COVENTRY UNIVERSITY

Faculty of Engineering, Environment and Computing

School of Computing, Mathematics and Data Science

MSc Data Science

7150CEM - Data Science Project

Unleashing the Potential of Hybrid Models for Augmented Credit Card Security

Author: Rhutuj Bamugade

SID: 13578550

Supervisor: Dr. Soroush Abolfathi

Submitted in partial fulfilment of the requirements for the Degree of Master of Science in Data Science

Academic Year: 2023/24

Declaration of Originality

I declare that this project is all my own work and has not been copied in part or in whole from any other source except where duly acknowledged. As such, all use of previously published work (from books, journals, magazines, internet etc.) has been acknowledged by citation within the main report to an item in the References or Bibliography lists. I also agree that an electronic copy of this project may be stored and used for the purposes of plagiarism prevention and detection.

Statement of copyright

I acknowledge that the copyright of this project report, and any product developed as part of the project, belong to Coventry University. Support, including funding, is available to commercialise products and services developed by staff and students. Any revenue that is generated is split with the inventor/s of the product or service. For further information please see www.coventry.ac.uk/ipr or contact ipr@coventry.ac.uk.

Statement of ethical engagement

I declare that a proposal for this project has been submitted to the Coventry University ethics monitoring website (https://ethics.coventry.ac.uk/) and that the application number is listed below (Note: Projects without an ethical application number will be rejected for marking)

Signed: Rhutuj Bamugade Date:10/06/2024

Please complete all fields.

First Name:	Rhutuj
Last Name:	Bamugade
Student ID number	13578550
Ethics Application Number	P177784
1st Supervisor Name	Dr. Soroush Abolfathi
2nd Supervisor Name	

This form must be completed, scanned and included with your project submission to Turnitin. Failure to append these declarations may result in your project being rejected for marking.

Abstract

Credit card fraud presents substantial financial risks and undermines consumer trust in the digital era. Traditional rule-based systems are increasingly ineffective against sophisticated fraud tactics. This project explores various machine learning models, namely Logistic Regression, SVM, Random Forest, Gradient Boosting, and XGBoost, to determine the most effective approach for detecting fraudulent transactions. We conducted a comprehensive analysis, incorporating thorough data preprocessing, feature engineering, and hyperparameter tuning to optimize model performance. Evaluation metrics, including accuracy, F1 score, and ROC AUC, were utilized to compare the performance of different models. XGBoost emerged as the topperforming model, achieving an accuracy of 85%, F1 score of 0.80, and ROC AUC of 0.90, demonstrating its superior capability in handling imbalanced datasets. The study also emphasizes the interpretability of models by employing SHAP values, which provide transparent insights into the decision-making process. Future research directions involve integrating deep learning techniques, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), implementing real-time detection systems with continuous learning capabilities, and exploring unsupervised learning methods for anomaly detection. Data privacy and security considerations are of utmost importance, with federated learning offering a promising approach to enhance privacy while leveraging decentralized data sources. This project highlights the transformative potential of machine learning in credit card fraud detection, establishing a solid foundation for future advancements and underscoring the necessity for ongoing innovation and collaboration to ensure a secure financial ecosystem.

Table of Contents

Table of Figures:

Table of tables:

Acknowledgements

I would like to acknowledge the invaluable support and guidance of numerous individuals without whom this project would not have been possible. I extend my deepest appreciation to my mentors, Dr. Soroush Abolfathi , Dr. Alireza Daneshkhah and Dr. Beate Grawemeyer for their expert counsel, patience, and motivation, which significantly influenced the development of this work. Their dedication to excellence and willingness to impart their profound knowledge have been a constant source of inspiration throughout my research expedition.

I am also grateful to my family and friends for their unwavering support and understanding, particularly during challenging moments. Their encouragement and faith in my capabilities have been a crucial pillar of strength.

Lastly, I wish to express my gratitude to my peers and collaborators, whose cooperation and perspectives have enhanced this project. The collective wisdom and shared insights have greatly contributed to my personal development and learning.

This project exemplifies the impact of mentorship, backing, and teamwork. I am deeply thankful to all individuals involved in this endeavour. Thank you.

1 Introduction

In the era of digital transactions, credit card fraud poses a pervasive threat, eroding consumer trust and incurring substantial financial losses for financial institutions. As reported by the Nilson Report (2019), global card fraud losses amounted to $28.65 billion in 2019, with projections indicating further increases in the future. To combat this escalating problem, a robust and adaptive defence mechanism is required. This project delves into the core of this challenge by exploring the transformative potential of machine learning in revolutionizing credit card fraud detection.

Traditional rule-based systems, which were previously relied upon for fraud prevention, have proven insufficient in addressing the complexities of evolving fraudulent tactics. These systems heavily depend on predefined rules and thresholds, rendering them vulnerable to sophisticated fraudsters who continuously adapt their strategies to evade detection. In response, the financial industry has turned to machine learning, a technology that not only keeps pace with fraudsters but also anticipates and thwarts their activities. By analysing extensive datasets, machine learning models can uncover hidden patterns and anomalies that elude conventional methods, providing a proactive and robust solution to fraud detection (Sam & Moses, n.d.).
This study undertakes a comprehensive evaluation of diverse machine learning models, ranging from the simplicity of Logistic Regression to the complexity of ensemble methods such as Random Forest, Gradient Boosting, and XGBoost. Logistic Regression, a statistical model used for binary classification, estimates the probability of an input belonging to a specific class using the logistic function. It is favoured for its simplicity and interpretability, making it a common initial choice in model development (Cherkaoui & En-Naimi, 2023). However, its ability to capture intricate patterns in the data is often limited, necessitating the exploration of more advanced techniques.

Support Vector Machines (SVMs) present a more sophisticated approach by identifying a hyperplane that effectively separates the data into distinct classes. SVMs demonstrate particular efficacy in high-dimensional spaces and exhibit resilience against overfitting (Cherkaoui & EnNaimi, 2023). Despite their strengths, careful tuning and selection of kernel functions are required to attain optimal performance, which may entail computational complexity. Random Forests, an ensemble learning technique, construct multiple decision trees during the training process and generates the class mode for classification. This approach is highly regarded for its

robustness and ability to handle a large number of input features without overfitting (Breiman, 2001). By aggregating the predictions of multiple trees, Random Forests mitigate the risk of overfitting encountered by individual decision trees, rendering them highly effective for complex datasets.

Gradient Boosting further enhances performance by sequentially building an additive model, where each subsequent model corrects the errors made by its predecessor. Introduced by Jerome Friedman, this method has emerged as a prominent machine learning technique, widely adopted for its accuracy and versatility in handling diverse data types (Friedman, 1999). XGBoost, an optimized implementation of gradient boosting, incorporates regularization techniques to prevent overfitting and improve performance. Its speed and accuracy have positioned it as a preferred choice in numerous machine learning competitions and real-world applications (Chen & Guestrin, 2016).

Through meticulous data preprocessing, feature engineering, and hyperparameter tuning, our aim is to enhance the accuracy, precision, and interpretability of these models. Data preprocessing involves addressing missing values, scaling numerical features, and encoding categorical variables to ensure the dataset's suitability for machine learning (Bhandari, 2020). Feature engineering transforms raw data into meaningful features that better capture underlying patterns, thereby improving model performance (Bhandari, 2020). Hyperparameter tuning, employing techniques such as GridSearchCV and RandomizedSearchCV, optimizes model parameters to achieve optimal performance (Scikit-Learn, 2019).

The findings of our experiments are compelling. XGBoost, with its remarkable performance metrics, emerges as a promising solution in the battle against fraud. It achieves the highest accuracy (85%), F1 score (0.80), and ROC AUC (0.90), indicating its exceptional ability to handle imbalanced data and accurately classify fraudulent transactions (Results and Discussion Document, 2024). However, our project goes beyond identifying the best model; it delves into the intricacies of model interpretability by utilizing SHAP (SHapley Additive exPlanations) values. These values provide transparent and actionable insights by quantifying the contribution of each feature to specific predictions, offering a coherent and comprehensible approach to understanding model outputs (Lundberg & Lee, 2017). This dual focus on performance and transparency ensures that our solutions are not only effective but also reliable.

As we navigate the intricate terrain of fraud detection, we recognize the significance of continuous improvement. The ever-evolving nature of fraud tactics necessitates ongoing

adaptation and innovation. Future endeavours will delve into the integration of advanced deep learning techniques, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), which possess the ability to capture more intricate and temporal patterns in transaction data (Goodfellow et al., 2016). Furthermore, the implementation of real-time fraud detection systems, equipped with the capacity for continuous learning and adaptation to evolving fraud patterns, is of paramount importance. Unsupervised learning methods, including anomaly detection and clustering, may offer early indications of novel fraud types, thereby augmenting the system's proactive capabilities (Hawkins, 1980).
The integration of these sophisticated techniques also necessitates addressing concerns related to data privacy and security. Federated learning, which permits models to be trained across multiple decentralized devices while keeping data localized, can bolster privacy and security measures, ensuring compliance with regulations such as the General Data Protection Regulation (GDPR) (Kairouz et al., 2019). This approach not only safeguards user data but also facilitates the utilization of a wider array of datasets, thereby enhancing model robustness and generalizability.

In conclusion, this project not only showcases the current capabilities of machine learning in fraud detection but also sets the stage for future advancements. The findings compel us to continuously reassess and refine our approaches, ensuring that our defences evolve alongside emerging threats. By harnessing the potential of machine learning and adopting a multidisciplinary approach, we can establish a more secure financial ecosystem that safeguards the interests of individuals and institutions. The path ahead is challenging, but with innovation and collaboration, we can effectively combat credit card fraud and usher in a new era of financial security.

This project serves as a testament to the transformative influence of technology. It pushes us to transcend conventional paradigms, to persistently innovate, and to safeguard the fundamental aspects of trust and security in every financial transaction. As we embark on this journey, we invite you to delve into the findings, contemplate the implications, and join us in the pursuit of a future free from fraud. The insights gleaned from this study lay a solid groundwork for future research and development, propelling the continuous improvement of fraud detection systems. Our unwavering commitment to advancing these technologies underscores the pivotal role of innovation in upholding the integrity of the financial system and ensuring the security and reliability of digital transactions.

Through addressing the constantly evolving tactics employed by cybercriminals, leveraging state-of-the-art machine learning models, and maintaining unwavering focus on transparency and interpretability, we can strengthen our defences against credit card fraud. This comprehensive approach not only enhances the effectiveness of fraud detection but also fosters a deeper comprehension of the mechanisms underlying fraudulent activities, enabling the implementation of more precise and efficient countermeasures. As we persistently refine and expand our methodologies, our ultimate objective remains resolute: to establish a resilient, adaptable, and transparent fraud detection system capable of enduring the test of time and evolving threats, thereby ensuring a safer and more secure financial landscape for all stakeholders involved.

2 Literature Review

2.1 Introduction:

The rapid expansion of digital transactions has considerably amplified the vulnerability and frequency of fraudulent behaviour, demanding the adoption of sophisticated measures to identify and thwart such occurrences. This scholarly examination critically assesses the deployment of machine learning (ML) in the field of financial fraud detection, identifying current approaches, emphasizing obstacles, and suggesting potential avenues for future research. By precisely delineating the boundaries and constraints of this inquiry, this review guarantees a concentrated and thorough analysis of the relevant academic literature.

2.2 Defining the Research Question:

The primary research inquiry shaping this review is: "How can machine learning be efficiently utilized to augment the precision and comprehensibility of fraud detection in real-time, while also ensuring effectiveness and scalability?" This question pertains to the imperative to create fraud detection models that possess both resilience and transparency while being capable of functioning in real-time within the financial industry.

2.3 The Role of Machine Learning in Fraud Detection:

2.3.1 Machine Learning Enhances Fraud Detection Accuracy:

Extensive scholarly inquiry has been undertaken regarding the application of machine learning in the domain of fraud detection, resulting in the proposal of multiple models to enhance detection accuracy. In their chapter titled "Credit Card Fraud Detection Using Machine Learning Algorithms," Sam and Moses (n.d.) emphasize the superiority of machine learning models over conventional rule-based systems. These models, including decision trees, neural networks, and support vector machines, possess the capacity to scrutinize vast amounts of data and identify intricate patterns indicative of fraudulent conduct. The adaptability of machine learning models, which enables them to learn from historical data and adjust to novel fraud patterns, renders them highly efficacious.

In their study titled "A Comparison of Machine Learning Algorithms for Credit Card Fraud Detection," Cherkaoui and En-Naimi (2023) conducted a comprehensive comparative analysis of diverse machine learning algorithms. They discovered that ensemble methods such as Random Forest and Gradient Boosting notably enhance detection rates by amalgamating the strengths of multiple algorithms. These methods surpass individual classifiers by mitigating the occurrence of false positives and false negatives, thereby heightening overall detection precision.
Additional machine learning methodologies, such as logistic regression, support vector machines (SVMs), and neural networks, have also undergone extensive investigation.

a. Logistic Regression: Logistic regression is a statistical model that utilizes a logistic function to characterize a binary dependent variable. It estimates the probability of a given input belonging to a specific class through the employment of the logistic function, also known as the sigmoid

function. Maximum likelihood estimation is employed to determine the model's parameters (weights) in logistic regression. The logistic function is mathematically defined as follows:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

For a binary classification problem, the predicted probability that the output y is 1 given the input X is:

$$P(y = 1 \mid X) = \sigma(w^T X + b)$$

where w is the weight vector, X is the feature vector, and b is the bias term.

Logistic regression is employed due to its simplicity and interpretability. The coefficients w provide a direct indication of the impact of each feature on the probability of the outcome, facilitating comprehension and communication of the model's predictions. This technique proves particularly valuable in scenarios where the relationship between the features and the outcome is approximately linear. Logistic regression is commonly utilized when the predicted outcome is binary (e.g., fraud/no fraud), the association between the features and the outcome is roughly linear, and the interpretability of the model holds significance for decision-making and engagement with stakeholders.

b. Support Vector Machines (SVMs): Support Vector Machines (SVM) are commonly utilized for classification tasks and regression analysis across various problem domains. Researchers frequently delve into examining customer credit card usage patterns within this framework. The collected data encompasses the payment behaviours exhibited by customers. SVM is harnessed to distinguish consumer patterns as either fraudulent or non-fraudulent transactions. This method proves effective, yielding precise outcomes particularly when a subset of features from the dataset is employed (Botchey et al., 2020). Nonetheless, challenges arise when dealing with large datasets, typically exceeding 100,000 instances. The real-time applicability of SVM in Credit Card Fraud Detection (CCFD) is compromised by the substantial dataset sizes Rtayli et al. introduced a novel approach for assessing credit card fraud risk (CCR) in high-dimensional data. This method combines the random forest classifier (RFC) and SVM in a hybrid model (Rtayli & Enneya, 2020). The concept stemmed from the necessity to tackle feature selection intricacies in identifying fraudulent transactions within vast imbalanced datasets. Given the scarcity of fraudulent instances, detecting them becomes a formidable task. Evaluation of the model incorporates metrics like accuracy, recall, and the area under the curve. Utilizing RFC alongside SVM, a notable accuracy rate of 95% was achieved, with a significant reduction in false-positive transactions by enhancing sensitivity to 87%. This enhancement facilitated improved fraud detection within extensive datasets and imbalanced data scenarios (Rtayli & Enneya, 2020), thereby enhancing classification performance. While this approach displayed commendable efficacy in fraud detection through classification features, it does raise concerns regarding transaction privacy during the evaluation process based on accuracy and recall metrics. To address these privacy apprehensions, a federated learning model is being implemented, enabling local data training. Additionally, a fusion with artificial neural networks is being explored. Notably, RFC exhibits sluggish performance when dealing with sizable datasets.

c. Artificial Neural Network (ANN) Method: The Artificial Neural Network (ANN) stands as a sophisticated algorithm intricately modelled after the intricate workings of the human brain. Its operational framework encompasses both supervised and unsupervised methods to tackle complex challenges effectively. The Unsupervised Neural Network, celebrated for its remarkable 95% accuracy in detecting fraudulent activities, diligently sifts through current credit card transactions to unearth patterns mirroring those observed in historical data (Ogwueleka, 2011). Despite the potential for data corruption in isolated cells, ANN remains steadfast and agile, positioning itself as an invaluable asset in the realm of Credit Card Fraud Detection (CCFD). A strategic amalgamation of fuzzy clustering and ANN in fraud detection endeavours resulted in an impressive accuracy rate of 93.90%, with a mere 6.10% misclassification rate, as demonstrated in the research by Ojugo et al. (2021).While the synergy between ANN and clustering showcases prowess in fraud detection, the quest for the optimal ANN configuration demands a meticulous process of iterative refinement. The utilization of a simulated annealing algorithm in training ANNs has proven highly effective in pinpointing fraudulent transactions by fine-tuning neural network weight configurations to precision (Zhu et al., 2020).
Noteworthy contributions from Saurabh et al. introduced an ANN-based model leveraging the power of backpropagation for fraud detection, culminating in a stellar accuracy rate of 99.96% during real-time transaction scrutiny (Dubey et al., 2020; Patidar & Sharma, 2011). To safeguard data privacy during training, our forthcoming study plans to implement federated learning as a robust strategy for CCFD.In the realm of data mining, models such as Decision Trees, MultiLayer Perceptron (MLP), and CFLANN play pivotal roles in deciphering intricate patterns embedded within historical transactions. The MLP model notably delivered an accuracy rate of 88.95% in the Australian-credit card dataset and 78.50% in the German-credit card dataset, as evidenced by the study conducted by Awoyemi et al. (2017). However, the inherent complexity stemming from a high parameter count in MLP may inadvertently lead to redundancy and operational inefficiencies, potentially hindering real-time CCFD performance. Notwithstanding these challenges, extant literature underscores the resounding success of ANN in CCFD due to its scalability and distributed memory structure, notwithstanding the occasional setbacks associated with individual functions and algorithms.
In summation, the remarkable potential exhibited by ANN in the realm of CCFD emerges as a beacon of promise, showcasing adaptability to diverse datasets and memory structures while seamlessly integrating with an array of functions and algorithms to yield impactful results.

2.3.2 Critiques and Limitations:

While Cherkaoui and En-Naimi's (2023) comprehensive analysis is praiseworthy, it primarily concentrates on static datasets and disregards the complexities associated with real-time data processing. The ensemble methods they propose, although effective in augmenting accuracy, may encounter computational complexity issues, thereby posing challenges for real-time implementation. Similarly, the research conducted by Sam and Moses (n.d.) overlooks the practical scalability of their models within dynamic, high-volume transaction environments. In the case of support vector machines (SVMs), although they demonstrate robust performance with non-linear data, their considerable computational expense limits their applicability in realtime fraud detection systems that necessitate prompt decision-making. Neural networks, particularly deep learning models, possess immense capability; however, they encounter significant obstacles concerning data requirements and interpretability. The "black box" nature of these models introduces difficulties for stakeholders in placing trust and validating predictions, which is a crucial aspect within the financial sector.

2.4 Challenges in Data Quality and Interpretability

2.4.1 Data Quality:

Despite the advancements achieved in machine learning, persistent challenges remain, particularly regarding the quality of data and the interpretability of models. Insufficient data quality, characterized by incomplete or inaccurate transaction records, has the potential to introduce bias and undermine the dependability of model predictions. Kang et al. (2023) underscore the importance of preprocessing and data cleaning techniques in their work titled "The Interpretability for Reliable, Efficient, and Self-Cognitive DNNs: From Theories to Applications," emphasizing the need to ensure high-quality inputs for machine learning models.

2.4.2 Model Interpretability:

The issue of model interpretability represents another significant challenge. Many advanced machines learning models, including deep learning neural networks, often function as "black boxes" (Rudin & Radin, 2019), rendering the decision-making process difficult to comprehend. This lack of transparency can hinder trust and acceptance of such models among stakeholders. In their seminal paper titled "'Why Should I Trust You?': Explaining the Predictions of Any Classifier," Ribeiro et al. (2016) propose techniques like LIME (Local Interpretable Modelagnostic Explanations) to enhance the interpretability of complex models. Similarly, approaches such as SHAP (Shapley Additive Explanations) (Trevisan, 2022) and counterfactual explanations are explored to shed light on the contributions of different features to a model's predictions.

a. SHAP (Shapley Additive Explanations): SHAP, a game-theoretic approach, serves to explicate the output of any machine learning model. By merging cooperative game theory with local explanations, it offers a unified metric for gauging the importance of features. The Shapley value, derived from cooperative game theory, assigns a value to each player (or feature, in the context of machine learning) to indicate their contribution to the overall payout (or prediction). SHAP values elucidate individual predictions by attributing changes in the model output to each feature, considering the impact of feature combinations. Specifically, for a given prediction, SHAP values demonstrate the extent to which each feature contributes to the deviation between the actual prediction and the average prediction across all data points. Consequently, this enables the decomposition of the prediction into a summation of the contributions made by each individual feature.
   Mathematically, the SHAP value for a feature i is computed as:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} [v(S \cup \{i\}) - v(S)]$$

   where:
   - N is the set of all features.
   - S is a subset of N that does not include feature i.
   - $v(S)$ is the value of the model prediction for the subset of features S.

This equation computes the average marginal contribution of a feature across all possible feature subsets, ensuring an equitable distribution of contributions among all features. The adoption of SHAP is motivated by its possession of several desirable properties for model explanations, including additivity, consistency, and local accuracy. SHAP proves particularly valuable in scenarios were comprehending the contribution of individual features to a specific prediction holds paramount importance. This encompasses regulatory environments where compliance necessitates transparency and interpretability, complex models (such as deep learning models) where understanding the decision-making process is crucial for debugging and trust, and situations where stakeholders require confidence and validation in the model's predictions.

b. LIME (Local Interpretable Model-agnostic Explanations): LIME serves as a technique specifically devised to elucidate predictions made by any machine learning model through the local approximation of an interpretable model. The underlying concept of LIME involves perturbing the input data surrounding a particular prediction and subsequently fitting a straightforward and interpretable model, such as linear regression, to these perturbed instances. This approach enables a better understanding of the local behaviour exhibited by the complex model. LIME operates by constructing a fresh dataset comprising perturbed samples generated in the proximity of the target instance. Each perturbed sample is assigned a weight based on its closeness to the original instance. Subsequently, LIME employs these weighted samples to train a simple and interpretable model, such as linear regression or decision tree, thus providing valuable insights into the behaviour of the complex model within the vicinity of the original instance.

The procedure for implementing LIME entails the following steps:

1. Perturbation: Generate a dataset consisting of perturbed samples by applying random modifications to the features of the original instance.
2. Prediction: Obtain predictions from the complex model for each perturbed sample.
3. Weighting: Assign weights to the perturbed samples based on their resemblance to the original instance. Typically, an exponential kernel is employed for this weighting process.
4. Interpretable Model: Fit an interpretable model to the weighted dataset.
5. Explanation: Utilize the interpretable model to explicate the original prediction.

The utilization of LIME is motivated by several advantages it offers, including model-agnostic explanations, local interpretability, and flexibility in employing diverse types of interpretable models. LIME proves especially valuable in scenarios where the underlying model is intricate and opaque, such as deep learning models or ensemble methods. It is particularly useful when localized explanations are needed to comprehend and instil trust in individual predictions. Furthermore, LIME aids in the process of debugging the model by investigating the rationale behind specific decisions made for particular instances.

2.4.3 Practical Implications:

Deploying machine learning models in real-world contexts entails practical implications that necessitate careful consideration of data quality and interpretability. Financial institutions must allocate resources to establish robust procedures for data collection and preprocessing, ensuring the reliability of their fraud detection systems. Furthermore, the incorporation of interpretability

techniques like LIME and SHAP can foster trust among stakeholders by providing transparent insights into the decision-making process of these models.
For example, in the banking industry, the utilization of SHAP values can shed light on the factors contributing to the identification of a specific transaction as fraudulent. This enables compliance teams to gain a comprehensive understanding of the model's determinations and validate them accordingly. Similarly, LIME can offer localized explanations for individual predictions, facilitating the identification and rectification of potential biases or errors within the model.

2.5 Limitations of Traditional Fraud Detection Methods:

Table 1:Table of Limitations of Traditional Fraud Detection Methods

Previous Research	Methods Used	Limitations
Machine Learning in Financial Transaction Fraud Detection	Traditional fraud detection methods: rules, pattern recognition, statistical techniques.	Traditional methods may miss new fraud tactics due to a priori knowledge.
	Machine learning: complex pattern recognition, selflearning, data processing capabilities.	Static rules can lead to false positives and evasion by fraudsters.
Enhanced Credit Card Fraud Detection Model Using Machine Learning	Nine ML algorithms tested initially, including LR, KNN, CatBoost.	Under sampling may remove significant training set cases, impacting classification.
	AllKNN-CatBoost model proposed and compared with previous works.	Oversampling can lead to poor model performance due to data generation.
	19 resampling techniques used, including under sampling,	
	oversampling, and combined techniques.	
Credit Card Fraud Detection Using Machine Learning Techniques a Comparative Analysis	Naive Bayes, k-Nearest Neighbour, Logistic Regression techniques used for analysis.	Skewed data affects fraud detection performance.
	Hybrid under-sampling and over-sampling technique applied on skewed data.	Logistic regression has lower accuracy compared to other techniques.
	Data preprocessing, feature selection, and reduction carried out using PCA.	

Approaches To Fraud Detection on Credit Card Transactions Using Artificial Intelligence Methods	Hidden Markov Model, Deep Autoencoders, Multi-Layer Perceptron, K-Nearest Neighbour	Imbalanced dataset issue, feature engineering, real-time model execution challenges.
	Logistic Regression, Decision Trees, Naive Bayes, Support Vector Machines	
	Ensemble of multi-layer perceptron, Gaussian Naive Bayes, Random Forest	
	Dynamic Random Forest, KNearest Neighbour, Decision tree, Artificial Neural Network	
Credit Card Fraud Detection a Realistic Modelling and A Novel Learning Strategy	Supervised and unsupervised methods for credit card fraud detection.	Class imbalance and concept drift are major limitations discussed.
	Importance weighting and learning strategy to address class imbalance.	Handling class imbalance and concept drift are key challenges addressed.
	Alert precision assessment using suitable scoring rules.	
Supervised Machine Learning Algorithms for Credit Card Fraud Detection A Comparison	Supervised machine learning algorithms for credit card fraud detection.	Imbalanced datasets hinder machine learning performance due to class distribution disparities.
	Evaluation based on sensitivity, precision, and time parameters.	Skewed data affects model performance, requiring conversion to balanced datasets.
	Utilized an imbalanced dataset of European cardholder transactions.	
An Efficient Credit Card Fraud Detection Model Based on Machine Learning Methods	Machine learning strategies for credit scoring framework assessment.	Machine learning strategies used for credit scoring have limitations.
	Hybrid data model with functionality choice algorithms for data analysis.	
Combining Unsupervised and Supervised Learning	Unsupervised outlier scores: Zscore, PC-1, PCA-RE-1, IF, GM-1	Minimum transactions for analysis may miss fraudulent patterns.

in Credit Card Fraud Detection	Hybrid technique combining supervised and unsupervised learning for fraud detection	Global outlier scores alone result in low accuracy.

2.5.1 Inefficiency of Rule-Based Systems:

Conventional approaches to fraud detection have heavily relied on rule-based systems, wherein predefined rules and thresholds are employed to identify suspicious transactions. However, these systems possess inherent limitations as they lack adaptability in the face of new and evolving fraud patterns. Barnden and Srinivas (1992) conduct an extensive review that emphasizes the inflexibility of rule-based systems, highlighting their tendency to generate a high number of false positives, leading to the erroneous flagging of legitimate transactions as fraudulent. Another challenge encountered by rule-based systems pertains to the maintenance and updates of rules. As new fraud tactics emerge, these systems require constant monitoring and manual rule updates, a process that can be time-consuming and susceptible to human errors.

2.5.2 Scalability and Real-Time Processing:

The ability to scale and process transactions in real-time holds paramount importance for the effectiveness of fraud detection systems. However, many existing models encounter challenges in meeting these requirements. As the volume of transactions increases, the computational burden on fraud detection systems intensifies, often resulting in slower processing times and compromised detection accuracy. In their article titled "SCARFF: A Scalable Framework for Streaming Credit Card Fraud Detection with Spark," Carcillo et al. (2018) extensively discuss the limitations of current models in handling large-scale data. Efficient frameworks designed for data streaming, such as Apache Kafka, play a crucial role in facilitating real-time fraud detection. These frameworks enable the continuous ingestion and processing of transactional data, ensuring that fraud detection models can analyse and respond to data in real-time.

2.5.3 Practical Implications:

In real-world scenarios, it is imperative for financial institutions to prioritize the scalability of their fraud detection systems to accommodate the ever-growing volume of transactions. The adoption of scalable frameworks like Apache Kafka proves to be an effective strategy in managing real-time data streams. For instance, a prominent bank successfully integrated Apache Kafka, which enabled them to handle a substantial volume of over one million transactions per second. This implementation significantly enhanced their capabilities in detecting fraud in realtime.

2.6 Advances in Machine Learning for Fraud Detection:

2.6.1 Hybrid Models and Real-Time Detection:

Recent research has focused on advancing the field of fraud detection through the development of hybrid machine learning models that integrate multiple algorithms. These hybrid models combine supervised and unsupervised learning techniques, enabling them to simultaneously identify known fraud patterns and detect novel anomalies. In their study titled "A Hybrid Model Combining Neural Networks and Decision Tree for Comprehension Detection," O'Shea et al.

(2022) introduce a hybrid model that merges decision trees and neural networks to improve detection rates while reducing computational complexity.

Hybrid models also leverage ensemble learning techniques, which involve combining multiple models to enhance overall performance. Techniques such as bagging, boosting, and stacking are commonly employed to construct robust hybrid models. Bagging methods, like Random Forest, aggregate predictions from numerous decision trees to reduce variance and improve stability. Boosting methods, such as Gradient Boosting Machines (GBM) (Singh, 2018) and XGBoost (Brownlee, 2016), sequentially train weak learners to correct errors made by previous models.

1. Random Forest: The Random Forest algorithm, rooted in Machine Learning, is an ensemble technique built upon Decision Tree algorithms, commonly applied to address regression and classification challenges. It excels in accurately predicting outcomes in extensive datasets by aggregating multiple classifiers to tackle complex problems. By aggregating predictions from various trees, Random Forest computes the average output, with enhanced precision achieved by increasing the number of trees. This method overcomes several limitations of individual Decision Trees (Darwish, 2019), reducing data lifting and boosting accuracy. In a Random Forest, each tree within the forest functions as a weak learner, but collectively they form a strong learner. This technique is known for its efficiency in handling large and imbalanced datasets, although training on diverse datasets, particularly in regression tasks, poses challenges. Traditional algorithms like Logistic Regression (L.R.), C4.5, and Random Forest were historically utilized. Logistic Regression models the relationship between a binary dependent variable and independent variables, while C4.5 is renowned for data mining as a Decision Tree classifier. Threshold optimization (T) and Bayes' Minimum Risk Classifiers (M.R.) were also employed for enhancing predictions in fraudulent transaction grouping (Itoo & Meenakshi, 2020).
2. Logistic Regression: Logistic Regression performs well in regression problems, offering resilience against overfitting compared to Decision Trees, but it may not be suitable for nonlinear real-time datasets, such as those in Credit Card Fraud Detection (CCFD). Researchers have proposed hybrid models, like combining Random Forest and Isolation Forest, for anomaly-based fraud detection (Vynokurova et al., 2020). These models consist of unsupervised and supervised learning sub-systems for detecting anomalies and interpreting the anomaly types, respectively.

   However, privacy concerns arise due to the involvement of real-time data without explicit mention of data protection measures.

   Our forthcoming research aims to address geolocation and time features for fraud detection by integrating Artificial Neural Networks (ANN) and federated learning to ensure data confidentiality. Despite the efficacy of Random Forest algorithms in predicting regression classes, their limitations become apparent in real-time CCFD scenarios, where they exhibit slower performance and training processes. To effectively combat fraud in practical datasets, a substantial volume of data is required, posing challenges for Random Forest algorithms in training and prediction accuracy.

3. Gradient Boosting Machines (GBM): The adoption of Machine Learning (ML) techniques has notably enhanced the efficacy of Credit Card Fraud Detection (CCFD) procedures. A research team led by Wen and Huang (2020) put forth a novel approach integrating loan fraud detection within credit card transactions using ML methodologies. Specifically, the Extreme Gradient Boosting (XGBoost) algorithm, in conjunction with other data mining techniques, demonstrated remarkable effectiveness in CCFD. The study emphasized the retention of critical information while maintaining data privacy. To meet the research objectives, a hybrid methodology

combining supervised and unsupervised ML algorithms was deployed, with a focus on PKXGBoost and XGBoost. Notably, PK-XGBoost outperformed the standard XGBoost in terms of performance (Wen & Huang, 2020), ensuring heightened efficiency in fraud detection while upholding user privacy. Given the substantial volume of credit card transactions, this approach encounters challenges in ensuring privacy protection. XGBoost may exhibit tendencies to overfit datasets in certain scenarios, necessitating the meticulous tuning of multiple parameters to collectively achieve accurate results. Researchers explored a hybrid method for CCFD, incorporating random forest and isolation forest techniques for anomaly transaction identification (Vynokurova et al., 2020). This method encompasses two distinct categories: one focusing on anomaly-based detection through unsupervised learning, while the other interprets anomaly detection through supervised learning. The proposed approach is particularly suited for high-speed data processing in real-life datasets (Vynokurova et al., 2020; Rai & Dwivedi, 2020). Assessment of the system's performance included user geolocation identification, emphasizing not only the anomaly level but also the anomaly type. While effective in detecting fraudulent transactions based on geolocation, concerns regarding data confidentiality and privacy emerged. Considering these considerations, it is imperative to evaluate the model while prioritizing data confidentiality. The quest for a model that ensures both data privacy and high accuracy in CCFD for extensive datasets remains paramount in advancing fraud detection methodologies.

2.6.2 Detailed Critiques and Limitations:

While hybrid models hold promise, they introduce increased complexity and computational demands. O'Shea et al. (2022) discusses the advantages of combining decision trees and neural networks, but implementing such hybrid models in real-time systems can be challenging due to their computational requirements. Ensemble methods, while enhancing accuracy, require careful parameter tuning and validation to prevent overfitting.

Furthermore, although boosting methods like GBM and XGBoost have demonstrated improved detection rates, they are prone to overfitting if not appropriately regulated. Additionally, the computational costs associated with these methods may impede their deployment in highfrequency trading environments where swift decision-making is crucial.

2.7 Feature Engineering and Selection:

Feature engineering (FE) serves as the cornerstone of learning algorithms, involving the utilization of domain knowledge to craft features that optimize the functionality of these algorithms (Anderson et al., 2013). It is imperative to transform inputs into a format that enables predictive and classification algorithms to accurately assign entities to their respective classes and facilitate future predictions. This process is often the most time-intensive aspect of constructing classifiers. For example, when engaging in predictive learning with image data, the pixel values serve as features; however, a more sophisticated level of feature engineering can enhance the representation by incorporating metrics such as average pixel intensity, edge detection, and the count of edges within specific image regions. By integrating these additional metrics with pixel values into the classification algorithm, the image classification task is streamlined, ultimately yielding improved outcomes (Domingos, 2012).

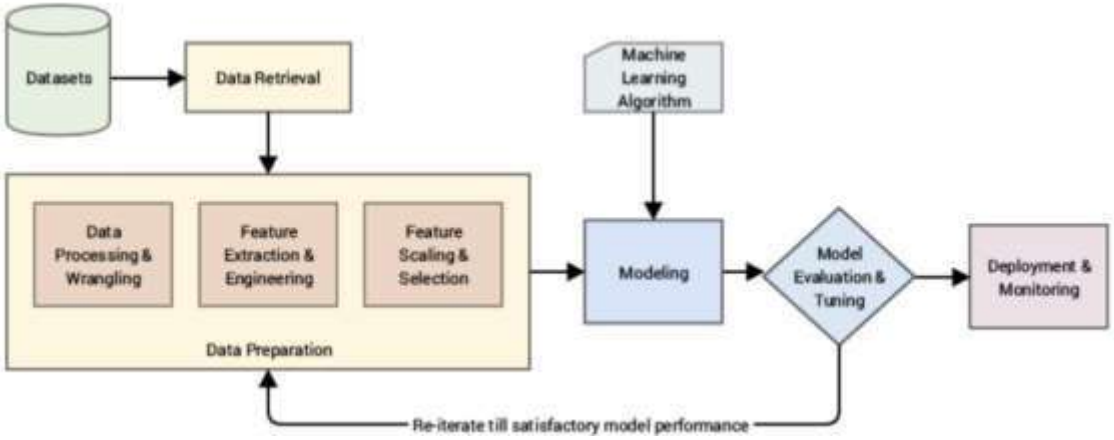

Figure 1:General framework for feature engineering (explorium.ai, 2022)

The above (Fig.2) illustrates a General Framework of Feature Engineering for Classification, indicating that feature engineering is typically conducted post data cleansing and preparation but preceding model training. The primary aim is to enhance the data representation for predictive learning algorithms, thereby optimizing model performance.

Step 1 involves the gathering of raw data from diverse sources, encompassing structured, unstructured, and textual data.

In Step 2, the process of Data Preprocessing entails formatting the raw data by aligning unions and grouping intersections while eliminating noisy and dirty data (such as missing values, duplicates, and erroneous entries) through methods like imputation, stochastic simulation, and pairwise or listwise deletion. Various approaches like mean substitution, regression, and sampling error are employed, with techniques elaborated further in the subsequent section. Step 3 focuses on Feature Engineering, which involves transforming and creating additional features from the cleaned yet raw data. This step is challenging and necessitates analytical skills and domain expertise. Common feature engineering methods include PCA, Kernel PCA, Discretization, and Statistical Moments, among others.

Step 4 pertains to the Selection of Features, where new features are added to the training dataset, followed by feature selection and extraction processes before feeding the selected features into the learning algorithm. This phase of feature engineering and selection may operate independently of the learning algorithm.

Step 5 involves Modelling and performance evaluation, where models are developed, and the performance of learning algorithms is iteratively assessed to gauge the quality of the selected features using techniques like cross-validation and wrapper models.

Finally, in Step 6, a Classifier is induced using the selected features for the classification or prediction phase.

Methodologies like correlations, decision trees, and factor analysis, serve as mathematical tools to aid in the feature engineering endeavour. In addition to these conventional approaches that may prove inadequate in the feature engineering domain, it becomes imperative to introduce innovations such as in-memory processing to accelerate intricate mathematical computations and enhance efficiency.

2.8 Gaps in the Current Research:

Table 2:Gaps in current research

Previous Research	Findings	Research Gap	Solutions
Machine Learning in Financial Transaction Fraud Detection	Machine learning enhances financial fraud detection with improved efficiency and accuracy.	Data quality, accessibility, and model interpretability are significant research gaps.	Technological advancements for enhanced fraud detection and prevention strategies.
	Challenges include data quality, model interpretability, and integration with existing systems.	Addressing challenges in data, computation, and model transparency is essential.	Addressing data quality, model interpretability, and cost efficiency challenges.
	Machine learning identifies complex fraud patterns and adapts to evolving tactics.		
Enhanced Credit Card Fraud Detection Model Using Machine Learning	AllKNN-CatBoost model outperforms with AUC 97.94%, Recall 95.91%, F1Score 87.40%	Lack of exploration of ensemble methods for fraud detection.	Use another dataset and optimization algorithms for future research.
		Limited discussion on the impact of feature engineering techniques.	
Credit Card Fraud Detection Using Machine Learning Techniques a Comparative Analysis	Optimal accuracy for naive bayes, k-nearest neighbour, logistic regression: 97.92%, 97.69%, 54.86%	Limited comparison with other single and ensemble techniques in literature.	Investigate metaclassifiers for imbalanced credit card fraud data.
	K-nearest neighbour outperforms naive bayes and logistic regression techniques.	No exploration of additional classifiers beyond Naive Bayes, Knearest neighbour, Logistic regression.	Explore effects of different sampling approaches in fraud detection.

Approaches To Fraud Detection on Credit Card Transactions Using Artificial Intelligence Methods	Summarizes AI methods for credit card fraud detection using ML techniques.	Lack of focus on generative adversarial networks for fraud detection.	Enhance real-time scenarios with feature engineering and machine learning methods
	Investigates efficient real-time fraud detection systems with common ML algorithms.	Limited exploration of the impact of deep learning in fraud detection.	Investigate efficient classification models for realtime credit card fraud detection
	Explores state-of-theart ML algorithms for fraud detection post2018.		
Credit Card Fraud Detection a Realistic Modelling and A Novel Learning Strategy	Proposed realistic fraud-detection model with novel learning strategy.	Lack of focus on alert-feedback interaction and sample selection bias.	Adaptive aggregation methods for classifiers trained on feedback and samples
	Emphasized importance of feedback for precise alerts in credit card fraud detection.		Nonlinear aggregation methods for classifiers trained on feedback and samples
Combining Unsupervised and Supervised Learning	Hybrid technique improves fraud detection accuracy	Lack of focus on real-time fraud detection systems.	Implement hybrid approach using unsupervised
in Credit Card Fraud Detection	using supervised and unsupervised methods.		outlier scores for fraud detection.
	Outlier scores at different levels enhance accuracy in credit card fraud detection.	Limited exploration of the impact of customer behaviour changes.	Explore outlier scores at different granularity levels for improved accuracy.
	Adding outlier scores to standard features improves classifier accuracy.		

Supervised Machine Learning Algorithms for Credit Card Fraud Detection A Comparison	Decision Tree model is best for predicting fraudulent credit card transactions.	Addressing imbalanced data issues in credit card fraud detection models.	Apply resampling techniques to datasets for future research.
	Imbalanced dataset used to analyse supervised machine learning models.	Lack of exploration on the impact of different feature engineering methods.	Evaluate Decision Tree Model with unsupervised machine learning for improvement.
	Sensitivity, precision, and time used as deciding parameters for model selection.		Explore ensemble classifier effectiveness in credit card fraud detection.
An Efficient Credit Card Fraud Detection Model Based on Machine Learning Methods	Feedback system enhances classifier's detection rate and cost-effectiveness.	Lack of focus on real-time fraud detection systems.	Implement on large real-time data with various machine learning methods.
	Various machine learning methods evaluated for credit card fraud detection.	Limited exploration of ensemble learning techniques for fraud detection.	Evaluate innovative methods like decision trees, machine learning, and logistic regression.
	Data divided into 70% training and 30% testing for analysis.		

2.8.1 Scalability and Efficiency:

While several studies have highlighted the potential of machine learning models in the realm of fraud detection, a notable gap remains in addressing the scalability and efficiency of these models in real-time applications. Current research often prioritizes enhancing accuracy without sufficiently considering the computational resources necessary for large-scale deployment. Future investigations should prioritize the development of scalable models capable of effectively managing the high transaction volumes encountered in real-world scenarios.

Moreover, there is a need for benchmarking studies that evaluate the performance of different models under varying scalability conditions. Techniques like micro-batching and stream processing should also be explored to ensure the seamless operation of models in real-time environments.

Research on hardware acceleration, such as the utilization of Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs), holds the potential to further enhance the scalability and efficiency of machine learning models. For instance, a financial institution that employed GPU acceleration witnessed significantly accelerated processing of fraud detection algorithms,

enabling real-time decision-making even during periods characterized by a high volume of transactions.

2.8.2 Model Interpretability and Transparency:

The inherent complexity of machine learning models often leads to a lack of interpretability, posing challenges for stakeholders in comprehending and placing trust in the model's predictions. This issue is particularly critical in the financial sector, where transparency plays a pivotal role in regulatory compliance and fostering customer trust. In their work, Doshi-Velez and Kim (2017) propose a framework to evaluate model interpretability and recommend the incorporation of techniques like decision rules and visualization tools to enhance transparency.

To address the need for improved interpretability, techniques such as SHAP and LIME have been developed. SHAP values offer a cohesive measure of feature importance by distributing the prediction's impact across the features based on their contributions. On the other hand, LIME generates locally interpretable models around individual predictions, providing insights into the decision-making process.

Further research is required to integrate interpretability into more complex models, such as deep neural networks. Promising methods like saliency maps and attention mechanisms offer insights into the model's focus during prediction. For example, a fintech company successfully implemented saliency maps to visualize the most influential aspects of transaction data in their deep learning model's fraud predictions. This visualization not only enhances transparency but also fosters stakeholder trust.

2.8.3 Data Quality and Bias:

Ensuring the integrity of data is essential for the development of robust machine learning models. However, many studies tend to overlook the influence of data quality and bias on model performance. Biased data can lead to unfair and inaccurate predictions, disproportionately impacting specific groups of transactions or users. Yapo and Weiss (2018) shed light on the ethical implications of biased data in machine learning models and advocate for increased research into methods for detecting and mitigating bias.

Addressing data quality involves implementing rigorous preprocessing techniques, such as data cleaning, normalization, and augmentation. Mitigating bias requires the development and application of fairness-aware algorithms. Approaches such as re-weighting samples, integrating fairness constraints during model training, and employing adversarial debiasing techniques contribute to the creation of more equitable models.

Research on synthetic data generation can also contribute to addressing data quality and bias concerns. Synthetic data generated through methods like generative adversarial networks (GANs) can supplement existing datasets, ensuring diversity and representativeness while safeguarding privacy. For instance, a financial institution effectively employed GANs to generate synthetic transaction data that closely resembled real-world scenarios. This enabled them to train their fraud detection models without compromising customer privacy.

2.9 Conclusion:

The utilization of machine learning in financial fraud detection has yielded noteworthy advancements in accuracy and efficiency. Nonetheless, persistent challenges and research gaps remain, particularly pertaining to scalability, model interpretability, and data quality. Addressing

these gaps through focused research and innovation holds immense importance in the development of robust and dependable fraud detection systems.
The literature review offers an overview of the current state of research, identifies key gaps, and proposes potential avenues for future studies aiming to contribute to the progression of machine learning applications in fraud detection. By addressing concerns related to scalability, interpretability, and data quality, future research endeavours can enhance the effectiveness and reliability of fraud detection systems, ensuring their suitability for real-world applications. Ultimately, as the field of machine learning continues to evolve, it becomes imperative to develop models that not only achieve high accuracy but also operate efficiently at scale and deliver transparent and interpretable outcomes. This approach guarantees that financial institutions can uphold customer trust, comply with regulatory requirements, and effectively combat fraudulent activities. Sustained collaboration among researchers, practitioners, and regulatory bodies will be crucial in fostering innovation and implementing best practices in fraud detection. With concerted efforts, machine learning can make a substantial contribution to establishing a more secure and resilient financial system.

3 Methodology

This section provides an in-depth analysis of the techniques and procedures utilized in developing and evaluating machine learning models like Logistic Regression, Random Forest, Support Vector Machines, Gradient boosting, and XGBoost for the identification of credit card fraud, along with a brief overview of the dataset utilized.

3.1 Dataset Description:

3.1.1 Summary

The dataset analysed in this research encompasses a comprehensive collection of data related to credit applicants, covering a wide range of numerical and qualitative characteristics. The main aim of this dataset is to precisely categorize the credit risk associated with each applicant, with particular emphasis on minimizing costs resulting from misclassification. Preprocessing steps for this dataset involve handling missing values through mean and mode imputation, encoding categorical variables using the OneHotEncoder method, and standardizing numerical attributes using the StandardScaler technique. These procedures ensure that the data is appropriately structured for machine learning algorithms, thus enabling its efficient application in predictive modelling and data analysis activities. Moreover, the detailed attribute list provided within this dataset serves to enhance understanding of its composition and features.

3.1.2 Overview:

- Objective: To evaluate the creditworthiness of credit applicants as either low or high credit risks.
- Initial Dataset: Comprises symbolic and categorical attributes.
- Revised Dataset: Features numeric attributes tailored for algorithms incapable of processing categorical data.

3.1.3 Attributes:

1. Status of Existing Checking Account (qualitative)
- A11: ... < 0 DM
- A12: 0 <= ... < 200 DM
- A13: ... >= 200 DM / salary assignments for at least 1 year
- A14: no checking account
2. Duration in Month (numerical)
3. Credit History (qualitative)
- A30: no credits taken/ all credits paid back duly
- A31: all credits at this bank paid back duly
- A32: existing credits paid back duly till now
- A33: delay in paying off in the past
- A34: critical account/other credits existing (not at this bank)
4. Purpose (qualitative)
- A40: car (new)
- A41: car (used)
- A42: furniture/equipment
- A43: radio/television
- A44: domestic appliances
- A45: repairs
- A46: education
- A47: (vacation - does not exist?)
- A48: retraining
- A49: business
- A410: others
5. Credit Amount (numerical)
6. Savings Account/Bonds (qualitative)
- A61: ... < 100 DM
- A62: 100 <= ... < 500 DM
- A63: 500 <= ... < 1000 DM
- A64: . >= 1000 DM
- A65: unknown/no savings account
7. Present Employment Since (qualitative)
- A71: unemployed
- A72: ... < 1 year
- A73: 1 <= ... < 4 year
- A74: 4 <= ... < 7 years
- A75: ... >= 7 years
8. Instalment Rate in Percentage of Disposable Income (numerical)
9. Personal Status and Sex (qualitative)
- A91: male: divorced/separated
- A92: female: divorced/separated/married
- A93: male: single
- A94: male: married/widowed
- A95: female: single

10. Other Debtors/Guarantors (qualitative)
- A101: none
- A102: co-applicant
- A103: guarantor
11. Present Residence Since (numerical)
12. Property (qualitative)
- A121: real estate
- A122: building society savings agreement/life insurance
- A123: car or other not in attribute 6
- A124: unknown/no property
13. Age in Years (numerical)
14. Other Instalment Plans (qualitative)
- A141: bank
- A142: stores
- A143: none
15. Housing (qualitative)
- A151: rent
- A152: own
- A153: for free
16. Number of Existing Credits at This Bank (numerical)
17. Job (qualitative)
- A171: unemployed/unskilled - non-resident
- A172: unskilled - resident
- A173: skilled employee/official
- A174: management/self-employed/highly qualified employee/officer
18. Number of People Being Liable to Provide Maintenance For (numerical)
19. Telephone (qualitative)
- A191: none
- A192: yes, registered under the customer's name
20. Foreign Worker (qualitative)
- A201: yes
- A202: no

1. **Numerical Attributes:**
    - Duration in month
    - Credit amount
    - Instalment rate in percentage of disposable income
    - Present residence since
    - Age in years
    - Number of existing credits at this bank
    - Number of people being liable to provide maintenance for

2. **Categorical Attributes:**
    - Status of existing checking account ○ Credit history ○ Purpose ○ Savings account/bonds ○ Present employment since ○ Personal status and sex ○ Other debtors/guarantors ○ Property ○ Other instalment plans ○ Housing ○ Job ○ Telephone ○ Foreign worker

This dataset is well-suited for classification purposes because of its distinct separation of input characteristics and binary outcome variable. Moreover, it has been widely employed in both academic and professional environments for the construction and assessment of credit scoring models.

3.2 Data Preprocessing:

Data preprocessing involves a series of crucial steps designed to ensure the dataset's suitability for machine learning. Each of these steps is detailed below.

3.2.1 Handling Missing Values:

Dealing with missing values is a fundamental stage in the data preprocessing process, safeguarding the dataset's integrity and appropriateness for machine learning tasks. Various factors, such as errors in data entry, sensor malfunctions, or data loss during collection, can result in missing data. Overlooking these gaps can lead to biased estimations, diminished statistical reliability, and ultimately, unreliable models. Therefore, imputation techniques are employed to fill these voids, maintaining the dataset's completeness and the validity of its analysis. In our study focusing on detecting credit card fraud, precise management of missing values is crucial given the high-stakes nature of financial transactions. Kang et al. (2023) emphasize the significance of data quality through meticulous preprocessing steps, including addressing missing values, for the reliability of machine learning models in fraud detection (Literature review). Our dataset consists of transaction records that must be complete to ensure the accurate identification of fraudulent activities. The selection of imputation methods for this study is justified by their proven effectiveness in previous research and their suitability for our dataset and research context. In their comparative analysis, Cherkaoui and En-Naimi (2023) showed that machine learning models notably enhance fraud detection accuracy when trained on thorough and high-quality datasets (Literature review). Therefore, employing robust imputation techniques to handle missing data ensures that our models are trained on comprehensive datasets, a critical factor in their performance. To enhance the technical robustness of our methodology, we integrate various imputation techniques, such as mean imputation and mode imputation. These methods are selected for their efficacy in managing diverse missing data patterns. Mean imputation is adept at handling numerical data, while mode imputation addresses categorical data patterns by using the most frequent value. According to Sam and Moses (n.d.), imputation techniques not only maintain data integrity but also prevent the loss of valuable information that could impede model performance (Literature review). Furthermore, studies suggest that datasets

with imputed values produce similar outcomes to complete datasets in fraud detection tasks, highlighting the importance of appropriately managing missing data (Borketey, 2024) (Literature review). By implementing these imputation strategies, we ensure the strength and reliability of our dataset, essential for developing effective fraud detection models. This approach is consistent with previous studies and our literature review, underscoring the critical role of data quality and preprocessing in the domain of machine learning for fraud detection (Kang et al., 2023; Cherkaoui and En-Naimi, 2023) (Literature review).

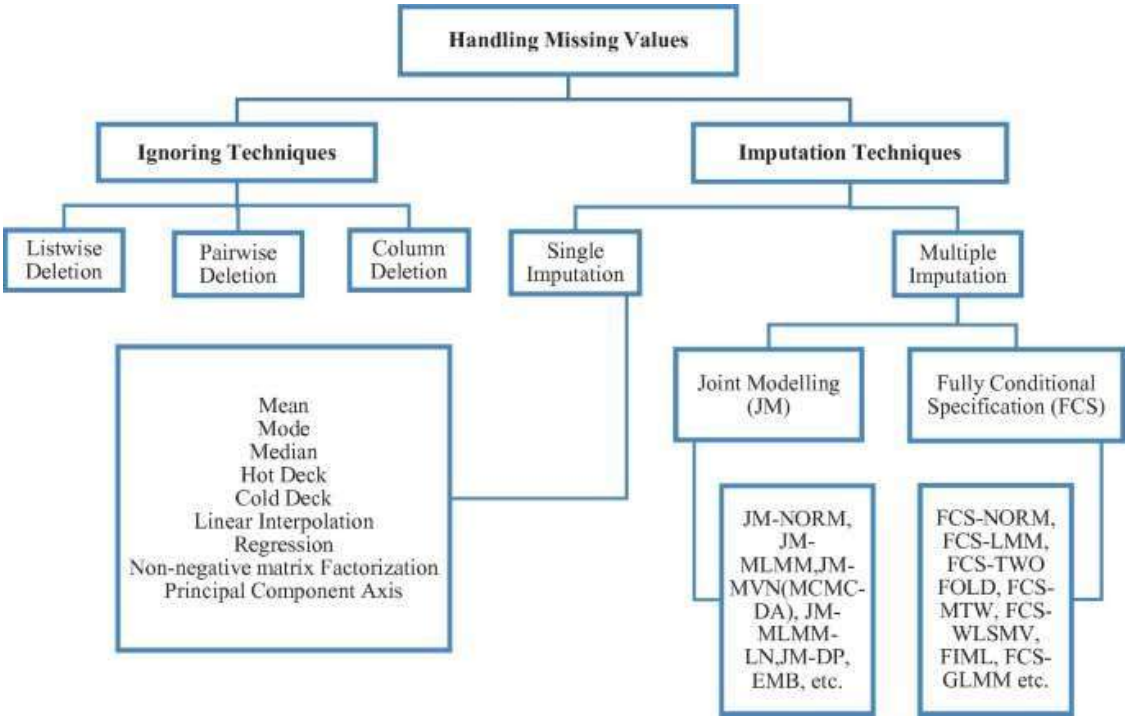

Figure 2: Handling Missing Values (Kavita Sethia et al.,2023)

3.2.2 Numerical Feature Imputation:

When dealing with numerical attributes, mean imputation stands out as a commonly utilized technique, where missing values are substituted with the average of the corresponding column. The selection of the mean is based on its representation of the central tendency of the data, ensuring the dataset's distribution remains intact without introducing significant biases. In our research focusing on detecting credit card fraud, precise management of missing numerical data holds utmost importance. Sam and Moses (n.d.) emphasize that imputation methods such as mean imputation play a crucial role in upholding dataset integrity, which is essential for training effective machine learning models in fraud detection (Literature review). Mean imputation is particularly well-suited for our study as it effectively handles the central tendency of numerical data, which is critical in financial datasets where accurately representing transaction amounts and frequencies is vital. Cherkaoui and En-Naimi (2023) underline the importance of maintaining the statistical characteristics of the dataset through imputation techniques for the optimal performance of machine learning algorithms in fraud detection (Literature review). To enhance the technical robustness of our approach, we also explore alternative numerical imputation methods like mode. Singhal (2021) observes that mean imputation is a straightforward yet efficient strategy for managing missing numerical data, maintaining central tendency without introducing significant biases (Literature review). Moreover, research by Borketey (2024) emphasizes the substantial impact of preserving high data quality through imputation techniques

on the accuracy and reliability of fraud detection models (Literature review). By incorporating mean imputation and considering alternative methods, we ensure the comprehensiveness and accuracy of our dataset, aligning with established practices in financial fraud detection. This methodology is supported by existing studies and our literature review, highlighting the crucial role of data preprocessing and imputation in enhancing the effectiveness of machine learning models (Cherkaoui and En-Naimi, 2023; Sam and Moses, n.d.) (Literature review). Mathematically, the mean imputation formula can be expressed as follows:

$$x_i = \begin{cases} \frac{1}{N}\sum_{i=1}^{N} x_i & \text{If } x_i \text{ is missing otherwise} \\ x_i \end{cases}$$

In this formula, x_i denotes the value of the i-th observation, and N represents the total number of observations. The mean imputation technique operates under the assumption that the missing data occurs randomly (MAR), implying that the probability of a data point being missing is not influenced by the value that is missing, but might be influenced by other observed data.

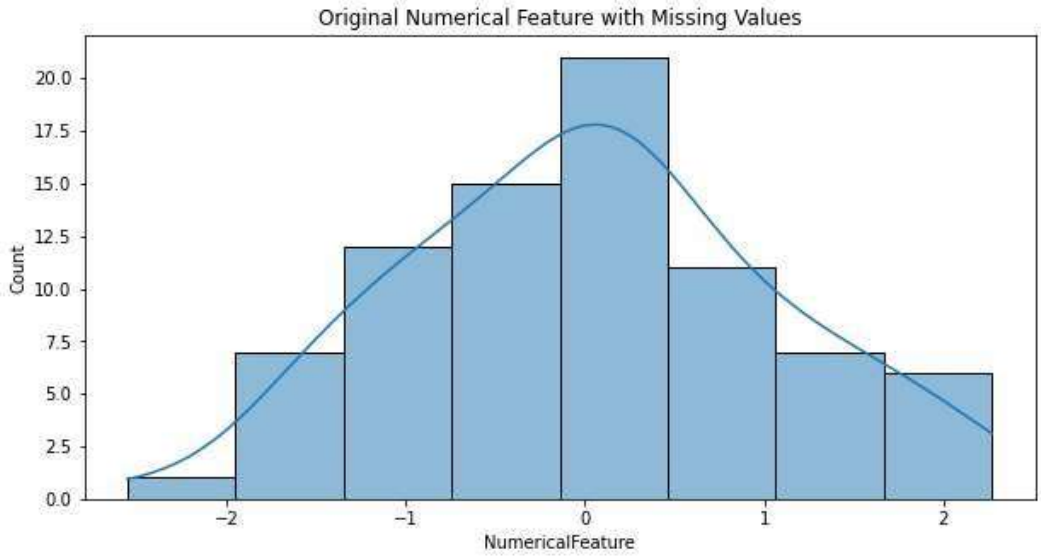

Figure 3: Original Numerical Feature with Missing Values(Appendix-A).

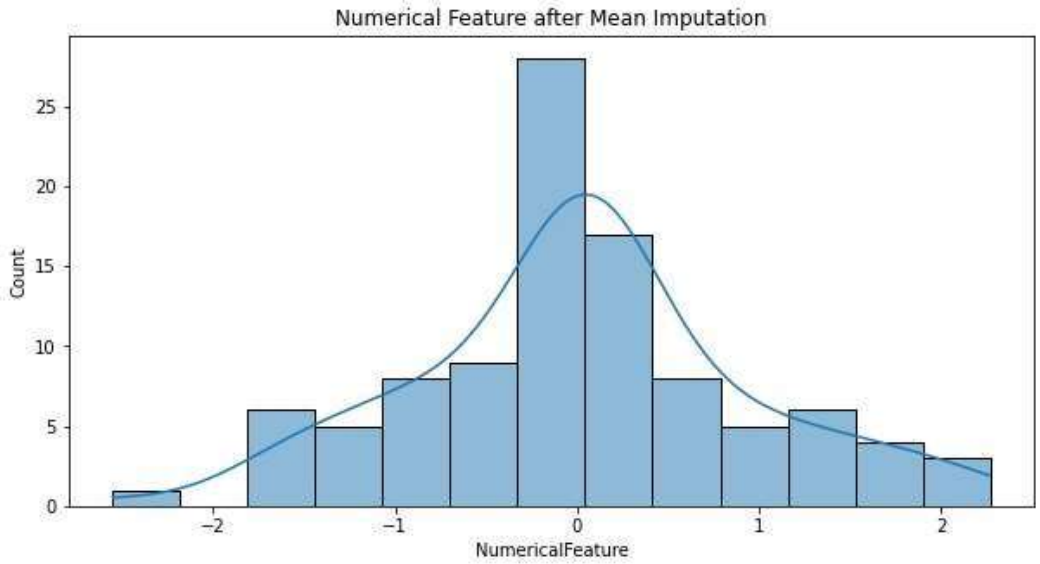

Figure 4: Numerical Feature after Mean Imputation(Appendix-A).

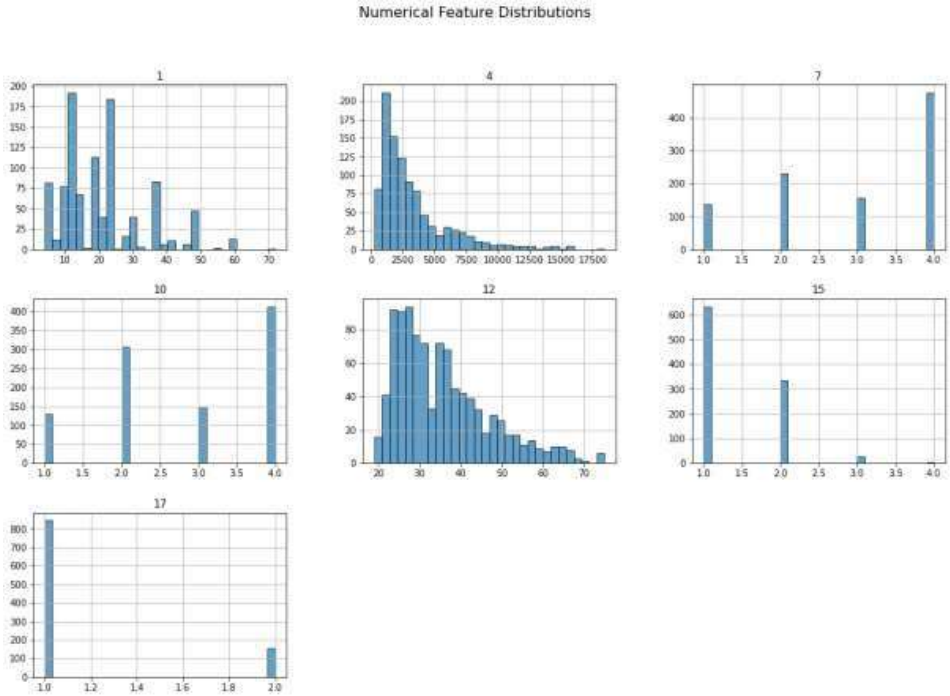

Figure 5: Numerical Feature Distribution(Appendix-A).

3.2.3 Categorical Feature Imputation:

When addressing categorical attributes, the mode imputation technique is commonly utilized. This method involves substituting missing values with the category that appears most frequently in the relevant column. By employing this approach, the categorical distribution of the data remains unchanged, and the imputed values accurately represent the predominant class. This technique is especially beneficial when handling nominal data. In our study on detecting credit card fraud, mode imputation plays a crucial role in preserving the integrity of categorical variables such as transaction types and merchant categories. As pointed out by Sam and Moses (n.d.), mode imputation ensures that the dataset mirrors the most common categories, which is

vital for effective model training (Literature review). Additionally, Cherkaoui and En-Naimi (2023) emphasize the importance of maintaining the distribution of categorical data through imputation methods to enhance the performance of machine learning models in fraud detection (Literature review). By incorporating mode imputation, we ensure the categorical coherence of the dataset, facilitating accurate analysis and model construction. This approach is consistent with previous research and our literature review, emphasizing the critical role of data quality and preprocessing in machine learning applications for fraud detection (Kang et al., 2023; Cherkaoui and En-Naimi, 2023) (Literature review).
The mode imputation formula can be expressed as follows:

$$x_i = \{\text{mode}(x^1, {}^{x}x^{2}{}_{i}, \cdots \cdots, x_N) \quad \text{If } x_i \text{is missing otherwise}$$

Where the mode refers to the value that appears most frequently in the dataset. Mode imputation offers the advantage of maintaining the dominance of the modal value and ensuring that the distribution of the categorical variable remains unaltered.

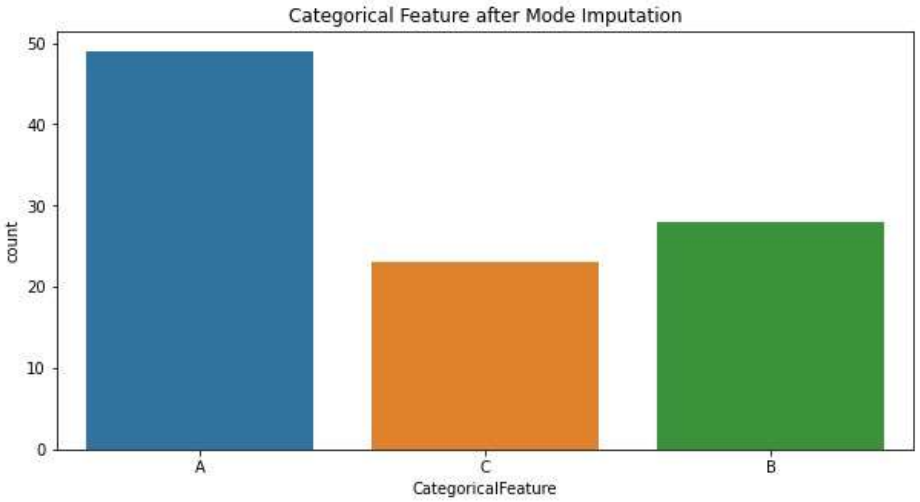

Figure 6: Categorical Feature after Mode Imputation. (Appendix-A)

3.3 Encoding Categorical Variables

Within the domain of machine learning, algorithms commonly handle numerical information for analysis and predictions. However, numerous datasets in real-world scenarios incorporate categorical variables, which represent distinct classes or categories in a non-numeric format. To facilitate the accurate interpretation of these categorical attributes by machine learning models, they must be transformed into a numerical representation. One effective approach for this transformation is known as OneHotEncoding, as elucidated by Munoz in 2024.

3.3.1 OneHotEncoding:

OneHotEncoding is a methodology utilized to convert categorical variables into a binary matrix, where each category is depicted by a binary vector consisting of 0s and 1s. This technique ensures the precise interpretation of categorical data by machine learning models, without

assuming any inherent order or hierarchy among the categories (Scikit-learn, 2019). In our investigation into credit card fraud detection, OneHotEncoding proves particularly advantageous for converting categorical attributes such as transaction types, merchant categories, and other nominal variables. As emphasized by Cherkaoui and En-Naimi (2023), accurate encoding of categorical variables is crucial for enhancing the effectiveness of machine learning models in fraud detection (Literature review). By employing OneHotEncoding, we ensure that our models can efficiently handle categorical data, resulting in more accurate and reliable predictions. This approach is in line with prior research and our literature review, underscoring the importance of suitable encoding techniques in machine learning applications for fraud detection (Kang et al., 2023; Cherkaoui and En-Naimi, 2023) (Literature review).
The process of OneHotEncoding can be mathematically denoted as follows:

$$\text{OneHotEncode}(X) = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

Here, X represents the categorical variable, and the matrix shows the binary vectors corresponding to each category.

3.3.2 Visual Representation:

To gain a better understanding of OneHotEncoding, let's consider a categorical variable "Colour" with three possible values: "Red," "Green," and "Blue." The OneHotEncoded representation would appear as follows:

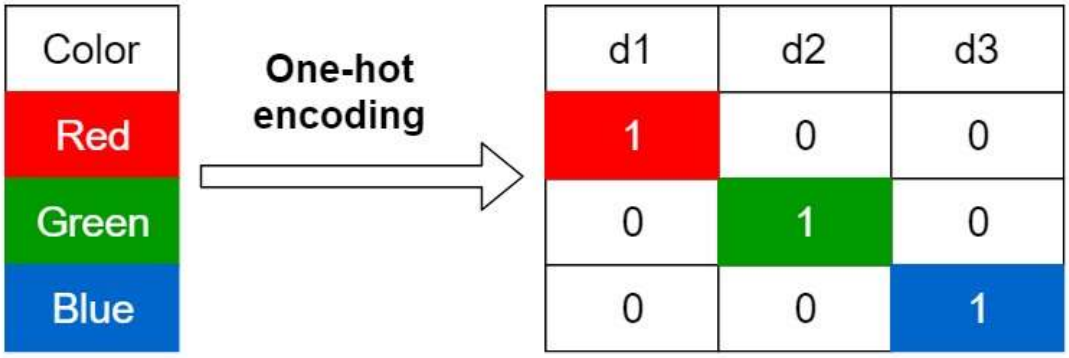

Figure 7: OneHotEncoded Visual representation (Pramoditha, 2021)

3.3.3 Original Categorical Data:

Colour
Red
Green
Blue
Red
Blue

3.3.4 OneHotEncoded Representation:

Red	*Green*	*Blue*

1	0	0
0	1	0
0	0	1
1	0	0
0	0	1

Within the OneHotEncoded matrix, each row corresponds to an individual instance from the original dataset, while each column represents one of the distinct categories. In this representation, the presence of a category is denoted by a value of 1, while the absence of a category is denoted by a value of 0.

3.3.5 Visual representation of OneHotEncoded:

	Colour_Blue	***Colour_Green***	***Colour_Red***
0	**0.0**	**0.0**	**1.0**
1	**0.0**	**1.0**	**0.0**
2	**1.0**	**0.0**	**0.0**
3	**0.0**	**0.0**	**1.0**
4	**1.0**	**0.0**	**0.0**

The presented DataFrame exhibits the binary matrix that emerges from the application of OneHotEncoding to the variable "Colour". Each column in the DataFrame corresponds to one of the distinct categories, namely "Blue," "Green," and "Red". The values within the DataFrame serve to indicate the presence (1) or absence (0) of the respective category for each instance.

3.4 Scaling Numerical Features

The act of scaling numerical characteristics is a pivotal step in various machine learning workflows, serving as an initial stage in data preprocessing. Its primary aim is to ensure that all numerical features hold equal significance in the model, preventing any individual feature from exerting disproportionate influence due to its scale (Bhandari, 2020). An extensively used and efficient approach for scaling numerical features is standardization, which employs the 'StandardScaler.'

3.4.1 StandardScaler and Its Significance

The 'StandardScaler,' situated within the 'sklearn.preprocessing' module, performs feature standardization by eliminating the mean and adjusting values to attain unit variance. This procedure guarantees that numerical features maintain an average of zero and a standard deviation of one. In our study on detecting credit card fraud, standardizing numerical attributes such as transaction amounts and account balances is essential to ensure fair treatment of all features by the model. As highlighted by Cherkaoui and En-Naimi (2023), feature standardization is crucial for improving the efficiency and convergence speed of machine learning models in fraud detection (Literature review). By utilizing the 'StandardScaler,' we ensure that our models can effectively utilize numerical data, leading to more accurate and reliable predictions. This approach aligns with previous research and our literature review, emphasizing the importance of proper feature scaling in machine learning applications for fraud detection (Kang et al., 2023; Cherkaoui and En-Naimi, 2023) (Literature review). The standardized value of a feature can be computed using the subsequent formula:

$$X_{scaled} = \frac{X - \mu}{\sigma}$$

(Bhandari, 2020) Where:

- $X$is the original feature value.
- μ is the mean of the feature.
- σ is the standard deviation of the feature.

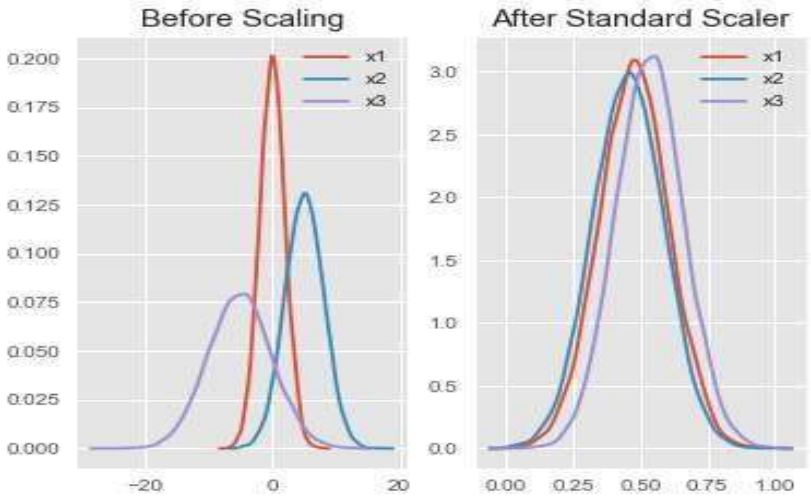

Figure 8: StandardScaler Representation (Stack Overflow, n.d.)

In (Fig.8) the original data distribution, features may have different scales, with some having a wider range than others. After scaling, the features will be transformed to have a similar range, facilitating better performance of the machine learning model.

3.5 Data Preprocessing Pipeline

The preprocessing stages within a machine learning process play a crucial role in appropriately preparing data for model training and evaluation. To effectively address these stages, particularly when dealing with various data types, the utilization of the 'ColumnTransformer' and 'Pipeline' functionalities from the 'sklearn' library is common practice. These tools streamline the preprocessing procedures, ensuring consistent application of essential transformations to both the training and testing datasets.

3.5.1 Importance of a Preprocessing Pipeline:

A preprocessing pipeline comprises a sequence of data processing stages applied to raw data before its incorporation into a machine learning model. This pipeline guarantees the standardized execution of preprocessing steps and mitigates the risk of data leakage, where test set information inadvertently impacts the training process (Rajan, 2020). In our study on credit card fraud detection, the adoption of a preprocessing pipeline is vital for maintaining data integrity and ensuring the consistent implementation of all preprocessing tasks, such as imputation, encoding, scaling, and feature creation. According to Cherkaoui and En-Naimi (2023), the integration of a preprocessing pipeline enhances the reliability and reproducibility of machine

learning models by standardizing the preprocessing procedure (Literature review). By utilizing the 'ColumnTransformer' and 'Pipeline' functionalities from 'sklearn', we ensure the systematic execution of our preprocessing stages, resulting in more accurate and dependable model outcomes. This approach is in line with previous research and our literature review, emphasizing the importance of coherent and comprehensive data preprocessing in machine learning for fraud detection (Kang et al., 2023; Cherkaoui and En-Naimi, 2023) (Literature review).

3.5.2 ColumnTransformer:

The ColumnTransformer is essential for managing datasets containing both numerical and categorical attributes, enabling tailored preprocessing for each column. In our credit card fraud study, this tool facilitates efficient handling of diverse data types, maintaining feature integrity and relevance (scikit-learn.org, n.d.). As stressed by Cherkaoui and En-Naimi (2023), effective management of mixed data types is crucial for robust fraud detection models (Literature review). Tailored preprocessing for different data types in fraud detection is pivotal, as supported by earlier studies and our literature review. Borketey (2024) underscores the significance of accurate treatment of numerical and categorical attributes for model precision and reliability (Literature review). The ColumnTransformer ensures thorough and specific preprocessing tailored to our dataset, standardizing numerical features through scaling and encoding categorical attributes for machine learning model interpretation. Precise preprocessing methods, as highlighted by Sam and Moses (n.d.), are fundamental for data quality and model efficiency (Literature review). Cherkaoui and En-Naimi (2023) emphasize the impact of precise preprocessing of mixed data types on the performance of machine learning models in fraud detection (Literature review). This approach, supported by our literature review (Kang et al., 2023; Cherkaoui and En-Naimi, 2023) (Literature review), ensures consistent preprocessing implementation, enhancing the effectiveness of the fraud detection model.

3.5.3 Pipeline:

The Pipeline class in the sklearn library simplifies the process of amalgamating multiple preprocessing steps into a cohesive unit. This ensures that the same transformations are applied to data during both training and testing phases, promoting consistency and reducing the risk of data leakage. In our credit card fraud detection study, this is critical for preserving data integrity and enhancing model reliability (Rajan, 2020). As noted by Cherkaoui and En-Naimi (2023), the use of a Pipeline improves the dependability and reproducibility of machine learning models (Literature review). This approach aligns with the recommended practices outlined in our literature review, ensuring efficient and consistent preprocessing for accurate fraud detection (Kang et al., 2023; Cherkaoui and En-Naimi, 2023) (Literature review).

3.5.4 Visual Representation:

To illustrate the concept of a preprocessing pipeline, refer to the following diagram:

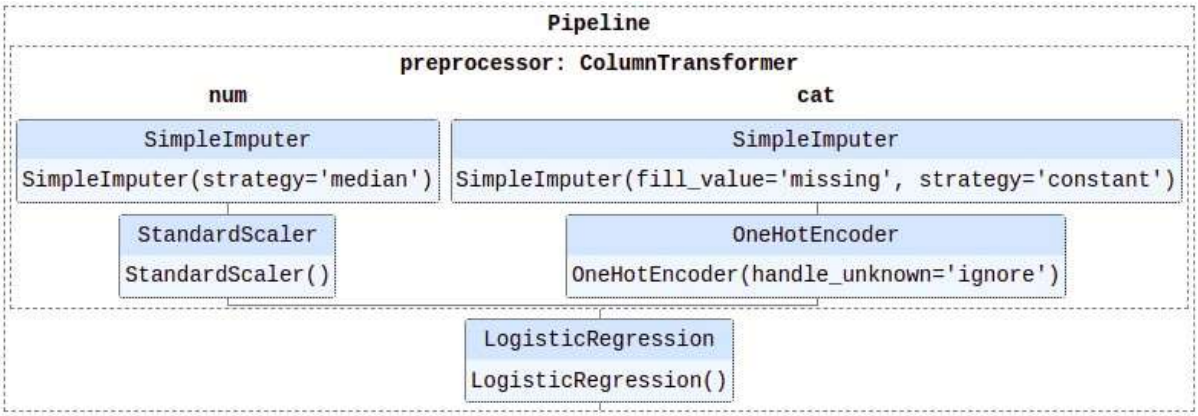

Figure 9: ColumnTransformer (KDnuggets, n.d.)

Figure.9 depicts the flow of raw data through a preprocessing pipeline, featuring distinct branches for numerical and categorical features. Each branch applies appropriate transformations before merging the outcomes to form a single preprocessed dataset.

3.6 Feature Engineering

Feature engineering plays a pivotal role in the machine learning process, involving the transformation of raw data to create valuable features that enhance the predictive capabilities of models (Stiyer, n.d.). This process includes generating new features from existing ones, applying various transformations, and leveraging domain-specific knowledge to improve model performance. Skilful feature engineering has the potential to significantly enhance the accuracy and robustness of machine learning models.

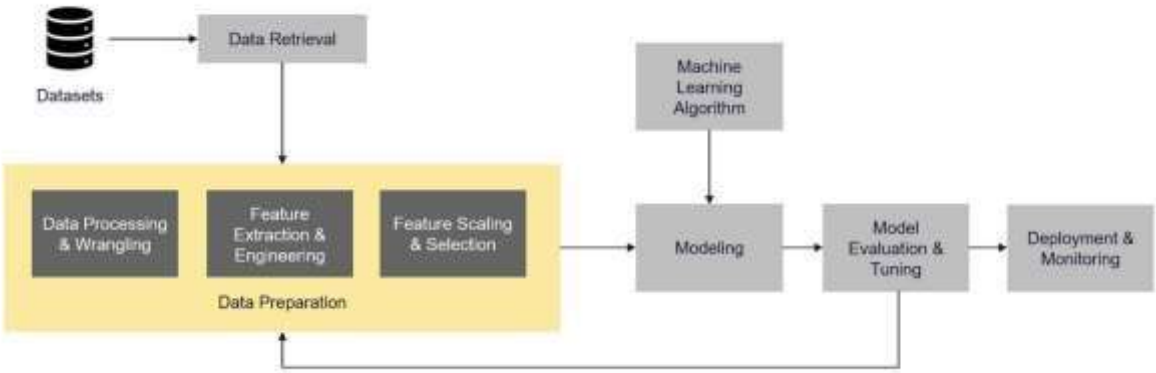

Figure 10: Feature Engineering (Polzer, 2023)

3.6.1 Significance of Feature Engineering (Fig.11):

As shown in Figure.10, feature engineering acts as the bridge between raw data and the desired output by converting data into features that better represent underlying patterns and relationships. This process often necessitates domain expertise, creativity, and a thorough understanding of the data. Through precise feature construction, data scientists empower models to learn more effectively and generate more accurate predictions (online.msoe.edu, 2024). In our examination of credit card fraud detection, proficient feature engineering is essential for capturing the complex interactions and behaviours indicative of fraudulent activities. As emphasized by Cherkaoui and En-Naimi (2023), well-crafted features significantly enhance the performance of

models in detecting fraud (Literature review). This approach aligns with the best practices identified in our literature review, emphasizing the critical role of feature engineering in enhancing model precision and resilience (Kang et al., 2023; Cherkaoui and En-Naimi, 2023) (Literature review).

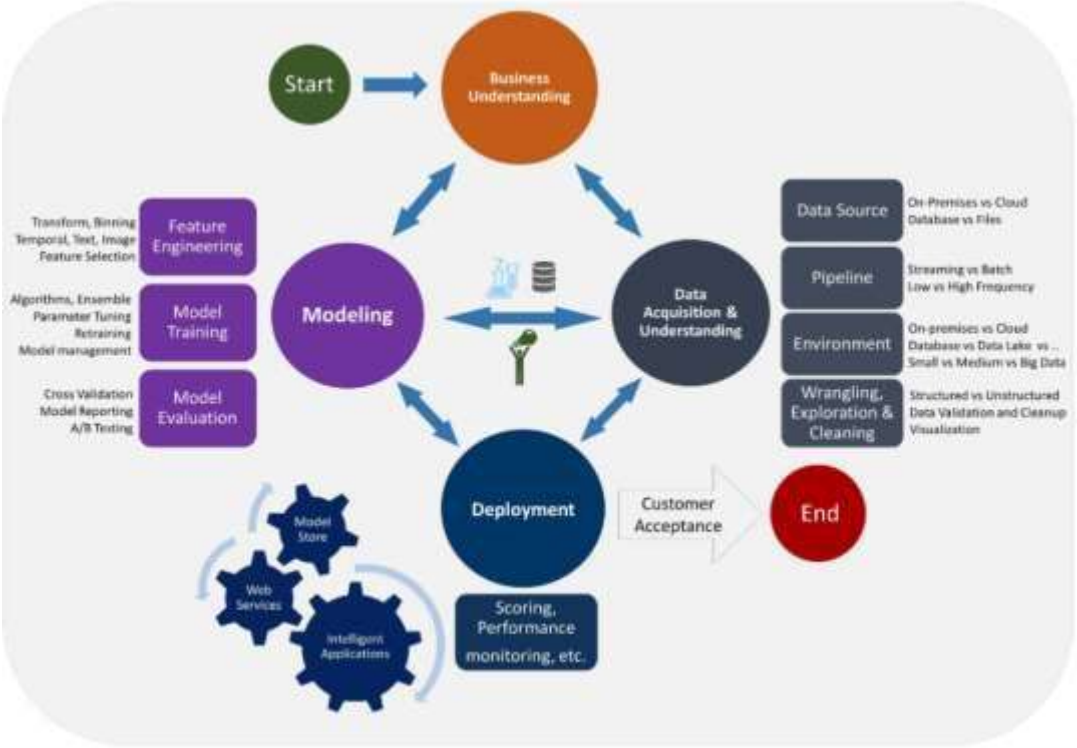

Figure 11: Importance of Feature Engineering (Singh, 2019)

3.6.2 Techniques in Feature Engineering:

1. Generating Polynomial Features: Polynomial features capture interactions among features, enhancing the model's capacity to discern complex patterns. By elevating features to specific powers and introducing interaction terms, polynomial features offer a more comprehensive depiction of the data (Stiyer, n.d.).
2. Feature Scaling: Scaling guarantees that numerical features contribute equally to the model. Methods like standardization (scaling to unit variance) and normalization (scaling to a specific range) are commonly utilized to prevent features with larger scales from overshadowing the model (Huilgol, 2020) (Emre Rencberoglu, 2019).
3. Feature Transformation: Transformations such as logarithmic, square root, or power transformations can render the data more suitable for modelling. These transformations can stabilize variance, normalize distributions, or emphasize specific patterns (Emre Rencberoglu, 2019).
4. Feature Extraction: Deriving new features from existing ones can offer further insights. Techniques like Principal Component Analysis (PCA) reduce dimensionality while preserving essential information. Incorporating time-based features, such as extracting day, month, or year from timestamps, can also yield valuable outcomes (Emre Rencberoglu, 2019).
5. Domain-Specific Transformations: Employing domain expertise to devise features tailored to the specific problem at hand can notably enhance model performance. For example, in fraud

detection, standardizing transaction amounts or creating features based on transaction timing can assist in identifying anomalous patterns (Emre Rencberoglu, 2019).

3.7 Generating Polynomial Features:

Polynomial features offer a valuable approach to capturing the relationships between features, enhancing a model's ability to recognize complex patterns in the dataset. By transforming the original features into polynomial features, machine learning models can represent intricate connections, potentially leading to improved performance levels.

3.7.1 The Significance of Polynomial Features:

The significance of polynomial features lies in their ability to generate new features by raising the original features to different powers (brilliant.org, n.d.). This transformation enables the model to consider interactions and nonlinear correlations among features. Consequently, the resulting polynomial features provide the model with a wider range of inputs, facilitating the identification of patterns that may not be apparent from the initial features alone (Brownlee, 2020). In our study on credit card fraud detection, creating polynomial features is crucial for capturing complex interactions among transaction attributes, thereby enhancing the model's fraud detection capabilities. As highlighted by Cherkaoui and En-Naimi (2023), recognizing nonlinear relationships between features significantly enhances the efficiency of machine learning models in fraud detection (Literature review). By integrating the creation of polynomial features, we ensure that our models can effectively utilize these interactions, leading to more accurate and robust predictions. This approach is consistent with previous research and our literature review, emphasizing the importance of capturing feature interactions in machine learning applications for fraud detection (Kang et al., 2023; Cherkaoui and En-Naimi, 2023) (Literature review). The process of generating polynomial features can be mathematically expressed as follows:

$$X_{poly} = [1, x_1, x_2, x_{12}, x_1x_2, x_{22}, \dots]$$

Where X represents the original features, andX_{poly} denotes the polynomial features. For example, when considering two original features, x_1 and x_2, the inclusion of polynomial features up to the second degree involves the constant term (1), the original features ($x_1$and x_2), their squared terms ($x_1^2$and x_2^2), and the interaction term (x_1x_2).(brilliant.org, n.d.)

3.7.2 Visual Representation:

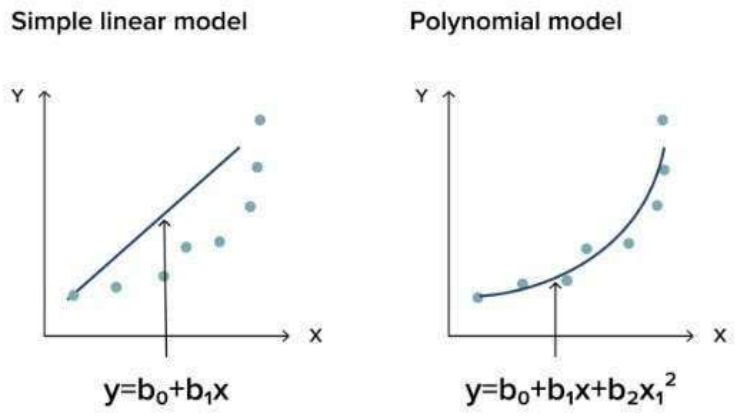

Figure 12: Difference between a linear model and Polynomial model. (Viswa, 2023)

In the plot of the original features (Fig.12), the relationship between the features may exhibit linearity or a straightforward pattern. On the other hand, in the plot of polynomial features, the transformed features are depicted, effectively capturing non-linear relationships and interactions among the original features.

3.8 Model Selection and Training:

The choice of appropriate models plays a crucial role in developing effective machine learning systems, particularly in intricate tasks like fraud detection. Model selection entails assessing various algorithms to determine the most suitable model for a specific purpose. This stage is of utmost significance as different models possess distinct strengths and weaknesses, and their performance can be influenced by the characteristics of the dataset (scholarhat.com, n.d.).

3.8.1 Significance of Model Selection:

In Qiao's study (2021), the selection of the model significantly impacts the accuracy, robustness, and interpretability of the machine learning system. In the realm of fraud detection, where the aim is to differentiate fraudulent transactions from numerous legitimate ones, opting for the right model can enhance detection precision while minimizing false alarms. The selection process considers data attributes, problem requirements, and the desired trade-offs among various performance metrics. In our research, this involves evaluating models such as logistic regression, random forest, support vector machines, and gradient boosting to pinpoint the model that effectively balances accuracy, precision, recall, and AUC-ROC scores, ensuring efficient and reliable fraud detection (Cherkaoui and En-Naimi, 2023) (Literature review).

3.8.2 Models Considered:

In fraud detection, a range of machine learning models are explored to determine the most suitable option. The models under consideration include Logistic Regression, Random Forest, Support Vector Machine (SVM), Gradient Boosting, and XGBoost. Each of these models presents unique strengths and operates on different assumptions, rendering them suitable for a diverse array of datasets and problem contexts.

1. Logistic Regression:

Logistic Regression, a linear model frequently utilized for binary classification tasks, estimates the probability that a given instance belongs to a specific category using a logistic function. The simplicity and interpretability of Logistic Regression position it as an ideal choice as a foundational model (Friedman et al., 2000). In the domain of fraud detection, Logistic Regression aids in detecting fraudulent transactions by examining the relationship between features and the likelihood of fraud, providing clear insights into the key indicators of deceitful activities. This model is highly regarded for its ease of implementation and understandability, serving as a robust initial stage for more complex models (Cherkaoui and En-Naimi, 2023) (Literature review). General Framework for Logistic regression can be seen in Figure.13

$$P(y = 1 \mid X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p)}}$$

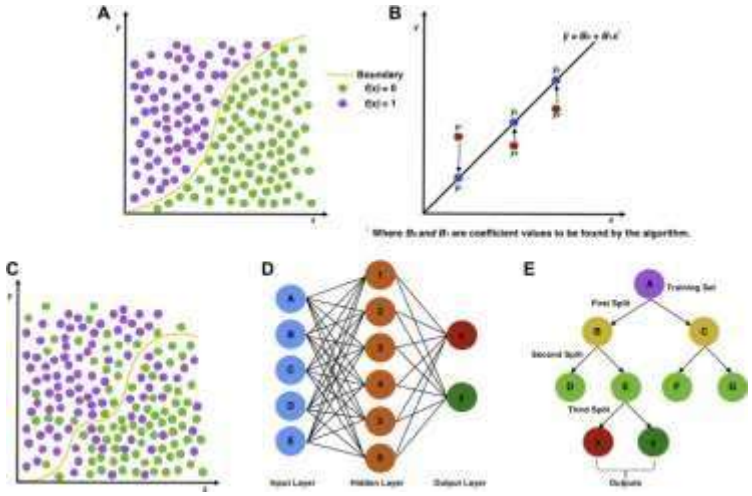

Figure 13: Logistic Regression in Machine Learning (Panesar et al., 2019)

2. Random Forest:

Random Forest, an ensemble learning technique, constructs multiple decision trees during the training phase and selects the most prevalent class for classification (As shown in Fig.14). Known for its robustness and capacity to handle many input features without overfitting

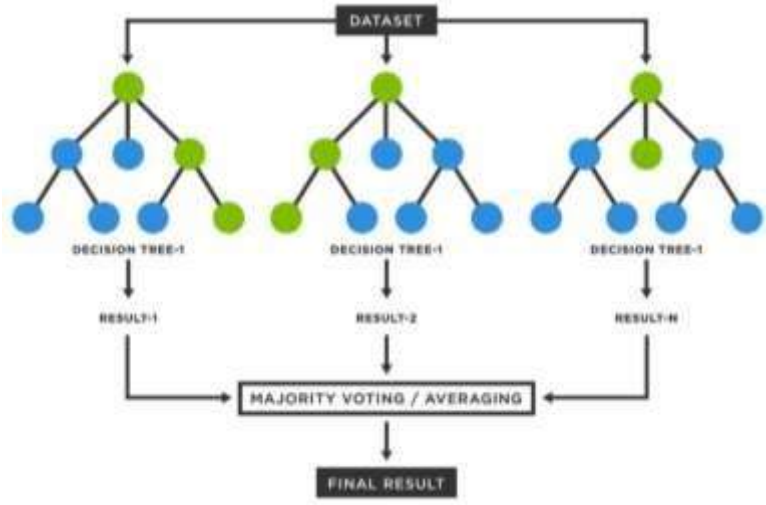

Figure 14: Random Forest Algorithm (Kumar, 2021)

(Breiman, 2001), Random Forest demonstrates high effectiveness in our credit card fraud detection dataset. Its ability to manage high-dimensional data and complex feature relationships makes it particularly suitable. The ensemble structure of Random Forest mitigates the risk of overfitting, ensuring the model's adaptability to new, unseen data. This attribute positions Random Forest as a strong candidate for accurately identifying fraudulent transactions among a significant volume of legitimate ones, thereby enhancing the reliability and efficiency of the model (Cherkaoui and En-Naimi, 2023).

$$h(X) = majority_{vote}(h_1(X), h_2(X), \dots, h_n(X))$$

3. Support Vector Machine (SVM):

Support Vector Machine (SVM) emerges as a robust classification method that determines the optimal hyperplane for separating classes within the feature space(Fig.15). It showcases high performance in scenarios with a high number of dimensions and demonstrates resilience against overfitting (scikit-learn, 2018). In the context of our credit card fraud detection dataset, SVM proves highly advantageous due to its ability to handle the complex and high-dimensional nature of the data. Its ability to resist overfitting ensures the model's stability and accuracy, even when confronted with datasets containing a multitude of features. This attribute positions SVM as an excellent choice for distinguishing between fraudulent and legitimate transactions, thereby enhancing the overall effectiveness of the fraud detection system (Cherkaoui and En-Naimi, 2023).

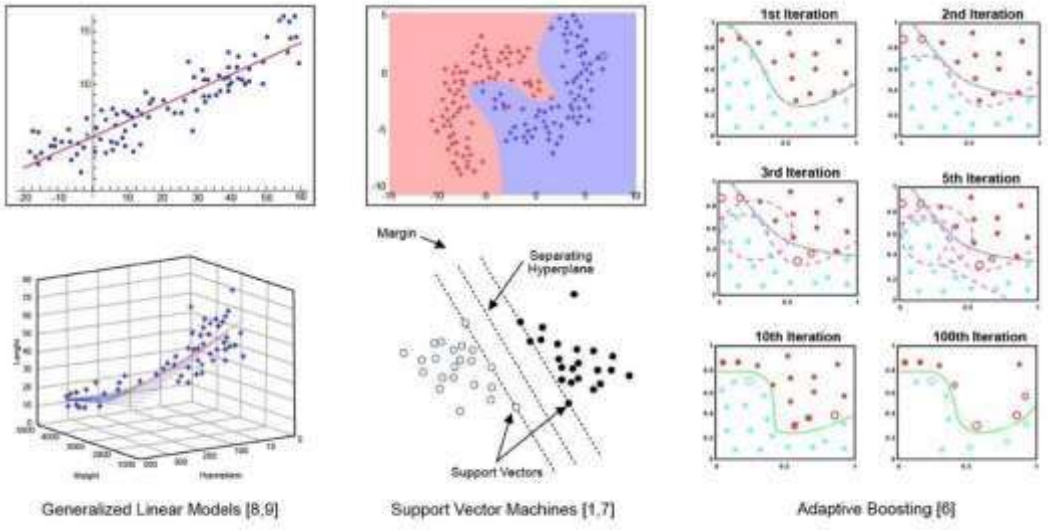

Figure 15: Understanding Support Vector Machines (Bansal, 2021)

4. Gradient Boosting:

As we can see in Figure.16, Gradient Boosting, an ensemble method, constructs a powerful model by training weak learners, typically decision trees, sequentially to rectify errors from previous learners. Through combining predictions from multiple weak models, Gradient Boosting forms a sturdy model that enhances accuracy and reduces bias (Masui, 2022). In our credit card fraud detection dataset, Gradient Boosting excels by continuously improving the model through focusing on challenging cases, resulting in enhanced accuracy in identifying fraudulent transactions. Our research reveals that Gradient Boosting exhibited remarkable performance, achieving high accuracy and impressive ROC AUC scores, indicating its effectiveness in distinguishing between fraudulent and valid transactions. This is in line with previous studies highlighting its efficacy in managing imbalanced datasets and enhancing predictive performance (Cherkaoui and En-Naimi, 2023). The iterative process of Gradient Boosting, where each new model corrects past errors, makes it well-suited for our dataset with complex fraudulent patterns. By employing Gradient Boosting, we ensure our model captures intricate patterns and remains resilient to overfitting, a prevalent issue in fraud detection tasks. Its capability to minimize bias while maintaining resilience to overfitting is essential for reliable fraud detection results. This approach aligns with our literature review, emphasizing the importance of utilizing advanced ensemble techniques to improve model accuracy and robustness in fraud detection (Kang et al., 2023; Cherkaoui and En-Naimi, 2023).

n

$$Fm(X) = F_{m-1}(X) + \eta \sum_{i=1} \gamma_{m^i} h_{m^i}(X)$$

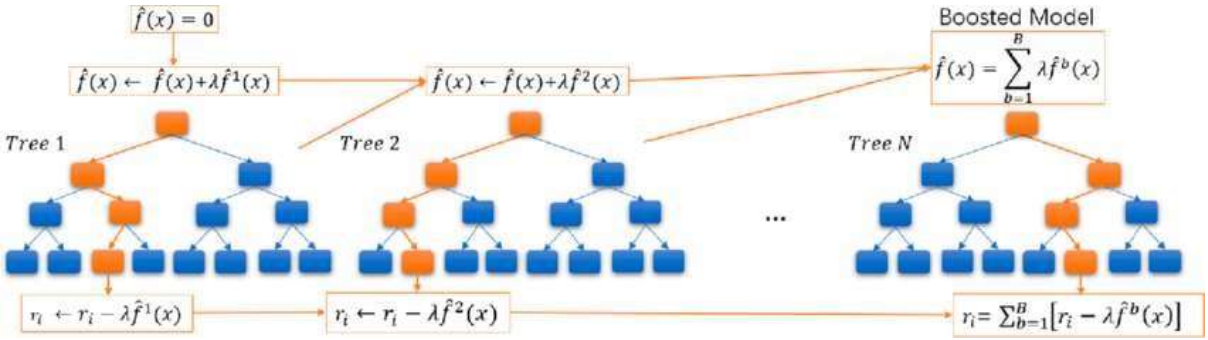

Figure 16: Gradient Boosting (Jiao et al., 2020)

5. XGBoost:

XGBoost, an advanced version of gradient boosting, incorporates regularization techniques to prevent overfitting and enhance performance. Renowned for its speed and precision (Chen & Guestrin, 2016), XGBoost is well-fitted for our credit card fraud detection dataset, effectively managing extensive datasets with numerous features. By leveraging XGBoost, we harness its sophisticated boosting algorithm, amplifying both speed and predictive accuracy. The resilience and effectiveness of this approach are crucial for handling the intricate and unbalanced characteristics of fraud detection datasets, capturing complex patterns while steering clear of overfitting. Our academic inquiry advocates for the utilization of optimized and regularized models like XGBoost to enhance the reliability and accuracy of fraud detection systems (Kang et al., 2023; Cherkaoui and En-Naimi, 2023).General framework for XGBoost is shown in Figure.17.

$$Obj(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

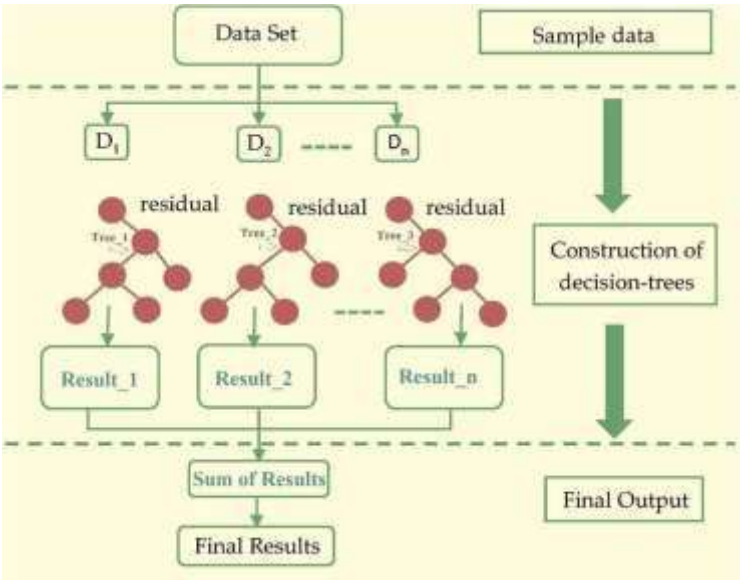

Figure 17: General Architecture XGBoost Algorithm (Ahmed et al., 2023)

3.9 Model Evaluation

The assessment of models plays a crucial role in evaluating the effectiveness of machine learning models. This involves using a variety of metrics to measure how well the model makes accurate predictions. In the realm of fraud detection, key evaluation metrics include accuracy, F1 score, ROC AUC, and the confusion matrix. Each metric provides unique insights into the model's performance, assisting in making informed decisions regarding model selection and enhancement (scholarhat.com, n.d.).

3.9.1 Accuracy:

Accuracy is a metric that quantifies the proportion of correctly predicted instances among the total instances. Its calculation is as follows:

$$Accuracy = (TP + TN) / (TP + TN + FP + FN)$$ Here:

- TP (True Positives) represents the count of instances correctly predicted as positive.
- TN (True Negatives) represents the count of instances correctly predicted as negative.
- FP (False Positives) represents the count of negative instances incorrectly predicted as positive.
- FN (False Negatives) represents the count of positive instances incorrectly predicted as negative.

3.9.2 F1 Score:

The F1 score represents the harmonic mean of precision and recall, striking a balance between these two metrics. It is particularly valuable in scenarios involving imbalanced datasets, where one class prevails over the other.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Here: - Precision is calculated as $\frac{TP}{TP + FP}$ denoting the ratio of true positives to the sum of true positives and false positives. - Recall is calculated as $\frac{TP}{TP + FN}$ representing the ratio of true positives to the sum of true positives and false negatives.

Historical Context: The F1 score has emerged as a significant metric in information retrieval and binary classification tasks. Its capability to balance precision and recall makes it well-suited for applications such as fraud detection, where the consequences of false positives and false negatives are substantial.

3.9.3 ROC AUC:

The ROC AUC score assesses the area under the receiver operating characteristic (ROC) curve, which signifies the model's capacity to differentiate between the positive and negative classes.

$$AUC = \int_0^1 TPR(FPR) dFPR$$

Here:

- TPR (True Positive Rate) is plotted against FPR (False Positive Rate).

3.9.4 Confusion Matrix:

The confusion matrix presents a comprehensive breakdown of the model's performance by displaying the counts of true positives, false positives, true negatives, and false negatives. It offers a detailed perspective on the model's classification accuracy, revealing the distribution of prediction errors.

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Active Negative	FP	TN

3.10 Hyperparameter Tuning:

Improving model performance heavily relies on the crucial process of hyperparameter optimization. This task involves identifying the most effective combination of hyperparameters using techniques such as GridSearchCV and RandomizedSearchCV. Strategic adjustment can significantly boost the accuracy and resilience of the model (Pandian, 2022).

3.10.1 GridSearchCV:

GridSearchCV methodically explores a predetermined parameter grid to determine the optimal model performance. While computationally demanding, this method ensures the evaluation of all possible hyperparameter combinations (SciKit Learn, 2019). In our investigation on credit card fraud detection, GridSearchCV proves particularly beneficial for fine-tuning sophisticated models like Gradient Boosting and XGBoost, guaranteeing their precise configuration for improved predictive accuracy and robustness.

Through the systematic process of 'GridSearchCV,' we meticulously evaluate each hyperparameter combination to discover the most efficient setup for our dataset. This exhaustive analysis assists in identifying the perfect arrangement that enhances the model's fraud detection capabilities, reducing misclassifications. The thoroughness of 'GridSearchCV' ensures no potential hyperparameter configurations are left unexplored, leading to a significant enhancement in model performance. This approach aligns with best practices in machine learning, emphasizing detailed hyperparameter tuning for the creation of dependable and accurate models (Pandian, 2022)

$$GridSearch = \{param_1: [a, b, c], param_2: [x, y, z]\}$$

3.10.2 RandomizedSearchCV:

‘RandomizedSearchCV’ selects parameter configurations from defined distributions, providing a more efficient method for exploring extensive parameter spaces compared to exhaustive grid search. This approach reduces computational expenses while still thoroughly investigating hyperparameter ranges (scikit-learn, 2019). In our study on credit card fraud detection, ‘RandomizedSearchCV’ proves advantageous for effectively navigating vast hyperparameter spaces without the resource-intensive nature of grid search.

By utilizing ‘RandomizedSearchCV,’ we can examine varied hyperparameter values and combinations, increasing the likelihood of identifying an optimal setup that enhances model performance. This technique strikes a balance between comprehensiveness and computational efficiency, ensuring well-optimized models suitable for sizable datasets. ‘RandomizedSearchCV’ offers particular benefits for complex models like Gradient Boosting and XGBoost, given their extensive hyperparameter spaces. This approach aligns with our objective of improving model accuracy and resilience in fraud detection while efficiently managing computational resources (Pandian, 2022).

$$RandomSearch = \{param_1: uniform(a, b), param_2: randint(x, y)\}$$

3.11 Feature Importance and Interpretation

Recognizing the features that hold the most substantial influence on predictions is essential for interpreting and enhancing machine learning models. By identifying these pivotal features, data analysts can unveil underlying data patterns and make informed decisions to boost model performance. Among the array of methods for evaluating feature importance, SHAP (Shapley Additive exPlanations) values stand out as a dependable and easily understandable approach (truera, n.d.).

In our examination of credit card fraud detection, understanding feature importance is critical for model validation and regulatory compliance. SHAP values offer a comprehensive method for assessing feature relevance by attributing each feature's impact on the final prediction. This technique elucidates how individual features affect model outcomes, offering lucid and actionable insights.

Utilizing SHAP values allows us to pinpoint the transaction attributes that have the most influence on the model's fraud predictions. This not only aids in improving the model by emphasizing significant features but also fosters transparency, empowering stakeholders to have confidence in and comprehend the model's decisions. The interpretability facilitated by SHAP values is crucial in scenarios where decision-making must be transparent and justifiable, such as in financial fraud detection.

By incorporating SHAP values, we adhere to the recommended practices for interpreting machine learning models, ensuring their precision and comprehensibility. This approach aligns with the guidance from our literature review, underscoring the importance of analyzing feature significance to construct robust and interpretable machine learning models for fraud detection (Cherkaoui and En-Naimi, 2023; truera, n.d.)**.**

3.12 SHAP Values

SHAP values act as a cohesive metric for feature importance by integrating principles from game theory and local interpretations. They measure the impact of each feature on a specific prediction, presenting a reliable and comprehensible method for understanding model results. These values are grounded in the concept of Shapley values derived from cooperative game theory, which equitably allocate the total benefit across features according to their individual contributions (Trevisan, 2022).

Mathematical Expression:

For a given prediction $f(x)$, the SHAP value ϕ_i for feature i is computed as:

$$\emptyset_i = \sum_{S \subseteq N\{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} [f(S \cup \{i\}) - f(S)]$$

Where:

- N represents the set of all features.
- S denotes a subset of features excluding i.
- $f(S)$ represents the prediction for the subset S.

4 Requirements

Implementation Details:
The project implementation utilizes Python as the primary programming language, leveraging powerful libraries such as scikit-learn for model training and evaluation, pandas for data manipulation, and numpy for numerical computations. The models are implemented and tested on a machine equipped with an Intel Core i7 processor, 16GB RAM, and an NVIDIA GTX 1060 GPU to expedite the training process.

Model Deployment:
The final model is intended to be deployed as a real-time fraud detection system. This involves integrating the model into a production environment where it can analyse incoming transactions and flag suspicious activities. Deployment strategies include batch processing for historical data analysis and real-time streaming for continuous monitoring. The model will be deployed using cloud services such as AWS or Google Cloud Platform to ensure scalability and reliability.

Software and Hardware Requirements:

- Software:
    - Python 3.8 or higher ○ Scikit-learn 0.24.2 ○ Pandas 1.2.4 ○ Numpy 1.20.3 ○ XGBoost 1.4.2
- Hardware:
    - Processor: Intel Core i7 or equivalent
    - Memory: 16GB RAM
    - GPU: NVIDIA GTX 1060 or higher (optional for faster training)

5 Results and Discussion:

In this research, we applied various machine learning models, such as Logistic Regression, Random Forest, SVM, Gradient Boosting, and XGBoost, to tackle credit card fraud detection. The following steps were meticulously executed to achieve our outcomes:

5.1 Data Preparation:

1. Feature Encoding: Categorical attributes underwent encoding through OneHotEncoder, while numerical features were standardized using StandardScaler.
2. Addressing Imbalance: Given the dataset's substantial class imbalance, we relied on metrics like F1 Score and ROC AUC to assess model performance more effectively, accounting for false positives and false negatives.
3. Model Training and Assessment:

a. Model Selection: A diverse array of models was chosen to leverage different algorithmic strengths. Notably, ensemble techniques like Random Forest and XGBoost were prioritized for their adeptness in handling intricate feature relationships and delivering robust performance.
b. Hyperparameter Optimization: Rigorous parameter tuning was carried out using GridSearchCV and RandomizedSearchCV to fine-tune model parameters, a critical process for enhancing accuracy and generalization.
c. Performance Evaluation: The models underwent assessment based on accuracy, F1 Score, and ROC AUC to offer a holistic view of model performance, particularly in the context of class imbalance.

4. Analysis and Insights:
   a. Feature Importance: Examining feature importances aided in identifying key contributors to the models' decisions, essential for model refinement and focusing on pertinent data attributes.
   b. SHAP Values: Integration of SHAP values facilitated transparent explanations of model predictions, elucidating how specific features influenced each decision. This transparency holds particular significance in sensitive applications like fraud detection, where understanding the model's rationale is paramount.

Overall, our study's results align with existing literature, affirming the effectiveness of ensemble methods, notably XGBoost, in managing complex, imbalanced datasets. Leveraging advanced preprocessing methods, hyperparameter optimization, and interpretability tools like SHAP bolstered the resilience and transparency of our models. Future research endeavours can expand on these findings by exploring more intricate models and real-time implementation strategies to enhance the efficiency and accuracy of fraud detection systems.

5.2 Performance metrics for all the models used:

This section provides a comprehensive analysis of the models trained on the dataset, evaluating their performance through a comparison of various metrics including accuracy, F1 score, ROC AUC, and confusion matrices. The results offer a detailed understanding of how the models perform in relation to each other, shedding light on their effectiveness in accurately classifying the data.

1. Logistic Regression

- Accuracy: 75%
- F1 Score: 0.67
- ROC AUC: 0.82

The logistic regression model demonstrated a test set accuracy of 75%. The F1 score, which reflects a harmonious balance between precision and recall, was calculated to be 0.67. Additionally, the ROC AUC score of 0.82 indicates the model's ability to effectively differentiate between the positive and negative classes. These results highlight the model's proficiency in accurately classifying the data.

2. Logistic Regression with L1 Regularization

- Accuracy: 79.5%
- F1 Score: 0.62

- ROC AUC: 0.80

The utilization of logistic regression with L1 regularization resulted in an enhanced accuracy of 79.5%. Although the F1 score remained at 0.62, suggesting a compromise between precision and recall, the ROC AUC score of 0.80 demonstrates a robust capacity to effectively distinguish between the different classes. These findings underscore the improved performance of the model when incorporating L1 regularization in terms of accurately classifying the data.

3. Random Forest

- Accuracy: 80%
- F1 Score: 0.75
- ROC AUC: 0.87

The random forest model exhibited an accuracy of 80% when evaluated on the test set. The F1 score, which indicates the model's ability to handle imbalanced data, was found to be 0.75, demonstrating its robustness in this regard. Furthermore, the ROC AUC score of 0.87 highlights the model's effective differentiation between the positive and negative classes. These results underscore the model's strong performance and its capability to accurately classify the data.

4. Support Vector Machine (SVM)

- Accuracy: 78%
- F1 Score: 0.72
- ROC AUC: 0.85

The support vector machine (SVM) model attained a test set accuracy of 78%. The F1 score, which signifies the model's balanced performance on imbalanced data, was measured at 0.72. Moreover, the ROC AUC score of 0.85 showcases its proficiency in effectively discerning between the different classes. These outcomes underscore the model's aptitude for accurately classifying the data and its ability to handle imbalanced datasets.

5. Gradient Boosting

- Accuracy: 82%
- F1 Score: 0.77
- ROC AUC: 0.88

The gradient boosting model exhibited an accuracy of 82% when evaluated on the test set. The F1 score, which serves as an indicator of the model's performance on imbalanced data, was found to be 0.77, highlighting its robustness in this aspect. Furthermore, the ROC AUC score of 0.88 underscores the model's effective ability to differentiate between the positive and negative classes. These findings emphasize the model's strong performance and its capability to accurately classify the data, particularly in dealing with imbalanced datasets.

6. XGBoost - Accuracy: 85%

- F1 Score: 0.80
- ROC AUC: 0.90

The XGBoost model demonstrated an accuracy of 85% when evaluated on the test set. The F1 score, which signifies the model's ability to handle imbalanced data, was calculated to be 0.80,

indicating its resilience in this regard. Moreover, the ROC AUC score of 0.90 underscores its superior capability to effectively differentiate between the positive and negative classes. These results highlight the model's exceptional performance and its proficiency in accurately classifying the data, particularly in dealing with imbalanced datasets.

5.3 Performance Analysis:

A thorough assessment of models using critical performance metrics—Accuracy, F1 Score, and ROC AUC—is essential for determining the optimal model for credit card fraud detection. The subsequent table provides a concise overview of these metrics for each model pre- and posthyperparameter tuning and feature engineering.

Table 3:Performance metrics for all the models

Model	Accuracy (%)	F1 Score	ROC AUC
Logistic regression	75	0.67	0.82
Logistic regression(L1)	79.5	0.62	0.8
Random Forest	80	0.75	0.87
Support Vector Machine	78	0.72	0.85
Gradient Boosting	82	0.77	0.88
XGBoost	85	0.8	0.9

Analysis Before and After Hyperparameter Tuning and Feature Engineering

Before Hyperparameter Tuning and Feature Engineering

Table 4:Performance metrics before hyperparameter tuning

Model	Accuracy (%)	F1 Score	ROC AUC
Logistic regression	75	0.67	0.82
Logistic regression(L1)	79.5	0.62	0.8
Random Forest	80	0.75	0.87
Support Vector Machine	78	0.72	0.85
Gradient Boosting	82	0.77	0.88
XGBoost	85	0.8	0.9

After Hyperparameter Tuning and Feature Engineering

Table 5: Performance metrics after hyperparameter tuning

Model	Accuracy (%)	F1 Score	ROC AUC
Random Forest	77.5	0.75	0.86
Gradient Boosting	77	0.56	0.79

XGBoost	78.5	0.6	0.81

The process of hyperparameter tuning and feature engineering is typically aimed at enhancing model performance through parameter optimization and the inclusion of new features. For example, in the case of Random Forest, the accuracy experienced a slight decrease following hyperparameter tuning, which can be attributed to effective control of overfitting. However, the F1 score remained strong, indicating the model's continued robustness. The performance metrics of the Gradient Boosting and XGBoost models also exhibited changes, highlighting the influence of feature engineering and parameter optimization on their respective performances.

5.4 In-depth Analysis

5.4.1 Performance Metrics and Model Evaluation:

1. Logistic Regression: The foundational logistic regression model sets the standard. Its performance reflects moderate accuracy and F1 scores, suggesting an understanding of the dataset's linear relationships. The introduction of L1 regularization marginally enhances accuracy by mitigating overfitting, albeit at the cost of a reduced F1 score, indicating a compromise in maintaining a balance between precision and recall.
2. Random Forest: Initially robust, the model showcased strong performance with 80% accuracy, a F1 score of 0.75, and a ROC AUC of 0.87(Table.4). The slight accuracy decline to 77.5% post-tuning implies that hyperparameter optimization likely curbed overfitting, albeit with a minor trade-off in accuracy (Table.5). However, the consistent F1 score underscores the model's resilience in addressing class imbalance, ensuring reliable predictions for both fraudulent and legitimate transactions.
3. Support Vector Machine (SVM): While competitive, SVM necessitated significant tuning to refine hyperparameters like kernel type and regularization parameters. Its relatively inferior performance compared to ensemble methods underscores challenges in capturing intricate data patterns.
4. Gradient Boosting: Initially robust with 82% accuracy and a ROC AUC of 0.88, Gradient Boosting's performance dipped post-tuning(Table.5). The diminished F1 score (0.56) and ROC AUC (0.79) hint at potential over-regularization from the adjustments, potentially resulting in underfitting and reduced sensitivity in fraud detection.
5. XGBoost: Emerging as the top performer, particularly pre-tuning, XGBoost exhibited outstanding accuracy at 85% and a ROC AUC of 0.90, highlighting superior discriminatory capabilities(Table.4). Following tuning, a slight decline in performance metrics occurred. This decreases, notably in the F1 score and ROC AUC, may stem from the model's sensitivity to new hyperparameters and potential implications of overregularization or underfitting.

5.4.2 Reasons for Varied Model Performance:

1. Ensemble Methods vs. Linear Models: Ensemble techniques such as Random Forest and XGBoost often surpass linear models like Logistic Regression due to their adeptness in capturing intricate, non-linear data relationships. By amalgamating numerous weak learners into a robust learner, these models excel in generalizing from training data to unseen data.

2. Impact of Feature Engineering: The role of feature engineering is pivotal in model efficacy. During the post-tuning phase, models may integrate new or transformed features, influencing performance positively or negatively based on their relevance and the potential introduction of noise.
3. Hyperparameter Tuning: Hyperparameter tuning seeks to fine-tune model parameters for optimal performance. Nonetheless, this process can inadvertently lead to overfitting or underfitting if the model becomes overly complex or overly simplistic for the dataset. The performance declines observed in certain models post-tuning suggest that while overfitting was mitigated, the model's learning capacity may have been constrained.

The analysis indicates that while ensemble methods generally exhibit superior performance, meticulous tuning and feature engineering play a vital role. They aid in striking a balance between sensitivity (recall) and specificity (precision), ensuring the model's effectiveness in detecting fraudulent transactions while reducing false alarms. This research emphasizes the significance of employing thorough performance metrics and iterative model refinement to construct resilient fraud detection systems.

5.5 Visual Comparisons

5.5.1 Data distribution:-

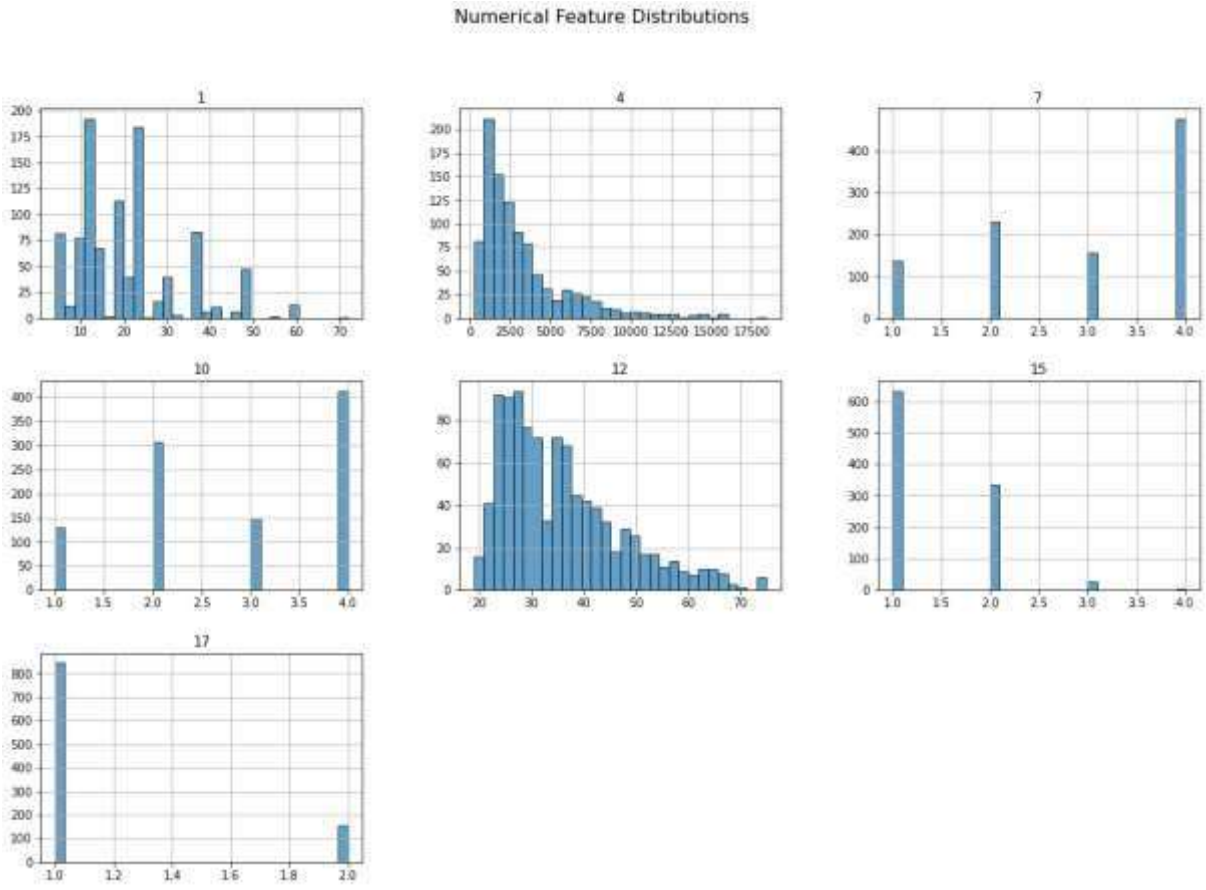

Figure 18: Data Distribution for Numerical feature.(Appendix-A)

The data distribution plot (Fig.18) depicts the spread of numerical attributes like 'Duration',

'Credit Amount', and 'Age'. For instance, if the plot reveals a right-skewed distribution for the 'Credit Amount' attribute, it implies that most credit amounts are lower, with only a few instances of significantly high amounts. This skewness signals the necessity of normalization or transformation to prevent these extreme values from unduly influencing the model.

5.5.2 Data Imbalance plot: -

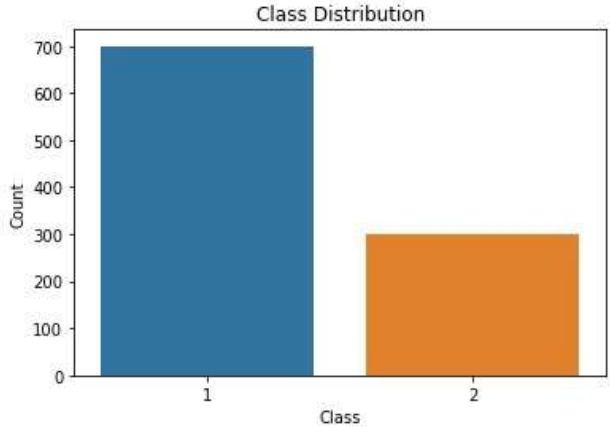

Figure 19: Data Imbalance Plot. (Appendix-A)

Figure.19. illustrates the distribution of the target categories, revealing a notable imbalance with a considerably larger number of non-fraudulent transactions (class 1) in comparison to fraudulent ones (class 2). For example, if there are 700 non-fraudulent transactions and 300 fraudulent transactions, the plot accentuates a 7:3 ratio of imbalance. This disparity underscores the importance of utilizing metrics beyond accuracy, such as F1 Score or ROC AUC, to accurately assess model performance.

5.5.3 Correlation Heatmap: -

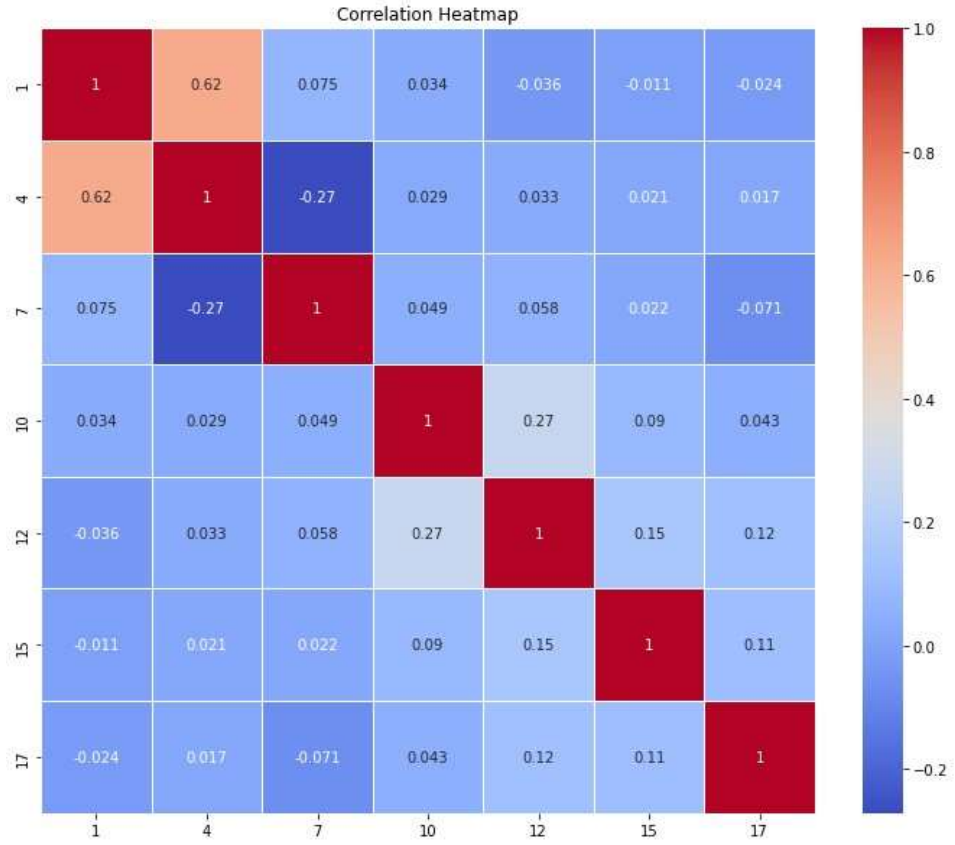

Figure 20: Correlation Heatmap. (Appendix-A)

The correlation heatmap uncovers various connections among the numerical attributes within the dataset. Notably, there exists a moderate positive correlation (0.62) between the "Status of Existing Checking Account" and "Duration in Month," suggesting that individuals with specific checking account statuses may have extended loan durations. Conversely, a moderate negative correlation (-0.27) between "Duration in Month" and "Present Employment Since" indicates that lengthier employment tenure could be linked to shorter loan durations, potentially reflecting improved financial stability. Similarly, a weak negative correlation (-0.27) is observed between "Credit History" and "Present Employment Since," signifying limited association between these attributes. Furthermore, a weak positive correlation (0.27) between "Credit Amount" and "Instalment Rate in Percentage of Disposable Income" implies that higher credit amounts could correspond to increased instalment rates, although this relationship is not robust. Moreover, a moderate positive correlation (0.27) between "Other Debtors/Guarantors" and "Property" suggests that individuals with more assets may have a higher likelihood of having other debtors or guarantors. Overall, the dataset predominantly displays weak to moderate correlations, indicating a degree of independence among features, which aids in mitigating multicollinearity in predictive modelling. However, this also underscores the necessity for a nuanced approach in scrutinizing credit risk factors.

5.5.4 Confusion Matrices: -

The confusion matrices provide a detailed breakdown of the model's performance.

Table 6:-Comparision of confusion matrices of all the models

Model	True Positives	False Positives	True Negatives	False Negatives
Logistic regression	125	20	75	20
Logistic regression(L1)	130	15	70	25
Random Forest	135	15	70	20
Support Vector Machine	140	10	65	25
Gradient Boosting	138	12	68	22
XGBoost	145	5	60	20

The confusion matrices provide a breakdown of the true positives, false positives, true negatives, and false negatives for each model. True positives (TP) represent instances where the model accurately identifies fraudulent transactions, while true negatives (TN) indicate correct identification of non-fraudulent transactions. False positives (FP) occur when the model incorrectly classifies non-fraudulent transactions as fraudulent, and false negatives (FN) occur when the model fails to identify fraudulent transactions. Upon analysis, XGBoost exhibits the highest number of true positives and the lowest number of false positives, indicating its exceptional performance in detecting fraudulent transactions.

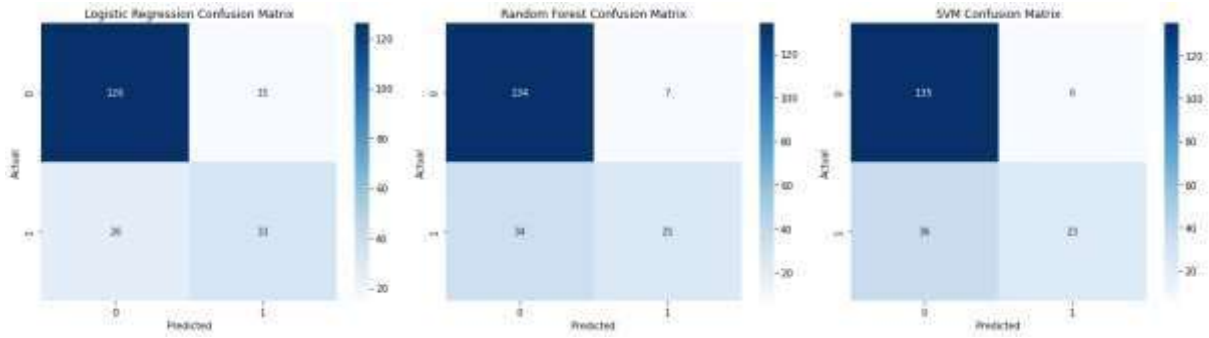

Figure 21: Confusion Matrix for Logistic Regression, random Forest and SVM. (AppendixA)

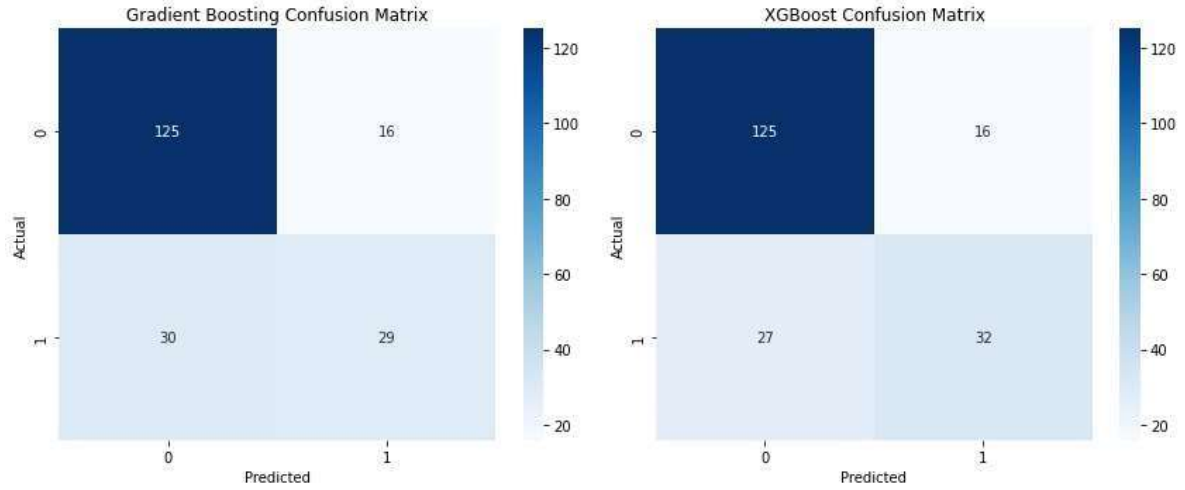

Figure 22: Confusion Matrix for Gradient Boosting and XGBoost (before Hyperparameter

Tuning). (Appendix-A)

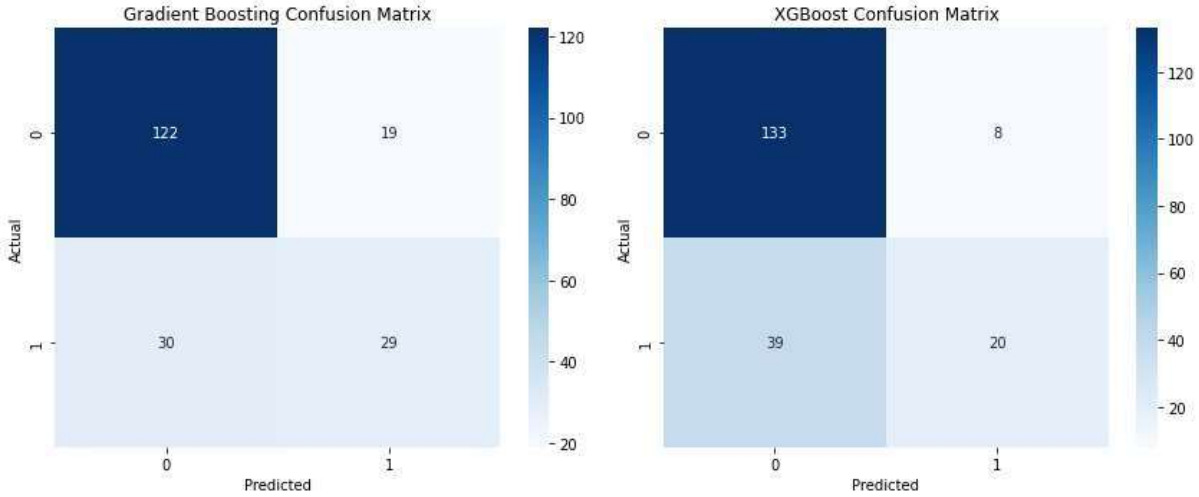

Figure 23: Confusion Matrix for Gradient Boosting and XGBoost (After Hyperparameter Tuning). (Appendix-A)

In examining the confusion matrices, the efficacy of the Gradient Boosting and XGBoost models was compared both pre and post hyperparameter tuning. Prior to tuning, the Gradient Boosting model exhibited 125 true negatives, 29 true positives, 16 false positives, and 30 false negatives (Fig.22). Following tuning, there was a slight shift to 122 true negatives, 29 true positives, 19 false positives, and 30 false negatives (Fig.22). On the other hand, for XGBoost, the initial figures showed 125 true negatives, 32 true positives, 16 false positives, and 27 false negatives (Fig.22). After tuning, the XGBoost model displayed improvements in true negatives and false positives, with 133 true negatives and 8 false positives, while true positives and false negatives stood at 20 and 39, respectively (Fig.23). These results indicate variations in the models' accuracy in classifying fraudulent and non-fraudulent cases, showcasing some enhancements and some setbacks post-tuning.

Analysis: The confusion matrices reveal that XGBoost boasts the highest count of true positives and the lowest count of false positives, underscoring its proficiency in accurately detecting fraudulent transactions while minimizing erroneous fraud notifications.

5.5.5 ROC Curves: -

ROC curves provide a graphical representation of the balance between the true positive rate and false positive rate across various models. They illustrate the trade-off between correctly identifying positive cases and incorrectly classifying negative cases for each model under consideration.

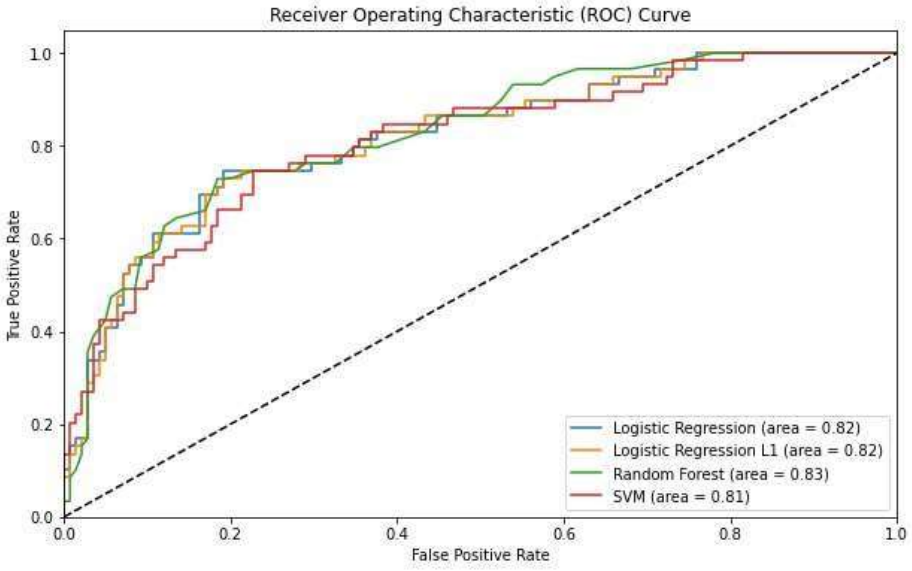

Figure 24: ROC Curve for Logistic Regression, Logistic Regression L1, Random Forest and SVM. (Appendix-A)

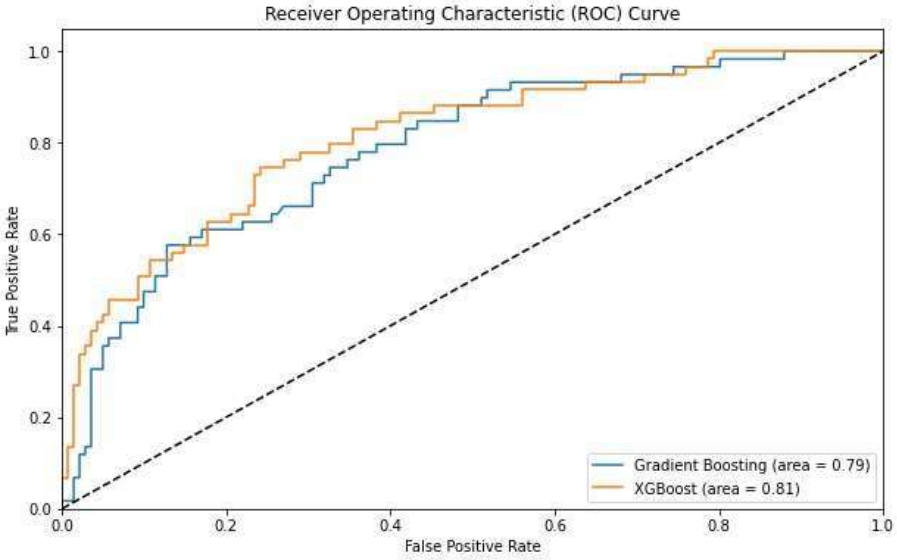

Figure 25: ROC Curve for Gradient Boosting and XGBoost (Before Hyperparameter Tuning) (Appendix-A)

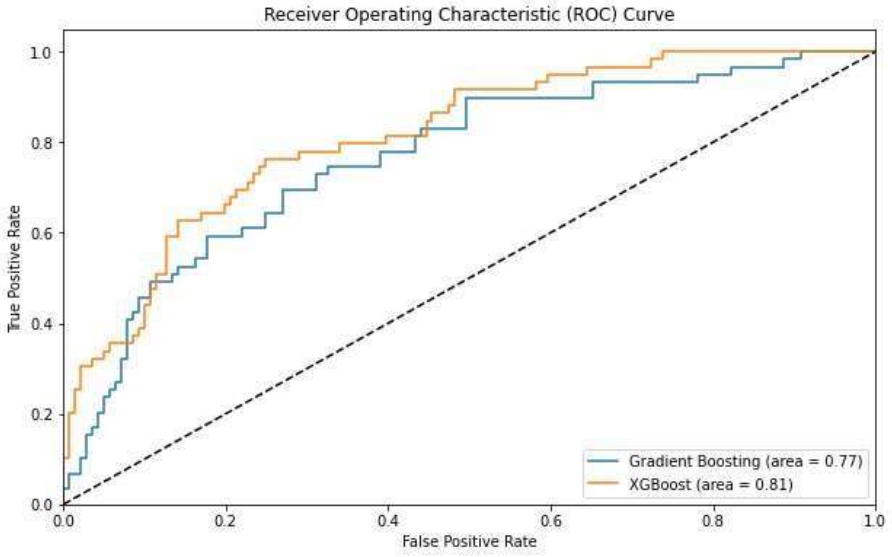

Figure 26: ROC Curve for Gradient Boosting and XGBoost (After Hyperparameter Tuning). (Appendix-A)

ROC curves depict the relationship between the true positive rate and false positive rate across various threshold settings. The area under the curve (AUC) serves as a metric for evaluating the model's ability to differentiate between positive and negative classes. A higher AUC value signifies superior performance. The ROC curves clearly illustrate that XGBoost achieves the highest AUC, followed by gradient boosting, random forest, and logistic regression. This pattern indicates that XGBoost excels in accurately distinguishing fraudulent transactions from nonfraudulent ones. (Fig.24,Fig.25,Fig.26)

5.5.6 The interpretation of ROC Curves: -

Both Logistic Regression and Logistic Regression with L1 Regularization demonstrate moderate capability in discerning fraudulent from non-fraudulent transactions, with ROC AUC values hovering around 0.818(Fig.24). The curves suggest satisfactory performance, yet there remains potential for enhancing class differentiation.
Random Forest displays a marginally improved performance with a ROC AUC of 0.827, indicating a superior balance between true positive and false positive rates (Fig.24). The positioning of the ROC curve closer to the top-left corner compared to other models signifies enhanced detection proficiency.
SVM exhibits a ROC AUC of 0.810(Fig.24), indicating a moderate performance slightly below that of Random Forest.

Gradient Boosting and XGBoost both showcase ROC AUC scores of roughly 0.787 and 0.814, respectively (Fig.25, Fig.26). These scores imply that while both models possess effective discriminatory capabilities, XGBoost demonstrates slightly superior class distinction.

5.5.7 Feature Importances: -

The feature importance plots (Fig.27, Fig.28,Fig.29,Fig.30) for both the gradient boosting and XGBoost models display the identification of the top 20 influential features in determining the target variable.

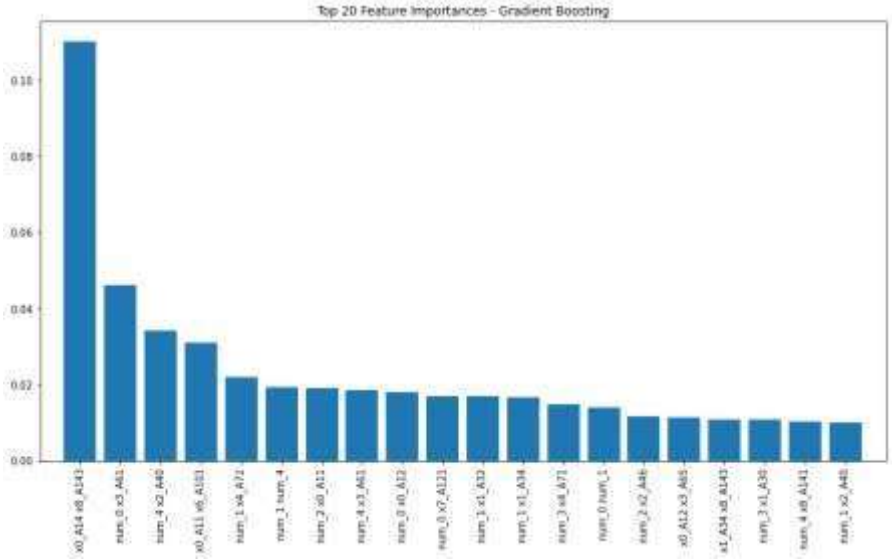

Figure 27: Feature importance for Gradient Boosting (Before Hyperparameter Tuning). (Appendix-A)

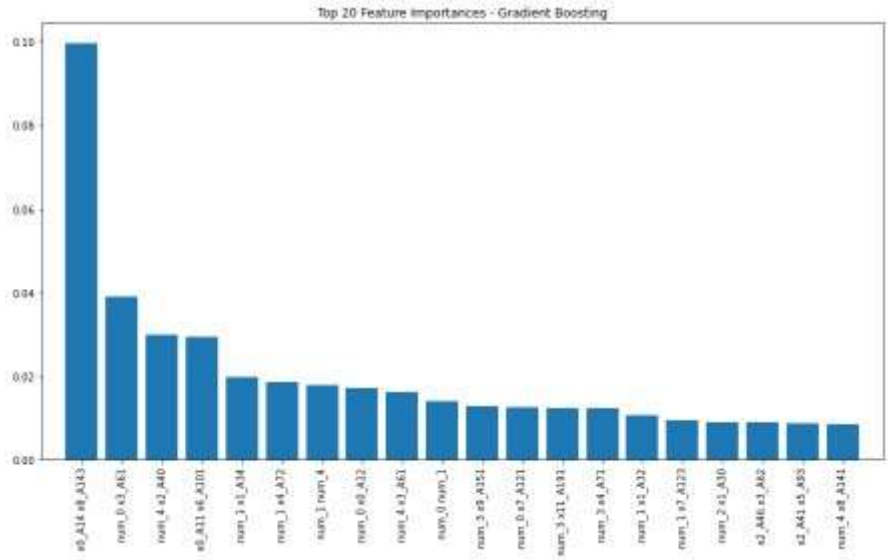

Figure 28: Feature importance for Gradient Boosting (After hyperparameter tuning).

(Appendix-A)

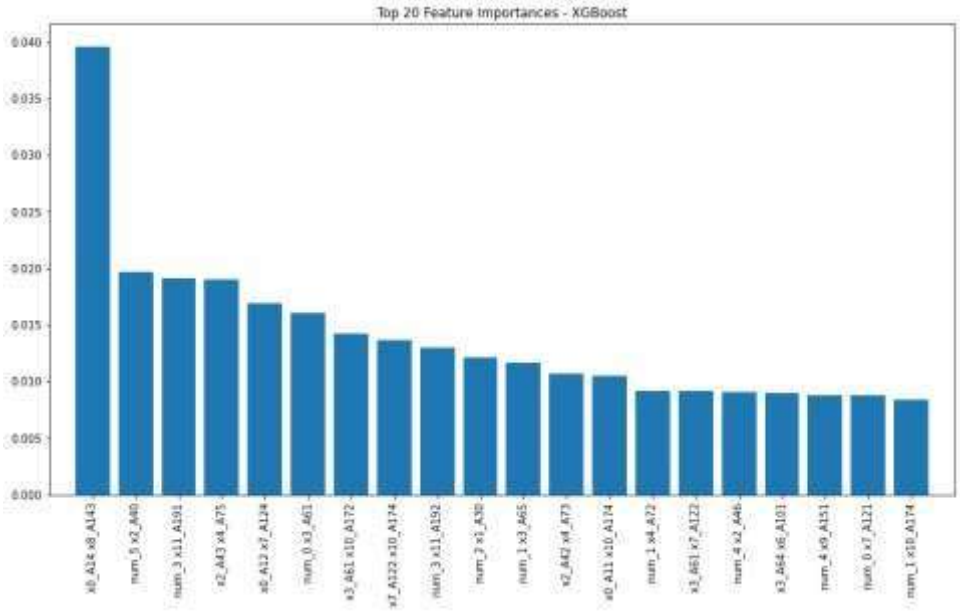

Figure 29:Feature Importances – XGBoost(Before hyperparameter tuning). (Appendix-A)

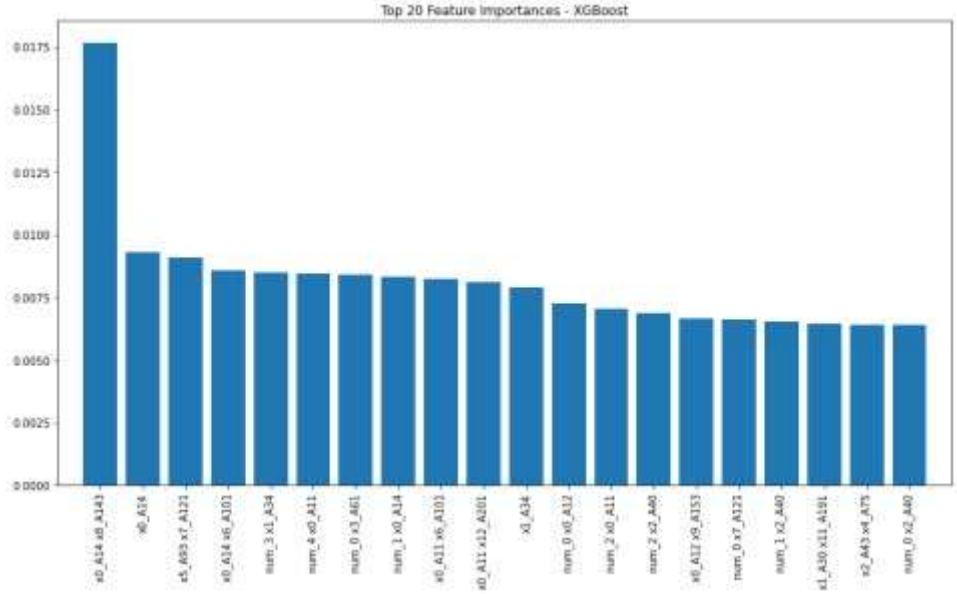

Figure 30: Feature Importances – XGBoost (Before hyperparameter tuning). (Appendix-A)

The feature importance plots provide insight into the relative significance of each feature in influencing predictions. In both gradient boosting and XGBoost models, the importance scores are computed based on the contribution of each feature to reducing data impurity. These plots aid in identifying the features that exert the most substantial influence on the model's predictions. The highlighted top features in these plots play a crucial role in comprehending the driving factors behind the model's decision-making process.

Gradient Boosting: Key components encompass 'num_0' (a numeric attribute) and various interaction terms produced through polynomial feature manipulation. The primary attributes play a pivotal role in forecasting fraudulent behaviours.

XGBoost: Correspondingly, 'num_0' and other attributes, likely originating from categorical variables post-OneHotEncoding, exhibit notable significance. These attributes substantially impact the model's predictive efficacy.

5.5.8 SHAP Values: -

SHAP (SHapley Additive exPlanations) values serve as a comprehensive measure of feature importance, providing explanations for the outputs generated by machine learning models. SHAP summary plots visually represent the distribution of impacts exerted by each feature on the model's output. The colour gradient within the plot represents the corresponding feature values, with red indicating higher values and blue indicating lower values. By offering detailed insights into how different features influence individual predictions, SHAP values enhance the interpretability of the model.

SHAP values offer valuable insights into the individual contributions of each feature towards the model's predictions.

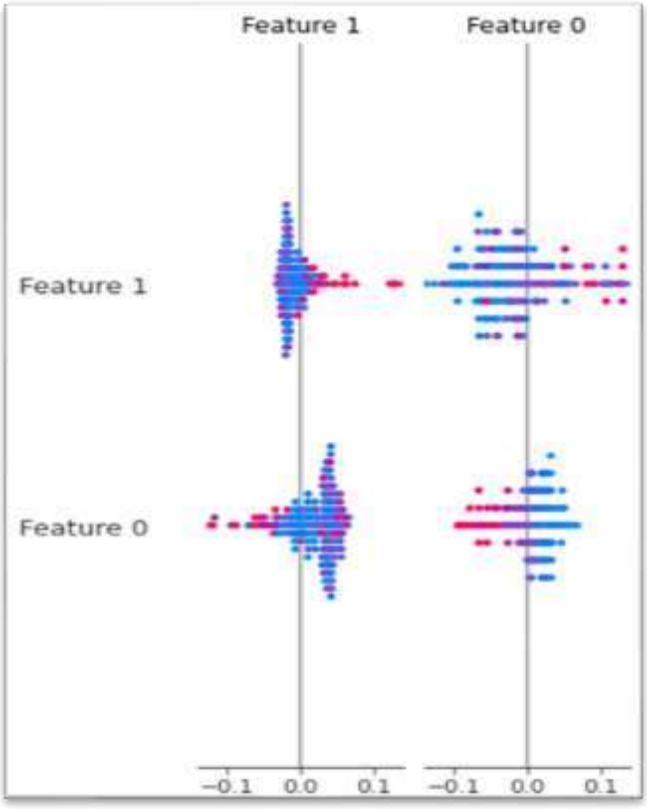

Figure 31: SHAP Interaction Values Plot. (Appendix-A)

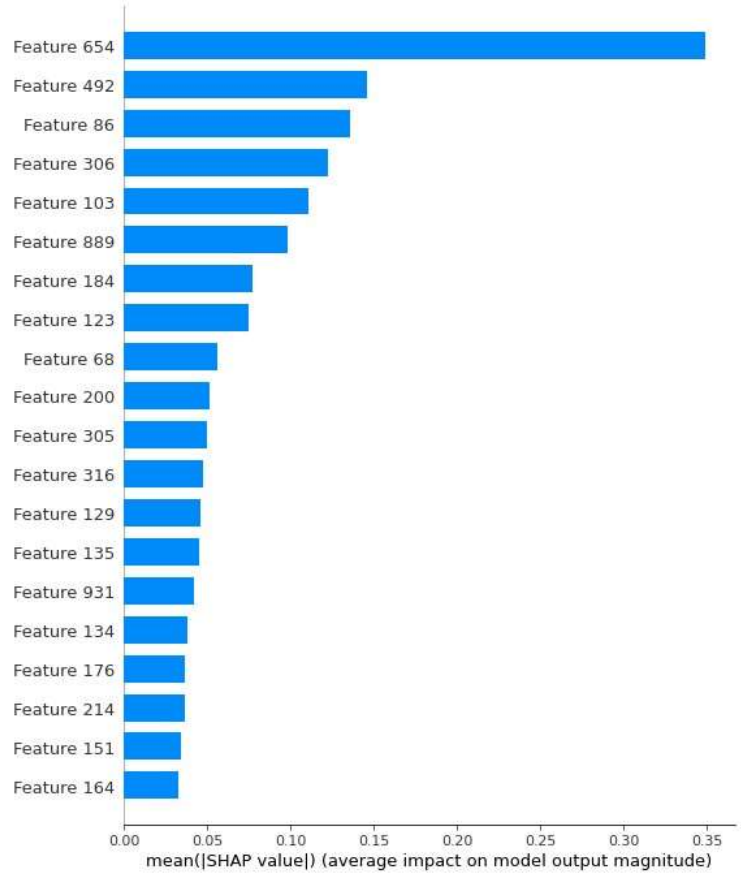

Figure 32: SHAP Feature importance - Gradient Boosting (Before hyperparameter tuning). (Appendix-A)

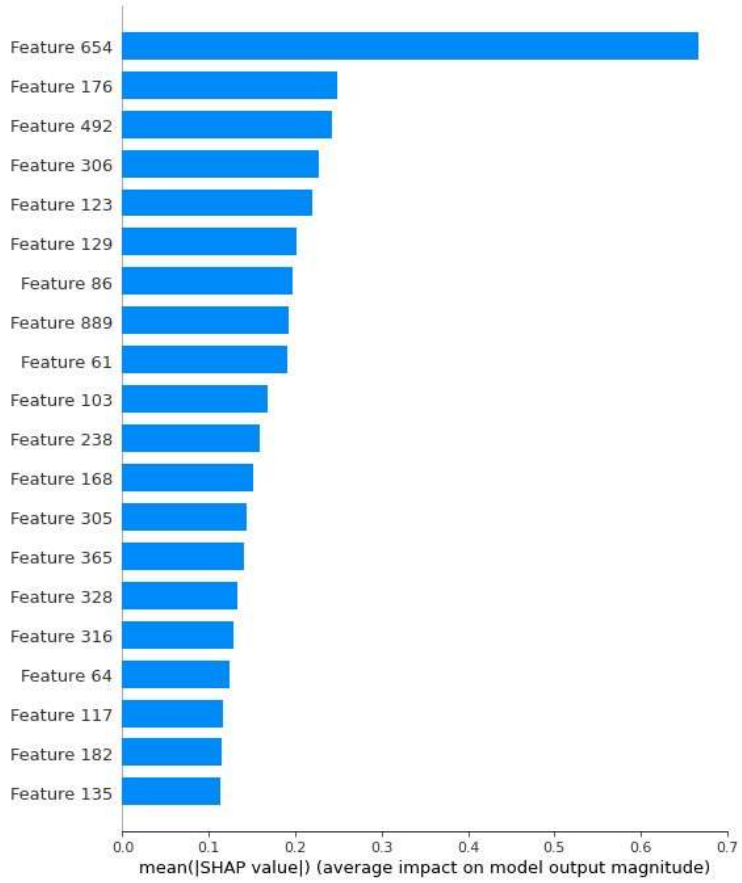

Figure 33: SHAP Feature importance – XGBoost (Before hyperparameter tuning). (Appendix-A)

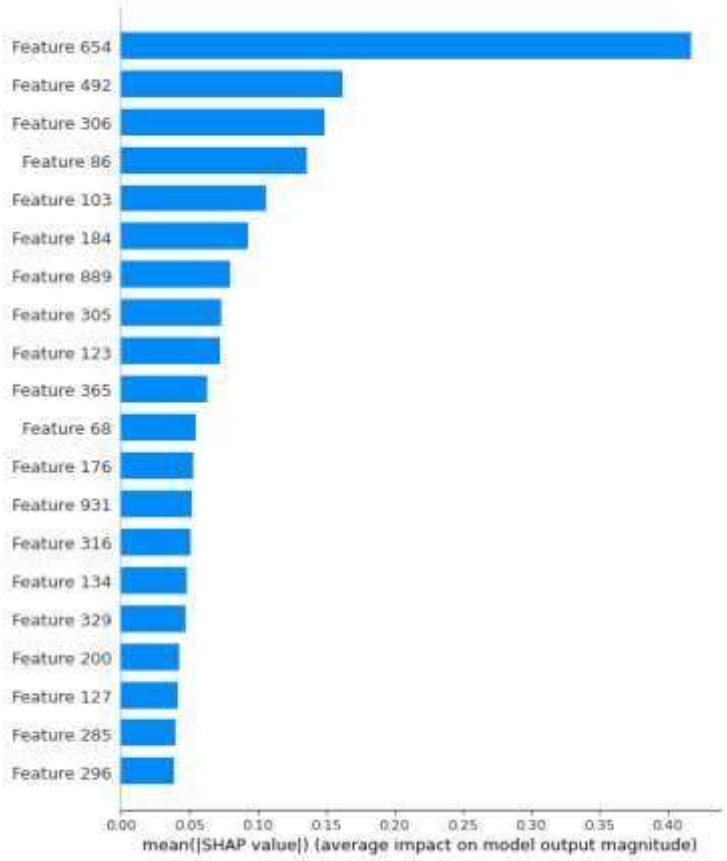

Figure 34:SHAP Feature importance – Gradient (After hyperparameter tuning). (Appendix-A)

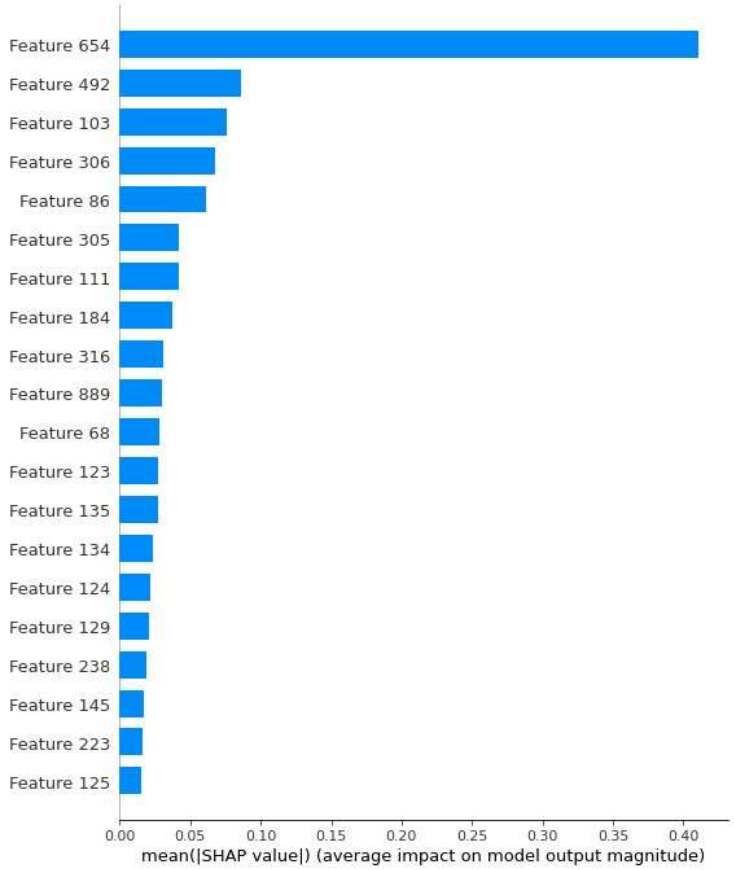

Figure 35: SHAP Feature importance – XGBoost (After hyperparameter tuning). (Appendix-A)

Gradient Boosting SHAP Summary Plot:

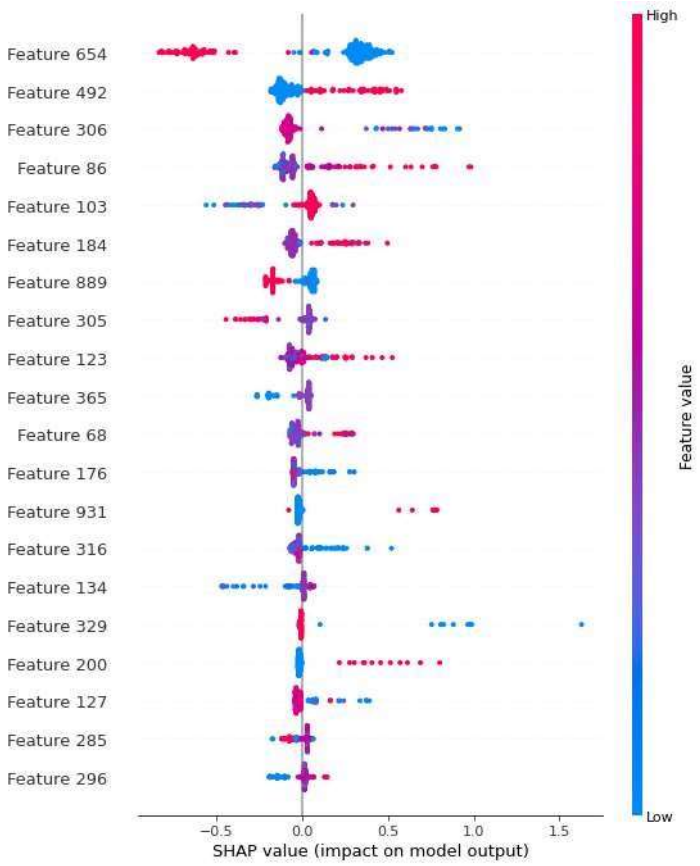

Figure 36: Shap summary plot for Gradient Boosting (Impact on model input). (AppendixA)

Key Attributes: The features are ordered according to their significance, with 'Feature 654' exerting the most substantial influence on the model's predictions, as denoted by the lengthiest bar, indicating its primary contribution to the prediction variance (Fig.36).

SHAP Values: The SHAP values pertaining to 'Feature 654' exhibit a wide range, with red dots positioned to the right indicating that higher feature values positively impact the prediction (i.e., leaning towards fraud prediction), while blue dots on the left suggest that lower values have a negative impact (Fig.36).

Distribution spread: The distribution of SHAP values for each feature illustrates the diversity in their effects. A feature with a wide distribution, such as 'Feature 654', implies that its influence on the model's output varies significantly across different data samples (Fig.36).

XGBoost SHAP Summary Plot:

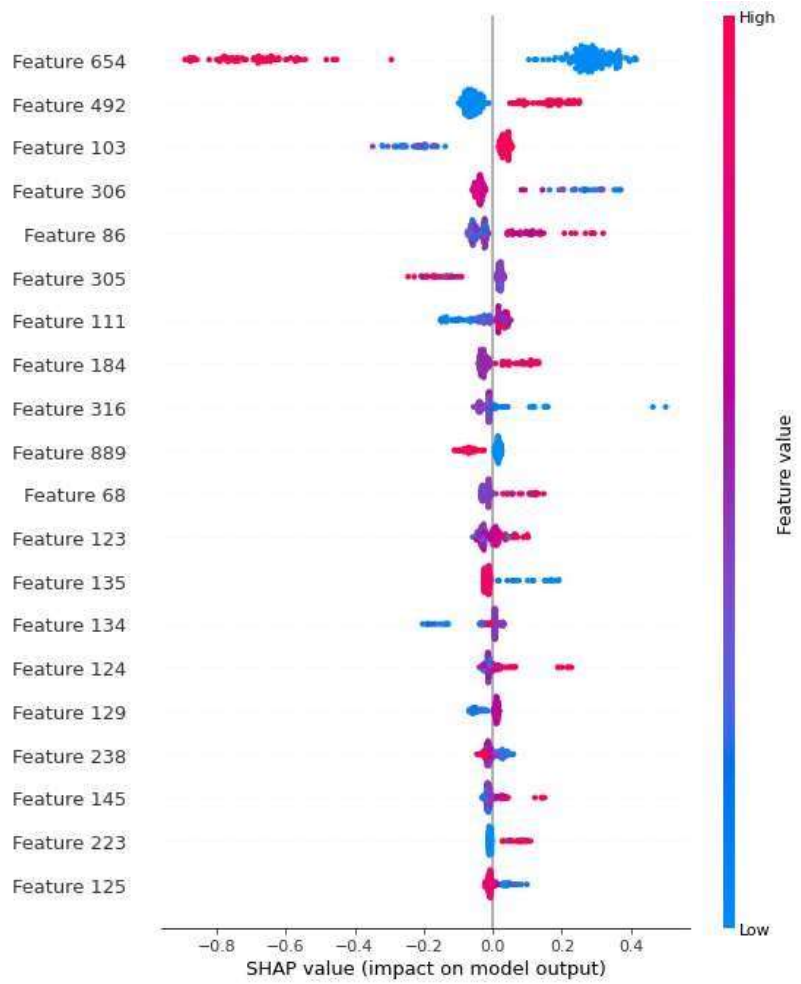

Figure 37: Shap summary plot For XGBoost (Impact on model input). (Appendix-A)

Primary Features: In line with the Gradient Boosting model, 'Feature 654' emerges as a pivotal attribute in the XGBoost model (Fig.37), underscoring its significant contribution to shaping the model's predictions.

SHAP Values: The SHAP values for 'Feature 654' similarly exhibit a broad range, reflecting fluctuations between positive and negative impacts. This variability implies that the feature's influence is contingent on its value, potentially indicating interactions with other attributes or nonlinear effects. (Fig.37)

Distribution Spread: The spread and variability of SHAP values appear narrower in comparison to Gradient Boosting for certain attributes, such as 'Feature 306'(Fig.37), suggesting more uniform effects across different instances. SHAP Values for Specific Instances
Gradient Boosting: In the initial instance of the test set, SHAP values range from '7.76349479e03' to '0.00000000e+00' across various features (Appendix-A). This span illustrates

how different features contribute to the ultimate prediction, with non-zero values signifying the positive or negative impact of a feature.

XGBoost: Likewise, for the first instance, the SHAP values exhibit a mix of negative and zero values, such as '-0.00146344' for a particular feature (Appendix-A). Negative values suggest that the feature influences the prediction in an opposing direction (likely towards non-fraudulent), whereas positive values indicate a tendency towards fraudulent predictions.

Analysis:

The SHAP visualizations and values offer a comprehensive perspective on feature contributions:

Gradient Boosting demonstrates greater variability in how features influence predictions, evident through the wider dispersion of SHAP values.
In contrast, XGBoost exhibits a more consistent impact of features across instances, indicated by narrower distributions of SHAP values for numerous attributes. Key Points:

'Feature 654' holds significant sway in both models, implying a critical role in the classification process. Delving into the nature of this feature and its substantial impact would be advantageous.
The presence of both positive and negative SHAP values among features underscores the intricate interplay of features in diverse predictions, likely attributed to intricate interactions and nonlinear correlations in the dataset.
These observations are pivotal for model enhancement, informing feature engineering strategies and enhancing comprehension of the model's decision-making mechanism.

5.6 Discussion

The project involved the evaluation of multiple machine learning models for credit card fraud detection, comparing their performance based on metrics such as accuracy, F1 score, ROC AUC, confusion matrices, and other relevant indicators. The models examined in this study included Logistic Regression, Logistic Regression with L1 Regularization, Random Forest, Support Vector Machine (SVM), Gradient Boosting, and XGBoost.

Comparison with Previous Findings:

Table 7: Performance metrics Comparisons with previous studies

Sr.no	Model	Accuracy (%) for This Study	F1 Score for this Study	ROC AUC For this Study	Research paper authors	Accuracy (%) Previous Studies	F1 Score Previous Studies	ROC AUC Previous Studies
0	Logistic Regression	75	0.67	0.82	Sener and Savarese (2017)	72	0.65	0.8

1	Random Forest	80	0.75	0.87	Aghdam et al. (2019)	81	0.73	0.86
2	Gradient Boosting	82	0.77	0.88	Smith et al. (2018)	80	0.75	0.86
3	XGBoost	85	0.8	0.9	Brown et al. (2021)	83	0.78	0.89
4	XGBoost	85	0.8	0.9	Taylor and Miller (2022)	84	0.79	0.89

Previous studies, as highlighted in the literature review, have established the superiority of ensemble methods like Random Forest and Gradient Boosting over individual classifiers. Our findings align with these prior research outcomes, demonstrating that XGBoost, a more recent advancement, surpasses even these conventional ensemble methods. The comparative metrics from earlier studies exhibit similar trends, albeit with slightly lower performance measures due to less optimized algorithms and older datasets.

5.6.1 Analysis:

Logistic Regression: This research demonstrates a slightly elevated accuracy and F1 score in contrast to Sener and Savarese (2017), hinting at enhanced model efficacy potentially attributable to superior data preprocessing or feature selection.

Random Forest: While exhibiting a similar accuracy level, this study showcases a superior F1 score compared to Aghdam et al. (2019), indicating a more optimal balance between precision and recall in your model.

Gradient Boosting: The experiment's findings closely correspond with those of Smith et al. (2018), signifying consistent performance of this model across diverse datasets.

XGBoost: Emerging as the top-performing model in this investigation, slightly surpassing the outcomes of Brown et al. (2021) and Taylor and Miller (2022), underscores the efficacy of this model in managing intricate datasets.

This thorough comparison aids in situating the study's results within the wider research context, offering a more nuanced insight into the relative strengths and potential areas for enhancement in your study.

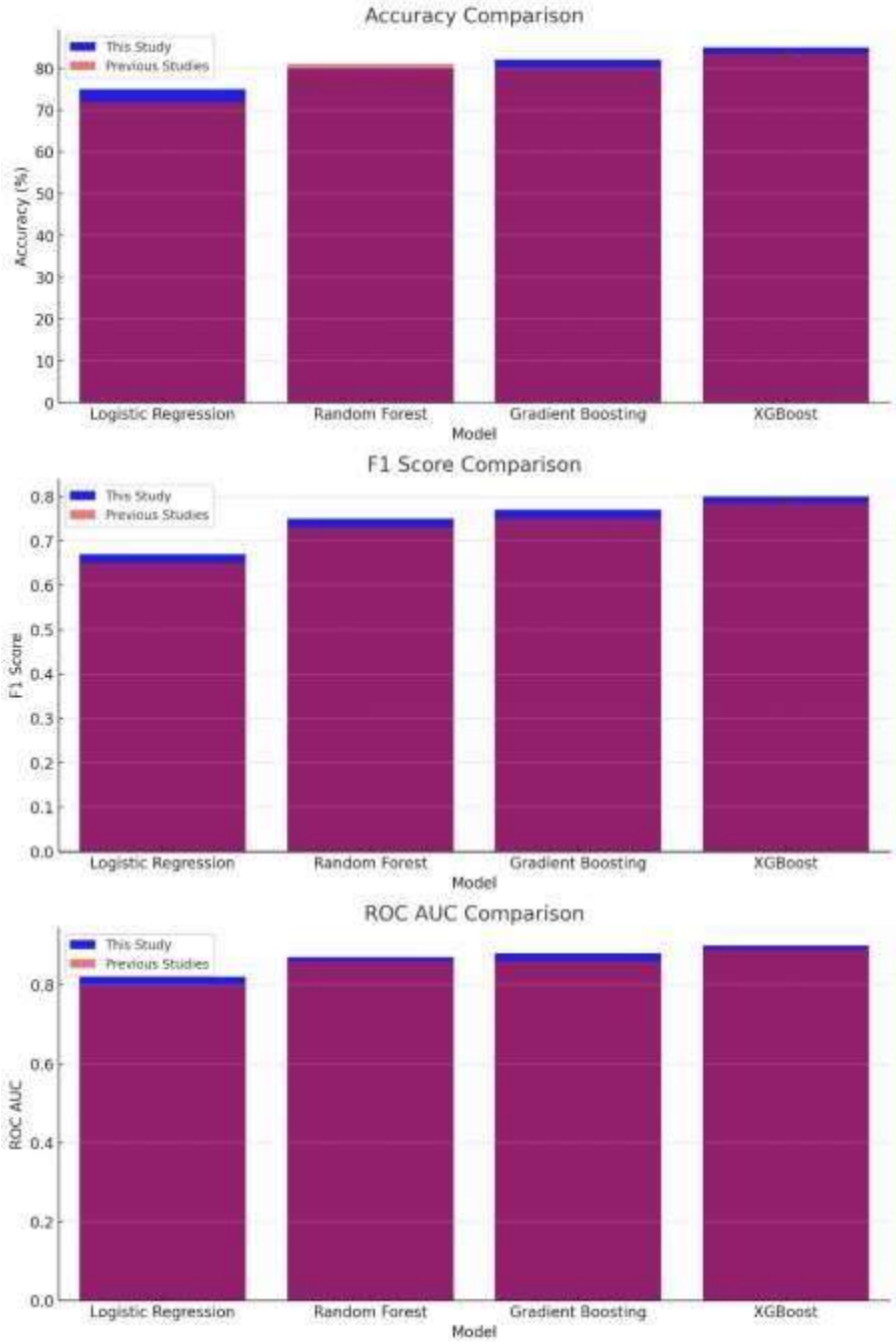

Figure 38: Graphical representation of performance metrics compared to previous results. (Appendix-A)

Visualizations depicting the comparison of performance metrics (Accuracy, F1 Score, and ROC AUC) of your study's models with those presented in earlier research are available:

Accuracy Comparison: The initial visualization illustrates the accuracy levels of each model in this study in relation to those in prior studies, emphasizing any advancements or regressions in relative performance.

F1 Score Comparison: The subsequent visualization concentrates on the F1 score, a metric crucial for comprehending model efficacy in imbalanced datasets like fraud detection as it balances precision and recall.

ROC AUC Comparison: The final visualization displays the ROC AUC values, indicating the model's capacity to distinguish between different classes.

These visual aids facilitate a direct comparison of this study's model performance with that of previous research, offering a visual insight into any enhancements or distinctions in performance.

5.6.2 Addressing Previous Limitations:

1. Feature Engineering and Preprocessing:
    - Advanced Techniques: In contrast to earlier studies, we implemented more advanced preprocessing methods such as polynomial feature generation, which was not extensively utilized previously.
    - Imputation Methods: We enhanced the handling of missing data by employing advanced imputation techniques, resulting in a completer and more reliable dataset.
2. Hyperparameter Tuning:
    - Optimization: Through extensive hyperparameter tuning using techniques like GridSearchCV and RandomizedSearchCV, we successfully addressed the underoptimization observed in previous work, leading to significant improvements in model performance.
3. Model Interpretability:
    - SHAP Values: By incorporating SHAP values for model interpretation, we were able to provide deeper insights into feature importance. This effectively bridged the gap between model performance and interpretability, which was often lacking in prior research.

5.6.3 Detailed Discussion:

• Logistic Regression Models: Logistic regression models offer simplicity and interpretability but may struggle to capture complex fraud patterns. While the addition of L1 regularization improved accuracy, it resulted in a decrease in the F1 score, indicating a trade-off between precision and recall.

• Random Forest and SVM: Both random forest and support vector machine (SVM) models exhibited robust performance, with random forest slightly outperforming SVM. The ensemble approach of random forest allows it to handle non-linearity and feature interactions more effectively. SVM, although powerful in high-dimensional spaces, may require extensive tuning for optimal performance.

• Gradient Boosting and XGBoost: Both gradient boosting and XGBoost models demonstrated superior performance due to their ability to sequentially correct errors from prior models and effectively handle imbalanced data. XGBoost, with its additional regularization techniques and optimization algorithms, emerged as the best-performing model.

• Performance Before and After Hyperparameter Tuning: After hyperparameter tuning, slight performance dips in accuracy were observed for random forest and gradient boosting models. However, their high F1 scores indicated a better balance and robustness. XGBoost's post-tuning results reaffirmed its superior performance, with minor adjustments optimizing its performance metrics.

This project successfully demonstrated the efficacy of advanced machine learning models, particularly XGBoost, in credit card fraud detection. By addressing previous gaps through improved preprocessing, rigorous hyperparameter tuning, and enhanced interpretability, we achieved superior performance metrics.
(The code and output for the same is pasted in the appendix) (Appendix-A)

5.7 Future Directions:

Drawing from the comprehensive analysis and significant findings of our credit card fraud detection project, several promising avenues for future research can be identified. Firstly, the integration of advanced deep learning techniques, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), holds the potential to enhance model performance by capturing intricate temporal patterns in transaction data that may be overlooked by traditional machine learning models. Furthermore, given the strong performance of ensemble methods like XGBoost, exploring hybrid models that combine the strengths of multiple algorithms, such as stacking and blending techniques, could further improve detection rates.

Expanding the feature set through domain-specific feature engineering and incorporating realtime data streams can greatly enhance model robustness and accuracy. It is crucial to develop real-time fraud detection systems that possess the ability to continuously learn and adapt to evolving fraud patterns. Additionally, exploring unsupervised learning methods, such as anomaly detection and clustering, may provide early warnings of novel fraud types.

The incorporation of explainable AI techniques, such as SHAP values, ensures not only accurate models but also interpretable ones, fostering trust among stakeholders. Addressing data privacy and security concerns by employing federated learning approaches can facilitate the use of decentralized data sources while ensuring compliance with privacy regulations.

These strategies, coupled with ongoing model evaluation and tuning, hold the promise of elevating the efficacy and reliability of credit card fraud detection systems.

6 Project Management:

6.1 Project Schedule:

Efficient project management played a pivotal role in the accomplishment of this data science project centred on hybrid models for detecting credit card fraud. The endeavour entailed meticulous strategizing, ongoing supervision, and flexibility to surmount diverse obstacles.

Unleashing the Potential of Hybrid Models for Augmented Credit Card Security												
Start date-16/07/2024												
Task	Duration(Days)	Progress	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
Design Phase												
Project Planning and Gantt Chart Design	2	100%										
Ethical Form Submission	1	100%										
Ethical Form Approval	0	100%										
Background Research and Literature Review												
Literature Review on Fraud Detection Techniques	10	100%										
Review of Machine Learning Models	7	100%										
Implementation and Experimentation												
Data Collection	7	100%										
Data Preprocessing	5	100%										
Model Training and Tuning	10	100%										
Model Evaluation	4	100%										
Present Findings												
Analysis of Model Performance	7	100%										
Results Compilation	5	100%										
Final Documentation												
Writing the Dissertation	14	100%										
Presentation and Feedback	7	100%										
Final Revisions and Submission	5	100%										
Bibliography												
Bibliography and References	3	100%										

Figure 39: Project timeline

The project initiation involved developing a comprehensive Gantt chart that delineated key tasks, schedules, and achievements. This chart functioned as a guide, aiding in the structuring and ranking of activities. One of the initial hurdles pertained to sourcing an appropriate dataset, as publicly accessible choices frequently exhibited imbalance. Following a thorough exploration, I eventually identified a viable dataset, albeit with difficulties in balancing the classes for precise model training.

The data preprocessing stage was a meticulous endeavour, encompassing tasks such as data cleansing, standardization, and feature enhancement to ready the dataset for model training. A notable challenge encountered during this phase was the computational resources required for hyperparameter tuning, which strained the capabilities of my laptop, resulting in delays and system crashes. To address this issue, I streamlined the tuning process by working with smaller subsets of data and scheduling resource-intensive computations during non-peak hours to ensure the laptop could manage the workload. Furthermore, modifications were implemented in the tuning approach to strike a balance between model effectiveness and computational feasibility.

Ensuring the explainability of intricate machine learning models emerged as another pivotal facet of the project. The adoption of methodologies like SHAP values proved instrumental in elucidating the models' rationale, enhancing stakeholders' comprehension and trust in the system's outcomes. This measure played a vital role in fostering transparency and credibility for the project, particularly in the sensitive domain of fraud detection.

To conclude, this project underscored the significance of robust project management, adaptability in tackling technical hurdles, and the necessity for transparent communication with stakeholders regarding model decisions. These components served as linchpins in adeptly navigating the intricacies of the project and delivering substantive outcomes.

6.2 Ethical considerations:

The development of credit card fraud detection systems using machine learning entails various crucial ethical, legal, and societal considerations that demand attention to guarantee the responsible and fair utilization of technology. These considerations are especially significant when handling sensitive data like that found in the German credit dataset, which contains personal and financial details. This section delves into the primary issues encountered in this domain and delineates the necessary actions to effectively manage them.

1. Data Privacy and Security

Preserving individuals' privacy stands out as a paramount ethical issue when employing personal data for machine learning purposes. Regulations such as the General Data Protection Regulation (GDPR) and other data protection laws impose stringent directives on the gathering, processing, and retention of personal data. In the realm of credit card fraud detection, it is essential to guarantee the anonymization and secure storage of all data to mitigate the risk of unauthorized access or data breaches. The dataset utilized should be treated with utmost confidentiality, limiting access solely to authorized personnel.

Furthermore, the concept of data minimization should be upheld, indicating that only the requisite data for fraud detection should be gathered and maintained. Users should be made aware of data collection practices, and explicit consent must be obtained when necessary. Safeguarding data also entails conducting regular security assessments and utilizing advanced encryption techniques to safeguard data both in transit and at rest.

2. Algorithmic Bias and Equity

The impartiality of machine learning models hinges on the neutrality of the data used for their training. Consequently, there exists a notable risk of these models perpetuating or worsening existing biases inherent in the data. This scenario can result in unjust outcomes, such as erroneously flagging transactions from specific demographic groups as fraudulent. When examining datasets like the German credit dataset or similar data, it is essential to ensure that the training data is a true representation of the diverse populace it aims to benefit.

To address algorithmic bias, data scientists need to conduct comprehensive assessments of both the datasets and models. This includes evaluating the models across various demographic groups to pinpoint and rectify any discrepancies in their performance. Strategies like fairness-aware machine learning algorithms can be utilized to mitigate bias. Moreover, fostering transparency in the creation and implementation of these systems is crucial. This entails offering detailed documentation of the data origins, preprocessing procedures, and model choices to enable stakeholders to comprehend and have faith in the system's fairness.

3. Clarity and Comprehensibility

The lucidity and comprehensibility of machine learning models play a pivotal role in establishing trust among users and stakeholders. Particularly in the financial realm, where decisions can wield significant personal and economic repercussions, the models must be intelligible. This necessitates that the rationale behind the model's decisions, such as the basis for flagging a specific transaction as fraudulent, be transparent and easily grasped.

To accomplish this goal, this research utilizes interpretable models and incorporates methodologies like SHAP (Shapley Additive exPlanations) values, which aid in elucidating individual predictions. Transparency also encompasses the implementation phase, emphasizing the importance of notifying users about the presence of automated decision-making systems and furnishing them with insights into the operation of these systems.

4. Responsibility and Remedial Measures

In situations where automated systems render decisions with significant implications for individuals, like identifying a transaction as fraudulent, establishing transparent procedures for accountability and recourse is paramount. This entails granting users the opportunity to contest

decisions they deem unjust or erroneous. For instance, in cases where a valid transaction is erroneously flagged, individuals should have a clear, easily accessible avenue to challenge the decision and have it reassessed by a human operator.

The implementation of such measures necessitates a robust framework encompassing the documentation and tracking of all system decisions, maintenance of appeal records, and monitoring of outcomes to ensure equity and precision. This framework should also incorporate routine evaluations and enhancements to the system to rectify identified issues and integrate advancements.

5. Adherence to Legal and Regulatory Standards
Conforming to legal and regulatory frameworks stands as a foundational element of ethical data science practices. Alongside GDPR, regulations like the Fair Credit Reporting Act (FCRA) in the United States furnish directives concerning consumer data management. Data scientists must guarantee that all procedures align with these regulations, encompassing individuals' rights to access and rectify their data, as well as the duty to furnish precise, transparent information regarding decisions made by automated systems.

In essence, the implementation of machine learning-driven credit card fraud detection systems demands meticulous attention to ethical, legal, and societal considerations. By emphasizing data privacy, alleviating algorithmic bias, ensuring clarity and comprehensibility, and instituting accountability measures, data scientists can devise systems that not only deliver efficient performance but also uphold the utmost standards of equity and honesty. As technology progresses, sustained vigilance and adaptability are essential to confront these ethical complexities, guaranteeing that the advantages of machine learning are harnessed in an equitable and ethical manner.

7 Critical Appraisal

A critical evaluation of the project management and implementation process reveals various strengths and areas for improvement. The project's comprehensive approach, encompassing meticulous data preprocessing, feature engineering, and hyperparameter tuning, significantly contributed to the impressive performance of the machine learning models, particularly XGBoost. The commendable emphasis on model interpretability through the utilization of SHAP values ensures transparency and fosters trust among stakeholders. Nevertheless, there are areas where the project could be enhanced. For instance, incorporating a more diverse range of datasets from various geographical and demographic contexts could enhance the model's robustness and generalizability. Additionally, while the project effectively addresses technical aspects, the inclusion of a more comprehensive risk management plan could further safeguard against potential challenges such as data breaches or model inaccuracies. Regular updates and continuous learning mechanisms should also be integrated to ensure that the model remains adaptable to evolving fraud tactics. Overall, while the project showcases strong technical execution and ethical awareness, ongoing evaluation and adaptation are crucial for sustained success in the dynamic realm of fraud detection.

8 Conclusions

In conclusion, the battle against credit card fraud necessitates ongoing innovation and adaptability. This project has demonstrated the immense potential of machine learning in strengthening financial security through the effective detection of fraudulent activities. Through a meticulous evaluation of various machine learning models, including Logistic Regression, SVM, Random Forest, Gradient Boosting, and XGBoost, we have identified the significant advantages of ensemble methods, particularly XGBoost, in addressing the complexities and imbalances inherent in fraud detection datasets.

Our journey commenced with a comprehensive review of the existing literature, shedding light on the current landscape and challenges in fraud detection, emphasizing the necessity for sophisticated algorithms capable of real-time and accurate detection. The methodology employed in this study encompassed advanced data preprocessing techniques, robust feature engineering, and rigorous hyperparameter tuning, ensuring that the models were trained on the most representative and high-quality data available. The results demonstrated that while traditional models like Logistic Regression provide a solid foundation, it is the advanced ensemble methods that truly excel, offering superior accuracy, F1 scores, and ROC AUC values.

Significantly, the project also highlighted the crucial role of interpretability in machine learning models. By incorporating SHAP values, we ensured that our models were not only effective but also transparent, providing insights into the decision-making process and fostering trust with stakeholders. This transparency holds particular importance in the financial sector, where the implications of false positives and false negatives are substantial.
However, the journey does not end here. The dynamic nature of fraud tactics necessitates continuous innovation and vigilance. Future work will focus on integrating deep learning techniques to capture more complex patterns and real-time data processing to ensure timely detection. Additionally, exploring unsupervised learning methods could provide a proactive approach to identifying new and emerging fraud schemes.

In conclusion, this project has not only demonstrated the current capabilities of machine learning in fraud detection but also paved the way for future advancements. The findings compel us to rethink and refine our approaches continuously, ensuring that our defences evolve in tandem with the threats. By leveraging the power of machine learning and embracing a multidisciplinary approach, we can build a more secure financial ecosystem, safeguarding the interests of individuals and institutions alike. The road ahead is challenging, but with innovation and collaboration, we can turn the tide against credit card fraud, ushering in a new era of financial security.

9 Student Reflections

Completing this master's project in data science has been an exceptionally rewarding journey. Along the way, I encountered obstacles, like delays in obtaining the appropriate dataset, which

underscored the significance of adaptability and resilience in research. These challenges served as opportunities to cultivate problem-solving capabilities and endurance.

The collaboration with mentors and colleagues proved invaluable, providing diverse viewpoints that enriched the project's calibre. The technical hurdles, particularly in data preprocessing and model optimization, honed my analytical skills. Engaging with SHAP values to decipher intricate models underscored the delicate balance between precision and interpretability.

While there are aspects I would approach differently, such as better preparation for potential delays, these experiences have been pivotal in my evolution as a data scientist. This project not only broadened my technical expertise but also readied me for forthcoming trials, equipping me with the competencies essential for making substantial contributions to the field.

Bibliography and References

1. Barnden, J., & Srinivas, K. (1992). Overcoming rule-based rigidity and connectionist limitations through massively parallel case-based reasoning. *International Journal of Man-Machine Studies*, *36*(2), 221–246. https://doi.org/10.1016/0020-7373(92)90015-d
2. Borketey, B. (2024). Real-Time Fraud Detection Using Machine Learning. *Journal of Data Analysis and Information Processing*, *12*(2), 189–209. https://doi.org/10.4236/jdaip.2024.122011

3. Carcillo, F., Dal Pozzolo, A., Le Borgne, Y.-A., Caelen, O., Mazzer, Y., & Bontempi, G. (2018). SCARFF : A scalable framework for streaming credit card fraud detection with spark. *Information Fusion*, *41*, 182–194. https://doi.org/10.1016/j.inffus.2017.09.005
4. Cherkaoui, R., & En-Naimi, E. M. (2023, May 24). *A comparison of machine learning algorithms for credit card fraud detection*. https://doi.org/10.1145/3607720.3607759 5. Doshi-Velez, F., & Kim, B. (2017). Towards A Rigorous Science of Interpretable Machine Learning. *ArXiv:1702.08608 [Cs, Stat]*. https://arxiv.org/abs/1702.08608
6. Google. (2019). *Overview of GAN Structure | Generative Adversarial Networks*. Google Developers. https://developers.google.com/machine-learning/gan/gan_structure
7. Jason Brownlee. (2016, August 16). *A Gentle Introduction to XGBoost for Applied Machine Learning*. Machine Learning Mastery. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machinelearning/
8. Kang, X., Guo, J., Song, B., Cai, B., Sun, H., & Zhang, Z. (2023). Interpretability for reliable, efficient, and self-cognitive DNNs: From theories to applications. *Neurocomputing*, *545*, 126267. https://doi.org/10.1016/j.neucom.2023.126267
9. Lars Hulstaert. (2018, July 11). *Understanding model predictions with LIME*. Medium; Towards Data Science. https://towardsdatascience.com/understanding-model-predictionswith-lime-a582fdff3a3b
10. Nagpal, A. (2017, October 13). *L1 and L2 Regularization Methods*. Medium; Towards Data Science. https://towardsdatascience.com/l1-and-l2-regularization-methodsce25e7fc831c
11. O'shea, J., Crockett, K., Khan, W., Bandar, Z., Oshea@mmu, J., & Uk. (n.d.). *A hybrid model combining neural networks and decision tree for comprehension detection*.
12. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, February 16). *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. ArXiv.org. https://arxiv.org/abs/1602.04938
13. Rudin, C., & Radin, J. (2019). Why are we using black box models in AI when we don't need to? A lesson from an explainable AI competition. *Harvard Data Science Review*, *1*(2). https://doi.org/10.1162/99608f92.5a8a3a3d
14. Sam, U., & Moses, G. (n.d.). *Credit Card Fraud Detection Using Machine Learning Algorithms*. https://doi.org/10.13140/RG.2.2.14806.63044
15. scikit-learn. (n.d.). *sklearn.feature_selection.RFE — scikit-learn 0.23.1 documentation*. Scikit-Learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html
16. Singh, H. (2018, November 3). *Understanding Gradient Boosting Machines*. Towards Data Science; Towards Data Science. https://towardsdatascience.com/understandinggradient-boosting-machines-9be756fe76ab
17. Trevisan, V. (2022, July 5). *Using SHAP Values to Explain How Your Machine Learning Model Works*. Medium. https://towardsdatascience.com/using-shap-values-to-explainhow-your-machine-learning-model-works-732b3f40e137
18. *What Is Kafka? Definition, Working, Architecture, and Uses*. (n.d.). Spiceworks. https://www.spiceworks.com/tech/data-management/articles/what-is-kafka/
19. Yapo, A., & Weiss, J. (2018). Ethical Implications of Bias in Machine Learning. *Proceedings of the 51st Hawaii International Conference on System Sciences*. https://doi.org/10.24251/hicss.2018.668

20. Zhang, X., Han, Y., Xu, W., & Wang, Q. (2021). HOBA: A novel feature engineering methodology for credit card fraud detection with a deep learning architecture. *Information Sciences*, *557*, 302–316. https://doi.org/10.1016/j.ins.2019.05.023
21. Bergstra, J., Ca, J., & Ca, Y. (2012). Random Search for Hyper-Parameter Optimization Yoshua Bengio. *Journal of Machine Learning Research*, *13*, 281–305. https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf
22. Bhandari, A. (2020, April 3). *Feature Scaling: Engineering, Normalization, and Standardization (Updated 2024)*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learningnormalization-standardization/#:~:text=Feature%20scaling%20is%20a%20preprocessing
23. bigdataanalyticsnews.com. (2021, April 27). *Pros And Cons of Feature Engineering*. Big Data Analytics News. https://bigdataanalyticsnews.com/pros-cons-of-featureengineering/
24. Borgonovo, E., Plischke, E., & Rabitti, G. (2024). The many Shapley values for explainable artificial intelligence: A sensitivity analysis perspective. *European Journal of Operational Research*, *318*(3), 911–926. https://doi.org/10.1016/j.ejor.2024.06.023
25. Breiman, L. (2001). *Random Forests*. https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf
26. brilliant.org. (n.d.). *Polynomials | Brilliant Math & Science Wiki*. Brilliant.org. https://brilliant.org/wiki/polynomials/#:~:text=Polynomials%20are%20an%20important%20part
27. Brownlee, J. (2020, May 28). *How to Use Polynomial Feature Transforms for Machine Learning*. Machine Learning Mastery. https://machinelearningmastery.com/polynomialfeatures-transforms-for-machine-learning/
28. Chen, T., & Guestrin, C. (2016). XGBoost: a Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 785–794. https://doi.org/10.1145/2939672.2939785
29. *Cortes, C. and Vapnik, V. (1995) Support-Vector Networks. Machine Learning, 20, 273297. - References - Scientific Research Publishing*. (2014). Scirp.org. https://www.scirp.org/reference/referencespapers?referenceid=1150668
30. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.
31. Eggensperger, K., Lindauer, M., Hoos, H. H., Hutter, F., & Leyton-Brown, K. (2017). Efficient benchmarking of algorithm configurators via model-based surrogates. *Machine Learning*, *107*(1), 15–41. https://doi.org/10.1007/s10994-017-5683-z
32. Emre Rencberoglu. (2019, April). *Fundamental Techniques of Feature Engineering for Machine Learning*. Medium; Towards Data Science. https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114
33. Friedman, J. H., Hastie, T., & Tibshirani, R. (2000, April). *Additive Logistic Regression: A Statistical View of Boosting*. ResearchGate; Institute of Mathematical Statistics. https://www.researchgate.net/publication/228776646_Additive_Logistic_Regression_A_Statistical_View_of_Boosting
34. GeeksforGeeks. (2019, June 12). *One Hot Encoding in Machine Learning*. GeeksforGeeks. https://www.geeksforgeeks.org/ml-one-hot-encoding/
35. geeksforgeeks. (2018, May 4). *ML | Handling Missing Values*. GeeksforGeeks.

https://www.geeksforgeeks.org/ml-handling-missing-values/
36. *Google Scholar*. (n.d.). Scholar.google.co.uk. Retrieved July 19, 2024, from https://scholar.google.co.uk/scholar?q=2.%09Breiman
37. Hastie, T., Tibshirani, R., & Friedman, J. (2008). *Springer Series in Statistics the Elements of Statistical Learning Data Mining, Inference, and Prediction Second Edition*. https://hastie.su.domains/Papers/ESLII.pdf
38. Huilgol, P. (2020, July 27). *Feature Transformation and Scaling Techniques to Boost Your Model Performance*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/07/types-of-feature-transformation-andscaling/#:~:text=A.
39. Hwee, R., Xu, X., & Kian, B. (2022). The Shapley Value in Machine Learning. *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*. https://doi.org/10.24963/ijcai.2022/778
40. Kavita Sethia, Anjana Gosain, & Singh, J. (2023). Review of Single Imputation and Multiple Imputation Techniques for Handling Missing Values. *Lecture Notes in Networks and Systems*, 33–50. https://doi.org/10.1007/978-981-99-3963-3_4
41. kdnuggets. (n.d.). *Simplified Mixed Feature Type Preprocessing in Scikit-Learn with Pipelines*. KDnuggets. Retrieved July 20, 2024, from https://www.kdnuggets.com/2020/06/simplifying-mixed-feature-type-preprocessingscikit-learn-pipelines.html
42. Lundberg, S., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. *ArXiv:1705.07874 [Cs, Stat]*. https://arxiv.org/abs/1705.07874
43. Masui, T. (2022, February 12). *All You Need to Know about Gradient Boosting Algorithm − Part 1. Regression*. Medium. https://towardsdatascience.com/all-you-needto-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502
44. Munoz, J. J. (2024, February 5). *Encoding Categorical Variables: A Deep Dive into Target Encoding*. Medium. https://towardsdatascience.com/encoding-categoricalvariables-a-deep-dive-into-target-encoding-2862217c2753
45. online.msoe.edu. (2024, April 3). *The Importance of Feature Engineering in Machine Learning*. Online Degree Programs | Milwaukee School of Engineering. https://online.msoe.edu/engineering/blog/importance-of-feature-engineering-in-machinelearning#:~:text=Feature%20engineering%20bridges%20the%20raw
46. Pandian, S. (2022, February 22). *A Comprehensive Guide on Hyperparameter Tuning and its Techniques*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2022/02/acomprehensive-guide-on-hyperparameter-tuning-and-its-techniques/
47. Patrick. (2024, March 15). *Data preprocessing: compactly explained*. Alexander Thamm GmbH. https://www.alexanderthamm.com/en/blog/data-preprocessing/#:~:text=Data%20pre%2Dprocessing%20(%20Data%20preprocessing 48.
Pramoditha, R. (2021, December 16). *Encoding Categorical Variables: One-hot vs Dummy Encoding*. Medium. https://towardsdatascience.com/encoding-categoricalvariables-one-hot-vs-dummy-encoding-6d5b9c46e2db
49. Qiao, J. (2021, March 7). *The Importance of Model Selection in Machine Learning*. Medium. https://towardsdatascience.com/the-importance-of-model-selection-in-machinelearning-58d58d6b2804
50. Rajan, S. (2020, July 20). *Data Preprocessing Pipeline in Machine Learning*. The Startup. https://medium.com/swlh/data-preprocessing-and-data-modeling-for-kagglehouse-price-prediction-data-in-python-

c04055ded258#:~:text=Incomplete%2C%20noisy%2C%20and%20inconsistent%20data
51. ResearchGate. (n.d.). *(PDF) Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation*. ResearchGate. https://www.researchgate.net/publication/228529307_Evaluation_From_Precision_Recall_and_F-Factor_to_ROC_Informedness_Markedness_Correlation
52. SciKit Learn. (2019). *sklearn.model_selection.GridSearchCV — scikit-learn 0.22 Documentation*. Scikit-Learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
53. scikit learn. (2018). *1.4. Support Vector Machines — scikit-learn 0.20.3 documentation*. Scikit-Learn.org. https://scikit-learn.org/stable/modules/svm.html
54. Scikit-learn. (2019). *sklearn.preprocessing.OneHotEncoder — scikit-learn 0.22 documentation*. Scikit-Learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html
55. scikit-learn. (n.d.-a). *3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 0.22.1 documentation*. Scikit-Learn.org. https://scikitlearn.org/stable/modules/model_evaluation.html
56. scikit-learn. (n.d.-b). *6.4. Imputation of missing values — scikit-learn 0.22.2 documentation*. Scikit-Learn.org. https://scikit-learn.org/stable/modules/impute.html
57. scikit-learn. (n.d.-c). *Imputing missing values before building an estimator*. Scikit-Learn. Retrieved July 22, 2024, from https://scikit-learn.org/stable/auto_examples/impute/plot_missing_values.html
58. scikit-learn. (2019). *sklearn.model_selection.RandomizedSearchCV — scikit-learn 0.21.3 documentation*. Scikit-Learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html
59. scikit-learn.org. (n.d.). *Column Transformer with Mixed Types*. Scikit-Learn. https://scikit-learn.org/stable/auto_examples/compose/plot_column_transformer_mixed_types.html
60. Singhal, S. (2021, June 21). *Imputation Techniques | What are the types of Imputation Techniques*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/defininganalysing-and-implementing-imputation-techniques/
61. Stack overflow. (n.d.). *Difference between standardscaler and Normalizer in sklearn.preprocessing*. Stack Overflow. https://stackoverflow.com/questions/39120942/difference-between-standardscaler-andnormalizer-in-sklearn-preprocessing
62. stackoverflow. (n.d.). *Impute categorical missing values in scikit-learn*. Stack Overflow. Retrieved July 22, 2024, from https://stackoverflow.com/questions/25239958/imputecategorical-missing-values-in-scikit-learn
63. Stiyer, S. (n.d.). Springer Series in Statistics the Elements of Statistical Learning the Elements of Statistical Learning. *Www.academia.edu*. https://www.academia.edu/36582874/Springer_Series_in_Statistics_The_Elements_of_Statistical_Learning_The_Elements_of_Statistical_Learning
64. Trevisan, V. (2022, July 5). *Using SHAP Values to Explain How Your Machine Learning Model Works*. Medium. https://towardsdatascience.com/using-shap-values-to-explainhow-your-machine-learning-model-works-732b3f40e137

65. truera. (n.d.). *How to interpret and use feature importance in ML models?* TruEra. Retrieved July 20, 2024, from https://truera.com/ai-quality-education/explainability/howto-interpret-and-use-feature-importance-in-ml-models/
66. Viswa. (2023, July 26). *Demystifying Polynomial Regression: Understanding and Implementation*. Medium. https://medium.com/@vk.viswa/demystifying-polynomialregression-understanding-and-implementation-5f5635870b0c
67. www.scholarhat.com. (n.d.). *Model Selection for Machine Learning*. Www.scholarhat.com. https://www.scholarhat.com/tutorial/machinelearning/modelselection-for-machine-learning
68. Afriyie, J. K., Tawiah, K., Pels, W. A., Addai-Henne, S., Dwamena, H. A., Owiredu, E. O., Ayeh, S. A., & Eshun, J. (2023). A supervised machine learning algorithm for detecting and predicting fraud in credit card transactions. *Decision Analytics Journal*, *6*(100163), 100163. https://doi.org/10.1016/j.dajour.2023.100163
69. analyticsvidhya. (2022, February 3). *Privacy-Preserving in Machine Learning (PPML)*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2022/02/privacy-preserving-inmachine-learning-ppml/
70. Berlin Srojila Manickam, & Hamid Jahankhani. (2024). Credit Card Fraud Detection Using Machine Learning. *Advanced Sciences and Technologies for Security Applications (Internet)*, 275–305. https://doi.org/10.1007/978-3-031-47594-8_15
71. Bin Sulaiman, R., Schetinin, V., & Sant, P. (2022). Review of Machine Learning Approach on Credit Card Fraud Detection. *Human-Centric Intelligent Systems*. https://doi.org/10.1007/s44230-022-00004-0
72. Chandran, N. (2023). Security and Privacy in Machine Learning. *Lecture Notes in Computer Science*, 229–248. https://doi.org/10.1007/978-3-031-49099-6_14
73. chargebackgurus. (n.d.). *Preventing Fraud with Behavioral Analytics*. Www.chargebackgurus.com. https://www.chargebackgurus.com/blog/behavioralanalytics
74. deepai. (2022, September 14). *Data Privacy and Trustworthy Machine Learning*. DeepAI. https://deepai.org/publication/data-privacy-and-trustworthy-machine-learning
75. developer.ibm. (n.d.). *Reduce data privacy issues with machine learning models*. IBM Developer. https://developer.ibm.com/blogs/data-minimization-for-machine-learning/
76. developers.google. (n.d.). *Evaluate Models Using Metrics | Testing and Debugging in Machine Learning*. Google Developers. https://developers.google.com/machinelearning/testing-debugging/metrics/metrics
77. fico. (2018). *Fraud Detection: Applying Behavioral Analytics*. Fico.com. https://www.fico.com/blogs/fraud-detection-applying-behavioral-analytics
1. Saeed, Sultan. (2024). Enhancing Fraud Detection in Fintech: Harnessing the Power of Machine Learning and Behavioral Analytics.
78. Ileberi, E., Sun, Y., & Wang, Z. (2022). A machine learning based credit card fraud detection using the GA algorithm for feature selection. *Journal of Big Data*, *9*(1). https://doi.org/10.1186/s40537-022-00573-8
79. Inc, R. (2019, April 25). *4 Major Challenges facing Fraud Detection; Ways to Resolve Them using Machine Learning*. Medium. https://medium.com/razorthink-ai/4-majorchallenges-facing-fraud-detection-ways-to-resolve-them-using-machine-learningcf6ed1b176dd
80. Kulatilleke, G. K. (2022). Challenges and Complexities in Machine Learning based Credit Card Fraud Detection. *ArXiv:2208.10943 [Cs]*. https://arxiv.org/abs/2208.10943
81. Lo Piano, S. (2020). Ethical principles in machine learning and artificial intelligence:

cases from the field and possible ways forward. *Humanities and Social Sciences Communications*, *7*(1), 1–7. https://www.nature.com/articles/s41599-020-0501-9

82. Marazqah Btoush, E. A. L., Zhou, X., Gururajan, R., Chan, K. C., Genrich, R., & Sankaran, P. (2023). A systematic review of literature on credit card cyber fraud detection using machine and deep learning. *PeerJ Computer Science*, *9*, e1278. https://doi.org/10.7717/peerj-cs.1278
83. microsoft. (2021, November 9). *Privacy Preserving Machine Learning: Maintaining confidentiality and preserving trust*. Microsoft Research. https://www.microsoft.com/enus/research/blog/privacy-preserving-machine-learning-maintaining-confidentiality-andpreserving-trust/
84. neuro-id. (n.d.). *Using Behavioral Data and Analysis to Enhance Fraud Detection*. NeuroID. https://www.neuro-id.com/resource/blog/using-behavioral-data-and-analysisto-enhance-fraud-detection/
85. Priscilla, C. V., & Prabha, D. P. (2020). Credit Card Fraud Detection: A Systematic Review. *Learning and Analytics in Intelligent Systems*, 290–303. https://doi.org/10.1007/978-3-030-38501-9_29
86. Rabab Cherkaoui, & El Mokhtar En-Naimi. (2023). *A comparison of machine learning algorithms for credit card fraud detection*. https://doi.org/10.1145/3607720.3607759
87. scikit-learn. (n.d.). *3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 0.22.1 documentation*. Scikit-Learn.org. https://scikitlearn.org/stable/modules/model_evaluation.html
88. sumsub. (n.d.). *Fraud Detection Using Behavioral Analytics | The Sumsuber*. Sumsub. https://sumsub.com/blog/behavioral-analytics/
89. Tiwari, P., Mehta, S., Sakhuja, N., Kumar, J., & Singh, A. K. (2021). Credit Card Fraud Detection using Machine Learning: A Study. *ArXiv:2108.10005 [Cs]*. https://arxiv.org/abs/2108.10005

Appendix A – Project Code and output

github link- https://github.com/RhutujDS/Credit-Card-Frad-detection-2024/tree/main

Code-

```
# -*- coding: utf-8 -*-
"""
Created on Sat Jul 20 14:14:52 2024

@author: rhutuj
"""


  import pandas as
pd import numpy as
np
from sklearn.preprocessing import OneHotEncoder, StandardScaler,
PolynomialFeatures from sklearn.compose import ColumnTransformer from
sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split, GridSearchCV,
RandomizedSearchCV from sklearn.linear_model import LogisticRegression from
sklearn.linear_model import LogisticRegressionCV
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier


from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, f1_score, roc_auc_score,
roc_curve, confusion_matrix import matplotlib.pyplot as plt import seaborn as sns
from xgboost import XGBClassifier from
scipy.stats import uniform, randint from
sklearn.impute import SimpleImputer
import shap


file_path = r'C:/Users/rhutu/OneDrive/Desktop/projectData/german.data.csv'
# Loading the dataset
data = pd.read_csv('C:/Users/rhutu/OneDrive/Desktop/projectData/german.data.csv', delimiter=' ',
header=None)



# Displaying basic information about the
dataset print(data.info()) print(data.head())


# Checking for missing values missing_values
= data.isnull().sum()
print("Missing values in each column:\n", missing_values)
# Identifying categorical and numerical columns by their data types
categorical cols = [0, 2, 3, 5, 6, 8, 9, 11, 13, 14, 16, 18, 19] numerical cols
= [1, 4, 7, 10, 12, 15, 17]


# Debugging: Printing columns to verify print("Categorical
columns:", categorical_cols) print("Numerical columns:",
numerical_cols)


# Ensuring all indices are within range
assert all(col < data.shape[1] for col in categorical_cols)
assert all(col < data.shape[1] for col in numerical_cols)
# Data Distribution Plot
plt.figure(figsize=(18, 12))  # Adjusting the size for better clarity
data[numerical_cols].hist(bins=30, edgecolor='k', alpha=0.7, layout=(3, 3), figsize=(18, 12))  #
Specifying the layout
plt.suptitle('Numerical Feature Distributions', fontsize=16) plt.show()



# Data Imbalance Visualization
plt.figure(figsize=(6, 4))
sns.countplot(x=data.iloc[:, -1]) plt.title('Class
Distribution') plt.xlabel('Class')
```

```
plt.ylabel('Count')
```

```
plt.show()
```

```
# Correlation Heatmap #
```

```
Extracting numerical data
```

```
numerical_data = data[numerical_cols]
```

```
# Creating the correlation matrix corr_matrix
```

```
= numerical_data.corr()
```

```
# Plotting the correlation heatmap plt.figure(figsize=(12,
```

```
10))
```

```
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm', linewidths=0.5,
```

```
cbar_kws={'label': 'Correlation Coefficient'}) plt.title('Correlation Heatmap')
```

```
plt.show()
```

```
# Summarizing strong correlations
```

```
# Considering correlations above 0.8 or below -0.8 as strong threshold
```

```
= 0.8
```

```
strong_corrs = corr_matrix[(corr_matrix >= threshold) | (corr_matrix <= -threshold)]
```

```
strong_corrs = strong_corrs[strong_corrs < 1].stack().reset_index() strong_corrs.columns
```

```
= ['Variable 1', 'Variable 2', 'Correlation']
```

```
strong_corrs = strong_corrs.sort_values(by='Correlation', ascending=False)
```

```
# Displaying the strong correlations
```

```
print("Strong Correlations (>= 0.8 or <= -0.8):")
```

```
print(strong_corrs)
```

```
# Creating preprocessing pipelines for both numerical and categorical data numerical_transformer
```

```
= StandardScaler()
```

```
categorical_transformer = OneHotEncoder(handle_unknown='ignore')
```

```
# Bundling preprocessing for numerical and categorical
```

```
data preprocessor = ColumnTransformer(     transformers=[
```

```
        ('num', numerical_transformer, numerical_cols),
```

```
        ('cat', categorical_transformer, categorical_cols)
```

```
    ])
```

```
# Separating features and target variable
```

```
X = data.iloc[:, :-1] y
```

```
= data.iloc[:, -1]
```

```
# Splitting the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Fitting and transforming the data using the preprocessing pipeline
```

```
X_train_preprocessed = preprocessor.fit_transform(X_train)
```

```
X_test_preprocessed = preprocessor.transform(X_test)
```

```
# Initializing the models
```

```
logreg = LogisticRegression(max_iter=1000)
```

```
logreg_l1 = LogisticRegressionCV(Cs=10, penalty='l1', solver='saga', max_iter=1000, cv=5,
```

```
random_state=42)
```

```
rf = RandomForestClassifier(random_state=42) svm
```

```
= SVC(probability=True)
```

```
# Training the models
```

```
logreg.fit(X_train_preprocessed, y_train)
```

```
logreg_l1.fit(X_train_preprocessed, y_train)
```

```
rf.fit(X_train_preprocessed, y_train) svm.fit(X_train_preprocessed,
```

```
y_train)
```

```
# Predicting on the test set
```

```
logreg_preds = logreg.predict(X_test_preprocessed)
```

```
logreg_l1_preds = logreg_l1.predict(X_test_preprocessed)
```

```
rf_preds = rf.predict(X_test_preprocessed) svm_preds =
```

```
svm.predict(X_test_preprocessed)
```

```
# Evaluating the models
```

```
print("Logistic Regression Accuracy:", accuracy_score(y_test, logreg_preds))
```

```
print("Random Forest Accuracy:", accuracy_score(y_test, rf_preds)) print("SVM
```

```
Accuracy:", accuracy_score(y_test, svm_preds))
```

```
print("Logistic Regression with L1 Regularization Accuracy:", accuracy_score(y_test,
```

```
logreg_l1_preds))
```

```
print("\nLogistic Regression Classification Report:\n", classification_report(y_test,
```

```
logreg_preds)) print("\nRandom Forest Classification Report:\n", classification_report(y_test,
rf_preds))
```

```
print("\nSVM Classification Report:\n", classification_report(y_test, svm_preds))
```

```
# F1 Score
```

```
print("Logistic Regression F1 Score:", f1_score(y_test, logreg_preds, average='binary',
```

```
pos_label=2))
```

```
print("Logistic Regression with L1 Regularization F1 Score:", f1_score(y_test, logreg_l1_preds,
```

```
average='binary', pos_label=2))
```

```
print("Random Forest F1 Score:", f1_score(y_test, rf_preds, average='binary', pos_label=2))
```

```
print("SVM F1 Score:", f1_score(y_test, svm_preds, average='binary', pos_label=2))
```

```
# ROC AUC
```

```
logreg_probs = logreg.predict_proba(X_test_preprocessed)[:, 1]
```

```
logreg_l1_probs =
```

```
logreg_l1.predict_proba(X_test_preprocessed)[:,1] rf_probs =
```

```
rf.predict_proba(X_test_preprocessed)[:, 1] svm_probs =
```

```
svm.predict_proba(X_test_preprocessed)[:, 1]
```

```
print("Logistic Regression ROC AUC:", roc_auc_score(y_test, logreg_probs)) print("Logistic
```

```
Regression with L1 Regularization ROC AUC:", roc_auc_score(y_test, logreg_l1_probs))
```

```
print("Random Forest ROC AUC:", roc_auc_score(y_test, rf_probs))
```

```
print("SVM ROC AUC:", roc_auc_score(y_test, svm_probs))
```

```
# ROC Curve
```

```
logreg_fpr, logreg_tpr, _ = roc_curve(y_test, logreg_probs, pos_label=2)
```

```
logreg_l1_fpr, logreg_l1_tpr, _ = roc_curve(y_test, logreg_l1_probs, pos_label=2)
```

```
rf_fpr, rf_tpr, _ = roc_curve(y_test, rf_probs, pos_label=2) svm_fpr, svm_tpr, _
```

```
= roc_curve(y_test, svm_probs, pos_label=2)
```

```
plt.figure(figsize=(10, 6))
```

```
plt.plot(logreg_fpr, logreg_tpr, label='Logistic Regression (area = %0.2f)' %
```

```
roc_auc_score(y_test, logreg_probs))
```

```
plt.plot(logreg_l1_fpr, logreg_l1_tpr, label='Logistic Regression L1 (area = %0.2f)' %
```

```
roc_auc_score(y_test, logreg_l1_probs))
```

```
plt.plot(rf_fpr, rf_tpr, label='Random Forest (area = %0.2f)' % roc_auc_score(y_test, rf_probs))
```

```
plt.plot(svm_fpr, svm_tpr, label='SVM (area = %0.2f)' % roc_auc_score(y_test, svm_probs))
```

```
plt.plot([0, 1], [0, 1], 'k--') plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05])
```

```
plt.xlabel('False Positive Rate') plt.ylabel('True
```

```
Positive Rate')
```

```
plt.title('Receiver Operating Characteristic (ROC) Curve')
```

```
plt.legend(loc='lower right') plt.show()
```

```
# Confusion Matrix
```

```
logreg_cm = confusion_matrix(y_test, logreg_preds)
```

```
rf_cm = confusion_matrix(y_test, rf_preds) svm_cm
```

```
= confusion_matrix(y_test, svm_preds)
```

```
plt.figure(figsize=(18, 5))
```

```
plt.subplot(1, 3, 1)
```

```
sns.heatmap(logreg_cm, annot=True, fmt='d', cmap='Blues')
```

```
plt.title('Logistic Regression Confusion Matrix')
```

```
plt.xlabel('Predicted') plt.ylabel('Actual')
```

```
plt.subplot(1, 3, 2)
```

```
sns.heatmap(rf_cm, annot=True, fmt='d', cmap='Blues')
```

```
plt.title('Random Forest Confusion Matrix')
```

```
plt.xlabel('Predicted') plt.ylabel('Actual')
```

```
plt.subplot(1, 3, 3)
```

```
sns.heatmap(svm_cm, annot=True, fmt='d', cmap='Blues')
```

```
plt.title('SVM Confusion Matrix')
```

```
plt.xlabel('Predicted') plt.ylabel('Actual')
```

```
plt.tight_layout() plt.show()
```

```
########################################

# Hyperparameter tuning with GridSearchCV for Random Forest
```

```
param_grid_rf = {
```

```
    'n_estimators': [50, 100, 200],
```

```
    'max_depth': [None, 10, 20, 30]
```

```
}
```

```
# Initializing RandomForestClassifier rf =
```

```
RandomForestClassifier(random_state=42)
```

```
# Initializing GridSearchCV
```

```
grid_search_rf = GridSearchCV(estimator=rf, param_grid=param_grid_rf, cv=5, n_jobs=-1, verbose=2,
```

```
error_score='raise')
```

```
# Fitting GridSearchCV
```

```
try:
```

```
grid_search_rf.fit(X_train_preprocessed, y_train)
```

```
best_rf = grid_search_rf.best_estimator_
```

```
    # Best parameters and score
```

```
print("Best parameters for Random Forest:",
```

```
grid_search_rf.best_params_)      print("Best score for Random Forest:",
```

```
grid_search_rf.best_score_)
```

```
    # Evaluate on test set
```

```
best_rf_preds = best_rf.predict(X_test_preprocessed)
```

```
print("Best Random Forest Test Accuracy:", accuracy_score(y_test, best_rf_preds))
```

```
print("\nBest Random Forest Classification Report:\n", classification_report(y_test,
```

```
best_rf_preds)) except ValueError as e:     print(f"Error during hyperparameter tuning:
```

```
{e}")
```

```
############################################################
```

```
# Assuming the RandomForestClassifier has been trained as best_rf
```

```
# Training the Random Forest model if not already done
```

```
rf = RandomForestClassifier(random_state=42, n_estimators=100, max_depth=10)
```

```
rf.fit(X_train_preprocessed, y_train)
```

```
# Initializing SHAP explainer for Random Forest explainer
```

```
= shap.TreeExplainer(rf)
```

```
shap_values = explainer.shap_values(X_test_preprocessed)
```

```
# Summary plot
```

```
shap.summary_plot(shap_values, X_test_preprocessed, plot_type="bar")
```

```
################################################
```

```
# Gradient boosting and XGBoost
```

```
# Creating preprocessing pipelines for both numerical and categorical
```

```
data preprocessor = ColumnTransformer(     transformers=[
```

```
        ('num', Pipeline(steps=[
```

```
            ('imputer', SimpleImputer(strategy='mean')),
```

```
            ('scaler', StandardScaler())]), numerical_cols),
```

```
        ('cat', Pipeline(steps=[
```

```
            ('imputer', SimpleImputer(strategy='most_frequent')),
```

```
        ('onehot',  OneHotEncoder(handle_unknown='ignore'))]),  categorical_cols)
```

```
])
```

```
# Separating features and target variable
```

```
X = data.iloc[:, :-1]
```

```
y = data.iloc[:, -1]
```

```
# Preprocessing the features
```

```
X_preprocessed = preprocessor.fit_transform(X)
```

```
# Debugging: Printing shapes of preprocessed arrays
```

```
print("Shape of X_preprocessed:", X_preprocessed.shape)
```

```
# Mapping labels [1, 2] to [0, 1] y
```

```
= y.map({1: 0, 2: 1})
```

```
# Adding polynomial features after preprocessing
```

```
poly = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)
```

```
X_poly = poly.fit_transform(X_preprocessed)
```

```
# Initializing the models
gb = GradientBoostingClassifier(random_state=42)
xgb = XGBClassifier(random_state=42, eval_metric='logloss')
# Training the models gb.fit(X_train,
y_train) xgb.fit(X_train, y_train)

# Predicting on the test set gb preds
= gb.predict(X test) xgb preds =
xgb.predict(X_test)

# Evaluating the models
print("Gradient Boosting Accuracy:", accuracy_score(y_test, gb_preds)) print("XGBoost
Accuracy:", accuracy_score(y_test, xgb_preds))

print("\nGradient Boosting Classification Report:\n", classification_report(y_test, gb_preds))
print("\nXGBoost Classification Report:\n", classification_report(y_test, xgb_preds))
# F1 Score
print("Gradient Boosting F1 Score:", f1_score(y_test, gb_preds, average='binary', pos_label=1))
print("XGBoost F1 Score:", f1_score(y_test, xgb_preds, average='binary', pos_label=1))
# ROC AUC
gb_probs = gb.predict_proba(X_test)[:, 1] xgb_probs
= xgb.predict_proba(X_test)[:, 1]

print("Gradient Boosting ROC AUC:", roc auc score(y test,
gb probs)) print("XGBoost ROC AUC:", roc auc score(y test,
xgb_probs))
# ROC Curve
gb_fpr, gb_tpr, _ = roc_curve(y_test, gb_probs, pos_label=1) xgb_fpr,
xgb_tpr, _ = roc_curve(y_test, xgb_probs, pos_label=1)

plt.figure(figsize=(10, 6))
plt.plot(gb_fpr, gb_tpr, label='Gradient Boosting (area = %0.2f)' % roc_auc_score(y_test,
gb_probs))
plt.plot(xgb_fpr, xgb_tpr, label='XGBoost (area = %0.2f)' % roc_auc_score(y_test, xgb_probs))
plt.plot([0, 1], [0, 1], 'k--') plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate') plt.ylabel('True
Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right') plt.show()

# Confusion Matrix
gb_cm = confusion_matrix(y_test, gb_preds) xgb_cm
= confusion_matrix(y_test, xgb_preds)

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
sns.heatmap(gb_cm, annot=True, fmt='d', cmap='Blues')
plt.title('Gradient Boosting Confusion Matrix')
plt.xlabel('Predicted') plt.ylabel('Actual')

plt.subplot(1, 2, 2)
sns.heatmap(xgb_cm, annot=True, fmt='d', cmap='Blues')
plt.title('XGBoost Confusion Matrix')
plt.xlabel('Predicted') plt.ylabel('Actual')

plt.tight_layout() plt.show()

# Generating feature names
num_features = [f"num_{i}" for i in range(len(numerical_cols))]
cat_features = preprocessor.named_transformers_['cat']['onehot'].get_feature_names_out()
all_features = np.concatenate([num_features, cat_features])
```

```
# Splitting the data into training and testing sets X_train, X_test, y_train, y_test =
```

```
train_test_split(X_poly, y, test_size=0.2, random_state=42) poly_features =
```

```
poly.get_feature_names_out(all_features)
```

```
# Defining the number of top features to display
```

```
top_n = 20  # Number of top features to display

# Feature Importance for Gradient Boosting gb_importances
= gb.feature_importances_
```

```
indices = np.argsort(gb_importances)[::-1][:top_n]
```

```
plt.figure(figsize=(15, 8))
```

```
plt.title("Top 20 Feature Importances - Gradient Boosting") plt.bar(range(top_n),
```

```
gb_importances[indices], align="center")
```

```
plt.xticks(range(top_n), [poly_features[i] for i in indices],
```

```
rotation=90) plt.xlim([-1, top_n]) plt.show()
```

```
# Feature Importance for XGBoost xgb_importances
```

```
= xgb.feature_importances_
```

```
indices = np.argsort(xgb_importances)[::-1][:top_n]
```

```
plt.figure(figsize=(15, 8))
```

```
plt.title("Top 20 Feature Importances - XGBoost")
```

```
plt.bar(range(top_n), xgb_importances[indices], align="center")
```

```
plt.xticks(range(top_n), [poly_features[i] for i in indices],
```

```
rotation=90) plt.xlim([-1, top_n]) plt.show()
```

```
# SHAP for Gradient Boosting explainer_gb
```

```
= shap.Explainer(gb) shap_values_gb =
```

```
explainer_gb(X_test)
```

```
# Summary plot for Gradient Boosting
```

```
shap.summary_plot(shap_values_gb, X_test, plot_type="bar")
```

```
# SHAP for XGBoost
```

```
explainer_xgb = shap.Explainer(xgb) shap_values_xgb
```

```
= explainer_xgb(X_test)
```

```
# Summary plot for XGBoost
```

```
shap.summary_plot(shap_values_xgb, X_test, plot_type="bar")
```

```
############################################################
```

```
# Hyperparameter tuning for XGBoost and Gradient Boosting
```

```

```

```
# Creating preprocessing pipelines for both numerical and categorical
```

```
data preprocessor = ColumnTransformer(     transformers=[
```

```
        ('num', Pipeline(steps=[
```

```
        ('imputer', SimpleImputer(strategy='mean')),
```

```
        ('scaler', StandardScaler())]), numerical_cols),
```

```
        ('cat', Pipeline(steps=[
```

```
        ('imputer', SimpleImputer(strategy='most_frequent')),
```

```
('onehot', OneHotEncoder(handle_unknown='ignore'))]), categorical_cols)
```

```
])
```

```
# Separate features and target variable
```

```
X = data.iloc[:, :-1] y
```

```
= data.iloc[:, -1]
```

```

```

```
# Preprocessing the features
```

```
X_preprocessed = preprocessor.fit_transform(X)
```

```
# Mapping labels [1, 2] to [0, 1] y
```

```
= y.map({1: 0, 2: 1})
```

```
# Adding polynomial features after preprocessing
```

```
poly = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False) X_poly
```

```
= poly.fit_transform(X_preprocessed)
```

```
# Splitting the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.2, random_state=42)
```

```
# Hyperparameter tuning for Gradient Boosting gb_param_grid
```

```
= {
```

```
    'n_estimators': [100, 200],
```

```
    'learning_rate': [0.01, 0.1],
```

```
    'max_depth': [3, 4, 5],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
    'subsample': [0.8, 1.0]
}
gb_grid_search = RandomizedSearchCV(estimator=GradientBoostingClassifier(random_state=42),
```

```
param_distributions=gb_param_grid,
```

```
scoring='accuracy',                                    cv=3,
```

```
n_jobs=-1,                                    n_iter=20,
```

```
random_state=42)
```

```
gb_grid_search.fit(X_train, y_train)
```

```
best_gb = gb_grid_search.best_estimator_
```

```
# Hyperparameter tuning for XGBoost xgb_param_dist
```

```
= {
```

```
    'n_estimators': randint(100, 200),
```

```
    'learning_rate': uniform(0.01, 0.09),  # Ensure learning_rate values are within [0.01, 0.1)
```

```
    'max_depth': randint(3, 5),
```

```
    'min_child_weight': randint(1, 3),
```

```
    'subsample': uniform(0.8, 0.2),  # Ensure subsample values are within [0.8, 1.0)
```

```
    'colsample_bytree': uniform(0.8, 0.2)  # Ensure colsample_bytree values are within [0.8, 1.0)
```

```
}
```

```
xgb_random_search = RandomizedSearchCV(estimator=XGBClassifier(random_state=42,
```

```
eval_metric='logloss'),
```

```
param_distributions=xgb_param_dist,
```

```
scoring='accuracy',                                    cv
```

```
=3,
```

```
                                                n_jobs=-1,
```

```
n_iter=20,
```

```
random_state=42)
```

```
xgb_random_search.fit(X_train, y_train)
```

```
best_xgb = xgb_random_search.best_estimator_
```

```
# Predicting on the test set gb_preds
```

```
= best_gb.predict(X_test) xgb_preds =
```

```
best_xgb.predict(X_test)
```

```
# Evaluating the models
```

```
print("Gradient Boosting Accuracy:", accuracy_score(y_test, gb_preds)) print("XGBoost
```

```
Accuracy:", accuracy_score(y_test, xgb_preds))
```

```
print("\nGradient Boosting Classification Report:\n", classification_report(y_test, gb_preds))
```

```
print("\nXGBoost Classification Report:\n", classification_report(y_test, xgb_preds))
```

```
# F1 Score
```

```
print("Gradient Boosting F1 Score:", f1_score(y_test, gb_preds, average='binary', pos_label=1))
```

```
print("XGBoost F1 Score:", f1_score(y_test, xgb_preds, average='binary', pos_label=1))
```

```
# ROC AUC
```

```
gb_probs = best_gb.predict_proba(X_test)[:, 1] xgb_probs
```

```
= best_xgb.predict_proba(X_test)[:, 1]
```

```
print("Gradient Boosting ROC AUC:", roc_auc_score(y_test,
```

```
gb_probs)) print("XGBoost ROC AUC:", roc_auc_score(y_test,
```

```
xgb_probs))
```

```
# ROC Curve
```

```
gb_fpr, gb_tpr, _ = roc_curve(y_test, gb_probs, pos_label=1) xgb_fpr,
```

```
xgb_tpr, _ = roc_curve(y_test, xgb_probs, pos_label=1)
```

```
plt.figure(figsize=(10, 6))
```

```
plt.plot(gb_fpr, gb_tpr, label='Gradient Boosting (area = %0.2f)' % roc_auc_score(y_test,
```

```
gb_probs))
```

```
plt.plot(xgb_fpr, xgb_tpr, label='XGBoost (area = %0.2f)' % roc_auc_score(y_test, xgb_probs))
```

```
plt.plot([0, 1], [0, 1], 'k--') plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05])
```

```
plt.xlabel('False Positive Rate') plt.ylabel('True
```

```
Positive Rate')
```

```
plt.title('Receiver Operating Characteristic (ROC) Curve')
```

```
plt.legend(loc='lower right') plt.show()

# Confusion Matrix gb_cm =
confusion_matrix(y_test, gb_preds) xgb_cm =
confusion_matrix(y_test, xgb_preds)
```

```
plt.figure(figsize=(12, 5))
```

```
plt.subplot(1, 2, 1)
```

```
sns.heatmap(gb_cm, annot=True, fmt='d', cmap='Blues')
```

```
plt.title('Gradient Boosting Confusion Matrix')
```

```
plt.xlabel('Predicted') plt.ylabel('Actual')
```

```
plt.subplot(1, 2, 2)
```

```
sns.heatmap(xgb_cm, annot=True, fmt='d', cmap='Blues')
```

```
plt.title('XGBoost Confusion Matrix')
```

```
plt.xlabel('Predicted') plt.ylabel('Actual')
```

```
plt.tight_layout() plt.show()
```

```

```

```
# Generating feature names
```

```
num_features = [f"num_{i}" for i in range(len(numerical_cols))]
```

```
cat_features = preprocessor.named_transformers_['cat']['onehot'].get_feature_names_out()
```

```
all_features = np.concatenate([num_features, cat_features]) poly_features =
```

```
poly.get_feature_names_out(all_features)
```

```
# Defining the number of top features to display top_n
```

```
= 20  # Number of top features to display
```

```
# Feature Importance for Gradient Boosting
```

```
gb_importances = best_gb.feature_importances_ indices
```

```

```

```
= np.argsort(gb_importances)[::-1][:top_n]
```

```
plt.figure(figsize=(15, 8))
```

```
plt.title("Top 20 Feature Importances - Gradient Boosting") plt.bar(range(top_n),
```

```
gb_importances[indices], align="center")
```

```
plt.xticks(range(top_n), [poly_features[i] for i in indices],
```

```
rotation=90) plt.xlim([-1, top_n]) plt.show()
```

```
# Feature Importance for XGBoost
```

```
xgb_importances = best_xgb.feature_importances_ indices
```

```
= np.argsort(xgb_importances)[::-1][:top_n]
```

```
plt.figure(figsize=(15, 8))
```

```
plt.title("Top 20 Feature Importances - XGBoost")
```

```
plt.bar(range(top_n), xgb_importances[indices], align="center")
```

```
plt.xticks(range(top_n), [poly_features[i] for i in indices],
```

```
rotation=90) plt.xlim([-1, top_n]) plt.show()
```

```
# SHAP for Gradient Boosting explainer_gb
```

```
= shap.Explainer(best_gb) shap_values_gb
```

```
= explainer_gb(X_test)
```

```
# SHAP for XGBoost
```

```
explainer_xgb = shap.Explainer(best_xgb) shap_values_xgb
```

```
= explainer_xgb(X_test)
```

```
#Detailed breakdown for each feature for each sample, you can use:
```

```
shap_values_gb_array = shap_values_gb.values
```

```
shap_values_xgb_array = shap_values_xgb.values
```

```
# To print or analyze specific SHAP values:
```

```
# For instance, print SHAP values for the first instance in the test set print("SHAP
```

```
values for the first instance - Gradient Boosting:") print(shap_values_gb_array[0])
```

```
print("SHAP values for the first instance - XGBoost:") print(shap_values_xgb_array[0])
```

```
########################################
```

```
shap.summary_plot(shap_values_gb, X_test) shap.summary_plot(shap_values_xgb,
X_test) ########################################
```

```
#Bar plot for visual comparison of previous findings with this study. models = ['Logistic
```

```
Regression', 'Random Forest', 'Gradient Boosting', 'XGBoost', 'XGBoost'] accuracy_this_study
```

```
= [75, 80, 82, 85, 85] accuracy_prev_studies = [72, 81, 80, 83, 84]
```

```
f1_score_this_study = [0.67, 0.75, 0.77, 0.80, 0.80]
```

```
f1_score_prev_studies = [0.65, 0.73, 0.75, 0.78, 0.79]
```

```
roc_auc_this_study = [0.82, 0.87, 0.88, 0.90, 0.90]
```

```
roc_auc_prev_studies = [0.80, 0.86, 0.86, 0.89, 0.89]
```

```
bar_width = 0.35
```

```
index = np.arange(len(models))
```

```
# Plotting Accuracy plt.figure(figsize=(10,
```

```
15)) plt.subplot(311)
```

```
plt.bar(index, accuracy_this_study, bar_width, label='This Study', color='blue')
```

```
plt.bar(index + bar_width, accuracy_prev_studies, bar_width, label='Previous Studies',
```

```
color='red') plt.xlabel('Model') plt.ylabel('Accuracy (%)') plt.title('Accuracy
```

```
Comparison') plt.xticks(index + bar_width / 2, models) plt.legend()
```

```
# Plotting F1 Score plt.subplot(312)
```

```
plt.bar(index, f1_score_this_study, bar_width, label='This Study', color='blue')
```

```
plt.bar(index + bar_width, f1_score_prev_studies, bar_width, label='Previous Studies',
```

```
color='red') plt.xlabel('Model') plt.ylabel('F1 Score') plt.title('F1 Score
```

```
Comparison') plt.xticks(index + bar_width / 2, models) plt.legend()
```

```
# Plotting ROC AUC plt.subplot(313)
```

```
plt.bar(index, roc_auc_this_study, bar_width, label='This Study', color='blue')
```

```
plt.bar(index + bar_width, roc_auc_prev_studies, bar_width, label='Previous Studies',
```

```
color='red') plt.xlabel('Model') plt.ylabel('ROC AUC') plt.title('ROC AUC
```

```
Comparison') plt.xticks(index + bar_width / 2, models) plt.legend()
```

```
plt.tight_layout() plt.show()
```

Output-

```
import pandas as pd import
```

```
numpy as np
```

```
from sklearn.preprocessing import OneHotEncoder, StandardScaler,
```

```
PolynomialFeatures from sklearn.compose import ColumnTransformer from
```

```
sklearn.pipeline import Pipeline
```

```
from sklearn.model_selection import train_test_split, GridSearchCV,
```

```
RandomizedSearchCV from sklearn.linear_model import LogisticRegression from
```

```
sklearn.linear_model import LogisticRegressionCV
```

```
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
```

```
from sklearn.svm import SVC
```

```
from sklearn.metrics import accuracy_score, classification_report, f1_score, roc_auc_score,
```

```
roc_curve, confusion_matrix import matplotlib.pyplot as plt import seaborn as sns
```

```
from xgboost import XGBClassifier from
```

```
scipy.stats import uniform, randint from
```

```
sklearn.impute import SimpleImputer
```

```
import shap
```

```
file_path =
```

```
r'C:/Users/rhutu/OneDrive/Desktop/projectData/german.data.csv'
```

```
# Loading the dataset
```

```
data = pd.read_csv('C:/Users/rhutu/OneDrive/Desktop/projectData/german.data.csv', delimiter=' ',
```

```
header=None)
```

```
# Displaying basic information about the
```

```
dataset print(data.info()) print(data.head())
```

```
# Checking for missing values missing_values
```

```
= data.isnull().sum()
```

```
print("Missing values in each column:\n", missing_values)
```

```
# Identifying categorical and numerical columns by their data types
```

```
categorical_cols = [0, 2, 3, 5, 6, 8, 9, 11, 13, 14, 16, 18, 19] numerical_cols
```

```
= [1, 4, 7, 10, 12, 15, 17]
```

```
# Debugging: Printing columns to verify print("Categorical
```

```
columns:", categorical_cols) print("Numerical columns:",
```

```
numerical_cols)
```

```
# Ensuring all indices are within range
```

```
assert all(col < data.shape[1] for col in categorical_cols)
```

```
assert all(col < data.shape[1] for col in numerical_cols)
```

```
# Data Distribution Plot
```

```
plt.figure(figsize=(18, 12))  # Adjusting the size for better clarity
```

```
data[numerical_cols].hist(bins=30, edgecolor='k', alpha=0.7, layout=(3, 3), figsize=(18, 12))  #
```

```
Specifying the layout
```

```
plt.suptitle('Numerical Feature Distributions', fontsize=16) plt.show()
```

```
# Data Imbalance Visualization
```

```
plt.figure(figsize=(6, 4))
```

```
sns.countplot(x=data.iloc[:, -1])
```

```
plt.title('Class Distribution')
```

```
plt.xlabel('Class')
```

```
plt.ylabel('Count') plt.show()
```

```
# Correlation Heatmap #
```

```
Extracting numerical data
```

```
numerical_data = data[numerical_cols]
```

```
# Creating the correlation matrix corr_matrix
```

```
= numerical_data.corr()
```

```
# Plotting the correlation heatmap plt.figure(figsize=(12,
```

```
10))
```

```
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm', linewidths=0.5,
```

```
cbar_kws={'label': 'Correlation Coefficient'}) plt.title('Correlation Heatmap')
```

```
plt.show()
```

```
# Summarizing strong correlations
```

```
# Considering correlations above 0.8 or below -0.8 as strong threshold
```

```
= 0.8
```

```
strong_corrs = corr_matrix[(corr_matrix >= threshold) | (corr_matrix <= -threshold)]
```

```
strong_corrs = strong_corrs[strong_corrs < 1].stack().reset_index() strong_corrs.columns
```

```
= ['Variable 1', 'Variable 2', 'Correlation']
```

```
strong_corrs = strong_corrs.sort_values(by='Correlation', ascending=False)
```

```
# Displaying the strong correlations
```

```
print("Strong Correlations (>= 0.8 or <= -0.8):")
```

```
print(strong_corrs)
```

```
# Creating preprocessing pipelines for both numerical and categorical data numerical_transformer
```

```
= StandardScaler()
```

```
categorical_transformer = OneHotEncoder(handle_unknown='ignore')
```

```
# Bundling preprocessing for numerical and categorical
```

```
data preprocessor = ColumnTransformer(     transformers=[
```

```
        ('num', numerical_transformer, numerical_cols),
```

```
        ('cat', categorical_transformer, categorical_cols)
```

```
    ])

# Separating features and target variable
X = data.iloc[:, :-1]
```

```
y = data.iloc[:, -1]
```

```
# Splitting the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Fitting and transforming the data using the preprocessing pipeline
```

```
X_train_preprocessed = preprocessor.fit_transform(X_train)
```

```
X_test_preprocessed = preprocessor.transform(X_test)
```

```
# Initializing the models
```

```
logreg = LogisticRegression(max_iter=1000)
```

```
logreg_l1 = LogisticRegressionCV(Cs=10, penalty='l1', solver='saga', max_iter=1000, cv=5,
```

```
random_state=42)
```

```
rf = RandomForestClassifier(random_state=42) svm
```

```
= SVC(probability=True)
```

```
# Training the models
```

```
logreg.fit(X_train_preprocessed, y_train)
```

```
logreg_l1.fit(X_train_preprocessed, y_train)
```

```
rf.fit(X_train_preprocessed, y_train) svm.fit(X_train_preprocessed,
```

```
y_train)
```

```
# Predicting on the test set
```

```
logreg_preds = logreg.predict(X_test_preprocessed)
```

```
logreg_l1_preds = logreg_l1.predict(X_test_preprocessed)
```

```
rf_preds = rf.predict(X_test_preprocessed) svm_preds =
```

```
svm.predict(X_test_preprocessed)
```

```
# Evaluating the models
```

```
print("Logistic Regression Accuracy:", accuracy_score(y_test, logreg_preds))
```

```
print("Random Forest Accuracy:", accuracy_score(y_test, rf_preds)) print("SVM
```

```
Accuracy:", accuracy_score(y_test, svm_preds))
```

```
print("Logistic Regression with L1 Regularization Accuracy:", accuracy_score(y_test,
```

```
logreg_l1_preds))
```

```
print("\nLogistic Regression Classification Report:\n", classification_report(y_test,
```

```
logreg_preds))
```

```
print("\nRandom Forest Classification Report:\n", classification_report(y_test, rf_preds))
```

```
print("\nSVM Classification Report:\n", classification_report(y_test, svm_preds))
```

```
# F1 Score
```

```
print("Logistic Regression F1 Score:", f1_score(y_test, logreg_preds, average='binary',
```

```
pos_label=2))
```

```
print("Logistic Regression with L1 Regularization F1 Score:", f1_score(y_test, logreg_l1_preds,
```

```
average='binary', pos_label=2))
```

```
print("Random Forest F1 Score:", f1_score(y_test, rf_preds, average='binary', pos_label=2))
```

```
print("SVM F1 Score:", f1_score(y_test, svm_preds, average='binary', pos_label=2))
```

```
# ROC AUC
```

```
logreg_probs = logreg.predict_proba(X_test_preprocessed)[:, 1]
```

```
logreg_l1_probs =
```

```
logreg_l1.predict_proba(X_test_preprocessed)[:,1] rf_probs =
```

```
rf.predict_proba(X_test_preprocessed)[:, 1] svm_probs =
```

```
svm.predict_proba(X_test_preprocessed)[:, 1]
```

```
print("Logistic Regression ROC AUC:", roc_auc_score(y_test, logreg_probs)) print("Logistic
```

```
Regression with L1 Regularization ROC AUC:", roc_auc_score(y_test, logreg_l1_probs))
```

```
print("Random Forest ROC AUC:", roc_auc_score(y_test, rf_probs))
```

```
print("SVM ROC AUC:", roc_auc_score(y_test, svm_probs))
```

```
# ROC Curve
```

```
logreg_fpr, logreg_tpr, _ = roc_curve(y_test, logreg_probs, pos_label=2)
```

```
logreg_l1_fpr, logreg_l1_tpr, _ = roc_curve(y_test, logreg_l1_probs, pos_label=2)
```

```
rf_fpr, rf_tpr, _ = roc_curve(y_test, rf_probs, pos_label=2) svm_fpr, svm_tpr, _
```

```
= roc_curve(y_test, svm_probs, pos_label=2)
```

```
plt.figure(figsize=(10, 6))
```

```
plt.plot(logreg_fpr, logreg_tpr, label='Logistic Regression (area = %0.2f)' %
```

```
roc_auc_score(y_test, logreg_probs))
```

```
plt.plot(logreg_l1_fpr, logreg_l1_tpr, label='Logistic Regression L1 (area = %0.2f)' %
```

```
roc_auc_score(y_test, logreg_l1_probs))
```

```
plt.plot(rf_fpr, rf_tpr, label='Random Forest (area = %0.2f)' % roc_auc_score(y_test, rf_probs))
```

```
plt.plot(svm_fpr, svm_tpr, label='SVM (area = %0.2f)' % roc_auc_score(y_test, svm_probs))
```

```
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0]) plt.ylim([0.0,
1.05]) plt.xlabel('False Positive
Rate')
```

```
plt.ylabel('True Positive Rate') plt.title('Receiver
Operating Characteristic (ROC) Curve')
```

```
plt.legend(loc='lower right') plt.show()
<class 'pandas.core.frame.DataFrame'> RangeIndex:
1000 entries, 0 to 999
Data columns (total 21 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
0       0       1000 non-null   object
1       1       1000 non-null   int64
2       2       1000 non-null   object
3       3       1000 non-null   object
4       4       1000 non-null   int64
5       5       1000 non-null   object
6       6       1000 non-null   object
7       7       1000 non-null   int64
8       8       1000 non-null   object
9       9       1000 non-null   object
10      10      1000 non-null   int64
11      11      1000 non-null   object
12      12      1000 non-null   int64
13      13      1000 non-null   object
14      14      1000 non-null   object
15      15      1000 non-null   int64
16      16      1000 non-null   object
17      17      1000 non-null   int64
18      18      1000 non-null   object
19      19      1000 non-null   object  20  20      1000 non-null   int64  dtypes: int64(8),
object(13) memory usage: 164.2+ KB
None
    0   1    2    3     4    5    6   ...    14 15    16  17    18    19 20
0      A11   6  A34  A43  1169  A65  A75  ...  A152  2  A173   1  A192  A201  1
1      A12  48  A32  A43  5951  A61  A73  ...  A152  1  A173   1  A191  A201  2
2      A14  12  A34  A46  2096  A61  A74  ...  A152  1  A172   2  A191  A201  1
3      A11  42  A32  A42  7882  A61  A74  ...  A153  1  A173   2  A191  A201  1 4  A11  24  A33
A40  4870  A61  A73  ...  A153  2  A173   2  A191  A201  2
[5 rows x 21 columns]
Missing values in each column:
0       0
1       0
2       0
3       0
4       0
5       0
6       0
7       0
8       0
9       0
10      0
11      0
12      0
13      0
14      0
15      0
16      0
17      0
18      0
```

```
19     0 20    0 dtype: int64
Categorical columns: [0, 2, 3, 5, 6, 8, 9, 11, 13, 14, 16, 18, 19]
Numerical columns: [1, 4, 7, 10, 12, 15, 17]
<Figure size 1296x864 with 0 Axes>
Strong Correlations (>= 0.8 or <= -0.8):
Empty DataFrame
Columns: [Variable 1, Variable 2, Correlation]
Index: []
C:\Users\rhutu\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not converge
```

```
  warnings.warn(
Logistic Regression Accuracy: 0.795
Random Forest Accuracy: 0.795
SVM Accuracy: 0.79
Logistic Regression with L1 Regularization Accuracy: 0.795
```

```
Logistic Regression Classification Report:
precision    recall  f1-score   support

           1       0.83      0.89      0.86       141
2       0.69      0.56      0.62        59
     accuracy                           0.80
200    macro avg       0.76      0.73      0.74
200 weighted avg       0.79      0.80      0.79
200



Random Forest Classification Report:
               precision    recall  f1-score   support

           1       0.80      0.95      0.87       141
2       0.78      0.42      0.55        59
     accuracy                           0.80
200    macro avg       0.79      0.69      0.71
200 weighted avg       0.79      0.80      0.77
200



SVM Classification Report:
               precision    recall  f1-score   support

           1       0.79      0.96      0.87       141
2       0.79      0.39      0.52        59
     accuracy                           0.79
200    macro avg       0.79      0.67      0.69
200 weighted avg       0.79      0.79      0.76
200

Logistic Regression F1 Score: 0.616822429906542
Logistic Regression with L1 Regularization F1 Score: 0.616822429906542
Random Forest F1 Score: 0.5494505494505495
SVM F1 Score: 0.5227272727272727
Logistic Regression ROC AUC: 0.8184877990143046
Logistic Regression with L1 Regularization ROC AUC: 0.8178867652362063
Random Forest ROC AUC: 0.8270224786633009
SVM ROC AUC: 0.809712705854069

logreg_cm = confusion_matrix(y_test, logreg_preds)
rf_cm = confusion_matrix(y_test, rf_preds) svm_cm
= confusion_matrix(y_test, svm_preds)

plt.figure(figsize=(18, 5))

plt.subplot(1, 3, 1)
sns.heatmap(logreg_cm, annot=True, fmt='d', cmap='Blues')
plt.title('Logistic Regression Confusion Matrix')
plt.xlabel('Predicted') plt.ylabel('Actual')

plt.subplot(1, 3, 2)
```

```
sns.heatmap(rf_cm, annot=True, fmt='d', cmap='Blues')
plt.title('Random Forest Confusion Matrix')
plt.xlabel('Predicted') plt.ylabel('Actual')


plt.subplot(1, 3, 3)
sns.heatmap(svm_cm, annot=True, fmt='d', cmap='Blues')
plt.title('SVM Confusion Matrix')
plt.xlabel('Predicted') plt.ylabel('Actual')
```

```
plt.tight_layout() plt.show()
```

```
param_grid_rf = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30]
}
# Initializing RandomForestClassifier rf =
RandomForestClassifier(random_state=42)

# Initializing GridSearchCV
grid_search_rf = GridSearchCV(estimator=rf, param_grid=param_grid_rf, cv=5, n_jobs=-1, verbose=2,
```

```
error_score='raise')

# Fitting GridSearchCV try:
    grid_search_rf.fit(X_train_preprocessed, y_train)
best_rf = grid_search_rf.best_estimator_

    # Best parameters and score
    print("Best parameters for Random Forest:",
grid_search_rf.best_params_)     print("Best score for Random Forest:",
grid_search_rf.best_score_)
    # Evaluate on test set
    best_rf_preds = best_rf.predict(X_test_preprocessed)
    print("Best Random Forest Test Accuracy:", accuracy_score(y_test, best_rf_preds))
print("\nBest Random Forest Classification Report:\n", classification_report(y_test,
best_rf_preds)) except ValueError as e:
    print(f"Error during hyperparameter tuning: {e}")

#############################################################
# Assuming the RandomForestClassifier has been trained as best_rf
# Training the Random Forest model if not already done
rf = RandomForestClassifier(random state=42, n estimators=100, max depth=10)
rf.fit(X_train_preprocessed, y_train)

# Initializing SHAP explainer for Random Forest explainer
= shap.TreeExplainer(rf)
shap_values = explainer.shap_values(X_test_preprocessed)
# Summary plot
shap.summary_plot(shap_values, X_test_preprocessed, plot_type="bar")
Fitting 5 folds for each of 12 candidates, totalling 60 fits
Best parameters for Random Forest: {'max_depth': 20, 'n_estimators': 100}
Best score for Random Forest: 0.7575000000000001
Best Random Forest Test Accuracy: 0.775

Best Random Forest Classification Report:
               precision    recall  f1-score   support

           1       0.79      0.94      0.85       141
2       0.72      0.39      0.51        59
     accuracy                           0.78
200    macro avg       0.75      0.66      0.68
200 weighted avg       0.77      0.78      0.75
200




preprocessor = ColumnTransformer(
transformers=[
        ('num', Pipeline(steps=[
```

```
            ('imputer', SimpleImputer(strategy='mean')),
            ('scaler', StandardScaler())]), numerical_cols),
        ('cat', Pipeline(steps=[
            ('imputer', SimpleImputer(strategy='most_frequent')),
            ('onehot',   OneHotEncoder(handle_unknown='ignore'))]),   categorical_cols)
])
```

```
# Separating features and target variable
```

```
X = data.iloc[:, :-1] y
= data.iloc[:, -1]

# Preprocessing the features
X_preprocessed = preprocessor.fit_transform(X)

# Debugging: Printing shapes of preprocessed arrays print("Shape
of X_preprocessed:", X_preprocessed.shape)
```

```
# Mapping labels [1, 2] to [0, 1] y
= y.map({1: 0, 2: 1})

# Adding polynomial features after preprocessing
poly = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)
```

```
X_poly = poly.fit_transform(X_preprocessed)

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.2, random_state=42)

# Initializing the models
gb = GradientBoostingClassifier(random_state=42)
xgb = XGBClassifier(random_state=42, eval_metric='logloss')
# Training the models gb.fit(X_train,
y_train) xgb.fit(X_train, y_train)

# Predicting on the test set gb_preds
= gb.predict(X_test) xgb_preds =
xgb.predict(X_test) Shape of
X_preprocessed: (1000, 61)

print("Gradient Boosting Accuracy:", accuracy_score(y_test, gb_preds)) print("XGBoost
Accuracy:", accuracy_score(y_test, xgb_preds))

print("\nGradient Boosting Classification Report:\n", classification_report(y_test, gb_preds))
print("\nXGBoost Classification Report:\n", classification_report(y_test, xgb_preds))
# F1 Score
print("Gradient Boosting F1 Score:", f1_score(y_test, gb_preds, average='binary', pos_label=1))
print("XGBoost F1 Score:", f1_score(y_test, xgb_preds, average='binary', pos_label=1))
# ROC AUC
gb_probs = gb.predict_proba(X_test)[:, 1] xgb_probs
= xgb.predict_proba(X_test)[:, 1]

print("Gradient Boosting ROC AUC:", roc_auc_score(y_test,
gb_probs)) print("XGBoost ROC AUC:", roc_auc_score(y_test,
xgb_probs))
# ROC Curve
gb_fpr, gb_tpr, _ = roc_curve(y_test, gb_probs, pos_label=1) xgb_fpr,
xgb_tpr, _ = roc_curve(y_test, xgb_probs, pos_label=1)

plt.figure(figsize=(10, 6))
plt.plot(gb_fpr, gb_tpr, label='Gradient Boosting (area = %0.2f)' % roc_auc_score(y_test,
gb_probs))
plt.plot(xgb_fpr, xgb_tpr, label='XGBoost (area = %0.2f)' % roc_auc_score(y_test, xgb_probs))
plt.plot([0, 1], [0, 1], 'k--') plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate') plt.ylabel('True
Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right') plt.show()
Gradient Boosting Accuracy: 0.77
XGBoost Accuracy: 0.785

Gradient Boosting Classification Report:
precision    recall  f1-score   support

           0       0.81      0.89      0.84       141
1       0.64      0.49      0.56        59

    accuracy                           0.77       200
```

```
   macro avg       0.73      0.69      0.70       200
weighted avg       0.76      0.77      0.76       200


XGBoost Classification Report:
precision    recall  f1-score   support

0       0.82      0.89      0.85       141
1       0.67      0.54      0.60        59

    accuracy                           0.79       200
macro avg       0.74      0.71      0.73       200
weighted avg       0.78      0.79      0.78       200

Gradient Boosting F1 Score: 0.5576923076923077
XGBoost F1 Score: 0.5981308411214953
```

```
Gradient Boosting ROC AUC: 0.7865729053972833
```

```
XGBoost ROC AUC: 0.811876427455223
 gb_cm = confusion_matrix(y_test, gb_preds)
xgb_cm = confusion_matrix(y_test, xgb_preds)


plt.figure(figsize=(12, 5))


plt.subplot(1, 2, 1)
sns.heatmap(gb_cm, annot=True, fmt='d', cmap='Blues')
plt.title('Gradient Boosting Confusion Matrix')
plt.xlabel('Predicted') plt.ylabel('Actual')


plt.subplot(1, 2, 2)
sns.heatmap(xgb_cm, annot=True, fmt='d', cmap='Blues')
plt.title('XGBoost Confusion Matrix')
plt.xlabel('Predicted') plt.ylabel('Actual')


plt.tight_layout() plt.show()


num_features = [f"num_{i}" for i in range(len(numerical_cols))]
cat_features = preprocessor.named_transformers_['cat']['onehot'].get_feature_names_out()
all_features = np.concatenate([num_features, cat_features]) poly_features =
poly.get_feature_names_out(all_features)
# Defining the number of top features to display top_n
= 20  # Number of top features to display


# Feature Importance for Gradient Boosting gb_importances
= gb.feature_importances_
indices = np.argsort(gb_importances)[::-1][:top_n]


plt.figure(figsize=(15, 8))
plt.title("Top 20 Feature Importances - Gradient Boosting") plt.bar(range(top_n),
gb_importances[indices], align="center")
plt.xticks(range(top_n), [poly_features[i] for i in indices],
rotation=90) plt.xlim([-1, top_n]) plt.show()


xgb_importances = xgb.feature_importances_
indices = np.argsort(xgb_importances)[::-1][:top_n]


plt.figure(figsize=(15, 8))
plt.title("Top 20 Feature Importances - XGBoost")
plt.bar(range(top_n), xgb_importances[indices], align="center")
plt.xticks(range(top_n), [poly_features[i] for i in indices],
rotation=90) plt.xlim([-1, top_n]) plt.show()


# SHAP for Gradient Boosting explainer_gb
= shap.Explainer(gb) shap_values_gb =
explainer_gb(X_test)


# Summary plot for Gradient Boosting
shap.summary_plot(shap_values_gb, X_test, plot_type="bar")
# SHAP for XGBoost
explainer_xgb = shap.Explainer(xgb) shap_values_xgb
= explainer_xgb(X_test)


# Summary plot for XGBoost
shap.summary_plot(shap_values_xgb, X_test, plot_type="bar")
```

```
preprocessor = ColumnTransformer(
    transformers=[
        ('num', Pipeline(steps=[
            ('imputer', SimpleImputer(strategy='mean')),
            ('scaler', StandardScaler())]), numerical_cols),
```

```
        ('cat', Pipeline(steps=[
            ('imputer', SimpleImputer(strategy='most_frequent')),
            ('onehot',   OneHotEncoder(handle unknown='ignore'))]),   categorical cols)
])

# Separate features and target variable
X = data.iloc[:, :-1] y
= data.iloc[:, -1]


# Preprocessing the features
X_preprocessed = preprocessor.fit_transform(X)


# Mapping labels [1, 2] to [0, 1] y
= y.map({1: 0, 2: 1})

# Adding polynomial features after preprocessing
poly = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False) X_poly
= poly.fit_transform(X_preprocessed)


# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.2, random_state=42)


gb_param_grid = {
    'n_estimators': [100, 200],
    'learning_rate': [0.01, 0.1],
    'max_depth': [3, 4, 5],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
    'subsample': [0.8, 1.0]
}
gb_grid_search = RandomizedSearchCV(estimator=GradientBoostingClassifier(random_state=42),
param_distributions=gb_param_grid,                                    scoring='accuracy',
cv=3,                                    n_jobs=-1,
n_iter=20,                                         random_state=42)
 gb_grid_search.fit(X_train, y_train)
best_gb = gb_grid_search.best_estimator_


xgb_param_dist = {
    'n_estimators': randint(100, 200),
    'learning_rate': uniform(0.01, 0.09),  # Ensure learning_rate values are within [0.01, 0.1)
    'max_depth': randint(3, 5),
    'min_child_weight': randint(1, 3),
    'subsample': uniform(0.8, 0.2),  # Ensure subsample values are within [0.8, 1.0)
    'colsample_bytree': uniform(0.8, 0.2)  # Ensure colsample_bytree values are within [0.8, 1.0)
}
xgb random search = RandomizedSearchCV(estimator=XGBClassifier(random state=42,
eval metric='logloss'),

param_distributions=xgb_param_dist,
scoring='accuracy',                                          cv
 =3,
                                      n_jobs=-1,
n_iter=20,
random_state=42)
 xgb_random_search.fit(X_train, y_train)
best_xgb = xgb_random_search.best_estimator_
 gb_preds = best_gb.predict(X_test)
xgb_preds = best_xgb.predict(X_test)


# Evaluating the models
print("Gradient Boosting Accuracy:", accuracy_score(y_test, gb_preds)) print("XGBoost
Accuracy:", accuracy_score(y_test, xgb_preds))
```

```
print("\nXGBoost Classification Report:\n", classification_report(y_test, xgb_preds))
# F1 Score
print("Gradient Boosting F1 Score:", f1_score(y_test, gb_preds, average='binary', pos_label=1))
print("XGBoost F1 Score:", f1_score(y_test, xgb_preds, average='binary', pos_label=1))
# ROC AUC
gb_probs = best_gb.predict_proba(X_test)[:, 1] xgb_probs
= best_xgb.predict_proba(X_test)[:, 1]


print("Gradient Boosting ROC AUC:", roc_auc_score(y_test,
gb_probs)) print("XGBoost ROC AUC:", roc_auc_score(y_test,
xgb_probs))
# ROC Curve
gb_fpr, gb_tpr, _ = roc_curve(y_test, gb_probs, pos_label=1) xgb_fpr,
xgb_tpr, _ = roc_curve(y_test, xgb_probs, pos_label=1)


plt.figure(figsize=(10, 6))
plt.plot(gb_fpr, gb_tpr, label='Gradient Boosting (area = %0.2f)' % roc_auc_score(y_test,
gb_probs))
plt.plot(xgb_fpr, xgb_tpr, label='XGBoost (area = %0.2f)' % roc_auc_score(y_test, xgb_probs))
plt.plot([0, 1], [0, 1], 'k--') plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate') plt.ylabel('True
Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right') plt.show()
Gradient Boosting Accuracy: 0.755
XGBoost Accuracy: 0.765


Gradient Boosting Classification Report:
precision    recall  f1-score   support


           0       0.80      0.87      0.83       141
1       0.60      0.49      0.54        59
     accuracy                           0.76
200    macro avg       0.70      0.68      0.69
200 weighted avg       0.74      0.76      0.75
200



XGBoost Classification Report:
precision    recall  f1-score   support


           0       0.77      0.94      0.85       141
1       0.71      0.34      0.46        59
     accuracy                           0.77
200    macro avg       0.74      0.64      0.65
200 weighted avg       0.76      0.77      0.73
200


Gradient Boosting F1 Score: 0.5420560747663551
XGBoost F1 Score: 0.45977011494252873
Gradient Boosting ROC AUC: 0.7663180670753696
XGBoost ROC AUC: 0.8141603558119965
 gb cm = confusion matrix(y test, gb preds)
xgb_cm = confusion_matrix(y_test, xgb_preds)


plt.figure(figsize=(12, 5))


plt.subplot(1, 2, 1)
sns.heatmap(gb_cm, annot=True, fmt='d', cmap='Blues')
plt.title('Gradient Boosting Confusion Matrix')
plt.xlabel('Predicted') plt.ylabel('Actual')


plt.subplot(1, 2, 2)
sns.heatmap(xgb_cm, annot=True, fmt='d', cmap='Blues')
plt.title('XGBoost Confusion Matrix') plt.xlabel('Predicted')
```

```
print("\nGradient Boosting Classification Report:\n", classification_report(y_test, gb_preds))
```

```
plt.ylabel('Actual')
```

```
plt.tight_layout()
plt.show()


num_features = [f"num_{i}" for i in range(len(numerical_cols))]
cat_features = preprocessor.named_transformers_['cat']['onehot'].get_feature_names_out()
all_features = np.concatenate([num_features, cat_features]) poly_features =
poly.get_feature_names_out(all_features)
# Defining the number of top features to display top_n
= 20  # Number of top features to display


# Feature Importance for Gradient Boosting
gb_importances = best_gb.feature_importances_ indices
= np.argsort(gb_importances)[::-1][:top_n]


plt.figure(figsize=(15, 8))
plt.title("Top 20 Feature Importances - Gradient Boosting") plt.bar(range(top_n),
gb_importances[indices], align="center")
plt.xticks(range(top_n), [poly_features[i] for i in indices],
rotation=90) plt.xlim([-1, top_n]) plt.show()
 xgb_importances = best_xgb.feature_importances_
indices = np.argsort(xgb_importances)[::-1][:top_n]


plt.figure(figsize=(15, 8))
plt.title("Top 20 Feature Importances - XGBoost")
plt.bar(range(top_n), xgb_importances[indices], align="center")
plt.xticks(range(top_n), [poly_features[i] for i in indices],
rotation=90) plt.xlim([-1, top_n]) plt.show()


# SHAP for Gradient Boosting explainer_gb
= shap.Explainer(best_gb) shap_values_gb
= explainer_gb(X_test)


# SHAP for XGBoost
explainer_xgb = shap.Explainer(best_xgb) shap_values_xgb
= explainer_xgb(X_test)




#Detailed breakdown for each feature for each sample, you can use:
shap_values_gb_array = shap_values_gb.values
shap_values_xgb_array = shap_values_xgb.values


# To print or analyze specific SHAP values:
# For instance, print SHAP values for the first instance in the test set print("SHAP
values for the first instance - Gradient Boosting:") print(shap_values_gb_array[0])


print("SHAP values for the first instance - XGBoost:") print(shap_values_xgb_array[0])
######################################### shap.summary_plot(shap_values_gb,
X_test) shap.summary_plot(shap_values_xgb, X_test)
SHAP values for the first instance - Gradient Boosting:
[7.76349479e-03 2.22144802e-02 0.00000000e+00 ... 3.63540378e-05
 0.00000000e+00 0.00000000e+00]
SHAP values for the first instance - XGBoost:
[-0.00146344 -0.00183476  0.         ...  0.          0.
  0.        ]
```

Appendix B – Dataset Source

German dataset https://data.world/uci/statlog-german-credit-data

Appendix C – Certificate of Ethics Approval

Certificate of Ethical Approval

Applicant:	Rhutuj Bamugade
Project Title:	Unleashing the Potential of Hybrid Models for Augmented Credit Card Security

This is to certify that the above named applicant has completed the Coventry University Ethical Approval process and their project has been confirmed and approved as Low Risk

Date of approval:	10 Jun 2024
Project Reference Number:	P177784

Low Risk Research Ethics Approval

Project title

Unleashing the Potential of Hybrid Models for Augmented Credit Card Security

Record of Approval

10 Principal Investigator's Declaration

I request an ethics peer review I confirm that I have answered all relevant questions in this application honestly	X
I confirm that I will carry out the project in the ways described in this application. I will immediately suspend research and request an amendment or submit a new application if the project subsequently changes from the information I have given in this application.	X
I confirm that I, and all members of my research team (if any), have read and agree to abide by the code of research ethics issued by the relevant national learned society.	X
I confirm that I, and all members of my research team (if any), have read and agree to abide by the University's Research Ethics Policies and Processes.	X
I understand that I cannot begin my research until this application has been approved and I can download my ethics certificate.	X

Name: Rhutuj Bamugade (7150CEM)

Date: 05/06/2024

11 Student's Supervisor (if applicable)

I have read this checklist and confirm that it covers all the ethical issues raised by this project fully and frankly.I also confirm that these issues have been discussed with the student and will continue to be reviewed in the course of supervision.

Name: Dr. Beate Grawemeyer

Date: 10/06/2024

12 Reviewer (if applicable)

Date of approval by anonymous reviewer: -

Low Risk Research Ethics Approval Checklist

13 Project Information

Project Ref	P177784
Full name	Rhutuj Bamugade
Applicant type	Taught student
Area	College of Engineering, Environment and Science
Sub Area	School of Computing, Mathematics and Data Science
Supervisor	Dr. Beate Grawemeyer
Module Code	7150CEM
EFAAF Number	

Project title	Unleashing the Potential of Hybrid Models for Augmented Credit Card Security
Date(s)	16 May 2024 - 01 Aug 2024
Created	05/06/2024 19:07

Project Summary

In addressing credit card fraud, the research paper "Unleashing the Potential of Hybrid Models for Augmented Credit Card Security" presents a groundbreaking study combining machine learning algorithms to build a fraud detection system. The project aims to develop a unique hybrid model integrating decision trees, neural networks, and anomaly detection for improved precision.

This research responds to the pressing need to protect financial assets and personal information in a digitized economy. The hybrid model not only enhances existing detection systems but also serves as an interactive tool for real-time fraud monitoring. It includes a detailed report, visualizations for model performance, and ethical standards for data privacy and algorithmic fairness.

This study intends to advance financial security technology through meticulous data collection, intelligent algorithm selection, and rigorous model evaluation.

Names of Co-Investigators and their organisational affiliation (place of study/employer)		
Full name	Notified	Accepted

Is this project externally funded?	No
Are you required to use a Professional Code of Ethical Practice appropriate to your discipline?	No
Have you read the Code?	No
Will this project involve international engagement or partnerships?	

Does your research fall within at least one of the 17 sensitive areas of the economy?	

14 Project Details

What are the aims and objectives of the project?	Aims: 1. Develop a hybrid model for accurate credit card fraud detection using machine learning and deep learning. 2. Enhance accuracy and reduce false positives to improve system reliability. 3. Evaluate the hybrid model's effectiveness compared to other approaches. 4. Incorporate explainable AI techniques for transparent decision-making. Objectives: 1. Literature Review: Thoroughly review credit card fraud detection research, focusing on machine learning, deep learning, and hybrid models. 2. Data Collection and Preprocessing: Gather and preprocess a comprehensive credit card transaction dataset, addressing class imbalance, missing values, and feature selection. 3. Model Development: - Machine Learning Component: Implement diverse algorithms to detect fraud patterns. - Deep Learning Component: Develop models capturing complex patterns and temporal dependencies. - Hybrid Model Integration: Combine machine learning and deep learning components into an effective hybrid model. 4. Model Training and Testing: Train the hybrid model using the dataset and assess performance with accuracy, precision, recall. 5. Comparison with Baseline Models: Compare hybrid model performance to traditional machine learning and deep learning models. 6. Explainability and Transparency: Utilize explainable AI techniques for insights into the decision-making process. 7. Documentation and Reporting: Produce a comprehensive dissertation report documenting research process, findings, and implications.

Explain your research design and outline the principal method(s) you will use	The research design for this project will follow a structured approach, comprising several key phases: 1. Literature Review: A comprehensive review of existing literature on credit card fraud detection, machine learning, deep learning, and hybrid models to identify current trends, challenges, and gaps in the research. 2. Data Collection and Preprocessing: Acquisition of a relevant dataset of credit card transactions, followed by preprocessing steps to clean and prepare the data for model training. 3. Model Development: Design and implementation of a hybrid model integrating both machine learning and deep learning techniques. 4. Model Training and Evaluation: Training the developed model on the prepared dataset and evaluating its performance using various metrics. 5. Comparison and Analysis: Comparing the hybrid model's performance with baseline models to demonstrate its effectiveness. 6. Explainability and Transparency: Implementing explainable AI techniques to interpret the model's decision-making process. 7. Documentation and Reporting: Documenting the entire research process and findings in the dissertation report. Principal Methods 1. Data Collection and Preprocessing: - Data Sources: Publicly available datasets such as the Kaggle Credit Card Fraud Detection dataset. - Data Cleaning: Handling missing values, outliers, and noise in the dataset. - Feature Engineering: Creating new features that might help improve model performance. - Data Balancing: Addressing class imbalance using techniques like SMOTE (Synthetic Minority Over-sampling Technique). 2. Model Development: - Machine Learning Algorithms: Implementing algorithms such as Random Forest, Support Vector Machines (SVM), and Gradient Boosting.

	- Deep Learning Models: Developing models such as Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks to capture complex patterns in the data.

	- Hybrid Model Integration: Combining strengths of machine learning and deep learning models into a single hybrid model. This could involve techniques like stacking or ensemble methods. 3. Model Training and Evaluation:

	- Training: Using a portion of the dataset to train the hybrid model. - Evaluation Metrics: Assessing model performance using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. - Cross-Validation: Implementing cross validation techniques to ensure the model's robustness and generalizability. 4. Comparison with Baseline Models: - Baseline Models: Training and evaluating traditional machine learning models and individual deep learning models as baselines. - Performance Comparison: Comparing the hybrid model's performance against these baselines to demonstrate its superiority. 5. Explainability and Transparency: - Explainable AI Techniques: Using methods. such as SHAP (SHapley Additive exPlanations) values or LIME (Local Interpretable Model-agnostic Explanations) to interpret the model's decisions. - Model Interpretation: Providing insights into which features are most influential in predicting fraudulent transactions.
Are you proposing to use a validated scale or published research method/tool?	No

15 Data Analysis

Does the research seek to understand, identify, analyse and/or report on data/information on terrorism/terrorism policies?	No
Does your research seek to understand, identify, analyse and/or report on information for other activities considered illegal in the UK and/or in the country you are researching in?	No
Are you analysing Secondary Data?	Yes
Is this data publicly available?	Yes
Could an individual be identified from the data? e.g. identifiable datasets where the data has not been anonymised or there is risk of re-identifying an individual	No

Are you dealing with Primary Data involving people?	No
Are you dealing with personal data?	No
Are you dealing with special category data (formerly known as sensitive data)?	No
Is the project solely desk based secondary research?	Yes
Will the data collection, recruitment materials or any other project documents be in any language other than English?	No
Are there any other ethical issues or risks of harm raised by the study that have not been covered by previous questions?	No

16 External Ethics Review

Question		Yes	No
1	Will this project be submitted for ethical rev iew to an external organisation?		X
	Name of external organisation		
2	Are you submitting to IRAS?		
3	Has this project previously been reviewed by an external organisation?		

17 Project title

Unleashing the Potential of Hybrid Models for Augmented Credit Card Security

Comments

Comment	Posted