

# Under-sampling technique for imbalanced data using minimum sum of euclidean distance in principal component subset

Chatchai Kasemtaweekok<sup>1</sup>, Worasait Suwannik<sup>2</sup>

<sup>1</sup>Department of Computer Science and Information, Faculty of Science at Sriracha, Kasetsart University, Chonburi, Thailand

<sup>2</sup>Department of Computer Science, Faculty of Science, Kasetsart University, Bangkok, Thailand

## Article Info

### Article history:

Received Jan 18, 2023

Revised Apr 26, 2023

Accepted May 7, 2023

### Keywords:

Classification algorithms

Data preprocessing

Hybrid methods

Imbalanced data

Sampling methods

## ABSTRACT

Imbalanced datasets are characterized by a substantially smaller number of data points in the minority class compared to the majority class. This imbalance often leads to poor predictive performance of classification models when applied in real-world scenarios. There are three main approaches to handle imbalanced data: over-sampling, under-sampling, and hybrid approach. The over-sampling methods duplicate or synthesize data in the minority class. On the other hand, the under-sampling methods remove majority class data. Hybrid methods combine the noise-removing benefits of under-sampling the majority class with the synthetic minority class creation process of over-sampling. In this research, we applied principal component (PC) analysis, which is normally used for dimensionality reduction, to reduce the amount of majority class data. The proposed method was compared with eight state-of-the-art under-sampling methods across three different classification models: support vector machine, random forest, and AdaBoost. In the experiment, conducted on 35 datasets, the proposed method had higher average values for sensitivity, G-mean, the Matthews correlation coefficient (MCC), and receiver operating characteristic curve (ROC curve) compared to the other under-sampling methods.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



## Corresponding Author:

Chatchai Kasemtaweekok

Department of Computer Science and Information, Faculty of Science at Sriracha, Kasetsart University

199 Moo 6, Sukhumvit Road, Tung Sukla, Sriracha, Chonburi, Thailand

Email: chatchai.ka@ku.th

## 1. INTRODUCTION

Data mining and machine learning rely on data to train models and make predictions. Thus, the characteristic of data significantly impacts the performance of machine learning algorithms. An imbalanced dataset has unequal distribution of classes. There are a small amount of important data represented as positive class and a large amount of low importance data in the negative class. This characteristic is frequently observed in real-world scenarios, such as disease diagnosis [1], fraud monitoring [2], computer network intrusion [3], and credit risk [4]. Model trained from imbalanced data sets is unable to effectively classify the minority class data.

There are three approaches that can be used to address the problem of imbalanced data [5]–[8]: cost-sensitive, algorithm, and sampling. The first approach, cost-sensitive approach, involves assigning different cost to different class during the training. Misclassifying instances from the minority class has higher cost. As a result, cost-sensitivity methods will increase the accuracy of the minority class data so that it is higher than for the accuracy of the majority class data.

The second approach, known as algorithm approach, improves the algorithms currently in use to deal with imbalanced data by adjusting classification rules to introduce biased against a minority class. Algorithm

approaches can be divided into two categories: one-side learning and ensemble learning [9]. The first type, one-side learning algorithms, train a model for only one class. For example, a one-sided fuzzy support vector machines based on sphere method introduces a hypersphere reduction approach by finding minimal hypersphere of the majority class to reduce the majority noises [10]. Another example is weighted one class support vector machine method, where the algorithm assigns higher weight to prediction of the minority class to minimize the influence of the majority class on support vector machine class classification [11]. The second type is ensemble learning algorithms. These methods are applied as bagging and boosting algorithms to improve classification performance in imbalanced data sets such as random under-sampling (RUS) with a boosting algorithm [12], AdaBoost with clustering algorithm [13], fuzzy rough set theory [14], and bagging techniques with replacement with the segregation of majority and minority classes [15].

The third approach, sampling approach, involves changing the size of either majority or minority class to achieve a balanced dataset. There are three types of sampling methods: over-sampling, under-sampling, and hybrid techniques. The oversampling methods add to the synthesized training data by randomly generating a minority data of the attributes from samples in the minority class. The synthetic minority over-sampling technique (SMOTE) algorithm is a popular over-sampling method [16]. This method reconstructs the minority by randomly interpolating between two neighboring minority points. The SMOTE can be improved by regenerating the minority class only in the border region of minority clusters [17] or in the highest safe area of the minority class point [18]. Some over-sampling methods combine several clustering techniques to create minority clusters and then add a new minority class such as DBSCAN, clustering using representatives (CURE)-SMOTE [19], k-means SMOTE [20], radius-SMOTE [21], and Gaussian Kernels with diagonal smoothing matrices [22].

Under-sampling methods remove majority data from the training data set using a variety of criteria such as decision boundary, the redundant majority class groups, and the majority class groups that are close to or overlapping with the minority class groups. The Tomek link method (TML) [23] removes majority classes when nearby data is noise or border data. The one-side selection method (OSS) [24] deletes majority classes when they are redundant data that is far from the data border using Hart's condensing technique or borderline data that is close to boundary between majority and minority regions using Tomek link technique. The neighborhood cleaning rule method (NCR) [25] was introduced to screen most of the data classes minus Wilson's editing techniques [26] that find and remove noisy data from the dataset. The instance hardness threshold method (IHT) [27] was presented for deleting most of the hardness data that are a misclassified from multiple models. The hardness of a data point can be determined based on three metrics: k disagreeing neighbors, class likelihood, and class likelihood difference. Finally, the NearMiss method (NRM) [28] was proposed to remove majority class data that are close to minority class data. The method described above uses the nearest neighborhood principle to find majority class data that are close to minority class data or a majority class that is far from other majority class groups as the decision criterion for reducing the size of the data.

Some under-sampling algorithms use clustering techniques to group majority class, and then applying the majority reduction. A cluster-based instance selection that combines the use of affinity propagation and k-means techniques to cluster the majority class into subsets and to select representatives from the subsets using some algorithms: genetic algorithm, instance based learning, and incremental reduction optimization procedure. The clustering-based undersampling method selects k cluster centers from k-mean algorithm [29]. The DBIG-US algorithm [30] is a two-step process of selecting the majority class clusters. The density-based spatial clustering of applications with noise (DBSCAN) algorithm screens the noisy majority class in the first step. The second step selects most representatives of the data using a graph-based procedure. The balanced data set derived from the DBIG-US algorithm increases the geometric mean of the classification. The undersampling framework with denoising, fuzzy c-means clustering, and representative sample selection (UFFDFR) algorithm uses three techniques [31], where the first step removes noisy, boundary, and redundant data in majority clusters with Tomek links. The second step clusters the majority data based on fuzzy c-means clustering. The last step selects the most representatives using the max-min concept.

Hybrid methods combine the benefits of both under-sampling and over-sampling by combining the process of removing noise in the majority classes with the synthetic minority class creation process. Bagging and boosting ensemble methods have been proposed to optimize the screening of majority classes or to create new minority classes [32], [33]. The SMOTETomek method [34] uses the SMOTE method to equalize the minority of the data with the majority, and then uses the Tomek link technique to reduce noise and decision border data. A balanced data selection method expands the efficiency of discriminating the minority class through filtering and transforming the noisy majority class to a minority class with relabeling and amplification techniques [35]. A clustering and density-based hybrid (CDBH) method segments the minority class using the k-means clustering algorithm [36].

In general, the sampling methods are independent of classifier and have a wide range of applications. It has been reported that under-sampling methods are more efficient than over-sampling methods because the

latter tend to cause over-fitting [37]. However, under-sampling methods can exclude some useful majority class information from the training data.

To overcome this limitation, this study presents a method for selecting representative instances from the majority class using principal component analysis (PCA), which is widely used for reducing dimensions and data clustering [38]–[40]. PCA transforms all numerical columns into principal component (PC) columns, resulting in a reduced number of columns compared to the original dataset. The primary goal of this analysis is to preserve the data's variability through new PC axes derived from eigenvectors and eigenvalues of the covariance matrix. PC columns are sorted in descending order based on the variance of the projected data. PC1 represents the axis with the highest variance, while the subsequent PCs represent progressively lower amounts of variance [41]–[45].

After the transformation, the majority class representatives in each partition (i.e., subset of instances) are selected from the data with the smallest sum of Euclidean distances between other data sets within the same partition. The number of the partition is equal to the number of minority instances. The training data sets derived from the proposed algorithms are compared with the results from six currently used algorithms using 35 data sets, based on the predictive efficiency of five indicators: sensitivity, specificity, G-mean, Matthew's correlation coefficient (MCC) and receiver operating characteristic curve (ROC curve).

**2. METHOD**

This study introduces a new under-sampling method to keep only majority classes which have the minimum sum of the Euclidean distance principal component value (PC-MIN) for imbalanced data classification. The method leverages the minimum sum of Euclidean distances within a principal component subset (i.e., a column-wise subset in the transformed space). By applying this algorithm, the number of instances belonging to the majority classes in the training dataset can be significantly reduced, approaching the quantity of instances in the minority class groups. Following the under-sampling process, the resulting training dataset with balanced classes is utilized to train three classification models, with performance evaluation of classification using the imbalanced test data set, as shown in Figure 1.

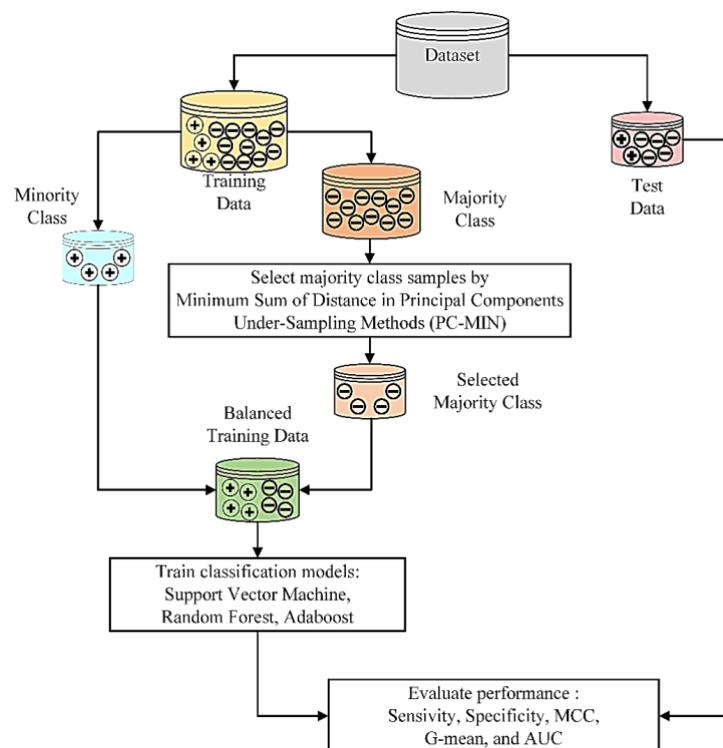


Figure 1. Process flow of PC-MIN method

In algorithm 1, the PC-MIN pseudocode has three input variables: majority data (MJ), minority data (MN), and variance ratio (VR). The seven steps of the algorithm can be divided into two parts: data transformation and selection of representative majority instances. The first four steps belong to the first part. The specific pseudo

code for each function is omitted due to its well-established nature. In our implementation, we simply call functions from a library. The 1<sup>st</sup> step, `Normalize_Standard_Scale` normalizes the values in each column to the standard z-score. The 2<sup>nd</sup> step, `Transform_PC_Data`, converts the z-score to PC columns. The 3<sup>rd</sup> step, `Get_PC_Columns`, calculates the number of PC columns based on variance ratio within the range of 0 to 1. The 4<sup>th</sup> step, `Create_PC_Table`, creates a data table using the number of PC columns obtained from the previous step.

The second part of Algorithm 1 selects representative of majority instances from each partition. `Sort_Data` sorts the majority class instances in ascending order based on their PC values (i.e., starting with PC1, then PC2, and so on). After that, `Split_MajorityData_into_Partition` divides the majority class data into disjoint partitions based on the order of sorted instances.

#### Algorithm 1. PC-MIN

Input: *MJ*: Majority Data, *MN*: Minority Class, *VR*: Variance Ratio

Output: *BS*: Balanced Dataset

```

1: SDF = Normalize_Standard_Scale (MJ)
2: PCD = Transform_PC_Data (SDF)
3: PC_COLS = Get_PC_Columns (PCD, VR)
4: PC_DF = Create_PC_Table (PCD, PC_COLS)
5: PC_DF = Sort_Data(PC_DF)
6: PDF = Split_MajorityData_Into_Partition (PC_DF, Size(MN))
7: RS = [ ]
8: for i = 1 to Size(MN) do
9: REP = Get_Representative (PDFi)
10: Append REP to RS
11: end for
12: BS = Create_Balance_Dataset (RS, MN)
13: return BS

```

Next, `Get_Representative` selects majority class representatives of each partition. This selection is based on identifying data points with a minimum distance from other data points within the same partition. Algorithm 2 shows the pseudocode for `Get_Representative`, which takes the PC columns table (*PDF<sub>i</sub>*) as input. First, a square metric table, *pdist*, is created by calculating pairwise Euclidean distance between *n* data points as shown in Table 1. Then, a list of total distance to all instances in the same partition, *sumdist*, is calculated. In the set of data points in the partition *PDF<sub>i</sub>*, the criteria for selecting representatives can be expressed using (1). The final step of Algorithm 1 creates a balanced dataset by combining representative of majority data with minority data. The balanced dataset (BS) is returned as a result by the PC-MIN algorithm.

$$Representative = \arg \min_{x_i, y \in PDF_i} \sum_{i=1}^n d(x_i - y) \quad (1)$$

#### Algorithm 2. Get\_Representative

Input: *PDF<sub>i</sub>*: a set of majority data in partition *i*

Output: *REP*: a representative of partition *i*

```

1: pdist = Create_PairwiseDistances_Metric(PDFi)
2: mindist = MaxFloatValue()
3: for i = 1 to Size(PDFi) do
4: if sumdist(i) < mindist
5: REP = Get_Majority_Data(i)
6: mindist = sumdist(i)
7: end if
8: end for
9: return REP

```

Figure 2 visualizes how to partition and select representatives in PC-MIN algorithm. The algorithm takes advantage of PCA to reduce the dimensionality of the data. Figure 2(a) shows the dataset that was transformed from high dimension to 2 dimensions, PC1 and PC2, from the six datasets: `abalone`, `libras_move`, `ozone_level`, `wine_quality`, `breast-cancer`, and `colon-cancer`. In this experiment, the PC-MIN algorithm uses only the 2-dimensional PC to capture the essence of higher dimensions in 19 datasets. In Figure 2(b), the representatives have minimum sum of distance in each partition. To ensure clarity and avoid confusion, we have chosen a subset of the dataset and limited the number of partitions to fewer than 20.

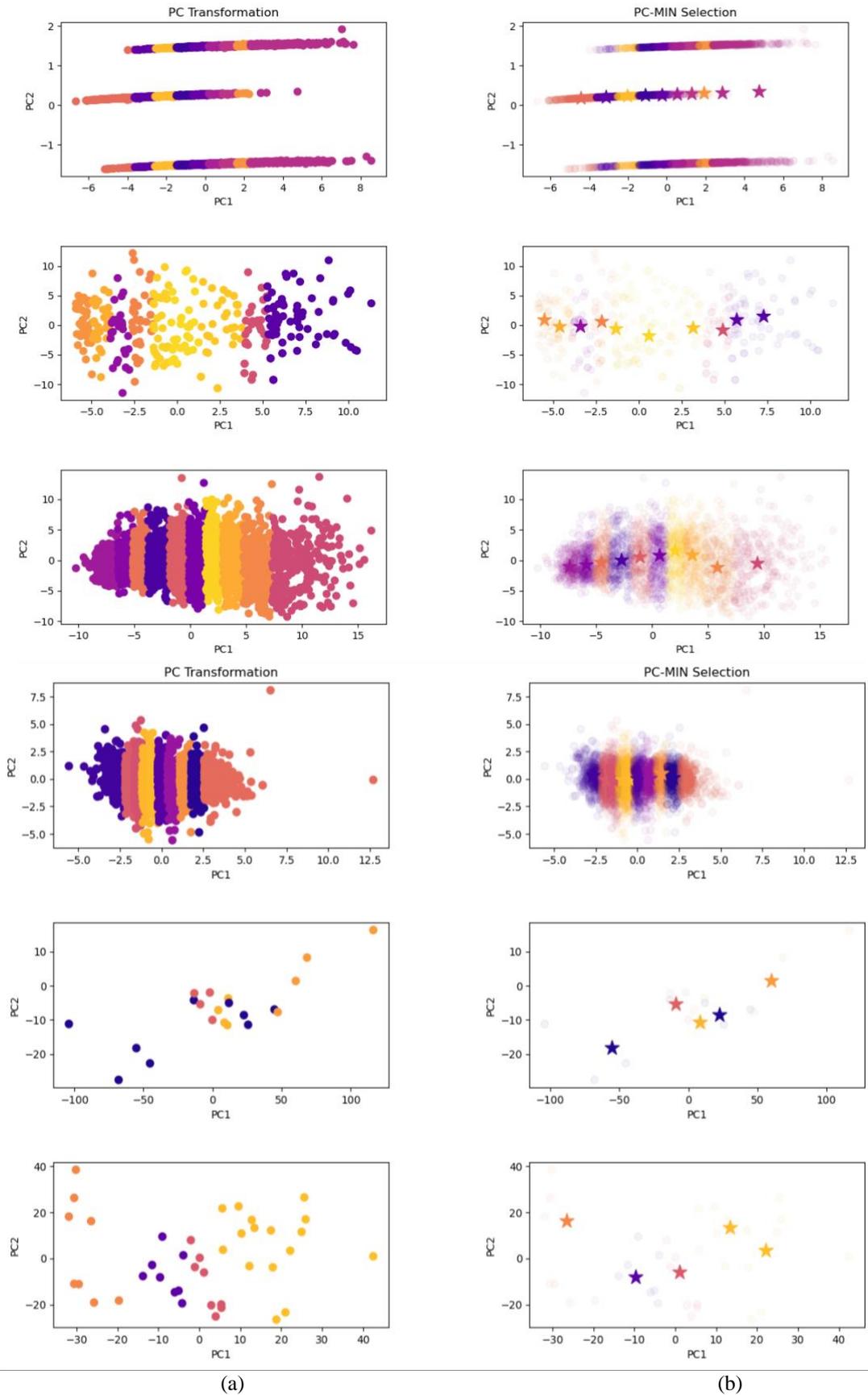


Figure 2. Scatter plots of PC1 and PC2 for (a) data points of assigned partitions (colored dot) and (b) representative (colored star) of the six datasets

Table 1. Pairwise distances between n data points in m -dimensional space

Distance metric	$x_1$	$x_2$	...	$x_i$	...	$x_n$	Sumdist
$y_1$	$d(x_1 - y_1)$	$d(x_2 - y_1)$	...	$d(x_i - y_1)$	...	$d(x_n - y_1)$	$Sumdist(1) = \sum_{i=1}^n d(x_i - y_1)$
$y_2$	$d(x_1 - y_2)$	$d(x_2 - y_2)$	...	$d(x_i - y_2)$	...	$d(x_n - y_2)$	$Sumdist(2) = \sum_{i=1}^n d(x_i - y_2)$
...	...	...	...	...	...	...	...
$y_i$	$d(x_1 - y_i)$	$d(x_2 - y_i)$	...	$d(x_i - y_i)$	...	$d(x_n - y_i)$	$Sumdist(i) = \sum_{i=1}^n d(x_i - y_i)$
...	...	...	...	...	...	...	...
$y_n$	$d(x_1 - y_n)$	$d(x_2 - y_n)$	...	$d(x_i - y_n)$	...	$d(x_n - y_n)$	$Sumdist(n) = \sum_{i=1}^n d(x_i - y_n)$

### 3. RESULTS AND DISCUSSION

A test was conducted to determine the performance of the PC-MIN algorithm based on comparisons with six currently used algorithms obtained from the imbalanced-learn library [46]: IHT, NRM, NCR, OSS, RUS, and TML. Table 2 lists the undersampling algorithms compared and the parameters used in testing each method. Additionally, the baseline model was trained from the original imbalanced data set (FULL) to be normalized in this experiment.

Table 2. List of compared methods

Methods	Parameters
IHT	sampling_strategy = auto, random_state = 42, cv = 5, estimator = RandomForestClassifier
NRM	n_neighbors = 3
NCR	threshold_cleaning = 0.5, n_neighbors = 3, kind_sel = all
OSS	random_state = 42, n_seeds = 1
RUS	sampling_strategy = auto, random_state = 42, replacement = False
TML	random_state = 42

The PC-MIN algorithm that appears in algorithm 1 has three input variables: majority class (MS), variance ratio (VR), and minority class (MR). The two variables MS and MR have values and data items that depend on the data set used in the test and as such cannot be adjusted. Therefore, in this experiment, the PC-MIN algorithm was adjusted for the required coverage variance (VR) using five values: 0.5 (PC-MIN50), 0.4 (PC-MIN40), 0.3 (PC-MIN30), 0.2 (PC-MIN20), and 0.1 (PC-MIN10).

Table 3 shows the details of the 35 test data sets. The data were sorted by data set name [47]–[49]. In Table 3, the fifth column shows the number of principal component columns used by the PC-MIN30 algorithm. In almost all of the datasets in this experiment, this number is less than 10 columns, with the exceptions of the gisette and madelon datasets. For high dimensionality performance tests, we tested methods performance using the six data sets [50]: breast-cancer [51], colon-cancer [52], gisette [53], leukemia [54], madelon [53], and w1a [55]. The current study used a 5-folds cross-validation test. The resulting data sets of each method were used to train three classification models (support vector machine, random forest, and AdaBoost).

In this study, the performance of the classification models trained with training data from all methods was compared based on five indicators [56]. The first two sensitivity (recall) or true positive rate (TPR) measured the effectiveness of the classification of the model in each class. The positive class classification accuracy rate, was calculated using (2). The specificity or true negative rate (TNR) metric indicates the rate of accuracy in classifying negative classes, as shown in (3). The false positive rate (FPR) is the proportion of all negatives that be predicted positive by model, as shown in (4).

In this experiment, positive classes were represented as minority class groups and negative classes were majority class groups.

$$sensitivity = \frac{TP}{(TP+FN)} \quad (2)$$

$$specificity = \frac{TN}{(TN+FP)} \quad (3)$$

$$false\ positive\ rate = \frac{FP}{(FP+TN)} \quad (4)$$

Where true positive (TP) is the number of correctly predicted responses in the positive class, false positive (FP) is the number of incorrectly predicted response in the positive class, true negative (TN) is the number of correctly predicted responses in the negative class, and false negative (FN) is the number of incorrectly predicted responses in the negative class. Additionally, this study used three indicators to assess the efficacy of a balanced classification model that weighed two classes. The first was the Matthews correlation coefficient

(MCC) is a measure of association for true and predicted values, as shown in (5). The G-means indicator is commonly used in this area of research and is defined as the average between sensitivity and specificity as shown in (6). The measure of the ROC curve shows the probability that a classification model predicts a true-positive class is higher than for a false-positive class, as shown in (7).

$$MCC = \frac{(TP \cdot TN) - (FP \cdot FN)}{\sqrt{(FP+TP)(TP+FN)(TN+FP)(TN+FN)}} \quad (5)$$

$$G - mean = \sqrt{sensitivity \times specificity} \quad (6)$$

$$ROC \text{ curve} = \frac{1 + TPR - FPR}{2} \quad (7)$$

To compare the performance based on MCC, G-means, and A, we used a t-test between the PC-MIN algorithm's average and the compared algorithm. This was divided into two hypotheses. In the benchmark comparison test, MCC, G-means, and ROC curve test assumed the null hypothesis  $H_0$ : the averages of the PC-MIN algorithm and the compared algorithm are not significantly different. The alternative hypothesis was  $H_1$ : the average for the PC-MIN algorithm is higher than the average for the compared algorithm at a significance level of 0.1.

Table 3. Data set information

Dataset	Imbalance ratio	Number of samples	Number of features	PC-MIN30 columns
abalone	9.7:1	4,177	10	2
abalone19	130:1	4,177	10	2
arrhythmia	17:1	452	278	5
breast-cancer	4.2:1	26	7,129	2
car_eval34	12:1	1,728	21	5
car_eval4	26:1	1,728	21	5
coil_2000	16:1	9,822	85	6
colon_cancer	8:1	45	2,000	2
credit_card	577.8:1	284,807	29	8
drunk	26.4:1	595,212	57	9
ecoli	8.6:1	336	7	2
gisette	100:1	3,030	5,000	53
isolet	12:1	7,797	617	3
letter_img	26:1	20,000	16	2
leukemia	3.8:1	24	7,129	2
libras_move	10:1	360	90	2
madelon	12:1	1,728	20,958	62
mammography	42:1	11,183	6	2
oil	22:1	937	49	2
optical_digits	9.1:1	5,620	64	4
ozone_level	34:1	2,536	72	2
pen_digits	9.4:1	10,992	16	2
protein_homo	111:1	145,751	74	2
satimage	9.3:1	6,435	36	2
scene	13:1	2,407	294	3
sick_euthyroid	9.8:1	3,163	42	2
solar_flare_m0	19:1	1,389	32	3
spectrometer	11:1	531	93	2
thyroid_sick	15:1	3,772	52	4
us_crime	12:1	1,994	100	2
w1a	649:1	3,900	300	10
webpage	33:1	34,780	300	8
wine_quality	26:1	4,898	11	2
yeast_me2	28:1	1,484	8	2
yeast_ml8	13:1	2,417	1,031	5

### 3.1. Individual class classification performance

In Figure 3 shows the average sensitivity and specificity of the PC-MIN algorithm and the eight algorithms compared in the three models. The average sensitivity of the models obtained from the four PC-MIN algorithms was greater than 80%, which was higher than for all the methods compared in Figure 3(a). These results indicated that the classification models obtained from all the PC-MIN algorithms could predict the positive class data with an accuracy of more than 80% of the total number of positive class data. Figure 3(b) shows the average specificity of the PC-MIN algorithm and the compared algorithms. The average specificity of the PC-MIN algorithm was less than the values for the FULL, NCR, OSS, and TML algorithms.

The results showed that the negative class data were correctly classified by the PC-MIN algorithm for approximately 80% of all negative class data. The average sensitivity values of the FULL, NCR, OSS, and TML methods were lower than 50% while these models had high specificity rates of 90%. The NCR, OSS, and TML methods removed few majority classes so the number of majority classes from these methods was not different from that of the original dataset (FULL). Due to the substantially larger size of the majority class data, the specificity rates of the NCR, OSS, and TML algorithms were much higher than their sensitivity rates.

Otherwise, the NRM, IHT, and RUS methods achieved a balance between sensitivity and specificity. RUS had the second-highest sensitivity rate, while also having a high specificity rate. The average sensitivity and specificity values of the PC-MIN algorithm were greater than those for the RUS, IHT, and NRM methods. Considering both values simultaneously as shown in Figure 4, all PCMIN algorithms and the RUS, IHT, and NRM algorithms provided sensitivity close to the specificity, while the PC-MIN30 algorithm provided a higher level of sensitivity than those of the IHT, NRM, and RUS algorithms by approximately 15%, 10%, and 3%, respectively. Furthermore, the remaining three algorithms (NCR, OSS, and TML) provided much higher specificity than sensitivity because they selected data sets that caused the classification model to weigh the accuracy of the negative class predictions over the positive class predictions based on the asymmetric nature of the data set.

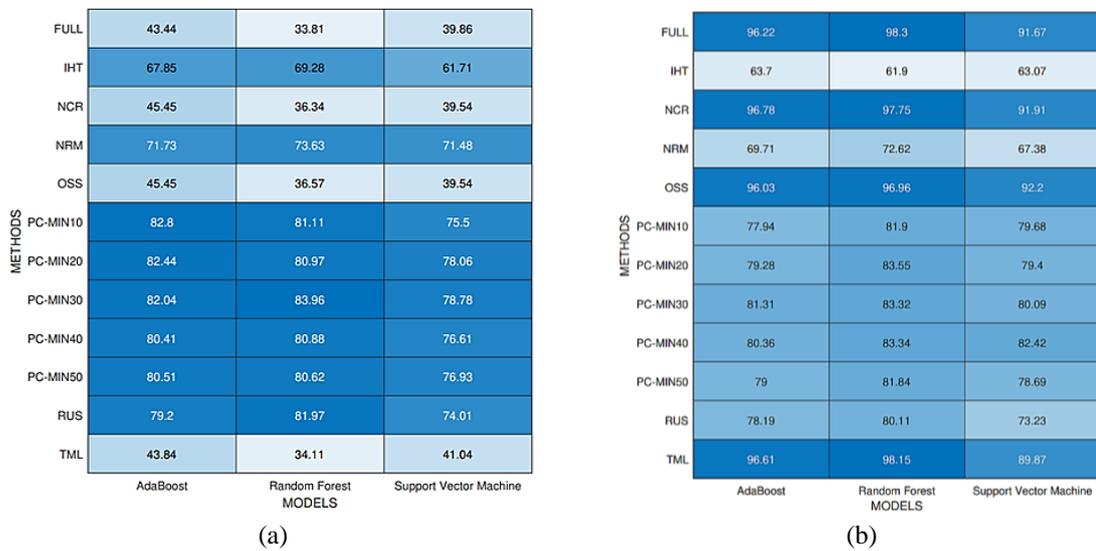


Figure 3. Average (a) sensitivity and (b) specificity of PC-MIN and compared methods using three classification models

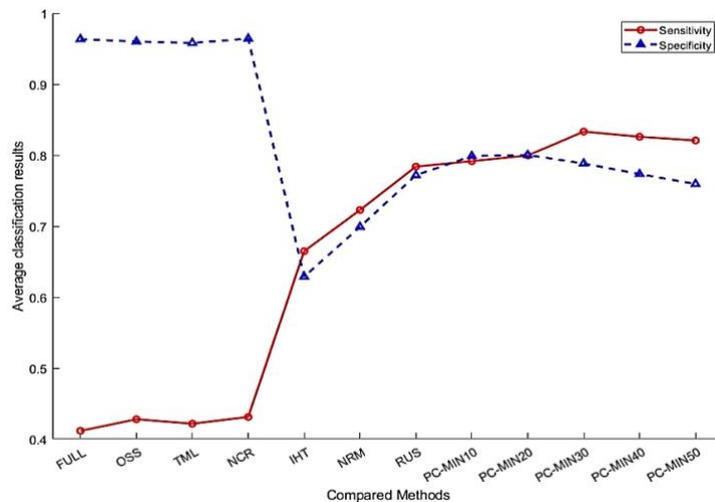


Figure 4. Average sensitivity and specificity comparison of PC-MIN30 and compared methods

Figure 5(a) shows that the PC-MIN30 algorithm had the highest sensitivity among all the algorithms evaluated. In contrast, Figure 5(b) shows that the specificity distribution of the FULL, NCR, OSS, and TML methods had very small variances and high values, while the specificity of the PC-MIN algorithms was lower. However, the PC-MIN30 algorithm had a better balance between both sensitivity and specificity than the other algorithms.

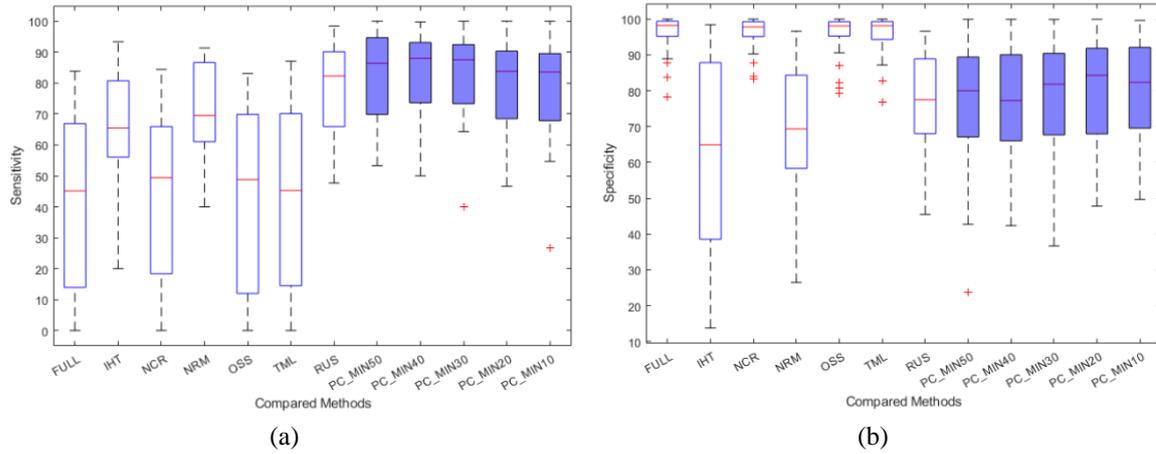


Figure 5. The distributions of compared methods for (a) sensitivity and (b) specificity

**3.2. Overall classification performance**

Figure 6 shows the average MCC value of the PC-MIN algorithm was higher than the average of the other algorithms for the three classification models, except for the NCR, OSS, and TML algorithms using the AdaBoost model. The model obtained from the PC-MIN algorithm was able to give a higher association of true and predicted values than the other algorithms. Figure 7 shows the average G-Mean of the PC-MIN algorithm was higher than the other algorithms. These results indicated that the classification models based on the PC-MIN method could predict both classes with a higher balance than the other algorithms. Figure 8 shows the average ROC curve value of the PC-MIN algorithm was higher than the average of the other algorithms for the three classification models. The model obtained from the PC-MIN algorithm had a higher ratio of true positive rate prediction to the false positive rate than the other algorithms.

METHODS	MODELS		
	AdaBoost	Random Forest	Support Vector Machine
FULL	0.43	0.37	0.37
IHT	0.25	0.29	0.2
NCR	0.44	0.38	0.35
NRM	0.26	0.32	0.29
OSS	0.44	0.38	0.35
PC-MIN10	0.38	0.41	0.37
PC-MIN20	0.38	0.43	0.39
PC-MIN30	0.42	0.47	0.41
PC-MIN40	0.4	0.44	0.4
PC-MIN50	0.39	0.41	0.36
RUS	0.37	0.42	0.32
TML	0.44	0.37	0.35

Figure 6. Average values for MCC of PC-MIN and compared methods using three classification models

METHODS	MODELS		
	AdaBoost	Random Forest	Support Vector Machine
FULL	48.6	41.09	40.71
IHT	38.85	42.97	35.29
NCR	49.61	41.8	40.27
NRM	37.41	43.28	39.74
OSS	49.77	42.09	40.43
PC-MIN10	45.1	48.32	44.48
PC-MIN20	45.52	49.55	46.35
PC-MIN30	48.28	52.56	47.97
PC-MIN40	47.05	50.08	47.25
PC-MIN50	45.94	47.31	44.16
RUS	45.15	49.37	42.37
TML	48.85	40.36	41.15

Figure 7. Average G-mean of PC-MIN and compared methods using three classification models

Among the three classification models from the PC-MIN30 algorithm, the averages for random forest were higher than for the AdaBoost and support vector machine models. This difference is indicative of the random forest model's effective utilization of selected data, resulting in 10% more accurate decision trees compared to those generated by the random forest using the FULL. Similarly, the support vector machine model using the PC-MIN30 algorithm showed classification performance that was 5% superior to that of the support vector machine using the FULL. However, the performance of the AdaBoost model using the PC-MIN30 algorithm did not differ significantly from that of the AdaBoost model using the FULL. Consequently, the under-sampling algorithm provided a smaller training dataset that facilitated the creation of decision trees capable of more efficient classification compared to the other two models.

METHODS	MODELS		
	AdaBoost	Random Forest	Support Vector Machine
FULL	0.7	0.66	0.67
IHT	0.66	0.66	0.63
NCR	0.71	0.67	0.67
NRM	0.71	0.73	0.7
OSS	0.71	0.67	0.66
PC-MIN10	0.8	0.82	0.78
PC-MIN20	0.81	0.82	0.79
PC-MIN30	0.82	0.84	0.79
PC-MIN40	0.8	0.82	0.8
PC-MIN50	0.8	0.81	0.78
RUS	0.79	0.81	0.74
TML	0.7	0.66	0.67

Figure 8. Average ROC curve of PC-MIN and compared methods using three classification models

The average G-mean of all algorithms are presented in Figure 9(a). The PC-MIN30 algorithm clearly had smaller G-mean distribution boxes than the other methods. This could be attributed to the fact that the classification models based on the PC-MIN30 algorithm provided more stable sensitivity and specificity metrics than did the other algorithms across all test datasets. Specifically, the PC-MIN30 algorithm had the highest ROC curve, with lower variance in Figure 9(b), indicating that the balanced datasets generated by the PC-MIN30 algorithm improved the predictive performance in the positive class of classification models.

Table 4 shows the average values for MCC, G-mean, and ROC curve of the classification models using the PC-MIN30 algorithm were significantly greater than those of the classification models based on the other algorithms. From these statistical comparison results, PC-MIN30 provides higher MCC, G-mean, and ROC curve values than almost compared algorithms, except for MCC of NRM. These results showed that the classification models from PC-MIN30 correctly classified most of the minority classes while the majority prediction rate was high. When handling high-dimensional datasets, PC-MIN30 had lower average specificity values than FULL, NCR, and TML; however, for other metrics it had higher performance values than those for other under-sampling methods, and FULL as shown in Table 5.

Three algorithms could be used to reduce the small amount of majority class data and create an imbalanced dataset: NCR, TML, and OSS. The NCR algorithm screens most of the data classes minus Wilson's editing techniques that find and remove noisy data from the dataset. The TML algorithm removes majority classes when nearby data are noise or border data. The OSS algorithm removes majority classes when they are redundant data that are far from the data border (using Hart's condensing technique) or borderline data that are close to the boundary between majority and minority regions (using the TML technique). By using these techniques, The NCR, OSS, and TML algorithms can remove a few majority classes, so the number of majority classes from these methods is not different from that of the original dataset (FULL). Due to the substantially larger size of the majority class data, the specificity rates of the NCR, OSS, and TML algorithms were much higher than their sensitivity rates. For this reason, the MCC and G-mean values of these algorithms were nearly equal to that of PC-MIN30. On the other hand, their AUC values were lower than that of PC-MIN30.

The remaining compared algorithms created balanced datasets by reducing most of the majority class data. First, the IHT method deleted most of the hardness data that were misclassified by multiple models. Second, the NRM method removed majority class data close to minority class data. The method described above used the nearest neighborhood principle to find majority class data close to minority class data or a majority class that was far from other majority class groups as the decision criterion for reducing the size of the data. Both methods balanced datasets by removing majority class data near the decision boundary based on the nearest neighbor. This technique may work well with dense majority class groups. The MCC and G-mean values of both methods were lower than those of PC-MIN and other methods due to low sensitivity and specificity. However, their AUC values were comparable to those of other methods except for PC-MIN30.

Lastly, RUS create a balanced dataset by randomly selecting majority class data without replacement. In RUS, each majority data point was chosen with an equal and fair probability. Due to high sensitivity and specificity, the G-mean and AUC values of RUS were high compared to the other methods because the random selection process of RUS selected majority class data that were widely spread over the distribution of the majority data. However, all performance indicators of RUS were quite lower than those of PC-MIN30 because RUS may have encountered limitations when dealing with some high-dimensional datasets. By using PCA to reduce the dimensionality of the data, only the first few principal components are retained, and a lower-dimensional plane is identified that captures the information of the dataset. This results in closer proximity of the PC majority data partitions, which reduces the complexity of majority data selection. The small size of data in selected PC columns could capture the information of the majority of the data so the PC-MIN30 algorithm conveniently selected the appropriate set of majority class representatives.

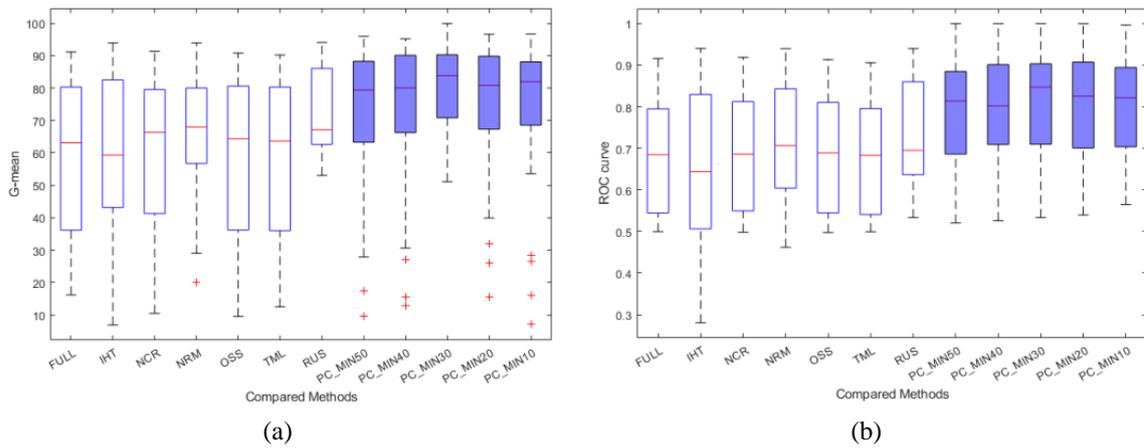


Figure 9. The distributions of compared methods for (a) G-mean and (b) ROC curve

Table 4. Results of the t-test for MCC, G-mean, and ROC curve comparisons

Compared methods	MCC		G-mean		ROC curve	
	p-value	Result	p-value	Result	p-value	Result
PC-MIN30 vs. FULL	0.0774	Reject H <sub>0</sub>	4.9607e-10	Reject H <sub>0</sub>	2.1558e-11	Reject H <sub>0</sub>
PC-MIN30 vs. IHT	3.3326e-05	Reject H <sub>0</sub>	9.9963e-10	Reject H <sub>0</sub>	4.5422e-09	Reject H <sub>0</sub>
PC-MIN30 vs. NRM	5.1105e-06	Reject H <sub>0</sub>	3.3991e-07	Reject H <sub>0</sub>	6.3678e-07	Reject H <sub>0</sub>
PC-MIN30 vs. NCR	0.1025	Accept H <sub>0</sub>	6.6343e-09	Reject H <sub>0</sub>	5.3936e-11	Reject H <sub>0</sub>
PC-MIN30 vs. OSS	0.0944	Reject H <sub>0</sub>	1.4221e-09	Reject H <sub>0</sub>	1.3223e-10	Reject H <sub>0</sub>
PC-MIN30 vs. RUS	0.0015	Reject H <sub>0</sub>	8.3600e-04	Reject H <sub>0</sub>	4.0531e-04	Reject H <sub>0</sub>
PC-MIN30 vs. TML	0.0846	Reject H <sub>0</sub>	3.0199e-09	Reject H <sub>0</sub>	2.8340e-11	Reject H <sub>0</sub>

Table 5. Results of five performance measures for PC-MIN30 compared to the other methods for high-dimensional datasets

Measures	Methods							
	PC-MIN30	FULL	IHT	NRM	NCR	OSS	RUS	TML
Sensitivity	75.17	39.49	59.8	62.73	40.94	44.38	69.12	41.51
Specificity	83.91	91.34	59.87	71.18	91.17	89.74	76.34	90.49
G-mean	78.09	54.29	36.08	41.88	56.25	50.8	71.68	52.5
MCC	0.432	0.387	0.267	0.296	0.39	0.389	0.369	0.386
ROC curve	0.794	0.629	0.593	0.671	0.634	0.645	0.727	0.621

#### 4. CONCLUSION

This experiment presented a method called ‘PC-MIN’ that reduced most of the data set by selecting the majority class with the smallest sum of Euclidean distances between other majority classes within the similarity group. The PC-MIN30 algorithm produced a balanced training data set to train classification models that generated more balanced and stable predictions of sensitivity and specificity than from using other algorithms. The values for sensitivity, MCC, G-mean, and ROC curve of the PC-MIN algorithm were higher than for other methods. However, the PC-MIN algorithm had lower specificity than many of the algorithms compared in the experiment because those algorithms aim to increase or decrease the amount of data generated that could classify almost all minority class data but could only partially correctly classify majority class data in order to achieve better results for sensitivity, G-mean, and ROC curve. In contrast, the PC-MIN algorithm selected data sets that generated highly sensitive classifying models for a minority class by selecting majority data at the center of the majority class. Consequently, the classification models derived from the PC-MIN algorithm predicted a smaller majority class and increased the likelihood of predicting minority classes. Therefore, the PC-MIN algorithm is suitable for tasks that require higher sensitivity than specificity, such as investment analysis involving high profit potential with low investment and pre-diagnosis screening of patients. The PC-MIN algorithm is limited in use where there are large outlier and noisy data sets because selecting majority data that is at the center of the data group may result in the PC-MIN algorithm selecting some of the outlier or noisy data. The PC-MIN algorithm can continue to evolve in parallel computing and distributing processing architectures for improved efficiency in processing and handling large data sets.

#### ACKNOWLEDGEMENTS

This research was also supported by the Faculty of Science at Sriracha, Kasetsart University under Grant Sci-Src01/2564. We would like to give thankful to Research and Development Institute, Kasetsart University for supporting proofreading services.

#### REFERENCES

- [1] P. Vuttipittayamongkol and E. Elyan, “Improved overlap-based undersampling for imbalanced dataset classification with application to epilepsy and parkinson’s disease,” *International Journal of Neural Systems*, vol. 30, no. 8, p. 2050043, Jul. 2020, doi: 10.1142/S0129065720500434.
- [2] C. H. Cheng, Y. F. Kao, and H. P. Lin, “A financial statement fraud model based on synthesized attribute selection and a dataset with missing values and imbalanced classes,” *Applied Soft Computing*, vol. 108, p. 107487, Sep. 2021, doi: 10.1016/j.asoc.2021.107487.
- [3] S. Bagui and K. Li, “Resampling imbalanced data for network intrusion detection datasets,” *Journal of Big Data*, vol. 8, no. 1, Jan. 2021, doi: 10.1186/s40537-020-00390-x.
- [4] J. Chi *et al.*, “Learning to undersampling for class imbalanced credit risk forecasting,” in *Proceedings - IEEE International Conference on Data Mining, ICDM*, Nov. 2020, vol. 2020-November, pp. 72–81, doi: 10.1109/ICDM50108.2020.00016.
- [5] T. Kim, B. Do Chung, and J. S. Lee, “Incorporating receiver operating characteristics into naive Bayes for unbalanced data classification,” *Computing*, vol. 99, no. 3, pp. 203–218, Feb. 2017, doi: 10.1007/s00607-016-0483-z.
- [6] P. Soltanzadeh and M. Hashemzadeh, “RCSMOTE: Range-controlled synthetic minority over-sampling technique for handling the class imbalance problem,” *Information Sciences*, vol. 542, pp. 92–111, Jan. 2021, doi: 10.1016/j.ins.2020.07.014.
- [7] L. Cao and H. Shen, “Imbalanced data classification using improved clustering algorithm and under-sampling method,” in *Proceedings - 2019 20th International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT 2019*, Dec. 2019, pp. 358–363, doi: 10.1109/PDCAT46702.2019.00071.
- [8] Z. Chen, J. Duan, L. Kang, and G. Qiu, “A hybrid data-level ensemble to enable learning from highly imbalanced dataset,” *Information Sciences*, vol. 554, pp. 157–176, Apr. 2021, doi: 10.1016/j.ins.2020.12.023.
- [9] L. Wang, M. Han, X. Li, N. Zhang, and H. Cheng, “Review of classification methods on unbalanced data sets,” *IEEE Access*, vol. 9, pp. 64606–64628, 2021, doi: 10.1109/ACCESS.2021.3074243.
- [10] H. Han, B. Mao, H. Lv, Q. Zhuo, and W. Wang, “One-sided fuzzy SVM based on sphere for imbalanced data sets learning,” in *Proceedings - Fourth International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2007*, 2007, vol. 2, pp. 166–170, doi: 10.1109/FSKD.2007.430.
- [11] B. Krawczyk, M. Woźniak, and F. Herrera, “Weighted one-class classification for different types of minority class examples in imbalanced data,” in *IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - CIDM 2014: 2014 IEEE Symposium on Computational Intelligence and Data Mining, Proceedings*, Dec. 2015, pp. 337–344, doi: 10.1109/CIDM.2014.7008687.
- [12] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, “RUSBoost: A hybrid approach to alleviating class imbalance,” *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 40, no. 1, pp. 185–197, Jan. 2010, doi: 10.1109/TSMCA.2009.2029559.
- [13] E. Ramentol *et al.*, “IFROWANN: imbalanced fuzzy-rough ordered weighted average nearest neighbor classification,” *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 5, pp. 1622–1637, Oct. 2015, doi: 10.1109/TFUZZ.2014.2371472.
- [14] C. M. Vong and J. Du, “Accurate and efficient sequential ensemble learning for highly imbalanced multi-class data,” *Neural Networks*, vol. 128, pp. 268–278, Aug. 2020, doi: 10.1016/j.neunet.2020.05.010.
- [15] P. Zybiewski, R. Sabourin, and M. Woźniak, “Preprocessed dynamic classifier ensemble selection for highly imbalanced drifted data streams,” *Information Fusion*, vol. 66, pp. 138–154, Feb. 2021, doi: 10.1016/j.inffus.2020.09.004.
- [16] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic minority over-sampling technique,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002, doi: 10.1613/jair.953.

- [17] H. Han, W. Y. Wang, and B. H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *Lecture Notes in Computer Science*, vol. 3644, no. PART I, Springer Berlin Heidelberg, 2005, pp. 878–887.
- [18] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5476 LNAI, Springer Berlin Heidelberg, 2009, pp. 475–482.
- [19] L. Ma and S. Fan, "CURE-SMOTE algorithm and hybrid algorithm for feature selection and parameter optimization based on random forests," *BMC Bioinformatics*, vol. 18, no. 1, Mar. 2017, doi: 10.1186/s12859-017-1578-z.
- [20] G. Douzas, F. Bacao, and F. Last, "Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE," *Information Sciences*, vol. 465, pp. 1–20, Oct. 2018, doi: 10.1016/j.ins.2018.06.056.
- [21] G. A. Pradipta, R. Wardoyo, A. Musdholifah, and I. N. H. Sanjaya, "Radius-SMOTE: A new oversampling technique of minority samples based on radius distance for learning from imbalanced data," *IEEE Access*, vol. 9, pp. 74763–74777, 2021, doi: 10.1109/ACCESS.2021.3080316.
- [22] G. Menardi and N. Torelli, "Training and assessing classification rules with imbalanced data," *Data Mining and Knowledge Discovery*, vol. 28, no. 1, pp. 92–122, Oct. 2014, doi: 10.1007/s10618-012-0295-5.
- [23] I. Tomek, "Two modifications of CNN," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-6, no. 11, pp. 769–772, Nov. 1976, doi: 10.1109/TSMC.1976.4309452.
- [24] M. Kubat and S. Matwin, "Addressing the curse of imbalanced data sets: One-sided sampling," *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 179–186, 1997, [Online]. Available: <http://sci2s.ugr.es/keel/pdf/algorithm/congreso/kubat97addressing.pdf>.
- [25] K. S. Smith *et al.*, "Physician attitudes and practices related to voluntary error and near-miss reporting," *Journal of Oncology Practice*, vol. 10, no. 5, pp. e350–e357, Sep. 2014, doi: 10.1200/JOP.2013.001353.
- [26] J. Laurikkala, "Improving identification of difficult small classes by balancing class distribution," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2101, Springer Berlin Heidelberg, 2001, pp. 63–66.
- [27] C. F. Tsai, W. C. Lin, Y. H. Hu, and G. T. Yao, "Under-sampling class imbalanced datasets by combining clustering analysis and instance selection," *Information Sciences*, vol. 477, pp. 47–54, Mar. 2019, doi: 10.1016/j.ins.2018.10.029.
- [28] J. Zhang and I. Mani, "kNN approach to unbalanced data distributions: A case study involving," *International Conference on Machine Learning*, vol. 1999, no. December, pp. 1–6, 2003.
- [29] W. C. Lin, C. F. Tsai, Y. H. Hu, and J. S. Jhang, "Clustering-based undersampling in class-imbalanced data," *Information Sciences*, vol. 409–410, pp. 17–26, Oct. 2017, doi: 10.1016/j.ins.2017.05.008.
- [30] A. Guzmán-Ponce, J. S. Sánchez, R. M. Valdovinos, and J. R. Marcial-Romero, "DBIG-US: A two-stage under-sampling algorithm to face the class imbalance problem," *Expert Systems with Applications*, vol. 168, p. 114301, Apr. 2021, doi: 10.1016/j.eswa.2020.114301.
- [31] M. Zheng *et al.*, "UFFDFR: Undersampling framework with denoising, fuzzy c-means clustering, and representative sample selection for imbalanced data classification," *Information Sciences*, vol. 576, pp. 658–680, Oct. 2021, doi: 10.1016/j.ins.2021.07.053.
- [32] K. Yang *et al.*, "Hybrid classifier ensemble for imbalanced data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 4, pp. 1387–1400, Apr. 2020, doi: 10.1109/TNNLS.2019.2920246.
- [33] K. Yang *et al.*, "Progressive hybrid classifier ensemble for imbalanced data," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 4, pp. 2464–2478, Apr. 2022, doi: 10.1109/TSMC.2021.3051138.
- [34] G. E. Batista, A. L. C. Bazzan, M.-C. Monard, G. E. A. P. A. Batista, and M. C. Monard, "Balancing training data for automated annotation of keywords: A case study. Missing data imputation view project automatic genetic generation of fuzzy classification systems. View project balancing training data for automated annotation of keywords: A cas," *WOB*, no. May 2014, 2003, [Online]. Available: <https://www.researchgate.net/publication/221322870>.
- [35] J. Stefanowski and S. Wilk, "Improving rule based classifiers induced by MODLEM by selective pre-processing of imbalanced data," *Proc. of the RSKD Workshop at ECML/PKDD*, pp. 54–65, 2007, [Online]. Available: [http://www.ecmlpkdd2007.org/CD/workshops/RSKD/workshop\\_print.pdf#page=60](http://www.ecmlpkdd2007.org/CD/workshops/RSKD/workshop_print.pdf#page=60).
- [36] B. Mirzaei, B. Nikpour, and H. Nezamabadi-pour, "CDBH: A clustering and density-based hybrid approach for imbalanced data classification," *Expert Systems with Applications*, vol. 164, p. 114035, Feb. 2021, doi: 10.1016/j.eswa.2020.114035.
- [37] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 42, no. 4, pp. 463–484, Jul. 2012, doi: 10.1109/TSMCC.2011.2161285.
- [38] M. Benito and D. Peña, "A fast approach for dimensionality reduction with image data," *Pattern Recognition*, vol. 38, no. 12, pp. 2400–2408, Dec. 2005, doi: 10.1016/j.patcog.2005.03.022.
- [39] J. Shlens, "A tutorial on principal component analysis," 2014, [Online]. Available: <http://arxiv.org/abs/1404.1100>.
- [40] M. E. Tipping and C. M. Bishop, "Mixtures of probabilistic principal component analyzers," *Neural Computation*, vol. 11, no. 2, pp. 443–482, Feb. 1999, doi: 10.1162/089976699300016728.
- [41] E. Arias-Castro, G. Lerman, and T. Zhang, "Spectral clustering based on local PCA," *Journal of Machine Learning Research*, vol. 18, pp. 1–57, 2017.
- [42] C. Ding and X. He, "K-means clustering via principal component analysis," in *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004*, 2004, pp. 225–232, doi: 10.1145/1015330.1015408.
- [43] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay, "Clustering large graphs via the Singular Value Decomposition," *Machine Learning*, vol. 56, no. 1–3, pp. 9–33, Jul. 2004, doi: 10.1023/B:MACH.0000033113.59016.96.
- [44] M. Du, S. Ding, and H. Jia, "Study on density peaks clustering based on k-nearest neighbors and principal component analysis," *Knowledge-Based Systems*, vol. 99, pp. 135–145, May 2016, doi: 10.1016/j.knsys.2016.02.001.
- [45] C. Kasemtaweekchok, P. Pharkdepinoy, and P. Doungmanee, "Large-scale instance selection using center of principal components," in *2021 Joint 6th International Conference on Digital Arts, Media and Technology with 4th ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunication Engineering, ECTI DAMT and NCON 2021*, Mar. 2021, pp. 157–160, doi: 10.1109/ECTIDAMTNCN51128.2021.9425702.
- [46] G. Lemaitre, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, pp. 1–5, 2017.
- [47] A. Frank and A. Asuncion, "UCI Machine learning repository," 2010, [Online]. Available: <http://archive.ics.uci.edu/ml>.
- [48] A. D. Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi, "Calibrating probability with undersampling for unbalanced classification," in *Proceedings - 2015 IEEE Symposium Series on Computational Intelligence, SSCI 2015*, Dec. 2015, pp. 159–166, doi: 10.1109/SSCI.2015.33.

- [49] Z. Ding, “Diversified ensemble classifiers for highly imbalanced data learning and their application in bioinformatics,” *Computer Science Dissertations*, p. 149, 2011, doi: 10.57709/1997714.
- [50] C. C. Chang and C. J. Lin, “LIBSVM: A Library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–27, Apr. 2011, doi: 10.1145/1961189.1961199.
- [51] M. West *et al.*, “Predicting the clinical status of human breast cancer by using gene expression profiles,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 98, no. 20, pp. 11462–11467, Sep. 2001, doi: 10.1073/pnas.201162998.
- [52] U. Alon *et al.*, “Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 96, no. 12, pp. 6745–6750, Jun. 1999, doi: 10.1073/pnas.96.12.6745.
- [53] I. Guyon, S. Gunn, A. Ben Hur, and G. Dror, “Result analysis of the NIPS 2003 feature selection challenge,” *Advances in Neural Information Processing Systems*, pp. 13–18, 2005.
- [54] T. R. Golub *et al.*, “Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring,” *Science*, vol. 286, no. 5439, pp. 531–527, Oct. 1999, doi: 10.1126/science.286.5439.531.
- [55] Z. Q. Zeng, H. Bin Yu, H. R. Xu, Y. Q. Xie, and J. Gao, “Fast training support vector machines using parallel sequential minimal optimization,” in *Proceedings of 2008 3rd International Conference on Intelligent System and Knowledge Engineering, ISKE 2008*, Nov. 2008, pp. 997–1001, doi: 10.1109/ISKE.2008.4731075.
- [56] P. Vuttipittayamongkol, E. Elyan, and A. Petrovski, “On the class overlap problem in imbalanced data classification,” *Knowledge-Based Systems*, vol. 212, p. 106631, Jan. 2021, doi: 10.1016/j.knosys.2020.106631.

## BIOGRAPHIES OF AUTHORS



**Chatchai Kasemtaweechok**    received the Ph.D. degree in computer science from Kasetsart University. He was programmer and senior programmer with IBM software partner for ten years. He is lecturer with the Computer Science and Information Department, Faculty of Science at Sriracha, Kasetsart University Sriracha Campus. His research interests include data pre-processing, data reduction, and parallel computing techniques. He can be contacted at email: chatchai.ka@ku.th.



**Worasait Suwannik**    received the Ph.D. degree in computer engineering from Chulalongkorn University, Thailand. He is an Associate Professor at the Faculty of Science, Kasetsart University. He is an industry-certified security professional (CISSP, CISA). His research interests include artificial intelligence and parallel programming. He can be contacted at email: worasait.suwannik@gmail.com.