science, engineering and health studies

Research Article

Science, Engineering and Health Studies
https://lio1.tci-thaijo.org/index.php/sehs
ISSN (Online): 2630-0087

Service priority classification using machine learning

Teratam Boonprapapan, Pusadee Seresangtakul, and Punyaphol Horata*

Department of Computer Science, College of Computing, Khon Kaen University, Khon Kaen 40002, Thailand

ABSTRACT

This article details a procedure for classifying service cases with various priority levels based on machine learning (ML). It accurately defines the priority level of each service case. The presence of imbalanced datasets in service cases poses a challenge for achieving reliable classification accuracy. To address this, the use of the synthetic minority over-sampling technique (SMOTE) was proposed as the method for balancing the datasets prior to applying the ML method. From these experimental results, an improvement in the precision of the learning process was observed, which led to better outcomes in the test sets. This improvement was measured using the efficiency metrics from the confusion matrix. The experiment involved 6,182 service cases, categorized into four levels: critical, serious, moderate, and low. These were based on test comparisons with other ML methods. The accuracy achieved in the test data was 94.37%. By employing a hybrid technique to address the imbalance in SMOTE and the support vector machine model, it was found to be more effective than the comparative term frequency-inverse document frequency model that was used in conjunction with cosine similarity, which achieved an evaluation score of 70.14%.

Keywords: machine learning; classification; imbalanced data; SMOTE

***Corresponding author**:
Pusadee Seresangtakul
pusadee@kku.ac.th

***Received**: 17 February 2023*
***Revised**: 8 January 2024*
***Accepted**: 27 January 2024*
***Published**: 3 October 2024*

***Citation**:
Boonprapapan, T., Seresangtakul, P., and Horata, P. (2024). Service priority classification using machine learning. Science, Engineering and Health Studies, 18, 24020002.

1. INTRODUCTION

The classification analysis for service delivery is grounded in service level agreements (SLAs). Numerous businesses and organizations utilize SLA-based techniques to improve service delivery and the efficiency of their employees by categorizing services, which are typically described in textual form. The service categories within the SLA dataset are divided into four levels: critical, serious, moderate, and low. The accurate classification of the service types within the SLA dataset is crucial in order to ensure the appropriate delivery of services by organizations. Boonprapapan et al. (2022) reported that using the term frequency-inverse document frequency (TF-IDF) technique and combining it with cosine similarities can classify service categories (Boonprapapan et al., 2022). However, the accuracy of the study was not found to be adequate for application in real situations due to the SLA dataset being imbalanced.

To address the imbalance data classification problem, numerous studies have widely examined the issue. Khamphakdee and Seresangtakul (2021) studied sentiment analysis for the classification of opinions at the levels of sentences and documents. The article discussed the use of machine learning (ML) algorithms with data extraction by dichotomizing the classification of reviews. The research compared many algorithms. The most suitable ML for classifying opinion polarities using datasets was collected from online media. It was based on the 11 steps of preprocessing and used the Delta TF-IDF, TF-IDF, N-Gram, and Word2Vec techniques for features extraction, as well as many ML for the classification of opinions.

Chemchem et al. (2019) presented an estimated wheat yield based on meteorological data, in which the synthetic minority over-sampling technique (SMOTE) method was employed to solve the problem of the unbalanced datasets. The ML process and SMOTE in combination with the

random forest (RF) algorithm yielded the best results. The supervised learning method was also compared to the nine-machine learning algorithm and served as an example of using the imbalance classification framework.

Sreejith et al. (2020) presented an imbalance classification framework, which involved a dataset that could be used to develop a clinical decision support system (CDSS) that can address similar imbalances. Three datasets were tested, which were measured using a RF classifier. The most suitable accuracy (89.04%) was achieved by employing the PID dataset.

Intayoad et al. (2018) presented a classification and grouping of passing or failing students. The educational (test) data used was both diverse and unbalanced. This research optimized the accuracy of the classified data using web data, by conducting comparative experiments on SMOTE, Borderline-SMOTE1, Borderline-SMOTE2, and SVM-SMOTE. All of these are balancing methods for the dataset, which were performed in conjunction with the classification. This study determined that synthetic minority over-sampling methods had improved the detection of minority classes and had also improved classification performances in precision, recall, and F1-score. This was a comparison experiment of balancing methods.

Davagdorj et al. (2020) examined a comparative technique of ML techniques to solve dataset imbalances using SMOTE techniques and an adaptive synthetic (ADASYN). The prediction model and subsequent F1-score results of the analyses demonstrated that the SMOTE and ADASYN balancing techniques, which were used in combination with the gradient of the classifier boosting trees (GBT), RF, and multilayer perceptron neural network (MLP), had performed the best.

The imbalance handling techniques motivated us to apply the techniques with supervisor classification methods in ML in order to remedy the effects of imbalances in the service case priority dataset. Therefore, this paper proposes a hybrid model to address the data imbalance problem of the service case priority data set through the combination of SMOTE, ADASYN, Borderline SMOTE, and Near-Miss for algorithm balancing with various ML techniques in order to study their classifying effectiveness. Therefore, the relevant balancing techniques are first discussed in this article.

Chawla et al. (2002) introduced SMOTE, a method employed in data science and ML to tackle the challenge of class imbalance in classification problems. Class imbalances arise when the instances of one class significantly outnumber those of another, potentially leading to biased or inaccurate predictions by models. SMOTE effectively addresses this issue by generating synthetic samples for the minority class. The processes of SMOTE can be outlined in the following manner:

- Identifying the minority class: SMOTE first identifies the minority class that needs to be over-sampled to balance the class distribution.
- Synthetic sample generation: For each instance in the minority class, SMOTE considers its nearest neighbors (usually the *k*-nearest neighbors). Then it randomly selects one of those neighbors and computes a vector between the instance and its neighbor. A new, synthetic sample is created by adding a fraction of this vector to the original instance. This fraction is multiplied by a random number between 0 and 1, which allows the synthetic sample to be a point along the line segment between the original instance and its selected neighbor.
- Increasing variety: By varying the fraction for each synthetic sample, SMOTE creates diverse instances rather than exact copies, which adds more generalization to the model.
- Balancing the dataset: The process is repeated until the class distribution has been sufficiently balanced, allowing the models trained on the dataset to better generalize and to not be biased towards the majority class.

The advantage of SMOTE lies in its ability to create new, proper instances of the minority class instead of merely duplicating the existing ones, which can cause overfitting. This approach leads to a more varied and representative dataset.

ADASYN is a sampling algorithm proposed by He et al. (2008). It works similarly to the SMOTE algorithm by selecting the closest neighbor *k* adaptively (Brandt and Lanzén, 2021). The weighted distribution of the minority samples can be divided by their level of learning difficulty, in which the synthetic data generates minority samples that are more difficult to learn. This reduces the learning bias caused by an unbalanced distribution of data, which can also shift the decision boundaries to reach a sample that is difficult to learn (He et al., 2008).

Borderline-SMOTE represents an improvement from the original SMOTE method. However, unlike SMOTE, Borderline-SMOTE creates synthetic instances near its boundaries, where instances on or near the boundary line are more likely to be misclassified than those instances that are further away from the boundaries in classification. As a result, the classifier's efficiency is enhanced (Wang et al., 2015). Its performance is similar to the density-based spatial clustering of applications with noise (DBSCAN).

Near-Miss, which is a technique for balancing a low-sampling unbalanced dataset with a balanced class distribution, randomly eliminates most classes when two points belonging to different classes are very close. This algorithm eliminates the data points of larger classes. Of the sampling techniques, the Near-Miss method is the most widely used (IIT Kanpur, Prutor Online Academy).

2. MATERIALS AND METHODS

The model sequence for service case priorities, utilizing comparative ML algorithms in collaboration with SMOTE before applying an ML method. The first part was presented the imported dataset and the term frequency-inverse document frequency (TF-IDF) used for feature extraction in preprocessing with weighting, which represents a statistical calculation that determines how important a word is to a document. The second part calculated similarity scoring by evaluating the sentence from word embedding and service case testing, which consists of five processes: data collection, preprocessing, feature extraction, ML algorithm, and evaluating the ML algorithms & analysis (Figure 1).

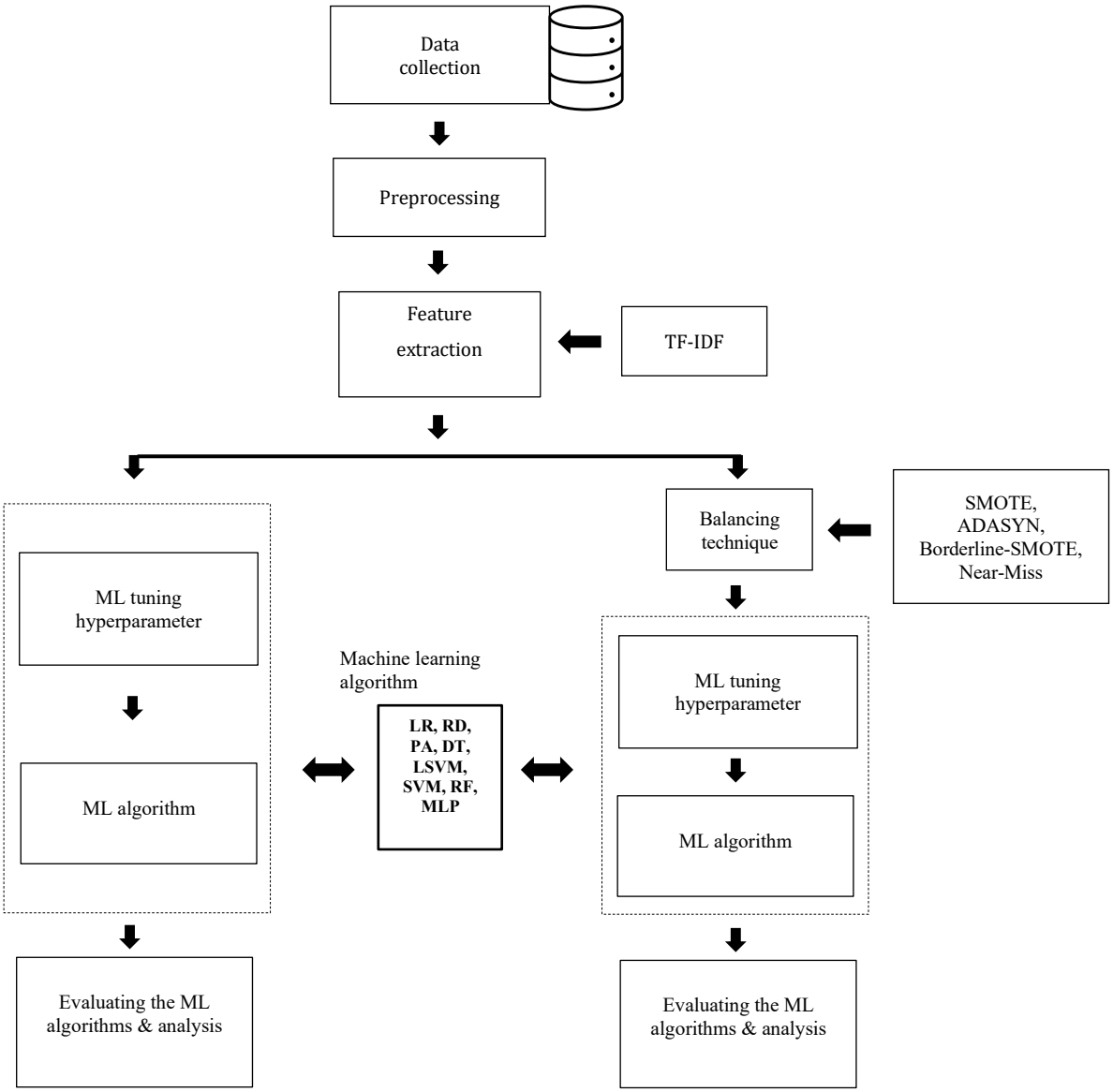

Figure 1. The step-by-step overview of similarity measurements using the machine learning technique

As seen in Figure 1, this research aimed to improve the efficiency of the classification accuracy of the service case priorities and to predict the learning performance in ML. The study was developed within the organization's service management system, which aimed at categorizing the criticality of the service cases in order to meet the requirements for classification in the aforementioned study by Boonprapapan et al. (2022). The authors built on the problem of unbalanced datasets, which resulted in low accuracy, leading to inaccurate predictions in order of importance. Our solution to this problem was to manage and balance the input dataset across all classes, which consisted of four techniques: SMOTE, ADASYN, Borderline-SMOTE, and Near-Miss. Further methods were offered for optimizing ML models with hyperparameter optimization, which resulted in a model with high accuracy and/or with a reduction of any loss to the lowest value. Lastly, the performance of that type of model was compared to determine the most suitable model for classification.

2.1 Data collection

Datasets were imported from the corporate web service database in Microsoft Dynamics 365® (2022) to create an experimental dataset based on real-world data from within the organization. To easily refer to this, the collected dataset was named as the service case and priority (SCP) dataset. This system is capable of providing access to service data for employees, such as engineers. The input data is comprised of two main elements: the service details and the service case priorities, with four identified priority classes and their corresponding scores: (1) critical, (2) serious, (3) moderate, and (4) low. Table 1 shows the number and description of the samples for each class in the SCP dataset.

Table 1. The service case

Number	Priority	Data numbers	Description
1	Critical	243	There are equipment or system failures, unresponsiveness, or equipment failures without provisioning a redundant system, which will severely affect the serviceability. The problem must be fixed and be back in use within four hours.
2	Serious	435	Access to the system is unavailable or there is incomplete operation of the system, including the inefficiency of the device. Troubleshooting must be completed within twenty-four hours.
3	Moderate	4,208	There is the occurrence of usability problems with systems or devices that are not important or have little impact on users and systems, which are not difficult to fix. The corrective actions must be resolved within three days.
4	Low	1,296	There is incomplete access to the application or there is little impact on the user because basic advice can be given, and the basic problems can be fixed. Troubleshooting is usually resolved within five days.

2.2 The preprocessing stages

In the preprocessing stage, also known as data preparation, feature extraction was employed to transform the service cases into vectors. This transformation occurred within a sub-process of the data collection phase. Initially, the data was imported from the corporate web service as raw data and was then formatted to meet the requirements of the TF-IDF feature extraction method. Preprocessing involved several steps: word segmentation, stop word removal, text normalization, and the application of the name-entity recognition techniques. Data preparation entailed manipulating the data to align with the chosen model for feature extraction. After this process, the dataset was prepared for model training in the vector format, as applied in the researcher's experimental model, which is illustrated in Figure 1 in the feature extraction selection portion. This process was detailed by Boonprapapan et al. (2022). The Python® Package (Python Programming Language, 2022) and the PyThaiNLP library (PyThaiNLP, 2022) were used for these tasks. Tables 2 and 3 present the examples of the preprocessing attributes and the service cases, respectively. Table 2 provides the explanatory information about the research-relevant variables, such as the service cases and priority cases, while Table 3 describes the examples of the service cases and their associated priority levels used in the research

Table 2. Examples of the data preprocessing attributes

Number	Attributes	Descriptions
1	Service cases	Service cases arise from service tasks between the organization and the customer: "The building network device is unavailable."
2	Priority cases	Service case priorities are defined by the service level agreements: critical, serious, moderate, and low.

Table 3. Examples of data preprocessing in the service cases

Number	Service cases	Priority
1	ปุ่มกดถ่ายทอดสดสูญหายจากโปรแกรม Teams ของผู้ใช้งาน (The live event button disappears from Teams client.)	1 (Critical)
2	รายงานผู้ใช้งานจากภายนอกไม่สามารถส่งออกอีเมลไปยังผู้ใช้งาน (The user report from external cannot send mail to the user.)	2 (Serious)
3	เขียนสคริปต์ย้ายนักศึกษาที่จบการศึกษาไปเป็น Alumni และลบ Account บุคลากรที่ลาออก (Write a script to transfer the graduating students to alumni, and then delete the resigned personnel account.)	3 (Moderate)
4	ย้าย VM จากโปรแกรม Hyper-V ไปยังเครื่อง AHV Nutanix (Move VM from Hyper-V to AHV Nutanix.)	4 (Low)

2.3 Feature extraction

At this point, ML methods were utilized to extract the existing features by transforming the data into a format that could be utilized, thereby reducing the data size of the model. In other words, a service case dataset is a vector-based dataset requiring technical processing methods. This is used to understand the context of the term frequency and inverse document frequency text to find the relevant content. The keywords in the sentences by the service case can be converted to numbers to better understand the relationship between them (Khamphakdee and Seresangtakul, 2021). Feature extraction through the employment of the TF-IDF technique was applied due to the superior performance results achieved when it is used with the SVM technique.

Feature extraction conducted herein was comprised of two equations. Equation 1 was used to calculate the word frequency of the service case by measuring

the word frequency *w* in document *d*. Word frequency serves as a measure of the word occurrences in a document in contrast to the total number of words found in a document.

In the next step, Equation 2, the IDF variable was used to define the importance of a word. The inverse document frequency of the word w is defined as the total number of documents (*N*) in the text corpus *D* divided by the number of documents containing w, as follows:

From the above equations, the *TF* and *IDF* can be multiplied to get the TF-IDF weight for that term. A high TF-IDF describes a term with high frequency in the computed documents, yet a lower frequency in other documents. Furthermore, calculating the TF-IDF weight helps to filter out common terms, enabling only the essential words in each document.

2.4 Experimental setup

To generate the training data, a random sequencing function from SQL Server® (Transact-SQL) (Microsoft, 2019) was employed, which produced either a random number or a number within a specified range. An example of this data collection is presented in Table 3. The raw materials used for conducting the research consisted of Visual Studio Code, Jupyter Notebook, Anaconda, Microsoft SQL Server, and a laptop with a 12th Gen Intel(R) i5-1235U processor (1.30 GHz), 16.0 GB RAM, and a 64-bit Windows 10 Enterprise operating system. The experiments were set up based on the K-Fold cross-validation approach to split a dataset into parts in order to verify the model's validity, in which k=10. This process allowed us to avoid overfitting in a predictive model, especially by limiting the data.

$$Term\ Frequency\ = \frac{Number\ of\ instances\ of\ word\ w\ in\ document\ d}{Total\ number\ of\ words\ in\ document\ d} \quad (1)$$

$$IDF\ = log\frac{total\ number\ of\ document\ (N) in\ text\ corpus\ D}{number\ of\ documents\ containing\ w} \quad (2)$$

3. EXPERIMENTS AND EVALUATION

3.1 Evaluation metrics

This research conducted performance evaluations of the most frequently used ML approaches; such as the support vector machine, decision tree classifier, RF classifier, multi-layer perceptron classifier, logistic regression classifier, passive-aggressive classifier, ridge regression classifier, and the linear support vector classifier. This research utilized three evaluation metrics (Hripcsak and Rothschild, 2005): F1-score, precision, and recall to evaluate the classification performance of the comparative models for the SCP dataset. The F1-*score*, *Precession*, and *Recall* were computed as follows:

$$\mathrm{F1} - score\ = 2 \times \frac{precision \times recall}{precision + recall} \quad (3)$$

$$Precision\ = \frac{TP}{TP+FP} \quad (4)$$

$$Recall\ = \frac{TP}{TP+FN} \quad (5)$$

where: *TP* is the number of true positives, *TN* is the number of true negatives, FP is the number of false positives, and *FN* is the number of false negatives (Chemchem and Drias, 2015).

The results indicated that the accuracy caused by the recall value had been low (<0.5). In contrast, most of the precision values had been high (Table 4). The recall values affirmed that if the input data had been unbalanced, the classification of most datasets was found to be problematic, given a value of less than 0.5, which indicated a higher *FN* value. Conversely, both the *TP* and *FP* values had been low. When the precision was high, this effect was inefficient, which designated a low *FP*. Our experiments, therefore, investigated techniques in which problems can be solved in those instances, in which the precision was high, and the recall was low, as seen in the Venn diagram of recall and the high precision values (Klintberg, 2017). The recommended preprocessing steps were performed on the unbalanced dataset to evaluate the performance of the dataset with the various ML models: SVM; DT; RF; MLP; LR; PA; RR; and LSVM methods (Figure 1). An experiment employing the TF-IDF method in combination with cosine similarity showed that the efficacy evaluation of the model had been low. The experiment was, therefore, redesigned using the original dataset (Figure 1), consisting of the service and priority cases (Table 4), which encompassed the original data set, as well as an adaptation of the original rebalanced data set.

F1-scores, as well as precision and recall, were used to evaluate the performance of each model. As shown in Table 4, the recall was found to have had a low efficiency in the first part of the performance evaluation. It is a metric that quantifies the correct number of positive predictions out of all the possible positive predictions, as opposed to that of the precision value. In this way, recall provides some concepts that are related to positive class coverage and is also used to measure the coverage performance of the minority class within the unbalanced datasets. Therefore, it may be concluded that the problems created from the experiment results of the first part had been caused by the unbalanced dataset affecting the evaluation of the performance of various models measured as the recall value. Within the second part of the process, the researcher added the following balancing techniques to the experiment: SMOTE, ADASYN, Borderline-SMOTE, and Near-Miss.

Recall measures positive predictions by building on all the positive samples of the dataset (Mohajon, 2020). The values of low recall and high precision values that were displayed showed that the error of the positive sample had been high, or the FN had been high. Yet, the predicted examples had been positive, truly positive (low FP). In this paper, the focus was on enhancing the classifier in order to achieve a high recall value. A high recall value implies that there are fewer false negative cases, which, in turn, enhances the service quality of those corporations that utilize this application.

3.2 Hyperparameter tuning

The objective of hyperparameter tuning is to find the parameters with the highest model performance and the lowest error rates. Hyperparameter optimization is a

model characteristic that is defined by constants, which describe the process of setting variables before learning or testing the data in order to obtain the best value for the model. Methods were used such as GridSearchCV, which is a customized hyperparameter tuning approach that strives to obtain the most suitable variables for the desired model. As a built-in function, there is the scikit-learn® package (scikit-learn, GridSearchCV). The trial hyperparameter tuning, which utilized different methods during model development, helped to improve model performance. The most important hyperparameters, which were discovered for the ML algorithms, are depicted in Table 4.

Table 4. The tuning of the hyperparameters

Models	Parameters
Support vector machine (SVM)	kernel = rbf, c = 50, gamma = scale
Random forest (RF)	n_estimators = 1,000, max_features = log2, min_samples_split = 2, min_samples_leaf = 1
Multi-layer perceptron (MLP)	solver = lbfgs, learning_rate = constant, activation = tanh
Logistic regression (LR)	c = 100, max_iter = 100, tol = 1e-4
Passive aggressive (PA)	c = 0.1, tol = 1e-3, max_iter=1,000
Linear support vector machine (LSVM)	c=10, penalty='l2'
Ridge regression (RR)	alpha=0.5, tol=1e-3
Decision tree (DT)	criterion = entropy, max_features = log2, max_depth = 100

3.3 Evaluating the classifier performance on an unbalanced SCP dataset: A comparative analysis of ML algorithms

To assess the impact of training various classifiers with an SCP dataset characterized by unbalanced properties, Table 5 presents the precision, recall, and F1-scores for eight ML algorithms: SVM, RF, MLP, LR, PA, LSVM, RR, and DT. The findings indicated that using an unbalanced dataset without hyperparameter tuning had resulted in the highest accuracy for RF, followed by SVM, setting them up with 10-fold cross-validation. However, these results revealed that the performance metrics of all eight methods had remained low, rendering them unsuitable for practical applications.

Additionally, the aim was to explore the classification performance of each class in SVM, when specifically trained using input data without preprocessing by imbalance techniques. Table 6 displays the low recall values for Class 1, Class 2, and Class 4. In contrast, Class 3 demonstrated a high recall value. This discrepancy was attributed to the initial dataset used for the experiment, which contained a large number of instances in Class 3, leading to higher recall accuracy for that class.

Table 5. Predicting the comparison of the machine learning algorithms for testing the unbalanced samples

Models	Term frequency – inverse document frequency			
	Precision	Recall	F1-scores	Accuracies
SVM	0.756	0.333	0.353	0.710
RF	0.648	0.361	0.391	**0.715**
MLP	0.519	0.319	0.329	0.697
LR	0.675	0.331	0.348	0.702
PA	0.494	0.378	0.404	0.661
LSVM	0.542	0.315	0.321	0.697
RR	0.464	**0.386**	**0.409**	0.667
DT	0.541	0.273	0.251	0.682

Table 6. A comparison of the average confusion matrix values, categorized by service case classes, using the SVM algorithm without imbalance handling on the unbalanced test samples

Classes	Precision	Recall	F1-scores	Support
1	**0.95**	0.17	0.29	243
2	0.67	0.01	0.03	435
3	0.71	**0.98**	**0.82**	**4,208**
4	0.69	0.17	0.27	1,296

Table 6 displays the confusion matrix results for the SVM algorithm. These results show that the prediction accuracy had been the highest for Class 3, with the accuracy having been significantly influenced by the distribution of the data across classes. For instance, Class 1 (as seen in Table 6) had shown an exceptionally low recall value of 0.17. This implied that, in those instances, in which the actual class was Class 1, the model correctly predicted only 17% of these cases. Specifically, out of the 243 instances, in which the true class was Class 1, the model had accurately predicted only 42 cases and had incorrectly predicted 201 cases. Such findings indicated a markedly low prediction accuracy for instances, in Class 1.

Additionally, the unbalanced nature of the ML test dataset is referenced in Table 7. With the results, ML calculated the highest dataset and probability of prediction in Class 3. The first recall value, which was 0.17, had been the result of the ML algorithm's ability to correctly predict the critical class out of 42 from a total of 243 classes. However, 201 were predicted incorrectly. The second recall value for Class 2 was 0.01, which had been derived from the correct ML prediction of six serious cases out of 435 classes, but 429 were incorrectly predicted. The third recall value, 0.98, had resulted from the correct prediction of 4,125 out of 4,206 moderate classes, with 81 being incorrectly predicted. The last recall value for Class 4 had been 0.17, which resulted from the ML's correct prediction of 215 low classes out of 1,296 classes, in which 1,081 were incorrectly predicted. The results described above showed that the model had predicted the answers for Class 3 (moderate), which indicated that the dataset used for training the model had been unbalanced for all classes.

Table 7. Predictions of the service case numbers divided by class using SVM without balancing techniques: demonstrating the impact on class-specific recall

		Service case labels (actual)				
		Critical (1)	**Serious (2)**	**Moderate (3)**	**Low (4)**	**Totals**
Machine output (predicted)	1	42	0	194	7	243
	2	0	6	419	10	435
	3	2	3	**4,125**	78	**4,208**
	4	0	0	1,081	215	1,296
Total						6,182

Table 7 displays the confusion matrix results for the SVM algorithm. These results showed that that the prediction accuracy had been the highest for Class 3, with the accuracy having been significantly influenced by the distribution of the data across classes. For instance, Class 1 had an exceptionally low recall value of 0.17. This implied that in actual Class 1 instances, the model had correctly predicted only 17% of these cases. Specifically, out of 243 instances in which the true class was Class 1, the model had accurately predicted only 17 cases and incorrectly predicted 226 cases. Such findings indicated a markedly low prediction accuracy for those instances, in which the actual class had been Class 1.

The results, after addressing the imbalance issue of the dataset using the SMOTE technique in combination with the SVM algorithm, are shown in Table 8. They reveal that the precision, recall, and F1-scores had significantly improved compared to in Table 6. It was evident that using imbalance handling techniques had had a positive impact on the accuracy of the predictions in each class.

Table 8. Average of confusion metrics of the SVM algorithm with imbalanced handling for testing samples

Classes	**Precision**	**Recall**	**F1-scores**	**Support**
1	**0.97**	**0.99**	**0.98**	4,208
2	**0.97**	0.97	0.97	4,208
3	0.93	0.89	0.91	4,208
4	0.90	0.93	0.91	4,208

From the prediction results in each class after addressing the imbalance issue as shown in Table 9, the accuracy measurements of the SVM algorithm were presented in the form of a confusion matrix, which was influenced by the number of instances in each class. Specifically, according to Table 7, Class 1 (critical) had had a total of 243 service cases, with predictions of 42 for Class 1 (critical), 0 for Class 2 (serious), 194 for Class 3 (moderate), and 7 for Class 4 (low). This indicated that the predictions had shown an accuracy of 42 out of 243. Furthermore, based on the results from Table 9, Class 1 (critical) had had a total of 4,290 service cases, with predictions of 4,160 for Class 1 (critical), 47 for Class 2 (serious), 60 for Class 3 (moderate), and 23 for Class 4 (low). This indicated that the predictions had shown an accuracy of 4,160 out of 4,290. Therefore, by addressing the imbalance issue of the dataset, the accuracy of the predictions for each class had been improved.

Based on the number of instances of each class from averaging across all 10 folds, a comparison can be seen of the average without balancing (Figure 2) and with balancing (Figure 3). It was found that the prediction results for each class had increased when adjusted for balance.

Table 9. The accuracy measurements of the SVM algorithm, as indicated by the confusion matrix, were influenced by the number of instances in each class

		Service case labels (actual)				
		Critical (1)	**Serious (2)**	**Moderate (3)**	**Low (4)**	**Totals**
Machine output (predicted)	1	**4,160**	20	4	24	4,208
	2	47	**4,067**	11	83	4,208
	3	60	74	**3,758**	316	4,208
	4	23	36	249	**3,900**	4,208
Total						16,832

		Confusion matrix				
Predicted	class 1	**42 (0.68%)**	0 (0.0%)	194 (3.14%	7 (0.11%)	243 (17.28%)
	class 2	0 (0.0%)	**6 (0.10%)**	419 (6.78%	10 (0.16%)	435 (1.38%)
	class 3	2 (0.03%)	3 (0.05%)	**4125 (66.73%**	78 (1.26%)	4208 (98.03%)
	class 4	0 (0.0%)	0 (0.0%)	1081 (17.48%)	**215 (3.48%)**	1296 (16.59%)
		44 (95.45%)	9 (66.67%)	5819 (70.89%)	310 (69.35%)	**6182 (70.98%)**
		class 1	class 2	class 3	class 4	
		Actual				

Figure 2. The average for each class of the SVM algorithm without balancing

		Confusion matrix				
Predicted	class 1	**4160 (24.71%)**	20 (0.12%)	4 (0.22%)	24 (0.14%)	4208 (98.86%)
	class 2	47 (0.28%)	**4067 (24.16%)**	11 (0.07%)	83 (0.49%)	4208 (96.65%)
	class 3	60 (0.35%)	74 (0.44%)	**3758 (22.33%)**	316 (1.88%)	4208 (89.31%)
	class 4	23 (0.14%)	36 (0.21%)	249 (1.48%)	**3900 (23.17%)**	4208 (92.68%)
		4290 (96.97%)	4197 (96.90%)	4022 (93.44%)	4323 (90.22%)	**16832 (94.37%)**
		class 1	class 2	class 3	class 4	
		Actual				

Figure 3. The average for each class of the SVM algorithm with balancing

3.4 The effectiveness of the oversampling techniques on the imbalanced data: A comparative study using ML models

To evaluate the effectiveness of classification using various techniques for handling the imbalanced data, the findings indicated that using a balanced dataset without hyperparameter tuning had resulted in the highest accuracy for SVM, followed by balancing techniques. In the preprocessing step, the dataset was oversampled by creating synthetic samples with methods, such as SMOTE, Borderline SMOTE, ADASYN, and Near-Miss. Subsequently, the models were trained using various ML methods by setting them up with 10-fold cross-validation. The experimental results of this process are presented in Table 10.

Table 10. The accuracy measurements of various ML models that were preprocessed using the balancing techniques

Models	SMOTE			Borderline SMOTE			ADASYN			Near-Miss		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
SVM	**0.918**	**0.918**	**0.918**	0.905	**0.905**	**0.908**	**0.913**	**0.913**	**0.913**	**0.633**	**0.630**	**0.630**
RF	0.900	0.893	0.893	**0.913**	0.903	0.905	0.898	0.893	0.893	0.570	0.568	0.558
MLP	0.900	0.895	0.893	0.888	0.880	0.878	0.900	0.898	0.895	0.460	0.463	0.463
LR	0.868	0.865	0.865	0.858	0.858	0.855	0.855	0.853	0.850	0.495	0.495	0.493
PA	0.850	0.850	0.850	0.850	0.850	0.845	0.865	0.868	0.865	0.455	0.458	0.455
LSVM	0.873	0.868	0.863	0.873	0.870	0.868	0.868	0.865	0.860	0.500	0.503	0.498
RR	0.868	0.860	0.855	0.855	0.853	0.848	0.855	0.850	0.845	0.485	0.490	0.483
DT	0.833	0.828	0.828	0.850	0.845	0.845	0.828	0.825	0.823	0.503	0.520	0.520

The comparison of different balancing techniques showed that in our experiment, integrating SMOTE with the SVM model had produced the most favorable outcomes. This combination achieved a high precision of 0.918 and improved accuracy. To attain a data balance, the SMOTE algorithm was employed as a preprocessing step, in conjunction with the TF-IDF processing method.

After applying hyperparameter tuning through GridSearchCV, the performance of various ML algorithms were trained on the SCP dataset, which had undergone preprocessing with imbalance handling techniques, such as SMOTE, ADASYN, Borderline-SMOTE, and Near-Miss. The findings, which are illustrated in Table 11, highlighted the fact that the SVM combined with SMOTE had achieved the most effective results, attaining the highest score of 0.994 across all metrics, particularly in the F1-score. This high F1-score suggested the remarkable ability of the model to accurately classify, while also maintaining minimal false error rates. Such an approach was proven to be highly effective in addressing the imbalance issue prevalent in the service priority dataset, thereby ensuring the accuracy and relevance of its importance classification.

Table 11. The accuracy measurements for the machine learning algorithm after applying hyperparameter tuning through GridSearchCV

Models	Term frequency – inverse document frequency			
	Synthetic minority oversampling technique			
	Precision	Recall	F1-scores	Accuracies
SVM	**0.944**	**0.944**	**0.944**	**0.944**
RF	0.926	0.923	0.924	0.923
MLP	0.912	0.911	0.909	0.913
LR	0.891	0.890	0.887	0.890
PA	0.878	0.875	0.871	0.879
LSVM	0.879	0.877	0.872	0.877
RR	0.863	0.859	0.854	0.859
DT	0.826	0.828	0.827	0.826

4. CONCLUSION

Our experimental model followed six essential steps: preprocessing, feature extraction, the SMOTE application, hyperparameter tuning, ML algorithm implementation (including SVM, DT, RF, MLP, LR, PA, RR, and LSVM), and algorithm evaluation. To understand service case frequency, service case datasets were used via analyzing word weight frequencies. This helped to assess the impact of the unbalanced datasets, which can often lead to lower performance outcomes.

To combat dataset imbalance, SMOTE was applied, facilitating a comparison between the balanced and unbalanced data via hyperparameter tuning. Eight ML algorithms were evaluated with a focus on identifying the most effective classifier, using the confusion matrix for performance assessment. This research primarily targeted service case classification, examining various learning methods across these algorithms. The SVM had achieved the highest classification accuracy of 0.944, addressing the key challenge of the imbalanced data distribution due to employee service assessments.

The SVM-SMOTE combination emerged as the most effective method. Looking ahead, there are plans to broaden our experiments to include diverse datasets and to explore advanced feature extraction techniques like Delta-TFIDF. The goals are to refine our model for prioritizing a wider range of datasets and to enhance its capacity for proposing solutions to complex service cases. This research aimed at boosting operational efficiency and effectiveness in the organizational and employee contexts.

ACKNOWLEDGMENT

The Department of Computer Science, College of Computing, Khon Kaen University, Thailand, supported all research-related operations.

REFERENCES

Boonprapapan, T., Horata, P., and Seresangtakul, P. (2022). Incident task sequence for service priority using cosine similarity. In *Proceedings of the 1st International Conference on Technology Innovation and Its Applications (ICTIIA)*, pp. 87–92. Tangerang, Indonesia.

Brandt, J., and Lanzén, E. (2021). *A comparative review of SMOTE and ADASYN in imbalanced data classification*. Bachelor Degree. Uppsala Universitet, Sweden.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321–357.

Chemchem, A., Alin, F., and Krajecki, M. (2019). Combining SMOTE sampling and machine learning for forecasting wheat yields in France. In *Proceedings of the IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pp. 9–14. Sardinia, Italy.

Chemchem, A., and Drias, H. (2015). From data mining to knowledge mining: Application to intelligent agents. *Expert Systems with Applications*, 42(3), 1436–1445.

Davagdorj, K., Lee, J. S., Pham, V. H., and Ryu, K. H. (2020). A comparative analysis of machine learning methods for class imbalance in a smoking cessation intervention. *Applied Sciences*, 10(9), 3307.

Dynamics 365. (2022). *What is dynamics 365?* [Online URL: https://dynamics.microsoft.com/th-th/what-is-dynamics365/] accessed on July 24, 2022.

He, H., Bai, Y., Garcia, E. A., and Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 1322–1328. Hong Kong.

Hripcsak, G., and Rothschild, A. S. (2005). Agreement, the f-measure, and reliability in information retrieval. *Journal of the American Medical Informatics Association*, 12(3), 296–298.

Intayoad, W., Kamyod, C., and Temdee, P. (2018). Synthetic minority over-sampling for improving imbalanced data in educational web usage mining. *ECTI Transactions on Computer and Information Technology*, 12(2), 118–129.

Khamphakdee, N., and Seresangtakul, P. (2021). Sentiment analysis for Thai language in hotel domain using machine learning algorithms. *Acta Informatica Pragensia*, 10(2), 155–171.

Klintberg, A. (2017). Explaining precision and recall. *Medium.* [Online URL: https://medium.com/@klintcho/explaining-precision-and-recall-c770eb9c69e9] accessed on July 24, 2022.

Microsoft. (2019). *Microsoft SQL Server.* [Online URL: https://www.microsoft.com/en-gb/sql-server/sql-server-2019] accessed on July 24, 2022.

Mohajon, J. (2020). Confusion matrix for your multi-class machine learning model. *Medium.* [Online URL: https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826] accessed on July 24, 2023.

PyThaiNLP. (2022). *PyThaiNLP Library.* [Online URL: https://pythainlp.org/] accessed on July 24, 2022.

Python Programming Language. (2022). *Python.* [Online URL: https://www.python.org] accessed on July 24, 2022.

scikit-learn. (2022). *GridSearchCV.* [Online URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html] accessed on July 24, 2022.

Sreejith, S., Nehemiah, H. K., and Kannan, A. (2020). Clinical data classification using an enhanced SMOTE and chaotic evolutionary feature selection. *Computers in Biology and Medicine*, 126, 103991.

Wang, K.-J., Adrian, A. M., Chen, K.-H., and Wang, K.-M. (2015). A hybrid classifier combining Borderline-SMOTE with AIRS algorithm for estimating brain metastasis from lung cancer: A case study in Taiwan. *Computer Methods and Programs in Biomedicine*, 119(2), 63–76.