Effectiveness of a hybrid Machine Learning Model for War Against Money Laundering in Nigeria

[1] Okeke Ogochukwu C., [2]Adigwe, A. I., [3]Nwokedi Chidiogo C.
anthonyadigwe@gmail.com ogoookeke@yahoo.com oxyonyi2012@yahoo.com
[2] Computer Science Department, Federal Polytechnic Oko, Anambra State
[1] Computer Science Department, Chukwuemeka Odumegwu Ojukwu University, Anambra State
[3] Computer Science Department, Federal Polytechnic Oko, Anambra State
DOI: 10.56201/ijcsmt.v10.no6.2024.pg83.95

Abstract
This paper examined the effectiveness of machine learning (ML) Models for the ongoing war against money laundering in Nigeria. In advanced economies, the rise of machine learning has revolutionized the way financial institutions and governments combat illegal financial activities emanating from mobile money transactions. By using advanced algorithms and data analysis techniques, machine learning has proven to be an effective tool in identifying suspicious financial transactions and patterns, thereby helping authorities take proactive rather than reactive measures in preventing fraudulent transactions. This study aimed to address the research gap in the use of reactive approach by Nigeria government, where the prevalence of money laundering has risen in recent years, and explored how ML techniques can be utilized to enhance the country's efforts in combating this financial crime.The datasets for this study were obtained from Kaggle website that contains fraudulent transactions on money laundering, which was used to train, validate and test the ML models. Logistic regression (LR) and Support Vector Machine (SVM) were used as the baseline models while Sparse Autoencoder (SAE) neural network was used for feature learning and dimensionality reduction. The results indicate that LR classifier still showed reasonable performance but did not outperform the other models. Among all the measures, SVM exhibited outstanding performance, with over 90% prediction accuracy. The amount of money transferred and location of transactions emerged as top features for predicting money laundering transactions in online money transfers. These findings suggest that further research is needed to enhance the logistic regression model, and sparse autoencoder neural network should be explored as potential tool for law enforcement agencies and Nigeria financial Institutions to proactively learn representative data from high dimensional datasets as quality of data improves performance of predictive models.
Keywords: *Machine Learning, Revolution, Money Laundering, Algorithms, Artificial Intelligence*

Introduction

Money laundering has become a pervasive and insidious crime that presents a significant threat to Nigeria's financial sector, with far-reaching consequences for economic stability, national security, and global financial integrity (Canhoto, 2021). The Financial Action Task Force (FATF) estimates that up to 5% of global GDP is laundered annually, with developing economies like Nigeria disproportionately affected (FATF, 2020).

In Nigeria, Government approach to the fight against money laundering largely hinges on the law enforcement and CBN regulations.This simply implies that Nigeria Government's efforts in combating money laundering have primarily focused on traditional methods such as law enforcement by Economic and Financial Crime Commission (EFCC) and regulatory actions by the Central Bank of Nigeria (CBN). This approach has been more of reactive than proactive, often involving lengthy litigation period. The commercial Banks which are under the obligation of CBN policies on money laundering to furnish EFCC with information needed for investigation and conviction of money laundering offenders rely on traditional anti-money laundering (AML) methods known as rule-based expert systems. However, due to the explosive growth of electronic transaction information generated through online payment channels (Jipeng et al., 2021), this traditional rule-based systems have proven inadequate in detecting and preventing these complex financial crimes (Shi et al., 2019).

To address this challenge, researchers have explored innovative solutions, including machine learning, data analytics and deep learning approaches (Wan et al., 2019; Abavisani & Patel, 2019; Zamini et al., 2019). Machine learning (ML), a subset of artificial intelligence, has demonstrated remarkable potentials in detecting and preventing financial crimes, including money laundering (Hanbing., 2021). For example, Honlam (2022) agreed that a number of ML methods have been developed to learn patterns in credit card frauds, and that these ML methods usually depend on sophisticated feature engineering to improve their performances.

By leveraging ML algorithms and data analytics, financial institutions can improve detection accuracy, reduce false positives, and enhance customer due diligence (Jorge et al., 2024). Research has equally highlighted the benefits of machine learning in AML, including improved detection of suspicious transactions (Canhoto et al., 2021), enhanced risk assessment (Shi et al., 2019), and reduced compliance cost (Anubha, et al., 2022)

In Nigeria, the need for effective AML solutions is particularly pressing. The country's financial system is characterized by a high volume of cash transactions, limited financial inclusion, and inadequate AML infrastructure (Omri, 2021). While the Nigeria government has implemented various AML measures, including the money laundering (prohibition) Act of 2011 and the Central Bank of Nigeria's AML/CFT policy, challenges still persist (CBN, 2020).

This study aims to contribute to the development of effective machine learning-based AML solutions in Nigeria, building on the existing literature and exploring innovative approaches rather than reactive one to enhance financial security and combat economic crime. The datasets for this study were obtained from Kaggle website that contains fraudulent transactions on money laundering, which was used to train, validate and test the ML models. Logistic regression (LR) and SVM were used as the baseline model and sparse autoencoder (SAE) neural network was deployed for representative learning. We combined supervised and unsupervised learning approaches.

2.1 Related Works

In recent time, autoencoder has found significant applications in various unsupervised learning tasks in several application areas such as heart disease prediction and fraud detection. In this regard, (Ibomoiye et al., 2020; Ali et al. 2021) proposed a method that combined Support Vector Machine (SVM) and Sparse Auto-encoder (SAE) for money laundering detection. The rational there was that the classical SVM has limitations on large scale applications; hence, the need to use a sparse auto-encoder to improve its performance. The authors employed multiple layers of sparse autoencoder to perform feature learning and used the SVM for classification, thereby improving the performance of the SVM in handling large scale datasets. The combined model resulted in 80% accuracy in anti-money laundering detection. In a related study (Moussavi & Jamshidi, 2019), proposed a method to perform feature learning using sparse auto-encoder to improve the performance of logistic regression model on real-valued time series data. The architecture consists of different layers of sparse autoencoder. The aim of the research was to enhance vehicular traffic flow forecasting. However, in an attempt to increase the accuracy of the sparse autoencoder, they proposed a cascaded model which leverages on the combination of low and high level features, and a stochastic gradient descent algorithm was employed as the regression method.

Also in Abavisani and Patel (2019), a sparse representation based classification method was proposed using a transductive deep learning based formulation. The network comprises of a fully connected layer and a convolutional autoencoder. The fully connected layer is placed between the encoder and decoder, and its function is to find the sparse representation, whereas the autoencoder network learns effective deep features for classification. When the estimated sparse codes are used for classification of some datasets, the proposed method showed improved performance.

In Wan et al. (2019), an approach was proposed to derive a formulation that effectively determines the sparse hyper-parameter in sparse auto-encoder, in addition to deriving the relationship between the average activation of hidden units and sparse hyper-parameter. The authors conducted two experiments and they obtained good performance. Another connected study on a novel method equally shows where a sparse autoencoder is used for automatic modulation classification (Ibomolye et al., 2020; Ali & Yangyu, 2019). The network was trained using a non-negativity constraint algorithm. Experimental results show that the autoencoder with the non-negativity constraint enhances the sparsely and minimizes the reconstruction error as compared to the traditional sparse autoencoder.

In a nutshell, the above related studies conclude that a combination of supervised and unsupervised approach to machine learning applications have the potentials of improving models performance accuracies. However, while only few studies have deployed sparse autoencoder to learn representative variables, non has experimented with the three models: LR, SVM and SAE (SAE-SVM-LR). Unsupervised ML method, compared to the supervised, has the advantage of detecting unseen relationships between variables and finding representative features, enabling it to replace feature engineering. Therefore, processing the raw data with unsupervised ML may generate useful and representative variables. Having representative variables is helpful for many supervised ML methods.

Previously, Principal Component Analysis (PCA), an unsupervised approach, was commonly used for feature engineering task. But, PCA has two drawbacks. First, it can only detect linear relationships between variables. Second, it can only generate low dimensional variable space,

leading to loss of some information and limiting the ability of the model to learn representative variables. Thus, it performs poorly when the relationship is non-linear, or the number of variables is of high dimensional. This is where Sparse Autoencoder comes in to address this knowledge gap and ensure quality data are used for model training and evaluation. Loss of information, over-fitting challenges, and false positives are greatly minimized, leading to an improved model performance through deployment of Adaptive Moment Estimator (Adam) Optimization algorithm.

2.2 Data Collection

Due to challenges in gathering internal or real World data from financial institutions or buying from vendors, we relied on money laundering datasets collected from a reliable open source, Kaggle website. The collected dateset contains money laundering transactions, which occurred between January and June 2023. In addition, the dateset is highly imbalanced with 568,630 transactions, containing 380,982 negative examples (non-fraudulent). The latter accounts for about 67% of all data, while the remaining 187,648 are classified as positive examples (frauds) accounting for 33% of the total datasets (568630).

For want of space, only the first 6 rows of the datasets are shown in table 1. We have a total of 32 transactions data and all variables are numerical. An extraction of 6 rows and 10 columns (features) of the money laundering datasets are presented in table 1 below.

Table 1 First 6 rows of the money laundering Dataset

id	V1	V2	V3	V4	V5	V6	V7	amount	class
0	-0.26064	-0.46964	2.49626	-0.083723	0.12968	0.73289	0.51901	17982.1	0
1	0.98509	-0.35604	0.55805	-0.429653	0.27714	0.428604	0.40646	6531.37	0
2	-0.26027	-0.94938	1.72853	-0.457986	0.07406	1.419481	0.74351	2513.54	0
3	-0.15215	-0.50895	1.74684	-1.090177	0.24948	1.143312	0.51826	5384.44	1
4	-0.20681	-0.16528	1.52705	-0.448292	0.10612	0.530548	0.65884	14278.9	1
5	0.025302	-0.14051	1.19113	-0.707978	0.43049	0.458973	0.61104	6901.49	0

2.2.1 Data preprocessing

This is the second and a crucial step in the machine learning method. It involves cleaning the data (removing duplicates, correcting errors), handling missing data (either by removing it or filling it in), and normalizing the data (scaling the data to a standard format). This approach improves the quality of collected data and ensures that proposed AML machine learning model can interpret it correctly. This procedure can significantly improve the accuracy of the proposed model and enhance the performance of the model for AML detection.

2.2.2 Data Balancing

There are many ways of handling imbalanced data which are distinguished into data and level algorithms. Among these techniques, the preferred one is the algorithm method which involves the use of a confusion matrix and balanced accuracy metrics to modify the dataset, re-balance the imbalanced data, and remove noise between two classes before using the data in the algorithms. In this regard, the datasets (table 1) were divided into 75% training/ 25 % testing and 70% training /30% testing for all the supervised Learning algorithms being considered in this study. The algorithms are Logistic regression (LR) and Support Vector Machine (SVM).

2.3 Methodology

Sparse auto-encoder (SAE), logistic regression and support vector machine were considered in this study. SAE is an unsupervised ML approach for automatic feature engineering tasks. While the auto-encoder is considered for feature extraction, Logistic regression and support vector machines were used to train, validate and make prediction based on the pre-processed datasets.

2.3.1 Sparse Autoencoder

The SAE consists of two parts: encoder and decoder (figure1). The encoder transforms input layer into encode layer, then the decoder transforms encode layer into output layer. The loss function of SAE is the reconstruction error between the input layer and the output layer (equation 4). By minimizing this loss function, SAE learns representative features in the encode layer. Because the dimension of the encode layer is smaller than that of the input layer, the learned representation should be the most important part of the input data. Otherwise, the output derived from the encode layer will differ from the input data.

In addition, Sparse Auto-encoder further applies sparsely constraints to prevent over-fitting (equation 6). That is, the loss function of SAE adds on a penalty term proportional to magnitude of the encode layer. By using activity regularization, SAE limits the number of active neurons in the encode layer, thus preventing the neural network from simply copying output from input (Honlam, 2022; Hanbing, 2021; Ibomoiye, 2020).

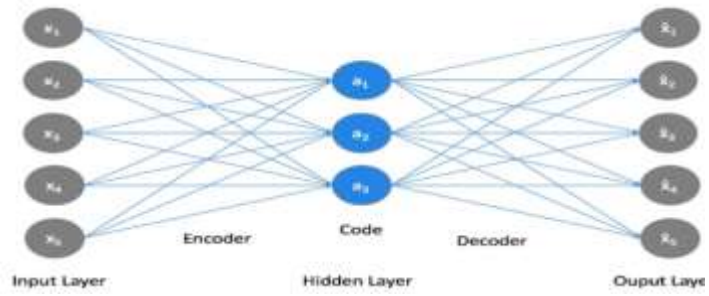

Figure 1 Sparse Autoencoder work flow (Jian et al., 2019)

The encoder maps the input to a new representation. This new representation is then decoded at the output to reconstruct the input $\hat{x}$ according to Equations (1) and (2), where x is the input and z the new representation.

$$\mathbf{Z = h\ (w_x + b)} \quad [1]$$

$$\mathbf{= g(\hat{w} z + \hat{b})} \quad [2]$$

In the above formulation, h is the activation function for the hidden layer neurons and g is for the output layer neurons, w and $\hat{w}$ are weight matrices, b and $\hat{b}$ are respectively the encoder and

decoder bias vectors. In this paper, the sigmoid activation function is utilized, which is shown in Equation (3) instead of the others such as ReLU, Tanh etc.

$$\mathbf{f(h)=} \quad \frac{1}{1+e^{-h}} = \frac{1}{1+e^{-(bo+b1x)}} \qquad \textbf{[3]}$$

As we move forwards through more layers, the level of abstraction increases. Let's now analyse the activation functions in little more detail. Previously our activation was just a simple function that outputs 0 or 1 from the relation Z= **wx + b**

Unfortunately, there is a pretty dramatic function since small changes such as (0.1, 0.2,-1.0 etc.) are not reflected, So, it would be nice if we have a more dynamic function which combines sigmoid and activation functions (Sigmoid function+ activation function). So we can integrate sigmoid function with activation function to handle not just 0 and 1 output but as well as small changes. We can replace h in equation 3 with z. So we have:

$$f(h) = f(x) = \frac{1}{1+e^{-h}} = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-(wx+b)}} \qquad \textbf{[4]}$$

Recall that z = wx + b from equation 1

The reconstruction loss (error) function E between the input x and reconstructed input uses the mean squared error (MSE) function shown in Equation (4).

$$E = \frac{1}{N}\sum_{i=1}^{N} xi + x'i^{2} \qquad \textbf{[5]}$$

Note: = x'

N represents the number of input samples (in this case 32). However, in this research it is important to express that a sparse auto-encoder is utilized to obtain an effective low-level representation of the input data under sparse constraints. Hence, sparsity is introduced by including regularization to the loss function. Let $_i$ be the average activation of neurons in the hidden layer.

$$_i = \frac{1}{n}\sum_{j=1}^{n} zi\ (\ xj\ ) \qquad \textbf{[6]}$$

From Equation (5) i, n, and j represents the ith neuron, total number of training samples, and jth training sample respectively. The average activation i approaches p that is a constant close to zero. Hence, the Kullback-Leibler (KL) divergence is used to add the regularizer to the loss function. The KL divergence is introduced to achieve sparsity.

$$\Omega_{\text{sparsity}} = \sum_{i=1}^{d} p\log(\frac{p}{p'}) + (1-P)\log(\frac{1-p}{1-p'}) \qquad \textbf{[7]}$$

Note: i = P'

From Equation (6) d represents the total number of neurons in a layer, whereas p is the sparsity proportion, which is the needed activation value. Therefore, the SAE error function now comprises of the mean square error and the regularization terms. Furthermore, in order to control the weights and prevent over-fitting, L2 regularization (L2R) is introduced in the loss function.

$$\Omega_{\text{weights}} = \frac{1}{2}\sum_{i}^{L}\cdot\sum_{j}^{N}\cdot\sum_{i}^{K}(wij\ )^{(I)} \qquad \textbf{[8]}$$

L and K represent the number of hidden layers and number of features in a sample, respectively (Mienye et al., 2020; Zia & Rehman, 2019). The weight attenuation units as seen in Equation (8)

is then included. After adding the various regularization terms, i.e. Equations (6) and (7) into Equation (4) which is the reconstruction error E, our loss function becomes:

$$E = \frac{1}{N}\sum_{n=1}^{N}\sum_{K=1}^{K}(x_{kn} - x'_{kn})^2 + \lambda * \Omega_{weights} + \beta * \Omega sp_{arsity} \quad \textbf{[9]}$$

There are three optimization parameters here: λ which is the coefficient for L2R and it prevents over-fitting, the second parameter is β, the sparsity regularization parameter, and it sets the sparsity penalty term. Lastly, p is the sparsity proportion which controls the needed sparsity level. The optimization parameter values for λ, β, and p are 0.0001, 0.01, and 0.5 respectively.

2.3.2 Adaptive Moment Estimator (Adam) Optimizer

Furthermore, in order to train a robust SAE, the Adam algorithm is used in place of the Adaptive Gradient (AdaGrad) algorithm, or Root-mean-square propagation algorithm proposed in 2012. The Adam optimization algorithm surfaced in 2014. The choice of this algorithm offers us the opportunity to use a different learning rates for various parameters and to come up with dynamic adjustment of various parameters by obtaining the gradient first-order moment estimate, mt, and second-order moment estimate, vt, shown in Equations (10)– (12).

Gradient Computation:

For every iteration t, Adam computes gradient gt. This gradient is the derivative of the objective function concerning the current parameter θt.

$$gt = \forall \theta ft(\theta t - 1) \quad \textbf{[10]}$$

Where:

gt represents the gradient at iteration t , $\forall$ denotes the gradient for parameter , and $ft(\theta t - 1)$ is the objective function being optimized, evaluated at the parameter values from the previous iteration t - 1.

With the gradient computed in equation (10), the next step will be to update the first-moment estimator (Mt), which stores the moving average of the gradient. This update simply combines the previous value of Mt and the new gradient, weighted by parameters 1 and 1- $\beta 1$ respectively.

$$Mt = \beta 1.M_t - 1 + (1 - \beta 1).g_t \quad \textbf{[11]}$$

Where:

- Mt is the first-moment vector at time step t
- $\beta 1$ is the exponential decay rate for the first-moment estimates (commonly set to be around 0.9)
- gt is the gradient at time step t

Similarly, the second-moment vector V_t is also updated. This vector gives an estimate of variance (or unpredictability) of the gradient, therefore it stores the squared gradients that are accumulated. Just like the first -moment (Mt), this is also a weighted combination, but of the past squared gradient and current gradient as shown in equation (12).

$$V_t = \beta 2.V_{t-1} + (1 - \beta 2).gt^2 \quad \textbf{[12]}$$

Where:

V_t is the second moment vector at time step t, $\beta 2$ is the exponential decay rate for the second-moment estimates (commonly set to around 0.999)

The need to calculate the correct bias in the moments is also essential, Being that mt and Vt are initialized to 0, they are biased towards 0, especially during the initial time steps. Adam deals with this bias by correcting the vector using decay rate, which is 1 for Mt and β_2 for Vt. This correction is vital as it ensures that the moving averages are more represented, particular in the early stage of training. Equations (13) and (14) show the mathematical expressions for computing correct bias.

$$M't = \frac{Mt}{1 - \beta_1{}^t} \quad \textbf{[13]}$$

$$V't = \frac{vt}{1 - \beta_2{}^t} \quad \textbf{[14]}$$

Finally, Adam updates model parameters using equation (15). This is the step where the actual optimization occurs, moving the parameters in the direction that minimizes the loss function. Parameter update makes use of adaptive learning rates computed in the previous equations

$$\Rightarrow \quad t + 1 = t - \frac{\alpha . M't}{\sqrt{V't} + \in} \quad \textbf{[15]}$$

Where :

- t + 1 represents parameters after the update
- t represents the current parameters before the update
- α is the learning rate, which is an important hyper-parameter that determines the size of step taken towards minimizing the loss function
- M't is the bias-corrected first moment estimate of the gradient
- V't is bias corrected second moment estimate of the gradient
- $\in$ (Epsilon) is a small scalar (e^{-8}) added to prevent division by zero and maintain numerical stability.

2.3.2 Logistic Regression

In this section, we will explore logistic regression model to train, test, evaluate and predict fraudulent behaviour from reconstructed pre-processed data using dimensionality reduction techniques of SAE. Logistic regression is similar to linear regression, but the difference is that it produces a curve while linear regression produces a straight line. Based on the usage of one or more predictors or independent variables, logistic regression generates logistic curves that depict the values between zero and one (Omri, 2021). There are many different forms of logistic regression models, including binary, multiple, and binomial logistic models (Omri, 2021). The binary logistic regression model is used to predict the likelihood of a binary response (0 or 1) based on one or more factors. The equation below represents the logistic regression in mathematical form.

$$p = \frac{e^{\alpha+\beta_n X}}{1 + e^{\alpha+\beta_n X}}$$

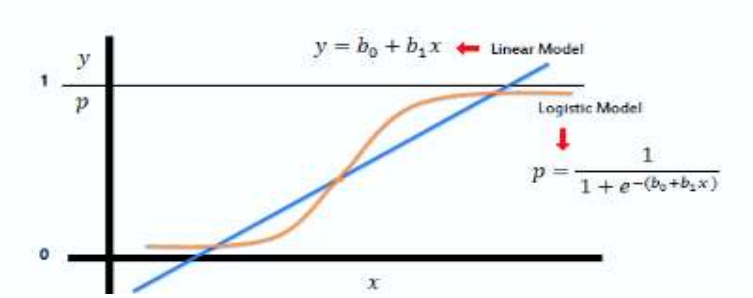

Figure 2: Source (Omri, 2021)

The contrast between linear regression and logistic regression is seen in this graph, where logistic regression depicts a curve and linear regression depicts a straight line (Omri, 2021; Wright, 1995).

2.3.3 Support Vector Machine

Support Vector Machine (SVM) is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. It is primarily used for classification problems in machine Learning. SVM maps the variables into a high dimensional space using a kernel function, which then finds a hyperplane to maximize the gap between support vectors and to minimize the error of miscalculation simultaneously (Pisner & Schnyer, 2020). Hyperplane is a space whose dimension is lower than the mapped space. Support vectors are points close to the Hyperplane. On one hand, SVM tries to maximize the margin between support vectors to increase the generality of the model. On the other hand, SVM minimizes the misclassification to prevent under-fitting. By choosing an appropriate C value, it finds a balance in this bias-variance trade-off. In this research, the SVM algorithm was implemented using the Sklearn package in python. The kernel was 'rbf' and the C value was 1. Consider the below diagram in which two different categories are classified using a decision boundary or hyperplane:

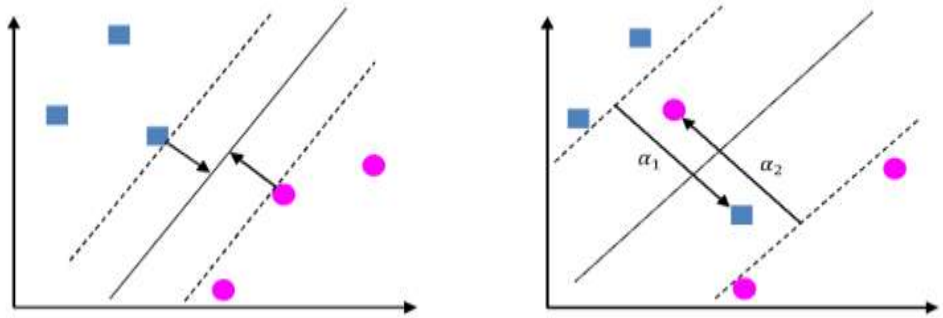

Figure 3: Support Vector Machine (Omri, 2021)

2.4 Implementation

The implementation was done by importing relevant python programming language libraries such as Scikit Learn (Sklearn), pandas, NumPy, Tensor flow, Matplotlib, seaborn, and Keras. This was

done using the Jupiter application programming Interface. The Scikit learn was used to train the Logistic regression and Support Vector Machine Models, splitting the given dataset into training and test datasets, usually in an imbalanced manner. After training the Models, they were able to classify or predict new or unseen datasets as represented in table 2 and 3.

2.5 Results and Discussions

The performance evaluation of the trained datasets for the prediction of unseen or new input is done with a confusion matrix to determine how well the models have performed. The choice of a confusion matrix is based on imbalanced datasets. The data-set was evaluated with the 2 algorithms to obtain the numbers of true positive (TP), True Negative (TN), False positive (FP), and False Negative (FN). True positive means that positive examples are correctly assigned to the positive class. In this datasets, it means fraudulent transactions. True negative (TN) refers to the negative examples correctly assigned to the negative class, meaning no fraudulent transactions occurred. False positive (FP) means that the algorithm incorrectly considers negative examples as positive examples. That is predicting non fraudulent transactions as fraudulent. In other words, when a sample transaction is non fraudulent, the algorithm mistakenly flags it as an abnormal transaction. False negative (FN) is a situation where positive examples are wrongly allocated to a negative class. It means the classified transaction is legitimate; however, the algorithm misunderstood this as an illegitimate transaction. The confusion matrix for the data-set is shown in the Tables below for different splits of training and testing datasets.

Table 2: Confusion Matrix For Fraud Detection (75% training,25%testing)

Logistics Regression			Support Vector Machine		
Predicted no	Predicted yes	**N = 142158**	Predicted no	Predicted yes	N = **142158**
TN 47127	**FN 23896**	**Actual no**	**TN 71020**	**FN 3**	**Actual no**
FP 5998	**TP 65137**	**Actual Yes**	**FP 109**	**TP 71026**	**Actual Yes**

Table 3: Confusion Matrix For Fraud Detection (70% training,30%testing)

Logistics Regression			Support Vector Machine		
Predicted no	Predicted yes	N = 170589	Predicted no	Predicted yes	N = 170589
TN 85060	FN 89	Actual no	TN 85124	FN 25	Actual no
FP 166	TP 85274		FP 34	TP 85406	Actual Yes

		Actual Yes			

The first model could have been judged by its overall accuracy, which works well for most datasets splits. However, the accuracy might be insufficient to reflect the performance of a model in the imbalanced dataset. So, balanced accuracy is used in this case to determine whether an algorithm has performed well or not. Mathematically, it is represented as:

Balanced Accuracy $= \frac{Sensitivity+Specificity}{2}$

Balanced Accuracy $= \frac{1}{2}\left(\frac{Correct\ positive\ predictions}{Number\ of\ positives}\right) + \left(\frac{Correct\ Negative\ Predictions}{Number\ of\ Negatives}\right)$

$\rightarrow$ Balanced Accuracy $= \frac{1}{2}\left(\frac{TP}{TP+FP} + \frac{TN}{TN+FN}\right)$

47127: TN	**23896** : FN
5998: FP	**65137**: TP

In the above formula, TP is true positive, and FP is false positive. TN is True Negative, and FN is False Negative. The higher the balanced accuracy is, the more the classification is put into the right place. The balanced accuracy analysis in terms of the predictive probabilities of each model is shown in Table 4.

Here, we found that the proposed AML model (SAE-SV M-LR) with 0.89 predictive probability or 89% balanced accuracy outperformed SAE-SV M (Honlam, 2022) which has 80% accuracy when training and testing datasets is split to 75% and 25% respectively. Similarly, a better performance prediction holds with a dataset split of 70%/ 30%, which shows an improvement prediction of the proposed model to 99%.

Table 4: comparison SAE-LR-SVM and SAE-SVM

Dataset splits	**LR**	**SVM**	SAE-SVM-LR (Avg of LR and SVM)	**SAE-SVM**
75%/25%	0.78	0.99	0.89	0.80
70%/30%	0.99	0.99	0.99	0.89

Conclusion and Recommendation

ML is effective in the area of money laundering Detection (MLD). However, its sterling performance relies heavily on sophisticated feature engineering, which is expensive to scale. In previous studies, few attempts have been made to combine unsupervised ML with supervised ML. In this work, the proposed model combined SAE, LR and SVM. In the first step, the model used SAE to perform the pr-processing task of data reconstruction. As a result, SAE extracted representative features in it's encode layer. Then the model passed representative features to the LR and SVM models. This work used the MLD datasets from Kaggle as input to the model and training and model evaluation carried out using the proposed model. The result showed that using SAE to extract representative features improved the performance of SVM. This SAE-LR-SV M method achieves 89% and 99% balanced accuracy on two different data-set splits for training and testing phases, compared to 80% accuracy obtained from SAE-SV M method proposed by (Honlam, 2022). In addition, the SAE-LR-SV M model outperforms other auto-encoder-based

models regarding the balanced accuracy (Zamini et al., 2019). This is a remarkable performance because no model has been able to have 100% prediction due to several factors such information loss, imbalanced dataset, overfitting, quality and quantity of dataset to mention but few. Overall, Machine Learning algorithms can be successfully used for financial fraud transaction detection. In the future, this study will further fine-tune the SAE model and try different classification methods besides SVM and LR. Also, this research will be conducted on other MLD datasets to further investigate how well SAE can encode complicated transactions.

Recommendations

To reduce the over-fitting issue in the training Logistic Regression, Lasso, and Ridge regularization can be applied to the datasets to improve the performance. In addition, cross validation concept and optimization algorithms with different learning are also the way forward in solving the over-fitting and minimizing information loss respectively. We equally recommend availability and use of real World datasets from financial institutions for model training, testing and validation.

References

Abavisani, M. and Patel, V. M. (2019). Deep Sparse representation-based classification. *IEEE Signal Process letter* 26(6): 48–52. https://doi.org/10.1109/ LSP.2019.2913022

Albanian (2019). Assessing the Introduction of Anti Money Laundering Regulation on Bank Stock Valuation: An Empirical Analysis. *Journal of Money Laundering Control.* I76-88. https://doi.org/10.1108/JMLC.2018.00

Ali, A. and Yangyu,F.(2019) Automatic Modulation Classification using Deep Learning Based on Sparse Autoencoder with non-negative constraints. *IEEE Signal Process Lett:* 1625-30. https://doi.org/10.1109/lsp.2017.2752459

Anubha, P., Alekhya, B., Shiv, M. and Deepak, B. (2022). Adversarial Fraud Generation for Improved Detection . *3rd ACM International Conference on AI in France.* New York, USA: https://doi.org/3533271..3561723

Archana, P. and Mangu, N.(2019). Credit Card Fraud Detection Using XGBoost for Imbalanced dataset. *Fifteeth International Conference on Contemporary Computing, August 03-05, Noida, India, ACM New York, USA.* https://doi/org/10.1145/3607947

Canhoto, A.I.(2021). Leveraging Machine Learning in the Global Fight Against Money Laundering and Terrorism Financing: An affordances Perspective. *Journal of Business Research.* 24(3): 14 -28

FATF(2020). History of Financial Action Task Force: https//www.fatf.gafi.or/about/history of fatf/#d.en.3157. Accessed 26th July , 2024

Hanbing, Z.(2021). Analysis of Best Sampling Strategy in Credit Card Fraud Detection Using Machine Learning. *6th International Conference on Intelligent Information Technology, February 25-28, Hochi Minh, Vietnam. ACM, New York, USA.* https://doi.org/10.1145/3460179.3460188

Honlam, L. (2022). Credit Card Fraud Detection Based on Combination of Sparse Autoencoder and Support Vector Machine. *6th International Conference on Electronic*

Information Technology and Computer Engineering (EITCE), October 21-23, Xiamen China. ACM, New York, NY, USA .https//doi.org/10.1145/35734

Ibomoiye, D. M., Yanxia, S. and Zhenghui, W. (2020). Improved sparse autoencoder Based Artificial Neural Network Approach for Prediction of Heart Disease. *Journal of Information in Medicine Unlocked.* 5(2): 80-89 https:/doi.org/10.1016/j.imu

Jian, S., Yin, L., Charley, C., Jihae, L., Xin, L. and Zhongping, Z. (2020). FDHelper: Assist Unsupervised fraud detection experts with interactive feature selection and evaluation. *Proceedings of China conference on human factors in computing system, April 22 -24* , pp 1-12 . https:/doi.org/10.1145/331831/3376140

Jipeng, C., Chungang, Y. and Cheng, W. (2021). Learning Transaction Cohesiveness for online Payment Fraud Detection. *ACM Conference on computing and Data Science (CONFCDS) January 28 – 30th, Stanford, CA, USA, New York, NY, USA*. https:// doi.org/10.1145/3448734.3450489

Jorge, F.M.P.,Jorge,G.., Dandi, B.L., Jorge, A.R.M., and Moises, M.R.A. (2024). Fraud Transaction Detection for Anti-Money Laundering Systems Based on Deep Learning. *Journal of Emerging Computer Techniques*. 12(31): 29-34. Doi: 10.57020/Ject.1428.46

Moussavi,K.A. and Jamshidi, M.(2019). Conducting a Deep Regression Model Utilizing Cascaded Sparse Autoencoder and Stochastic Gradient Descent. 15th IEEE *International Conference on Machine Learning and Applications.* https://Doi.org/10.1109/ICMLA.2016.0096

Omri, R.(2020). Applying Supervised Machine Learning Algorithms for Fraud Detection in Anti Money Laundering. *Journal of Modern Issues in Business Research*.1(1), 441-452

Shi, Y, Lei, J., Yin, Y., Cao, K., Li, Y. and Chang, C.I. (2019). Discriminative Feature Learning with distance Constrained Stacked Detection . GeoSci Rem Sens Lett IEEE. 16(9).14-26. httpts://doi.org/10.1109/LGRS.2019.2901019

Wan, Z., He, H. and Tang, B.(2019). A generative Model for Sparse hyper-parameter determination.*IEEE Transactions on Big Data.* 4(1): 2-10. https://DOI.org/10.1109/TBDATA 2017.2689790

Zamini, M. and Gholamali, M (2019). Credit card fraud detection using autoencoder based clustering. *9th International Symposium on Telecommunication (IST)*, IEEE