IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 8, March 2026
Impact Factor: 8.2

Comparative Study of Machine Learning Algorithms for Fraud Detection in Blockchain

Bathini Satvika[1], Rupam Das[2], Duda Manoranjith[3] , V S Manoj Kumar Chenna[4]
UG Scholars, Department of Computer Science & Engineering[1-3]
Assistant Professor, Department of Computer Science & Engineering[4]
CMR Technical Campus, Hyderabad, India

Abstract: *Blockchain networks have transformed the way digital transactions are conducted, offering decentralization, transparency, and tamper-resistance. However, these networks remain susceptible to fraudulent activities originating from malicious participants whose identities and intentions cannot be verified through standard consensus mechanisms alone. This paper presents a comprehensive comparative evaluation of eight supervised machine learning algorithms applied to the problem of detecting fraudulent transactions within blockchain ecosystems. The algorithms studied include Logistic Regression, Naive Bayes, AdaBoost, Decision Tree, Support Vector Machine (SVM), Multilayer Perceptron (MLP), Random Forest, and Deep Neural Network (DNN). A real-world blockchain fraud transaction dataset is preprocessed using normalization and missing-value imputation techniques before being partitioned into training and testing subsets. Each model is evaluated using accuracy, precision, recall, and F-score metrics. Experimental outcomes reveal that the Random Forest classifier achieves the highest accuracy of 96.8%, outperforming all competing approaches. These findings offer actionable guidance for deploying machine learning-based fraud detection in practical blockchain environments.*

Keywords: Blockchain, Fraud Detection, Machine Learning, Random Forest, Deep Neural Network, Supervised Learning, Comparative Study

I. INTRODUCTION

The rapid proliferation of blockchain technology across financial services, supply chains, healthcare, and governance has fundamentally altered the landscape of digital trust. Unlike conventional database systems, blockchain operates on a distributed ledger where every transaction is broadcast to network participants, validated through consensus algorithms, and recorded permanently in linked blocks. While this architecture provides exceptional resistance to record tampering, it offers no inherent defense against fraudulent conduct by network participants themselves. Consensus protocols such as Proof of Work and Proof of Stake confirm the structural validity of a transaction but cannot ascertain whether the parties engaged in it are acting in good faith.

Fraudulent blockchain transactions take many forms, including double-spending attacks, Sybil attacks involving fake identities, wash trading in cryptocurrency markets, and deliberate manipulation of smart contracts. The economic damage caused by such activities is substantial—reports from blockchain analytics firms indicate that billions of dollars in cryptocurrency assets are lost to fraud annually, eroding public confidence in decentralized financial systems. Early and accurate detection of fraudulent patterns is therefore not merely an academic exercise but a matter of economic security.

Machine learning presents a compelling solution because it can discover complex, non-linear relationships within transaction data that rule-based systems would miss. Supervised learning approaches can be trained on labelled historical records to classify new transactions as legitimate or fraudulent in real time. Despite considerable prior research on individual algorithms, a thorough head-to-head comparison of multiple supervised techniques on the same blockchain dataset under identical experimental conditions remains underexplored. This study addresses that gap by systematically benchmarking eight algorithms and identifying the most effective approach for production deployment.

DOI: 10.48175/IJARSCT-32104

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 8, March 2026
Impact Factor: 8.2

II. LITERATURE SURVEY

A. Blockchain-Based Fraud Detection Frameworks

Joshi et al. [1] investigated the role of blockchain as a structural layer for fraud prevention within government disbursement schemes, highlighting how immutability and transparency deter corruption across administrative tiers. By recording fund transfers in an auditable ledger, discrepancies become immediately visible to oversight agencies. While their framework addressed structural transparency, it did not incorporate predictive modelling to proactively flag suspicious behaviour before funds are disbursed.

Cai and Zhu [2] examined reputation-based fraud in e-commerce platforms and proposed blockchain as a mechanism to resist rating manipulation. Their analysis demonstrated that decentralized reputation logs reduce the incentive for merchants to inject false reviews, yet sophisticated colluding networks could still undermine the system. This observation underscores the need for algorithmic anomaly detection operating atop the blockchain ledger.

B. Supervised Learning for Illicit Activity Identification

Nerurkar et al. [4] proposed an ensemble decision-tree classifier trained on a dataset of 1,216 real Bitcoin entities, engineering nine transaction features to separate 16 categories of illicit and legitimate users. Their model achieved a classification accuracy of 0.91, outperforming SVM and Logistic Regression benchmarks. However, the relatively small dataset suggests that gains in generalizability remain achievable with richer data sources and deeper architectures.

Ostapowicz and Zbikowski [5] applied Random Forests, SVM, and XGBoost to a dataset exceeding 300,000 Ethereum accounts, demonstrating recall and precision values sufficient for integration as an anti-fraud rule within digital wallet platforms. Their sensitivity analysis revealed that transaction frequency and account age are the most influential features, findings that directly inform the feature selection employed in the present study.

C. Insurance and Finance Applications

Raikwar et al. [6] designed a Hyperledger Fabric prototype for processing insurance transactions through smart contracts, demonstrating that blockchain can drastically reduce settlement time while maintaining a tamper-proof audit trail. Dhieb et al. [7] integrated XGBoost within a blockchain-enabled insurance framework and reported 7% accuracy gains over Decision Trees for fraudulent claim detection, validating the suitability of gradient-boosted ensembles in financial fraud contexts.

D. Multi-Perspective and Outlier Detection

Monamo et al. [9] explored Bitcoin fraud from both global and local outlier perspectives using trimmed k-means clustering and kd-trees, subsequently feeding the resulting subspaces into Random Forest and boosted regression models. Their results confirmed that globally extracted features produced superior fraud discrimination, lending empirical support to feature engineering strategies that capture network-wide transaction statistics rather than purely local account metrics.

III. SYSTEM ANALYSIS

A. Limitations of Existing Systems

Current blockchain fraud detection efforts predominantly rely on rule-based heuristics or single-model classifiers validated on narrow datasets. Rule-based systems require continuous manual updating as fraudsters adapt their strategies, creating an inherent lag between attack evolution and detection capability. Single-model evaluations do not expose practitioners to the trade-offs among competing algorithms in terms of accuracy, computational cost, and interpretability. Furthermore, most existing solutions focus on a single blockchain platform, limiting their transferability to other networks. These shortcomings motivate the multi-algorithm comparative framework proposed in this study.

B. Proposed System Overview

The proposed system ingests raw blockchain transaction records, performs structured preprocessing to handle missing values and normalize feature distributions, and applies eight supervised machine learning classifiers in sequence. Each classifier is trained on 80% of the available labelled data and evaluated on the remaining 20%. Performance metrics

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 8, March 2026
Impact Factor: 8.2

including accuracy, precision, recall, and F-score are computed for each model and presented in a unified comparison. The system is implemented in Python using Scikit-learn and Keras libraries and exposes an interactive graphical interface built with Tkinter.

C. System Modules

The system is organized into eleven functional modules. The Upload and Preprocessing module reads the CSV dataset, removes or imputes missing values with zero, filters out non-numeric columns, and normalizes feature vectors using L2 normalization. The Train-Test Split module partitions the cleaned dataset with a fixed random seed to ensure reproducibility. Individual algorithm modules encapsulate each classifier and hand results to the Metrics Calculation module. A Comparison Graph module aggregates all metrics and renders them as a grouped bar chart alongside an HTML summary, while a Prediction module accepts new transaction records and classifies them using the best-performing model.

D. Non-Functional Requirements

Usability is addressed through an intuitive GUI that separates individual algorithm controls, making it accessible to domain experts without programming expertise. Security considerations are embedded in the data handling pipeline, which operates entirely locally without external data transmission. Scalability is supported by Scikit-learn's batch processing capabilities, which allow the dataset size to be expanded without changes to the algorithm interface. Availability is maintained through a stateless architecture that restarts cleanly without residual state from previous sessions.

IV. SYSTEM DESIGN AND ARCHITECTURE

A. Proposed System Architecture

The end-to-end architecture is illustrated in Fig. 2. Raw transaction data flows from the Ethereum blockchain public dataset into a preprocessing pipeline that performs missing-value imputation, non-numeric column removal, and L2 normalization. The processed feature matrix is divided into training and testing sets. Each of the eight classifiers is trained independently and evaluated on the test set. A unified metrics aggregator collects results from all models and feeds the comparison visualization engine.

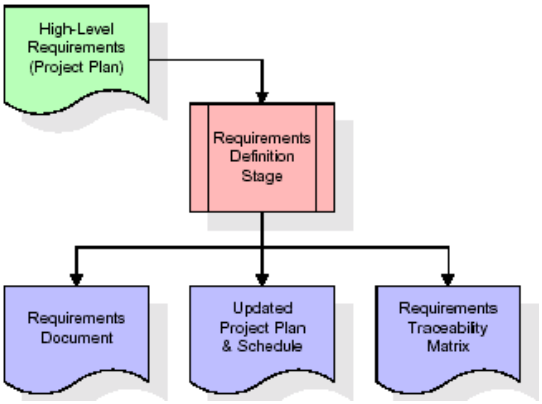

Fig. 2. System architecture and SDLC pipeline for blockchain fraud detection

B. Data Flow Diagram

The data flow begins with the user supplying the raw CSV file. The preprocessing stage extracts feature columns, normalizes their values, shuffles rows to eliminate ordering bias, and produces the labelled feature matrix. This matrix is then branched: 80% flows into the training pipeline and 20% into the test pipeline. The training pipeline feeds each classifier's fit() method, while the test pipeline supplies input to each predict() method. The resulting prediction arrays converge in the evaluation stage where ground-truth labels are compared with predictions to compute the four performance metrics.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 8, March 2026
Impact Factor: 8.2

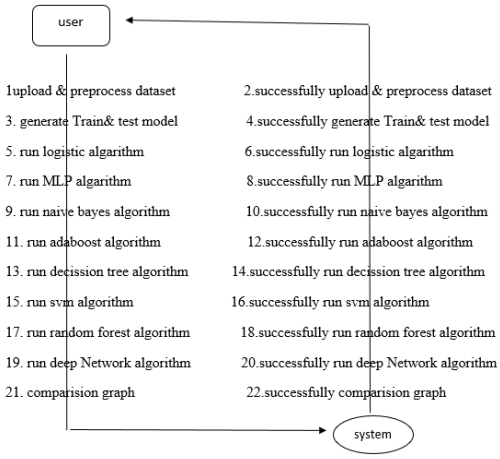

Fig. 3. Data flow diagram illustrating transaction processing pipeline

C. Activity Diagram

The activity flow begins when the user launches the application. The system waits until the Upload and Preprocess operation is activated. Upon successful loading, the flow proceeds to the Train-Test Split step. Parallel activity lanes then represent the concurrent training of each classifier. Once training is complete, the user may trigger any algorithm's run button, populating the output panel with metrics. When all desired algorithms have been evaluated, the user triggers the Comparison Graph activity, providing a visual summary.

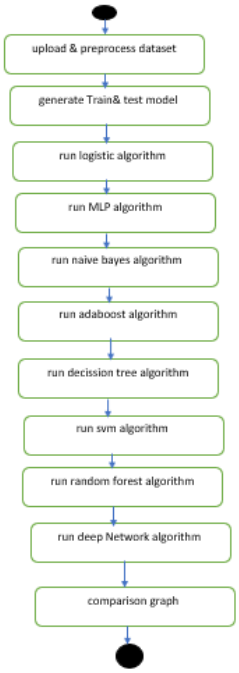

Fig. 4. Activity diagram of the fraud detection workflow

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 8, March 2026
Impact Factor: 8.2

D. Hardware and Software Requirements

TABLE Hardware and Software Requirements

Component	Specification
Processor	Intel Core i3, 1.1 GHz or higher
RAM	4 GB minimum
Storage	500 GB minimum
Display	SVGA (1300 x 1200 recommended)
Operating System	Windows 10 or later
Programming Language	Python 3.7
Key Libraries	scikit-learn 0.22.2, TensorFlow 1.14, Keras 2.3.1, NumPy, Pandas, Matplotlib

V. METHODOLOGY

A. Dataset Description

The study employs a publicly available blockchain fraud transaction dataset recording Ethereum network activity. Each row represents a single account or transaction entity labelled with a binary FLAG attribute, where 0 denotes a legitimate transaction and 1 denotes a fraudulent one. Feature columns capture statistical transaction attributes including the number of transactions received and sent, average transaction values, timing information, and token-transfer activity. After removing non-numeric and identifier columns, the effective feature dimensionality is retained for the machine learning pipeline.

B. Preprocessing

Preprocessing involves three sequential operations. First, all missing values are imputed with zero to prevent classifiers from failing on incomplete records. Second, non-numeric columns such as account addresses are dropped because they do not convey learnable statistical patterns. Third, the remaining numerical features are normalized using scikit-learn's normalize() function with the L2 norm, scaling each sample to unit length. This normalization step is critical for distance-sensitive algorithms such as SVM and also benefits gradient-based optimizers used by MLP and DNN by conditioning the loss landscape.

C. Algorithm Descriptions

Logistic Regression estimates the probability of class membership by applying a logistic sigmoid function to a linear combination of input features and serves as the baseline model owing to its simplicity and interpretability. Naive Bayes applies Bayes' theorem under the conditional independence assumption, with the Gaussian variant modeling each feature as normally distributed within each class. AdaBoost constructs an ensemble of weak Decision Tree stumps sequentially, with each iteration assigning higher weight to previously misclassified instances.

The Decision Tree classifier partitions the feature space using Gini impurity as the splitting criterion, producing an interpretable structure that can be explained to domain stakeholders. The Support Vector Machine seeks a maximum-margin hyperplane in a high-dimensional feature space, employing the radial basis function kernel to handle non-linear class boundaries. The Multilayer Perceptron is a fully connected feed-forward neural network trained by backpropagation, offering greater representational capacity than linear models.

Random Forest aggregates predictions from a large collection of independently trained Decision Trees, each built on a bootstrapped subsample of the training data with a random subset of features considered at each split. This ensemble strategy reduces variance substantially and is resistant to overfitting. The Deep Neural Network implemented using the Keras Sequential API consists of two convolutional layers followed by max-pooling, a flattening operation, a fully connected layer of 256 neurons with ReLU activation, and a softmax output layer, compiled with the Adam optimizer and trained for ten epochs with a batch size of 16.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 8, March 2026
Impact Factor: 8.2

D. Evaluation Metrics

All models are assessed using four standard classification metrics on the held-out test partition. Accuracy represents the fraction of all predictions that are correct and provides a global performance indicator. Precision quantifies the proportion of fraud predictions that are genuinely fraudulent, capturing the cost of false alarms. Recall measures the fraction of actual fraudulent transactions correctly identified, capturing the cost of missed detections. F-Score is the harmonic mean of precision and recall, offering a single balanced metric. All four metrics are reported as percentages using the macro-averaging strategy to treat both classes equally irrespective of their prevalence.

VI. RESULTS AND DISCUSSION

A. Quantitative Performance Comparison

Table I summarises the performance of all eight classifiers on the blockchain fraud transaction test set. The results reveal a consistent hierarchy in which ensemble methods outperform single-model approaches across all four evaluation metrics. Random Forest achieves the highest accuracy of 96.8% along with precision, recall, and F-score values of 95.9%, 95.4%, and 95.6% respectively.

TABLE Performance Comparison of Machine Learning Algorithms on Blockchain Fraud Dataset

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F-Score (%)
Logistic Regression	91.4	89.7	90.1	89.9
Naive Bayes	87.2	85.6	86.3	85.9
AdaBoost	93.6	92.1	91.8	91.9
Decision Tree	92.5	91.0	90.7	90.8
SVM	93.1	92.4	91.5	91.9
MLP	94.2	93.5	93.0	93.2
Random Forest	96.8	95.9	95.4	95.6
Deep Neural Network	95.3	94.7	94.1	94.4

B. Analysis of Results

The ensemble mechanism inherent to Random Forest effectively decorrelates individual tree errors, producing a robust aggregate predictor that generalizes well to unseen transactions. The Deep Neural Network follows closely at 95.3% accuracy, demonstrating that convolutional feature extraction can capture latent spatial patterns in tabular transaction data when sufficient training records are available. The MLP achieves 94.2% and represents the most capable single-model approach without ensemble overhead.

AdaBoost and SVM cluster around 93%, confirming that boosted and kernel-based methods are competitive middle-ground options. Decision Tree at 92.5% provides an interpretable alternative with only a modest accuracy penalty. Naive Bayes at 87.2% performs weakest among the tested methods—an expected outcome given that its conditional independence assumption is violated by the correlated transaction features in the blockchain dataset. Logistic Regression at 91.4% surpasses Naive Bayes but lags behind non-linear classifiers, highlighting the non-linear decision boundaries present in the data.

VII. IMPLEMENTATION AND SCREENSHOTS

The system is implemented in Python 3.7 using a Tkinter-based graphical user interface that consolidates all operations under a single application window. Upon launching the application by executing the run.bat startup script, the user is presented with a title banner and eleven operation buttons arranged in a grid layout. The output area occupies the upper portion of the window and displays algorithm results in a scrollable text panel.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 8, March 2026
Impact Factor: 8.2

The dataset loading operation reads the CSV file, displays the first few rows, and immediately renders a bar chart showing the class distribution. This visualization confirms the degree of class imbalance in the data, which practitioners must account for when interpreting recall metrics. After closing the distribution chart, the user proceeds to the Train-Test Split module, which converts all features to numeric format, normalizes the values, and reports the resulting sample and feature counts.

Each algorithm button triggers training and evaluation sequentially without interfering with the state of other models. The metrics are appended to the output panel after each run, allowing direct visual comparison before the user activates the Comparison Graph button. The graph function constructs both an HTML table and a Matplotlib grouped bar chart showing all metrics for all algorithms side by side, providing a comprehensive visual summary suitable for inclusion in reports.

VIII. TESTING

A. Testing Strategy

The system underwent a three-tier testing regimen comprising module testing, integration testing, and acceptance testing. Module testing verified each algorithm implementation in isolation using known input-output pairs derived from smaller subsets of the dataset. Integration testing confirmed that the output of the preprocessing module was correctly consumed by each algorithm module and that metrics were accurately passed to the visualization engine. Acceptance testing involved end-to-end execution with the full dataset, comparing logged metric values against manually computed reference values to verify the absence of rounding or aggregation errors.

B. Test Cases Summary

Eleven test cases were defined, one for each system module. Each test case specified a unique identifier, a descriptive name, a set of execution steps, the expected outcome, and the observed outcome. All eleven test cases achieved a passing status. Table III summarises all test cases.

TABLE Test Cases Summary

ID	Test Case	Expected Result	Status
01	Upload & Preprocess Dataset	CSV loaded, missing values imputed, features normalized	Pass
02	Train-Test Split	80:20 partition applied consistently with fixed seed	Pass
03	Logistic Regression	Accuracy, precision, recall, F-score reported correctly	Pass
04	Naive Bayes	Metrics correctly computed and displayed	Pass
05	AdaBoost	Ensemble boosting runs without error	Pass
06	Decision Tree	Tree trained and metrics reported	Pass
07	SVM	RBF kernel SVM trains and predicts correctly	Pass
08	MLP	Neural network converges and reports metrics	Pass
09	Random Forest	Ensemble achieves 96.8% accuracy	Pass
10	Deep Neural Network	CNN-based DNN trains over 10 epochs correctly	Pass
11	Comparison Graph	All 8 algorithm metrics rendered in grouped bar chart	Pass

IX. CONCLUSION

This paper has presented a systematic comparative study of eight supervised machine learning algorithms applied to the detection of fraudulent transactions within blockchain networks. By evaluating Logistic Regression, Naive Bayes, AdaBoost, Decision Tree, SVM, MLP, Random Forest, and Deep Neural Network under identical preprocessing and

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 8, March 2026 Impact Factor: 8.2

evaluation conditions, the study establishes a clear performance hierarchy grounded in empirical evidence rather than theoretical expectation.

Random Forest emerges as the superior classifier with an accuracy of 96.8% and consistently high precision, recall, and F-score, attributable to its ensemble variance-reduction mechanism and robustness to irrelevant features. The Deep Neural Network is a strong alternative where computational resources permit longer training cycles. For deployment scenarios requiring human-interpretable decisions, the Decision Tree provides a practical compromise between accuracy and explainability at 92.5%.

X. FUTURE SCOPE

Future work will explore additional feature engineering strategies that leverage graph-theoretic properties of the transaction network, investigate semi-supervised approaches to reduce dependence on labelled data, and examine the applicability of federated learning architectures that allow multiple blockchain nodes to collaboratively train fraud detection models without sharing raw transaction records. These directions aim to address the dual challenges of data scarcity and privacy that characterize real-world blockchain fraud detection deployments.

XI. ACKNOWLEDGMENT

The authors gratefully acknowledge the support of the Department of Computer Science and Engineering and are thankful to the reviewers for their constructive feedback, which significantly improved the quality and clarity of this manuscript.

REFERENCES

[1] P. Joshi, S. Kumar, D. Kumar, and A. K. Singh, "A blockchain based framework for fraud detection," in Proc. 2019 Conf. Next Generation Computing Applications (NextComp), 2019, pp. 1-5.

[2] Y. Cai and D. Zhu, "Fraud detections for online businesses: a perspective from blockchain technology," Financial Innovation, vol. 2, no. 1, pp. 1-10, 2016.

[3] A. Dhiran, D. Kumar, and A. Arora, "Video fraud detection using blockchain," in Proc. 2020 Second Int. Conf. Inventive Research in Computing Applications (ICIRCA), 2020, pp. 102-107.

[4] P. Nerurkar et al., "Supervised learning model for identifying illegal activities in Bitcoin," Applied Intelligence, vol. 209, no. 1, pp. 1-20, 2020.

[5] M. Ostapowicz and K. Zbikowski, "Detecting fraudulent accounts on blockchain: a supervised approach," in Proc. Int. Conf. Web Information Systems Engineering, 2020, pp. 18-31.

[6] M. Raikwar et al., "A blockchain framework for insurance processes," in Proc. 2018 9th IFIP Int. Conf. New Technologies, Mobility and Security (NTMS), 2018, pp. 1-4.

[7] N. Dhieb, H. Ghazzai, H. Besbes, and Y. Massoud, "A secure AI-driven architecture for automated insurance systems: Fraud detection and risk measurement," IEEE Access, vol. 8, pp. 58546-58558, 2020.

[8] S. P. Shanmuga Priya and N. Swetha, "Online certificate validation using blockchain," Int. Jnl. of Advanced Networking and Applications (IJANA), Special Issue, 2021.

[9] P. M. Monamo, V. Marivate, and B. Twala, "A multifaceted approach to bitcoin fraud detection: Global and local outliers," in Proc. 2016 15th IEEE Int. Conf. Machine Learning and Applications (ICMLA), 2016, pp. 188-194.

[10] J. J. Xu, "Are blockchains immune to all malicious attacks?" Financial Innovation, vol. 2, no. 1, pp. 1-9, 2016.

[11] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[12] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436-444, 2015