# Reproducing Experiments on Credit Card Fraud Detection Using Machine Learning

Raya Namazi

August 16, 2025

**Abstract**

This paper presents the reproduction of experiments outlined in ”Credit Card Fraud Detection using Machine Learning: A STUDY” by Tiwari et al[7]. We implemented a range of machine learning techniques to detect credit card fraud and evaluated their performance.

## 1 Introduction

CREDIT cards play a crucial role in the modern world by simplifying transactions and enhancing convenience for consumers and businesses alike. However, this convenience comes with a significant drawback: credit card fraud poses a persistent and costly challenge to the financial industry. Each year, billions of dollars are lost globally due to fraudulent transactions, impacting financial institutions, merchants, and cardholders. As a result, developing effective methods to detect and prevent credit card fraud has become a pressing priority.

In this paper, we aim to reproduce the experiments reviewed in the study by Tiwari et al. [7], which evaluates multiple machine learning approaches to address the credit card fraud detection problem. Their work explores a range of techniques, including Hidden Markov Models, Decision Trees, Logistic Regression, Support Vector Machines, Genetic Algorithms, Neural Networks, Random Forests, and Bayesian Belief Networks. By replicating their experiments, we seek to verify their results and evaluate the practical effectiveness of these methods in identifying fraudulent activities.

Reproducing these experiments, however, presents several challenges. One significant obstacle is the limited availability of real-world transaction data. Due to security and privacy concerns, financial institutions rarely release authentic datasets to the public, forcing researchers to rely on alternative sources. Additionally, fraud detection datasets are typically characterized by a severe class imbalance: legitimate transactions vastly outnumber fraudulent ones, often by a ratio of thousands to one. This imbalance complicates the task of accurately classifying fraudulent transactions, as machine learning models may become biased toward predicting transactions as legitimate. To overcome this, we employ data augmentation techniques and cost-sensitive algorithms designed to prioritize the detection of fraud, thereby improving model performance.

For this study, we utilize multiple datasets, some of which are not derived from real transactions but are instead generated using data simulators. While these synthetic datasets enable us to test the machine learning approaches, they may not fully reflect the intricacies of actual fraud patterns encountered in practice.

## 2 Methodology

### 2.1 Experimental Datasets

In this study, we utilized the following datasets to evaluate machine learning techniques for credit card fraud detection:

- **European Dataset**: This dataset, hosted on Kaggle [4], includes 28 anonymized features (unexplained for security reasons) along with explicit features such as Time and Amount of the transaction. Due to its highly imbalanced nature, the oversampling technique SMOTE (Synthetic Minority Oversampling Technique) [2] was employed to address the disparity between fraudulent and legitimate transactions.

- **S-FFSD Dataset**: Simulated by F. Kang et al[3]. for their Convolutional Neural Network (CNN) experiment, this dataset includes key transaction features such as Time, Source, Target, Amount, Location, and Type.

## 2.2 Data Preprocessing

Before analysis, the following steps were applied which are:

- **Handling missing values**: We verified that all datasets were free of missing values to prevent issues during training.
- **Categorical Endcoding**: All categorical features were transformed into numerical format using techniques such as one-hot encoding, ensuring compatibility with machine learning models that require numeric input.
- **Outlier Removal**: In certain datasets, outliers were identified and removed to prevent introducing bias into the trained models.
- **Train-Test Splitting**: The processed datasets were divided into training and test sets. For highly imbalanced datasets, such as the European Dataset, we ensured that the class balance ratio remained consistent between the sets. Splitting was performed before normalization to prevent data leakage.
- **Normalization/Standardization**: All numerical features were standardized to ensure uniform scaling across datasets, mitigating biases during model training.
- **OverSampling/UnderSampling:** Data augmentation techniques, including oversampling and undersampling, were applied to training sets with high class imbalance to enable models to effectively learn patterns for both fraudulent and legitimate transactions, reducing bias toward the majority class.

## 2.3 Machine Learning Techniques

We replicated the machine learning techniques evaluated in the original study:

**Hidden Markov Model (HMM).** A stochastic model that captures cardholder transaction sequences using hidden states to represent purchase types (e.g., low, medium, high price ranges). Implemented as described in [6] with three observation symbols, it detects deviations from typical spending behavior using Python's hmmlearn library. Training was conducted offline, while detection occurred online, flagging fraudulent transactions to prevent processing and storing legitimate ones for future model updates. For our reproduction, we utilized an existing implementation available at [5].

**Decision Trees.** A supervised learning algorithm that recursively splits data into hierarchical nodes based on feature values to classify transactions as fraudulent or legitimate. Implemented using scikit-learn with pruning to prevent overfitting by removing low-impact branches, as described in [7]. A cost-sensitive approach was applied to address class imbalance, assigning higher penalties to misclassified fraudulent transactions to enhance detection performance.

**Logistic Regression.** Logistic regression is a linear classification algorithm that models the probability of a transaction being fraudulent as a logistic (sigmoid) function of a weighted sum of the input features. Given an input vector $x$, the model computes the log-odds of the positive class as

$$\log \frac{P(y=1 \mid x)}{P(y=0 \mid x)} = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n,$$

and applies the logistic transformation to map this value to the interval $[0, 1]$. Predictions are made by thresholding the probability, with lower thresholds often used in fraud detection to increase recall.

In our experiments, we implemented logistic regression using `scikit-learn` and systematically explored different solvers (`saga` and `liblinear`), regularization penalties ($\ell_1$, $\ell_2$, and elastic net), inverse regularization strengths $C \in \{0.01, 0.1, 1, 10, 100\}$, and maximum iteration limits. Because the dataset is highly imbalanced, we considered multiple class weighting schemes, including `'balanced'` and manually assigned weights ($\{0 : 1,\ 1 : 10\}$ and $\{0 : 1,\ 1 : 100\}$), to penalize misclassification of the minority (fraud) class more heavily.

To assess the impact of class-imbalance handling strategies, models were trained on three variants of the training data: (i) the original imbalanced dataset, (ii) an oversampled dataset generated by SMOTE, and (iii) an undersampled dataset obtained by random undersampling of legitimate transactions. Hyperparameter tuning was performed via `RandomizedSearchCV`, optimizing for metrics relevant to fraud detection such as precision, recall, and F1-score. This approach allowed us to identify configurations that maximized detection performance while controlling false positives.

**Support Vector Machines (SVM).** Support Vector Machines are supervised learning algorithms that find an optimal separating hyperplane between classes by maximizing the margin between support vectors. For nonlinear classification problems, the kernel trick is used to implicitly map inputs into a higher-dimensional feature space where a linear separation is possible. In our experiments, we employed the radial basis function (RBF) kernel, which is well-suited for complex, nonlinearly separable patterns such as those found in fraudulent transaction data.

We implemented the SVM using `scikit-learn`'s `SVC` class with probability estimates enabled to allow ROC and precision–recall analysis. The model was tuned via `RandomizedSearchCV` over key hyperparameters: the regularization parameter $C$ (sampled from a log-uniform distribution over $[10^{-3}, 10^{3}]$), the RBF kernel coefficient $\gamma$ (log-uniform over $[10^{-4}, 1]$), and various class-weight configurations to give additional importance to the minority fraud class ($\{0 : 1,\ 1 : w\}$ for $w \in \{1, 10, 50, 100, 200\}$). The search used 20 random combinations evaluated via 3-fold cross-validation, optimizing for the F1-score to balance precision and recall.

To examine the impact of class imbalance handling, models were trained on three versions of the training data: (i) the original imbalanced dataset, (ii) a SMOTE-oversampled dataset, and (iii) a randomly undersampled dataset. For each dataset variant, the best model from the random search was saved for later evaluation. On the held-out test set, we computed accuracy, precision, recall, F1-score, and confusion matrix entries, with metrics stored in JSON format for reproducibility and visualized using `ConfusionMatrixDisplay`. This procedure allowed us to directly compare SVM performance under different data balancing strategies.

**Neural Networks.** Feed-forward neural networks (multilayer perceptrons) are flexible nonlinear models capable of capturing complex interactions between features. In this work, we implemented a fully connected architecture using TensorFlow/Keras, consisting of three to four hidden layers (16–128 units) with ReLU activation and interleaved dropout layers (rate 0.3) to mitigate overfitting. The output layer used a single sigmoid neuron to produce a fraud probability between 0 and 1. Models were compiled with the Adam optimizer and binary cross-entropy loss, with evaluation metrics including precision, recall, and F1-score for the fraud class.

To address class imbalance, we employed two strategies: (i) computing class weights from the training data and passing them to `model.fit` for the original imbalanced dataset, and (ii) applying SMOTE oversampling or random undersampling, in which case class weights were not used. We trained on three dataset variants—original, SMOTE, and undersampled—each with batch sizes in $\{32, 64, 128, 256\}$, using a validation split of 0.2 and early stopping (patience = 5 epochs) to restore the best weights.

After training, models were saved in `.h5` format and evaluated on the held-out test set at decision thresholds 0.3, 0.4, and 0.5. For each configuration, we computed precision and F1-score for the fraud class, storing the results in a table and plotting performance curves as a function of batch size and threshold for each dataset variant. This allowed direct comparison of how network capacity, data balancing, and decision threshold jointly affect fraud detection performance.

**Random Forest.** Random Forests are ensemble learning methods that construct a collection of decision trees during training and output the mode of their predictions. Each tree is trained on a bootstrap sample of the data with a random subset of features considered at each split, which reduces variance and improves generalization. This approach is well-suited for high-dimensional, tabular datasets like credit card transactions, as it can capture nonlinear feature interactions and is relatively robust to noise.

We implemented the model using `scikit-learn`'s `RandomForestClassifier` with `class_weight='balanced'` to mitigate the effects of class imbalance. Hyperparameters were tuned via `RandomizedSearchCV` over a search space including the number of trees ($n\_estimators \in \{50, 100, 200, 500\}$), maximum tree depth (`None`, 10, 20, 30), minimum samples per split (2, 5, 10), minimum samples per leaf (1, 2, 4), and maximum features considered at each split (`auto`, `sqrt`, `log2`). The search used three-fold cross-validation and optimized for the F1-score of the fraud class.

After selecting the best hyperparameters, we retrained the model on the full training data and evaluated it on the held-out test set. We reported accuracy, precision, recall, F1-score, and confusion matrix entries, storing metrics in JSON format for reproducibility and visualizing the confusion matrix using `ConfusionMatrixDisplay`. This process provided a strong ensemble baseline for comparison with other models in terms of fraud detection accuracy and robustness.

**Bagging Classifier.** Bootstrap Aggregating (Bagging) is an ensemble method designed to reduce variance and improve the stability of machine learning algorithms. It works by training multiple base learners—commonly decision trees—on different bootstrap samples of the training data and aggregating their predictions through majority voting (for classification) or averaging (for regression). By combining diverse models, bagging mitigates overfitting and enhances generalization, particularly for unstable learners.

We implemented the bagging approach using `scikit-learn`'s `BaggingClassifier` with a `DecisionTreeClassifier` as the base estimator. Hyperparameters were tuned via `RandomizedSearchCV` over a parameter grid including the number of estimators (`n_estimators` $\in \{50, 100, 500\}$), maximum proportion of samples per estimator (`max_samples` $\in \{0.1, 0.4, 0.7, 1.0\}$), maximum proportion of features per estimator (`max_features` $\in \{0.1, 0.4, 0.7, 1.0\}$), and whether sampling was performed with or without replacement (`bootstrap` $\in \{\mathrm{True}, \mathrm{False}\}$). The search used five-fold cross-validation and optimized for F1-score.

The best model was retrained on the full training dataset and evaluated on the test set. We computed the Fraud Catching Rate (FCR), False Alarm Rate (FAR), Balanced Classification Rate (BCR), Matthews Correlation Coefficient (MCC), and F1-score. These metrics provided a comprehensive assessment of the model's performance, balancing detection effectiveness with minimization of false positives. All evaluation results were stored in JSON format for reproducibility and the confusion matrix was visualized for interpretation.

**Long Short-Term Memory (LSTM) Recurrent Neural Network.** LSTM networks are a type of recurrent neural network (RNN) designed to model sequential data by capturing temporal dependencies through gated memory cells. In credit card fraud detection, transactions occur as time-ordered sequences, and LSTMs can learn patterns in these sequences that may indicate fraudulent behavior. Our implementation involves preprocessing the transaction data into fixed-length sequences of 10 consecutive transactions, assigning the label of the last transaction in each sequence as the target. We address class imbalance using three training variants: the original imbalanced data with class weights computed to penalize misclassification of fraud more heavily, SMOTE oversampled data, and randomly undersampled data without class weights.

The network architecture consists of two stacked LSTM layers with 64 and 32 units, respectively, interspersed with dropout layers (rate 0.2) to prevent overfitting. The output layer is a single sigmoid neuron that outputs the probability of fraud for the last transaction in the sequence. Models are compiled with the Adam optimizer and binary cross-entropy loss, with precision, recall, and accuracy monitored during training.

We train the models using batch sizes of 32, 64, 128, and 256 with early stopping based on validation loss (patience of 5 epochs) to restore the best weights. Evaluation is performed on the held-out test sequences with multiple decision thresholds (0.3, 0.4, 0.5) to analyze trade-offs between precision and recall. Metrics such as precision, recall, and F1-score for the fraud class are computed and compared across dataset variants and batch sizes.

This LSTM approach leverages temporal transaction dynamics to enhance fraud detection beyond static feature-based models and shows promising results, particularly when combined with oversampling techniques to mitigate class imbalance.

**K-Nearest Neighbors (KNN).** KNN is a non-parametric, instance-based learning algorithm that classifies a data point based on the majority class among its k closest neighbors in the feature space. Its simplicity and interpretability make it a common baseline method for fraud detection. In our experiments, we applied KNN to the credit card transaction data using the European Dataset. We addressed class imbalance by training on both the original imbalanced dataset and an oversampled dataset generated with SMOTE. Model hyperparameters were tuned via randomized search with 5-fold cross-validation, optimizing the F1-score. The search space included the number of neighbors $k \in 1, 3, 5, 7, 9, 11, 13, 15$, weight functions (uniform, distance, and balanced), distance metrics (minkowski and euclidean), and the Minkowski parameter $p \in 1, 2$.

**Convolutional Neural Network (CNN).** CNNs are deep learning models primarily designed for spatial data and are widely used in image processing. In the context of credit card fraud detection, CNNs can be applied to structured transaction data by transforming features into a spatially meaningful format or by leveraging temporal/spatial relationships within the data. For this study, we used the CNN architecture and experimental setup from the repository [1], which provided a robust framework for fraud detection using deep learning. The model consists of convolutional layers that extract hierarchical feature representations, followed by fully connected layers that perform binary classification between fraudulent and legitimate transactions.

The CNN was trained on both original and augmented datasets (using oversampling techniques like SMOTE) to address class imbalance. Model performance was evaluated using precision, recall, F1-score, and confusion matrices on a held-out test set. This approach demonstrates the potential of CNNs to capture complex feature interactions and improve fraud detection accuracy beyond traditional machine learning methods.

After identifying the best hyperparameters, the model was retrained on the respective training set and evaluated on the held-out test set. We computed standard classification metrics such as accuracy, precision, recall, F1-score, as well as confusion matrix components to assess the model's ability to detect fraud while controlling false positives.

The KNN approach benefits from its straightforward implementation and interpretability but may be challenged by high-dimensional data and class imbalance. Using oversampling methods like SMOTE improved its performance by providing a more balanced representation of the minority fraud class during training.

**Bayesian Belief Network.** A Bayesian Belief Network (BBN) is a probabilistic graphical model that represents the dependencies among variables using a directed acyclic graph (DAG). Each node in the graph corresponds to a random variable, and edges represent conditional dependencies. BBNs are particularly well-suited for fraud detection problems because they can incorporate prior knowledge and handle uncertainty in predictions.

In our implementation, we first selected a subset of features from the European credit card fraud dataset, focusing on variables such as `V3`, `V4`, `V7`, `V10`, `V11`, `V12`, `V14`, `V16`, `V17`, and the `Amount` of the transaction, along with the target class label. These continuous features were discretized into five bins using the `KBinsDiscretizer` with a quantile strategy, enabling the Bayesian network to process them as categorical variables.

We employed a Tree-Augmented Naive Bayes (TAN) structure learning algorithm with `V14` as the root node, as recommended in the literature for improved feature dependency modeling. The `pgmpy` library was used to estimate the network structure and fit the parameters using a Dirichlet prior with pseudo-count smoothing to avoid zero-probability issues.

Once trained, inference was performed using Variable Elimination. For each transaction in the test set, the model calculated the posterior probability of the fraud class given the observed evidence (feature values). Transactions with a predicted probability above a threshold of 0.35 were classified as fraudulent. The threshold was chosen empirically to balance precision and recall for the highly imbalanced dataset.

Model evaluation included metrics such as Precision, Recall, F1-score, and AUC-ROC. This approach allowed the model to explicitly capture feature interdependencies, which is beneficial in fraud detection where certain combinations of transaction attributes are more indicative of fraud than individual features alone.

**Fuzzy C-Means Clustering** Fuzzy C-Means (FCM) is a soft clustering algorithm where each data point belongs to multiple clusters with varying degrees of membership. This is useful for fraud detection, as fraudulent patterns may overlap with legitimate transactions.

We implemented FCM from scratch, initializing a random membership matrix and iteratively updating cluster centers and memberships until convergence, using a fuzziness parameter $m = 2.0$. The dataset was preprocessed by filling missing values, scaling `Time` and `Amount`, and sampling all fraud cases with a balanced subset of non-fraud cases. PCA was applied for visualization.

Anomaly scores were computed as $1 - \max(\text{membership values})$, with the classification threshold chosen via Youden's J statistic from the ROC curve. Model performance was evaluated using AUC, Precision, Recall, and F1-score, alongside cluster quality metrics such as Silhouette Score, Calinski–Harabasz Index, and Davies–Bouldin Index.

**Genetic Algorithm Optimization** We employed a Genetic Algorithm (GA) to jointly optimize feature selection and hyperparameters for a Gradient Boosting Classifier (GBC). The GA operates on individuals encoding both a binary mask for feature inclusion and continuous values for hyperparameters (*learning rate*, *max depth*, *subsample*, and *min samples split*).

A population of size 15 was evolved over 8 generations using two-point crossover, bit-flip mutation, and tournament selection. The fitness function was defined as the mean AUC score over 3-fold stratified cross-validation on a small subset of the training data to reduce computation time.

The best individual from GA optimization was then retrained on the full training set and evaluated on the test set. This approach yielded an optimal subset of features and tuned hyperparameters, aiming to improve fraud detection performance while reducing model complexity.

## 3 Results

We evaluated each method using standard metrics including precision, recall, and F1-score. Table 1 summarizes the performance of our models and compares them to the results reported by Tiwari et al. [7].

It is important to note that the original datasets referenced in [7] are not publicly available due to security and privacy concerns. Therefore, we could not replicate their experiments on the exact same data and cannot directly compare our results to theirs. Instead, we used publicly available datasets such as the European credit card fraud dataset from Kaggle, which may differ in feature distribution and fraud patterns from the original datasets.

Figure 1 illustrates the impact of batch size and data augmentation on the F1 score and precision for the LSTM model. The plot reveals how different batch sizes influence the stability and effectiveness of model training, with moderate batch sizes generally yielding better performance. Data augmentation techniques such as SMOTE help mitigate class imbalance, improving recall but sometimes at the cost of precision. This visualization highlights the importance of careful hyperparameter tuning to optimize model outcomes.

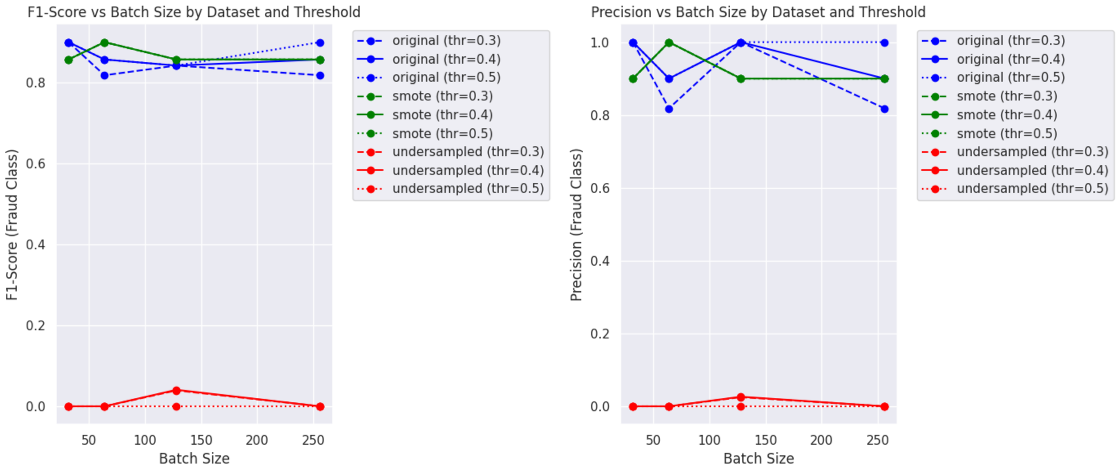

Figure 1: Effect of batch size and data augmentation on F1 score and precision for the LSTM model.

Training with smaller batch sizes introduces more gradient noise, leading to less stable learning and fluctuating F1 scores. Additionally, undersampling the majority class to address imbalance reduces the amount of training data, potentially causing loss of valuable information and negatively impacting model performance. Finally, selecting an optimal sigmoid threshold is crucial as it balances precision and recall, maximizing the F1 score by minimizing false positives and false negatives.

| Method | Dataset Source | Dataset Version | F1-score | Precision | Recall |
|---|---|---|---|---|---|
| Logistic Regression | European Dataset | Original | 0.87 | 0.93 | 0.81 |
| Support Vector Machine | European Dataset | Original | 0.86 | 0.88 | 0.84 |
| Random Forest | European Dataset | Original | 0.78 | 0.88 | 0.69 |
| Neural Network | European Dataset | SMOTE | 0.70 | 0.63 | 0.78 |
| Bagging Classifier | European Dataset | Original | 0.78 | 0.86 | 0.72 |
| Recurrent Neural Network | European Dataset | SMOTE | 0.85 | 0.90 | 0.81 |
| K-Nearest Neighbors | European Dataset | Original | 0.75 | 0.80 | 0.71 |
| Decision Tree | European Dataset | SMOTE | 0.76 | 0.76 | 0.76 |
| Bayesian Belief Network | European Dataset | Original | 0.82 | 0.77 | 0.91 |
| Convolutional Neural Network | S-FFSD Dataset | Original | 0.68 | 0.66 | 0.7 |
| Fuzzy Clustering | European Dataset | Original | 0.61 | 0.59 | 0.82 |
| Genetic Algorithm | European Dataset | Original | 0.51 | 0.5 | 0.52 |

Table 1: Performance of reproduced models on the credit card fraud detection task.

# Discussion

Across all methods, **Logistic Regression** and **Support Vector Machine (SVM)** on the original European dataset achieved the highest overall performance, with F1-scores of 0.87 and 0.86, respectively. Logistic Regression exhibited the highest precision (0.93), while SVM provided a slightly more balanced precision–recall trade-off (0.88/0.84).

Ensemble methods such as **Random Forest** and **Bagging Classifier** maintained high precision (0.88 and 0.86) but lower recall (0.69 and 0.72), indicating a tendency to miss positive cases.

Neural models showed greater variability. The **Recurrent Neural Network (RNN)** with SMOTE augmentation achieved competitive performance (F1 = 0.85, precision = 0.90, recall = 0.81), surpassing the feedforward **Neural Network** with SMOTE (F1 = 0.70), which prioritized recall (0.78) at the expense of precision (0.63).

**Genetic Algorithm Optimization** yielded the lowest scores (F1 = 0.51), reflecting limited capacity to capture complex patterns in the data.

Overall, results indicate that linear/kernel-based models and SMOTE-augmented RNNs are most effective for this dataset, while ensemble methods may require tuning to improve recall.

# Conclusion

This study demonstrates that Logistic Regression and Support Vector Machine achieve the best balance of precision and recall on the European dataset. Additionally, SMOTE-augmented Recurrent Neural Networks provide competitive performance, suggesting that combining data augmentation with advanced models can effectively address class imbalance. Future work should focus on optimizing ensemble methods to improve recall without sacrificing precision.

# References

[1] AI4Risk. Antifraud: A repository for financial fraud detection, 2025. Accessed: 2025-08-10.

[2] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[3] T. Yi Z. Liqing F. Kang, C. Dawei. Credit card fraud detection using convolutional neural networks. pages 483–490, 2016.

[4] Kaggle. ”credit card fraud detection dataset”. `https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud`, 2016.

[5] Saman Khamesian. Credit-card-fraud-detection, n.d. Accessed: 2025-08-10.

[6] Ajay Srivastava, Amlan Kundu, Shamik Sural, and Arun Majumdar. Credit card fraud detection using hidden markov model. *IEEE Transactions on Dependable and Secure Computing*, 5(1):37–48, Jan.-March 2008.

[7] Pooja Tiwari, Simran Mehta, Nishtha Sakhuja, Jitendra Kumar, and Ashutosh Kumar Singh. Credit card fraud detection using machine learning: A study. *arXiv preprint arXiv:2108.10005*, 2021.