A Minimum Error-Based PCA for Improving Classifier Performance in Detecting Financial Fraud

Bayu Nur Pambudi[1], Silmi Fauziati[2], and Indriana Hidayah[2*]

[1]*Financial Transaction Reports and Analysis Center*
Jl. Ir. Haji Juanda No.35, Jakarta, 10120, Indonesia

[2]*Department of Electrical and Information Engineering, Universitas Gadjah Mada*
Jl. Grafika No. 2, Yogyakarta, 55281, Indonesia

[*]*Corresponding author. Email: indriana.h@ugm.ac.id*

Abstract— The main challenge of data mining approaches to detect fraud in financial transaction data is the imbalance of data classes in available datasets, with a much smaller fraud class proportion than the non-fraud. This imbalance affects the f1-score to be low due to imbalanced precision and recall. Therefore, the model can predict one class well, but it does not apply to another class. In addition, the lengthy training time duration and high computational resource requirements in implementing data mining also make them a particular concern. Therefore, solely handling imbalanced data is still insufficient to produce the expected performance. Reduction of data dimensions can be a solution to increase the speed of the process. However, this method actually reduces the classifier's performance when it comes to classification. Furthermore, this study intends to improve the performance of the data mining approach based on the Support Vector Machine (SVM) classifier aiming at detecting financial fraud transactions. The SVM performance was refined by tuning the kernel and hyperparameter integrated with the Random Under Sampling (RUS) and our Minimum error-based Principal Component Analysis (MebPCA). The RUS was used to handle imbalanced data, while MebPCA modified data dimension reduction techniques based on classification errors to speed up computational time without disturbing the performance of SVM. This combination improves the classifier's performance in detecting fraud effectively with a precision improvement of 29.31% and f1-score of 19.8%, and efficiently reduces the duration of training time significantly by 36.39% compared to previous research regarding the SVM method for fraud detection.

Keywords— data mining; financial fraud detection; MebPCA; RUS; SVM

I. INTRODUCTION

This research is motivated by the fact that money laundering is classified as a transnational and extraordinary crime. Thereby, it becomes a serious threat to financial institutions as well as the political and economic stability of a country [1]. Fraudsters are constantly working on developing their approaches to exploit the vulnerabilities of current money laundering countermeasures, many of which target the financial sector [2]. In this case, Anti Money Laundering is an important significance for world monetary stability.

The main indication of money laundering is a fraudulent or suspicious transaction in which the transaction does not have a clear business purpose and is carried out outside the usual transaction pattern. Traditional or manual approaches to Anti Money Laundering include identification through Customer Due Diligence, detection by Transaction Analysts, prevention by anti-money laundering socialization, and monitoring by the Supervisory and Regulatory Institution. The transactional banking data volume that continues to increase has caused these approaches to be no longer optimal because the opportunity for manual fraud detection has become increasingly inaccurate due to fatigue and hassles of transaction analysts as humans [3].

Data mining techniques are needed to help the transaction analysts work faster, more effectively, and efficiently. However, there are problems in implementing data mining for fraud detection, including imbalanced datasets [4], data mining methods and metrics that are very sensitive to imbalanced data [5], long training time duration, and high computational resource requirements in applying data mining. The handling of imbalanced data, selecting appropriate metrics [6], and choosing the data dimension reduction method with adjustments for classification can help improve classifier performance [7].

Our study utilized the PaySim Dataset, a dataset of non-real financial transactions that has been generated from simulations in the study [8]–[10]. The PaySim dataset produced by these studies was imbalanced, with the proportion of fraud and non-fraud classes being far from equal. Hence, it could not be directly implemented in the data mining process and this imbalance needs to be handled to produce the expected performance. Several previous studies have used Paysim Dataset for fraud detection, including [5], [8]–[12].

Various resampling techniques such as Random Over Sampling (ROS), Synthetic Minority Over-sampling Technique (SMOTE), and Random Under Sampling (RUS) have been compared to handle imbalanced datasets [13], but neither method was significantly better because it would depend on the degree of the dataset imbalance.

Handling imbalanced datasets alone is not sufficient to produce the expected performance [14]. The SVM classifier has been chosen in our study because fraud detection is a binary classification problem so the SVM concept fits well. It is also supported by the results of the SVM classification in previous studies [5] and [12], which, despite its insignificancy, are better than other methods. However, some parts were still

Received 25 March 2022, Accepted 15 June 2022, Published 27 June 2022.
DOI: https://doi.org/10.15294/jte.v14i1.35787

unperformed in studies [5] and [12]. Several issues have not been handled in the previously proposed classification method, i.e., imbalanced data as well as Kernel and SVM hyperparameter tunings. Both of these are part of the main concern of our research so that this research is important in improving the performance of SVM.

Improving SVM performance can also be done by evaluating the dimensions of the datasets through the Principal Component Analysis (PCA) approach. Studies [15]–[17] show how dimension reduction approaches in improving engine performance. Nevertheless, when it comes to pattern recognition, such as face recognition or even fraud detection, modification techniques based on subspace similarity in [15], multilevel approach in [16], and feature extraction in [17] may still mix elements of data. For example, when recognizing a human face with and without a mustache, there can be confusion in getting the right density of a certain face area, as the false positive that occurred in the case of fraud detection.

Reducing data dimensions using PCA can be a solution to increase process speed. However, this method actually reduces the performance of the classifier in terms of classification. Therefore, PCA implementation must go through a scalable approach.

This research continues previous study [11] in which tuned Support Vector Machine (SVM) was combined with Random Under Sampling (RUS) to increase the classifier's performance for fraud detection purposes. The study [11] results have shown a significantly improved performance in precision of 40.82% and f1-score of 22.79% compared to previous work in research [5]. However, this previous research [11] still left a problem with the lengthy duration of model training. Our research intends to improve previous models' performance by combining the previous method [11] with Minimum error-based Principal Component Analysis (MebPCA) to reduce training time without reducing model performance. This is very important regarding the effectiveness of classifiers and resource efficiency.

Improving the performance of SVM in classifying fraud and non-fraud instances is the main objective of this research. This SVM performance improvement is achieved by tuning the kernel and hyperparameters integrated with RUS and our MebPCA. This combination has implications for the classifier's ability to detect fraudulent transactions on imbalanced financial transaction datasets effectively with increased precision and f1-scores, as well as efficiently reducing training time duration significantly. This article will further elaborate on the model for financial fraud detection, Paysim Dataset, how SVM tuning is done and how to balance imbalanced data, MebPCA, and what metrics are suitable for evaluating imbalanced data.

II. METHOD

This section describes the optimization of the model, starting with the proposed model for financial fraud detection, then the dataset used is explained in the "PaySim Dataset" subsection. How to handle imbalanced data in PaySim Dataset and tune the classifier will be discussed in the "Imbalanced Data Handling and SVM Parameters Tuning" subsection. Next, Minimum error-based PCA and Minimum Classification Error are proposed as approaches to improve classifier efficiency and performance. Classifier performance is then measured in terms of metrics discussed in the "Metrics for Evaluating Imbalanced Data" subsection.

A. Financial Fraud Detection Model

The proposed system model for detecting financial fraud is shown in Figure 1. In Figure 1, stratified sampling is applied to split the dataset into 70% train set and 30% test set.

Stratified sampling aims to maintain the data generated from the dataset separation represents the same proportion of each class as in the initial dataset. Figure 1 shows that preprocessing is applied to the train data first, then RUS is used to balance the class between fraudulent and non-fraudulent transactions. This process shortens model training time and degrades model performance. Performance degradation is then

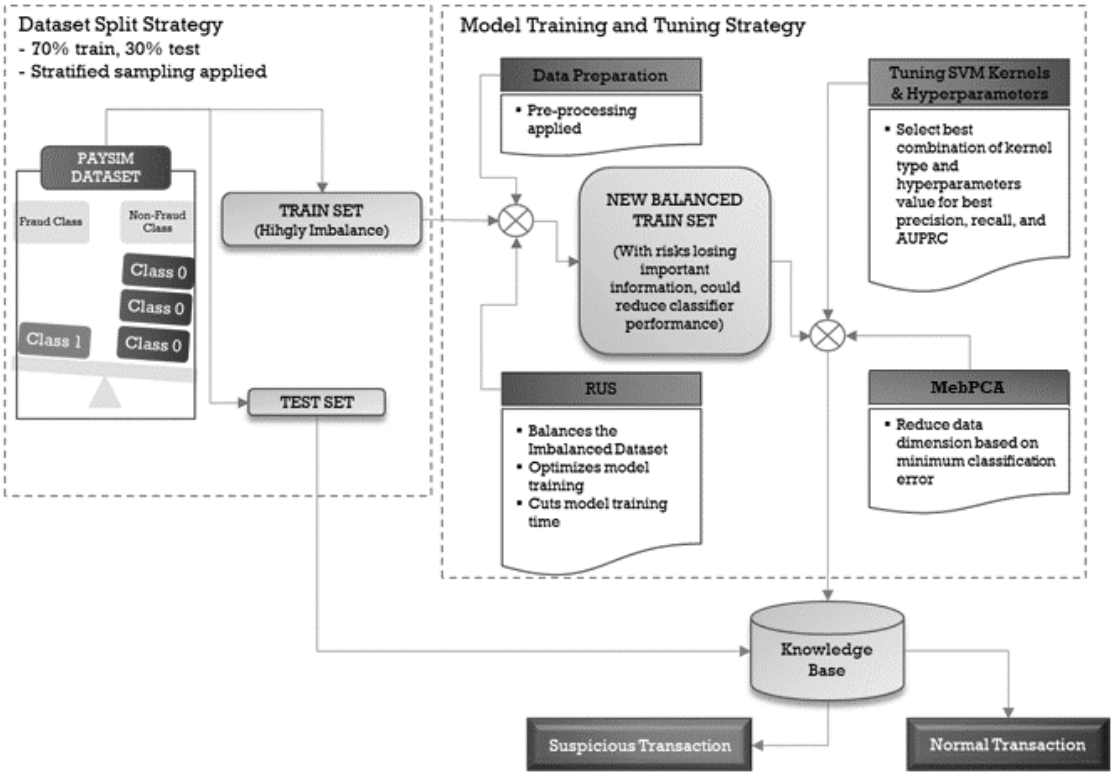

Figure 1. Proposed model for financial fraud detection

overcome by tuning in SVM parameters using 5-fold cross-validation. In this 5-fold cross-validation, the dataset is divided into 4 subsets of train data and 1 subset of test data, which is done 5 times, with each piece of data having the opportunity to become a test set.

The tuning results were then combined with MebPCA. Specifically, MebPCA modified the technique of reducing data dimensions based on the minimum classification error. This approach cuts the duration of model training time and maintains SVM performance. Furthermore, the results of the training were stored as a classifier model in the knowledge base for the prediction of test sets, whether as normal transactions or suspicious transactions.

The combination scheme of RUS+Tuned_SVM+MebPCA in extracting features can be seen in Figure 2. RUS forms a balanced data class (instance) by freezing minority class (fraud) and randomly taking majority class (non-fraud) as much as fraud class data. Balanced instances optimized training and reduced model training time due to a reduction of majority instances. Classifiers worked best when the majority and minority classes were rebalanced [4]. While MebPCA reduced the data dimensions data by considering the value of classification errors using the Tuned SVM Classifier.

The classification error value for each *N* component was calculated, and then the smallest classification error value was taken. *N* components with the smallest classification error were used for classification because a small classification error value did not interfere with the classifier's performance. Thus, this MebPCA sped up computing time and maintained SVM performance.

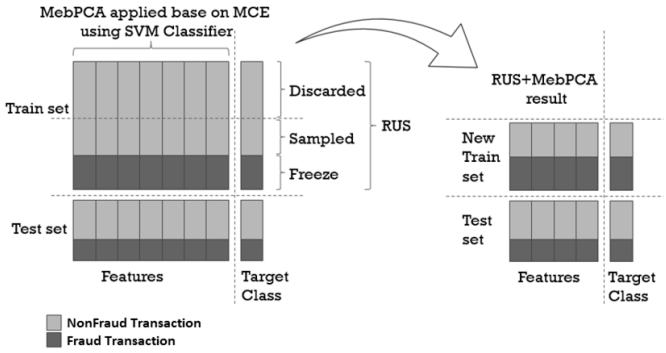

Figure 2. RUS+Tuned_SVM+MebPCA in extracting features

B. PaySim Dataset

The research material used in this study is PaySim Dataset, which has been obtained from https://www.kaggle.com/ntnu-testimon/paysim1. The PaySim is imbalanced data, with the proportion of fraud and non-fraud classes being far from equal. Hence, it could not be directly implemented in the data mining process to produce the expected performance, so this imbalance needs to be handled first. This dataset has 6.3 million transactions distributed in various transaction types, as shown in Figure 3.

Fraudulent transactions are only found in Transfer and Cash-out transactions with a less than 1% fraud distribution, so the proportion of fraud is much smaller than non-fraud. Table I shows the detail about this distribution. The use of a public dataset in this study due to the difficulty in obtaining real banking transaction data because of bank secrecy law provisions. In this study, only the Transfer transaction type was used to test the proposed model.

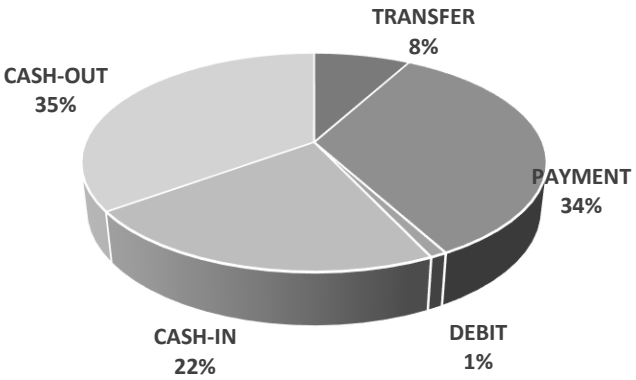

Figure 3. Transaction distribution in PaySim Dataset

TABLE I. FRAUD DISTRIBUTION IN PAYSIM DATASET

No	Type of Transactions	Non-Fraud	Fraud	Total Records
1.	*TRANSFER*	528,812	4,097	532,909
2.	*PAYMENT*	2,151,494	0	2,151,494
3.	*DEBIT*	41,432	0	41,432
4.	*CASH-IN*	1,399,284	0	1,399,284
5.	*CASH-OUT*	2,233,384	4,116	2,237,500

C. Imbalanced Data Handling and SVM Parameters Tuning

This study controls class imbalance on the PaySim Dataset with RUS by freezing minority instances (fraud) and randomly taking majority instances (non-fraud) as much as minority instances to establish balanced instances between fraud and non-fraud classes. Balanced classes avoid underfit or overfit to get an optimal training model [18]. Reducing most instances in RUS speeds up model training time and risks eliminating useful information that can degrade model performance.

SVM was used to find the best hyperplane as a function separating fraud and non-fraud instances in the input space. This hyperplane was represented as the kernel, determined by measuring the hyperplane's margin and finding its maximum point [19]. SVM parameters tuning aimed to select the kernel and hyperparameter to obtain the value combination producing the best precision and recall based on the cross-validation results on the train set. Kernel selection was applied to the linear, polynomial, radial basis function (RBF), and sigmoid kernel. Meanwhile, hyperparameter selection in this study was performed only on gamma (ɤ) and C parameters. These parameters tuning would improve the classifier's performance which was down due to the implementation of RUS.

The kernel functions commonly used in the SVM method are linear, RBF, polynomial, and sigmoid. Mathematically, the linear kernel can be represented as follows:

$$K(X,Y) = X^T Y \quad (1)$$

where X is input instance, X^T is transpose X, and Y is output instance. While the polynomial kernel is formulated as follows:

$$K(X,Y) = (\gamma . X^T Y + r)^d, \gamma > 0 \quad (2)$$

where r, d, and γ are kernel parameters. The RBF kernel is represented by the following formula:

$$K(X,Y) = exp(-\gamma . \|X - Y\|^2), \gamma > 0 \quad (3)$$

While the sigmoid kernel is formulated as follows:

$$K(X,Y) = tanh(\gamma . X^T Y + r) \quad (4)$$

Selection of the appropriate kernel function is very important because it will determine the feature space where the classifier function will be searched. As long as the kernel

functions match, SVM will operate correctly even if it doesn't know which mapping function to use. Output classifier as a balanced dataset is defined as follows:

$$s = sign(\sum_{k=1}^{K} f_k) \quad (5)$$

This formula is further explained in pseudocode for handling imbalanced data and SVM parameters tuning as shown in Figure 4.

```
Determine R as the original training set
for k = 1,2,..,K do
    Form the R_k subset containing the same number of fraud and non-fraud classes by randomly taking instances with or without replacement at the N_d/N_m level. N_d is the sample size of fraud (the desired size), while N_m is the non-fraud sample size (original class size)
    From subset R_k, train a classifier f_k
    Output classifier as a balanced dataset,
        s = sign(∑_{k=1}^{K} f_k)
end for
To train the model with dataset s:
Set grids of parameter: type of svm kernel, range of C & gamma
Set strategy by defining the number of k for k-fold cross-validation
Set scoring parameters: f1 as a harmonic average of precision & recall
for (score in scoring parameter) do
    for (k-fold cross-validation in the parameter grid) do
        Find the best proportion of scoring parameter
        Find the best combination from the parameter grid (best_params)
    end for
end for
Output the best solution found, best_params, as the final result
```

Figure 4. Pseudocode for RUS and SVM parameters tuning

D. MebPCA and Minimum Classification Error

Principal Component Analysis (PCA) is a dimension reduction technique that is applied to data that has multicollinearity, which is a condition that shows a perfect or almost perfect linear relationship between some or all of the variables. This multicollinearity is determined by the number of conditions (k) as follows:

$$k = \frac{\lambda_{max}}{\lambda_{min}} \quad (6)$$

λ is the eigenvalue of the covariance variable matrix, with the limits of the condition number (k) as follows.

- k < 100; there is weak multicollinearity.
- 100 ⩽ k ⩽ 1000; moderate to strong multicollinearity occurs.
- k > 1000; there is a very strong multicollinearity.

If the data has multicollinearity between variables, then PCA is applied first to the dataset used. Thus, a number of principal components *(PC)* which are orthogonal to each other will be formed.

The principal component is a form of variable transformation which is a linear combination of variables. The process of forming the principal component in detecting suspicious or fraudulent financial transactions is by determining the X matrix which is the data of banking customers' financial transactions. From this X matrix, then calculate the covariance matrix to determine the eigenvalues (λ). Based on the eigen matrix, the principal components *(PC)* formed are as follows:

$$\begin{aligned} PC_1 &= z_j v_{j1} = z_1 v_{11} + z_2 v_{21} + \cdots + z_p v_{j1} \\ PC_2 &= z_j v_{j2} = z_1 v_{12} + z_2 v_{22} + \cdots + z_p v_{j2} \\ &\vdots \\ PC_p &= z_j v_{jp} = z_1 v_{1p} + z_2 v_{2p} + \cdots + z_p v_{jp} \end{aligned} \quad (7)$$

where *z* and *v* are data variables, *PC1* is first principal component, *PC2* is second principal component and so on.

In conventional PCA, an eigenvector with a greater eigenvalue is chosen as the main component, making minimum data variance. From the classification view, this argument is less meaningful because classification demands a series of projection vectors that can provide the highest discrimination among different classes. Therefore, choosing the main component with the largest eigenvalue as the basis for dimensional reduction results in different class recognition not being optimal [20].

In addition, PCA is a type of statistical and unsupervised algorithm for extracting features without using class information from input data. In certain cases, the extracted features may not be suitable for classification. Therefore, principal components are not always useful for classification because they are not the most discriminating features [21]. Some components with small eigenvalues may have better classification performance than those with larger values.

Eigenvalues and eigenvectors are fundamental parameters, hence, they are impossible to eliminate. Therefore, a PCA modification applicable for classification purposes is a slightly altered approach concerning minimizing errors rather than the eigenvalue approach. It is why the approach is called the Minimum error-based PCA (MebPCA). The modification of the PCA concept is basically to choose the feature vector projected along the k-minimum error instead of considering the feature vector value based on the k-largest eigenvalue. Pseudocode for MebPCA is shown in Figure 5.

```
Define pipeline for combining PCA and SVM
Define range of n_components to be calculated
for (N_components in range of N_components) do
    Pipelining PCA and SVM with selected parameters.
    Fit the classifier combination
    Calculate the classification error using zero-one-loss for each N_component and sort ascending
    Select the n_component with the minimum classification error value
end for
```

Figure 5. Pseudocode for MebPCA

In Figure 5, the pipeline is a method for carrying out several tasks together but in different stages, which is flowed continuously to the processing unit. In this way, the processing unit always works, with MebPCA output from the selected *N*_component range will be the input of the SVM classifier. The error rate is then determined by the zero-one-loss function from the Python Library. It was based on the argument that the binary classification includes fraud detection. The minimum value of classification error is chosen from these error rate values. In other words, the classification error value for each component was calculated, and then the smallest error value was taken.

The Minimum Classification Error (MCE) [22] is a type of discriminant analysis that reaches a minimum classification error using the gradient descent method. This method applies the loss function as a differentiated function of the misclassification size, defined as a close estimate of the actual misclassification. As such, the MCE algorithm is a more direct way to achieve a minimum misclassification level than conventional discriminatory training algorithms. This algorithm can be summarized in the following procedure:

- Define a discriminant function with Simple Euclidean Distance:

$$D_i^{(p)} = \left\| TX^{(p)} - \mu_i \right\|^2 \quad (8)$$

where T is the transformation matrix ranked d ($d \leqslant D$), D is the original data dimension, and μ_i is the mean vector of class i.

- Determine the size of the classification error by embedding the classification criteria in the overall minimum classification error formulation.

$$d_k(x^{(p)}) = -g_k(x^{(p)}, \Lambda) + \sum_{for\ all\ i \neq k} \frac{1}{N-1} g_i(x^{(p)}, \Lambda) \quad (9)$$

where $g_i(x^{(p)}, \Lambda)$, $i = 1, 2, ..., N$ is a set of discriminant functions; $x^{(p)}$ is the pth observation vector; N is the number of classes; Λ is the parameter set for each class; ζ represents a positive number for N-1.

- Determine the Loss function as a monotonic sigmoid function suitable for the gradient algorithm to smooth the size of misclassification. The sigmoid function is used because it is a zero-one function suitable for the gradient algorithm.

$$L(x^{(p)}) = \frac{1}{1 + e^{-\alpha d(x^{(p)}, \Lambda)}} \quad (10)$$

The total loss function is defined as:

$$L = \sum_{p=1}^{P} L^{(p)} \quad (11)$$

E. Metrics for Evaluating Imbalanced Data

Classification performance in imbalanced data domain is more effective if measured independently from positive and negative classes [4]. This measurement is based on the following confusion matrix in Table II.

TABLE II. CONFUSION MATRIX MODEL

True Class	Predicted Class	
	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

From this confusion matrix model, appropriate metrics for imbalanced data can be formed, including precision, recall, and f1-score, expressed in the following formulas:

$$Precision(Fraud) = \frac{TN}{TN+FN} \quad (12)$$

$$Precision(NonFraud) = \frac{TP}{TP+FP} \quad (13)$$

$$Recall(Fraud) = \frac{TN}{TN+FP} \quad (14)$$

$$Recall(NonFraud) = \frac{TP}{TP+FN} \quad (15)$$

TP reflects correctly classified positive instances, whereas FP means negative instances misclassified as TP. TN shows correctly classified negative instances, while FN reflects positive instances misclassified as TN.

Precision and recall state how precise and robust a model is. These two metrics are linearly unrelated. If a model has good precision, it does not necessarily work well at recall, and vice versa. Another way to evaluate the model's performance on imbalanced data is to take the harmonic average between precision and recall. This metric is called f1-score, which is expressed as follows:

$$\text{F1 } score = 2 . \frac{precision\ .\ recall}{precision + recall} \quad (16)$$

In contrast, AUPRC is a trade-off between precision and recall using different probability thresholds. A perfect AUPRC means that the model can find all positive instances (perfect recall) without incorrectly classifying negative instances as positive (perfect precision). All of these metrics are more accurate when working with imbalanced data [23].

III. RESULTS AND DISCUSSION

This section discusses the tuning results obtained from the selection of kernels and hyperparameters in SVM and the number of components for MebPCA based on classification errors. The model's performance is evaluated using metrics such as precision, recall, f-measure (f1-score), and Area Under Precision-Recall Curve (AUPRC). In addition, the duration of training time is also a concern to assess the model's performance.

A. Parameters Selection for Training

The best values of each parameter resulted from 5-fold Cross-Validation (CV) in the tuning process of the kernel and hyperparameter are shown in Figure 6. The use of 5-fold CV aims to reduce computation time while maintaining estimation accuracy. This is because the use of the default 10-fold cross validation on large data demands large computing resources and will require much longer computational time for 10 times subset splitting, training & testing. In addition, the opportunity for 1 subset for testing containing fraud data will be smaller because the fraud class is much smaller than non-fraud, so the testing subset will be biased.

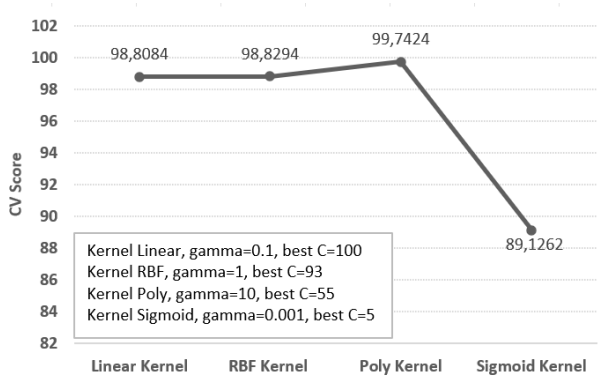

Figure 6. 5-Fold CV mean test score for each kernel

Figure 6 shows that the best mean test score for each kernel is achieved in a combination of C and gamma parameter values. The best score of the linear kernel is achieved at a score of 0.988084 when gamma (γ) = 0.1 and C = 100, while the RBF score 0.988294 is achieved when gamma (γ) = 1 and C = 93. The results of CV scores on poly kernels reach 0.997424, obtained when gamma (γ) = 10 and C = 55, and the sigmoid kernel reaches a score of 0.891262 when gamma (γ) = 0.001 and C = 5. It can be seen that poly kernel is the best kernel, followed by RBF, linear, and sigmoid in sequence.

Gamma (γ) is a parameter for a non-linear hyperplane kernel, it is used to control the speed of the learning process. While C is the penalty parameter associated with the error, it is is used to control the trade-off between hyperplane margins and classification errors.

Furthermore, the number of principal components *(N)* for MebPCA was determined based on the MCE by utilizing the zero-one-loss library contained in the Python loss function library. The results of the Classification Error (CE) measurement for the application of MebPCA are shown in Table III.

TABLE III. CE-BASED N FOR MEBPCA DEPLOYMENT

MebPCA Combined With					
SVM (linear)		*RUS + SVM (linear)*		*RUS + Tuned_SVM*	
N	***CE***	***N***	***CE***	***N***	***CE***
1	0.016365	1	0.092593	1	0.129494
2	0.015752	2	0.086370	2	0.108079
3	0.014539	3	0.081532	3	0.105706
4	0.014513	4	0.067619	4	0.012967
5	0.004971	5	0.067539	5	0.009582
6	0.004531	6	0.056984	6	0.005357
7	0.004531	7	0.056984	7	0.005357
8	0.004531	8	0.056984	8	0.005357
9	0.004531	9	0.056984	9	0.005357

The number of CE-based PCA components for MebPCA deployment forms a certain pattern trend. Trend in the results of classification error measurements for each number of MebPCA components are shown in Figure 7.

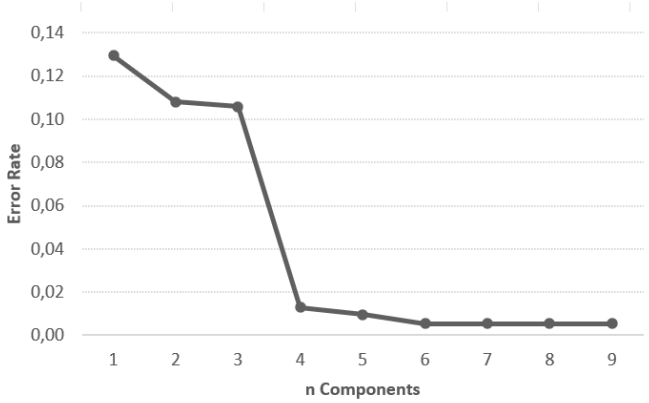

Figure 7. Classification error for each number of MebPCA components

Figure 7 shows that the number of components with the smallest classification error value is obtained at value n starting at 6 and converging to the next value of n. Therefore, the value of n=6 is used in the MebPCA implementation. A small error value does not significantly affect the classifier's performance. Thus, MebPCA speeds up computation time by reducing data dimensions and maintaining SVM performance. Then, the combination of parameters from the tuning is applied to the fraud prediction in the test set.

B. Model Performance Evaluation

The results of our re-measurement on various SVM combination models of previous studies and their comparison with our model results are shown in Table IV. Our model (RUS+Tuned_SVM+MebPCA) and previous model with a slight modification (RUS+Tuned_SVM [11]), being compared with the previous research model (RBF SVM [5]), are able to improve precisions from 0.58 to 0.75 and 0.76 or improvement of 29.31% and 31.03%. Improvement of f1-scores reached 17.8% and 19.18% from 0.73 to 0.86 and 0.87, while the recall result increased by 2%. A slight change in previous study [11] was the application of stratified sampling to the dataset separation, which caused slightly different results to our current results. Whereas if it is compared with linear SVM model [12] or combined with RUS [24], are able to improve f1-score and recall perfectly. This has the effect of significantly improving the misclassified fraud.

The RUS+SVM (Kernel:linear) model [24] did not use PaySim as the main dataset. Therefore, adjustment and re-measurement of the model with PaySim was carried out. Likewise, the SVM (Kernel:linear) [12] model, even though it already used PaySim, but due to the number of samples was different, then re-measurements were also carried out.

F1-score is the harmonic average of precision and recall. This measure shows how precise and robust a model is. In this case, it is important to have a trade-off between precision and robustness. When precision and recall are proportional to each other, the F1-score will be maximized. The F1-score will be degraded when only one of the metrics is optimized. Because F1-scores are available per class, for example fraud and non-fraud classes, the F1-score for fraud classes will be more important than non-fraud classes. This is because it is more important to classify fraud cases correctly than non-fraud ones.

TABLE IV. COMPARISON OF MEASUREMENT RESULTS

No	Methods	Precision	Recall	F1-Score	AUPRC	Training Time (hh:mm:ss)	Misclassified Non-Fraud	Misclassified Fraud
1.	SVM *(kernel=linear)* [12]	0.96	0.75	0.84	0.96	00:08:36 (516.02s)	36	304
2.	SVM *(linear)* + MebPCA (n=6)	0.96	0.75	0.84	0.96	00:01:44 (104.13s)	36	304
3.	RUS + SVM *(kernel=linear)* [24]	0.22	1.00	0.36	0.85	00:00:01 (1.24s)	4,275	1
4.	RUS + SVM *(linear)* + MebPCA (n=6)	0.22	1.00	0.36	0.85	0.963s	4,275	1
5.	RUS+*Tuned_SVM* [11] (*poly*,γ=10,*C*=55)	0.76	1.00	0.87	0.91	00:04:52 (292.25s)	377	2
6.	RUS+*Tuned_SVM* + MebPCA (n=6)	0.75	1.00	0.86	0.85	00:03:05 (185.9s)	399	3
7.	(SVM *kernel* RBF, *class weight*:16) [5]	0.58	0.98	0.73	0.98	-	436	7

It can be seen in Table IV that the implementation of MebPCA did not significantly degrade the classifier's performance. It only affected the duration of the model training. All methods in which MebPCA was applied had improved training time (training time was getting shorter). In contrast, the implementation of RUS caused performance degradation due to the reduction of majority (non-fraud) classes. It could be seen in the results of SVM (kernel: linear) [12] that the performance dropped after RUS was applied (RUS+SVM_Linear) [24], then the performance was improved again by tuning the kernel and hyperparameter (RUS+Tuned_SVM) [11].

Table IV also shows how the effect of applying RUS, MebPCA, and tuning to the kernel and hyperparameter in changing the classifier's performance. This trend is shown in Figure 8 where RUS+Tuned_SVM+MebPCA significantly improve precision and the f1-score when compared to methods without any tuning or by default hyperparameters [RUS+SVM (linear)].

In Figure 8, the RBF kernel SVM model [5] and RUS+SVM (kernel=linear) [24] shows that the recall result is very high, while the precision is very low. In contrast, SVM (kernel: linear) [12] yields high precision with relatively low recall. These two situations cause the f1-score to be low because recall and precision are not balanced, so the AUPRC is biased. A high recall causes the model to be able to predict fraud class well, while poor precision means that non-fraud class cannot be well predicted by the model in these studies [5], [24]. This model only tuned the class-weight parameter with preprocessing, which might differ from our study. On the other hand, a low recall causes the model [12] predict fraud incorrectly much larger than the error in predicting non-fraud. In addition, the handling of an imbalanced dataset was also not implemented, so the model was underfit because it did not get perfect training.

Furthermore, AUPRC results in model RUS+Tuned_SVM & RUS+Tuned_SVM+MebPCA are shown in Figure 9 and Figure 10. The AUPRCs of both models were lower than the models [5], [12] because our AUPRCs reflected a broader and evener area under the curve due to more balanced precision and recall. As a result, our models could suppress misclassification in the fraud and non-fraud classes equally well.

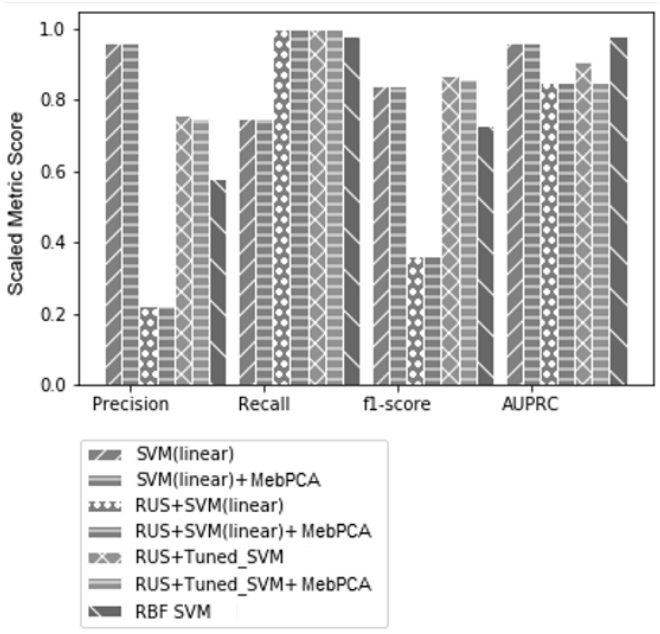

Figure 8. Effects of RUS, MebPCA and tuning parameters on SVM performance

The measurement results on the confusion matrix for both models are shown in Table V. This table shows that both models can predict fraud and non-fraud classes well. In [RUS+Tuned_SVM], 377 (0.51%) non-fraud transactions were misclassified as fraud transactions, and 2 fraud transactions (0.16%) were misclassified as non-fraud. Whereas in [RUS+Tuned_SVM+MebPCA], 399 (0.54%) non-fraud transactions were misclassified as fraud, and 3 fraud transactions (0.24%) were mistakenly predicted as non-fraud transactions. A comparison of model training time to evaluate the effect of applying RUS and MebPCA is shown in Figure 11.

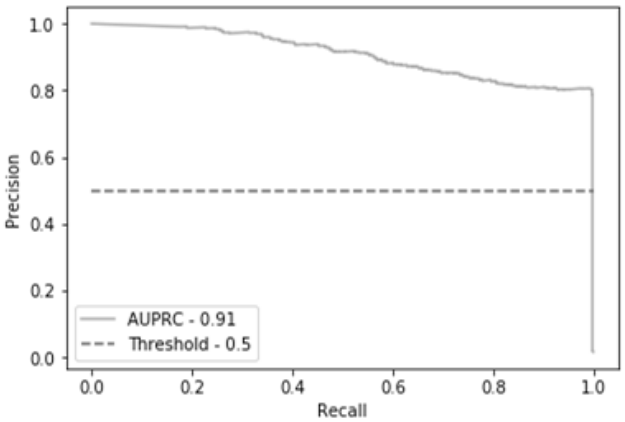

Figure 9. AUPRC of RUS+Tuned_SVM

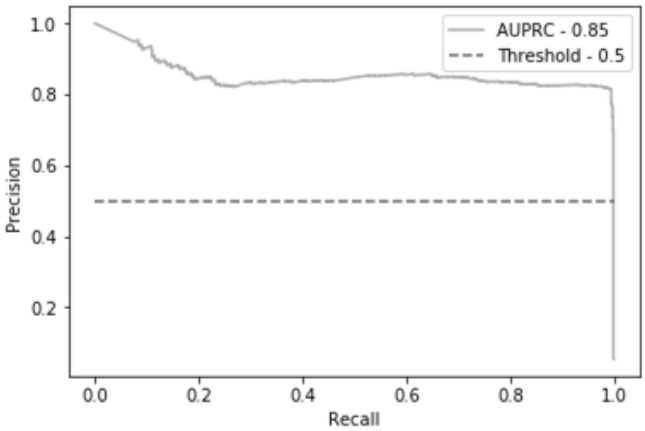

Figure 10. AUPRC of RUS+Tuned_SVM+MebPCA

TABLE V. CONFUSION MATRIX RESULTS

True Class	*Predicted Class*			
	RUS+Tuned_SVM		**RUS+Tuned_SVM+ MebPCA**	
	Non-Fraud	***Fraud***	***Non-Fraud***	***Fraud***
Non-Fraud	73,433	377	73,411	399
Fraud	2	1,226	3	1,225

In Figure 11, the combination of RUS and MebPCA together cuts the model training time significantly. This is because RUS handles imbalanced datasets and reduces model training time. Meanwhile, mPCA modifies the data dimension reduction technique based on classification error, which will not only speed up computation time, but also maintain SVM performance. If our two approaches are compared, it can be seen that there is a significant reduction in training time of 36.39% in the [RUS+Tuned_SVM+MebPCA] method, although there is a slight decrease in the precision and f1-score results.

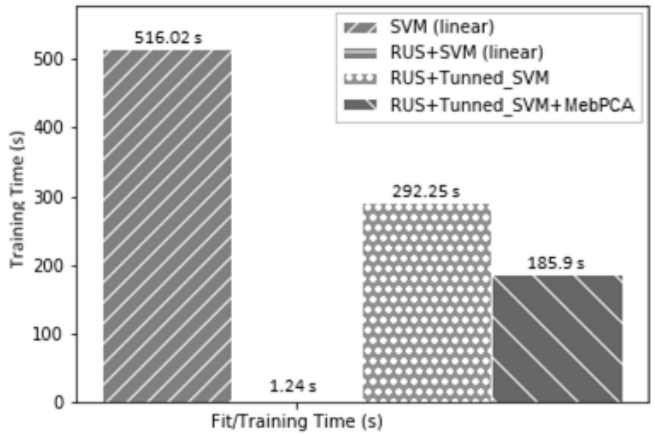

Figure 11. Comparison of model training time

When dealing with a dataset with large dimensions but limited computing resources, the combination of [RUS+Tuned_SVM+MebPCA] was more suitable because it accommodated the speed of model training with good prediction results. Overall, it can be concluded that the MebPCA method maintains classifier performance or has no major impact on classification results, but it is significant for the computational time.

From the results above, it can be seen that the under-sampling method specifically handles class imbalances in the PaySim Dataset and shortens the model training time. SVM tuning was performed to improve the classifier's performance, which had been declining due to the application of RUS. The best tuning results for Transfer transaction types are poly kernels with gamma=10 and C=55. The use of PCA helps reduce computing time significantly, but it risks degrading SVM performance due to reduced dataset dimensions. The application of MebPCA based on the Minimum Classification Error (MCE) cuts training time and maintains SVM performance so that MebPCA is better than PCA for classification purposes. The assessment to determine the number of components in traditional PCA is subjective, depending on the desired accuracy value. At the same time, MebPCA explicitly selects the number of components based on the lowest classification error value that does not interfere with the classifier performance. The number of components selected based on the lowest classification error value is 6. This number of PCA components is determined based on the Minimum Classification Error (MCE) which is calculated by utilizing the zero-one-loss library contained in the Python loss function library. This is based on the argument that fraud detection belongs to the category of binary classification.

IV. Conclusion

The research intends to combine the tuned SVM, RUS, and MebPCA methods in improving the performance of SVM in classifying fraud and non-fraud cases. The proposed model can reduce the number of misclassifications, predict fraud and non-fraud classes well, and effectively provide better classification results when compared to the methods in previous studies or if it is run without parameters tuning, with a precision improvement of 29.31%, recall 2% and f1-score of 19.8% compared to the previous study. In addition, the proposed model also becomes efficient, which can significantly reduce training time without reducing classifier performance, with an improvement in training time efficiency of 36.39%. For further work, PaySim Dataset can be developed by modifying this dataset to accommodate digital currency transactions such as bitcoins that are not included in custodian bank services and electronic money. Then deep learning can be applied to predict the possibility of the emergence of new suspicious transaction patterns. Furthermore, transaction data analysis and processing can be performed with a graph database to provide a visual representation of transaction patterns as a recommendation for decision-makers.

Acknowledgment

The authors are grateful for the support from Universitas Gadjah Mada, especially the Department of Electrical and Information Engineering for this publication, and appreciate the Indonesian Ministry of Communications and Information Technology's program in advancing education in Indonesia through e-Government scholarships.

References

[1] N. S. Alfaiz and S. M. Fati, "Enhanced Credit Card Fraud Detection Model Using Machine Learning," *Electronics*, vol. 11, no. 662, pp. 1–16, 2022, doi: https://doi.org/10.3390/electronics11040662.

[2] W. Hilal, S. A. Gadsden, and J. Yawney, "Financial Fraud: A Review of Anomaly Detection Techniques and Recent Advances," *Expert Syst. Appl.*, vol. 193, p. 116429, 2022, doi: 10.1016/j.eswa.2021.116429.

[3] S. Stefanov, D. Georgieva, and J. Vasilev, "Issues in the Disclosure of Financial Information by Multinational Enterprises," *TEM J.*, vol. 11, no. 1, pp. 5–12, 2022, doi: 10.18421/TEM111-01.

[4] T. Le, "A comprehensive survey of imbalanced learning methods for bankruptcy prediction," *IET Commun.*, vol. 16, no. 5, pp. 433–441, 2022, doi: 10.1049/cmu2.12268.

[5] A. Oza, "Fraud Detection using Machine Learning," *Stanford Univ. CS229 Proj. Publ.*, vol. 261, pp. 1–6, 2018.

[6] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui, "A Comparative Evaluation of Outlier Detection Algorithms: Experiments and Analyses," *Pattern Recognit. J.*, vol. 74, pp. 406–421, 2018, doi: https://doi.org/10.1016/j.patcog.2017.09.037.

[7] A. O. Adewumi and A. A. Akinyelu, "A Survey of Machine Learning and Nature-Inspired Based Credit Card Fraud Detection Techniques," *Int J Syst Assur Eng Manag 8*, vol. 8, pp. 937–953, 2017, doi: https://doi.org/10.1007/s13198-016-0551-y.

[8] E. A. Lopez Rojas, S. Axelsson, and D. Baca, "Analysis of Fraud Controls using the PaySim Financial Simulator," *Int. J. Simul. Process Model.*, vol. 13, no. 4, pp. 377–386, 2018, doi: 10.1504/ijspm.2018.10014984.

[9] E. A. Lopez-Rojas and C. Barneaud, "Advantages of the PaySim Simulator for Improving Financial Fraud Controls," *Springer Nat. Switz. AG 2019*, vol. 998, pp. 727–736, 2019, doi: 10.1007/978-3-030-22868-2_51.

[10] E. A. Lopez-Rojas, A. Elmir, and S. Axelsson, "PaySim: A Financial Mobile Money Simulator for Fraud Detection," *Eur. Model. Simul. Symp.*, no. c, pp. 249–255, 2016.

[11] B. N. Pambudi, I. Hidayah, and S. Fauziati, "Improving Money Laundering Detection Using Optimized Support Vector Machine," *2019 Int. Semin. Res. Inf. Technol. Intell. Syst.*, pp. 273–278, 2019, doi: 10.1109/ISRITI48646.2019.9034655.

[12] R. Pech, "Fraud Detection in Mobile Money Transfer as Binary Classification Problem," *Eagle Tech. Inc Publ.*, pp. 1–15, 2019.

[13] H. Ubaya and R. S. Juairiah, "Performance of RUS and SMOTE Method on Twitter Spam Data Using Random Forest," *J. Phys. Conf. Ser.*, vol. 1500, no. 1, pp. 1–8, 2020, doi: 10.1088/1742-6596/1500/1/012130.

[14] G. Pang, C. Shen, L. Cao, and A. Van Den Hengel, "Deep Learning for Anomaly Detection: A Review," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 1–38, 2022, doi: https://doi.org/10.1145/3439950.

[15] Z. Fan *et al.*, "Modified Principal Component Analysis: An Integration of Multiple Similarity Subspace Models," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 25, no. 8, pp. 1538–1552, 2014.

[16] D. J. J. Farnell, H. Popat, and S. Richmond, "Multilevel Principal Component Analysis (mPCA) in Shape Analysis: A Feasibility Study in Medical and Dental Imaging," *Comput. Methods Programs Biomed.*, 2016, doi: 10.1016/j.cmpb.2016.01.005.

[17] S. Guo, P. Rösch, J. Popp, and T. Bocklitz, "Modified PCA and PLS: Towards a Better Classification in Raman Spectroscopy – based Biological Applications," *J. Wiley Chemom.*, no. October 2019, pp. 1–10, 2020, doi: 10.1002/cem.3202.

[18] A. Salehi, M. Ghazanfari, and M. Fathian, "Data Mining Techniques for Anti Money Laundering," *Int. J. Appl. Eng. Res.*, vol. 12, no. 20, pp. 10084–10094, 2017.

[19] A. Rojas-Domínguez, L. C. Padierna, M. J. Carpio Valadez, H. J. Puga-soberanes, and H. J. Fraire, "Optimal Hyper-Parameter Tuning of SVM Classifiers With Application to Medical Diagnosis," *IEEE Open Access J.*, vol. 6, no. March 9, 2018, pp. 7164–7176, 2018, doi: 10.1109/ACCESS.2017.2779794.

[20] M. Riera, J. M. Arnau, and A. González, "DNN Pruning with Principal Component Analysis and Connection Importance Estimation," *J. Syst. Archit.*, vol. 122, p. 102336, 2022, doi: 10.1016/j.sysarc.2021.102336.

[21] C. He, J. Li, W. Liu, and J. Peng, "A Low-Complexity Quantum Principal Component Analysis Algorithm," *Quantum Comput.*, vol. 3, pp. 1–13, 2022, doi: 10.1109/TQE.2021.3140152.

[22] N. Bhargava, A. Kumar, D. Kumar, and Meenakshi, "A Modified Concept of PCA to Reduce the Classification Error using Kernel SVM Classifier," *Int. J. Sci. Eng. Res.*, vol. 6, no. 6, pp. 1509–1513, 2015.

[23] T. Saito and M. Rehmsmeier, "The Precision-Recall Plot is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets," *PloS one. 10. e0118432*, pp. 1–21, 2015, doi: 10.1371/journal.pone.0118432.

[24] M. B. Abidine, B. Fergani, and F. J. Ordóñez, "Effect of Over-sampling Versus Under-sampling for SVM and LDA Classifiers for Activity Recognition," *Int. J. Des. Nat. Ecodynamics*, vol. 11, no. 3, pp. 306–316, 2016, doi: 10.2495/DNE-V11-N3-306-316.