm with each resampling technique for each type of evaluation value is based on the mean

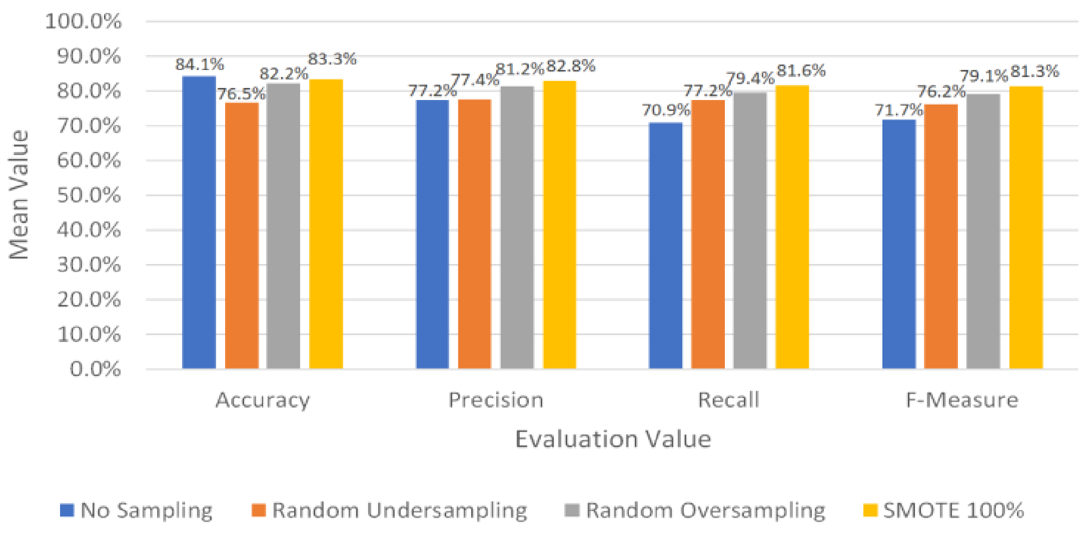


Fig. 4. Evaluation of the Backpropagation Neural Network with each resampling technique for each type of evaluation value is based on the mean

Based on the results, the classification results without resampling of each algorithm give the best performance than the other three resampling techniques in accuracy. SMOTE gives better results based on the recall and f-measure values of the three classification algorithms. Based on the precision value, the use of resampling gives different results for each algorithm, where Random Undersampling gives the best performance of the precision value on the Gaussian Naïve Bayes Algorithm, Random Oversampling provides the best performance of the precision value in the C4.5 Decision Tree Algorithm, and SMOTE provides the best performance of the precision value in the BPNN Algorithm.

Combining the BPNN Algorithm with SMOTE provides the best performance. This is because SMOTE can provide new samples so the classification algorithm can learn more data patterns. SMOTE provides better performance than random undersampling and random oversampling because, in random undersampling, there is a possibility that essential data will be lost due to the random data deletion process so that the classifier cannot recognize more varied patterns and can cause a decrease in the performance of the classifier. Meanwhile, random oversampling provides the same data the due to the random duplicating data so that it can lead to overfitting, where the classifier has better performance because it predicts correctly on the same data so that when classifying the new data, the classifier will misclassify the class because the classifier does not recognize new data patterns. It can cause a decrease in the performance of the classifier.

The classification results from the unresampled datasets give better accuracy but give lower precision, recall, and f-measure than the other three resampling techniques because classification using unresampled datasets can provide overfitting classification results, where the classification algorithm has better performance due to the classification algorithm predicts correctly on the majority data so that when classifying new data which should be a minority class, the classification algorithm will misclassify into the majority class. This is because the classification algorithm does not learn minority data well and is better at recognizing data patterns in the majority class. The impact of this is a decrease in the performance of the classification algorithm because the FP and FN values become high because of classification errors.

The BPNN can give the best performance than Gaussian Naïve Bayes and Decision Tree with C4.5 Algorithm because, in the Gaussian Naïve Bayes, the process of calculating the probability of each attribute in each class uses a gaussian distribution, so class determination is very dependent on the mean and the standard deviation value for each attribute. If the mean and standard deviation values are more significant in a class, the algorithm will be more likely to determine a new class with a higher mean and standard deviation value. Meanwhile, in the Decision Tree with C4.5 Algorithm, there is a possibility that the entropy value of 0 will appear during the process of calculating the number of classes so that the Decision Tree model will immediately determine the class based on the attribute that has the number of 0 on a split point, causing a class determination at the beginning of the branching because there is the possibility of entropy is 0. So class determination is only determined by one attribute. This can reduce the algorithm's performance because other attributes are not chosen that can influence class determination. In the BPNN, there is a process of calculating the error between the prediction results and the actual class and adjusting the weight and as that can support more optimal classification results. The t-test results from resampling, in general, can be shown in Table 1.

To determine the significance of the results, this study used a z value of 1.960 for T-Paired and an α value for P-Paired as the threshold for determining the hypothesis. If the test results between the two resampling techniques have a T-Paired value less than the z value and a P-Paired value more than α, then the two resampling techniques do not provide significant results. The paired t-test results above gave three scenarios a yellow highlighter with a T-Paired value less than the z value and the P-Paired value more than the α value. So those three scenarios did not provide significant results.

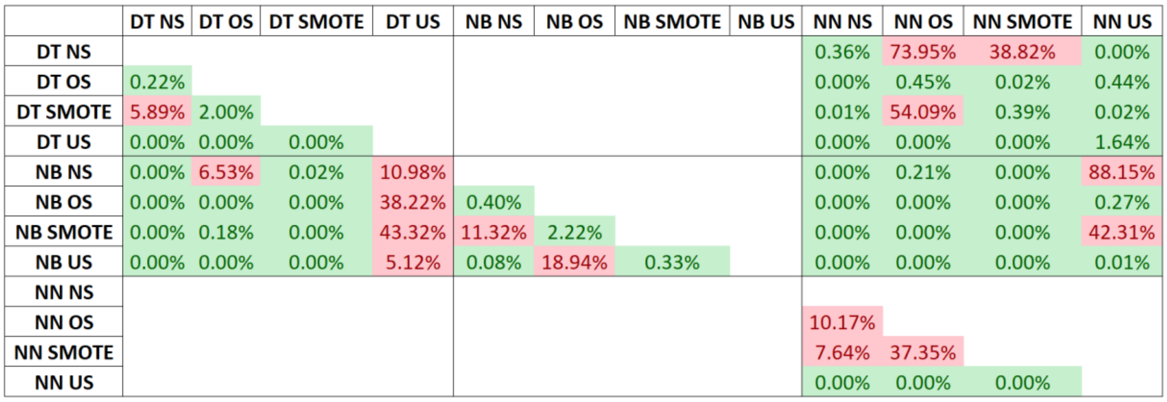
Table 1. T-Test result based on resampling in general

T-Test Scenario Name	T-Paired	P-Paired
T-test between No Sampling and Oversampling based on the accuracy value	4.44832	1.22E-05
T-test between No Sampling and SMOTE based on the accuracy value	2.948332	0.003447
T-test between No Sampling and Undersampling based on the accuracy value	9.447194	1.04E-18
T-test between Oversampling and No Sampling based on the value of precision	4.254278	2.81E-05
T-test between Oversampling and No Sampling based on the recall value	2.080658	0.038316
T-test between Oversampling and No Sampling based on the F-Measure value	2.818378	0.005149
T-test between Oversampling and Undersampling based on the accuracy value	8.783319	1.26E-16
T-test between Oversampling and Undersampling based on the value of Precision	3.069031	0.002344
T-test between Oversampling and Undersampling based on the recall value	0.102836	0.918162
T-test between SMOTE and No Sampling based on Precision values	7.728664	1.66E-13
T-test between SMOTE and No Sampling based on the recall value	6.42767	5.11E-10
T-test between SMOTE and No Sampling is based on the F-Measure value	7.733038	1.62E-13
T-test between SMOTE and Oversampling based on the accuracy value	3.050948	0.002486
T-test between SMOTE and Oversampling based on the value of Precision	1.791596	0.074209
T-test between SMOTE and Oversampling based on the recall value	4.182419	3.79E-05
T-test between SMOTE and Oversampling based on the F-Measure value	4.729757	3.47E-06
T-test between SMOTE and Undersampling based on the accuracy value	9.088606	1.42E-17
T-test between SMOTE and Undersampling based on the value of Precision	4.278243	2.54E-05
T-test between SMOTE and Undersampling based on the recall value	5.115083	5.61E-07
T-test between SMOTE and Undersampling based on the F-Measure value	5.05701	7.44E-07
T-test between Undersampling and No Sampling based on the value of Precision	2.405361	0.016764
T-test between Undersampling and No Sampling based on the recall value	2.529396	0.01194
T-test between Undersampling and No Sampling based on the F-Measure value	3.487873	0.00056
T-test between Undersampling and Oversampling based on the F-Measure value	0.25086	0.802095
	z = 1,960	α = 0,05

The following result, the paired t-test, is a test between 2 combinations of classification algorithms with resampling techniques. The next paired t-test is a test between 2 combinations of classification algorithms with resampling techniques. The red column shows that the p-value is more than 5%, and the green column shows that the p-value is less than 5%. The following is the abbreviation of the combination name in the paired t-test result:

- NB NS: The combination of the Naïve Bayes Algorithm without resampling
- NB OS: The combination of the Naïve Bayes Algorithm with Random Oversampling
- NB SMOTE: The combination of the Naïve Bayes Algorithm with SMOTE
- NB US: The combination of the Naïve Bayes Algorithm with Random Undersampling
- DT NS: The combination of Decision Tree Algorithm without resampling
- DT OS: The combination of Decision Tree Algorithm with Random Oversampling
- DT SMOTE: The combination of the Decision Tree Algorithm with SMOTE
- DT US: The combination of Decision Tree Algorithm with Random Undersampling
- NN NS: The combination of Neural Network Algorithms without resampling
- NN OS: The combination of Neural Network Algorithm with Random Oversampling
- NN SMOTE: The combination of Neural Network Algorithm with SMOTE
- NN US: The combination of Neural Network Algorithm with Random Undersampling

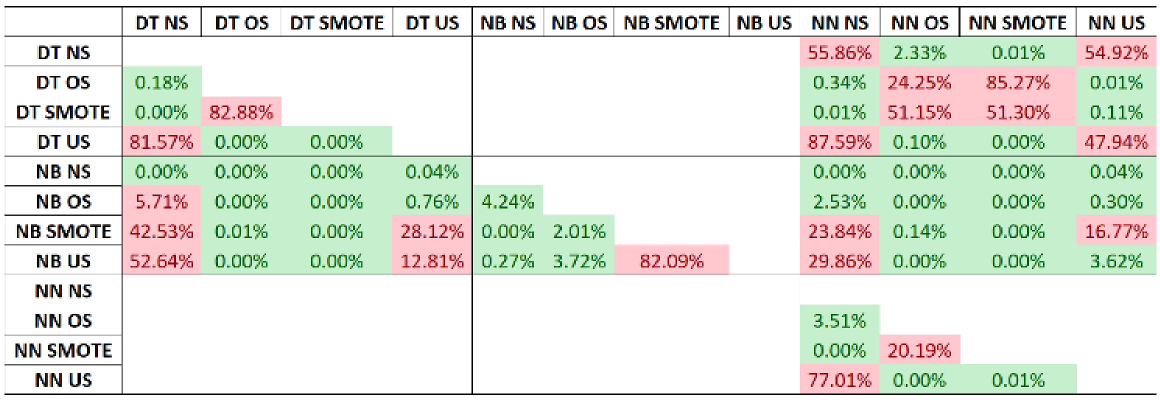
Paired t-test results between 2 combinations of classification and resampling algorithms can be shown in Figure 5 to Figure 8.



	DT NS	DT OS	DT SMOTE	DT US	NB NS	NB OS	NB SMOTE	NB US	NN NS	NN OS	NN SMOTE	NN US
DT NS									0.36%	73.95%	38.82%	0.00%
DT OS	0.22%								0.00%	0.45%	0.02%	0.44%
DT SMOTE	5.89%	2.00%							0.01%	54.09%	0.39%	0.02%
DT US	0.00%	0.00%	0.00%						0.00%	0.00%	0.00%	1.64%
NB NS	0.00%	6.53%	0.02%	10.98%					0.00%	0.21%	0.00%	88.15%
NB OS	0.00%	0.00%	0.00%	38.22%	0.40%				0.00%	0.00%	0.00%	0.27%
NB SMOTE	0.00%	0.18%	0.00%	43.32%	11.32%	2.22%			0.00%	0.00%	0.00%	42.31%
NB US	0.00%	0.00%	0.00%	5.12%	0.08%	18.94%	0.33%		0.00%	0.00%	0.00%	0.01%
NN NS												
NN OS									10.17%			
NN SMOTE									7.64%	37.35%		
NN US									0.00%	0.00%	0.00%	

Fig. 5. Result of paired t-test between a combination of classifier and resampling based on accuracy values

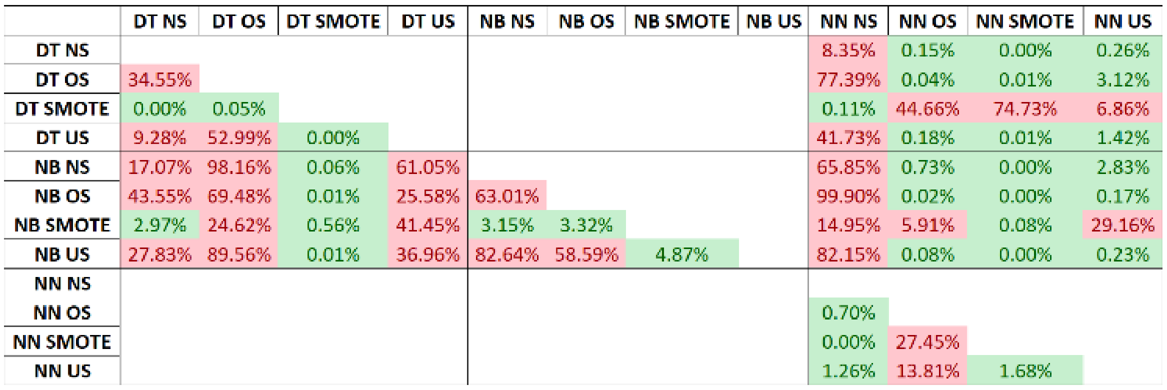
Paired t-test results based on accuracy values as shown in Figure 5, the combination of NN Algorithms without resampling gives the most significant results than the other 11 combinations. Meanwhile, the Gaussian Naïve Bayes with random undersampling does not give good results. Figure 5 shows that 5 out of 9 tests comparing the classification results from using the resampling technique and without applying the resampling technique to each algorithm give significant results.



	DT NS	DT OS	DT SMOTE	DT US	NB NS	NB OS	NB SMOTE	NB US	NN NS	NN OS	NN SMOTE	NN US
DT NS									55.86%	2.33%	0.01%	54.92%
DT OS	0.18%								0.34%	24.25%	85.27%	0.01%
DT SMOTE	0.00%	82.88%							0.01%	51.15%	51.30%	0.11%
DT US	81.57%	0.00%	0.00%						87.59%	0.10%	0.00%	47.94%
NB NS	0.00%	0.00%	0.00%	0.04%					0.00%	0.00%	0.00%	0.04%
NB OS	5.71%	0.00%	0.00%	0.76%	4.24%				2.53%	0.00%	0.00%	0.30%
NB SMOTE	42.53%	0.01%	0.00%	28.12%	0.00%	2.01%			23.84%	0.14%	0.00%	16.77%
NB US	52.64%	0.00%	0.00%	12.81%	0.27%	3.72%	82.09%		29.86%	0.00%	0.00%	3.62%
NN NS												
NN OS									3.51%			
NN SMOTE									0.00%	20.19%		
NN US									77.01%	0.00%	0.01%	

Fig. 6. Result of paired t-test between a combination of classifier and resampling based on precision values

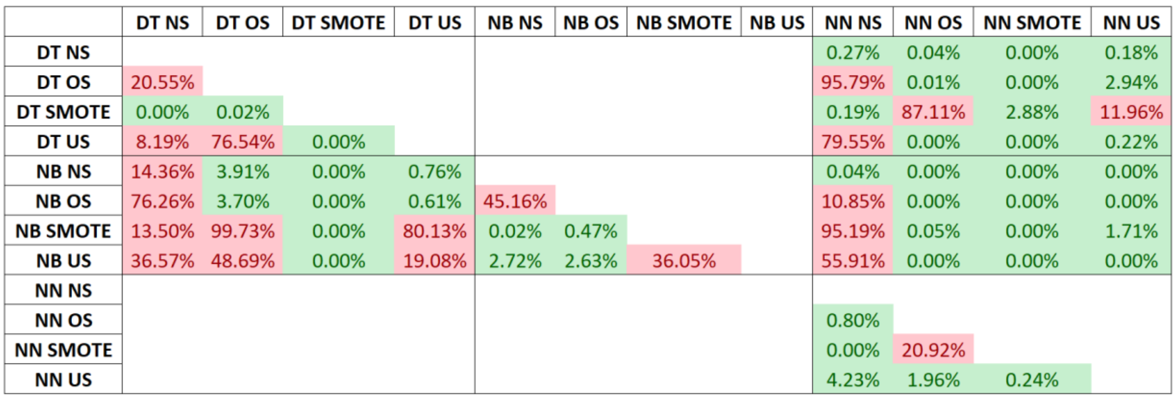
Paired t-test results based on precision values, as shown in Figure 6, the combination of NN Algorithms without resampling gives the most significant results than the other 11 combinations. Meanwhile, the Gaussian Naïve Bayes without resampling does not give good results. Figure 6 shows that 7 out of 9 tests comparing the classification results from using the resampling technique and without applying the resampling technique to each algorithm give significant results.



	DT NS	DT OS	DT SMOTE	DT US	NB NS	NB OS	NB SMOTE	NB US	NN NS	NN OS	NN SMOTE	NN US
DT NS									8.35%	0.15%	0.00%	0.26%
DT OS	34.55%								77.39%	0.04%	0.01%	3.12%
DT SMOTE	0.00%	0.05%							0.11%	44.66%	74.73%	6.86%
DT US	9.28%	52.99%	0.00%						41.73%	0.18%	0.01%	1.42%
NB NS	17.07%	98.16%	0.06%	61.05%					65.85%	0.73%	0.00%	2.83%
NB OS	43.55%	69.48%	0.01%	25.58%	63.01%				99.90%	0.02%	0.00%	0.17%
NB SMOTE	2.97%	24.62%	0.56%	41.45%	3.15%	3.32%			14.95%	5.91%	0.08%	29.16%
NB US	27.83%	89.56%	0.01%	36.96%	82.64%	58.59%	4.87%		82.15%	0.08%	0.00%	0.23%
NN NS												
NN OS									0.70%			
NN SMOTE									0.00%	27.45%		
NN US									1.26%	13.81%	1.68%	

Fig. 7. Result of paired t-test between a combination of classifier and resampling based on recall values

Paired t-test results based on recall, as shown in Figure 7, the combination of NN Algorithms without resampling gives the most significant results than the other 11 combinations. Meanwhile, the Decision Tree C4.5 Algorithm without resampling does not give good results. Figure 7 shows that 5 out of 9 tests comparing the classification results from using the resampling technique and without applying the resampling technique to each algorithm give significant results.



	DT NS	DT OS	DT SMOTE	DT US	NB NS	NB OS	NB SMOTE	NB US	NN NS	NN OS	NN SMOTE	NN US
DT NS									0.27%	0.04%	0.00%	0.18%
DT OS	20.55%								95.79%	0.01%	0.00%	2.94%
DT SMOTE	0.00%	0.02%							0.19%	87.11%	2.88%	11.96%
DT US	8.19%	76.54%	0.00%						79.55%	0.00%	0.00%	0.22%
NB NS	14.36%	3.91%	0.00%	0.76%					0.04%	0.00%	0.00%	0.00%
NB OS	76.26%	3.70%	0.00%	0.61%	45.16%				10.85%	0.00%	0.00%	0.00%
NB SMOTE	13.50%	99.73%	0.00%	80.13%	0.02%	0.47%			95.19%	0.05%	0.00%	1.71%
NB US	36.57%	48.69%	0.00%	19.08%	2.72%	2.63%	36.05%		55.91%	0.00%	0.00%	0.00%
NN NS												
NN OS									0.80%			
NN SMOTE									0.00%	20.92%		
NN US									4.23%	1.96%	0.24%	

Fig. 8. Result of paired t-test between a combination of classifier and resampling based on accuracy values

Paired t-test results based on accuracy values as shown in Figure 8, the combination of NN Algorithms without resampling gives the most significant results than the other 11 combinations. Meanwhile, the Gaussian Naïve Bayes without resampling does not give good results. Figure 8 shows that 6 out of 9 tests comparing the classification results from using the resampling technique and without applying the resampling technique to each algorithm give significant results.

IV. Conclusion

Based on the results and discussion of the research that has been done, it can be concluded that. The BPNN with SMOTE performs best based on accuracy, precision, recall, and f-measure. The mean and paired t-test values are better than the 11 combinations of classification algorithms and other resampling techniques. The combination of the classification algorithm and the resampling technique does not provide significant results based on the type of evaluation: (1) based on the accuracy, combining the Gaussian Naïve Bayes with random undersampling does not provide the most significant performance results; (2) based on precision, and f-measure, combining the Gaussian Naïve Bayes without resampling does not provide the most significant performance results; (3) combining the Decision Tree C4.5 Algorithm without resampling does not provide the most significant performance results based on recall. Using resampling can provide significant results on the classification algorithm's performance compared to the classification algorithm's performance on the dataset without resampling. It is shown that most of the test results from comparing classification results from datasets that apply resampling techniques and from datasets without resampling techniques give significant results. However, combining multiple resampling techniques may improve classification performance even further. Future research could explore the effectiveness of combining different resampling techniques and the impact on classification performance.

Declarations

Author contribution

All authors contributed equally as the main contributor of this paper. All authors read and approved the final paper.

Funding statement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflict of interest

The authors declare no known conflict of financial interest or personal relationships that could have appeared to influence the work reported in this paper.

Additional information

Reprints and permission information are available at http://journal2.um.ac.id/index.php/keds.

Publisher's Note: Department of Electrical Engineering - Universitas Negeri Malang remains neutral with regard to jurisdictional claims and institutional affiliations.

References

[1] F. Thabtah, S. Hammoud, F. Kamalov, and A. Gonsalves, "Data imbalance in classification: Experimental evaluation," *Inf. Sci. (Ny).*, vol. 513, pp. 429–441, Mar. 2020.

[2] A. Ali-Gombe and E. Elyan, "MFC-GAN: Class-imbalanced dataset classification using Multiple Fake Class Generative Adversarial Network," *Neurocomputing*, vol. 361, pp. 212–221, Oct. 2019.

[3] U. Pujianto, "Random forest and novel under-sampling strategy for data imbalance in software defect prediction," *Int. J. Eng. Technol.*, vol. 7, no. 4, pp. 39–42, 2018.

[4] T. Chen, Y. Lu, X. Fu, N. N. Sze, and H. Ding, "A resampling approach to disaggregate analysis of bus-involved crashes using panel data with excessive zeros," *Accid. Anal. Prev.*, vol. 164, p. 106496, Jan. 2022.

[5] B. Mirzaei, B. Nikpour, and H. Nezamabadi-pour, "CDBH: A clustering and density-based hybrid approach for imbalanced data classification," *Expert Syst. Appl.*, vol. 164, p. 114035, Feb. 2021.

[6] C. Zhang *et al.*, "Over-Sampling Algorithm Based on VAE in Imbalanced Classification," in *Lecture Notes in Computer Science (LNISA,volume 10967)*, 2018, pp. 334–344.

[7] J. Hancock, T. M. Khoshgoftaar, and J. M. Johnson, "The Effects of Random Undersampling for Big Data Medicare Fraud Detection," in *2022 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, Aug. 2022, pp. 141–146.

[8] R. Zhou *et al.*, "Prediction Model for Infectious Disease Health Literacy Based on Synthetic Minority Oversampling Technique Algorithm," *Comput. Math. Methods Med.*, vol. 2022, pp. 1–6, Mar. 2022.

[9] G. Wang, J. Wang, and K. He, "Majority-to-minority resampling for boosting-based classification under imbalanced data," *Appl. Intell.*, vol. 53, no. 4, pp. 4541–4562, Feb. 2022.

[10] M. Janicka, M. Lango, and J. Stefanowski, "Using Information on Class Interrelations to Improve Classification of Multiclass Imbalanced Data: A New Resampling Algorithm," *Int. J. Appl. Math. Comput. Sci.*, vol. 29, no. 4, pp. 769–781, Dec. 2019.

[11] R. Ghorbani and R. Ghousi, "Comparing Different Resampling Methods in Predicting Students' Performance Using Machine Learning Techniques," *IEEE Access*, vol. 8, pp. 67899–67911, 2020.

[12] S. Saeed, A. Abdullah, N. Z. Jhanjhi, M. Naqvi, and A. Nayyar, "New techniques for efficiently k-NN algorithm for brain tumor detection," *Multimed. Tools Appl.*, vol. 81, no. 13, pp. 18595–18616, May 2022.

[13] H. Xu and Y. Chen, "A block padding approach in multidimensional dependency missing data," *Eng. Appl. Artif. Intell.*, vol. 120, p. 105929, Apr. 2023.

[14] H. A. Abu Alfeilat *et al.*, "Effects of Distance Measure Choice on K-Nearest Neighbor Classifier Performance: A Review," *Big Data*, vol. 7, no. 4, pp. 221–248, Dec. 2019.

[15] J. Li, S. Fong, S. Hu, R. K. Wong, and S. Mohammed, "Similarity Majority Under-Sampling Technique for Easing Imbalanced Classification Problem," in *Communications in Computer and Information Science*, 2018, pp. 3–23.

[16] J. Fonseca, G. Douzas, and F. Bacao, "Improving Imbalanced Land Cover Classification with K-Means SMOTE: Detecting and Oversampling Distinctive Minority Spectral Signatures," *Information*, vol. 12, no. 7, p. 266, Jun. 2021.

[17] Z. Shi, "Improving k-Nearest Neighbors Algorithm for Imbalanced Data Classification," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 719, no. 1, p. 012072, Jan. 2020.

[18] N. Salmi and Z. Rustam, "Naïve Bayes Classifier Models for Predicting the Colon Cancer," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 546, no. 5, p. 052068, Jun. 2019.

[19] S. Wahyuni, "Implementation of Data Mining to Analyze Drug Cases Using C4.5 Decision Tree," *J. Phys. Conf. Ser.*, vol. 970, p. 012030, Mar. 2018.

[20] T. Thomas, A. P. Vijayaraghavan, and S. Emmanuel, "Applications of Decision Trees," in *Machine Learning Approaches in Cyber Security Analytics*, Singapore: Springer Singapore, 2020, pp. 157–184.

[21] R. Benkercha and S. Moulahoum, "Fault detection and diagnosis based on C4.5 decision tree algorithm for grid connected PV system," *Sol. Energy*, vol. 173, pp. 610–634, Oct. 2018.

[22] G. S. Reddy and S. Chittineni, "Entropy based C4.5-SHO algorithm with information gain optimization in data mining," *PeerJ Comput. Sci.*, vol. 7, p. e424, Apr. 2021.

[23] I. Gonzalez-Fernandez, M. A. Iglesias-Otero, M. Esteki, O. A. Moldes, J. C. Mejuto, and J. Simal-Gandara, "A critical review on the use of artificial neural networks in olive oil production, characterization and authentication," *Crit. Rev. Food Sci. Nutr.*, vol. 59, no. 12, pp. 1913–1926, Jul. 2019.

[24] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5455–5516, Dec. 2020.

[25] X. Qi, G. Chen, Y. Li, X. Cheng, and C. Li, "Applying Neural-Network-Based Machine Learning to Additive Manufacturing: Current Applications, Challenges, and Future Perspectives," *Engineering*, vol. 5, no. 4, pp. 721–729, Aug. 2019.

[26] H. Dagdougui, F. Bagheri, H. Le, and L. Dessaint, "Neural network model for short-term and very-short-term load forecasting in district buildings," *Energy Build.*, vol. 203, p. 109408, Nov. 2019.

[27] Y. Wu, R. Gao, and J. Yang, "Prediction of coal and gas outburst: A method based on the BP neural network optimized by GASA," *Process Saf. Environ. Prot.*, vol. 133, pp. 64–72, Jan. 2020.

[28] J. C. R. Whittington and R. Bogacz, “Theories of Error Back-Propagation in the Brain,” *Trends Cogn. Sci.*, vol. 23, no. 3, pp. 235–250, Mar. 2019.
[29] D. Chicco, N. Tötsch, and G. Jurman, “The Matthews correlation coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation,” *BioData Min.*, vol. 14, no. 1, p. 13, Feb. 2021.
[30] J. Miao and W. Zhu, “Precision–recall curve (PRC) classification trees,” *Evol. Intell.*, vol. 15, no. 3, pp. 1545–1569, Sep. 2022.
[31] G. Mahalle, O. Salunke, N. Kotkunde, A. K. Gupta, and S. K. Singh, “Neural network modeling for anisotropic mechanical properties and work hardening behavior of Inconel 718 alloy at elevated temperatures,” *J. Mater. Res. Technol.*, vol. 8, no. 2, pp. 2130–2140, Apr. 2019.