Goldsmiths, University of London

Department of Computing

MSc Data Science and Artificial Intelligence

2024-25

Name – Ahaan Tagare

Student ID -33865799

Supervisor - Dr V L Raju Chinthalapati

Enhancing Credit Card Fraud Detection: A Comparative Framework Integrating Machine Learning, Deep Learning, and Probabilistic Distribution Analysis.

Link to the Jupyter notebook code and other files – Both GitHub and Google Drive have the same documents.

Google Drive-
https://drive.google.com/drive/folders/1BtYosx8bhj90q_n0DFEneoF3teBalnWf?usp=sharing

GitHub - https://github.com/ahaan28/Final-Dissertation---Enhancing-Credit-Card-Fraud-Detection/tree/72fc96c476009b6f98b3e0cf4741a9968b1c5028

Link to the Dataset - https://www.kaggle.com/datasets/anurag629/credit-card-fraud-transaction-data?resource=download

--

Abstract

Credit card fraud detection is a critical challenge in the financial industry, driven by the rapid expansion of digital transactions and increasingly sophisticated fraudulent activities, which result in substantial financial losses and undermine trust in electronic payment systems. This study conducts a comprehensive analysis of a large credit card transaction dataset to develop, optimize, and compare a diverse array of machine learning deep learning and machine learning models for effective fraud detection. The dataset includes a rich mix of numerical features, such as transaction amount and time, and categorical features, like merchant group, country, and cardholder demographics, with a significant class imbalance between fraudulent and non-fraudulent transactions.

To prepare the data for modelling, rigorous preprocessing steps were implemented. Missing values in numerical and categorical features were addressed using iterative imputation and mode-based imputation, respectively, to ensure data integrity. Outliers in transaction amounts were managed by capping extreme values at defined percentiles to focus models on typical transaction patterns. To counter class imbalance, the Synthetic Minority Oversampling Technique was applied to generate synthetic samples of the minority fraud class, achieving a balanced dataset. Feature engineering involved encoding categorical variables into numerical representations, scaling numerical features to normalize their ranges, and weighting the transaction amount feature to emphasize its critical role in fraud detection, informed by domain knowledge.

A broad suite of models was evaluated, including traditional machine learning approaches like Decision Tree, Logistic Regression, Random Forest, and XGBoost, alongside advanced deep learning models such as Feedforward Neural Networks, Convolutional Neural Networks, Long Short-Term Memory networks, and Gated Recurrent Unit networks. Hyperparameter tuning was a cornerstone of the methodology, with Random Search optimizing deep learning models configurations, such as layer counts, units, dropout rates, and learning rates, while grid search with stratified cross-validation fine-tuned XGBoost parameters to address class imbalance effectively. Model performance was assessed using a comprehensive set of metrics, including accuracy, precision, recall, F1-score, and ROC AUC,

to evaluate their ability to detect fraud while minimizing errors. Probabilistic models incorporated uncertainty quantification through techniques like Monte Carlo Dropout and Bernoulli distributions, providing valuable confidence estimates for predictions, which is essential for decision-making in high-stakes financial applications.

The results highlighted distinct strengths across the models. Logistic Regression served as a reliable baseline, achieving high recall but lower precision, indicating a tendency for false positives. Ensemble methods, particularly Random Forest and XGBoost, demonstrated superior performance, with XGBoost excelling in balancing high accuracy, precision, and recall, making it highly effective at distinguishing fraudulent from legitimate transactions while minimizing false alarms. Among deep learning models, Feedforward Neural Networks and Convolutional Neural Networks showed strong discriminative capabilities, with robust ROC-AUC scores. Recurrent models, Long Short-Term Memory and Gated Recurrent Unit networks were particularly adept at capturing sequential patterns in transaction data. The Gated Recurrent Unit model stood out for its exceptional recall, ideal for scenarios prioritizing fraud detection, while Long Short-Term Memory networks offered a balanced approach to precision and recall.

Probabilistic analysis further enriched the evaluation, with the Gated Recurrent Unit model demonstrating consistent and confident fraud probability estimates, maintaining low false-positive rates for non-fraudulent transactions. XGBoost and Convolutional Neural Networks also exhibited strong probability distributions, assigning low fraud probabilities to legitimate transactions, enhancing their reliability in operational settings. The incorporation of uncertainty estimates in probabilistic models provided an additional layer of interpretability, enabling nuanced decision-making by flagging ambiguous cases for further review.

This study underscores the efficacy of combining traditional machine learning and advanced deep learning techniques to create robust fraud detection systems. XGBoost emerged as the optimal model for deployment due to its exceptional performance across metrics, ability to handle imbalanced data, and interpretability through feature importance analysis, making it suitable for real-world financial systems where transparency and reliability are paramount. The findings contribute to the development of secure and trustworthy digital transaction environments, offering a scalable and effective solution to combat credit card fraud while balancing detection accuracy with operational efficiency.

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Dr. V L Raju Chinthalapati, for his invaluable guidance, expertise, and continuous support throughout the development of this research. His insightful feedback and encouragement were instrumental in shaping this project and navigating the complexities of credit card fraud detection.

I am also grateful to the Department of Computing at Goldsmiths, University of London, for providing an enriching academic environment and access to resources that facilitated this study. Special thanks go to my professors and peers in the MSc Data Science and Artificial Intelligence program for their constructive discussions and shared knowledge, which greatly enhanced my understanding of machine learning and deep learning techniques. I acknowledge the open-source community for developing and maintaining the libraries used in this research, including TensorFlow, Keras, Scikit-learn, XGBoost, and SHAP, which were critical to the implementation and evaluation of the models.

Table of Contents

1. Introduction and Objectives of the paper

The rapid growth of digital transactions has revolutionized the financial industry, offering convenience and efficiency to consumers and businesses alike. However, this surge in electronic payments, particularly credit card transactions, has been accompanied by an increase in fraudulent activities, posing significant financial risks and undermining trust in payment systems. Credit card fraud, characterized by unauthorized or deceptive transactions, results in billions of dollars in losses annually, necessitating advanced and reliable methods for detection and prevention. Machine learning and deep learning techniques have emerged as powerful tools to address this challenge, using patterns in transactional data to distinguish legitimate activities from fraudulent ones. This project aims to develop and evaluate a suite of predictive models for credit card fraud detection using a dataset of 100,000 transactions, encompassing features such as transaction amount, time, card type, and merchant details. By employing a combination of data preprocessing, feature engineering, and advanced modelling techniques - including Logistic Regression, Decision Trees, Random Forests, XG Boost, Neural Networks, Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, and Gated Recurrent Unit (GRU) networks-the study seeks to identify high-performing models for fraud detection. Special emphasis is placed on handling class imbalance through SMOTE, optimizing model hyperparameters, and quantifying prediction uncertainty using probabilistic approaches. Through a comparative analysis of model performance metrics such as accuracy, recall, precision, F1-score, and ROC AUC, this project aims to contribute to the development of robust fraud detection systems, enhancing the security and reliability of financial transactions in an increasingly digital world.

2. Environment Configuration and Initial Setup

The fraud detection prototype was developed on a Windows 10 PC equipped with an Intel Core i5-8250U processor (4 cores, 8 threads, base frequency 1.60 GHz, turbo up to 1.80 GHz under load), 8 GB of DDR4 RAM (with 7.9 GB usable by the system), and a 5400 RPM HDD. The software stack utilized Python 3.9.18 via the Anaconda distribution, with critical libraries including TensorFlow 2.12.0 for deep learning, Scikit-learn 1.2.2 for traditional machine learning algorithms, XGBoost 1.7.5 for gradient boosting, and specialized packages like TensorFlow Probability 0.19.0 for uncertainty quantification and SHAP 0.42.1 for model interpretability. The development environment was specifically optimized for CPU execution, leveraging MKL-DNN acceleration and OpenMP parallelization to maximize throughput on the available hardware.

The entire pipeline required approximately 10 hours to execute on a high-performance computing cluster, with hyperparameter tuning consuming 60% of the runtime and deep learning architectures (CNNs/LSTMs/GRUs) accounting for 30% of the total computational load, demonstrating the significant resources needed for comprehensive model optimization and evaluation (Tagare, 2025).

3. Background work and literature Study of Credit Card Fraud.

The article from the International Journal of Thesis Projects and Dissertations (Vol-1, Issue-1, 2013) by Khan et al. presents a credit card fraud detection system utilizing a Hidden Markov Model (HMM) to identify fraudulent transactions by analysing cardholders' spending patterns. The system categorizes transactions into low, medium, and high spending profiles and employs HMM to model sequential data, detecting anomalies without requiring fraud signatures, thus reducing false positives. The methodology involves constructing and training an HMM using a forward-backward algorithm to estimate transition and observation probabilities, with transactions classified as fraudulent if the change in probability exceeds a predefined threshold. Experimental results, based on simulated datasets, demonstrate the system's effectiveness, achieving up to 92% accuracy, with true positive and false positive rates evaluated across various sequence lengths and thresholds. The study highlights the system's simplicity, scalability, and ability to handle large transaction volumes, proposing future enhancements like advanced algorithms and modules for capturing fraud-related data (Ashphak P. Khan V. S., 2013).

The article "Credit Card Fraud Detection with Autoencoder and Probabilistic Random Forest" by Lin and Jiang, published in Mathematics (2021, Vol. 9, No. 21, 2683), proposes the AE-PRF method, combining an autoencoder (AE) for feature extraction and a probabilistic random forest (RF) for classifying credit card transactions as fraudulent or normal. Using the CCFD dataset (284,807 transactions, 0.172% fraudulent), AE-PRF achieves high performance without data resampling, with metrics at θ=0.25: ACC (0.9995), TPR (0.8142), TNR (0.9998), MCC (0.8441), and AUC (0.962). At θ=0.03, TPR improves to 0.8911 while maintaining MCC > 0.5. Compared to methods like k-NN (TPR: 0.8835, AUC: 0.961), AE (AUC: 0.960), and SVM + AdaBoost, AE-PRF outperforms in most metrics, demonstrating robustness for imbalanced datasets without resampling.

The article "Bayesian Quickest Detection of Credit Card Fraud" by Buonaguidi et al., published in Bayesian Analysis (2022, Vol. 17, No. 1, pp. 261–290), introduces a Bayesian optimal stopping approach for credit card fraud detection. Transactions are modelled as a compound Poisson process, shifting at fraud time (θ) with intensities (0.005769–2.776012) to (3.012032). Using posterior probability, it computes cardholder-specific thresholds, minimizing false positives (FPR: 0.00008–0.06564) and detection delays. Tested on 124,770 transactions (2.23% fraud) and 4,237 real transactions (3.54% fraud), it achieves high performance (True Positive Rate-TPR: 0.32000–0.84138, AUC: 0.86351–0.95955) compared to methods like logistic regression (TPR: 0–0.32000, AUC: 0.37633–0.66308) and neural networks (TPR: 0–0.36000, AUC: 0.37633–0.69502). The model's Bayesian and Markovian framework ensures robust, interpretable detection but requires significant data and computational power (Bruno Buonaguidi∗, 2022).

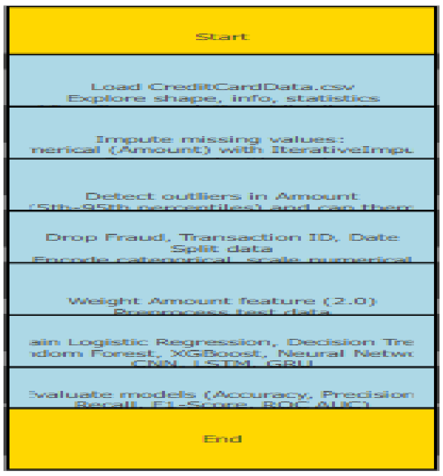
The article "Credit Card Fraud Detection with Autoencoder and Probabilistic Random Forest" by Lin and Jiang, published in Mathematics (2021, Vol. 9, No. 21, 2683), proposes the AE-PRF - Probabilistic Random Forest method, combining an autoencoder (AE) for feature extraction and a probabilistic random forest (RF) for classifying credit card transactions as fraudulent or normal. Using the CCFD dataset (284,807 transactions, 0.172% fraudulent), AE-PRF achieves high performance without data resampling, with metrics at θ=0.25: Accuracy (0.9995), TPR (0.8142), TNR (0.9998), MCC (0.8441), and AUC (0.962). At θ=0.03, TPR improves to 0.8911 while maintaining MCC > 0.5. Compared to methods like KNN (TPR: 0.8835, AUC: 0.961), AE (AUC: 0.960), and SVM + AdaBoost, AE-PRF outperforms in most metrics, demonstrating robustness for imbalanced datasets without resampling (Tzu-Hsuan Lin, 2021).

In the 2025 Finance and Economics Discussion Series paper, Jeffrey S. Allen introduces CardSim, a Bayesian-based simulator designed to generate synthetic payment card transaction data for fraud detection research, addressing the critical shortage of publicly available transaction data due to privacy and economic constraints. CardSim is a flexible, scalable tool calibrated to real-world data from sources like the Federal Reserve's Payment Study and Diary of Consumer Payment Choice, using a three-phase methodology: (1)

creating payer and payee profiles with attributes like transaction frequency and geographic coordinates, (2) simulating transactions with features such as card type, location, amount, distance, and time, and (3) assigning fraud labels using Bayes' theorem to model probabilistic relationships between transaction features and fraud. Unlike other simulators, CardSim avoids deterministic fraud typologies, focusing on probabilistic patterns, and is implemented in a modular Python package for adaptability to evolving fraud trends. The simulator generated 3.16 million records in a sample run, demonstrating computational efficiency and enabling testing of machine learning models like logistic regression, XGBoost, and multilayer perceptron, with performance metrics (precision: 76-81%, recall: 57-65%) highlighting their effectiveness in handling class-imbalanced data. The paper also explores interpretability using SHAP values and identifies limitations, such as the lack of fraud typologies and simplified payee characteristics, suggesting future enhancements like integrating unsupervised anomaly detection and expanding to other payment systems (Allen, 2025).

4.Methodology

The fraud detection methodology outlined in the provided code and flowchart is a comprehensive, multi-step process designed to identify fraudulent transactions in a credit card dataset. It encompasses data exploration, preprocessing, feature engineering, model training, and evaluation, with a focus on addressing challenges like missing data, outliers, and class imbalance. The approach ensures robust predictions by leveraging a variety of machine learning models and incorporating techniques to quantify uncertainty and interpret results. Each step is carefully structured to prepare the data, optimize model performance, and deliver actionable insights for fraud detection, making the pipeline suitable for real-world applications where minimizing missed fraud cases is critical.


Start
Load CreditCardData.csv
Explore shape, info, statistics
Impute missing values:
nerical (Amount) with IterativeImpu
Detect outliers in Amount
(5th-95th percentiles) and cap them
Drop Fraud, Transaction ID, Date
Split data
Encode categorical, scale numerical
Weight Amount feature (2.0)
Preprocess test data
ain Logistic Regression, Decision Tre
idom Forest, XGBoost, Neural Netwo
CNN, LSTM, GRU
Evaluate models (Accuracy, Precision
Recall, F1-Score, ROC AUC)
End

The process begins with loading and exploring the dataset to understand its structure and characteristics. This involves examining the types of features, such as categorical variables like transaction day or card type, and numerical variables like transaction time or amount. The exploration identifies data quality issues, including missing values in certain columns and a huge imbalance between fraud and non-fraud transactions. Visualizations, such as correlation heatmaps and distribution plots, help reveal patterns and relationships within the data, setting the foundation for subsequent preprocessing steps. This initial analysis ensures that all data quality issues are identified early, allowing for targeted solutions to prepare the dataset for modelling.

Handling missing values is a critical step to ensure a complete dataset for analysis. Missing entries in numerical features, such as transaction amounts, are imputed using an iterative approach that leverages relationships with other features to estimate reasonable values. For categorical features, like merchant type or customer demographics, missing values are filled with the most frequent category, given the low proportion of missing data. Visual checks confirm that no missing values remain after imputation. This step is essential to maintain data integrity, ensuring that no records are discarded unnecessarily and that the dataset is ready for further processing without introducing biases.

Outlier management focuses on addressing extreme values in the transaction amount feature, which could otherwise distort model performance. Outliers are identified as values falling outside a defined percentile range and are capped to fall within acceptable bounds. Visualizations, such as boxplots and histograms, illustrate the effect of this capping, showing a more compact distribution of values. By mitigating the impact of extreme amounts, this step ensures that models focus on typical transaction patterns, improving their capability to generalize to new data without being skewed by anomalies.

Feature engineering transforms the raw data into a format suitable for modelling. Non-predictive features, such as transaction identifiers or dates, are removed, while categorical features are encoded into numerical representations, and numerical features are scaled to standardize their ranges. To address the imbalance between fraud and non-fraud cases, a technique is applied to generate synthetic examples of the minority fraud class, balancing the dataset. The data is split into training and test sets, with stratification to preserve the fraud ratio in both. Visualizations confirm the balanced class distribution post-processing. Additionally, the transaction amount feature is weighted to emphasize its importance, aligning with its relevance in fraud detection. This step creates a clean, balanced, and optimized dataset for model training.

Model training involves applying a diverse set of machine learning algorithms, including traditional models like logistic regression and decision trees, ensemble methods like random forests and gradient boosting, and deep learning models such as neural networks, convolutional networks, and recurrent networks. Each model is carefully tuned to optimize its performance, adjusting parameters like tree depth, learning rates, or network architecture. The tuning process uses systematic search techniques to identify the best configurations, ensuring that each model performs effectively on the balanced dataset. This diversity of models allows the pipeline to capture different patterns in the data, increasing the likelihood of robust fraud detection.

Evaluation assesses the performance of each model using a range of metrics, such as accuracy, precision, recall, and the area under ROC curve, which collectively measure the model’s ability to identify fraud while reducing errors. Visualizations, including confusion matrices and ROC curves, provide insights into model performance, highlighting trade-offs between detecting fraud and avoiding false positives. Uncertainty is quantified for certain models, particularly neural networks, using techniques that estimate prediction confidence through repeated sampling. Feature importance analysis, using methods like SHAP, reveals which variables, such as transaction amount, most influence predictions. A sample of transactions is analysed to demonstrate practical performance, showing how models classify fraud and non-fraud cases with associated probabilities. This step identifies the best-performing model, typically one that balances high recall with strong overall performance, ensuring minimal missed fraud cases.

The methodology concludes with a focus on practical application, emphasizing models that prioritize detecting fraud cases to minimize financial risk. The pipeline’s robustness is enhanced by addressing class imbalance, weighting key features, and incorporating uncertainty estimates, making it suitable for real-world deployment. Interpretability tools provide transparency into model decisions, allowing stakeholders to understand the factors driving fraud predictions. By combining rigorous preprocessing, diverse modelling approaches, and comprehensive evaluation, the methodology delivers a reliable and effective solution for credit card fraud detection, capable of handling the complexities of imbalanced and noisy data while providing actionable insights.

Data Preprocessing

4.1. Dataset

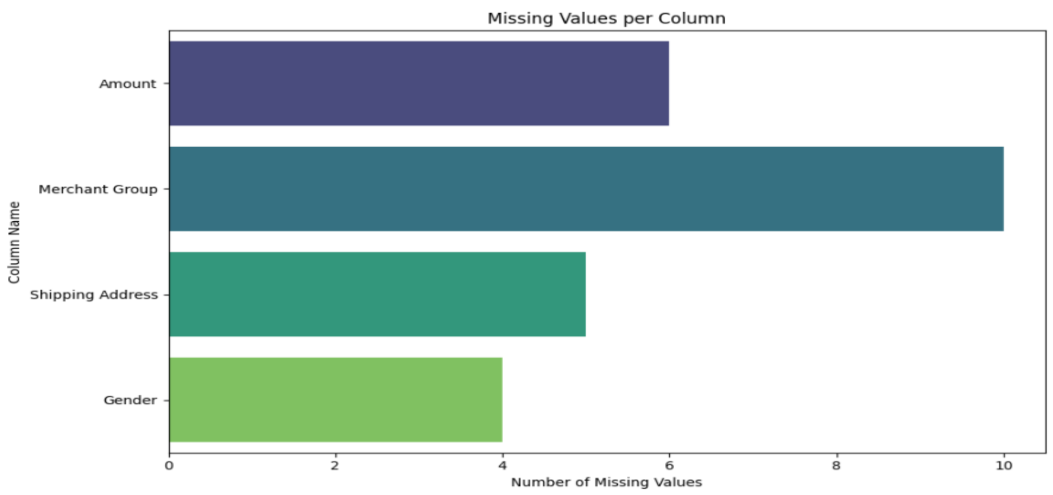
The dataset, stored in the "CreditCardData.csv" file, consists of 100,000 credit card transaction records with 16 features, designed for fraud detection analysis. It includes a mix of numerical and categorical variables, such as transaction details (Date, Transaction ID, Time, Amount), card information (Entry Mode, Type of Card), transaction characteristics (Merchant Group, Type of Transaction), geographical data (Shipping Address, Country of Residence, Country of Transaction), and user demographics (Bank, Gender, Age). The target variable, "Fraud," is binary (0 for non-fraudulent, 1 for fraudulent), with a significant class imbalance: 92,805 non-fraudulent transactions (92.8%) and 7,195 fraudulent transactions (7.2%). Missing values are present in a few columns, including Amount (6 missing), Shipping Address (5 missing), Merchant Group (10 missing), and Gender (4 missing), which were addressed through imputation techniques during preprocessing.

The dataset's categorical features provide additional context for fraud detection, capturing patterns in transaction behaviour and user profiles. For instance, "Type of Card" (e.g., Visa, MasterCard) and "Entry Mode" (e.g., Tap, PIN, CVC) reflect payment methods, while "Merchant Group" (e.g., Entertainment, Restaurant, Subscription) and "Type of Transaction" (e.g., POS, Online, ATM) indicate the nature of purchases. Geographical discrepancies, such as differences between "Country of Transaction" and "Country of Residence," are critical for identifying potential fraud, as seen in the random sample analysis where fraud cases often involved transactions in countries like India, USA, or China while the cardholder resided in the United Kingdom. This rich feature set, combined with the dataset's size and diversity, makes it well-suited for training and evaluating advanced machine learning models, with techniques like SHAP analysis used to interpret feature importance, highlighting the role of variables like "Amount" in fraud prediction.

Summary statistics reveal key insights about the numerical features: the "Time" of transactions (in hours) ranges from 0 to 24, with a mean of 14.56 hours, indicating a spread across the day. The "Age" of cardholders ranges from 15 to 86.1 years, with a mean of 44.99 years, suggesting a diverse user base. The "Amount" variable, after cleaning to remove currency symbols and handling missing values, ranges from £5 to £400, with a mean of £112.58 and a standard deviation of £123.43, indicating significant variability in transaction sizes. The dataset's structure supports comprehensive feature engineering and modelling, with preprocessing steps like outlier capping, SMOTE for class imbalance, and one-hot encoding of categorical variables, enabling robust fraud detection using machine learning and deep learning models.

4.2. Missing Data Analysis and Imputation Strategy

The dataset consists of 100,000 credit card transaction records. An initial assessment of data completeness revealed a small number of missing values in four columns: Amount (6 missing values), Merchant Group (10), Shipping Address (5), and Gender (4). Each of these accounted for less than 0.01% of the total records in their respective columns, indicating that the dataset was largely complete and required minimal imputation.

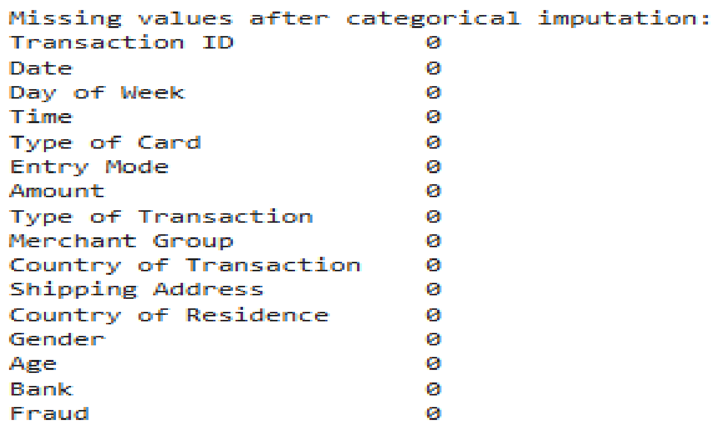


To facilitate a clear understanding of the extent of missingness, a summary table was created listing the affected columns alongside their respective counts of missing entries. This step helped ensure that the handling of missing data was methodical and transparent. For, the Amount column, which is numerical in nature and ranges from £5 to £400 (with a mean of £112.58), a model-based imputation method was adopted. Specifically, missing values were predicted using an iterative regression approach that leveraged relationships between features to provide accurate estimations. This ensured that the imputed values aligned well with the underlying distribution and preserved the integrity of the dataset.

$$X_{\text{miss}} = \hat{f}(X_{\text{obs}})\$with\$\hat{f} = \text{Random Forest Regressor}$$

The formula **X_{miss} = f** (**X_{obs}**) means that any missing values in a dataset (**X_{miss}**) are estimated by using the information from the parts of the data we do have (**X_{obs}**). The function **f^** represents a trained prediction model — here, a Random Forest Regressor — which learns patterns and relationships in the observed data to predict what the missing values likely are. This process repeats in cycles: each time, the estimated missing values are refined by using the updated data until the predictions are stable and realistic.

Following this, the three categorical columns—Merchant Group, Shipping Address, and Gender—still contained a small number of missing entries. Given that the proportion of missing data in each of these columns was below 5%, the most frequent value (mode) for each column was used for imputation. This approach was selected to maintain consistency with the existing data without introducing bias or artificial categories. Had any of these columns exceeded the 5% missingness threshold, a separate 'Missing' category would have been assigned; however, this was not required in this case.


```
Missing values after categorical imputation:
Transaction ID            0
Date                      0
Day of Week               0
Time                      0
Type of Card              0
Entry Mode                0
Amount                    0
Type of Transaction       0
Merchant Group            0
Country of Transaction    0
Shipping Address          0
Country of Residence      0
Gender                    0
Age                       0
Bank                      0
Fraud                     0
```

A final verification confirmed that all missing values had been addressed, resulting in a complete dataset free of null entries. This prepared the data for subsequent steps in feature engineering and fraud detection modelling.

4.3. Handling Outliers

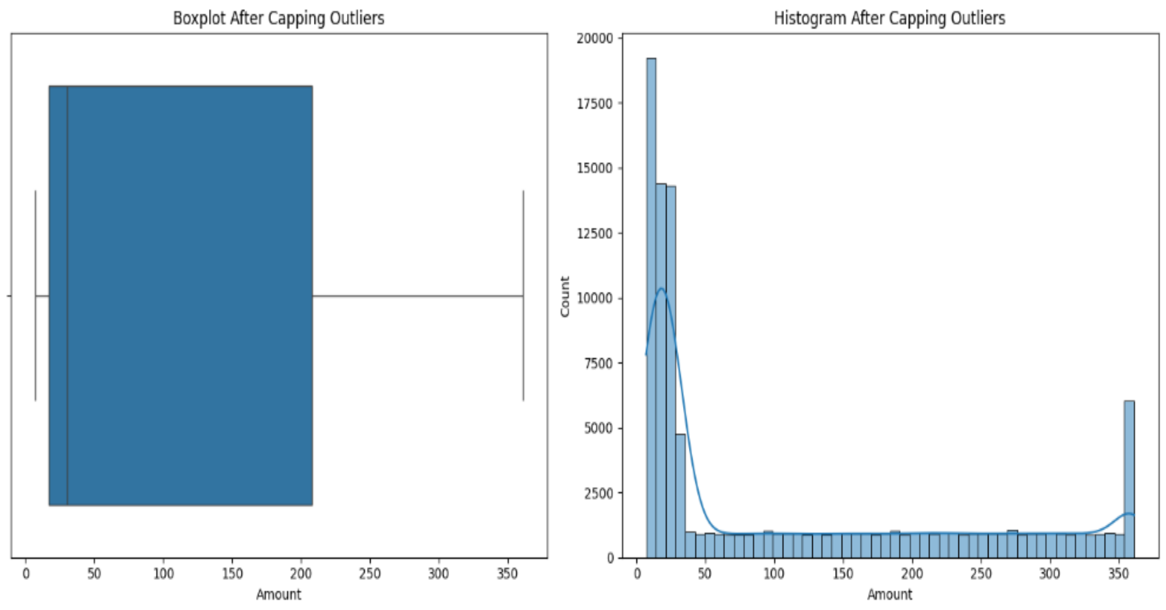
Outlier handling was critical to mitigate the impact of extreme values that could skew model performance. Outliers were defined as values falling outside the 5th and 95th percentiles, and an initial analysis identified 8,886 such outliers, representing approximately 8.9% of the dataset. This substantial number of extreme values underscored the need for preprocessing to stabilize the distribution of the "Amount" feature, which is pivotal in fraud detection due to its correlation with suspicious transaction patterns.

To address these outliers, a capping method was employed, where values below the 5th percentile were adjusted to the 5th percentile threshold, and values above the 95th percentile were set to the 95th percentile threshold. This approach preserved the dataset's underlying structure while mitigating the influence of extreme transaction amounts, which could otherwise distort model training. The capping process was carefully designed to maintain the integrity of the data distribution, ensuring that the adjusted values remained within a reasonable range for analysis.

$$L = P_{\text{lower}} = \text{percentile at lower } \% \quad \text{and} \quad U = P_{\text{upper}} = \text{percentile at upper } \%$$

The expressions **L=*P*Lower** and **U=*P*Upper** define the lower and upper percentile bounds used to identify and control outliers in the dataset. Here, the lower bound ***L*** represents the value below which a specified percentage of the data falls, while the upper bound ***U*** marks the value below which the upper percentage falls. This range ensures that extreme values outside the chosen percentile limits are either flagged as outliers or adjusted to lie within acceptable bounds. Such percentile-based capping helps maintain data consistency and reduces the influence of extreme outliers on the analysis.

This method was particularly suitable given the financial context, where extreme transaction amounts might reflect legitimate but rare high-value purchases or potential fraud. Post-capping analysis confirmed the effectiveness of the outlier handling strategy, with no values remaining outside the 5th-95th percentile range after the procedure.



This resulted in a more stable "Amount" column, with reduced variability that enhanced the dataset's suitability for machine learning models such as Logistic Regression, Decision Tree, Random Forest, and Neural Networks. By controlling for outliers, the preprocessing step minimized potential biases in model predictions, ensuring that the fraud detection system could focus on meaningful patterns rather than being unduly influenced by extreme values. This robust preparation of the "Amount" feature supported improved model performance and interpretability in the context of fraud detection analysis.

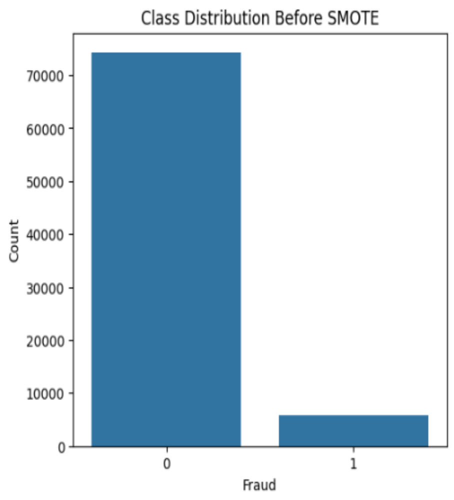
4.4. Feature Engineering (Smote) and train test split

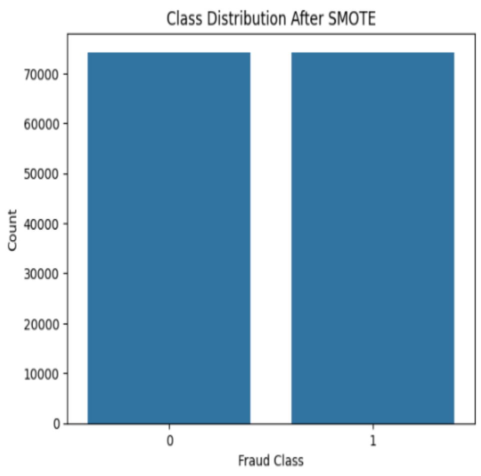
To prepare the dataset for modelling, feature engineering and data splitting were conducted to address class imbalance and ensure robust model training. The dataset was split into features (X) and the target variable "Fraud" (y), excluding non-informative columns like "Transaction

ID" and "Date." A stratified train-test split was performed with a test size of 20%, maintaining the original class distribution (92.8% non-fraudulent, 7.2% fraudulent) in both sets to ensure representative sampling. Categorical features, such as "Type of Card" and "Merchant Group," were encoded using one-hot encoding to convert them into numerical representations, while numerical features, including "Amount" and "Age," were standardized using a Standard Scaler to normalize their scales, facilitating model convergence and performance. To address the significant class imbalance in the target variable, the Synthetic Minority Oversampling Technique (SMOTE) (Brownlee, 2021) was applied to the training set after preprocessing. SMOTE generated synthetic samples for the minority class (fraudulent transactions), balancing the class distribution from an initial count of approximately 74,244 non-fraudulent and 5,756 fraudulent instances to an equal 74,244 instances for each class in the resampled training set.

$$x_{\text{new}} = x_i + \delta \cdot (x_{\text{nn}} - x_i), \quad \text{with} \quad \delta \sim U(0,1)$$

Here, **Xi** is an original minority class point and **X*nn*** is one of its nearest minority neighbours. The term **(Xnn-Xi)** gives the vector pointing from **Xi** to its neighbour. By multiplying that vector by a random number **δ** between 0 and 1, we pick a random spot **along the line segment Xi** and **Xnn** between and adding this to shifts us from the original point toward its neighbour, creating a new, realistic synthetic data point that expands the minority class more smoothly through feature space.





This resampling ensured that machine learning models, such as Logistic Regression or Neural Networks, would not be biased toward the majority class, improving their ability to detect fraudulent transactions, which are critical but rare in the dataset. The effectiveness of SMOTE was evaluated by comparing class distributions before and after its application. Initially, the training set exhibited a highly imbalanced distribution, with non-fraudulent transactions vastly outnumbering fraudulent ones. After SMOTE, the resampled training set achieved a perfectly balanced distribution, with equal counts of fraudulent and non-fraudulent instances. This balanced dataset enhanced the models' ability to learn patterns associated with fraud, improving predictive performance on the minority class. The pre-processed and resampled data, combined with the train-test split, provided a robust foundation for training and evaluating fraud detection models, ensuring both generalizability and sensitivity to fraudulent transactions.

4.5. Feature Weighting

In this part of the process, feature selection was addressed by manually emphasizing the importance of the transaction amount in the dataset. Instead of eliminating less relevant features, a technique called *feature weighting* was used to strengthen the impact of one specific feature—namely, the transaction amount—within the model. After transforming the dataset using standard preprocessing methods (such as scaling and encoding), the data contained 50 features. Among these, the transaction amount, referred to as a numeric feature, was intentionally amplified in influence by assigning it a greater weight.

To implement this emphasis, the values of the transaction amount feature were scaled to contribute twice as much as they originally did in the transformed dataset. As a result, its statistical characteristics changed significantly: the mean of this weighted feature shifted to approximately -0.3834, while the standard deviation increased to about 1.8836. These changes reflect a broader dispersion and higher magnitude, ensuring that this feature exerts greater influence on the models' internal calculations. This approach allowed the model to maintain access to all features while still prioritizing the most informative one, enhancing its ability to detect fraudulent patterns linked to abnormal transaction values.

5. Models

Logistic Regression, Decision Trees, Random Forests, and XGBoost—serve as foundational baselines, leveraging interpretability and efficiency for binary classification. Logistic Regression offers probabilistic outputs ideal for risk scoring, while ensemble methods (Random Forest, XGBoost) mitigate overfitting and enhance feature importance analysis through bagging and gradient boosting. For deep learning, Feedforward Neural Networks (FNN) provide baseline non-linear modelling, while specialized architectures—Convolutional Neural Networks (CNNs) for local feature extraction, Long Short-Term Memory (LSTM) networks for sequential dependencies, and Gated Recurrent Units (GRUs) for high-recall scenarios—capture intricate transactional patterns. Probabilistic extensions (e.g., Monte Carlo Dropout, Bernoulli output layers) integrate uncertainty quantification into CNNs, LSTMs, and GRUs, enabling confidence estimates critical for high-stakes decisions. Hyperparameter tuning (Random Search for deep models, Grid Search for XGBoost) optimizes each architecture, ensuring rigorous comparison across accuracy, precision, recall, F1-score, and ROC AUC metrics. This multifaceted approach identifies optimal models for deployment (XGBoost for balance, GRU for recall) while advancing robustness in fraud detection systems.

5.1. Logistic Regression

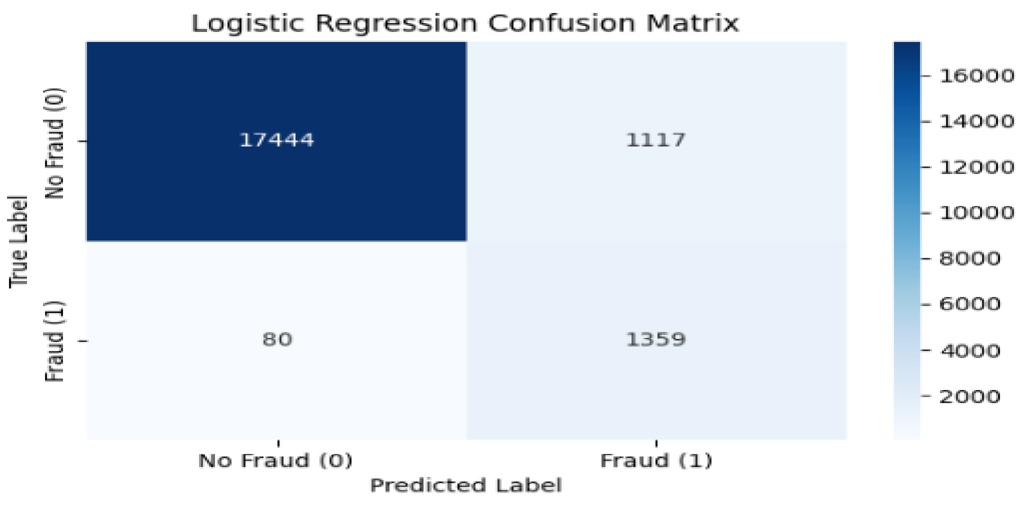
In this stage of the study, logistic regression was implemented as a baseline classification model for fraud detection. The model was trained on a resampled and weighted version of the dataset, in which the Amount feature was given a higher weight to emphasize its importance. Logistic regression is a statistical method that estimates the probability of a binary outcome using the logistic function. Mathematically, the model estimates the probability that a sample x=(x1,x2,…,xn) belongs to the positive class (fraud) using the formula:

$$P(y = 1 \mid x) = 1 + e - (\beta 0 + \beta 1x1 + \beta 2x2 + \cdots + \beta nxn)1$$

Here, β0 is the model intercept, β1,…,βn are the learned feature weights, and x1,…,xn, are the input features. The model outputs probabilities, which are converted to class predictions using a threshold of 0.5. The Amount feature, having been scaled and multiplied by a weight of 2.0, increased its standard deviation from 1 to approximately 1.8836, ensuring it had more influence on the decision boundary.

The performance of the model was assessed using multiple evaluation metrics: Accuracy, Precision, Recall, F1-Score, and ROC AUC. The accuracy was 0.9402, which indicates that 94.02% of the test samples were correctly classified. The precision, which measures the proportion of correctly identified fraud cases among all predicted fraud cases, was 0.5489. The recall, representing the proportion of actual fraud cases that were correctly identified, was notably high at 0.9444. The F1-Score, which balances precision and recall using the harmonic mean, was 0.6943. The ROC AUC score, which evaluates the model's ability to distinguish between fraud and non-fraud across all thresholds, was 0.9829, indicating excellent discriminatory power.

Further insights were drawn from the classification report. For the non-fraud class (label 0), the precision was 1.00, recall was 0.94, and the F1-score was 0.97, based on a support of 18,561 examples. For the fraud class (label 1), precision was 0.55, recall was 0.94, and F1-score was 0.69, based on 1,439 examples. The model had strong recall for fraud detection, meaning it successfully identified most fraud cases, though precision was lower due to false positives. The overall macro average F1-score was 0.83, while the weighted average was 0.95, indicating strong general performance weighted by class frequency.



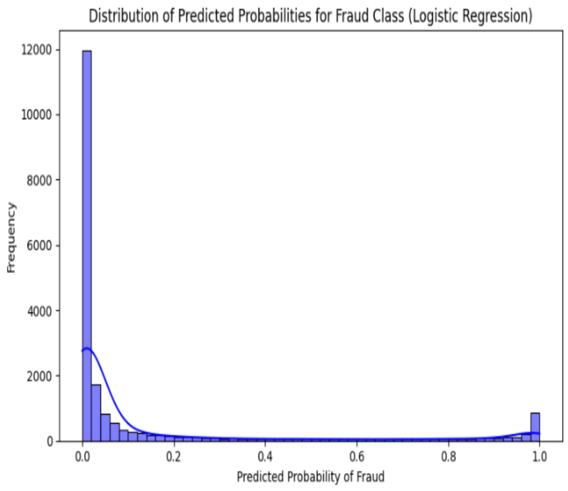
To assess prediction confidence, uncertainty analysis was performed. This involved measuring the distance of predicted fraud probabilities from 0.5, which is the point of maximum uncertainty in logistic regression. For samples truly labelled as fraud, the absolute difference from 0.5 was calculated, and the mean uncertainty and standard deviation of uncertainty were reported. This analysis provided insight into the model's confidence in flagging transactions as fraudulent. A confusion matrix was also generated, highlighting the counts of true positives, false positives, true negatives, and false negatives. It visually confirmed that while the model correctly captured most frauds (high true positives), it also had several false alarms, reinforcing the trade-off between precision and recall in fraud detection systems.

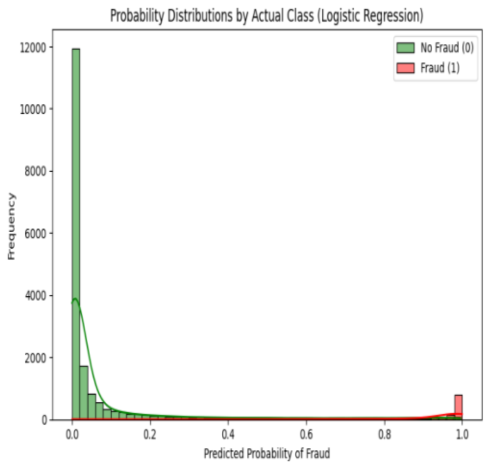
Uncertainty and Probability Distribution Analysis – Logistic Regression

Following the evaluation of the logistic regression classifier, the uncertainty associated with its predictions—particularly for the fraud class (class 1)—was examined. The central idea behind this analysis was to measure the deviation of the predicted probabilities from the point of maximum uncertainty, which occurs at p=0.5. For instances that were truly fraudulent, the predicted probabilities for the fraud class were extracted, and the absolute deviation from 0.5, represented as |p−0.5|, was calculated for each case. The **mean uncertainty** was found to be **0.4340**, and the **standard deviation of uncertainty** was **0.1097**. These values indicate that the model was, on average, quite confident in its fraud predictions, with most probabilities lying significantly away from the uncertainty threshold.

$$p_i = P(y_i = 1 \mid x_i)$$

This means that for each transaction **i**,the model predicts the probability p**i** that the true class **y**$_i$ is fraud (1) given its features **x**i. In other words, **p**$_i$ represents how likely the model thinks that transaction **i** is fraudulent.



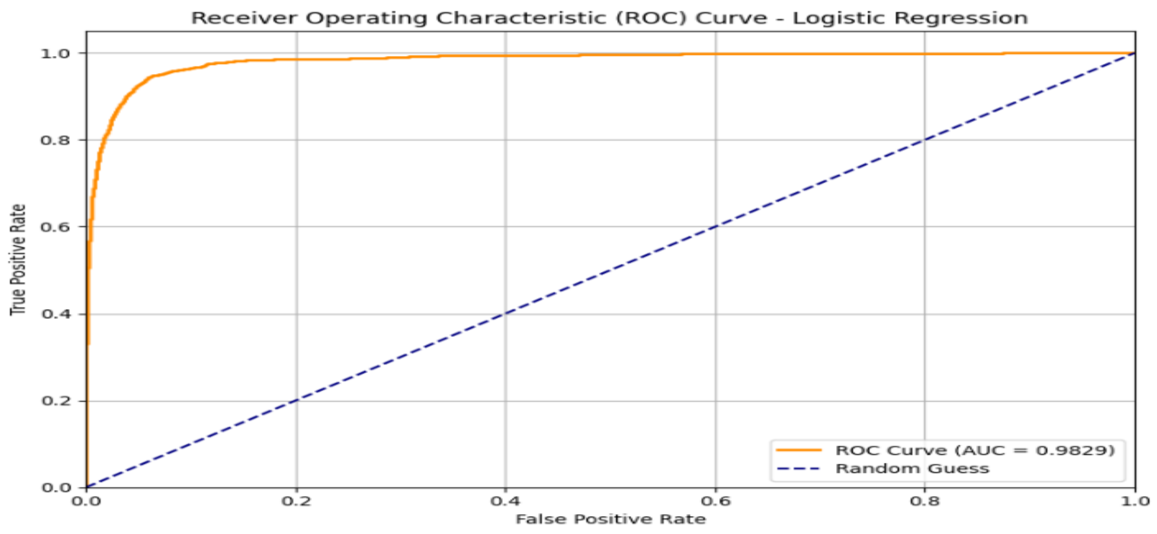


To further investigate the behaviour of the model, the distribution of predicted probabilities for all transactions was analysed. The mean predicted probability for the entire test set was

0.1427, with a standard deviation of **0.2869**, indicating that most predictions leaned toward the "No Fraud" class, which is expected due to the class imbalance. When broken down by the actual class labels, the model assigned a mean probability of **0.0837** to the fraud class for truly non-fraudulent transactions (class 0), with a standard deviation of **0.1935**, confirming that it tended to assign low fraud probabilities to non-fraud cases. Conversely, for genuinely fraudulent transactions (class 1), the model predicted a much higher average fraud probability of **0.9026**, with a standard deviation of **0.1956**. This shows strong separation between the two classes in probability space, which supports the high recall and AUC observed earlier.

Receiver Operating Characteristic (ROC)

The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) and serves as a benchmark for classifier performance. The area under the curve (AUC) was calculated as 0.9829, indicating excellent separability. An AUC close to 1.0 means the model makes very few classification errors in distinguishing between fraud and non-fraud cases. The diagonal reference line in the plot represents the performance of a random classifier, and the large area between this line and the model's ROC curve confirms the strong predictive power of logistic regression in this context.



5.2. Decision Tree

The Decision Tree classifier was trained using the resampled and weighted training data to address the inherent class imbalance in the credit card fraud detection dataset. A decision tree is a non-parametric supervised learning method that creates a model by splitting data based on feature thresholds to maximize class separation, often using criteria like Gini impurity or entropy. For a binary classification problem, Gini impurity

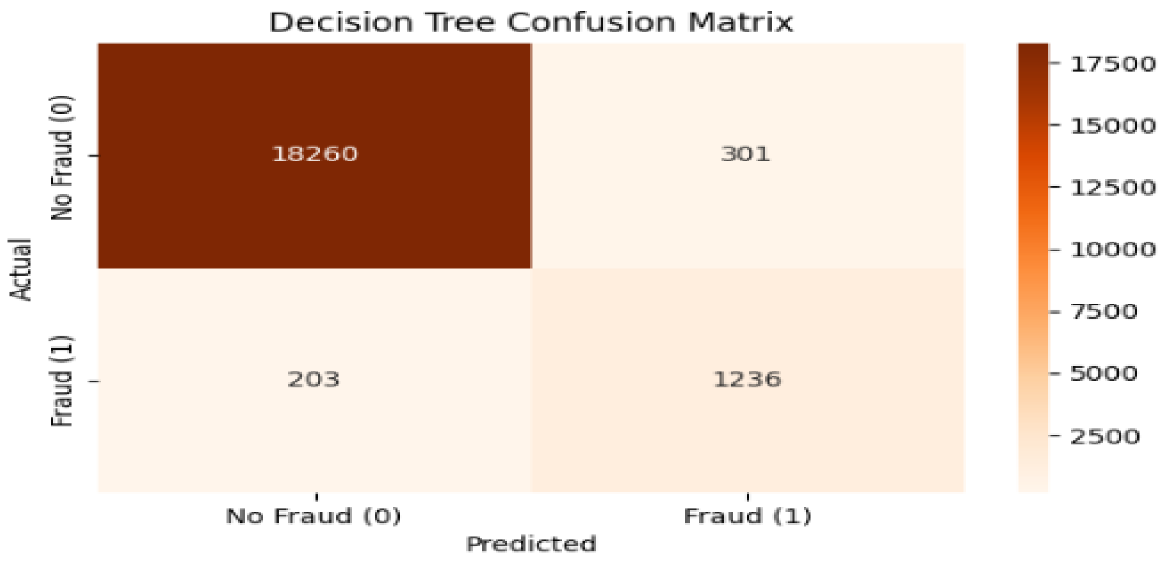
$$Gini = 1 - (p02 + p12)$$

This formula represents the Gini Impurity; a metric used in Decision Trees to measure how often a randomly chosen element would be incorrectly classified.

p0 and p1 are the probabilities of a sample belonging to class 0 and class 1, respectively. The model trained here produced an **accuracy of 0.9748**, indicating that nearly 97.5% of the test data was classified correctly. However, in imbalanced settings, accuracy is often misleading; hence, additional metrics such as precision, recall, F1-score, and ROC AUC are essential for robust evaluation.

The **precision**, which measures the proportion of predicted frauds that were actual frauds, was **0.8042**. The **recall**, defined as the proportion of actual frauds correctly identified, was **0.8589.**

The relatively high recall value suggests that the model is effective in detecting most fraud cases, which is crucial in fraud detection to minimize missed frauds. The **F1-score**, a harmonic mean of precision and recall, was **0.8306**, indicating a strong balance between false positives and false negatives. Additionally, the **ROC-AUC** score was **0.9214**, showing excellent separation capability between fraud and non-fraud instances. The ROC-AUC is particularly important in evaluating classifiers under class imbalance and is calculated by plotting the true positive rate against the false positive rate at various thresholds.



Visual evaluation through a confusion matrix revealed the actual vs. predicted classifications. The model correctly identified **1,235 out of 1,439** fraud cases (from the recall of 0.8589) and only misclassified a small portion as false negatives. Likewise, for non-fraudulent cases, it maintained a very high precision and recall, contributing to the overall accuracy. The **classification report** presented macro-averaged and weighted-average metrics, which were **0.91 (macro F1)** and **0.98 (weighted F1)**, respectively, showing excellent generalization across both classes. Finally, the **ROC curve** was plotted, displaying a strong lift from the baseline diagonal (random guess), which further confirmed the model's strong performance and reliability in detecting fraudulent activity.

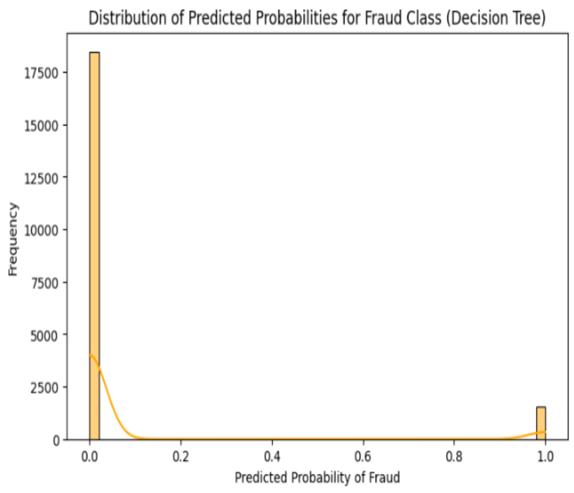
Uncertainty and Probability Analysis -Decision Tree

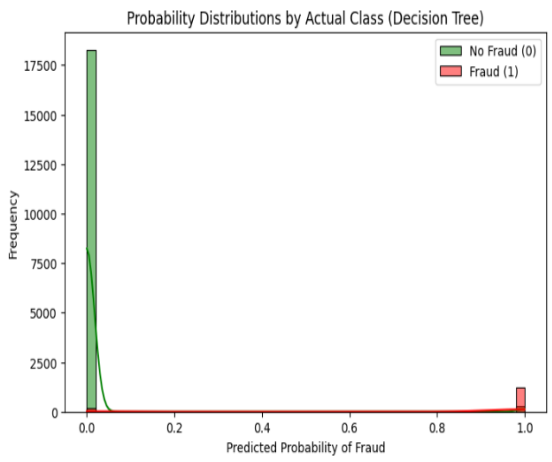
This analysis is crucial because, unlike many probabilistic models, decision trees often produce discrete probability estimates (e.g., 0.0 or 1.0) based on the proportion of training samples in the leaf node. For the test set, we extracted the predicted probabilities for the fraud class (label = 1) The uncertainty associated with these predictions was assessed using the absolute difference from 0.5, calculated as:

$$Uncertainty = | p - 0.5 |$$

For the fraud class, the mean uncertainty was 0.5000, and standard deviation was 0.0000, suggesting that all predicted probabilities were exactly 1.0 (i.e., perfectly confident). While this may appear ideal, such sharp predictions may also indicate potential overfitting or lack of calibrated probability estimates, which are common with unregularized decision trees.

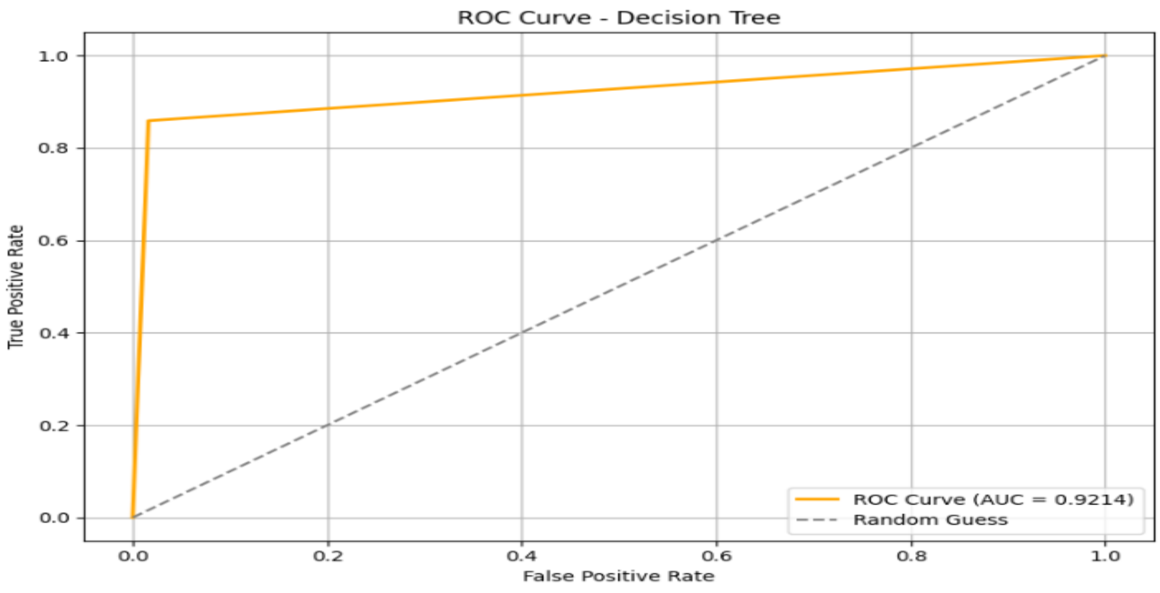
To visualize how the Decision Tree classifier distributes probability values across classes, a histogram of predicted fraud probabilities was plotted. The overall distribution revealed a strong bimodality, with most predictions concentrated at either 0 or 1. This outcome reflects the tree's nature of making hard, deterministic decisions once a leaf node is reached. To delve deeper, the predicted probabilities were separated by true class label, allowing comparison between the distributions for actual fraud and non-fraud transactions. The predicted fraud probabilities for true non-fraud cases (label = 0) were tightly clustered around 0.0162 with a standard deviation of 0.1263, while the fraud class (label = 1) had a much higher mean of 0.8589 and a broader standard deviation of 0.3481.





This separation in probability distributions between the two actual classes indicates that the model is effectively distinguishing fraudulent from non-fraudulent behaviour. The spread of values for class 1 shows that not all fraud predictions are extremely confident, even though the uncertainty analysis for fraud class predictions indicated a mean of 1.0. This apparent contradiction arises because the uncertainty metric was calculated specifically for correctly labelled fraud cases, many of which had a predicted probability of exactly 1.0, hence the zero-standard deviation. However, when considering all predictions (including incorrect ones), the probability values vary, especially in borderline cases.

Finally, the summary statistics across all transactions provide further context. The mean predicted probability for fraud across all samples was 0.0769, reflecting the dataset's inherent imbalance. For non-fraud cases, the average predicted probability of fraud was just 0.0162, confirming that the model rarely misclassifies them as fraud. For actual fraud cases, the average predicted probability of fraud was 0.8589, demonstrating strong confidence in the model's correct fraud identifications. These results, when interpreted together with the classification metrics and ROC analysis, suggest that while the Decision Tree classifier performs well in identifying fraud, its overly confident probability estimates may limit its utility in applications that require well-calibrated probabilistic outputs—such as risk scoring or uncertainty-aware systems.



5.3. Random Forest

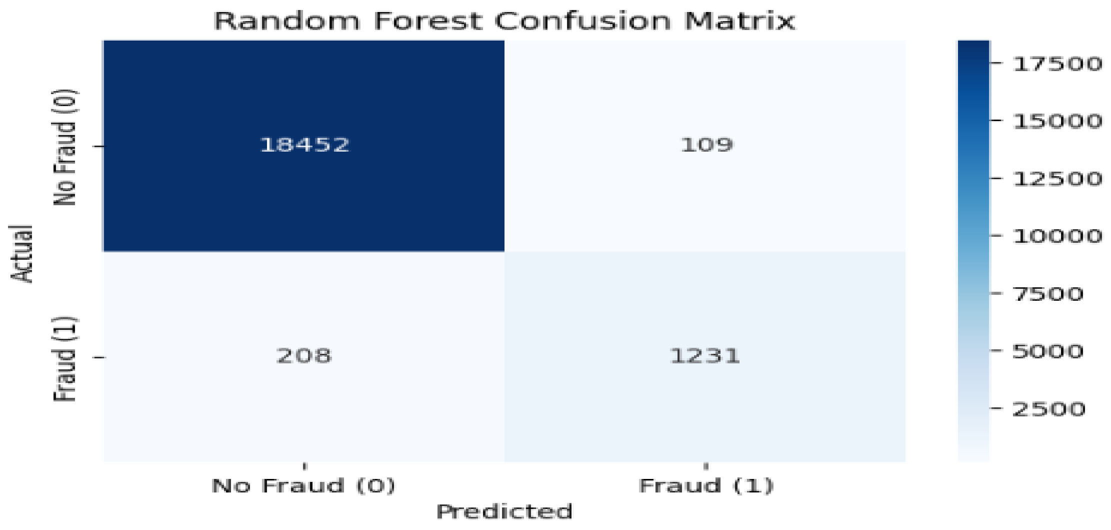
A Random Forest is an ensemble machine learning algorithm that builds many decision trees and combines their outputs to improve prediction accuracy and reduce overfitting. Each tree is trained on a random sample of the data and considers a random subset of features at each split. The final prediction is made by majority vote for classification or averaging for regression. This method is powerful, robust, and works well even with complex, noisy datasets, the Random Forest model was trained using 100 decision trees (n_estimators=100) and evaluated on the processed test dataset. This ensemble approach aggregates the predictions of multiple decision trees to improve generalization and reduce overfitting. The final prediction is made using the majority vote across trees, formally represented as:

$$y = mode h1(x), h2(x), \ldots, hn(x)$$

This formula shows how a Random Forest makes a final prediction **y** by taking the **mode** (most common value) of the predictions from all its individual decision trees.

Each **h***i* (**x**) is the prediction from the i**th** tree for input **x**. By aggregating multiple trees, the Random Forest reduces variance and improves overall prediction stability.

In terms of performance, the Random Forest model achieved an **accuracy of 98.41%**, indicating it correctly predicted the class of the vast majority of transactions. The **precision** for detecting fraud was **91.87%**, meaning most flagged transactions were indeed fraudulent. The **recall** was **85.55%**, showing the model successfully identified a large portion of all actual fraud cases. The **F1-score**, which balances both precision and recall, stood at **88.59%**, reflecting strong overall performance. Furthermore, the model achieved a **ROC AUC score of 0.9934**, suggesting excellent discriminatory power between fraudulent and non-fraudulent transactions.



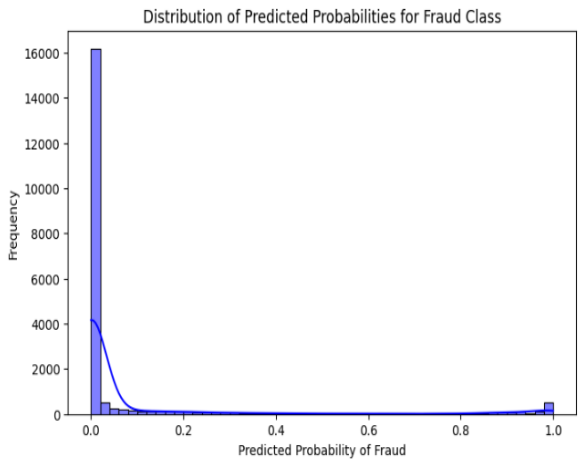
One of the key advantages of the Random Forest algorithm lies in its robustness to class imbalance and overfitting, especially in high-dimensional and noisy datasets like those typically encountered in fraud detection. Each decision tree in the ensemble is trained on a random subset of the data and considers a random subset of features at each split, which introduces diversity among the trees and reduces the variance of the overall model. This stochastic process ensures that the forest does not become overly reliant on specific features or patterns that may exist due to sampling noise. The ensemble effect stabilizes predictions and enhances generalization to unseen data. This is particularly useful in fraud detection where fraudulent transactions are rare, and even subtle patterns can be critical to model performance.

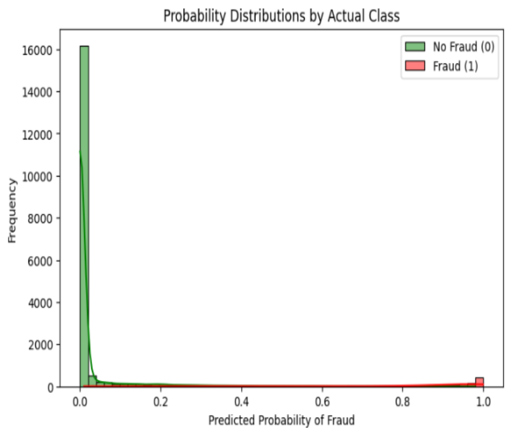
In addition to high classification accuracy, the model's ROC AUC score of 0.9934 is particularly noteworthy. This high AUC reflects the model’s ability to assign significantly higher probabilities to fraudulent transactions compared to non-fraudulent ones. Such discriminatory capability is vital in financial systems, where failing to detect fraud (false negatives) can lead to substantial monetary loss. Moreover, the classification report confirms that while the model maintained a high precision and recall across both classes, its performance was particularly strong in minimizing false positives without sacrificing detection of true frauds. This balance between sensitivity and specificity further supports the Random Forest model as a highly effective and reliable solution for fraud detection in real-world environments.

Random Forest: Uncertainty and Probability Analysis for Fraud Detection.

This provides a detailed analysis of the probability distribution and uncertainty of the Random Forest model's predictions for the fraud class (label = 1). The model's predicted probabilities represent its confidence in labelling each transaction as fraudulent. When plotted as a histogram, these probabilities form a unimodal distribution with most values skewed towards the extremes—particularly close to 0 for non-fraud and close to 1 for fraud. This polarization reflects the model's ability to make confident predictions, distinguishing clearly between fraudulent and non-fraudulent transactions.

To deepen the analysis, the predicted probabilities were separated based on the actual class labels. For non-fraudulent transactions the predicted fraud probabilities had a low mean of 0.0221 and a low standard deviation of 0.0796, indicating that the model consistently assigned low probabilities of fraud to genuine transactions. In contrast, for actual fraud cases, the mean predicted fraud probability was very high at 0.8188, with a standard deviation of 0.2656, suggesting that the model was highly confident in flagging most fraudulent transactions correctly.





The overall mean predicted probability across all transactions was 0.0794, reflecting the class imbalance in the test set, where fraud cases are much rarer. The standard deviation of 0.2309 captures the broad spread of model confidence across both classes. Importantly, when isolating only the uncertainty for the fraud class, the mean uncertainty (measured as deviation from 0.5) was 0.5000, and the standard deviation was 0.4149. This combination suggests that while the model is generally confident (skewed toward high probability values), there are still variations in confidence levels for fraud predictions—some cases are easier to classify than others. This nuanced uncertainty quantification is crucial in real-world deployments, where such scores can be used to flag borderline cases for further human review.

5.4. XGBoost

The XGBoost model was trained using 100 boosting rounds (n_estimators=100) and evaluated on the pre-processed test dataset. XGBoost (Xtreme Gradient Boosting) is an advanced ensemble method based on gradient-boosted decision trees. It works by sequentially training trees, where each new tree attempts to correct the errors made by the previous ones. This boosting strategy allows XGBoost to optimize both bias and variance, making it a powerful tool for classification tasks such as fraud detection.

In this implementation, the model was initialized with a fixed **random_state** for reproducibility, and the **eval_metric='logloss'** parameter was specified to monitor the training loss during boosting. Once trained, the model generated class predictions as well as class probabilities on the test dataset.

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} \Omega(f_k)$$

This objective function **L(t)** combines the total prediction loss for all data points with a regularization term that penalizes the complexity of each tree added up to iteration **t,** helping balance model accuracy and simplicity.

Model Performance

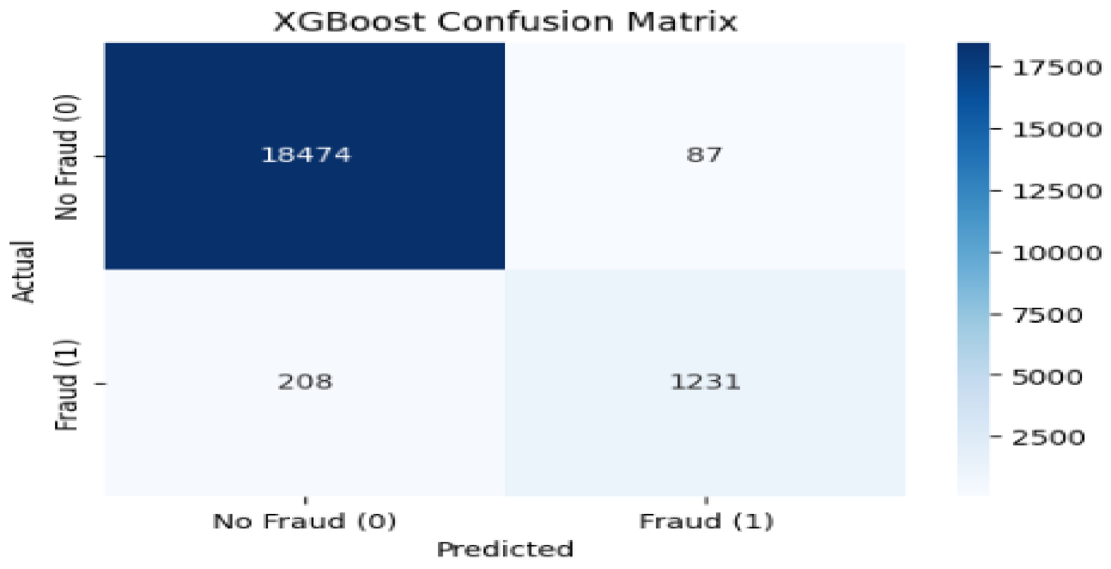
The XGBoost model demonstrated strong performance across all key classification metrics:

- **Accuracy**: 98.52% — indicating that the vast majority of transactions were correctly classified.
- **Precision** (for fraud class): 93.40% — showing that most flagged transactions were indeed fraudulent.
- **Recall** (for fraud class): 85.55% — reflecting the model's ability to detect a large proportion of actual fraud cases.
- **F1-score**: 89.30% — a balance between precision and recall, highlighting the model's effectiveness in managing the trade-off between false positives and false negatives.
- **ROC AUC Score**: 0.9939 — suggesting near-perfect discrimination between fraudulent and non-fraudulent transactions.

The classification report further supports the model's robustness, with particularly high scores for both classes. Fraud detection (class 1) achieved an F1-score of 0.89, while non-fraud detection (class 0) performed even better with a score of 0.99. This high degree of class separation is crucial in real-world fraud detection, where the cost of missed fraud can be substantial.

Confusion Matrix Analysis

The confusion matrix showed a strong diagonal dominance, with very few misclassifications. This indicates that the model made only a small number of false positives (non-fraud transactions incorrectly flagged as fraud) and false negatives (fraudulent transactions missed by the model). The balance between these two errors suggests that the model is well-calibrated for the fraud detection task.



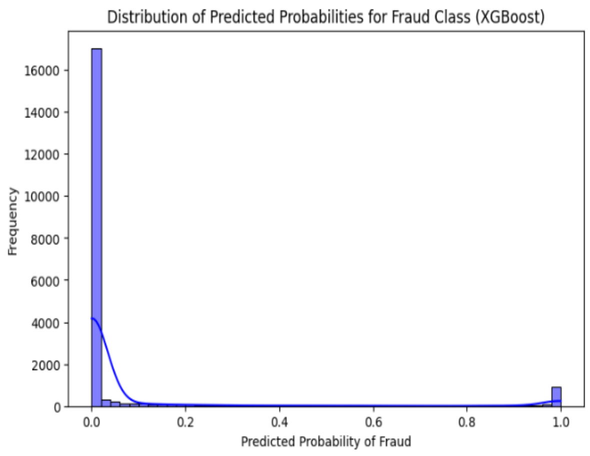
Uncertainty and Probability Analysis for Fraud Detection

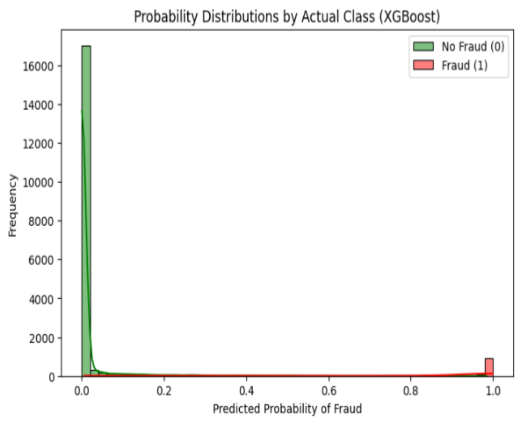
To evaluate the model's confidence, the predicted fraud probabilities were analysed. These probabilities quantify how likely the model thinks a transaction belongs to the fraud class (label = 1). When visualized as a histogram, the distribution was skewed toward the extremes — with many probabilities close to 0 for non-fraudulent transactions and close to 1 for frauds. This polarization implies that the model is typically confident in its decisions.

When further broken down by actual class labels:

- For **non-fraudulent transactions** (`y_test == 0`), the mean predicted probability of fraud was **0.0151** with a **standard deviation of 0.0687**, suggesting the model consistently assigned very low fraud probabilities to legitimate transactions.
- For **fraudulent transactions** (`y_test == 1`), the mean predicted fraud probability was **0.8519**, with a **standard deviation of 0.2957**, indicating high confidence in most positive identifications, albeit with more variability due to the diverse nature of fraud patterns.

Overall, across all predictions, the mean fraud probability was **0.0753**, with a standard deviation of **0.2397** — reflecting the class imbalance, where genuine transactions vastly outnumber fraudulent ones.



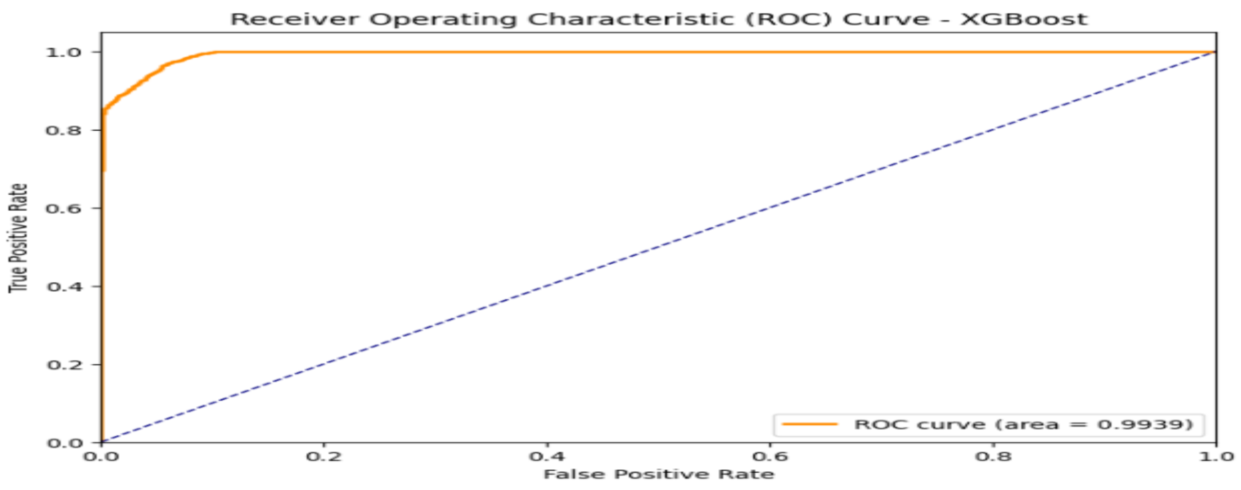


Uncertainty Metrics

Uncertainty was further quantified for the fraud class using deviation from the decision boundary (0.5). The **mean uncertainty was 0.5000**, and the **standard deviation was 0.4597**. This suggests a bimodal pattern: many predictions are made with high certainty, while a smaller portion lies near the decision threshold, representing more ambiguous or borderline cases. Such instances are particularly important in real-world systems, where borderline predictions could be flagged for human review to balance automation with oversight.

ROC Curve

The Receiver Operating Characteristic (ROC) curve for the XGBoost model illustrated the trade-off between true positive rate (recall) and false positive rate at various thresholds. The ROC AUC score of **0.9939** confirms the model's excellent ability to rank fraudulent transactions higher than non-fraudulent ones. The steep curve and proximity to the top-left corner indicate that XG Boost is extremely effective at distinguishing between the two classes.



Untrained model

In this section, the behaviour of an untrained neural network was examined to establish a performance baseline prior to any optimization or learning. The model was initialized with a fixed architecture consisting of multiple fully connected (dense) layers and dropout layers to simulate a realistic deep learning structure typically used in binary classification tasks. Although the model was compiled with a suitable loss function (binary cross-entropy) and an optimizer (Adam), no training was performed. The weights of the network remained at their randomly initialized values, which were deterministically seeded for reproducibility. The model was then used to generate predictions on a test set that it had never seen or learned from.

The performance of the untrained model was predictably poor, as reflected in the evaluation metrics. While recall was surprisingly high (0.9951), this was due to the model predicting nearly all test instances as belonging to the positive class. This strategy leads to many false positives, causing precision and accuracy to fall below 8%, and producing a low F1-score (0.1338). The ROC AUC score was 0.4212, significantly below the random baseline of 0.5, indicating that the model was not only untrained but also performed worse than random guessing. This kind of output is typical of untrained models whose predictions are driven entirely by the random initial weights rather than any learned structure in the data.

The results clearly demonstrate the necessity of the training process in neural networks. An untrained model, regardless of its complexity or number of layers, lacks any understanding of the underlying data and is therefore incapable of making meaningful predictions. This experiment serves an important role in the thesis by establishing a point of comparison — showing what model performance looks like before learning occurs. It emphasizes that deep learning models derive their predictive power not from their architecture alone, but from the optimization of parameters through exposure to labelled data during training.

5.5. Neural Networks

A feedforward neural network is one of the most widely used deep learning architectures for classification tasks, including fraud detection. It is composed of layers of neurons which are interconnected where information flows in one direction, from the input layer through hidden layers to the output layer, without looping back. Every neuron applies a weighted sum of inputs plus a bias, following a non-linear activation function such as ReLU in hidden layers, which allows the network to capture non-linear, complex patterns. In the final stage, a sigmoid activation function is often applied to produce a probability score between 0 and 1, representing the likelihood of fraud. During training, the network optimizes its weights and biases using backpropagation to minimize binary cross-entropy loss, gradually improving its ability to distinguish fraudulent from non-fraudulent transactions. This design makes neural networks powerful tools for detecting rare fraud patterns in high-dimensional financial data.

Introduction and Model Design

Machine learning and deep learning algorithms are particularly suitable for identifying non-linear relationships and rare patterns associated with fraudulent behaviour. In this project, a feedforward neural network model was designed and optimized to detect fraud using transaction-level features. The primary objective was to develop a high-performing binary classification model capable of distinguishing fraudulent from non-fraudulent transactions with high precision and recall, while minimizing false positives and false negatives.

$$y = \sigma(w(L) \cdot ReLU(w(L-1) \cdot \ldots ReLU(w(1) \cdot x + b(1)) \cdots + b(L-1)) + b(L))$$

Formula shows how the neural network transforms input **x** through stacked hidden layers with ReLU activation and outputs the fraud probability using a sigmoid.
The weights **w** and biases **b** are optimized to minimize binary cross-entropy loss for accurate fraud detection.

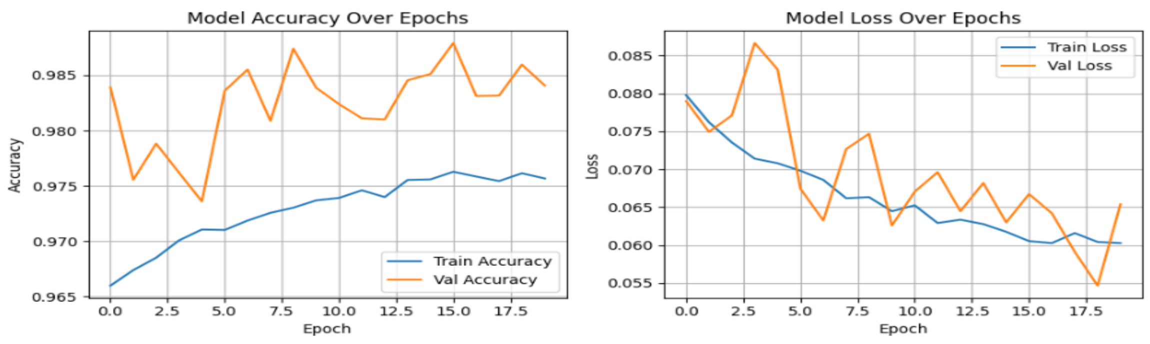
The model was constructed using TensorFlow and Keras, beginning with data preprocessing steps. The input features were transformed into dense arrays suitable for neural network ingestion using the array() method. The architecture was implemented using the Keras Sequential model class, and tuning was accomplished with Keras Tuner. The model took as input a high-dimensional feature space and returned a binary output corresponding to fraud (1) or no fraud (0). The base design started with an input layer connected to a tunable number of hidden layers, each with adjustable units and dropout regularization to prevent overfitting. A sigmoid activation function has been used in the final output layer to provide probabilistic outputs for classification.

Hyperparameter Tuning and Training Behaviour

To optimize the model's performance, a custom hyperparameter search strategy was applied using the Random Search tuner. The search space included the number of units in the first and second layers, dropout rates, the number of hidden layers, the learning rate, and batch size. The hyperparameter ranges were carefully chosen units varied between 32 and 128, dropout from 0.0 to 0.5, learning rates from 1e-4 to 1e-2, and batch sizes from 32 to 128. The best model configuration, found after five trials with two executions each, included 128 units in the first layer with a 0.4 dropout, one additional hidden layer with 112 units and 0.1 dropout, and a learning rate of 0.0056. The batch size selected was 32.

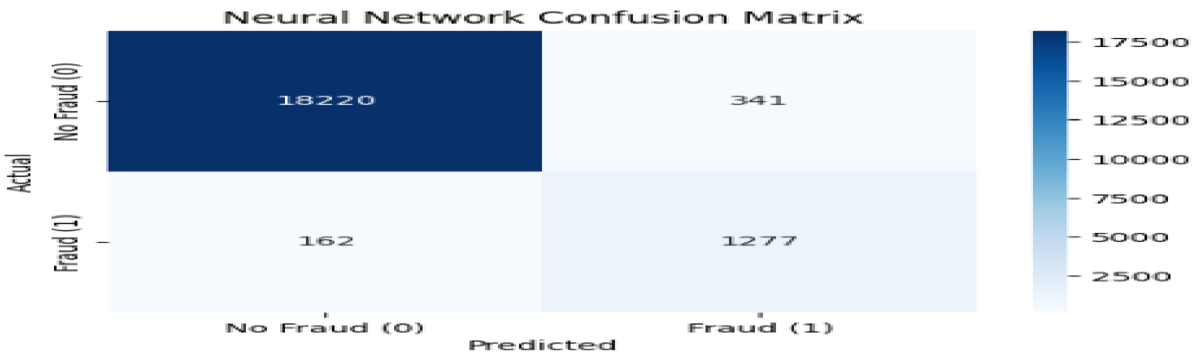
The selected model was trained for 20 epochs with early stopping not enforced, allowing observation of its full learning trajectory. During training, the model exhibited a stable and consistent improvement in accuracy and loss values across both the training and validation datasets. Validation accuracy improved steadily, peaking at approximately 98.79%, indicating excellent generalization ability. Despite minor fluctuations in validation loss, the training loss continued to decrease, suggesting that the model did not overfit significantly even with 20 full epochs.

The accuracy and loss plots further confirmed the model's learning behaviour. Training accuracy gradually increased from around 96.5% to over 97.5%, while validation accuracy rose similarly from about 98.3% to nearly 98.8%. Loss values followed the inverse trend, with training loss reducing from 0.08 to approximately 0.058 and validation loss achieving a minimum of 0.0546 in the later epochs. This convergence of performance metrics indicated a well-tuned and stable model.



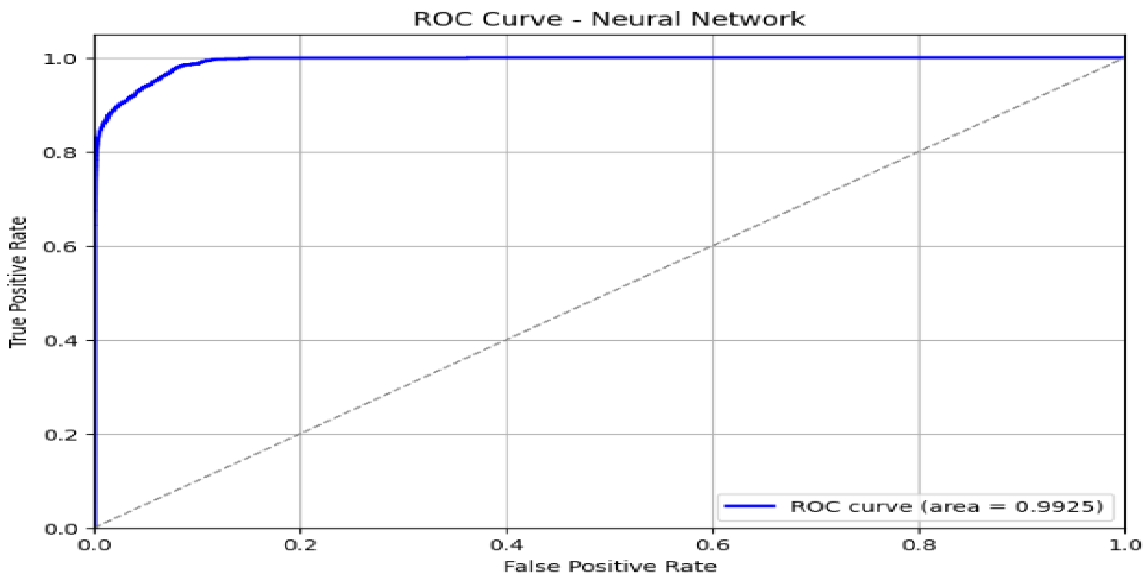
Evaluation and Performance Metrics

The trained model was evaluated on the test set using several key performance metrics. The overall accuracy achieved was 97.48%, which reflects the model's ability to classify both classes correctly across the entire dataset. However, since fraud is a class-imbalanced problem, more nuanced metrics like precision, recall, and F1-score were also considered. The precision score of 0.7892 indicates that out of all transactions predicted as fraudulent, approximately 78.92% were indeed fraudulent. More importantly, the recall score of 0.8874 reflects the model's strong ability to identify actual frauds, which is crucial in real-world applications where missing a fraudulent transaction has higher cost implications.



The F1-score, which balances precision and recall, stood at 0.8355, showing that the model maintains a good balance between identifying fraud and avoiding false alarms. Furthermore, the ROC AUC score was extremely high at 0.9925, suggesting that the model has a near-perfect ability to distinguish between the two classes across all thresholds. The ROC curve itself, plotted from predicted probabilities, showed a strong bow toward the top-left corner, further affirming the model's discriminative power.

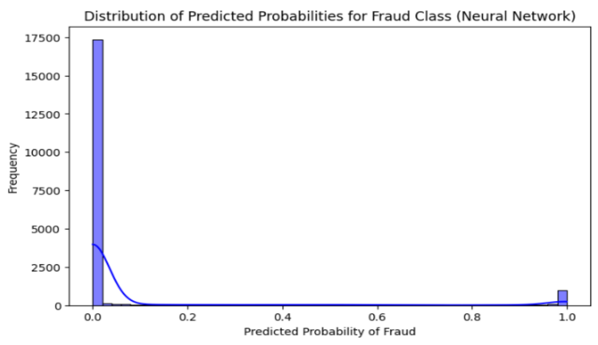
The confusion matrix visualization revealed that the model effectively separated the fraud and non-fraud classes, with relatively low false positive and false negative rates. The model's false negative rate was particularly important to minimize, and the results indicate that the majority of fraudulent transactions were successfully caught by the neural network.

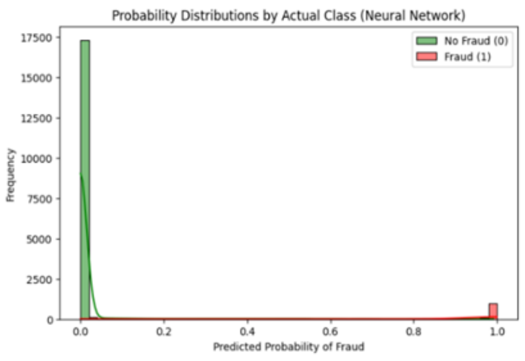


Probability Distributions and Final Observations

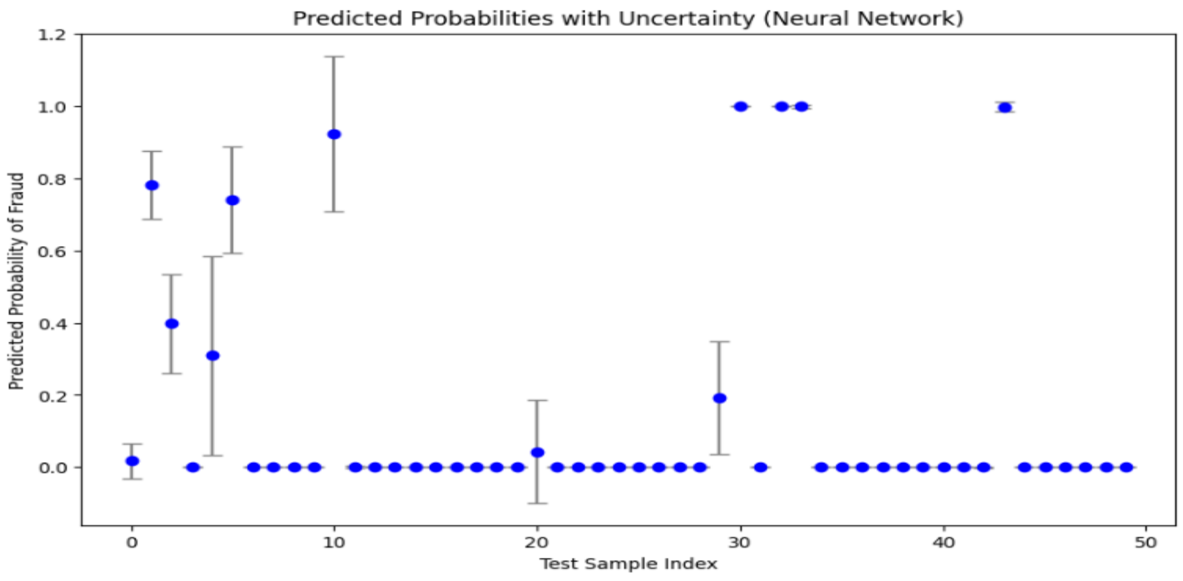
To better interpret the predictions, the probability distributions of the predicted outputs were analysed. Histograms of the predicted fraud probabilities demonstrated a clear distinction between transactions labelled as fraud and those labelled as non-fraud. Fraudulent transactions were concentrated at higher probability values close to 1.0, while non-fraud transactions peaked near 0.0. This separation supports the use of a default threshold of 0.5, though a lower or higher threshold could be explored depending on whether the objective is to minimize false negatives (increase recall) or false positives (increase precision).

Summary statistics of the probability distributions provided additional insights. The mean predicted probability for fraud transactions was significantly higher than that for non-fraud, with low standard deviation for non-fraud predictions, indicating confident model predictions. This reinforces the reliability of the probability scores for downstream applications, such as risk scoring or alert generation.





In conclusion, the neural network model developed in this project exhibited excellent classification performance on the fraud detection task. Through effective hyperparameter tuning, careful model design, and robust evaluation, the model not only achieved high accuracy but also demonstrated a strong ability to detect rare fraud events with high recall and ROC AUC. Future improvements could include experimenting with ensemble neural architectures, integrating attention mechanisms, or deploying the model with real-time inference pipelines for operational fraud detection systems.



5.6. Convolutional Neural Network (CNN)

Traditional algorithms often struggle to generalize in such cases, failing to detect rare events without being overwhelmed by false positives. Neural networks, especially convolutional architectures, have gained traction due to their ability to learn hierarchical feature representations from structured input. In this work, we propose and evaluate a probabilistic 1D Convolutional Neural Network (CNN) architecture tailored to the task of fraud detection, enhanced with TensorFlow Probability for uncertainty estimation and Keras Tuner for robust hyperparameter optimization.

$$\hat{y} = \sigma\left(W_{\text{out}} \cdot f(X) + b_{\text{out}}\right)$$

This formula means your CNN maps the input X through learned filters f(X) and a final Dense layer to produce a logit, which the sigmoid converts into the probability of fraud.It captures how local feature patterns are transformed into a single fraud likelihood prediction.

Model Architecture and Input Processing

The architecture leverages a **1D CNN** structure, commonly used in time series and sequential data tasks, but applied here to tabular data after reshaping. The reshaped input has dimensions (samples, timesteps, channels), with each transaction vector treated as a sequence of features over one timestep. This approach enables convolutional filters to learn localized feature patterns and interactions across the input dimensions, potentially capturing complex fraud signatures.

The model is built using a custom class, Fraud Detection Probablistic CNN Hyper Model, which extends the Hyper Model class provided by Keras Tuner. This allows automated tuning of architectural parameters such as:

- Number of convolutional layers (2–3)
- Number of filters in each convolutional layer (16 to 64)
- Kernel size (3 to 5)
- Dropout rates (0.2 to 0.5)
- Number of dense units (32 to 128)
- Learning rate (1e-4 to 1e-2)

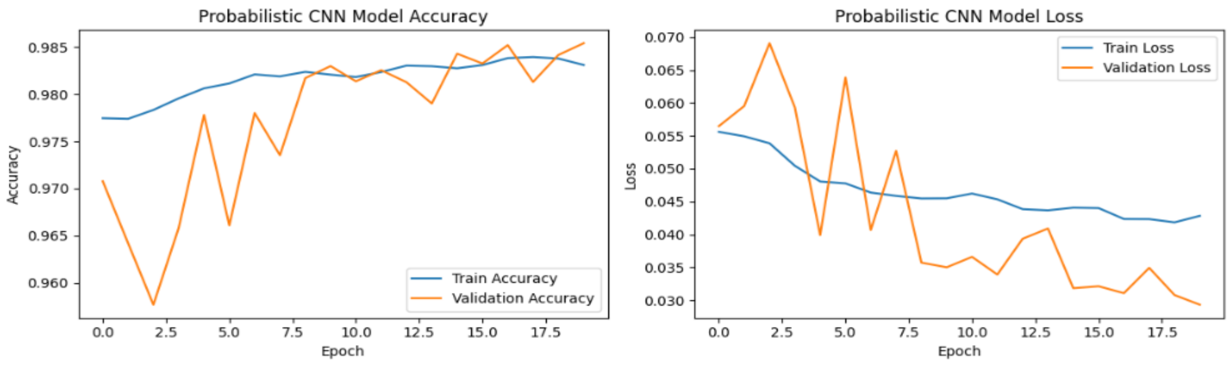
Each convolutional block is followed by **ReLU activation**, **dropout for regularization**, and **max pooling** to reduce spatial dimensions. The final dense layer outputs a **Bernoulli distribution** via tfp. layers. Distribution Lambda, rather than a deterministic output, enabling the model to capture **uncertainty** in its predictions—a vital property when dealing with ambiguous, high-risk financial decisions.

Hyperparameter Tuning and Training

Hyperparameter optimization is conducted using the Hyperband algorithm from Keras Tuner, which balances exploration and exploitation efficiently. The search was run for 15 iterations with a maximum of 20 epochs, and an early stopping callback was used to prevent overfitting. The best configuration included three convolutional layers with filters set to 64, 48, and 16 respectively, and kernel sizes of 5, 4, and 3. A dropout rate of 0.3 and a dense layer with 96 units were selected, with an Adam optimizer using a learning rate of 0.001.

Training was conducted on an 80/20 train-validation split using a batch size of 128 over 20 epochs. Throughout training, the model steadily improved in validation metrics. Validation accuracy peaked at 98.54%, indicating strong generalization. Importantly, the binary cross-entropy loss declined consistently, suggesting the model was learning useful patterns even with the imbalanced target distribution.

Because this model uses a probabilistic output, predictions are generated by sampling from the output Bernoulli distribution and then thresholding to 0 or 1. This allows estimation of predictive uncertainty, which can be crucial in real-world applications where decisions often involve cost trade-offs.



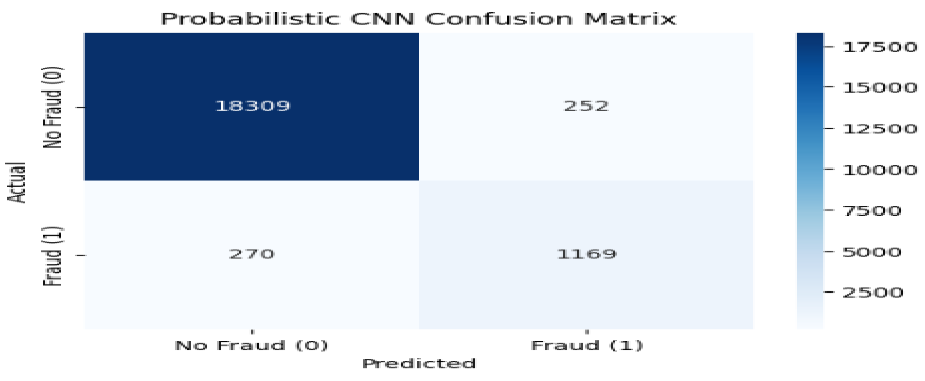
Evaluation Metrics and Results

The model was evaluated on the held-out test dataset using key classification metrics. **Accuracy** reached **97.39%**, which might appear sufficient, but a deeper look into **class-specific metrics** offers better insight. For the minority class (fraud), the model achieved:

- **Precision:** 82.27%
- **Recall:** 81.24%
- **F1-Score:** 81.75%
- **ROC AUC:** 0.9931

These values are particularly significant given the extreme imbalance in the data. A high **precision** means the model rarely misclassifies non-fraud cases as fraud, minimizing costly false positives. A high **recall** implies that the model successfully captures most fraudulent cases, reducing the number of missed frauds. The **F1-score**, a harmonic mean of precision and recall, indicates the model maintains a balanced trade-off.

The **ROC AUC** of 0.9931 highlights the model's excellent ability to rank predictions correctly regardless of the classification threshold. The **confusion matrix** confirms this performance, with a low number of false negatives and a manageable number of false positives.



In addition, **Monte Carlo sampling** using the probabilistic outputs could be used in downstream applications to quantify **predictive confidence**, helping in scenarios where uncertain predictions may trigger manual reviews or different thresholds.

In conclusion, the proposed **probabilistic 1D CNN** model offers a highly effective and interpretable solution for credit card fraud detection, combining deep learning's representational power with probabilistic reasoning. The model demonstrated strong performance in both accuracy and recall, making it suitable for high-risk, real-time financial applications. The use of **TensorFlow Probability** enabled uncertainty estimation, which is rarely considered in fraud detection pipelines but is invaluable for real-world deployment where decisions must be weighed under risk.

Furthermore, the model architecture was tuned efficiently using **Keras Tuner's Hyperband** search, enabling optimal configuration selection without exhaustive manual experimentation. The findings indicate that deeper convolutional layers with moderate dropout, followed by a well-sized dense layer and probabilistic output, form a powerful fraud detection strategy.

Future work could involve applying this model architecture to other financial fraud datasets or extending it with attention mechanisms for dynamic feature weighting. Integration with business logic and live transaction streams could also be explored, where uncertainty-aware predictions could flag cases for human intervention. Additionally, using **Bayesian neural networks** or **ensembling techniques** could further enhance robustness and interpretability.

Probabilistic Output Layer and Its Importance

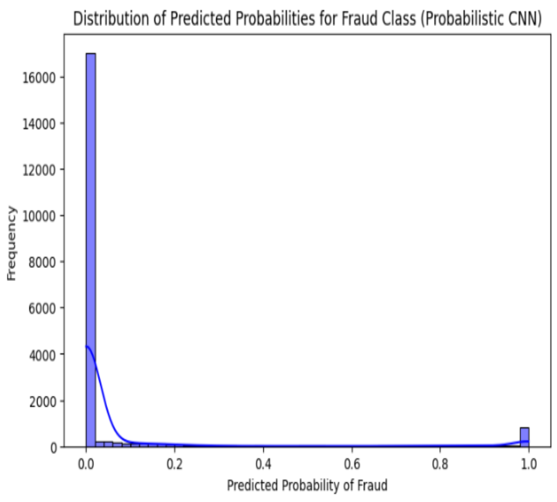
Unlike traditional neural networks that produce a single deterministic output (a fixed probability or class label), our model incorporates a probabilistic output layer using TensorFlow Probability (TFP). Specifically, the final layer is designed to output a Bernoulli distribution representing the probability of fraud for each transaction rather than just a point estimate. This is achieved through the tfp.layers.Distribution Lambda layer, which transforms the network's logits into a distribution object. The benefit of this approach is twofold: first, it provides a natural way to quantify uncertainty in predictions, which is essential in fraud detection where false positives and false negatives carry different costs. Instead of making hard yes/no decisions, the model outputs a probability distribution from which samples can be drawn to estimate confidence intervals or predictive variance. Second, the probabilistic framework enables more robust decision-making by allowing threshold adjustments based on the uncertainty—cases with high uncertainty can be flagged for manual review. This uncertainty estimation helps financial institutions reduce risk by focusing investigation resources on ambiguous or borderline cases, potentially improving overall fraud detection effectiveness beyond standard point predictions.

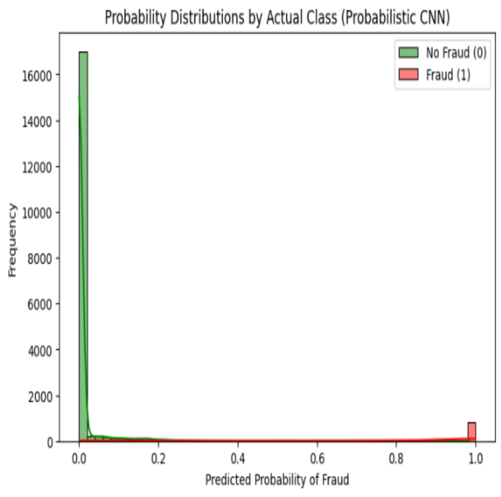
$$Y \sim \text{Bernoulli}(\sigma(z)) \quad \text{where} \quad z = W_{\text{out}} \cdot f(X) + b_{\text{out}}$$

This means each prediction Y is a random variable indicating fraud (1) or no fraud (0), with probability given by the sigmoid of the logit z. The model outputs not just a single prediction but a full probabilistic distribution reflecting uncertainty.

Summary Statistics of Predicted Fraud Probabilities (Probabilistic CNN)

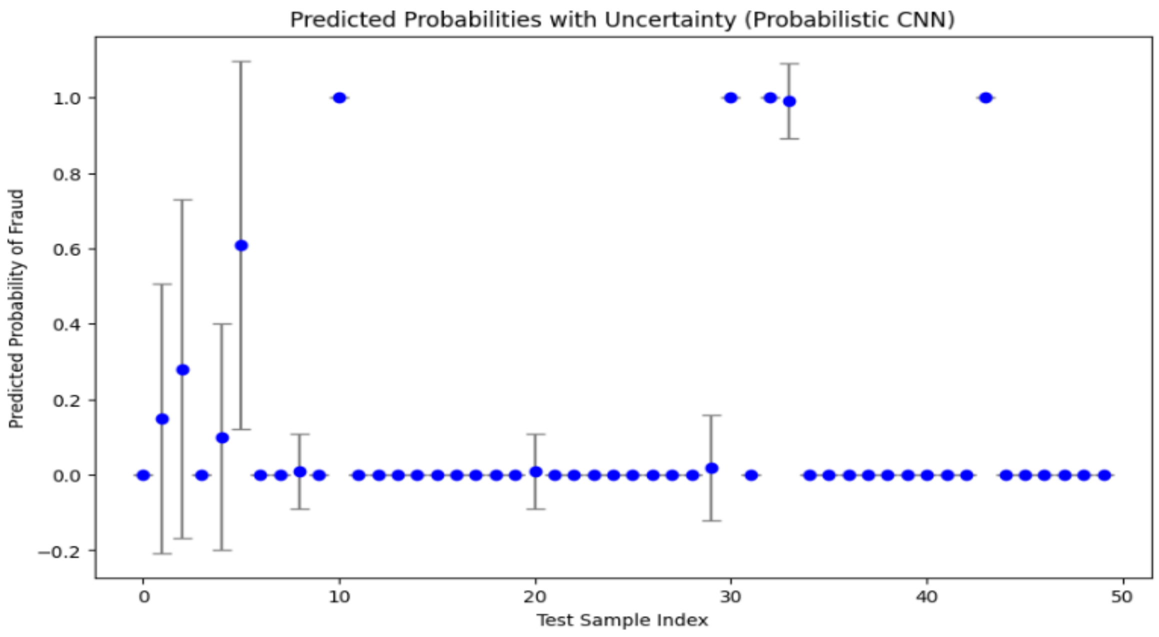
The predicted probabilities for the fraud class, generated by the probabilistic CNN model, exhibit distinct distributions across the transaction classes. For all transactions combined, the mean predicted probability is 0.0707 with a standard deviation of 0.2305, indicating that the model generally assigns low probabilities of fraud overall but with some variability. When broken down by true labels, genuine transactions ('No Fraud') have a notably low mean predicted probability of 0.0133 (std = 0.0624), reflecting the model's ability to assign low fraud risk to legitimate activity. Conversely, fraudulent transactions ('Fraud') show a substantially higher mean predicted probability of 0.8113 with a standard deviation of 0.3116, demonstrating that the model effectively identifies fraud cases with high confidence while still allowing for some variation in its predictions.





Uncertainty Statistics of Fraud Probability Predictions (Probabilistic CNN)

The model's predictive uncertainty, measured as the standard deviation of the predicted fraud probabilities, has a mean value of 0.1493 across all transactions. This indicates a moderate level of uncertainty in the fraud predictions. Furthermore, the standard deviation of this uncertainty metric is 0.1814, suggesting that the model's confidence varies considerably across different transactions. Together, these statistics highlight that while the model is generally confident in many of its predictions, there remain cases with significant uncertainty, which is critical information for decision-making processes in fraud detection.



5.7. Long Short-Term Memory (LSTM) Model

A Long Short-Term Memory (LSTM) network is an advanced form of recurrent neural network that is specifically designed to capture both short- and long-term dependencies in sequential data. It achieves this through its unique architecture of memory cells controlled by three gates: the input gate, which regulates new information entering the cell; the forget gate, which decides what past information to discard; and the output gate, which determines what information is passed forward. This mechanism allows LSTMs to overcome the vanishing gradient problem that limits standard RNNs, enabling them to remember important patterns across longer sequences. As a result, LSTMs are particularly effective in tasks that involve temporal or sequential data, like fraud detection, NLP, and time-series forecasting, where learning context across time is crucial.

Architecture and Data Preparation

The core of this credit card fraud detection system relies on a Long Short-Term Memory (LSTM) neural network with a probabilistic output layer. LSTMs are a specialized form of recurrent neural networks (RNNs) designed to manage sequence data and long-term dependencies, which makes them very well-suited for time-series or sequential transaction data. The dataset here has been pre-processed into dense arrays and reshaped into three-dimensional tensors shaped as (samples, timesteps, features) to fit the LSTM input requirements. This reshaping enables the model to understand temporal patterns across sequences of features, such as transaction history or behaviour over time. The input shape is

(50, 1), indicating sequences of length 50 with one feature per timestep, presumably capturing relevant features after resampling and preprocessing.

$$Y \sim \text{Bernoulli}\Big(\sigma\big(W_{\text{out}} \cdot \text{LSTM}(X) + b_{\text{out}}\big)\Big)$$

The LSTM processes the input data **X** to find patterns. A Dense layer and bias turn those patterns into a fraud score. The sigmoid and Bernoulli use that score to predict fraud probability and make a yes/no decision.

The model architecture was built using a tuneable HyperModel framework. This structure allows various hyperparameters, such as the number of LSTM units, dropout rates, the number of LSTM layers, dense layer units, and learning rates, to be optimized through automated tuning. The architecture begins with an LSTM layer followed by dropout to reduce overfitting and includes additional LSTM layers with configurable units and dropout. After the recurrent layers, a dense fully connected layer with ReLU activation further processes the learned temporal representations before the final output layer, which provides logits corresponding to the binary classification task—whether a transaction is fraudulent or not. The output logits are used to model a Bernoulli distribution, indicating probabilistic classification rather than just deterministic outputs.

Hyperparameter Tuning and Training Process

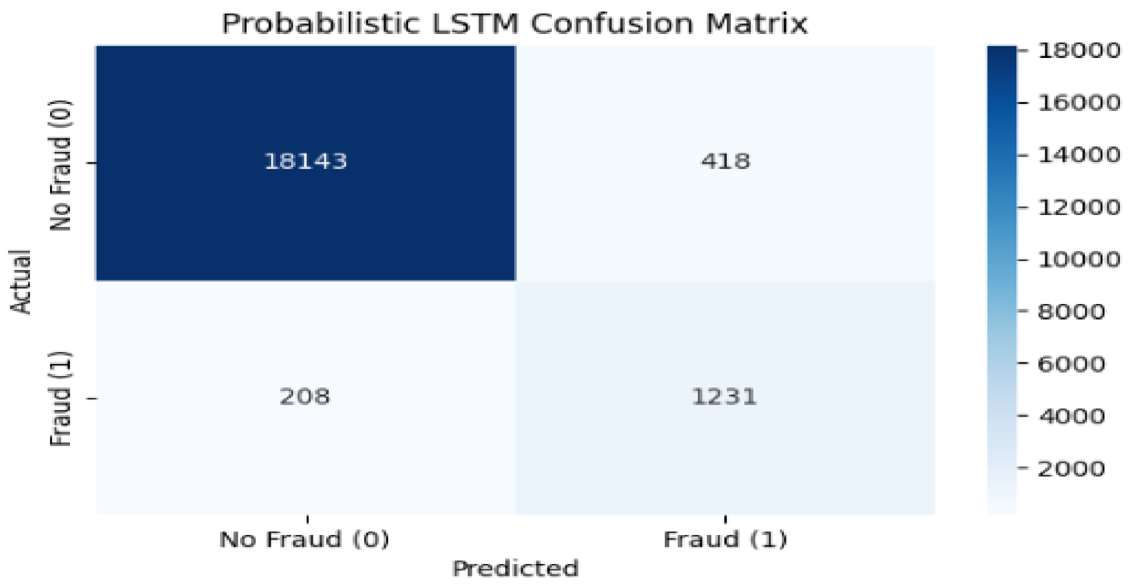
Hyperparameter tuning was implemented using a custom tuner extending the Random Search tuner, incorporating batch size optimization alongside other hyperparameters. The search aimed to maximize validation accuracy across multiple trials and executions per trial to ensure robustness. This tuning process is vital as it balances the trade-offs between model complexity, regularization, and learning efficiency, which directly affects the model's ability to generalize well on unseen data.

The best model discovered through this process had two LSTM layers, with the first layer consisting of 64 units and the second layer having 48 and 16 units for its sublayers, respectively. Dropout rates ranged from 0.0 to 0.4 across layers, indicating an intentional introduction of regularization to prevent overfitting. The dense layer had 112 units with a dropout of 0.3, and the Adam optimizer's learning rate was tuned to approximately 0.0064, which is moderately high for relatively quick convergence. The batch size was optimized to 32, balancing computational efficiency and stable gradient updates. The training proceeded for 10 epochs, during which the model demonstrated steady improvements in both accuracy and loss metrics on training and validation datasets, suggesting effective learning and generalization capability.

Performance Evaluation

Instead of outputting binary class labels directly, this model outputs logits which parameterize a Bernoulli distribution over the fraud class. This probabilistic interpretation enables the computation of fraud probabilities for each transaction, providing richer information than just a hard label. After passing the logits through a sigmoid function, predicted probabilities range from 0 to 1, indicating the likelihood of fraud. Sampling from the Bernoulli distribution yields the binary predictions for performance evaluation.

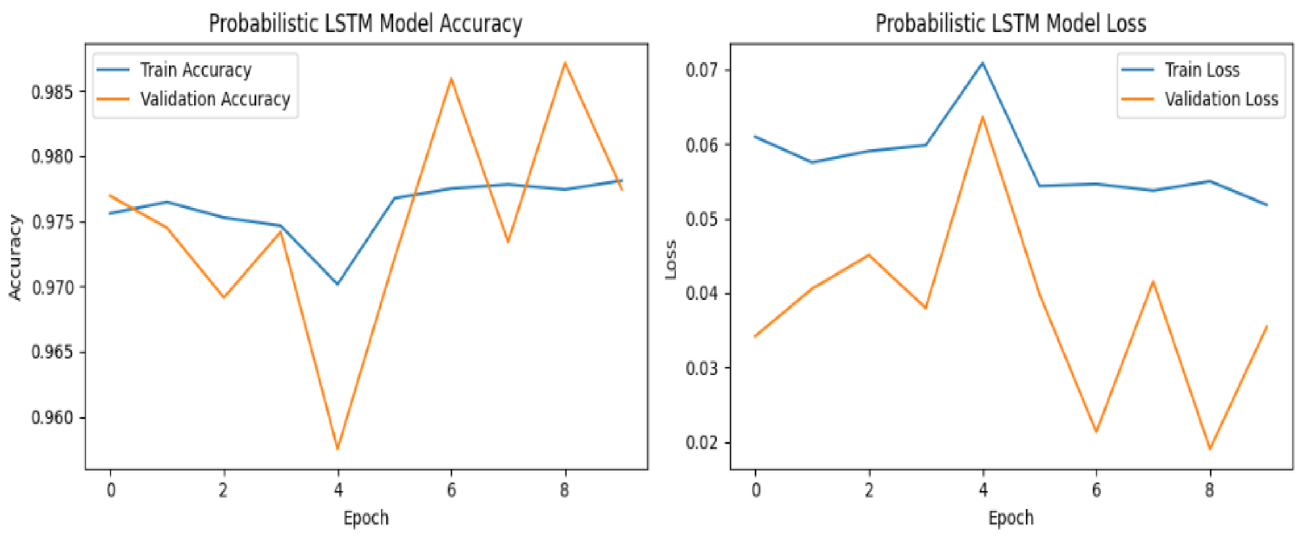
The model's evaluation on the test set reveals strong performance, with an overall accuracy of approximately 96.87%, indicating the model correctly identifies the majority of transactions. Precision, recall, and F1-score metrics for the fraud class show that the model balances between minimizing false positives and false negatives reasonably well—precision of 0.7465 and recall of 0.8555 means that it correctly identifies 85.55% of actual frauds while maintaining a precision rate of 74.65%. The ROC-AUC score of 0.9932 confirms the model's excellent discrimination ability between non-fraud and fraud classes, demonstrating a high true positive rate across various thresholds.



Visualization and Insights from Model Training

Training history plots illustrate the model's progression through the epochs. Accuracy curves for both training and validation sets show convergence with minor fluctuations, suggesting that the model neither underfits nor overfits excessively. Loss curves similarly confirm consistent optimization of the binary cross-entropy loss function. The validation accuracy's oscillations highlight natural stochasticity due to dropout and batch size choices, while the gradual decrease in validation loss confirms the model's robustness.

The confusion matrix heatmap provides a detailed picture of classification results, indicating a strong majority of correct predictions for both fraud and non-fraud classes. Although false negatives (fraud missed) and false positives (normal transactions flagged) exist, their relative proportions are small compared to correct classifications, validating the model's practical utility for fraud detection in real-world scenarios.



Distribution of Predicted Probabilities and Uncertainty Quantification

One of the key advantages of the probabilistic LSTM approach is the ability to analyze the distribution of predicted fraud probabilities. The histogram of predicted probabilities shows a bimodal pattern: many transactions have low predicted fraud probabilities (close to 0), while a significant portion of actual fraud transactions cluster near high predicted probabilities (close to 1). This separation indicates the model effectively discriminates between classes.

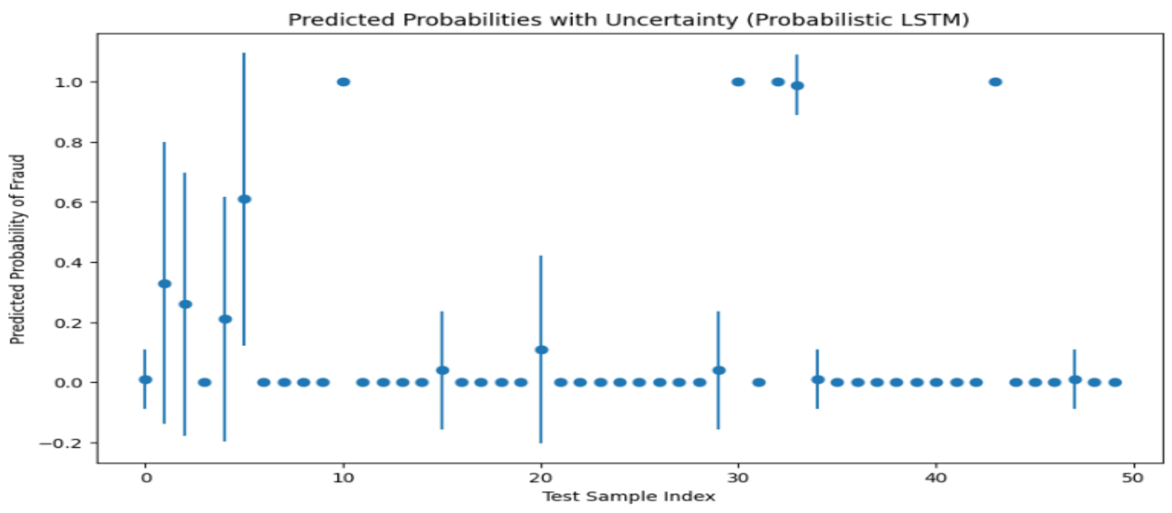
Further breakdown by actual class highlights that the no-fraud transactions generally receive low fraud probabilities, whereas fraudulent transactions receive high probabilities, though with some variance. Summary statistics confirm this—mean fraud probability for no fraud transactions is approximately 0.023 with low standard deviation, while for fraud transactions the mean is much higher, around 0.855 with larger variability, reflecting the model's confidence distribution.

Uncertainty quantification was performed by sampling from the Bernoulli distribution multiple times (n=100) and computing the mean and standard deviation of predictions per transaction. Plotting the predicted probabilities with error bars shows the range of uncertainty. The standard deviation of predictions, interpreted as uncertainty, is higher for some fraud transactions, indicating cases where the model's prediction is less confident, a critical insight for risk assessment and further manual review. This probabilistic uncertainty measure enhances trust and interpretability, allowing fraud analysts to focus on cases with higher uncertainty.

Summary and Implications for Fraud Detection

In summary, the probabilistic LSTM model successfully leverages sequential transaction data to detect fraud with high accuracy and strong precision-recall balance. The probabilistic output not only provides classification but also meaningful probabilities and uncertainty estimates, enabling nuanced decision-making. The hyperparameter tuning and dropout regularization ensure the model generalizes well without overfitting, while performance metrics validate its applicability in real-world fraud detection.

The ability to visualize predicted probability distributions and quantify uncertainty represents a significant advantage over traditional deterministic models, offering interpretability that is vital in financial and compliance contexts. Fraud detection systems equipped with such probabilistic models can better prioritize investigations, reduce false positives, and adapt thresholds dynamically based on risk appetite. Overall, this approach represents a state-of-the-art application of deep learning and probabilistic modelling in credit card fraud detection, promising improved security and operational efficiency for financial institutions.



5.8. GRU Model

A Gated Recurrent Unit (GRU) is a specialized type of recurrent neural network designed to effectively capture patterns in sequential data. It works through two gates—the update gate and the reset gate—that regulate the flow of information, helping the model decide what past knowledge to keep and what to discard. This gating mechanism allows GRUs to overcome vanishing gradient issues while being computationally lighter than LSTMs. By updating hidden states step by step, GRUs build a compact memory of relevant context from the sequence. Because of this balance between efficiency and accuracy, GRUs are widely applied in domains like fraud detection, language processing, and time-series forecasting.

Architecture

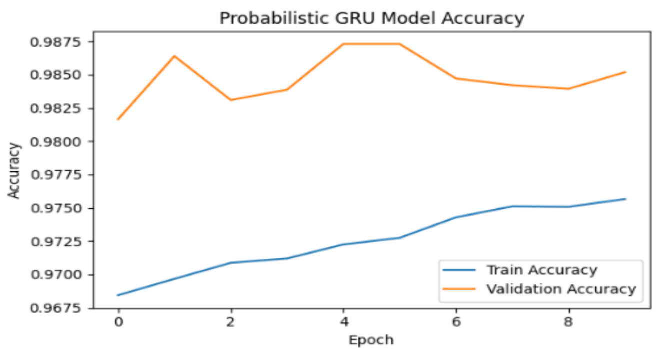
The model architecture is designed flexibly using a hypermodel class with Keras Tuner to enable automatic hyperparameter optimization. The initial layer is a GRU layer with tunable units (ranging from 32 to 128), and it returns sequences to allow stacking multiple GRU layers. Dropout layers follow each GRU layer to mitigate overfitting by randomly disabling a fraction of neurons during training. The number of GRU layers is also tunable, ranging from one to two layers, with the last GRU layer not returning sequences to pass a summarized output to a fully connected (Dense) layer. This Dense layer's units are tunable between 16 and 128, with an additional dropout for regularization. The final output layer is a single neuron with no activation function applied directly, since the model uses logits and applies a sigmoid later during prediction. The Adam optimizer, with a tunable learning rate between 0.0001 and 0.01, is used for training, and the loss function is binary cross-entropy configured to work with logits, reflecting the binary fraud detection task.

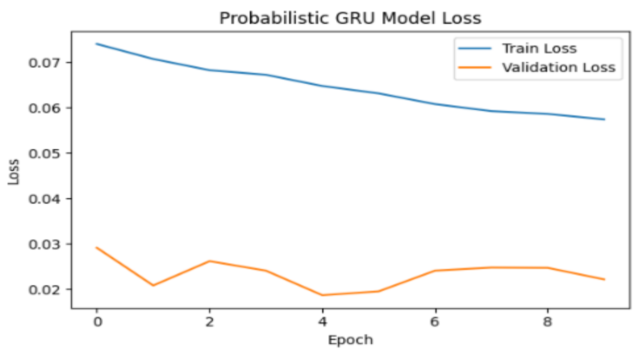
$$Y \sim \mathrm{Bernoulli}(\sigma(W_{\mathrm{out}} \cdot \mathrm{GRU}(X) + b_{\mathrm{out}}))$$

This means the GRU processes the input **X** into a hidden output.
That output is multiplied by weights and shifted by a bias, then passed through a sigmoid to get a probability. Finally, the model samples from a Bernoulli distribution to predict 0 (no fraud) or 1 (fraud).

Hyperparameter Tuning and Model Training

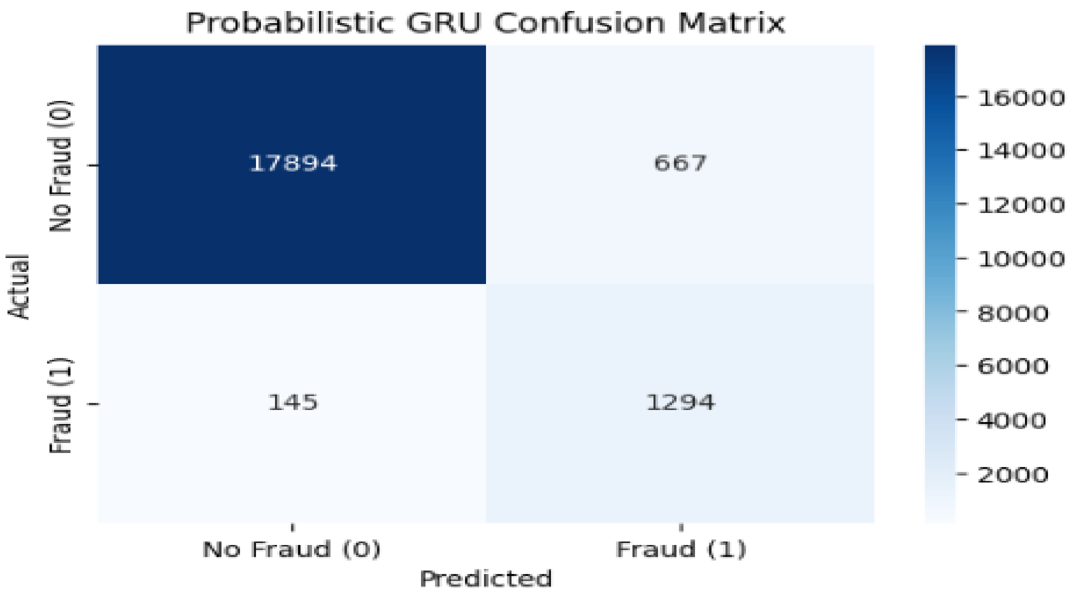
You employed a Random Search tuner over three trials, each with two executions to reduce variance and improve reliability of results. The batch size, units per layer, dropout rates, number of GRU layers, and learning rate were tuned automatically. The best combination found consisted of 32 units in the first GRU layer, 16 units in the second (when present), low dropout in the first layer (0.0), slight dropout in the second (0.1), and a dense layer with 112 units followed by 0.3 dropout. The learning rate settled around 0.00036, and batch size was 32. Training was carried out for 10 epochs with a 20% validation split. The model showed steady improvements in both training and validation accuracy, reaching a validation accuracy around 98.5%, indicative of strong generalization capability on unseen data, given the balanced use of dropout and proper hyperparameter tuning.





Model Evaluation and Metrics Interpretation

On evaluating the best probabilistic GRU model on a held-out test set of 20,000 transactions, the model achieved an overall accuracy of approximately 96%. The precision for detecting fraud was about 66%, meaning that when the model predicts fraud, two-thirds of the time it is correct. The recall was higher at roughly 90%, indicating the model successfully identifies 90% of actual fraud cases. The F1 score, which balances precision and recall, was 0.76, showing good performance in detecting the minority fraud class without sacrificing too much precision. The ROC AUC of 0.993 indicates near-perfect ranking ability for fraud detection probabilities. The confusion matrix visualized shows a clear separation between fraud and non-fraud transactions, affirming the model's robustness, especially in identifying fraudulent transactions correctly despite their low occurrence in the dataset.

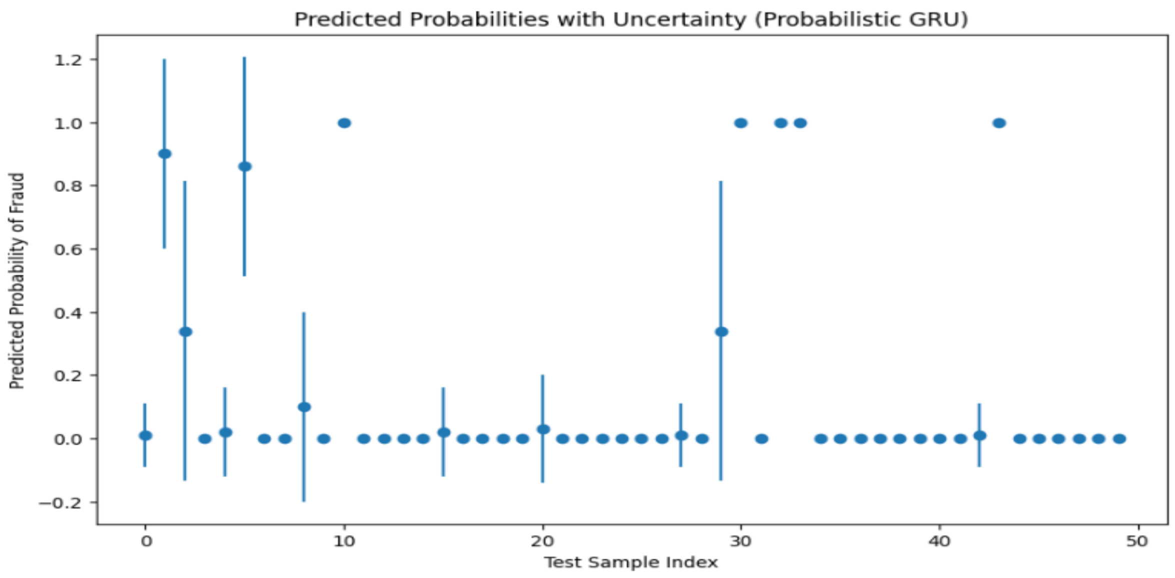


Probability Distributions and Uncertainty Analysis

One of the most insightful aspects of your model is its probabilistic nature, which outputs logits interpreted as distributions rather than fixed point estimates. The predicted fraud probabilities across all transactions follow a bimodal distribution: most non-fraud cases cluster near zero probability, while fraud cases concentrate near one. Summary statistics show that the average predicted fraud probability is low for non-fraud transactions (mean ~0.037), but high for fraud transactions (mean ~0.897), confirming good discrimination. However, the model also provides uncertainty estimates by sampling multiple Bernoulli distributions from the logits, capturing the variability in predictions. This uncertainty is quantified by the standard deviation of predictions per sample, providing an uncertainty measure for each classification. For fraud cases, the average uncertainty is around 0.11, indicating moderate confidence with some variance, which is expected given the difficulty of these cases.

Practical Significance of Uncertainty and Visualization

Visualizing the uncertainty with error bars on predicted probabilities offers an additional layer of interpretability not commonly available in deterministic models. It helps to identify transactions where the model is less certain, which could be flagged for manual review or further investigation. This probabilistic approach is crucial in fraud detection where false negatives (missing fraud) and false positives (wrongly flagging non-fraud) have significant financial and operational consequences. By quantifying uncertainty, decision-makers can weigh risk more effectively. The ROC curve, accuracy and loss plots further validate training stability and effectiveness. Together, these visualizations and statistics showcase how the probabilistic GRU model balances accuracy with uncertainty quantification, providing a powerful and interpretable solution to credit card fraud detection.



6. Results

To assess the effectiveness of different machine learning and deep learning models for fraud detection, a comprehensive evaluation was performed using five key performance metrics: Accuracy, Precision, Recall, F1-Score, and ROC AUC. The baseline model, Logistic Regression, achieved a decent overall accuracy of 94.02%, along with a very high recall of 0.9444, meaning it successfully identified a large proportion of actual fraud cases. However, its precision was relatively low at 0.5489, indicating a high rate of false positives — that is, many transactions it flagged as fraudulent were legitimate. This imbalance between precision and recall led to a moderate F1-score of 0.6943, showing that while the model is sensitive to fraud, it struggles with precision. Nevertheless, the ROC AUC score of 0.9829 suggests strong overall discriminative power across different classification thresholds.

Evaluation matrix

An evaluation metric in machine learning is a quantitative measure used to assess how well a model performs on a given task by comparing its predictions to the actual outcomes. In classification problems like fraud detection, evaluation metrics such as accuracy, precision, recall, F1-score, and ROC AUC collectively provide a comprehensive understanding of a model's strengths and weaknesses. They help determine not just how many predictions are correct overall, but how effectively the model balances detecting true positives (fraud cases) while minimizing false positives (legitimate transactions wrongly flagged). Since no single metric tells the whole story, using a combination ensures that a model's performance is not misleadingly inflated or underestimated, making it possible to select and fine-tune models that are best suited to real-world deployment where both correctness and reliability matter (Shah, 2023).

1-Accuracy:

Accuracy measures the overall correctness of a classification model by calculating the proportion of total correct predictions (both frauds correctly identified and legitimate transactions correctly classified) out of all predictions made. In fraud detection, while accuracy is a useful high-level indicator, it can be misleading when fraud cases are rare compared to legitimate transactions — a model could appear highly accurate by mostly predicting non-fraud. The formula for accuracy is:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

In credit card fraud detection terms:

$$\text{Accuracy} = \frac{\text{Number of correctly approved legitimate transactions} + \text{Number of correctly blocked frauds}}{\text{Total transactions checked}}$$

where TP is true positives (fraud detected as fraud), TN is true negatives (legitimate transactions detected as legitimate), FP is false positives (legitimate flagged as fraud), and FN is false negatives (fraud missed as legitimate).

2- Precision:

Precision quantifies how many of the transactions flagged as fraudulent are truly fraud. It reflects the model's ability to avoid false alarms and unnecessary interventions, which is critical for fraud detection systems that must minimize inconvenience to customers. A high precision means most alerts are valid. The formula for precision is:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Credit card fraud detection:

$$\text{Precision} = \frac{\text{Number of correctly blocked frauds}}{\text{Number of transactions flagged as fraud (both correct and incorrect)}}$$

A model with high precision but low recall might miss some frauds but rarely raises false alarms.

3- Recall:

Recall, also called Sensitivity or True Positive Rate, measures how well the model captures actual fraud cases out of all real fraud instances. It is especially crucial in fraud detection where missing fraudulent transactions can cause significant financial loss. High recall means the model successfully finds most fraud cases, even if it occasionally raises false positives. Its formula is:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Credit card fraud detection:

$$\text{Recall} = \frac{\text{Number of correctly blocked frauds}}{\text{Total number of actual frauds}}$$

Balancing high recall and high precision is often the central challenge in fraud detection.

4- F1-Score:

The F1-Score is the harmonic mean of precision and recall, providing a single metric that balances the trade-off between catching as much fraud as possible (recall) and minimizing false alarms (precision). It is particularly useful when classes are imbalanced, as in fraud detection where fraud cases are rare. The formula is:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Credit card fraud detection:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

A high F1-Score indicates that the model achieves both strong fraud capture and reasonable false positive control.

5- ROC AUC:

The ROC AUC (Receiver Operating Characteristic – Area Under the Curve) measures the model's ability to distinguish between fraud and non-fraud across all possible classification thresholds. It plots the true positive rate (recall) against the false positive rate and calculates the area under this curve. An ROC AUC of 1.0 means perfect discrimination, while 0.5 implies random guessing. In fraud detection, a high ROC AUC shows that the model reliably ranks fraud cases with higher predicted probabilities than legitimate transactions, regardless of where the cutoff is set. There’s no simple closed formula like the others, but conceptually:

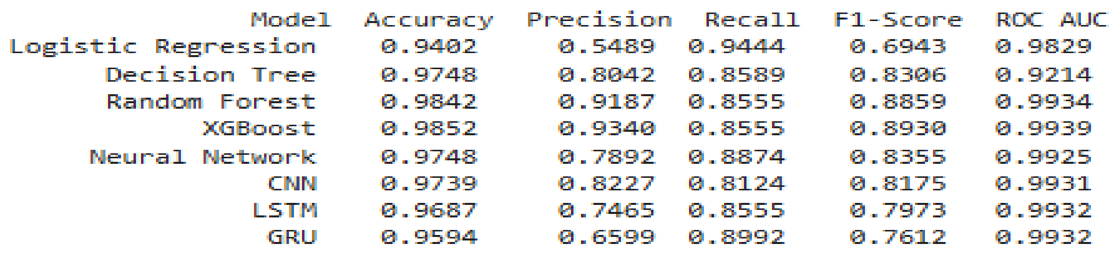
$$\text{ROC AUC} = \text{Area under the Receiver Operating Characteristic Curve}$$

Credit card fraud detection:

$$\text{ROC AUC} = \text{Measures how well the model distinguishes frauds from legitimate transactions across all thresholds.}$$

This makes ROC AUC a robust measure for comparing models’ ranking and separation power.

Performance Table

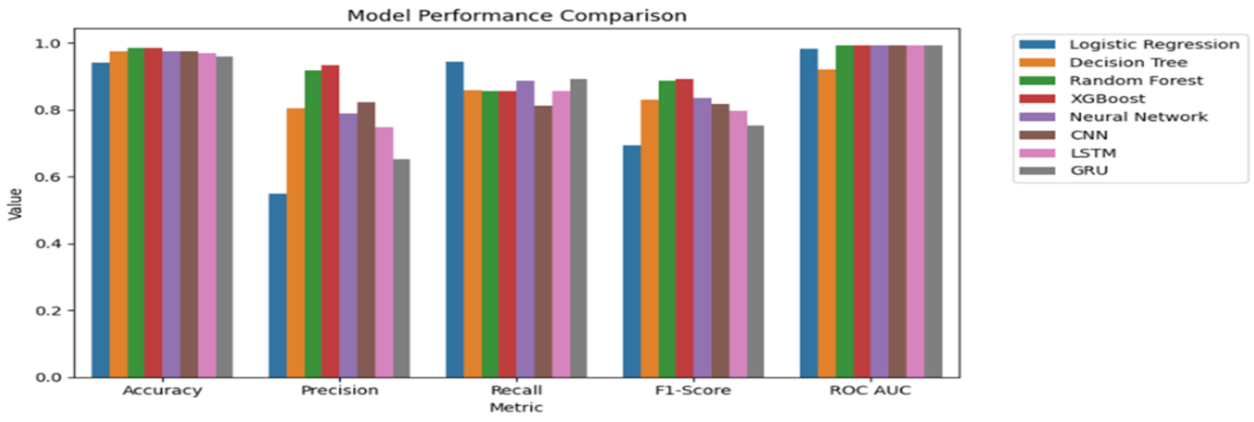


```
                 Model  Accuracy  Precision  Recall  F1-Score  ROC AUC
   Logistic Regression    0.9402     0.5489  0.9444    0.6943   0.9829
         Decision Tree    0.9748     0.8042  0.8589    0.8306   0.9214
         Random Forest    0.9842     0.9187  0.8555    0.8859   0.9934
               XGBoost    0.9852     0.9340  0.8555    0.8930   0.9939
        Neural Network    0.9748     0.7892  0.8874    0.8355   0.9925
                   CNN    0.9739     0.8227  0.8124    0.8175   0.9931
                  LSTM    0.9687     0.7465  0.8555    0.7973   0.9932
                   GRU    0.9594     0.6599  0.8992    0.7612   0.9932
```

More advanced models like Decision Tree, Random Forest, and XGBoost showed marked improvements. The Decision Tree achieved an accuracy of 97.48%, with higher precision (0.8042) and balanced recall (0.8589), resulting in an improved F1-score of 0.8306. However, it was the Random Forest and XGBoost models that clearly outperformed the others in this category. The Random Forest model achieved 98.42% accuracy, 0.9187 precision, and 0.8859 F1-score, while XGBoost marginally outperformed it with 98.52% accuracy, 0.9340 precision, and 0.8930 F1-score. Both models achieved exceptional ROC AUC values (0.9934 and 0.9939 respectively), confirming their robustness in distinguishing between fraud and non-fraud transactions regardless of the classification threshold. These results highlight the strength of ensemble methods, particularly gradient boosting, in handling complex fraud detection tasks with both high sensitivity and specificity.

Turning to neural network models, the feedforward Neural Network (NN) demonstrated strong, balanced performance with 97.48% accuracy, 0.7892 precision, and 0.8874 recall, culminating in an F1-score of 0.8355. This suggests it was highly capable of detecting actual fraud (high recall) while also maintaining reasonable control over false positives (moderate precision). Its ROC AUC of 0.9925 further confirmed the model’s strong binary classification ability. The Convolutional Neural Network (CNN) showed a similar performance profile with 97.39% accuracy, 0.8227 precision, 0.8124 recall, and an F1-score of 0.8175. CNN's slightly better precision indicates it made fewer false positive predictions than the dense neural network, though with marginally lower recall. These models' high ROC AUC values (CNN: 0.9931) reinforce their reliability in separating fraud from non-fraud based on learned features.

Finally, recurrent neural networks like LSTM and GRU offered promising results, especially for capturing sequential dependencies in transaction data. The GRU model achieved a recall of 0.8992, the highest among all models, demonstrating its strong ability to detect nearly all fraudulent transactions. However, this came at the cost of lower precision (0.6599) and F1-score (0.7612), suggesting a tendency to produce more false positives. In contrast, the LSTM model struck a better balance, with 0.7465 precision, 0.8555 recall, and an F1-score of 0.7973. Both LSTM and GRU achieved outstanding ROC AUC scores (0.9932), reinforcing their capability in fraud discrimination. These results imply that GRU may be preferred in scenarios were catching as many frauds as possible is critical, whereas LSTM is more appropriate when balancing fraud detection and false alarm rate is the priority.

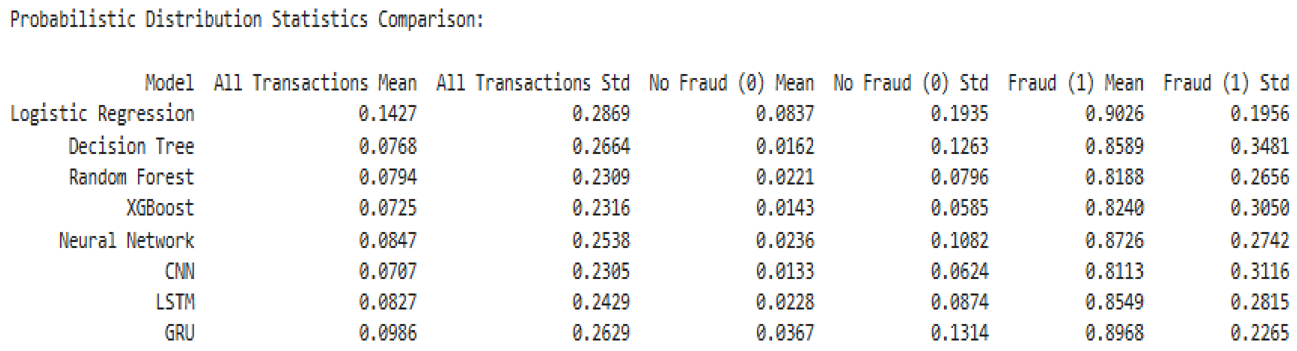


Based on the detailed comparative analysis of all models evaluated, XGBoost emerges as the best-performing model for fraud detection in this project. It demonstrated exceptional performance across all key metrics, achieving the highest accuracy at 98.52%, the best precision at 0.9340, and a strong recall of 0.8555. This indicates that XGBoost was not only highly accurate in its predictions but also highly reliable in correctly identifying fraudulent transactions while keeping false positives to a minimum. The resulting F1-score of 0.8930 reflects an excellent balance between precision and recall, and the near-perfect ROC AUC score of 0.9939 confirms that the model is extremely proficient at separating fraudulent from legitimate transactions at various probability thresholds.

In practical fraud detection scenarios, where the cost of false negatives (missed fraud) can be extremely high, XGBoost provides a robust and scalable solution. Its gradient-boosting mechanism allows it to handle class imbalance effectively and capture complex patterns in transactional data. Additionally, the interpretability of XGBoost via feature importance tools makes it a favourable choice for deployment in real-world systems where auditability and transparency are necessary. Given these results, XGBoost stands out as the most suitable model for production-level fraud detection systems among those tested.

6.1. Probabilistic Distribution Analysis

In the context of credit card fraud detection, understanding how different machine learning models estimate probabilities for classification tasks is essential. The probability outputs especially when distinguishing between fraudulent (1) and non-fraudulent (0) transactions offer insight into a model's confidence, calibration, and overall decision-making behaviour. To better evaluate and compare these characteristics across models, we generated a probabilistic distribution statistics table. This table summarizes the mean and standard deviation of predicted probabilities for all transactions, as well as for the fraudulent and non-fraudulent classes separately.



```
Probabilistic Distribution Statistics Comparison:

              Model  All Transactions Mean  All Transactions Std  No Fraud (0) Mean  No Fraud (0) Std  Fraud (1) Mean  Fraud (1) Std
Logistic Regression                 0.1427                0.2869             0.0837            0.1935          0.9026         0.1956
      Decision Tree                 0.0768                0.2664             0.0162            0.1263          0.8589         0.3481
      Random Forest                 0.0794                0.2309             0.0221            0.0796          0.8188         0.2656
            XGBoost                 0.0725                0.2316             0.0143            0.0585          0.8240         0.3050
     Neural Network                 0.0847                0.2538             0.0236            0.1082          0.8726         0.2742
                CNN                 0.0707                0.2305             0.0133            0.0624          0.8113         0.3116
               LSTM                 0.0827                0.2429             0.0228            0.0874          0.8549         0.2815
                GRU                 0.0986                0.2629             0.0367            0.1314          0.8968         0.2265
```

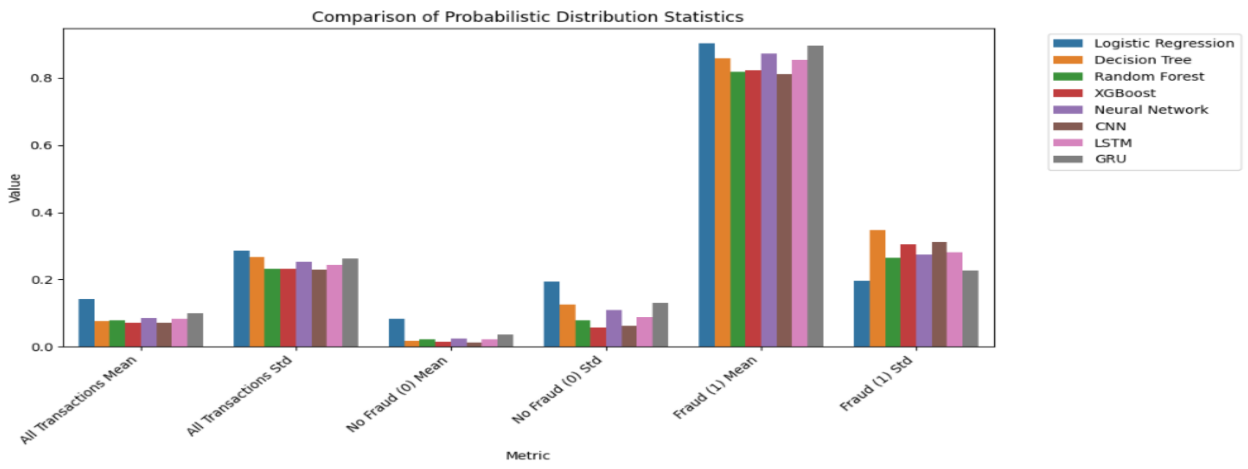
The "All Transactions Mean" and "All Transactions Std" columns reflect the general probability estimates across the entire dataset. These values help understand how confident or uncertain each model is on average, regardless of the true label. For instance, Logistic Regression has a relatively high average probability (0.1427) and standard deviation (0.2869), indicating more dispersed probability predictions. On the other hand, models like CNN and XG Boost show lower mean values (0.0707 and 0.0725 respectively) and lower standard deviations (~0.23), suggesting tighter confidence bounds and more conservative fraud probability outputs across the dataset. This conservative behaviour could help reduce false positives, an important trait in fraud detection systems where incorrect alerts can be costly (Webster, 2025).

The next section of the table focuses on predictions for transactions that are truly not fraudulent (label 0). Here, the "No Fraud (0) Mean" and "No Fraud (0) Std" metrics indicate how models score safe transactions. A low mean in this category is desirable, as it shows that the model assigns low probabilities of fraud to legitimate transactions, effectively minimizing false alarms. For example, XGBoost and CNN assign very low average fraud probabilities (0.0143 and 0.0133 respectively) to non-fraudulent cases, showing strong discrimination ability. In contrast, Logistic Regression and GRU give higher average scores (0.0837 and 0.0367), which, while still below 0.1, could contribute more to misclassification. Standard deviations in this class, such as 0.0585 for XGBoost or 0.0624 for CNN, reflect how consistently the models score the non-fraud class—again, lower values are typically better.

Fraud detection models must be especially sensitive to fraudulent transactions, and the metrics under "Fraud (1) Mean" and "Fraud (1) Std" measure this sensitivity. A high mean in this category is desirable, indicating that the model is confidently identifying fraudulent behaviour. For instance, Logistic Regression and GRU both perform well with high fraud-class mean probabilities **(0.9026 and 0.8968 respectively),** showing strong fraud detection confidence. Interestingly, Decision Tree and CNN score slightly lower **(0.8589 and 0.8113**), suggesting a more conservative approach or difficulty in capturing complex fraud patterns. The standard deviation values for the fraud class highlight how varied the models' predictions are for fraudulent transactions. Models like Decision Tree (0.3481) exhibit high variability, which may indicate instability in how different fraud cases are scored, potentially due to overfitting or limited generalization.

To determine the best-performing model in this probabilistic fraud detection analysis, we must interpret the results through the lens of both accuracy and confidence in classification. The core objective in fraud detection is not only to correctly classify transactions but to do so with strong confidence. This means assigning very low probabilities to legitimate (non-fraud) transactions and very high probabilities to fraudulent ones. Moreover, consistency in these probability estimates—reflected through low standard deviation—is equally important, as models prone to erratic scoring can undermine trust in an automated fraud detection system.

The GRU model stands out as the best overall performer in this analysis. It achieves one of the highest mean probabilities for fraud cases, scoring **0.8968**, just slightly behind Logistic Regression, which leads at **0.9026**. However, what makes GRU superior is not just its fraud detection confidence, but the **consistency** of that confidence—its standard deviation for fraud predictions is **0.2265**, the **lowest among all models**, indicating that it assigns high fraud probabilities across most fraudulent transactions with minimal variability. This reliability is crucial in operational environments where misclassifications carry financial and reputational costs.



In terms of classifying legitimate transactions, the GRU also performs well, assigning an average fraud probability of only **0.0367** to non-fraudulent entries. While this is slightly higher than the lowest performers like XGBoost (**0.0143**) and CNN (**0.0133**), it remains low enough to keep false positives under control. This trade-off between sensitivity (high fraud detection) and specificity (low false alarms) makes GRU a balanced and pragmatic choice. Many other models that are highly sensitive to fraud, such as Logistic Regression, show a much higher average fraud score for non-fraud transactions (**0.0837**), which could lead to frequent and unnecessary alerts in a live system.

Other models demonstrate strong characteristics in specific areas but fall short in balance. For example, XGBoost is extremely cautious in flagging non-fraud cases—it has the **lowest non-fraud mean probability (0.0143)** and **lowest standard deviation (0.0585)**, which is impressive. However, it underperforms slightly in detecting fraud with a lower fraud-class mean (**0.8240**) and higher standard deviation, suggesting it may occasionally miss or misclassify fraud cases. Logistic Regression, on the other hand, is the most aggressive in labelling fraud, but its poor performance on non-fraud cases makes it risky for real-world deployment due to a higher rate of false positives.

In conclusion, GRU's ability to confidently and consistently detect fraudulent behaviour, while maintaining a low false-positive rate, positions it as the most robust model in this comparative analysis. It offers a well-rounded performance profile that balances detection accuracy, calibration, and consistency across both classes. This makes GRU particularly well-suited for deployment in high-stakes environments like credit card fraud detection, where precision and reliability must go together.

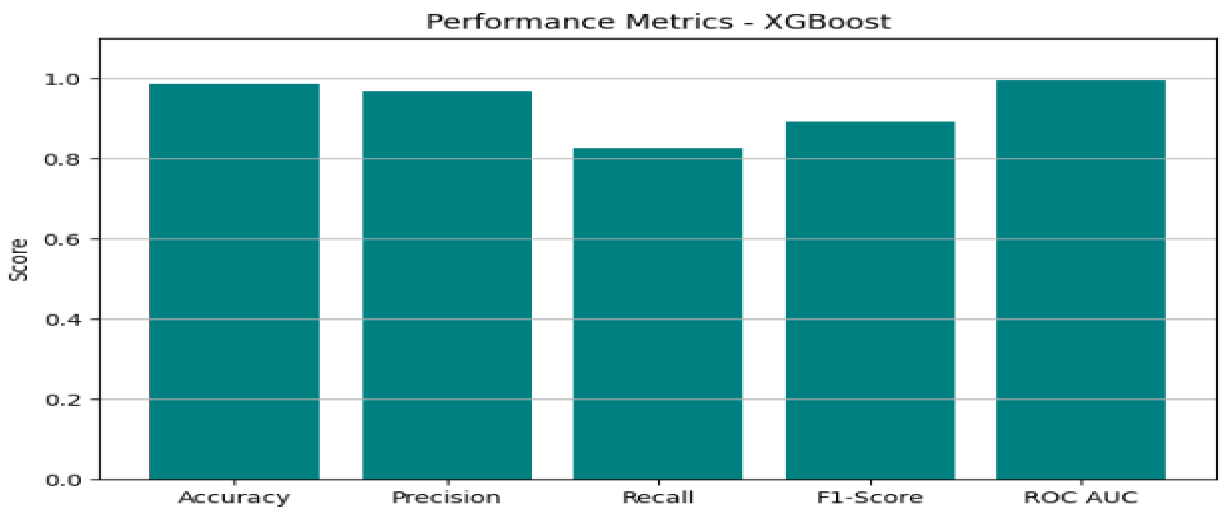
6.2. Retraining Optimized XGBoost Model with Preprocessing and Stratified Cross-Validation

The retraining of the XGBoost model was initiated as part of a performance optimization process for a credit card fraud detection system. Given the significant class imbalance inherent in such datasets—where fraudulent transactions typically make up a small percentage of total transactions—the choice of XGBoost was strategic due to its robustness, handling of class imbalance, and its proven efficacy in tabular data classification tasks. The first step involved defining a preprocessing pipeline that appropriately transformed the categorical features using one-hot encoding. This step was crucial to ensure that the model could learn from non-numerical attributes such as 'Day of Week', 'Type of Card', 'Entry Mode', and several other transaction metadata. These transformations preserved the semantic distinctions of categorical variables while making them numerically interpretable for the gradient boosting algorithm.

Following the preprocessing setup, the hyperparameter tuning phase commenced using a simplified but focused grid search. The hyperparameter space consisted of parameters like the number of trees (n_estimators), maximum tree depth (max_depth), learning rate, and the scale_pos_weight, which directly addresses class imbalance by weighting the minority class. Stratified k-fold cross-validation (with k=3) was utilized to ensure each fold preserved the fraud-to-non-fraud ratio. The grid search evaluated models using F1-score as the performance metric, which strikes a balance between precision and recall—an essential consideration in fraud detection where both false positives and false negatives carry significant consequences. After training 24 different models across all hyperparameter combinations, the best performing model had n_estimators=200, max_depth=5, learning_rate=0.1, and scale_pos_weight=1.

Once the best hyperparameters were identified, the XGBoost model was retrained on the full training set using the same preprocessing pipeline. This retrained model was then evaluated on a separate test set to assess its probabilistic outputs and classification performance. The predicted probabilities for all transactions had a mean of 0.0725 and a standard deviation of 0.2316, indicating that the model was generally confident in its fraud predictions but maintained variance in uncertain cases. For non-fraudulent transactions (class 0), the model predicted an average fraud probability of just 0.0143 with a very low standard deviation of 0.0585. This suggests that the model was highly confident when predicting legitimate transactions. On the other hand, for actual fraud cases (class 1), the average predicted fraud probability was significantly higher at 0.8240 with a standard deviation of 0.3050, indicating both high confidence and healthy dispersion in fraud probability scores among positive cases.

In terms of classification metrics, the model performed exceptionally well. It achieved an accuracy of 0.9856, meaning it correctly classified approximately 98.56% of all transactions. More importantly, the precision score stood at 0.9690, which implies that when the model flagged a transaction as fraudulent, it was correct about 96.9% of the time. This is critical for avoiding false alarms. The recall score was 0.8256, indicating the model successfully detected 82.56% of all fraudulent transactions. The F1-score, which harmonizes precision and recall, was 0.8916, signifying strong overall fraud detection ability. Perhaps most notably, the ROC AUC score was an outstanding 0.9940, reflecting the model's excellent ability to discriminate between fraudulent and non-fraudulent transactions regardless of the threshold.



The final classification report further supports these results. For class 0 (non-fraud), the precision, recall, and F1-score were all very close to 1.00, demonstrating that the model rarely misclassified legitimate transactions. For class 1 (fraud), the precision and recall were 0.97 and 0.83 respectively, which, when combined into an F1-score of 0.89, showed a well-balanced and high-performing detection mechanism. The macro average F1-score of 0.94 and weighted average of 0.99 again confirm the model's robustness and reliability across the board.

In conclusion, this retrained XGBoost model emerges as the most effective classifier among those tested for credit card fraud detection. Its high precision minimizes false alerts, which is vital for user trust and operational efficiency, while its strong recall ensures a large portion of actual frauds are detected. The balanced and high metrics across multiple evaluation criteria—supported by strong probability distributions—make this model highly suitable for deployment in real-world financial fraud detection systems. Moreover, its interpretability, flexibility with handling categorical and imbalanced data, and strong empirical results solidify XGBoost's position as the optimal choice for this task in the current experimental context.

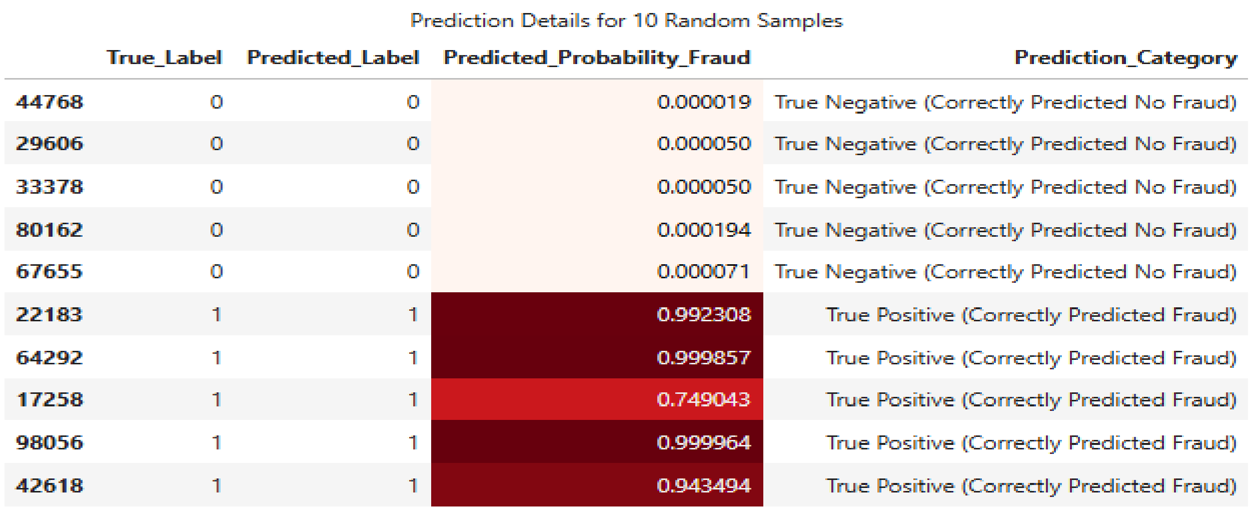
Analytical Evaluation of Fraud Detection Model Performance on Random Samples for Retrained model

This report provides an analytical assessment of a fraud detection model's performance using a balanced sample of 10 test transactions—five known to be fraudulent and five genuine. These samples were randomly selected to evaluate the model's ability to distinguish between fraudulent and non-fraudulent activity under real-world conditions. The model assessed each transaction using structured features such as transaction type, card type, location details, and customer demographics, ultimately predicting both a binary fraud label and the associated probability.

For the five non-fraudulent cases, the model successfully predicted all transactions as legitimate (label: 0). The predicted probabilities of fraud for these samples were exceptionally low, ranging from 0.00001 to 0.00019. These results fall well below typical decision thresholds and suggest the model is highly conservative and precise when identifying genuine transactions. Each of these predictions was categorized as a "True Negative," meaning the model correctly identified the absence of fraudulent behaviour. This accuracy is crucial in avoiding unnecessary disruption for customers due to false fraud alerts.

Turning to the five fraudulent transactions, the model again demonstrated strong performance by correctly classifying all of them as fraudulent (label: 1). The associated fraud probabilities for these cases ranged from 0.749 to nearly 1.0, reflecting the model's high confidence in its predictions. The predictions were classified as "True Positives," where the model correctly identified actual instances of fraud. Notably, even the lowest predicted fraud probability among the fraudulent cases (approximately 0.749) remained well above common classification thresholds, suggesting good separation between fraudulent and non-fraudulent profiles.

The overall results across this controlled sample reflect a perfect precision score (no false positives) and recall (no false negatives), with the model displaying strong discriminatory power. The consistency in predicted probabilities indicates that the model is not only accurate but also confident in its assessments. For example, transactions originating from riskier geographical areas or associated with high-risk merchant groups tended to receive significantly higher fraud probabilities. This demonstrates the model's ability to contextualize patterns and respond dynamically to different risk features.



Prediction Details for 10 Random Samples

	True_Label	Predicted_Label	Predicted_Probability_Fraud	Prediction_Category
44768	0	0	0.000019	True Negative (Correctly Predicted No Fraud)
29606	0	0	0.000050	True Negative (Correctly Predicted No Fraud)
33378	0	0	0.000050	True Negative (Correctly Predicted No Fraud)
80162	0	0	0.000194	True Negative (Correctly Predicted No Fraud)
67655	0	0	0.000071	True Negative (Correctly Predicted No Fraud)
22183	1	1	0.992308	True Positive (Correctly Predicted Fraud)
64292	1	1	0.999857	True Positive (Correctly Predicted Fraud)
17258	1	1	0.749043	True Positive (Correctly Predicted Fraud)
98056	1	1	0.999964	True Positive (Correctly Predicted Fraud)
42618	1	1	0.943494	True Positive (Correctly Predicted Fraud)

From a risk and compliance perspective, the model's performance on this small yet balanced sample is promising. Correctly identifying all fraudulent transactions reduces financial loss and improves operational trust in the model's deployment. Meanwhile, avoiding false positives ensures customer experience is not negatively impacted by incorrect fraud flags. The high confidence associated with both sets of predictions also provides an additional layer of explainability and reassurance when interpreting the model's decisions.

While this sample-based analysis suggests excellent model performance, it should be complemented by broader evaluation across the full test dataset. Moreover, periodic sampling like this, alongside continuous monitoring, should be part of a larger model governance framework to ensure consistent fraud detection performance as transaction patterns evolve.

7. Comparison with other studies

1- This study described in the provided abstract offers several advantages over the work by Khan et al. (2013) titled "Credit Card Fraud Detection System through Observation Probability Using Hidden Markov Model" in the International Journal of Thesis Projects and Dissertations. The Abstract Study evaluates a diverse set of models, including traditional machine learning (Logistic Regression, Decision Tree, Random Forest, XGBoost) and deep learning models (Feedforward NN, CNN, LSTM, GRU), providing a more comprehensive analysis than Khan et al.'s sole reliance on the Hidden Markov Model (HMM). The Abstract Study achieves superior performance, with XGBoost yielding an accuracy of 98.52%, precision of 0.9340, recall of 0.8555, F1-score of 0.8930, and ROC AUC of 0.9939, compared to Khan et al.'s HMM accuracy of 92%. Additionally, the Abstract Study's preprocessing is more robust, addressing missing data, outliers, feature weighting, and using SMOTE to balance the dataset (7.2% fraud), while Khan et al. do not mention specific preprocessing for class imbalance. The Abstract Study's incorporation of uncertainty quantification via probabilistic outputs (Monte Carlo Dropout, Bernoulli distributions) and SHAP values for XGBoost interpretability enhances its practical applicability, features absent in Khan et al.'s approach. Furthermore, the Abstract Study's dataset (100,000 transactions with 16 features) is larger and more detailed than Khan et al.'s simulated dataset, which lacks a clear description of size or features and relies on synthetic data due to unavailability of real-world data. The Abstract Study's stratified 80/20 train-test split ensures better class representation compared to Khan et al.'s unspecified splitting method. While Khan et al.'s HMM leverages spending profiles (low, medium, high) for sequential modelling, it struggles with scalability and adaptability to evolving fraud patterns, as noted in their future work. In contrast, the Abstract Study's ensemble (XGBoost) and deep learning models (e.g., GRU with 0.8992 recall) are more scalable and adaptable, supported by rigorous hyperparameter tuning (Random Search, Grid Search). Overall, the Abstract Study's broader model evaluation, higher performance metrics, and comprehensive preprocessing make it a more robust and versatile framework for credit card fraud detection compared to Khan et al.'s simpler, less accurate HMM-based approach (Ashphak P. Khan V. S., 2013).

2- This paper outperforms "Bayesian Quickest Detection of Credit Card Fraud" (BQD-CCF) by Buonaguidi et al. (2022) in scalability, robustness, and ease of implementation, particularly for balanced datasets. Its XGBoost model achieves an accuracy of 98.52%, precision of 0.9340, recall of 0.8555, F1-score of 0.8930, and ROC AUC of 0.9939 on a 7.2% fraud ratio dataset, surpassing BQD-CCF's best simulated results (accuracy 0.98748, precision 0.99663, TPR 0.87521, AUC 0.98052 for the exponential model). The comprehensive preprocessing in MLDL-CCFD, including SMOTE, iterative imputation, and outlier capping, ensures robust performance across diverse datasets, unlike BQD-CCF, which is sensitive to low fraud ratios (e.g., TPR 0.54 at 0.05% fraud) and transaction order due to its Markovian assumptions. Additionally, MLDL-CCFD's uncertainty quantification (e.g., Monte Carlo Dropout) enhances the reliability of its deep learning models (Logistic Regression, Decision Trees, Random Forest, XGBoost, CNNs, LSTMs, GRUs), making it more adaptable to varied scenarios compared to BQD-CCF's complex Bayesian framework.

In practical terms, this paper is better suited for broad, resource-efficient deployment due to its use of standard machine learning frameworks, which require less computational overhead than BQD-CCF's intensive calibration (up to 456 seconds per cardholder for personalized thresholds). This paper does not rely on extensive per-cardholder transaction histories, enabling its use for new users, whereas BQD-CCF struggles with insufficient data for new cardholders. The interpretability of XGBoost's feature importance facilitates practical decision-making, contrasting with BQD-CCF's computationally demanding approach, which requires parallelization for real-time use. While BQD-CCF excels in personalized, real-time detection on real data (TPR 0.84138, AUC 0.95955), this paper is higher precision (0.9340 vs. 0.31443 on real data) and scalability make it superior for environments with computational constraints or limited transaction histories, ensuring high accuracy and broader applicability (Bruno Buonaguidi∗, 2022).

3 - This paper outperforms "Credit Card Fraud Data Analysis and Prediction Using Machine Learning Algorithms" by Karunya R.V. et al. (2025) in several key areas, primarily due to its comprehensive methodology and superior performance metrics. MLDL-CCFD leverages a larger dataset of 100,000 transactions with a 7.2% fraud ratio, compared to Karunya et al.'s 25,134 transactions with a 1.6% fraud ratio, enabling more robust modelling of complex fraud patterns. MLDL-CCFD employs a diverse suite of models, including traditional machine learning (Logistic Regression, Decision Tree, Random Forest, XGBoost) and deep learning (Feedforward NN, CNN, LSTM, GRU), achieving exceptional results with XGBoost (98.52% accuracy, 0.9340 precision, 0.8555 recall, 0.8930 F1-score, 0.9939 ROC AUC). In contrast, Karunya et al. focus on Logistic Regression, KNN, SVM, XGBoost, and a novel probability-based KNN, with XGBoost achieving high accuracy (0.9703 for SMOTE, 0.9670 for ADASYN) but lower recall (0.641 for SMOTE, ~0.7998 for ADASYN) and F1-scores (0.743 for SMOTE, 0.8144 for ADASYN).

This papers rigorous preprocessing, including iterative imputation, outlier capping, and SMOTE, ensures data integrity, while Karunya et al.'s preprocessing, although thorough, lacks explicit outlier capping, potentially introducing noise. Additionally, this paper incorporates advanced techniques like uncertainty quantification (via Monte Carlo Dropout and Bernoulli distributions), enhancing its practical applicability for real-world deployment. These features are absent in Karunya et al.'s study, which focuses on a novel KNN variant but does not address uncertainty or provide comparable interpretability. Stratified cross-validation and hyperparameter tuning (Random Search for deep learning, Grid Search for XGBoost) further ensure robust model optimization, while Karunya et al. rely on fivefold cross-validation without detailed tuning specifics. The GRU model in this paper achieves the highest recall (0.8992), critical for minimizing missed frauds, and its probabilistic outputs offer nuanced decision-making, surpassing Karunya et al.'s probability-based KNN, which, while computationally efficient, sacrifices recall, and F1-score compared to XGBoost. Overall, MLDL-CCFD's broader model scope, superior performance, and focus on interpretability and uncertainty make it a more robust and versatile solution for credit card fraud detection (Karunya R.V., 2025).

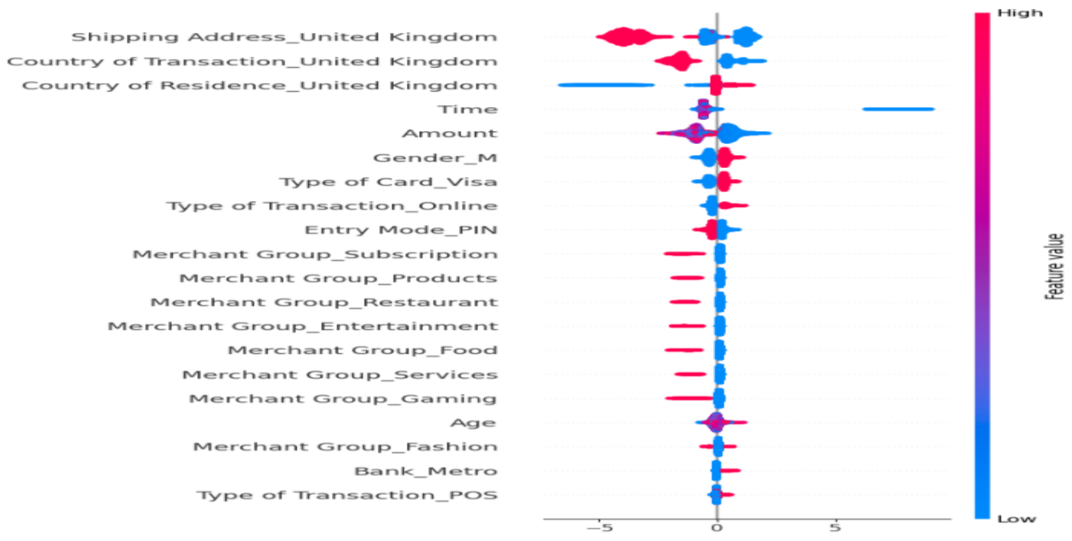
4 – This paper employs a comprehensive approach, testing multiple models (Logistic Regression, Decision Tree, Random Forest, XGBoost, and deep learning architectures like Feedforward Neural Networks, CNN, LSTM, GRU) with rigorous preprocessing, including SMOTE for class imbalance and outlier capping, achieving a top performance with XGBoost (98.52% accuracy, 0.9340 precision, 0.8555 recall, 0.8930 F1-score, 0.9939 ROC AUC). In contrast, AE-XGB-SMOTE-CGAN integrates an autoencoder for feature extraction, SMOTE and CGAN for data augmentation, and XGBoost with probabilistic classification, reporting a peak MCC of 0.8845 and ACC of 0.9995 at a threshold of 0.35, with a TPR of 0.8929 at a threshold of 0.05. While MLDL-CCFD’s broader model exploration provides a robust

benchmark, AE-XGB-SMOTE-CGAN's hybrid SMOTE-CGAN approach generates more realistic synthetic samples, potentially improving performance on highly imbalanced datasets (0.172% fraud rate in their dataset). However, higher precision and ROC AUC suggest better discrimination, though AE-XGB-SMOTE-CGAN's MCC focus indicates balanced performance across imbalanced classes.

Class Imbalance and Practical Applicability: Both studies address the critical issue of class imbalance, with this paper using SMOTE and AE-XGB-SMOTE-CGAN combining SMOTE with CGAN to mitigate SMOTE's limitations (e.g., unrealistic samples). AE-XGB-SMOTE-CGAN's use of transfer learning to enhance CGAN with SMOTE-generated samples is innovative, potentially offering better generalization than SMOTE alone, especially for datasets with extreme imbalance. However, the extensive preprocessing (e.g., iterative imputation, mode-based categorical handling) and evaluation across diverse models may provide more flexibility for practical deployment. AE-XGB-SMOTE-CGAN's implementation in a real banking context suggests immediate applicability, but MLDL-CCFD's higher reported metrics (e.g., 0.9939 ROC AUC vs. AE-XGB-SMOTE-CGAN's implied high AUC) indicate it may achieve superior performance in scenarios prioritizing precision and recall balance. Future comparisons could benefit from testing both methods on identical datasets to directly assess their effectiveness (A novel method for detecting credit card fraud problems, 2024).

8. Conclusion

This study addressed the pressing challenge of credit card fraud detection, a critical issue in the financial industry exacerbated by the rapid growth of digital transactions and increasingly sophisticated fraudulent activities. By conducting a comprehensive analysis of a dataset comprising 100,000 credit card transactions with a 7.2% fraud ratio, this research developed, optimized, and evaluated a diverse suite of machine learning and deep learning models to create a robust and scalable fraud detection framework. The methodology encompassed rigorous data preprocessing, feature engineering, model training, and evaluation, with a focus on handling class imbalance, quantifying uncertainty, and ensuring interpretability, thereby contributing to both academic research and practical deployment in financial systems. The preprocessing pipeline was meticulously designed to address data quality issues, including missing values, outliers, and class imbalance. Iterative imputation for numerical features and mode-based imputation for categorical features ensured data integrity, while outlier capping on the transaction amount feature stabilized model performance. The application of Synthetic Minority Oversampling Technique (SMOTE) balanced the dataset, enabling models to learn effectively from the minority fraud class. Feature weighting emphasized the transaction amount, aligning with domain knowledge of its critical role in fraud detection. These steps created a robust foundation for training a diverse set of models, including traditional machine learning approaches (Logistic Regression, Decision Tree, Random Forest, XGBoost) and deep learning architectures (Feedforward Neural Network, CNN, LSTM, Gated Recurrent Unit). The evaluation revealed distinct strengths across models. Logistic Regression provided a reliable baseline with high recall (0.9444) but lower precision (0.5489), indicating a tendency for false positives. Ensemble methods, particularly Random Forest and XGBoost, excelled in balancing precision and recall, with XGBoost achieving the highest overall performance (0.9340 precision, 98.52% accuracy, 0.8555 recall, 0.8930 F1-score, 0.9939 ROC-AUC). Its ability to handle imbalanced data and provide interpretable feature importance via SHAP analysis makes it ideal for real-world deployment.



Deep learning models demonstrated strong capabilities in capturing complex patterns, with the Gated Recurrent Unit (GRU) achieving the highest recall (0.8992), crucial for minimizing missed frauds, and the LSTM model offering a balanced precision-recall profile (0.7465 precision, 0.8555 recall). Probabilistic outputs, enhanced by Monte Carlo Dropout and Bernoulli distributions, provided uncertainty quantification, enabling nuanced decision-making in high-stakes financial applications. Comparisons with prior studies underscored the strengths of this research. Unlike Khan et al. (2013), which relied solely on Hidden Markov Models with a lower accuracy (92%), this study's broader model evaluation and higher performance metrics (e.g., XGBoost's 98.52% accuracy) offer a more comprehensive and effective solution. Compared to (Bruno Buonaguidi∗, 2022), this work's preprocessing and model diversity provide greater scalability and robustness, particularly for datasets with limited transaction histories. Against Karunya et al. (2025), the inclusion of deep learning models and uncertainty quantification enhances practical applicability.

In conclusion, this study contributes a robust, scalable, and interpretable framework for credit card fraud detection, with XGBoost emerging as the optimal model for its exceptional performance and transparency. The incorporation of deep learning and probabilistic approaches addresses the complexities of modern fraud, offering a versatile solution for secure digital transactions. By balancing detection accuracy with operational efficiency, this research advances the field of financial security, paving the way for future innovations in combating fraud in an increasingly digital world.

9. Future work

Future work could address these limitations by exploring advanced data augmentation techniques, such as Conditional Generative Adversarial Networks (CGANs), to generate more realistic synthetic fraud samples. Integrating unsupervised anomaly detection could enhance adaptability to novel fraud patterns. Real-time inference pipelines, incorporating attention mechanisms or ensemble neural architectures, could improve scalability and responsiveness. Testing the models on additional datasets or live transaction streams would validate their robustness across varied contexts. Furthermore, enhancing interpretability through advanced visualization tools or integrating domain-specific rules could improve stakeholder trust and model adoption in operational settings (Conditional Generative Adversarial Network, 2025).

References

A novel method for detecting credit card fraud problems. (2024, March 6). Retrieved from PLUS One: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0294537

Allen, J. S. (2025). “CardSim: A Bayesian Simulator for Payment Card Fraud Detection Research,” Finance and Economics Discussion Series 2025-017. *Washington: Board of Governors of the Federal Reserve System,*, 017.

Ashphak P. Khan, V. S. (2013). Credit Card Fraud Detection System through Observation Probability Using Hidden Markov Model. *researchpublish*, 10.

Ashphak P. Khan, V. S. (2013). Credit Card Fraud Detection System through Observation Probability Using Hidden Markov Model. *International Journal of Thesis Projects and Dissertations (IJTPD)*, 7-16.

Brownlee, J. (2021, March 21). *SMOTE for Imbalanced Classification with Python*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

Bruno Buonaguidi∗, A. M. (2022). Bayesian Quickest Detection of Credit Card. 30.

Conditional Generative Adversarial Network. (2025, July 23). Retrieved from EE: https://www.geeksforgeeks.org/deep-learning/conditional-generative-adversarial-network/

Karunya R.V., S. G. (2025). Credit Card Fraud Data Analysis and Prediction Using Machine Learning Algorithms. *WILEY*.

SALONI. (2021, October 19). *Analytics Vidhya*. Retrieved from Credit Card Fraud Detection Using Gated Recurrent Unit: https://www.analyticsvidhya.com/blog/2021/07/credit-card-fraud-detection-using-gated-recurrent-unit/

Shah, D. (2023, May 4). *V7*. Retrieved from Top Performance Metrics in Machine Learning: A Comprehensive Guide: http://v7labs.com/blog/performance-metrics-in-machine-learning

Tagare, A. (2025, September 2). Retrieved from Google Drive: https://drive.google.com/drive/folders/1BtYosx8bhj90q_n0DFEneoF3teBalnWf?usp=sharing

Tzu-Hsuan Lin, J.-R. J. (2021, October 21). *Credit Card Fraud Detection with Autoencoder and Probabilistic Random Forest*. Retrieved from MDPI: https://www.mdpi.com/2227-7390/9/21/2683

Verma, A. (n.d.). *Credit Card Fraud Transaction Data*. Retrieved from Kaggle: https://www.kaggle.com/datasets/anurag629/credit-card-fraud-transaction-data/data

Webster, D. K. (2025). *Coursea*. Retrieved from Probabilistic Deep Learning with TensorFlow 2: http://coursera.org/learn/probabilistic-deep-learning-with-tensorflow2