

***University of Ibadan Journal of
Science and Logics in ICT
Research (UIJSLICTR)***
ISSN: 2714-3627

A Journal of the Faculty of Computing, University of Ibadan, Ibadan, Nigeria

Volume 17 No. 1, June 2026

journals.ui.edu.ng/uijslictr
http://uijslictr.org.ng/
uijslictr@gmail.com



Development of Fraud Detection Model Using Support Vector Machine and Constraint Solver

¹Ayinde, T. T. and ²Ayorinde, I. T.

^{1&2}Department of Software Engineering, Faculty of Computing,
University of Ibadan, Ibadan, Nigeria.
Emails: ¹toluayinde200@gmail.com ; ²temiayorinde@yahoo.com

Abstract

The expansion of digital payments has led to a rise in sophisticated fraud, creating major financial and reputational risks for institutions. Although Support Vector Machines (SVMs) have demonstrated strong predictive performance in fraud detection, their decision-making processes often lack transparency and may not fully align with domain knowledge or regulatory requirements. In contrast, constraint solvers are inherently explainable and exhibit strong compliance with predefined rules, but they lack adaptability to complex, data-driven patterns. This paper addresses this limitation by developing a hybrid fraud detection model that integrates Support Vector Machines with Constraint Solvers so as to enhance fraud detection performance. SVM model was used to identify patterns distinguishing fraudulent from legitimate transactions based on historical data. Simultaneously, a constraint solver model was designed using rules derived from transaction characteristics, principal component analysis (PCA) value ranges, and anomaly detection results from an Isolation Forest. The Kaggle Credit Card Fraud Detection dataset was used for experimentation. Hybridization was achieved by combining the outputs of the SVM and constraint solver using logical OR and AND operators, resulting in two hybrid models. The implementation was done using Google Colab. Model performance was evaluated using precision, recall, F1-score, and confusion matrix metrics. The experimental results indicate that SVM model achieved strong performance, with a precision of 60.29%, recall of 83.67%, and an F1-score of 70%. The constraint-only model recorded a precision of 4.26%, recall of 63.27%, and an F1-score of 7.98%. The Hybrid (ML OR Constraints) model achieved a precision of 5.52%, recall of 83.67%, and an F1-score of 10.36%, while the Hybrid (ML AND Constraints) model attained a precision of 57.94%, recall of 63.27%, and an F1-score of 60.49%. In summary, according to the result, SVM model performed better than hybrid model in fraud detection.

Keywords: Fraud detection, Constraint solver, Support Vector Machine, Artificial Intelligence, Machine learning

1. Introduction

Artificial Intelligence (AI) is an interdisciplinary field of computer science concerned with the development of systems capable of performing tasks that typically require human intelligence, such as reasoning, learning, perception, and decision-making [33]. An essential characteristic of an artificially intelligent system is the presence of a representational or linguistic component through which knowledge can be expressed, manipulated, and used to guide intelligent behaviour [1].

Ayinde, T. T. and Ayorinde, I. T. (2026) Development of Fraud Detection Model Using Support Vector Machine and Constraint Solver, *University of Ibadan Journal of Science and Logics in ICT Research (UIJSLICTR)*, Vol. 17 No. 1, pp. 238 – 248.

Early developments in AI were largely dominated by symbolic and rule-based approaches, where expert knowledge was encoded using logical rules and heuristics to solve well-defined problems [26] [24]. While these methods demonstrated success in structured and controlled environments, they exhibited significant limitations in scalability, adaptability, and robustness when applied to real-world problems characterized by uncertainty, complexity, and dynamic change [9].

Advances in computational power, the availability of large-scale datasets, and improvements in algorithmic efficiency have led to a paradigm shift from knowledge-driven systems to data-driven approaches. This transition has enabled intelligent automation and adaptive

decision-making through learning from data rather than relying solely on explicitly encoded rules [22][23]. As a result, AI technologies have become increasingly pervasive, with widespread applications across domains such as finance, healthcare, transportation, and cybersecurity, where they support predictive analytics, risk assessment, and real-time decision-making [19]. These developments position AI as a foundational technology for addressing large-scale, dynamic, and uncertain problems, including fraud detection in modern financial systems.

Machine Learning (ML), a core subfield of AI, focuses on the design of algorithms that can automatically learn patterns and relationships from data and improve performance through experience without being explicitly programmed [25]. ML enables systems to extract value from data by making predictions or detecting anomalies, thereby enhancing decision-making processes across various application domains [28]. ML techniques are generally categorized into supervised, unsupervised, and reinforcement learning, with supervised learning being the most widely used approach for classification and anomaly detection tasks [20]. Algorithms such as Support Vector Machines (SVM), Logistic Regression, Decision Trees, Random Forests, and Neural Networks have demonstrated strong performance in capturing complex and non-linear patterns within high-dimensional datasets [8][11][34].

Within the financial sector, ML has become a dominant approach for fraud detection due to its ability to analyze large volumes of transactional data and identify subtle deviations from normal behavior [7][13]. Despite these advantages, ML-based fraud detection systems face several challenges. High-performing models are often opaque, functioning as “black boxes” that provide limited interpretability, thereby complicating regulatory compliance and stakeholder trust [15][30]. Additionally,

issues such as extreme class imbalance, concept drift, noisy data, and adversarial behavior can lead to elevated false-positive rates and reduced generalization to emerging fraud patterns [4][17]. These limitations highlight the need for complementary approaches that enhance transparency, control, and robustness.

Constraint solvers represent an alternative computational paradigm grounded in symbolic reasoning, formal logic, and explicit knowledge representation. Problems are modeled using variables, domains, and constraints that encode domain expertise, operational policies, or regulatory requirements [14][32]. Through techniques such as constraint propagation, backtracking search, and domain reduction, constraint solvers systematically explore the solution space to identify feasible and consistent solutions [2][35]. A key strength of constraint-based systems lies in their transparency and explainability, as decisions can be directly traced to specific rules or constraints (Bessiere, 2006). This makes them particularly suitable for domains such as finance and security, where accountability, compliance, and human-readable reasoning are essential (Fages and Lapègue, 2014). However, constraint solvers face notable challenges, including limited scalability when applied to large, high-dimensional, noisy, or dynamically evolving datasets [14] [21], which restricts their effectiveness in data-intensive tasks like fraud detection.

Fraud detection represents a critical application area where the complementary strengths and weaknesses of ML and constraint solvers become especially evident. The rapid expansion of electronic payments, online banking, and digital financial services has led to increasingly sophisticated and adaptive fraudulent activities, resulting in substantial financial losses and reputational damage for institutions worldwide [6][37]. Traditional rule-based fraud detection systems lack flexibility and require frequent manual

updates to remain effective against evolving fraud strategies [7]. While ML-based systems address some of these limitations through adaptability and learning from historical data, they often suffer from high false-positive rates and limited interpretability, undermining trust and regulatory acceptance [10][13].

To overcome these challenges, recent research has increasingly explored hybrid fraud detection approaches that integrate machine learning techniques with constraint solvers or rule-based reasoning frameworks [36]. Such hybrid models aim to combine the predictive power and adaptability of ML with the interpretability, consistency, and domain control provided by constraint-based systems. By leveraging the strengths of both paradigms, hybrid approaches have the potential to improve detection accuracy, reduce false alarms, and enhance explainability.

Building on this body of literature, this study investigates the enhancement of machine learning-based fraud detection through the integration of constraint solvers. The overarching objective is to develop a robust, interpretable, and efficient hybrid fraud detection framework suitable for deployment in real-world financial environments.

2. Literature Review

Machine learning (ML) and artificial intelligence (AI) have become central to modern fraud detection by enabling predictive modelling and anomaly detection over large and complex datasets [31]. ML systems can automatically learn patterns from data, detect deviations from normal behaviour, and support automated decision-making in domains such as finance, cybersecurity, and healthcare. Supervised, unsupervised, and reinforcement learning represent the main categories of ML, each addressing different problem settings. Supervised learning, through algorithms such as Support Vector Machines (SVM), logistic regression, and decision trees, is

widely used for classification and fraud detection due to its strong predictive capabilities. However, many ML models operate as black boxes, limiting interpretability, regulatory compliance, and robustness in the presence of class imbalance, noise, and evolving fraud patterns [15][30].

Constraint solvers provide a complementary symbolic reasoning paradigm based on formal representations of variables, domains, and constraints. Through techniques such as backtracking, constraint propagation, and domain reduction, constraint solvers offer transparent, explainable, and rule-compliant decision-making, making them particularly suitable for safety-critical and regulated domains [14]. Despite their strengths, constraint-based systems struggle with scalability, noisy data, and dynamic environments, reducing their effectiveness in large-scale, data-intensive tasks when used in isolation.

Fraud detection highlights the limitations of both traditional rule-based systems and purely data-driven ML approaches. Rule-based systems lack adaptability, while ML-based systems often suffer from high false-positive rates and poor interpretability. Consequently, recent research has increasingly focused on hybrid approaches that integrate machine learning with constraint solvers or rule-based reasoning. Empirical studies demonstrate that such hybrid models can improve detection accuracy, reduce false alarms, enhance explainability, and better satisfy regulatory and domain constraints.

Ayorinde and Idyorough [3] introduced a model of deception detection on the social media through the use of pattern matching, whereby Twitter discussions were presented as a case study. The research paper presented the increasing challenge of deceptive behaviours in online interactions through a combination of conversational trust algorithms and proportionality constraints along with pattern matching

methods comprised of brute-force. Communication frequency, balance, and duration were used to create trust between users whereas the frequency of individual linguistic cues including first-person pronouns, exclusive words, negative-emotion words, motion verbs was used to detect a deceptive intent. The model had the dependency on the word matching based on the rules and dictionary and not on the classifier based on learning and was written in PHP. The experimental findings indicated that the given model could be handy to determine a deceptive communication relying on the presence of deceptive word patterns, which means that constraint-guided pattern matching may be effective in fraud and deception detection in social networks, though scalability and ability to adapt to changing patterns of language were not thoroughly examined.

Gangalakshmi *et. al.* [18] offered a better fraud detection model based on hybrid machine learning methods by considering the shortcomings of traditional algorithms in the imbalanced and high-dimensional financial data. To reduce the instances of false positives and improve the detection of the anomalous patterns of transactions, the authors suggested a hybrid ensemble model, which combines logistic regression, random forest and gradient boosting classifiers.

Data preprocessing activities like normalization, feature selection and class balancing using Synthetic Minority Oversampling Technique (SMOTE) were applied in the proposed infrastructure to enhance model stability and generalization. The hybrid model was able to detect with 98.7 percent accuracy using real-world transaction datasets with a huge cut to the computational time of standalone ones. The study showed that hybrid ensemble approaches could offer a scalable and more efficient framework of real-time fraud detection in electronic payment systems and enhance financial cybersecurity by generating data-driven intelligence.

Raghavan and El Gayar [29] have demonstrated a comparative analysis of machine learning and deep learning methods of detecting fraud based on the dynamism and patternless nature of fraud. Within the framework, benchmarking of several algorithms, including k-nearest neighbor (KNN), support vector machine (SVM), random forest (RF), and ensemble models was performed with deep learning algorithms including autoencoders, convolutional neural networks (CNN), restricted Boltzmann machines (RBM), and deep belief networks (DBN) to test their efficiency in credit card fraud detection. Based on European, Australian and German financial transaction datasets, the research used evaluation metrics like Area Under the ROC Curve (AUC), Mathew Correlation Coefficient (MCC), and Cost of Failure to analyze the performance of the model.

The experimental findings showed that SVM, KNN, and CNN tended to give better accuracy and robustness, whereas the combination of these algorithms through the use of ensemble models gave even better detection performance especially when using smaller data sets. The work presented a hybrid learning-based environment that was able to conform to the changing patterns of fraud thus contributing to the advancement of smart fraud detection systems in financial institutions.

3. Methodology

This study presents Hybrid Fraud Detection Model that combines a Support Vector Machine (SVM) model with a Constraint Solver (CS) model to improve fraud detection performance. The SVM model and the Constraint Solver (CS) model are independent and autonomous units at the first level of the architecture. The training of the SVM model is established on past transaction data which helps it to learn the patterns that are used to separate fraudulent transactions and legitimate transactions. According to the acquired decision functionality, the SVM model makes a predictive output, which is normally a

classification of the transactions as either being fraud or non-fraud with a particular degree of confidence. Simultaneously, the Constraint Solver component used the domain knowledge and predefined rules of frauds in the form of formal constraints. The CS model compares every transaction with the existing constraints and comes up with a rational decision that shows whether the transaction meets or breaks the stipulated requirements. The results of the ML model and the CS model are sent to a logical connection layer which is the main integration process of the hybrid architecture. The last output of the logical connection layer is the input of the Hybridized Model that is the final product of the new system. The hybrid model yields a final fraud decision which is based on both historical data learning and constraint-based reasoning which are based on expert knowledge.

The stages of the implementation involved

- i. data collection and data processing.
- ii. development of SVM model.
- iii. constraint solver model.
- iv. hybridization SVM model and constraint solver.

A dataset used for this research was Kaggle Credit Card Fraud Detection Dataset. The dataset consisted of 284,807 credit card transactions. Out of these 492 transactions were found to have been fraudulent and this was mere 0.172 percent of the total. The dataset has transformed 28 anonymised items referred to as V1 to V28 and two non-transformed features, which are time and amount. The dependent variable, Class was given a binary value of 0

when the transaction was legitimate and 1 when the transaction was fraudulent.

Before training the model, data preparation process was done to guarantee integrity and comparability. The preprocessing of data was done using Python, which was implemented on the Google Colab platform, and applied to the credit card fraud detection dataset. The dataset is a combination of numerical variables based on the Principal Component Analysis (PCA) transformation, time and amounts.

Scaling was introduced to the Time and Amount attributes only as the derived components of PCA were already standardised by definition, and the aim is to represent all features using the same numerical values. This was done using the StandardScaler of scikit-learn preprocessing module. This transformation adjusted every feature to give the features a zero mean and unit variance, which reduced the magnitude differences that are detrimental to the model convergence.

The scaling of each raw feature x_i was done based on the transformation:

$$x'_{xi} = \frac{x_i - \mu_i}{\sigma_i}$$

where μ_i and σ_i represent the mean and standard deviation of feature x_i , respectively.

The result of this was the development of two new standardised features Amount-scaled and Time-scaled both included in the major feature matrix. This standardization did not only enable numerical stability in the optimisation, but also enabled connection with the constraint-based solvers which frequently depend on features of similar scales so as to guarantee effectiveness and accuracy in satisfaction of constraints in the propagation process.

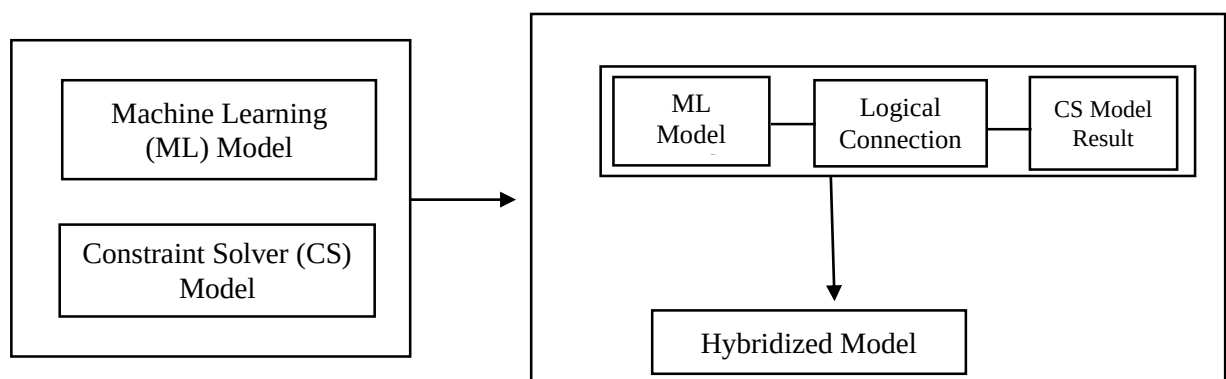


Figure 1: The flow diagram of the hybrid model

To maintain representativeness while preserving proportional class distributions, a stratified data partitioning strategy was adopted during the train-test split. Using the train-test-split function, the dataset was divided into training and testing subsets in an 80:20 ratio, with the stratify parameter ensuring that both subsets reflect the same class imbalance proportions. This approach preserved the statistical integrity of the data and prevented overfitting to the dominant class.

The final dataset after preprocessing was made up of:

Features (X): There are 30 variables (V1-V28), Amount_scaled, Time_scaled.

Target (y): This is a binary variable (Class) that indicates the presence of fraud. Class 0 represented legitimate transactions while class 1 represented fraud.

The model development, parameter tuning, and hybridization with constraint solvers were then performed on the training dataset and performance validation was done using the testing dataset, which was an unbiased set of evaluations. This phase is divided in two stages which are the training and testing.

The process of training took the following sequence:

- i. Data Partitioning: The dataset was divided into training and testing sets to prevent data leakage, allow generalisation evaluation and to ensure unbiased evaluation. The ratio of partitioning the dataset was 80% for the training and 20% for testing.
- ii. Undersampling: RandomUnderSampler was used to select a balanced sample of classes. The training set was the only set that was randomly undersampled to avoid information leakage.
- iii. Model Initialisation: It is a LinearSVC object that is initialised with constant random seed, higher iteration number, and a weight distribution that balances the classes. LinearSVC classifier was created, having a maximum number of iteration of 10,000 and class balancing switched-on.
- iv. Probability Calibration: LinearSVC is components of CalibratedClassifierCV wrapped with a threefold cross-validation.
- v. Fitting - The undersampled data set was trained with the calibrated model.

- vi. Selection for Inference – the calibrated model is selected as the major classifier to make predictions. The trained model was then applied to the resampled data in order to give calibrated probability predictions.

After carrying out the training and calibration processes, the prediction was performed on the test subset. This was done for the purpose of

- i. producing class predictions and probabilistic scores to evaluate model performance
- ii. determining how well the model generalises to data outside of the data used for training the model.

After training the calibrated Support Vector Machine classifier, the resulting model instance was used to generate both probabilistic model outputs and discrete class model predictions. The predict() method of hard classification and predict_proba method of generating probability scores were used in the model.

The procedure followed thus:

- i. Input Alignment: It was confirmed that the test set had the same feature space, structure and ordering as the resampled training data.
- ii. Probability-Based Scoring As the model was already wrapped with CalibratedClassifierCV, it produced class-wise probability distribution. This allowed more subtle interpretation of predictions, making it easier to rank, threshold analyze, and flag based on constraints.
- iii. Predictive Output Generation The model produced class assignments ($y^{\wedge}$) and probabilistic scores ($P(y=1|x)$) for each instance in the test subset. These outputs were retained for subsequent performance evaluation.

The second phase involved developing a Constraint Solver (CS) model using Google OR-Tools. This model introduced domain-specific logical and mathematical constraints that reflected realistic operational or business rules. The CS model ensured that only transactions satisfying all logical and contextual constraints were retained as valid. The constraints were defined as:

- i. Constraint A: Magnitude-Based Thresholding (Transaction Amount). The first limitation dealt with the risk of fraud on the presence of an unusual large transaction value. The threshold itself was empirically

obtained based on the data distribution due to a lack of knowledge on individual spending and the absence of user-level contextualisation. An adaptive upper bound was defined by the 99th percentile of amounts of transactions in the training set. Observations in any test-set that are above this mark were marked in Constraint A. Such quantile-based threshold is data based and thus makes sure the rule is robust over various distributions and is not based on arbitrary cut-offs.

- ii. Constraint B: PCA Components extremes. The second constraint detected abnormal patterns of the latent feature space. The data set has principal elements V1-V28; these elements represent combinations of original attributes of transactions. Any of the components, which show an absolute value above a specified limit, is flagged as transactions with extreme deviations. A $|V_i|$ threshold was taken as $|V_i| > 4$. Although the threshold value is adjustable. The purpose was to identify transactions that were well outside the normal range of variation.
- iii. Constraint C: Unsupervised Detection via Isolation Forest. The third limitation was based on an unmonitored anomaly detection system. An Isolation Forest model was trained using the training data only to be trained on the normal structure of the feature space without fraud labels. Samples in the test set that the model identifies as an outlier (i.e. assigned a prediction of -1) were flagged under Constraint C. This method extends the scope of detection to unusual configurations which are not detected by the classifier or hard coded limits. It is flexible in determining behavioural patterns that are not statistically common but are suspicious.

All the constraints were expressed as (Boolean) conditions and checked on each of the test set instances. The logic used in combining the constraints is given constraint A OR constraint B OR constraint C and the output of this process is called CS model. This is expressed as CS Model = Constraint A OR Constraint B OR Constraint C.

This research used a hybrid approach for fraud detection by combining the machine learning predictions with constraint-based reasoning. The hybridization process was done by combining

the result of SVM mode and Constraint solver mode using logical operator. The logical operators are logical OR and logical AND. The hybridization produced either of these results:

Hybrid(ML OR CS) = SVM Model OR Constraint Solver Model.

Hybrid(ML AND CS) = SVM Model AND Constraint Solve Model

The outcome was a parallel system where predictive and constraint-based signals are used in detection of suspicious behaviour.

4. Implementation and Discussion of Results

Python programming language was chosen because of its extensive libraries and ease of implementation.

The following Python libraries were used.

- i. imbalarn (imbalanced-learn): this offered the means to deal with imbalanced data using the resampling algorithms such as SMOTE and undersampling.
- ii. scikit-learn (sklearn): it provided a common platform to create, train and test machine learning models.
- iii. NumPy: Numpy was used to perform fast and efficient numerical calculations in arrays of many dimensions and mathematical functions.
- iv. pandas: This is a package that offered potent data manipulation and analysis methods with DataFrame and Series data frameworks.
- v. OR- Tools: This effectively solves optimization problems based on constraints including routing, scheduling and the allocation of the resources.

The implementation was done using google colab (<https://colab.research.google.com/>) whose configuration is Python 3 Google Compute Engine backend, System RAM: 12.7 GB and Disk: 107.72 GB

The results of implementing the fraud detection model using Support Vector Machine(SVM) and constraint solver are the presented here. The evaluation of the performance of the SVM, constraint solver, and SVM and/ constraint solver for fraud detection was done using some metrics such as precision, recall and F1-score.

The CreditCard dataset that was used for this work was got from Kaggle. The dataset was downloaded from

<https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>. It comprised of 284,807 transactions, containing 284315 non-fraud case and 492 fraud cases. Normalisation process was carried out in order to prevent bias when training the model. The amounts and the time of transaction were normalized.

The SVM classifier was trained on the preprocessed dataset. Using an 80:20 train-test split. Also, in order to apply constraint solver, constraints were set based on the available features in the dataset. The constraints were set on the amount, extreme Principal Component Analysis(PCA) values and isolation forest anomaly detection which are the outliers.

The performance metric of the model was assessed using key metric: Precision, Recall and F1-Score. Figure 2 and Table 1 show the snapshot of metrics of each model and the performance of the models respectively.

The results shown in Figure 3 indicated that the SVM model achieves the strongest overall performance, with high recall and a substantially higher F1-score, demonstrating its effectiveness in identifying fraud cases while maintaining a reasonable balance between precision and recall.

The constraints-only model performed poorly, exhibiting extremely low precision and F1-score, which meant that rule-based constraints alone generate a high number of false positives and lack sufficient discriminatory power. The hybrid model using ML OR Constraint Solver improved recall but suffered from very low precision, indicating that relaxing decision conditions increases fraud detection at the cost of excessive false alarms. In contrast, the hybrid ML AND constraint solver model provided a more balanced trade-off, significantly improving precision while maintaining moderate recall, resulting in a comparatively strong F1-score.

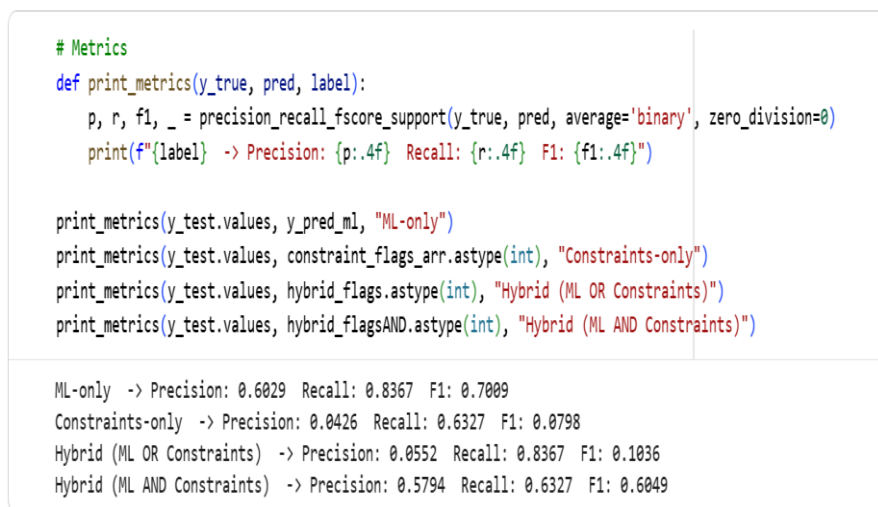


Figure 2: A snapshot of the metrics of each model

Table 1: Performance of the models.

Model	Precision	Recall	F1-Score
SVM(ML-Only)	60.29%	83.67%	70%
Constraint Solver-Only	4.26%	63.27%	7.98%
Hybrid(ML OR Constraint solver)	5.52%	83.67%	10.36%
Hybrid(ML AND Constraint Solver)	57.94%	63.27%	60.49%

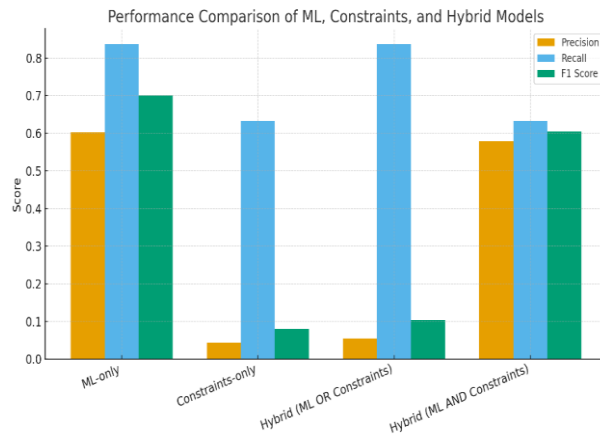


Figure 3: Performance Comparison of ML, Constraints and Hybrid Models.

The Confusion Matrix further illustrated the model's classification performance as shown figure 4.

```
# Show confusion matrices
print("Confusion matrix (ML-only):")
print(confusion_matrix(y_test.values, y_pred_ml))
print("Confusion matrix (Constraints):")
print(confusion_matrix(y_test.values, constraint_flags_arr.astype(int)))
print("Confusion matrix (Hybrid(ML OR Constraints):")
print(confusion_matrix(y_test.values, hybrid_flags.astype(int)))
print("Confusion matrix (Hybrid(ML AND Constraints):")
print(confusion_matrix(y_test.values, hybrid_flagsAND.astype(int)))

Confusion matrix (ML-only):
[[56810  54]
 [ 16  82]]
Confusion matrix (Constraints):
[[55470 1394]
 [ 36  62]]
Confusion matrix (Hybrid(ML OR Constraints):
[[55461 1403]
 [ 16  82]]
Confusion matrix (Hybrid(ML AND Constraints)):
[[56819  45]
 [ 36  62]]
```

Figure 4: A snapshot of the confusion matrix.

1. Key notes from Confusion Matrix (ML-only)

True Negatives (TN): 56,810 — transactions correctly identified non-fraud.
 False Positives (FP): 54 — non-fraud cases wrongly classified as fraud.
 False Negatives (FN): 16 — fraud cases missed by the model.
 True Positives (TP): 82 — correctly identified fraud transactions

2. Key notes from Constraints-based Confusion Matrix

TN: 55,470 non-fraud transactions correctly identified.
 FP: 1,394 non-fraud cases wrongly classified as fraud.
 FN: 36 fraud cases missed by the model.
 TP: 62 fraud cases correctly identified.

3. Key notes from Hybrid(ML OR Constraint Solver) Confusion Matrix

TN: 55,461 non-fraud transactions correctly identified.
 FP: 1,403 non-fraud cases wrongly classified as fraud.
 FN: 16 fraud cases missed by the model.
 TP: 82 fraud correctly identified.

4. Key notes from Hybrid(ML AND Constraint Solver) Confusion Matrix

TN: 56,819 non-fraud transactions correctly identified.
 FP: 45 non-fraud cases wrongly classified as fraud.
 FN: 36 fraud cases missed by the model.
 TP: 62 fraud cases correctly identified.

5.0 Discussion

The SVM model is very good with near perfect accuracy. The total samples are much more than the number of errors. It has a high predictive power with both high precision and recalls. The Constraint solver model generates a lot more false positives, or is more likely to mistake non-fraud transactions as fraud. Although it identifies the majority of frauds (62 of 98), the accuracy is worse due to a large proportion of false positive (non-fraud as fraud) cases. Hybrid (ML OR Constraint Solver) uses the logical OR, so it predicts fraud if either ML or the constraint model predicts fraud. This increases sensitivity such that it records fewer false negatives (only 16) but decreases precision (more false positives - 1,403). In essence, it's safer (harder to miss fraud) but noisier (predicting more non-fraud as fraud which is more false alarms). Hybrid (ML AND Constraint Solver) uses the logical AND. According to Hybrid (ML AND Constraint Solver), only the agreement between ML and constraint solver leads to prediction of fraud. The model is highly conservative (almost no incorrect identification of non-fraud as fraud), but it misses more fraud (36 out of 98).

While a combination of constraint solver and ML methods can be used to strike a balance between predictive capabilities and interpretability, the hybridization method enables a compromise on the precision and recall based on the needs of the domain. ML-only has the highest general predictive accuracy and the hybrid approaches have more controlled and explainable frameworks of decision-making.

6.0 Conclusion

The SVM model is recommended when the application needs a high level of accuracy and efficiency, whereas the Constraint-based and Hybrid models are more and more applicable in systems that are rule-based or safety-critical and the interpretability and logical validity of the system should be provided.

References

- [1] Akinkunmi, B. O., Bassey, P. C., & Ayorinde, I. T. (2013). A comparative analysis of knowledge modelling methodologies in artificial intelligence research. In *Proceedings of the International Conference on Science, Technology, Engineering, Education, Arts, Management & the Social Sciences (iSTEAMS Research Nexus 2013)*. University of Ibadan.
- [2] Apt, K. R. (2003). *Principles of constraint programming*. Cambridge University Press. <https://doi.org/10.1017/CBO9780511615320>
- [3] Ayorinde, I. T., & Idyorough, P. N. (2024). Exploring the frontiers of artificial intelligence: A comprehensive analysis. *Innovation in Science and Technology*, 3(4), 35–45. <https://doi.org/10.56397/IST.2024.07.02>
- [4] Bahnsen, A. C., Aouada, D., & Ottersten, B. (2015). Example-dependent cost-sensitive decision trees. *Expert Systems with Applications*, 42(19), 6609–6619. <https://doi.org/10.1016/j.eswa.2015.04.042>
- [5] Bessière, C. (2006). Constraint propagation. In F. Rossi, P. van Beek, & T. Walsh (Eds.), *Handbook of constraint programming* (pp. 29–83). Elsevier.
- [6] Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. C. (2011). Data mining for credit card fraud: A comparative study. *Decision Support Systems*, 50(3), 602–613. <https://doi.org/10.1016/j.dss.2010.08.008>
- [7] Bolton, R. J., & Hand, D. J. (2002). Statistical fraud detection: A review. *Statistical Science*, 17(3), 235–255. <https://doi.org/10.1214/ss/1042727940>
- [8] Breiman, L. (2001). Random forests. *Machine Learning*, 45, 5–32. <https://doi.org/10.1023/A:1010933404324>
- [9] Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence*, 47(1–3), 139–159. [https://doi.org/10.1016/0004-3702\(91\)90053-M](https://doi.org/10.1016/0004-3702(91)90053-M)
- [10] Carcillo, F., Le Borgne, Y.-A., Caelen, O., Kessaci, Y., Oblé, F., & Bontempi, G. (2021). Combining unsupervised and supervised learning in credit card fraud detection. *Information Sciences*, 557, 317–331. <https://doi.org/10.1016/j.ins.2019.05.042>
- [11] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20, 273–297. <https://doi.org/10.1007/BF00994018>
- [12] Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2015). Credit card fraud detection and concept-drift adaptation with delayed supervised information. In *2015 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8). IEEE. <https://doi.org/10.1109/IJCNN.2015.7280527>
- [13] Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2018). Credit card fraud detection: A realistic modeling and a novel learning strategy. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8), 3784–3797. <https://doi.org/10.1109/TNNLS.2017.2736643>
- [14] Dechter, R. (2003). *Constraint processing*. Morgan Kaufmann.
- [15] Doshi-Velez, F., & Kim, B. (2017). *Towards a rigorous science of interpretable machine learning* (arXiv:1702.08608). arXiv. <https://arxiv.org/abs/1702.08608>

- [16] Fages, J.-G., & Lapègue, T. (2014). Filtering AtMostNValue with difference constraints: Application to the shift minimisation personnel task scheduling problem. *Artificial Intelligence*, 212, 116–133. <https://doi.org/10.1016/j.artint.2014.04.001>
- [17] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4), Article 44, 1–37. <https://doi.org/10.1145/2523813>
- [18] Gangalakshmi, T., Sathya, V., Vinodhini, S., Chandralekha, E., Usha, V., & Ravikumar, S. (2023). Improving e-commerce fraud detection via machine learning: Comparative evaluation of model effectiveness. In *2023 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSSES)* (pp. 1–6). IEEE. <https://doi.org/10.1109/ICSSES60034.2023.10465515>
- [19] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.
- [20] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer.
- [21] Hooker, J. N. (2012). *Integrated methods for optimization* (2nd ed.). Springer. <https://doi.org/10.1007/978-1-4614-1900-6>
- [22] Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245), 255–260. <https://doi.org/10.1126/science.aaa8415>
- [23] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444. <https://doi.org/10.1038/nature14539>
- [24] McCarthy, J. (1989). Artificial intelligence, logic, and formalizing common sense. In R. H. Thomason (Ed.), *Philosophical logic and artificial intelligence* (pp. 161–190). Kluwer Academic Publishers. https://doi.org/10.1007/978-94-009-2448-2_6
- [25] Mitchell, T. M. (1997). Does machine learning really work? *AI Magazine*, 18(3), 11–20.
- [26] Newell, A., & Simon, H. A. (1976). Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3), 113–126. <https://doi.org/10.1145/360018.360022>
- [27] Nilsson, N. J. (2010). *The quest for artificial intelligence: A history of ideas and achievements*. Cambridge University Press.
- [28] Osunade, O., Ayorinde, I. T., & Ayinla, B. I. (2024). Knowledge discovery in research security practices among scientists using machine learning techniques. *International Journal of Computer Applications*, 186(9), 19.
- [29] Raghavan, P., & El Gayar, N. (2019). Fraud detection using machine learning and deep learning. In *Proceedings of the 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)* (pp. 334–338).
- [30] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). “Why should I trust you?”: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1135–1144). Association for Computing Machinery. <https://doi.org/10.1145/2939672.2939778>
- [31] Roberts, T., & Tonna, S. J. (2022). *Risk modeling: Practical applications of artificial intelligence, machine learning, and deep learning*. Wiley.
- [32] Rossi, F., van Beek, P., & Walsh, T. (Eds.). (2006). *Handbook of constraint programming*. Elsevier.
- [33] Russell, S. J., & Norvig, P. (2021). *Artificial intelligence: A modern approach* (4th ed.). Pearson.
- [34] Samuel, K. S., Olamiti, A. O., & Ayorinde, I. T. (2024). Crop recommendation analysis and validation in Nigeria using machine learning algorithms. *Journal of Science and Logistics in ICT Research*, 11(2).
- [35] Schulte, C., & Stuckey, P. J. (2008). Dynamic variable elimination during propagation solving. In *Proceedings of the 10th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming* (pp. 107–118). Association for Computing Machinery. <https://doi.org/10.1145/1389449.1389480>
- [36] Van Vlasselaer, V., Eliassi-Rad, T., Akoglu, L., Snoeck, M., & Baesens, B. (2017). GOTCHA! Network-based fraud detection for social security fraud. *Management Science*, 63(9), 3090–3110. <https://doi.org/10.1287/mnsc.2016.2489>
- [37] Whitrow, C., Hand, D. J., Juszczak, P., Weston, D., & Adams, N. M. (2009). Transaction aggregation as a strategy for credit card fraud detection. *Data Mining and Knowledge Discovery*, 18, 30–55. <https://doi.org/10.1007/s10618-008-0116-z>