République Algérienne Démocratique et Populaire
Université Abderrahmane MIRA de Béjaia
Faculté des Sciences Exactes

Département de Recherche Opérationnelle

Mémoire Présenté pour L'obtention du Diplôme de Master
en Mathématiques Appliquées

Spécialité: Mathématiques Financières

Multi-Objective Optimization of Random Forest for Credit Card Fraud Detection: An Operational Risk Management Approach

Présenté par :
Lalam Mouloud

Sous la direction de : Dr. Abbaci Leila
et Barache Bahia

Défendu le 29/06/2025, devant le jury composé de :

M^{r}S. Amroune	M.C. classe/ A	Président de jury	UAMB - Bejaia.
M^{r}B. Takhedmit	M.C. classe/ A	Examinatrice	UAMB - Bejaia
M^{r} L. Djerroud	M.C. classe/ A	Examinatrice	UAMB - Bejaia.
M^{r} S. Maiou	Doctorant	Examinateur	UAMB - Bejaia.

Année Universitaire 2024-2025

Acknowledgment

First and foremost, I would like to express my deepest gratitude to my supervisor, **Dr.Abbaci Leila and Barrache Bahia**, for their invaluable guidance, constructive feedback, and continuous support throughout this research. Their expertise and encouragement were crucial to the completion of this work.

I would also like to thank the faculty and staff of the **Operational Research Department** especially **Dr.L.Asli, B.Takhedmit, Z.Bouzeria** for providing a stimulating academic environment and access to resources that greatly contributed to my learning experience.

A sincere thank you goes to my classmates and friends especially **A.Waali, Khalil, A.Bouzarari** for their collaboration, helpful discussions, and moral support during challenging times.

Finally, I am profoundly grateful to my family, especially my parents, for their unconditional love, patience, and encouragement. Their belief in me has been a constant source of strength and motivation throughout my academic journey.

To all who have contributed to this work in any way thank you.

Contents

List of Figures

List of Tables

General introduction

In today's data-driven financial ecosystem, the rapid surge in electronic payments has revolutionized the convenience of transactions while simultaneously amplifying the risk of fraud. Among the most pressing challenges is credit card fraud, which continues to evolve in complexity as malicious actors exploit technological advancements to bypass traditional security systems. This dynamic threat landscape demands the development of intelligent, adaptive, and scalable fraud detection models.

The core motivation of this research is to enhance the detection of fraudulent transactions in highly imbalanced datasets using advanced machine learning and optimization techniques. Unlike conventional systems that often struggle to balance detection accuracy with computational efficiency, our approach seeks to improve recall especially for minority fraud cases without compromising precision or introducing excessive false positives.

To achieve this, our primary contribution lies in the design and implementation of a hybrid methodology that combines a **Random Forest classifier** with **multi-objective hyperparameter optimization** using the **Non-dominated Sorting Genetic Algorithm II (NSGA-II)**. This integration allows for the exploration of trade-offs between recall and precision, offering a range of Pareto-optimal solutions for better model performance in fraud detection scenarios. The model is trained and validated on a real-world credit card dataset and carefully evaluated using robust performance metrics. The final results demonstrate a significant improvement in the model's ability to correctly identify fraudulent transactions, even in the presence of severe class imbalance.

To support this work, the thesis is structured as follows:

- **Chapter 1** introduces the background of enterprise risk, with a special focus on operational and fraud risk taxonomy.
- **Chapter 2** presents the foundations of machine learning and describes classification methods such as decision trees and Random Forests.
- **Chapter 3** details the NSGA-II algorithm and its use in multi-objective optimization.
- **Chapter 4**, the core of this study, demonstrates the implementation and evaluation of our optimized fraud detection model.

Through this interdisciplinary approach, the study contributes to the development of more reliable, interpretable, and effective fraud detection systems supporting financial institutions in minimizing loss, enhancing trust, and staying ahead of adversarial fraud patterns.

1 Risk Operational

Introduction

In an increasingly volatile and digitized financial environment, risk management has become a strategic imperative for institutions worldwide. The ability to classify, understand, and mitigate a wide range of risks—financial, non-financial, and emerging—requires a structured approach rooted in standardized taxonomies and regulatory frameworks. This chapter explores the foundations of enterprise risk taxonomy, detailing how firms classify and govern diverse risk types to ensure resilience and compliance. It also delves into the operationalization of these frameworks under international standards such as Basel II, with particular emphasis on operational and fraud risk. Through case studies, empirical data, and regulatory insights, we aim to provide a comprehensive overview of how modern institutions address the challenges posed by internal failures, external threats, and fraud-related losses in a risk-sensitive economy.

1.1 Risk and Enterprise Taxonomy

Understanding and organizing risks through a structured taxonomy is essential for effective enterprise risk management.

1.1.1 General Risk Definition

Risk is the possibility or likelihood of an unwanted event or loss occurring due to uncertainty in outcomes. In financial and business contexts, **risk** denotes the measurable uncertainty regarding deviations from expected outcomes, characterized by the potential for loss or adverse consequences. Formally, risk (R) can be expressed as:

$$R = P \times I$$

Where:

- P = Probability of occurrence
- I = Impact magnitude

This encompasses both threats to objectives and opportunities for strategic advantage [**?**].

1.1.2 Firmwide Risk Taxonomy

A **firmwide risk taxonomy** is a structured classification system that categorizes all the risks a firm faces, enabling a consistent and comprehensive approach to risk identification, assessment, and management. Broadly, risks are grouped into three categories: **Financial Risks**, **Non-Financial Risks**, and **Strategic & Emerging Risks**.

Financial Risks include Market Risk, Credit Risk, and Liquidity Risk—each arising from exposure to market fluctuations, counterparty defaults, or funding constraints. **Non-Financial Risks** involve Operational Failures, Legal and Compliance issues, Cybersecurity threats, and Reputational damage, typically stemming from internal processes, external events, or stakeholder perceptions. **Strategic & Emerging Risks** encompass long-term uncertainties such as Strategic Missteps, Model Failures, ESG and Climate-related challenges, and Geopolitical Instability.

Together, this taxonomy supports firmwide risk governance by promoting clarity, accountability, and alignment with regulatory expectations[9].

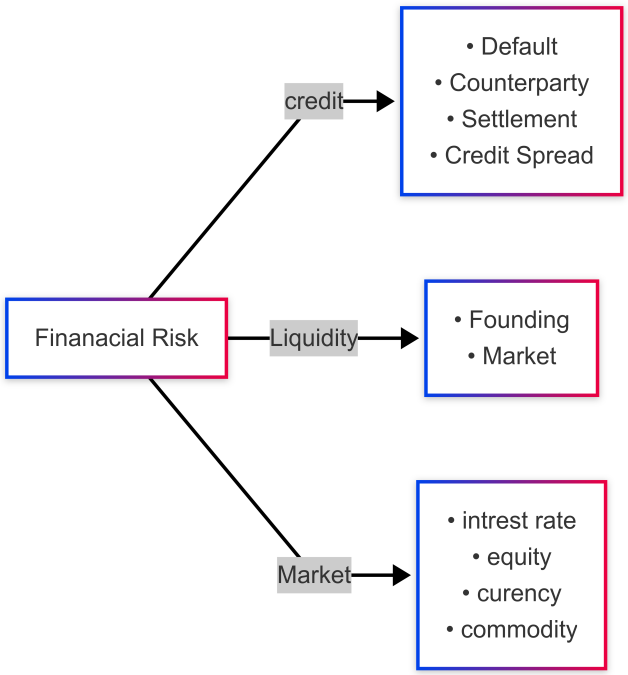

Figure 1.1: Financial Risks

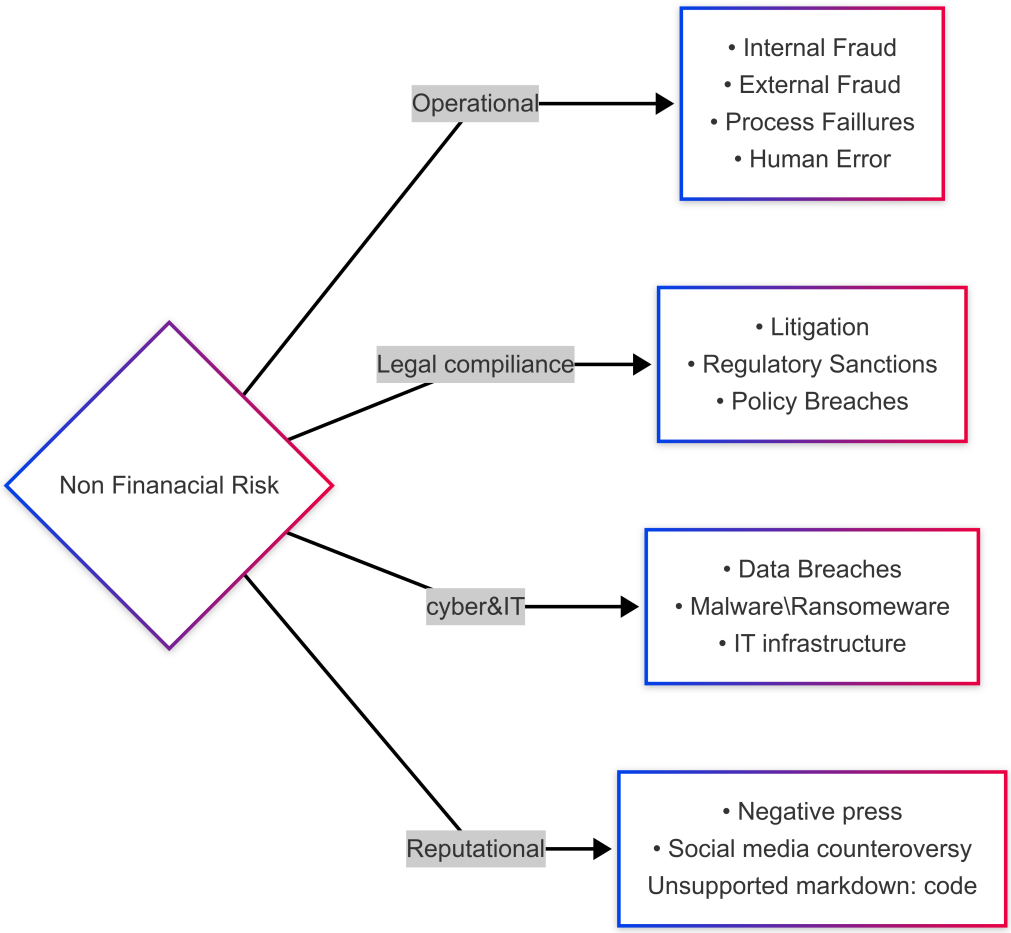

Figure 1.2: Non Finanacial Risks

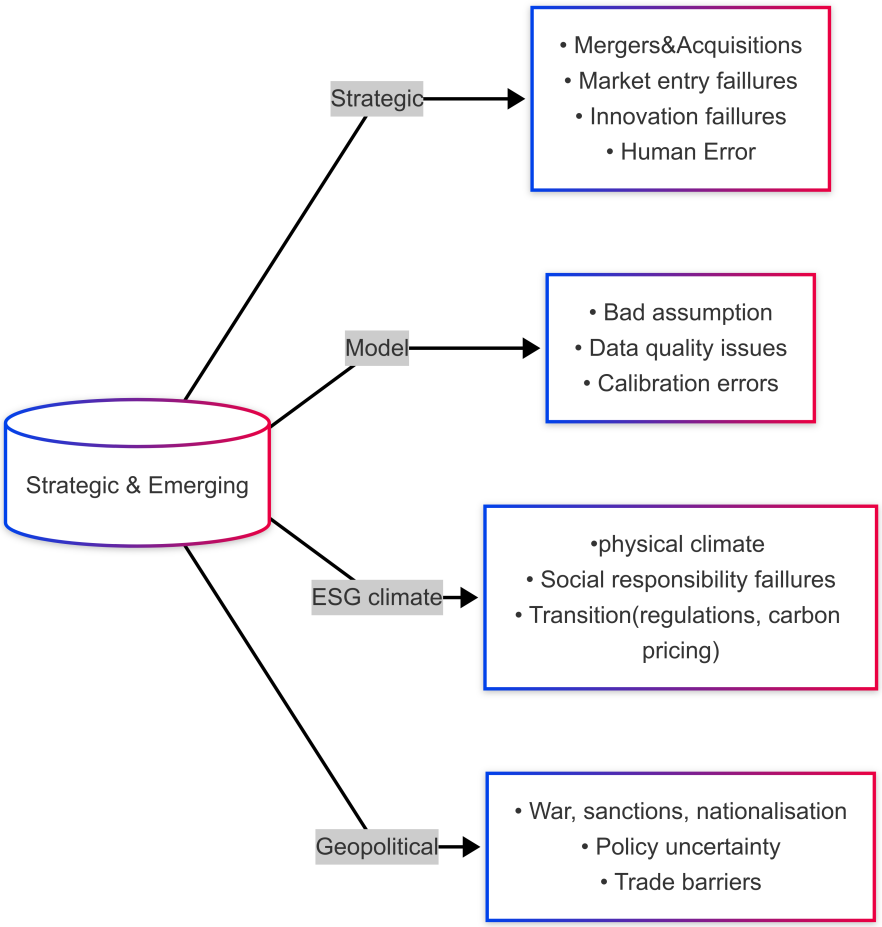

Figure 1.3: Strategic & Emerging Risks

1.1.3 Applications of Risk Taxonomy in Governance

A well-defined **risk taxonomy** is a foundational element in strengthening governance frameworks across financial institutions and large corporations. It provides a standardized structure for identifying, assessing, and managing risks effectively across all organizational levels.

Risk Identification and Categorization

A unified taxonomy ensures consistent risk identification across departments. This prevents duplication or omission of risk categories and facilitates internal control and audit processes.

Policy Development and Risk Appetite Definition

Risk taxonomy supports the definition of risk appetite by clearly delineating risk categories (e.g., credit, market, operational). Organizations can then allocate tolerance levels and policy limits accordingly.

Internal Controls and Risk Ownership

Each risk category can be linked to specific control measures and owners. This enhances accountability and monitoring through key risk indicators (KRIs) and escalation mechanisms.

Regulatory Reporting and Compliance Alignment

Supervisory authorities (e.g., Basel Committee, EBA)[2] require structured and comparable risk reports. Taxonomies help organizations align their internal governance with external regulatory requirements (e.g., Basel III/IV, Solvency II).

Scenario Analysis and Stress Testing

A clear taxonomy allows targeted scenario analyses and stress tests, enabling firms to evaluate their resilience under shocks such as cyberattacks, market crashes, or operational disruptions.

Cross-Risk Correlation and Aggregation

With a structured taxonomy, organizations can analyze the interdependencies between risk categories, leading to a comprehensive view of enterprise risk exposure.

Strategic Risk Management and Board Oversight

Risk taxonomy enhances board-level oversight by enabling clear visualization of aggregated risks and ensuring that strategic decisions reflect the firm's overall risk posture.

This framework enables:

1. Holistic risk exposure mapping across organizational functions
2. Clear ownership assignment for risk mitigation
3. Quantification approaches tailored to risk subtypes:
    - Value-at-Risk (VaR) for market exposures
    - Stress testing for credit/downgrade scenarios
    - Key Risk Indicators (KRIs) for operational risks

1.2 Operational Risk Management under Basel II

Basel II, introduced in 2004[1], revolutionized operational risk management by formalizing definitions, quantification methods, and capital requirements. This section examines its regulatory architecture, risk typologies, fraud dynamics, and implementation hurdles, contextualized through case studies and emerging data practices.

1.2.1 Basel-Compliant Risk

Basel II defines operational risk as *"the risk of loss resulting from inadequate or failed internal processes, people, systems, or from external events"*. This definition categorizes operational risks separately from credit and market risks, mandating banks to:

Quantify exposure using Advanced Measurement Approaches (AMA), combining: Internal loss data, External benchmarks, Scenario analyses, and Business environment factors[35].

Allocate capital against operational risks under:

- **Pillar 1**: Minimum capital requirements
- **Pillar 2**: Supervisory review
- **Pillar 3**: Disclosure transparency

Approaches:

Standardised Approach (SA) Fixed percentages (12–18%) applied to gross income across eight business lines

Advanced Measurement (AMA) Models loss distributions using statistical methods like Loss Distribution Approach (LDA)

1.2.2 Scope and Key Risk Drivers

Risk drivers are factors that significantly impact the likelihood or severity of risks. They vary by industry and context but generally fall into categories such as:

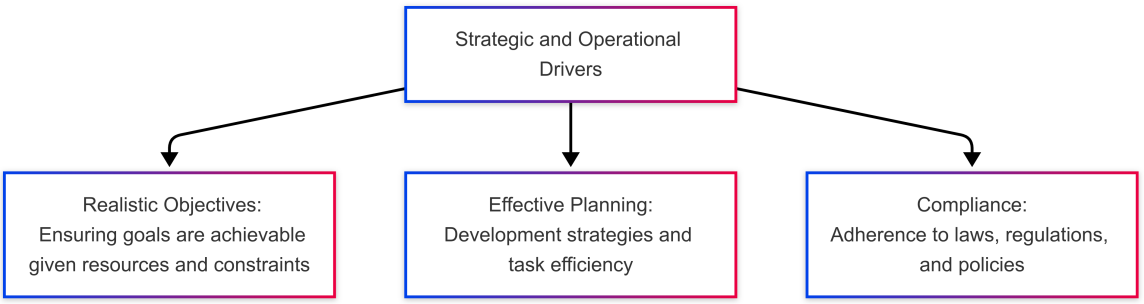

Figure 1.4: Strategic and operational drivers

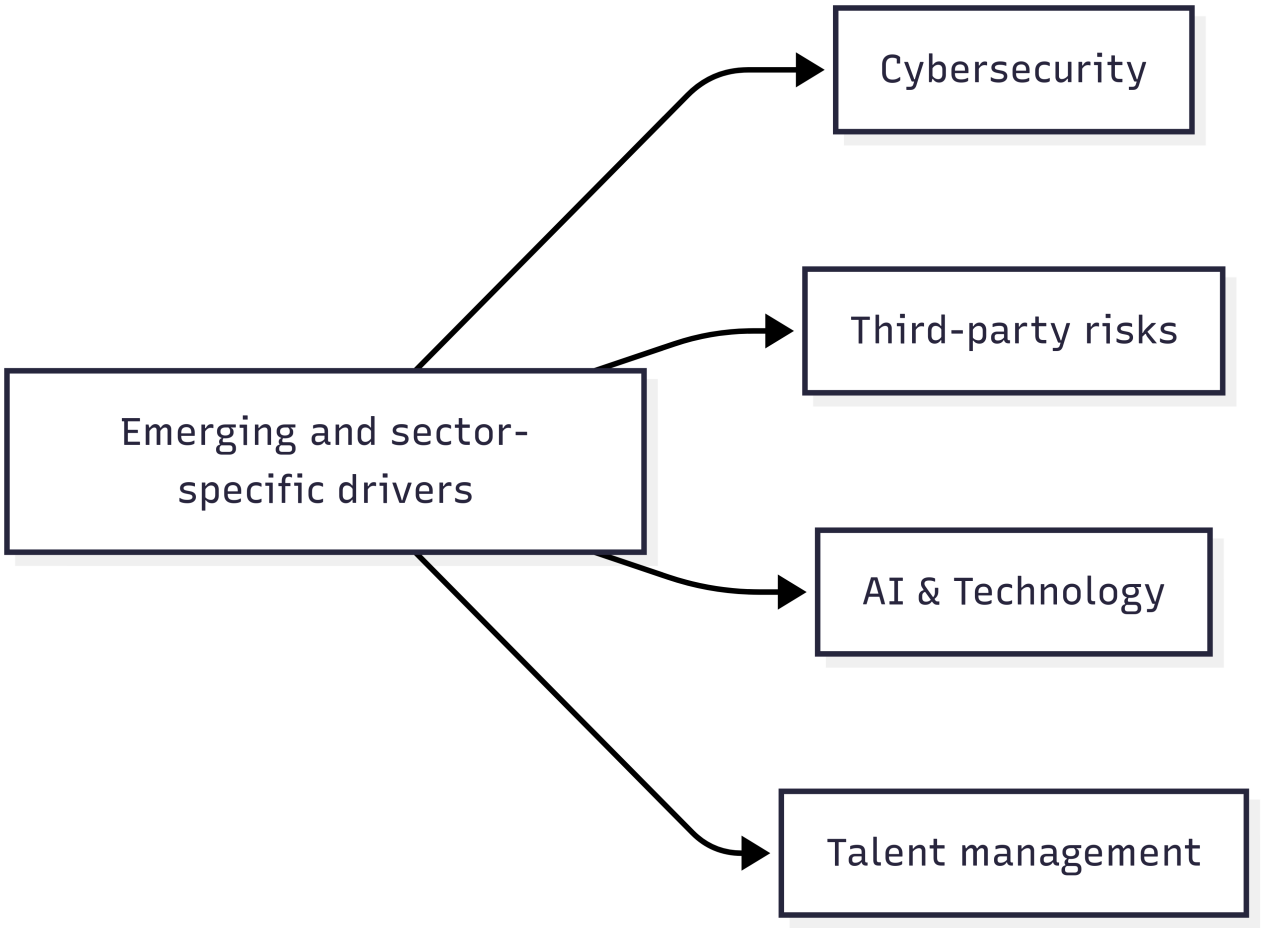

Figure 1.5: Emerging and sector-specific drivers

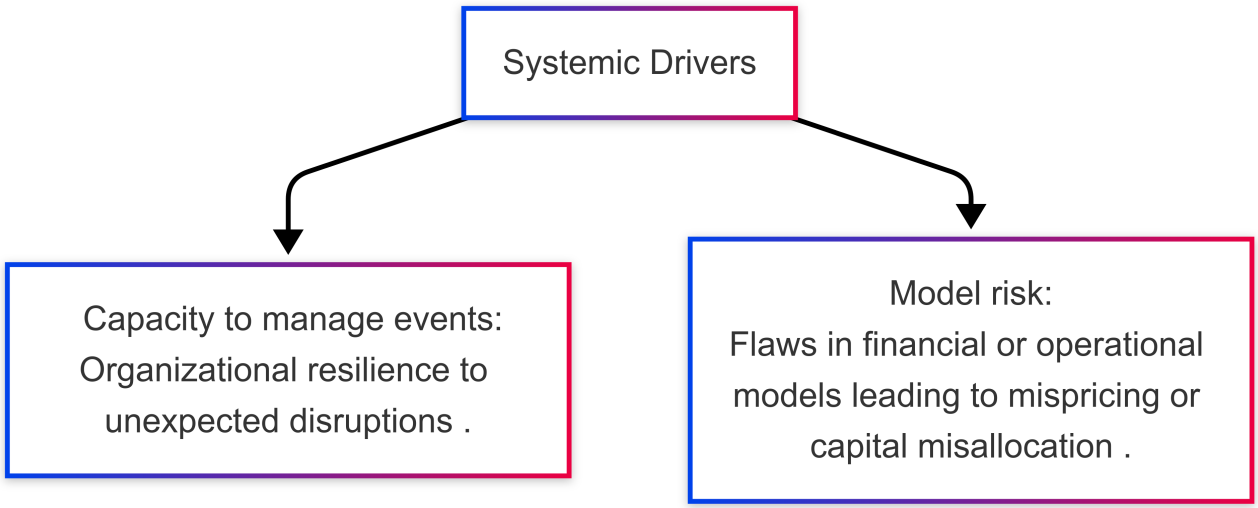

Figure 1.6: Systemic drivers

1.2.3 Operational Risk

Basel II delineates operational risk event types such as:

Internal Fraud: Deliberate deception by employees (e.g., unauthorized trading, embezzlement) for example: Nick Leeson's rogue trading at Barings Bank (1995), causing £827M losses

External Fraud: Third-party theft, hacking, or forgery for example: Carbanak cyberattacks (2014–2016), siphoning $1B from 100+ banks.

1.2.4 Internal and External Fraud Dynamics

Involve distinct yet equally critical threats to financial institutions.

Internal fraud typically arises from weak internal controls such as poor segregation of duties or a permissive corporate culture. A classic example is the collapse of Barings Bank, where Nick Leeson exploited dual trader and back-office roles to conceal massive losses. In response, reforms like the "four eyes principle" (requiring dual approval) and advanced behavioral analytics (e.g., HSBC's Falcon AI) have been introduced to detect anomalies early.

In contrast, *external fraud* has evolved with technological advancements, as seen in the Carbanak gang case, where cybercriminals used spearphishing and video surveillance to infiltrate banks' payment systems. Defensive strategies now include network segmentation and blockchain-based audit trails to enhance transparency and limit access. However, a regulatory gap remains: static capital buffers under Pillar 1 may not adapt swiftly enough to counter rapidly evolving fraud tactics, highlighting the need for more dynamic risk sensitive approaches.

Data limitations pose a significant challenge in operational risk modeling. One key issue is the *threshold bias* arising from the exclusion of losses below 10.000, which can distort the risk profile and underestimate true exposure. Additionally, *external data sources*, such as the P-COLD database (2023), face limitations—particularly with non-English incident reports—reducing both the reliability and completeness of available data.

Another major concern is **model risk**, especially the tendency to overfit Advanced Measurement Approach (AMA) models to minimize regulatory capital requirements. In response, regulators now mandate the inclusion of "unlikely but plausible" stress scenarios to enhance model robustness. For example, an EU bank's AMA model resulted in a 40% reduction in capital requirements compared to the Standardized Approach (SA), raising concerns about undercapitalization.

Organizational resistance further complicates AMA implementation. In 2004, it was reported that

68% of traders ignored established risk thresholds, illustrating the persistence of *siloed risk cultures*. Additionally, the high *cost burden* of compliance—estimated at 5-10 million annually for mid-sized banks—has discouraged adoption.

The **business impact** of operational risk events is also substantial. Following a major misconduct scandal, Wells Fargo experienced a loss of $20 billion in customer deposits[34]. In other cases, firms exhibited *strategic paralysis*, curbing innovation and product development due to heightened risk aversion.

Future improvements in operational risk management should focus on implementing adaptive capital buffers, promoting cross-institutional data sharing, and integrating AI-driven surveillance systems to enhance the early detection and control of risk exposures.

1.3 Fraud Risk: Concepts and Frameworks

1.3.1 Definition and Types of Fraud

Fraud risk is the potential that an individual, organization, or system might engage in deceptive or dishonest activities to gain financial or other benefits by misrepresenting information or exploiting vulnerabilities.

1.3.2 Credit Card Fraud in the Banking Sector

Credit card fraud constitutes the most prevalent form of financial crime in retail banking, accounting for approximately 42% of all banking fraud incidents globally. This sophisticated threat landscape evolves continuously, with fraudsters developing new techniques to exploit vulnerabilities in payment ecosystems.

1.3.2.1 Fraud Mechanics

1. **Data Harvesting:** Criminals obtain card details through:
    - Phishing campaigns (43% of breaches)
    - Malware-infected POS systems (29%)
    - Dark web marketplaces (average card price: $12-20)
    - Data breaches (over 1.1 billion records exposed in 2023)
2. **Validation Testing:** Small transactions ($0.99-$5.00) to verify card viability
3. **Monetization:** High-value purchases or cash withdrawals
    - Electronics and luxury goods (62%)
    - Gift card purchases (18%)
    - Cryptocurrency acquisitions (15%)
4. **Laundering:** Reselling goods through online marketplaces

1.3.2.2 Advanced Attack Vectors

Table 1.1: Emerging Credit Card Fraud Techniques

Technique	Mechanism & Impact
BIN Attack	Generate valid card numbers using Bank Identification Numbers (BINs) and brute-force algorithms. Success rate: 1 in 200 attempts.
AI-Powered Fraud	Machine learning algorithms that mimic normal spending patterns to evade detection. Accounted for $850M losses in 2023.
Multi-Channel Attacks	Combine physical card cloning with online account takeover. Average loss: $12,500 per incident.
Supply Chain Compromise	Intercept cards during delivery or compromise card issuance systems. Durbin Amendment compliance gaps increase vulnerability.
Token Manipulation	Exploit tokenization systems through replay attacks or token generation flaws. Emerging threat in mobile wallet ecosystems.

1.3.2.3 Detection & Prevention Framework

Modern banks deploy multi-layered defense systems that integrate real-time detection techniques and preventive controls. Real-time detection uses behavioral analytics, comparing transactions to over 200 parameters, such as purchase velocity, device fingerprinting, and behavioral biometrics like keystroke dynamics and swipe patterns . Network analysis helps uncover hidden links between fraudulent events, while adaptive machine learning models recalibrate every 4–6 hours to detect evolving threats.

Preventive controls are equally essential. EMV chip technology has reduced counterfeit fraud by 76% in many regions. The 3-D Secure 2.4 protocol uses over 300 device attributes, behavioral data, and contextual signals to perform risk-based authentication. Token vaults replace primary account numbers with merchant-specific tokens to prevent data reuse, and end-to-end encryption secures transaction data from the point-of-sale to the processor [24].

1.3.2.4 Regulatory Landscape

Fraud risk management is governed by an evolving regulatory framework. PCI DSS 4.0 mandates encryption and multi-factor authentication. Regulation E limits consumer liability in the U.S. to $50, and the EU's PSD2 regulation enforces Strong Customer Authentication (SCA). Regulatory penalties are severe—GDPR violations average $2.1 million in fines per breach, while PCI non-compliance can cost up to $100,000 per month.

1.3.2.5 Case Study: The 2023 Poly Network Exploit

The 2023 Poly Network breach highlighted systemic vulnerabilities in API security. Hackers compromised API keys at a card processor and generated valid tokens for 240,000 accounts, causing a $37 million loss across 18 institutions . Fraud systems failed to flag the activity, as the transactions mimicked typical holiday shopping behavior. The breach accelerated industry-wide API security reforms.

1.3.2.6 Economic Impact Analysis

The financial impact of fraud is growing. Direct losses are expected to reach $43.6 billion by 2025, growing at a 16.2% CAGR . Indirect costs include $15–$25 per fraud investigation, $3–$5 per reissued card, and 5–7× loss value in customer recovery costs. With over 700 million cards replaced annually, this impact is substantial . Reputational damage is another concern—27% of customers change banks after a fraud event, and stock prices fall by an average of 3.5% following major breaches .

1.3.2.7 Future Trends & Countermeasures

Emerging technologies aim to address persistent fraud challenges. Quantum cryptography seeks to make data resilient to quantum attacks, while blockchain-based decentralized identity systems restore personal data control. Federated learning enables institutions to build shared fraud detection models without exchanging sensitive data . Behavioral DNA introduces continuous authentication via subtle biometric patterns. Regulatory evolution will likely require real-time payment tracking and cross-network visibility. However, high false positive rates—still at 15–20%—continue to cost merchants over $20B annually in lost legitimate sales.

Conclusion

This chapter has demonstrated that robust risk taxonomies and regulatory frameworks are essential tools for enhancing enterprise-wide risk governance. By categorizing risks into financial, non-financial, and strategic domains, organizations can allocate resources more effectively, define risk appetites, and ensure alignment with supervisory expectations. Basel II's emphasis on operational risk quantification—through approaches like AMA and SA—highlights the importance of structured modeling, scenario analysis, and internal control systems. The evolving threat of fraud, particularly in the context of digital payments, necessitates multilayered defenses, adaptive machine learning models, and regulatory vigilance. As fraud techniques become more sophisticated and systemic drivers intensify, future risk management will hinge on technologies such as federated learning, behavioral biometrics, and quantum-resilient security. Ultimately, fostering a proactive, data-informed, and ethically grounded risk culture remains the cornerstone of long-term institutional resilience.

2
Machine Learning

Introduction

Data mining is a field born from the explosion of information volumes, thanks to advances in processing speed and storage capabilities. It aims to extract valuable information from large amounts of data to better understand it or predict its future behavior. Relying on statistical tools and **Machine Learning**, it involves discovering new knowledge by exploring vast datasets, often stored in data warehouses. These discoveries can include correlations, patterns, or general trends that were initially unknown. In science and engineering, although mathematical models are often used to analyze and verify systems, some complex or poorly understood problems require a data-driven approach based on experimental data, where data mining plays a crucial role. Since its emergence, this field has been enriched with numerous tools from statistics and artificial intelligence, with significant contributions from **machine learning** techniques, which automate and improve the efficiency of predictive and descriptive analyses. In this context, random forests stand out as one of the most powerful and popular supervised learning models. This method represents a major contribution to knowledge extraction from data, offering reliable and high-performance solutions to the challenges of data analysis and modeling. In this paper, we will further explore this model, its characteristics, evolution, and applications.We address in this thesis knowledge extraction from data using random forests, which combine the robustness of decision trees, the power of randomness that increases tree diversity in the forest, and the flexibility of fuzzy logic. They have the specificity to handle imperfect data, reduce error rates, and demonstrate greater robustness and interpretability.

2.1 Data Mining

2.1.1 Data Mining Process

It is essential to understand that data mining is not just about discovering patterns in a dataset. It is only one step in an entire process followed by scientists, engineers, or anyone seeking to extract knowledge from data.[38] In 1996, a group of analysts defined data mining as a process composed of five steps under the CRISP-DM standard (Cross-Industry Standard Process for Data Mining) as illustrated below:

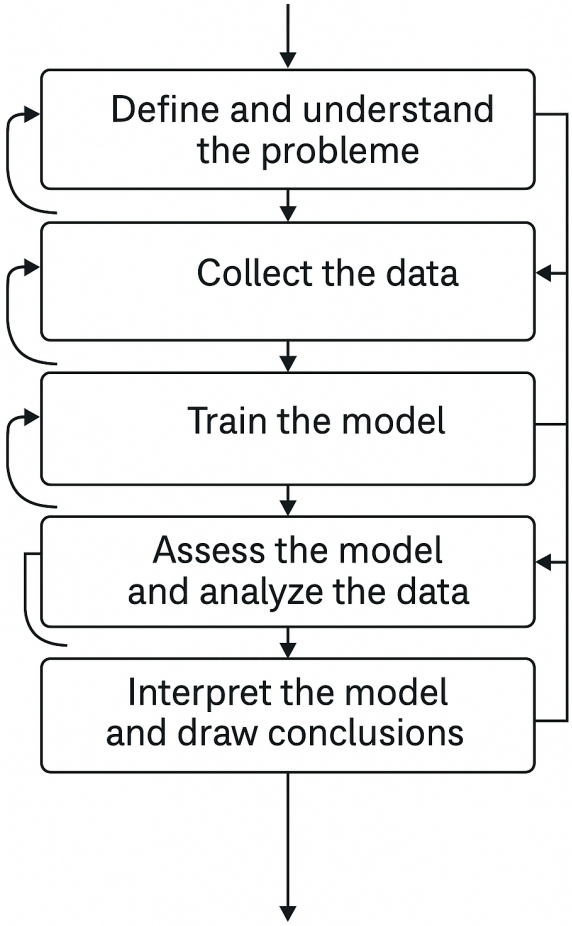

Figure 2.1: Data Mining Process

The data mining process consists of five interconnected and non-linear steps. Each step may require adjustments based on subsequent discoveries. Here is a summary of the steps:

- **Problem Understanding:**
  It is crucial to clearly define the problem and understand the application domain to prepare the necessary data and correctly interpret the results. Good understanding allows evaluating result relevance and justifying their cost.
- **Data Collection:**
  Data is generated and collected from various sources (databases, files, etc.). It is often divided into two sets: one to build the model (80%) and one to test it (20%).
- **Data Preprocessing:**
  Raw data must be cleaned, normalized, and sometimes reduced (e.g., via PCA). This corrects anomalies and prepares data for analysis. Visualization tools assist in this step.

- **Model Estimation:**
  Various techniques (neural networks, decision trees, clustering, etc.) are used to build models. Results are compared to select the most appropriate method.

- **Interpretation and Conclusions:** The final objective is to provide understandable models to facilitate decision-making. Simple models are generally more comprehensible but less accurate, while complex models offer better accuracy but are harder to interpret.

In summary, this process aims to extract actionable knowledge while adapting methods to users' specific needs.

Classification

Classification is the most common task in data mining and appears to be a fundamental human activity. To understand our daily lives, we are constantly required to classify, categorize, and evaluate. Classification involves studying the characteristics of a new object to assign it to a predefined class. The objects to classify are typically database records, and classification involves updating each record by determining the value of a class field. Classification operates in two phases. The first is the learning phase, where classification approaches use a training set where all objects are already associated with known reference classes. The classification algorithm learns from the training set and builds a model. The second phase is the actual classification phase, where the learned model is used to classify new objects.

Regression

Regression is a type of learning where the model's output is a continuous value. The objective is to predict a numerical quantity.

- Example: Predicting a house price based on features like area, location, number of rooms, etc.
- Output: A continuous numerical value (e.g.: €350,000).

2.2 Machine Learning

Machine learning (**ML**) is a field of artificial intelligence that aims to give machines the ability to learn from data through mathematical models. More precisely, it is the process by which relevant information is extracted from a dataset. After exploring the fundamental concepts of data mining, we will now examine how **machine learning** enriches this process by automating knowledge extraction and improving analysis accuracy[15].

2.2.1 Contributions of Machine Learning to Data Mining

Data mining uses statistical techniques and **machine learning** to automate data analysis, identify complex patterns, and make accurate predictions without human intervention, learning directly from data to extract trends and useful information from large datasets.

- **Automation of Data Analysis:**
  Machine learning enables autonomous analysis of large databases without human intervention, learning patterns from raw data.

- **Discovery of Hidden Relationships:**
  Machine learning algorithms identify complex relationships, trends, and structures that would be difficult to detect with traditional methods (e.g., classical statistics).
- **Prediction and Decision Making:**
  Machine learning predicts future events from historical data, such as sales forecasting, fraud detection, or medical diagnosis.
- **Continuous Improvement:**
  In machine learning, models can improve as they are exposed to new data, making knowledge extraction more accurate over time.

2.2.2 Supervised & Unsupervised Learning

Supervised Learning

Supervised learning in machine learning involves training a model with labeled data, where each data point is paired with a corresponding label . The goal is to enable the model to predict or classify new, unseen data based on these labeled examples [36].

Key Features:

- **Labeled Data:** The data consists of input (features) and the correct output (label).
- **Prediction or Classification:** The model learns to predict outputs for new data or classify data into categories .
- **Evaluation:** The model's performance can be quickly evaluated using metrics like accuracy, precision, and recall.

Standard Algorithms

- **Linear Regression** : for predicting continuous values.
- **Logistic Regression**: for binary classification.
- **Decision Trees** : for both classification and regression.
- **k-Nearest Neighbors (k-NN)** : for classification and regression.

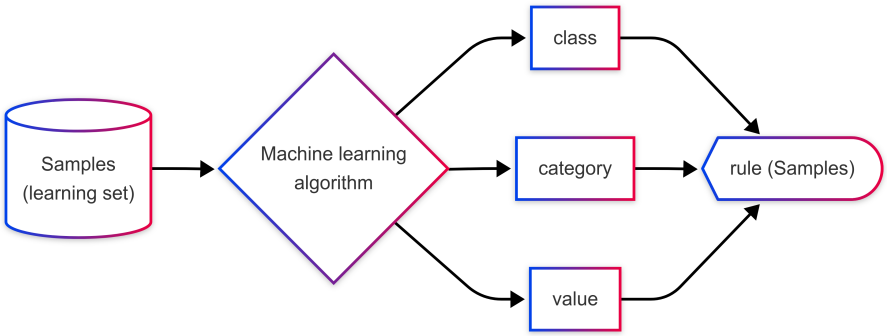

Figure 2.2: Learning&Training

- **Learning&Training:** build a classification or regression rule from a set of samples.

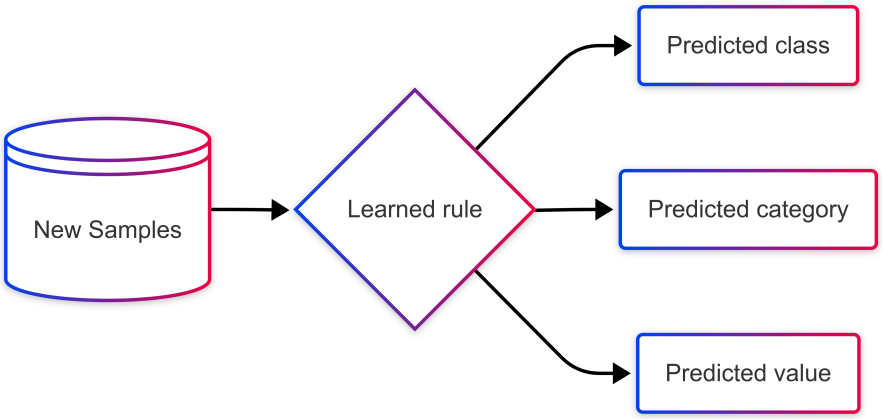

Figure 2.3: Prediction

- **Prediction:** assign a class or value to new samples.

Unsupervised Learning

Unsupervised learning, on the other hand, works with unlabeled data. The data does not have any predefined labels or correct answers. Instead, the goal of unsupervised learning is to identify patterns, structures, or groupings in the data without knowing what the outcomes should be.

Key Features

- **Unlabeled Data:** The data only includes input features with no associated output labels.
- **Pattern Discovery:** The model finds patterns, relationships, or groups within the data independently.
- **Evaluation:** Evaluating unsupervised learning models can be more subjective. It often uses internal metrics like cluster quality or dimensionality reduction effectiveness.

Standard Algorithms

- **k-Means Clustering**: for grouping similar data points together.
- **Principal Component Analysis (PCA)** : for reducing the number of features in the data.
- **DBSCAN**: for identifying clusters of varying shapes.
- **Hierarchical Clustering**: for creating a hierarchy of clusters.

In essence,understanding the difference between supervised and unsupervised learning is essential for choosing the right machine learning approach. Both techniques have unique strengths, and selecting between them depends on your available data and the problem you're trying to solve. Supervised learning is best for tasks where you have labeled data and need to make predictions or classifications. Unsupervised learning is perfect when you have unlabeled data and want to discover hidden patterns or groupings.

2.3 Decision Trees

A decision tree is the most popular method for classification and prediction in supervised learning. It is simple to use and interpret while maintaining very acceptable performance, delivering satisfactory results for complex problems (random forests). It is widely used in statistics, engineering, decision theory, and machine learning [27, 3].

These recursive partitioning methods were introduced as early as the 1960s. Many approaches have been proposed. The **CART** method (for *Classification And Regression Trees*), introduced by Breiman et al. (1984) is the most well-known. We present this method in the following section[8].

2.3.1 Construction of a Decision Tree

Decision trees are hierarchical supervised prediction models that behave as a series of successive conditional tests. They can be represented as a tree and are used for **classification** and **regression** problems.

- Each node in the tree tests a condition on a variable, and each of its children corresponds to a possible response to that condition.
- The leaves of the tree correspond to a label.
- To predict the label of an observation, we "follow" the responses to the tests from the root of the tree and return the label of the leaf we reach.

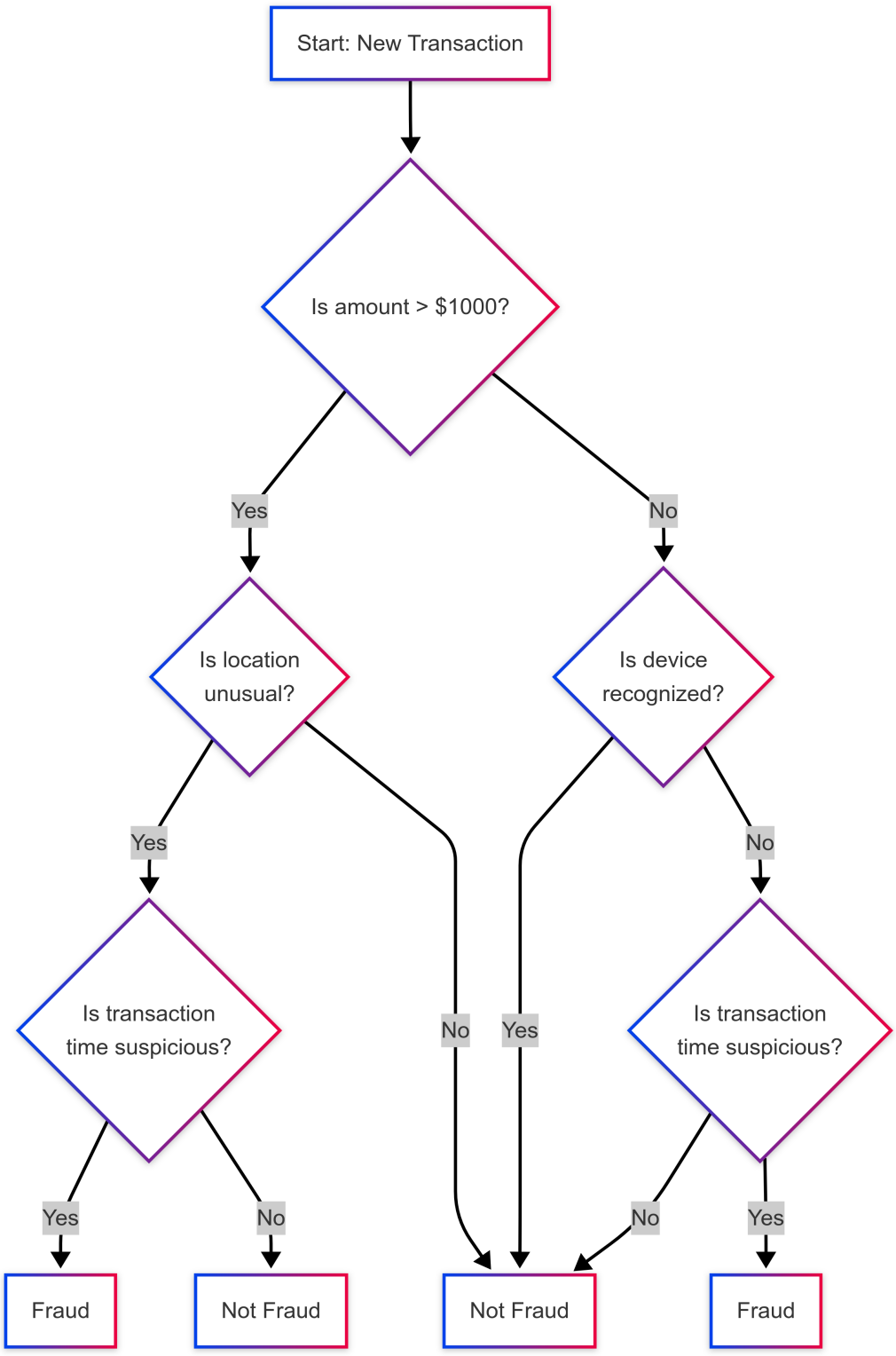

Figure 2.4: Decision tree for labeling a fraud

2.3.1.1 CART Trees

CART is an effective non-parametric method, simple to implement, and usable for both regression and classification. The general principle of **CART** is to construct a prediction rule through recursive binary partitioning of the data space. The resulting partition can be represented as an easily interpretable binary tree. Figure 4 illustrates the correspondence between a dyadic partition and a binary tree.

Divides data into subsets to build a decision tree. **CART** is an effective top-down greedy method.

The general principle of **CART** is to construct a prediction rule through recursive and **binary** partitioning of the data space. The resulting partition can be represented as an easily interpretable binary tree as shown in the following figure.

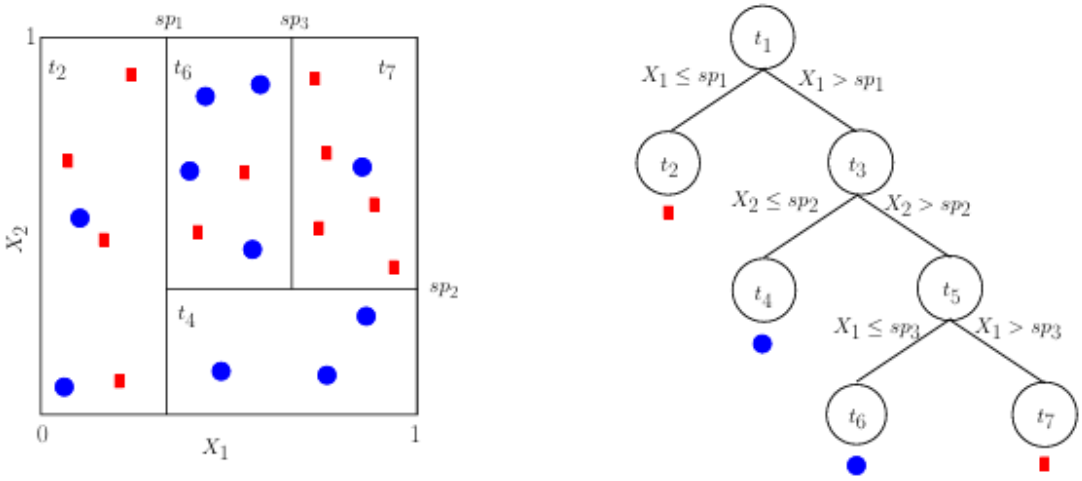

Figure 2.5: An example of CART tree in binary classification. Each leaf is associated with the best represented class.

2.3.2 Limitations of Decision Trees

- Sensitivity to overfitting (high variance, low bias when not pruned)
- Conversely, an overly simple decision tree (a tree with a single level) may suffer from underfitting (low variance, high bias)
- Instability with small variations in training data

2.4 Ensemble Methods

2.4.1 Definition

The general principle of ensemble methods is to construct a collection of predictors and then aggregate all their predictions [17]. In the regression framework: aggregating the predictions of q predictors amounts to averaging them, for example. Each predictor provides a y_i and the final prediction is then:

$$\frac{1}{q}\sum_{i=1}^{q} y_i \qquad (1.1)$$

In the classification framework: aggregation amounts to majority voting among the classes provided by the predictors. Ensemble methods generate multiple prediction rules and then pool their different responses. The heuristic behind these methods is that the final predictor should be better than each individual predictor. The heuristic explaining the success of these ensemble methods can be summarized as :

- Each individual predictor should be relatively good.
- Individual predictors should be different from each other.

Ensemble methods rely on either a random construction of a family of models (**"Bagging"**) or an adaptive, deterministic or random construction of a family of models (**"Boosting"**). There are two types of ensemble methods:

1. **Heterogeneous Ensemble Methods:**
   Combine a set of hypotheses from different algorithms on the same training set.

2. **Homogeneous Ensemble Methods:**
   Combine a set of hypotheses produced by the same algorithm on different training sets. They use adaptive strategies (**boosting**) or random strategies (**bagging**). In our work, we focus only on homogeneous ensemble methods such as:

2.4.2 Bagging

The word bagging is a contraction of **Bootstrap** and **Aggregating**, coined by Breiman in 1996. It is effective for correcting the lack of robustness in decision trees. It involves constructing a base rule on different **bootstrap** samples. We modify the predictions, thus building a collection of diverse predictors, which are then aggregated by voting (majority vote of model results) or averaging of estimates. This step produces a high-performing predictor by combining several classifiers.

Bootstrapping A bootstrap of a set T is the set obtained by drawing T elements from T uniformly at random with replacement. Bootstrapping a training set T produces a new set T' that contains on average 63% unique instances different from T when $T \gg 1$ [32] (creating multiple **bags** from a database).

Aggregation We produce several **bootstraps** T_m, each **bootstrap** T_i being used to train a predictor t_i (think here of a decision tree, but the technique applies to any family of predictors). Given an instance (x, y), we run each tree to get a set of predicted y_m values. These are then aggregated by calculating their average :

$$y = \frac{1}{m} \sum_{i=1}^{m} y_i \quad (1.2)$$

The main strength of bagging is thus to reduce instability to improve generalization performance. But another strength of bagging is the out-of-bag measures. We now explain this very useful tool in classification.

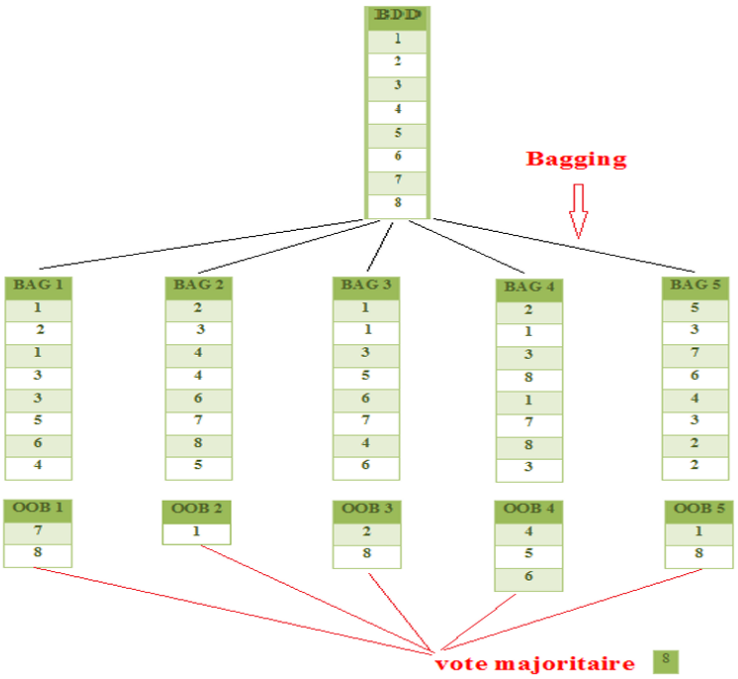

Figure 2.6: Illustration of the Bagging principle for a set of decision trees

Example

2.4.3 Out-Of-Bag Measures (OOB)

Consider a training base A of m data. If we perform random sampling with replacement, it is possible to draw the same data multiple times for the same **bootstrap** sample. Knowing that for each bootstrap sample, 63% of examples are unique to A, the rest being duplicates. This result is interesting because it indicates that each classifier learns only part of the data. The other data, which it does not know, are called **out-of-bag (OOB)** data. In the context of an ensemble of classifiers, if we wish to estimate generalization error via majority voting, it suffices to classify each training data of A by letting only those elementary classifiers vote for which it is part of the **out-of-bag** set. According to Breiman, many tools are based on the use of random forests, and many of them use **out-of-bag** data to avoid needing a validation dataset .

2.4.4 Boosting

Boosting is one of the most powerful ensemble methods today, introduced by Freund and Shapire in 1996 . It is an adaptive version of Bagging whose starting principle is the same, giving more weight in the next step to poorly predicted observations. Its goal is to improve learning algorithm performance and progressively create ensemble methods. The two mentioned ensemble methods share a common general principle: starting from a base prediction rule and perturbing this base rule.

2.5 Random Forests

We now detail random forests (Random Forest), which is an improvement of Bagging specific to binary tree models (**CART**). Partitioning methods and particularly the **CART** method enjoy significant success. They are now commonly used in many domains. However, these methods prove to be very unstable. Indeed, a simple perturbation of a few observations in the training sample can completely modify the constructed tree. The random forests of Breiman (2001) solve this weakness of decision

trees and improve predictive performance. We now recall the main principles of this method. There are several random forest methods, so to fix ideas, in this manuscript, the term random forests refers to the method introduced by Breiman (see the theses of Genuer (2010) and Scornet (2015) for a complete overview of different forest models).

2.5.1 Definition

A Random Forest is a classifier consisting of a set of elementary classifiers of binary decision tree type in which randomness has been introduced. Let $h(O_1)...h(O_q)$ be a collection of tree predictors, where $(O_1...O_q)$ is a sequence of random variables independent of the training sample. The random forest predictor is obtained by aggregating this collection of predictors.The term random forest comes from the fact that the individual predictors are explicitly tree predictors here, and from the fact that each tree depends on an additional random variable. A random forest is the aggregation of a collection of random trees. It can be constructed by generating random subsets of features for each tree and generating random subsets of training data for each tree (as in the **Bagging** method). **RF** generates a set of doubly perturbed trees where each tree in the set is generated starting from a **bootstrap** subsample of the complete training set.Many random forest models have been created corresponding to different ways of incorporating randomness in trees. Examples include:

- **Tree Bagging:** [14]introduces randomness in the initial sample by selecting certain points rather than others and lets the tree grow until each node contains a single element.
- **Random Subspace:** [32] consists of selecting K variables randomly at each node and among them choosing the one that minimizes a certain criterion.
- **Random Forest:** [14] which mixes CART without pruning, **bagging** and **Random Subspace**. For each tree, a sample is drawn from the initial sample. At each node, K variables are chosen randomly, and among them, the one that minimizes the **CART** algorithm criterion is taken. The tree is allowed to grow until there is only one element in each node.

2.5.2 Random Forest Algorithm

Random forests are a learning method based on **bagging**. They work by combining multiple random decision trees to improve prediction robustness and accuracy.

- **Bootstrap Sampling:** Generate a large number of samples (denoted $ntree$) from the initial training set by randomly drawing observations (with or without replacement).
- **Tree Construction:** Each bootstrap sample is used to build a decision tree by applying a modified version of the **CART** algorithm at each node, a random subset ($mtry$) of explanatory variables is selected. The best split is determined only among these variables. Trees are fully grown without pruning.
- **Aggregation:** The random forest is obtained by combining all trees ($ntree$).
  - **In regression:** by averaging the predictions.
  - **In classification:** by majority vote.

This process produces a robust and efficient predictive model by combining the strengths of multiple independent trees.

2.5.3 Random forest hyperparameters

The parameter *n_estimators* controls the number of decision trees in the ensemble. More trees generally improve accuracy and stability by reducing variance, though with diminishing returns beyond 100-500 trees, while linearly increasing computation time (default: 100 in scikit-learn) [28]. The parameter *max_depth* determines the maximum depth of individual trees, where `None` (default) allows the nodes to expand until pure or meets the criteria of `min_samples_split`. Lower values create simpler trees that reduce overfitting, while higher values capture more complex patterns, with 3-20 being typical for most datasets [6]. For node splitting, *min_samples_split* specifies the minimum samples required (default: 2), where higher values constrain tree growth and prevent nodes with few samples [31], while *min_samples_leaf* sets the minimum samples at leaf nodes (default: 1), with higher values smoothing predictions through additional regularization [18]. The *max_features* parameter determines how many features are considered at each split, accepting `"auto"` `"sqrt"` ($\sqrt{n_features}$, default for classification), `"log2"` ($\log_2(n_features)$), fractional values, or absolute integers - where lower values increase tree diversity [23]. The bootstrap parameter (default: `True`) controls whether each tree uses random subsets with replacement, while `False` uses the entire dataset [5]. Finally, *oob_score*(default: `False`) enables out-of-bag validation when `bootstrap=True`, providing unbiased estimates without separate validation sets [7].

2.5.4 Advantages and Limitations of Random Forests

2.5.4.1 Advantages

Random Forests offer several significant advantages that make them a powerful tool in both classification and regression tasks. First, they demonstrate robustness and high predictive performance. By aggregating the predictions of multiple decision trees, random forests reduce the risk of overfitting that often affects single trees. This aggregation leads to high accuracy, especially on complex datasets.Another key strength of random forests is their ability to handle high-dimensional data. They can process datasets with a large number of features without the need for dimensionality reduction, as the algorithm randomly selects a subset of variables (mtry) at each node. This also acts as an embedded form of variable selection, improving model interpretability and efficiency.They are also robust to noise and irrelevant variables. Because the model aggregates multiple trees, the influence of outliers and noisy features tends to be minimized. Random forests offer flexibility, being applicable to both categorical and numerical data, and are suitable for a variety of tasks including classification, regression, and feature selection.In addition, random forests are easy to use, with few hyperparameters to tune—mainly the number of trees (ntree) and the number of variables sampled at each split (mtry). They do not require data normalization or scaling, which simplifies preprocessing. A valuable property is their ability to estimate variable importance using measures like mean decrease in impurity or permutation importance, helping to identify key predictors in a dataset.Moreover, random forests exhibit robustness to missing data, being capable of making predictions by imputing missing values based on proximities in the forest. Lastly, because trees in the ensemble are independent, parallelization is possible, making model training faster when using multiple processors or distributed systems.

2.5.4.2 Limitations of Random Forests

Despite their strengths, random forests also come with certain limitations. One major drawback is their computational cost. Training hundreds or thousands of trees can be time-consuming and memory-intensive, particularly for large datasets. Each tree must be stored in memory, which can become a challenge in resource-limited environments.Another limitation is the complexity of the model, often described as a "black box." Unlike simpler models like logistic regression or single decision trees, it is harder to interpret the internal mechanics of a random forest, especially when

making decisions based on hundreds of trees.While random forests generally reduce overfitting, they can still overfit in some situations, especially if the data is very noisy or if too many trees are used without proper validation. Additionally, they may underperform when applied to datasets with simple linear relationships or highly imbalanced class distributions, where more tailored models may be more appropriate.In regression problems, a notable limitation is their inability to extrapolate beyond the range of the training data. Predictions are limited to the convex hull of the observed data. Furthermore, although the number of hyperparameters is limited, their performance still depends on tuning parameters like ntree, mtry, and the splitting criterion, which may require experimentation for optimal results.Lastly, for small datasets, random forests may fail to provide significant improvement over simpler models and could even overfit due to the lack of sufficient data for robust ensemble training.

Conclusion

In conclusion, the integration of random forests in the field of data mining and machine learning offers a powerful and efficient solution for analyzing complex data and extracting actionable knowledge. As a supervised ensemble model, random forests represent a major advancement over classical decision trees, thanks to their ability to reduce overfitting, handle noisy data, and provide robust and accurate predictions.By combining techniques such as bagging and random feature selection, this model capitalizes on tree diversity to improve overall performance. It stands as a tool of choice for various applications, ranging from classification and regression to variable importance assessment.However, although random forests are powerful, they present certain limitations, notably their computational complexity and difficult interpretability. These challenges underscore the importance of using them thoughtfully, considering the specific characteristics of the data and analysis objectives.Thus, random forests illustrate how the evolution of ensemble models has overcome the limitations of traditional approaches, strengthening the central role of machine learning techniques in the data mining process. They perfectly embody the intersection of algorithmic ingenuity and data analysis, paving the way for significant advances in numerous scientific and industrial domains.

3
Multiobjective Optimization

Introduction

Multiobjective optimization addresses complex decision-making scenarios where multiple conflicting objectives must be simultaneously optimized, a fundamental challenge encountered across engineering design, financial planning, and artificial intelligence systems. This paradigm fundamentally differs from single-objective optimization by producing sets of optimal solutions representing trade-offs between competing goals rather than single optimal points. Chapter 3 establishes the theoretical and algorithmic foundations for this field, beginning with formal mathematical definitions of multiobjective optimization problems (MOPs) and their representation in decision and objective spaces. The chapter then introduces the critical concept of Pareto optimality, including rigorous definitions of dominance relations, efficient solutions, and Pareto frontiers that redefine optimality in multi-dimensional spaces. A comprehensive classification of solution methodologies follows, organized by preference articulation timing: a priori methods requiring early preference specification, progressive methods incorporating preferences during optimization, and a posteriori methods generating complete Pareto front approximations. Detailed analysis of representative algorithms includes scalarization techniques like weighted-sum and ε-constraint methods, evolutionary approaches such as SPEA2 and MOEA/D, and culminates with NSGA-II as a benchmark algorithm. The chapter examines NSGA-II's complete workflow from non-dominated sorting to crowding distance calculations, providing both theoretical justification and practical implementation insights for this widely adopted technique in Pareto-optimal solution generation.

3.1 Multiobjective Optimization Problem

A multiobjective optimization problem (MOP) consists of simultaneously optimizing several objective functions $f_1, f_2, \ldots, f_p$, $p \geq 2$ over a feasible domain (set) S called the *decision space*, defined by inequalities $g_j(\mathbf{x}) \leq 0$, $j = 1, 2, \ldots, m$.

Mathematically, this problem can be written as follows:

$$\begin{cases} \min \text{ or } \max (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_p(\mathbf{x}))^T, & p \geq 2 \\ g_j(\mathbf{x}) \leq 0, & j = 1, 2, \ldots, m \end{cases} \tag{MOP}$$

Such that:

- $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ is the vector of n decision variables.
- Let $F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_p(\mathbf{x}))^T$ be the vector of $p \geq 2$ objective functions to be optimized.

  Where:

  $$f_i : \mathbb{R}^n \to \mathbb{R}, \quad i = 1, 2, \ldots, p$$

  $$\mathbf{x} \mapsto f_i(\mathbf{x})$$

- Denote by $S = \{\mathbf{x} \in \mathbb{R}^n : g_j(\mathbf{x}) \leq 0, j = 1, \ldots, m\}$ the decision set.

Consider the mapping:

$$F : S \to \mathbb{R}^p$$

$$\mathbf{x} \mapsto F(\mathbf{x})$$

The set $Z = F(S)$ is called the *criterion space* or *objective space*.

Figure 3.1 shows the decision space S with 3 decision variables and the objective space with 2 objectives.

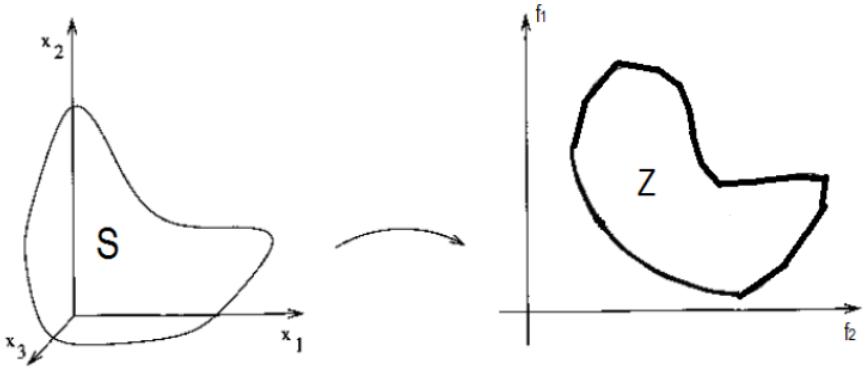

Figure 3.1: Decision space and objective space representation

Remark 1.

1. If all functions $f_i, i = 1, 2, \ldots, p$ and $g_j, j = 1, 2, \ldots, m$ are linear, we call this a *Multiobjective Linear Programming* problem, denoted **(MOLP)**.

2. If any of these functions is nonlinear, we refer to it as *Multiobjective Nonlinear Programming*, denoted **(MONLP)**.

3. If one or more decision variables are discrete (integer-valued), and the functions f_i, $i = 1, 2, \ldots, p$ and g_j, $j = 1, 2, \ldots, m$ are linear, we call this *Multiobjective Integer Linear Programming*, denoted **(MOILP)**.

3.1.1 Notions of Optimality and Dominance

In single-objective optimization, the optimal solution may be unique or multiple but the objective function value is unique. As soon as optimization becomes multiobjective, the notion of an optimal solution becomes meaningless. We instead speak of *efficient solutions* or *Pareto-optimal solutions* or *non-dominated solutions*.

3.1.2 Efficient Solution and Dominance Relation

At the end of the 19th century, economist Vilfredo Pareto formulated the concept of optimality bearing his name (*Pareto optimality*), which constitutes the origins of research in multiobjective optimization where the notion of optimality in single-objective optimization is meaningless. Indeed, there exists no feasible solution $\mathbf{x}$ that can decrease one objective without simultaneously increasing at least one other objective.[29]

However, in most cases, the Pareto optimum is not a single solution but a set of "best compromise" solutions called *efficient solutions* or *non-dominated solutions*.

When we wish to solve a multiobjective optimization problem, it involves determining a set of efficient solutions, i.e., the Pareto-optimal set or the set of non-dominated solutions.

Without loss of generality, consider a minimization-type multiobjective optimization problem written as follows:

$$\begin{cases} \text{"min" } F(\mathbf{x}) = \big(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_p(\mathbf{x})\big)^T, & p \geq 2 \\ g_j(\mathbf{x}) \leq 0, & j = 1, 2, \ldots, m \end{cases} \tag{MOP}$$

Definition 1 (Efficient Solution).

1. $\hat{\mathbf{x}} \in S$ is said to be an *efficient solution* or *Pareto-optimal solution* if and only if there does not exist another solution $\mathbf{x} \in S$ such that $f_i(\mathbf{x}) \leq f_i(\hat{\mathbf{x}})$ for all $i = 1, 2, \ldots, p$ with at least one strict inequality. The criterion vector $\hat{\mathbf{z}} = F(\hat{\mathbf{x}})$ is called a *non-dominated vector* or *non-dominated solution*.

2. $\hat{\mathbf{x}} \in S$ is said to be a *weakly efficient solution* or *weakly Pareto-optimal solution* if and only if there does not exist another solution $\mathbf{x} \in S$ such that $f_i(\mathbf{x}) < f_i(\hat{\mathbf{x}})$ for all $i = 1, 2, \ldots, p$. The criterion vector $\hat{\mathbf{z}} = F(\hat{\mathbf{x}})$ is called a *weakly non-dominated vector* or *weakly non-dominated solution*.

3. $\hat{\mathbf{x}} \in S$ is said to be *strongly efficient* if and only if there does not exist another solution $\mathbf{x} \in S$ such that $\mathbf{x} \neq \hat{\mathbf{x}}$ and $f_i(\mathbf{x}) \leq f_i(\hat{\mathbf{x}})$ for all $i = 1, 2, \ldots, p$. The criterion vector $\hat{\mathbf{z}} = F(\hat{\mathbf{x}})$ is called *strongly non-dominated*.

Definition 2 (Dominance Relation) Let $\mathbf{z}^1, \mathbf{z}^2 \in \mathbb{R}^p$. We say that vector $\mathbf{z}^1$ *dominates* vector $\mathbf{z}^2$ if:

- *Weak dominance*: $\mathbf{z}_i^1 \leq \mathbf{z}_i^2$ for all $i = 1, 2, \ldots, p$
- *Strong dominance*: There exists $j \in \{1, 2, \ldots, p\}$ such that $\mathbf{z}_j^1 < \mathbf{z}_j^2$

Proposition 1

Solutions that dominate others but are not dominated among themselves are called *non-dominated solutions* or *Pareto-optimal solutions*.

In other words, a feasible solution $\hat{\mathbf{x}} \in S$ is said to be efficient if its image $\hat{\mathbf{z}}$ under the mapping F (where $\hat{\mathbf{z}} = F(\hat{\mathbf{x}})$) is not dominated by any other solution.

In the remainder of this book, we denote by:

- X_E: The set of efficient solutions (in decision space)
- Z_{ND}: The set of non-dominated solutions (in objective space)

with the relationship $Z_{ND} = F(X_E)$.

Indeed, non-dominated solutions are sometimes called *non-dominated actions*. However, selecting one solution from the found non-dominated solutions constitutes another significant challenge called *multi-criteria decision aiding theory*, where the decision maker's goal is to model their choices or preferences.

3.1.3 Pareto frontier

Pareto efficiency or Pareto optimality is a situation where no individual or preference criterion can be better off without making at least one individual or preference criterion worse off. The concept is named from Vilfredo Pareto, an Italian engineer and economist. The concept of Pareto optimality was initially used in studies of economic efficiency and income distribution. The Pareto frontier is the set of all Pareto efficient allocations shown on the graph. The Pareto frontier is also known as Pareto front and Pareto set. Finding Pareto frontiers is particularly useful in engineering. By presenting all the potentially optimal solutions graphically, designers could focus on the tradeoffs within a constrained set of parameters, rather than considering a full range of solutions.

3.2 Classification of Solution Methods

According to Y. Collette and P. Siarry [10], multi-objective optimization methods can be classified into five main groups:

1. **Scalar methods**: Techniques that transform multiple objectives into a single objective function
2. **Interactive methods**: Approaches where decision-makers progressively refine preferences during optimization
3. **Fuzzy methods**: Methods using fuzzy logic to handle imprecise objectives
4. **Metahueristic-based methods**: Techniques employing evolutionary algorithms, swarm intelligence, etc.
5. **Decision support methods**: Frameworks facilitating solution selection after optimization

These five groups can be further organized into three principal families based on preference articulation timing:

3.2.1 Priori Preference Methods

The decision-maker specifies preferences *before* optimization begins. This family primarily includes aggregation methods where multiple objectives are combined into a single objective function. For example:

$$F_{\text{agg}}(\mathbf{x}) = \sum_{i=1}^{k} w_i f_i(\mathbf{x}), \quad \text{where } \sum w_i = 1$$

It characterised by: Single solution generation , Requires accurate preference knowledge and Computationally efficient.

3.2.2 Progressive Preference Methods

The decision-maker refines preferences *during* the optimization process. This family includes interactive methods where solution trajectories are adjusted based on ongoing feedback.

Characteristics:

- Iterative preference refinement
- Human-in-the-loop optimization
- Balanced exploration-exploitation

3.2.3 Posteriori Preference Methods

The decision-maker selects solutions *after* optimization completes. These methods generate an approximation of the Pareto front, allowing solution choice from the compromise surface.

Hybrid Classification

Some methods transcend these categories. For instance:

- Apriori method with random preferences can generate multiple solutions for posteriori selection
- Interactive methods can incorporate elements from fuzzy approaches
- Metaheuristics can implement any preference articulation strategy

The classification framework can be summarized as:

Table 3.1: Classification of multi-objective optimization methods

Preference Family	Method Groups	Key Features
Priori	Scalar, Fuzzy	Single solution, early commitment
Progressive	Interactive	Adaptive preferences, human guidance
Posteriori	Metaheuristic, Decision support	Pareto front, flexible choice
Hybrid	Combinations	Mixed strategies

This classification provides a structured framework for understanding the diverse landscape of multi-objective optimization techniques, acknowledging that modern approaches often blend elements from multiple categories.

3.3 A Priori Methods

These methods require the decision-maker to specify preferences *before* solving the problem.

3.3.1 Weighted Sum Method

A classical scalarization technique where a weight w_i is assigned to each objective function $f_i(x)$, converting the problem into a single-objective one:

$$\min_x \sum_{i=1}^{n} w_i f_i(x) \tag{3.1}$$

Use Case: Portfolio optimization — balancing return and risk.
Limitation: This method fails to detect Pareto-optimal points in non-convex regions of the Pareto front.

3.3.2 Scalarization-Based Methods

These methods transform a multi-objective optimization problem into one or more scalar subproblems, often solved sequentially.

3.3.3 ε-Constraint Method

One objective is optimized, while the others are converted into constraints. Let $f_1(x)$ be the primary objective:

$$\begin{aligned} &\min_x \quad f_1(x) \\ &\text{s.t.} \quad f_i(x) \leq \varepsilon_i, \quad \forall i \neq 1 \end{aligned} \tag{3.2}$$

Use Case: Environmental policy — minimize cost while constraining emissions and pollutants.

3.3.4 KKT Conditions in Multi-Objective Optimization

For a multi-objective optimization problem:

$$\min_x \left(f_1(x), f_2(x), \ldots, f_k(x)\right)$$

A point $x^* \in \mathbb{R}^n$ is *Pareto optimal* if there exists a set of non-negative scalars $\lambda_1, \ldots, \lambda_k$, such that:

$$\sum_{i=1}^{k} \lambda_i \nabla f_i(x^*) = 0, \quad \lambda_i \geq 0, \quad \sum_{i=1}^{k} \lambda_i = 1 \tag{3.3}$$

These are the **Karush–Kuhn–Tucker (KKT)** necessary conditions for Pareto optimality when all f_i are differentiable and there are no constraints. **Interpretation:** The solution lies where no descent direction exists that simultaneously improves all objectives — i.e., trade-offs exist between them.

3.4 A Posteriori Methods

In **a posteriori methods**, the Pareto front is first approximated without any prior knowledge of the decision-maker's preferences. After obtaining a set of non-dominated solutions, the final selection is made based on preference articulation.

3.4.1 Evolutionary Algorithms (EAs)

Evolutionary algorithms are well-suited for multi-objective optimization due to their ability to explore multiple solutions in parallel. They rely on principles of natural evolution, such as selection, crossover, and mutation.

Advantages:

- Provide a set of diverse Pareto-optimal solutions in a single run.
- Well-suited for non-linear, non-convex, and noisy problems.
- Do not require gradients or convexity assumptions.

Limitations:

- Typically slower than scalarization methods for low-dimensional problems.
- Require careful parameter tuning (e.g., population size, mutation rate).

3.4.2 Popular Multi-Objective Evolutionary Algorithms (MOEAs)

- **NSGA-II** (Non-dominated Sorting Genetic Algorithm II):
  Utilizes fast non-dominated sorting and crowding distance to ensure both convergence and diversity. It is one of the most widely used MOEAs.
- **SPEA2** (Strength Pareto Evolutionary Algorithm 2):
  Maintains an external archive and uses a fitness assignment strategy based on strength and density. It ensures elitism and diversity simultaneously.
- **MOEA/D** (Multi-Objective Evolutionary Algorithm based on Decomposition):
  Decomposes a multi-objective problem into multiple scalar sub-problems and solves them simultaneously, promoting good spread over the Pareto front.
- **NSGA-III**:
  An extension of NSGA-II designed to handle many-objective problems (typically more than three objectives) using reference points to guide selection.

Algorithm Table:

Algorithm	Description
NSGA-II	Fast non-dominated sorting with crowding distance to maintain diversity and ensure elitism.
SPEA2	Uses strength and density to assign fitness and maintains an archive of elite solutions.
MOEA/D	Decomposes the multi-objective problem into scalar subproblems and solves them cooperatively.
NSGA-III	Incorporates reference points for solving many-objective problems, extending NSGA-II.

Table 3.2: Popular Multi-Objective Evolutionary Algorithms (MOEAs)

3.5 NSGA-II

3.5.1 Definition of NSGA-II

NSGA-II (Non-dominated Sorting Genetic Algorithm II) is a popular **Multi-Objective Evolutionary Algorithm (MOEA)**designed to find a diverse set of high-quality solutions to problems involving two or more conflicting objectives. It is an improvement over the original NSGA and is widely recognized for its efficiency and robustness.It designed to approximate the Pareto front in multi-objective optimization problems. It improves upon the original NSGA by introducing fast non-dominated sorting, elitism, and crowding distance mechanisms.

NSGA: The First Version (1994) In 1994, N. Srinivas and Kalyanmoy Deb introduced the Non-dominated Sorting Genetic Algorithm (NSGA). It sorted the population into different Pareto fronts and applied fitness sharing to maintain diversity. **Kalyanmoy Deb**, with co-authors **Amrit Pratap, Sameer Agarwal, and T. Meyarivan**, proposed NSGA-II in their highly cited paper[12]:

"A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II" IEEE Transactions on Evolutionary Computation, 2002

NSGA-II became the de facto standard in multi-objective optimization, used in fields ranging from engineering to AI.

3.5.2 General Description of NSGA-II

The population is initialized as usual. Once the population in initialized the population is sorted based on non-domination into each front. The first front being completely non-dominant set in the current population and the second front being dominated by the individuals in the first front only and the front goes so on. Each individual in the each front are assigned rank (fitness) values or based on front in which they belong to. Individuals in first front are given a fitness value of 1 and individuals in second are assigned fitness value as 2 and so on. In addition to fitness value a new parameter called crowding distance is calculated for each individual. The crowding distance is a measure of how close an individual is to its neighbors. Large average crowding distance will result in better diversity in the population. Parents are selected from the population by using binary tournament selection based on the rank and crowding distance. An individual is selected in the rank is lesser than the other or if crowding distance is greater than the other 1 . The selected population generates offsprings from crossover and mutation operators, which will be discussed in detail in a later section. The population with the current population and current offsprings is sorted again based on non-domination and only the best N individuals are selected, where N is the population size. The selection is based on rank and the on crowding distance on the last front. [33]

3.5.3 NSGA-II Algorithm

NSGA-II Algorithm

Input: Population size N, Max generations T
Output: Approximation of Pareto front P_T

1. Initialize population P_0 of size N
2. Evaluate objective functions for all individuals in P_0
3. **for** $t = 0$ **to** $T - 1$:
   (a) Perform binary tournament selection on P_t
   (b) Apply crossover and mutation to create offspring Q_t
   (c) Evaluate objective functions for Q_t
   (d) Combine populations: $R_t = P_t \cup Q_t$ (size $|R_t| = 2N$)
   (e) Apply fast non-dominated sorting on $R_t \rightarrow$ fronts $F_1, F_2, \ldots$
   (f) Compute crowding distance for each front
   (g) $P_{t+1} \leftarrow \emptyset$, $i \leftarrow 1$
   (h) **while** $|P_{t+1}| + |F_i| \leq N$: (fill with complete fronts)
      - Add F_i to P_{t+1}
      - $i \leftarrow i + 1$
   (i) Sort F_i by crowding distance ($\searrow$)
   (j) Add first $(N - |P_{t+1}|)$ individuals from F_i to P_{t+1}
4. **return** P_T (final approximation set)

3.5.4 Working of NSGA-II

The Non-Dominated Sorting Genetic Algorithm II (NSGA-II) is a widely-used algorithm for solving multi-objective optimization problems. The following steps outline its working mechanism:

Step 1: Initialization of Population

•Randomly generate an initial population of individuals.
•Each individual is represented by a chromosome, a vector of decision variables.

Step 2: Evaluation of Fitness Functions

•Evaluate the fitness of each individual based on multiple objective functions.
•Determine the objective values to be minimized or maximized for each individual.

Step 3: Non-Dominated Sorting Sort the population into different non-domination levels (fronts).

•The first front consists of individuals that are not dominated by any other individuals.
•The second front consists of individuals dominated only by those in the first front, and so on.

Step 4: Calculation of Crowding Distance

Calculate the crowding distance for each individual within each front.
•The crowding distance measures the density of individuals surrounding a particular individual in the objective space.
•Individuals with larger crowding distances are preferred to promote diversity.

Step 5: Selection of Parents for Crossover

•Use binary tournament selection to choose parents for crossover: Randomly select two individuals.
•Compare their ranks (non-domination levels).
•Select the individual with the better rank.
•If both belong to the same front, select the one with the larger crowding distance.

Step 6: Application of Crossover and Mutation Operators

Crossover:

•Perform crossover on selected parents to generate offspring.
•Exchange portions of chromosomes between parents to create new individuals.

Mutation:
•Apply mutation to offspring by introducing small random changes in their chromosomes.
•Maintain genetic diversity and explore new areas of the solution space.
Step 7: Creation of the Next Generation
•Combine the parent and offspring populations to form an intermediate population.
•Perform non-dominated sorting on the combined population.
•Select the best individuals based on non-domination rank and crowding distance to form the next generation.
Step 8: Termination Criteria
•Repeat the process of selection, crossover, mutation, and generation update until a termination criterion is met.
•Common termination criteria include a fixed number of generations, a convergence threshold, or a maximum computational budget.

The final population contains the Pareto-optimal solutions representing the best trade-offs among the objectives.

3.5.5 Core Functions Explained

Fitness Evaluation

Each individual x is evaluated on m objective functions:

$$f(x) = [f_1(x), f_2(x), \ldots, f_m(x)]$$

This determines dominance relations between individuals.

Non-Dominated Sorting

Classifies the population into levels $F_1, F_2, \ldots$:

- F_1: Non-dominated individuals
- F_2: Dominated only by individuals in F_1, and so on

Crowding Distance

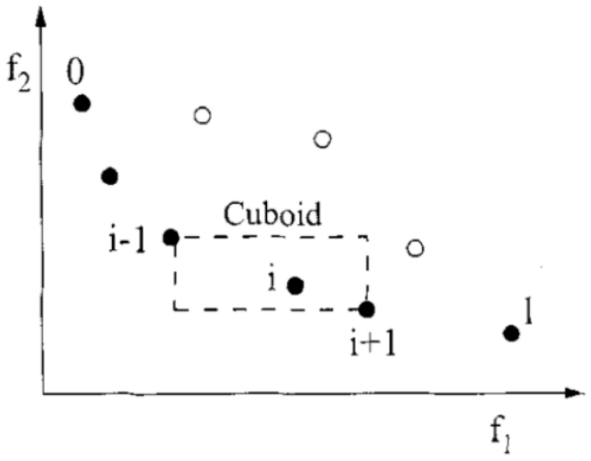

Figure 3.2: Crowding Distance calculation

Used to maintain solution diversity[4][12]. For each individual i:

$$CD_i = \sum_{k=1}^{m} \frac{f_k^{i+1} - f_k^{i-1}}{f_k^{\max} - f_k^{\min}}$$

Larger distance = less crowded = preferred during selection.

Selection

Binary tournament selection is used. Among two individuals:

- The one with lower rank (better front) wins
- If ranks are equal, the one with larger crowding distance wins

Crossover and Mutation

- **Simulated Binary Crossover (SBX)** generates offspring near the parents
- **Polynomial Mutation** perturbs solutions slightly to explore neighborhood

Procedure

1. Form combined population $R_t = P_t \cup Q_t$ of size $2N$
2. Perform non-dominated sorting on R_t to identify fronts $(F_1, F_2, \dots)$
3. Initialize new population $P_{t+1} = \emptyset$ and $i = 1$
4. While $|P_{t+1}| + |F_i| \leq N$:
   - Add all solutions from F_i to P_{t+1}
   - $i = i + 1$
5. Sort remaining solutions in F_i using crowded-comparison operator
6. Select best $(N - |P_{t+1}|)$ solutions from F_i
7. Create Q_{t+1} from P_{t+1} using:
   - Binary tournament selection (based on crowded-comparison)
   - Crossover and mutation[4].

Key Components

- **Non-dominated sorting**: Hierarchical ranking of solutions
- **Crowded-comparison operator** $(\prec_n)$:

$$i \prec_n j \quad \text{if} \quad (i_{\text{rank}} < j_{\text{rank}}) \quad \text{or} \quad (i_{\text{rank}} = j_{\text{rank}} \quad \text{and} \quad i_{\text{distance}} > j_{\text{distance}})$$

- **Crowding distance**: Density estimation in objective space
- **Elitism**: Maintained through combined population

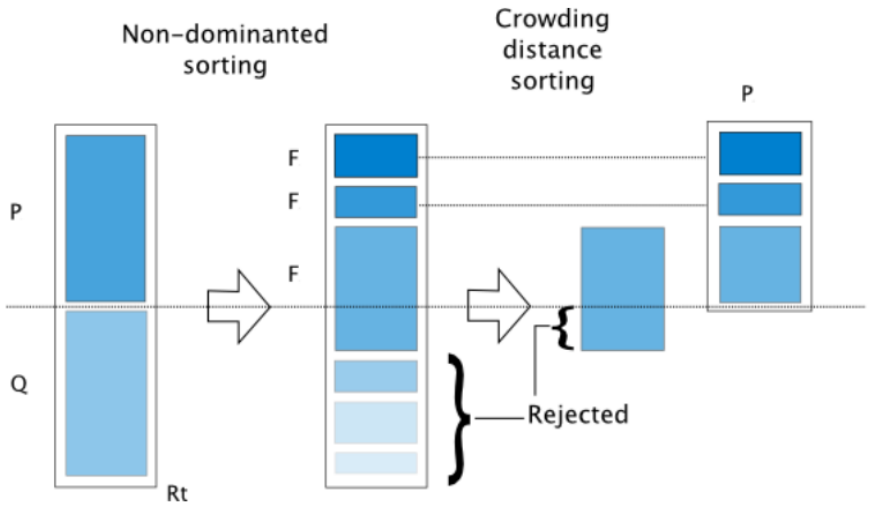

Figure 3.3: Procedure NSGA-II

Optimizations

- Early termination of non-dominated sorting when $|P_{t+1}| \geq N$
- Parameter-less diversity maintenance via crowding distance

Advantages

- Explicit elitism mechanism
- No niching parameters required
- Computationally efficient compared to NSGA

3.6 FON Problem

The FON (Fonseca–Fleming) problem is a well-known benchmark in multi-objective optimization. It is defined as follows:

Problem Formulation

Given a decision vector:

$$\mathbf{x} = (x_1, x_2, x_3) \in [-4, 4]^3,$$

the objective is to **maximize** the following two functions:

$$f_1(\mathbf{x}) = \exp\left(-\sum_{i=1}^{3}\left(x_i - \frac{1}{\sqrt{3}}\right)^2\right),$$
$$f_2(\mathbf{x}) = \exp\left(-\sum_{i=1}^{3}\left(x_i + \frac{1}{\sqrt{3}}\right)^2\right).$$

Explanation

- The problem has **two conflicting objectives** to be maximized.
- The function $f_1(\mathbf{x})$ is maximized when $\mathbf{x} \approx \left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right)$.
- The function $f_2(\mathbf{x})$ is maximized when $\mathbf{x} \approx \left(-\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}\right)$.
- Because these two ideal solutions are in conflict, it is not possible to simultaneously maximize both objectives.
- The goal is to find the **Pareto front**, which is the set of **non-dominated solutions** that represent the best trade-offs between f_1 and f_2.

Optimization Approach

To solve this problem, we apply a multi-objective evolutionary algorithm such as **NSGA-II**, which evolves a population of candidate solutions by applying:

- **Selection (e.g., NSGA-II based)**
- **Crossover**
- **Mutation**

At each generation, solutions are evaluated and ranked using Pareto dominance and diversity preservation techniques [20].

```
Out[14]: <matplotlib.legend.Legend at 0x7f26ce142c18>
```

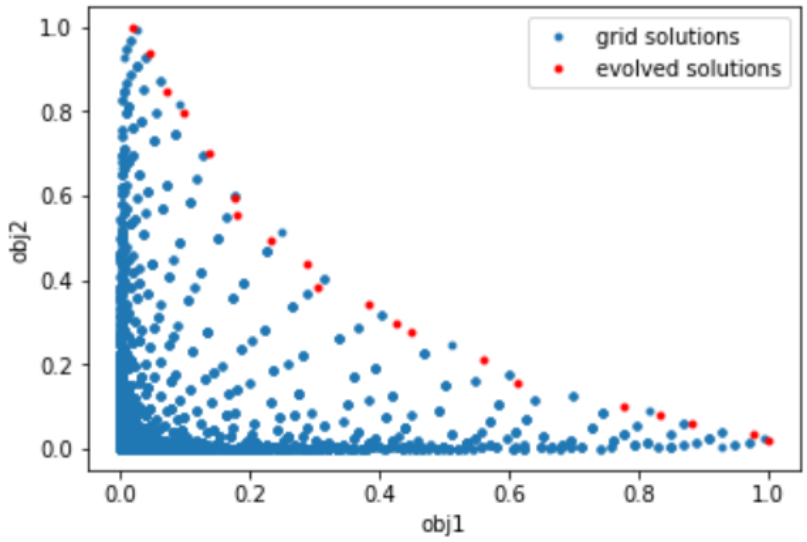

Figure 3.4: Pareto Front

Conclusion

This chapter has systematically developed the conceptual framework for multiobjective optimization, beginning with formal mathematical definitions of multiobjective optimization problems (MOPs) as simultaneous optimization of multiple objectives over constrained decision spaces. The fundamental shift from single-objective optimization was established through Pareto optimality concepts, where dominance relations define efficient solutions forming the Pareto frontier - the set of optimal trade-offs between competing objectives. A comprehensive classification of solution methodologies demonstrated how approaches diverge based on preference articulation timing: a priori methods like weighted-sum scalarization require early preference specification; progressive methods incorporate preferences during optimization; while a posteriori methods like evolutionary algorithms generate complete Pareto front approximations for post-optimization selection.

Detailed examination of evolutionary algorithms revealed their particular advantage in handling non-convex problems, with NSGA-II emerging as a benchmark technique due to its balanced approach to solution quality, diversity maintenance, and computational efficiency. The algorithm's core innovations - fast non-dominated sorting, crowding distance metrics, and elitist preservation - were analyzed through complete workflow descriptions from fitness evaluation to genetic operators. As multiobjective problems grow increasingly complex in domains like sustainable engineering and machine learning, future advancements will likely focus on scaling these techniques to many-objective problems, hybridizing evolutionary approaches with machine learning surrogates, and enhancing real-time decision-maker interaction. The foundational principles and algorithmic toolkit presented in this chapter provide essential capabilities for navigating the inherent trade-offs in optimization problems where conflicting objectives must be balanced.

4

Experimental Implementation and Evaluation

Introduction

In this chapter, we present the practical implementation of the methodology proposed for credit card fraud detection, combining machine learning with evolutionary optimization techniques. After establishing the theoretical background in the previous chapters, this section focuses on applying the Random Forest Classifier (RFC), a widely used ensemble learning method, to detect fraudulent transactions in a highly imbalanced dataset.

The core objective of this chapter is to enhance the performance of fraud detection by optimizing the hyperparameters of the RFC using a Genetic Algorithm (GA). The GA is employed to efficiently explore the hyperparameter space and improve the model's ability to balance competing performance metrics—particularly precision and recall, which are critical in imbalanced classification tasks.

We begin by describing the dataset and preprocessing steps, including normalization, data splitting, and class distribution analysis. The next sections detail the training procedure of the Random Forest model, its evaluation using standard classification metrics, and the interpretation of results via the confusion matrix. We then integrate the GA to fine-tune hyperparameters, followed by a comparative evaluation between the optimized and non-optimized models.

This experimental phase culminates in an in-depth discussion of the results, highlighting the improvements achieved through optimization and the challenges encountered during implementation. Despite data imbalance and computational constraints, our findings confirm that the hybrid RFC-GA approach significantly improves recall while maintaining acceptable precision, thus meeting the main objective of enhancing fraud detection capabilities.

The work presented in this chapter demonstrates the practical relevance of combining machine learning and multi-objective optimization in developing robust fraud detection systems for real-world applications.

4.1 Problematic

A central difficulty in fraud detection is dealing with the *imbalanced nature of the dataset* — where fraudulent transactions represent a very small minority. Traditional models may achieve high overall accuracy simply by predicting the majority class, but fail to detect actual frauds. Hence, **precision** and **recall** become critical performance metrics. However, optimizing both simultaneously presents a trade-off: increasing recall (catching more frauds) often reduces precision (more false alarms), and vice versa. This trade-off is particularly challenging in credit card fraud detection, where missing a fraud can have serious consequences, but overwhelming users with false positives also degrades system usability.

4.2 Data Preprocessing

4.2.1 Data Collection

The dataset used in this project consists of real-world credit card transactions collected over a two-day period by European cardholders in 2013. It includes 284,807 transactions, among which only 492 are labeled as fraudulent. This results in a highly imbalanced dataset, with fraud cases accounting for approximately 0.172% of the data. Due to confidentiality, most features are anonymized and transformed using Principal Component Analysis (PCA). Only the features `Time`, `Amount`, and `Class` (indicating fraud) are available in their original form.

Dataset loaded from Google Drive:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864	1.014480	-0.509348	1.436807	0.250034	0.943651	0.823731	0.77	0
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.012463	-1.016226	-0.606624	-0.395255	0.068472	-0.053527	24.79	0
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.037501	0.640134	0.265745	-0.087371	0.004455	-0.026561	67.88	0
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.163298	0.123205	-0.569159	0.546668	0.108821	0.104533	10.00	0
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.376777	0.008797	-0.473649	-0.818267	-0.002415	0.013649	217.00	0

284807 rows × 31 columns

Figure 4.1: data set

4.2.2 Data Cleaning

Before training any machine learning models, several preprocessing steps were carried out:

Missing Values: The dataset was checked for missing or null values. Fortunately, no such values were found.

```
DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

Figure 4.2: check missing values

Feature Scaling: The `Time` and `Amount` features for machine learning models by scaling them. 'Amount' is scaled using `RobustScaler` to handle potential outliers, and 'Time' is scaled to a 0-1 range using Min-Max scaling.

4.2.3 Data Visualization

To better understand the structure and characteristics of the data, the following visualizations were performed:

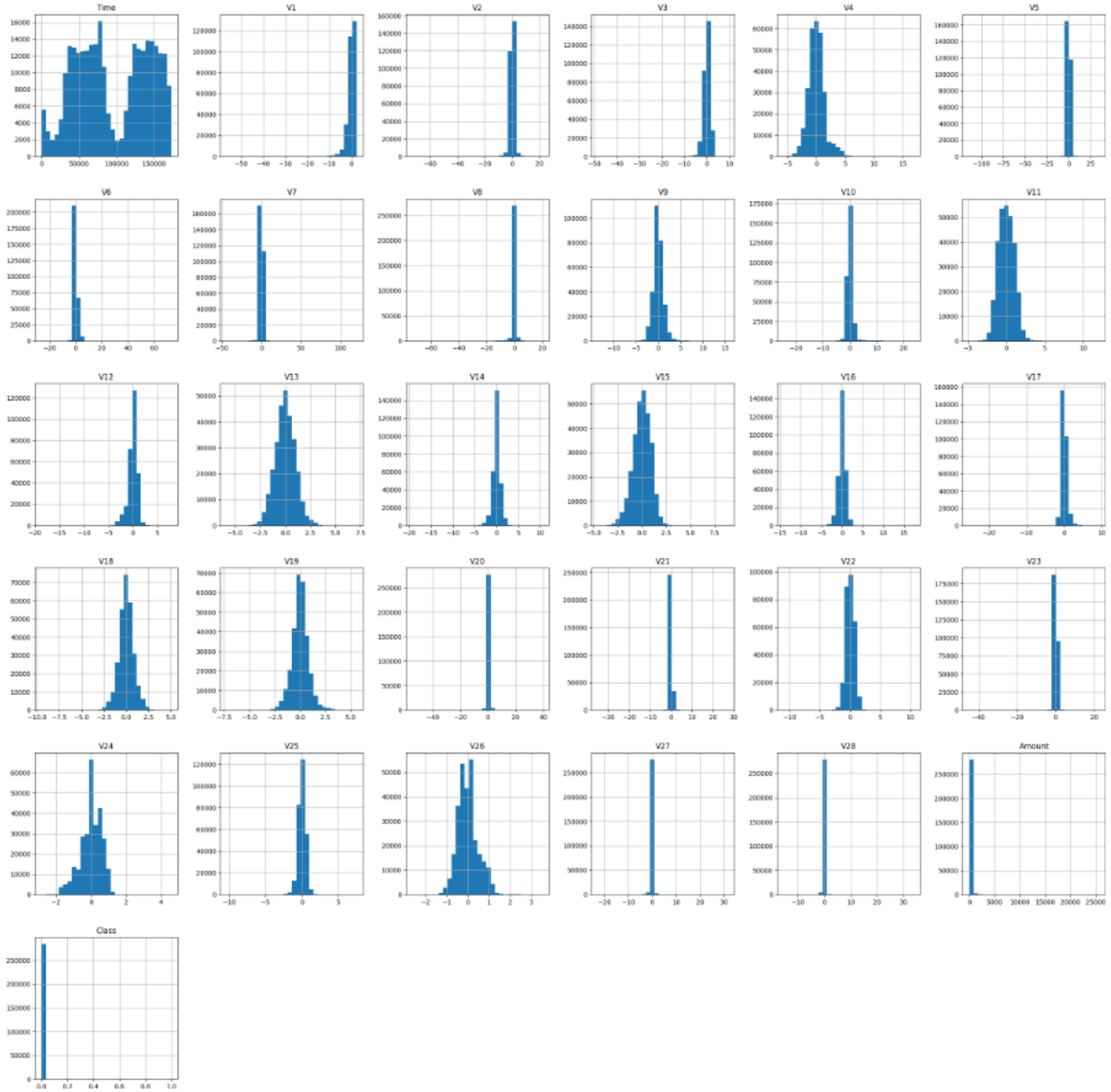

Figure 4.3: Data visualisation

The output shows a grid of histograms, one for each numerical column in the df DataFrame. By examining these histograms, you can gain insights into:

- *Distribution Shape:* Whether a feature's distribution is normal, skewed, uniform, etc.
- *Central Tendency:* The approximate center of the data.
- *Spread:* The range and variability of the data.
- *Presence of Outliers:* Values that are far from the majority of the data.
- *Modality:* Whether the distribution has one peak (unimodal) or multiple peaks (multimodal).

This visualization helps in understanding the characteristics of each feature and can inform decisions about data preprocessing or feature engineering.

4.3 Train-Test-Validation Split and Class Distribution

In supervised machine learning, data splitting is crucial to evaluate the generalization performance of a model,we split DataFrame into three parts: train, test, and val for training, testing, and validation of a machine learning model.

Data Splitting:

- `train = new_df[:240000]`: first 240,000 rows used to train the model.
- `test = new_df[240000:262000]`: next 22,000 rows used to assess the model's performance on unseen data.
- `val = new_df[262000:]`: remaining rows used to tune hyperparameters and evaluate the model during training.

```
(Class
 0    239589
 1       411
 Name: count, dtype: int64,
 Class
 0    21955
 1       45
 Name: count, dtype: int64,
 Class
 0    22771
 1       36
 Name: count, dtype: int64)
```

Figure 4.4: spliting

The output shows the number of instances for each class (0 for non-fraud, 1 for fraud) in the training, testing, and validation sets. This is important to check if the class distribution is similar across the splits, especially in imbalanced datasets like this one (where there are many more non-fraudulent transactions than fraudulent ones).

Purpose: Verify that the proportion of class 0 (non-fraud) and class 1 (fraud) remains consistent across splits, which is crucial for model evaluation on imbalanced data.

4.4 classify transaction

After ensuring a consistent class distribution across the training, validation, and testing datasets, the next step is to apply a classification model to detect fraudulent transactions. In this work, we use the Random Forest algorithm, which is well-suited for handling high-dimensional and imbalanced datasets. The following section presents how the classification is conducted and evaluates the model's performance using standard metrics.

4.4.1 Evaluation of the Random Forest Model

The Random Forest algorithm, a popular ensemble learning technique, is employed to classify financial transactions as either fraudulent or not. This section details the steps followed for model evaluation, from library imports to generating performance metrics.

1. Import Libraries

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
```

- **RandomForestClassifier**: Implements the Random Forest algorithm.
- **classification_report**: Generates a summary of classification metrics.

2. Initialize Random Forest

```
rf = RandomForestClassifier(n_jobs=-1)
```

- **n_jobs=-1**: Utilizes all CPU cores for faster training.

3. Train the Model

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

- **x_train**: Feature matrix of the training set.
- **y_train**: Target labels (fraud or not fraud).
- Trains the random forest using multiple decision trees.

4. Evaluate on Validation Set

```
y_pred = rf.predict(x_val)
print(classification_report(y_val, y_pred, target_names=['Not␣Fraud
    ', 'Fraud']))
```

- **rf.predict(x_val)**: Predicts fraud probabilities on validation features.
- **classification_report**: Outputs key metrics comparing predictions to actual labels.

5. Report Interpretation

The report includes:

- **Precision**: Proportion of predicted frauds that are actual frauds.
- **Recall**: Proportion of actual frauds correctly predicted.
- **F1-score**: Harmonic mean of precision and recall.
- **Support**: Number of true instances for each class.

Fraud Detection Classification Definitions

Notation:
Let y be the **actual class** of a transaction:

- $y = 1$: fraudulent transaction
- $y = 0$: legitimate transaction

Let $\hat{y}$ be the **predicted class**:

- $\hat{y} = 1$: model predicts fraud
- $\hat{y} = 0$: model predicts legitimate

Classification Outcomes:

- **True Positive (TP)**: $y = 1$ and $\hat{y} = 1$
  The model correctly detects a fraudulent transaction.
- **True Negative (TN)**: $y = 0$ and $\hat{y} = 0$
  The model correctly detects a legitimate transaction.
- **False Positive (FP)**: $y = 0$ and $\hat{y} = 1$
  The model incorrectly classifies a legitimate transaction as fraud (Type I error).
- **False Negative (FN)**: $y = 1$ and $\hat{y} = 0$
  The model incorrectly classifies a fraudulent transaction as legitimate (Type II error).

Evaluation Metrics

Accuracy: Accuracy measures the proportion of correctly classified instances (both true positives and true negatives) among all the samples. It indicates how often the model predicts the correct output.

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

Precision: Precision is the proportion of correctly predicted positive observations (true positives) out of all predicted positive cases.

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

Recall: Recall (also called sensitivity) is the proportion of correctly predicted positive observations out of all actual positive cases.

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

F1-Score: F1-score is the harmonic mean of precision and recall. It is most useful when you need to balance precision and recall.

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

4.4.2 Confusion Matrix

The confusion matrix is a visual representation of a classifier's performance, displaying the number of correct and incorrect predictions for each class. It is especially useful in evaluating binary classification models, such as fraud detection.

By examining the values in the confusion matrix, we can gain deeper insights into the model's strengths and weaknesses[13].

This matrix allows us to calculate key performance metrics such as:

- **Precision:** How often the predicted frauds are actually fraudulent.
- **Recall:** How many of the actual frauds the model is able to detect.

These insights help us evaluate whether the model is better at avoiding false alarms or at catching fraudulent behavior, which is crucial in imbalanced classification problems such as credit card fraud detection.

4.4.3 Results

A. Training Data Ouput (Random Forest)

The accuracy, recall, precision and F1-score of training data is 100%.

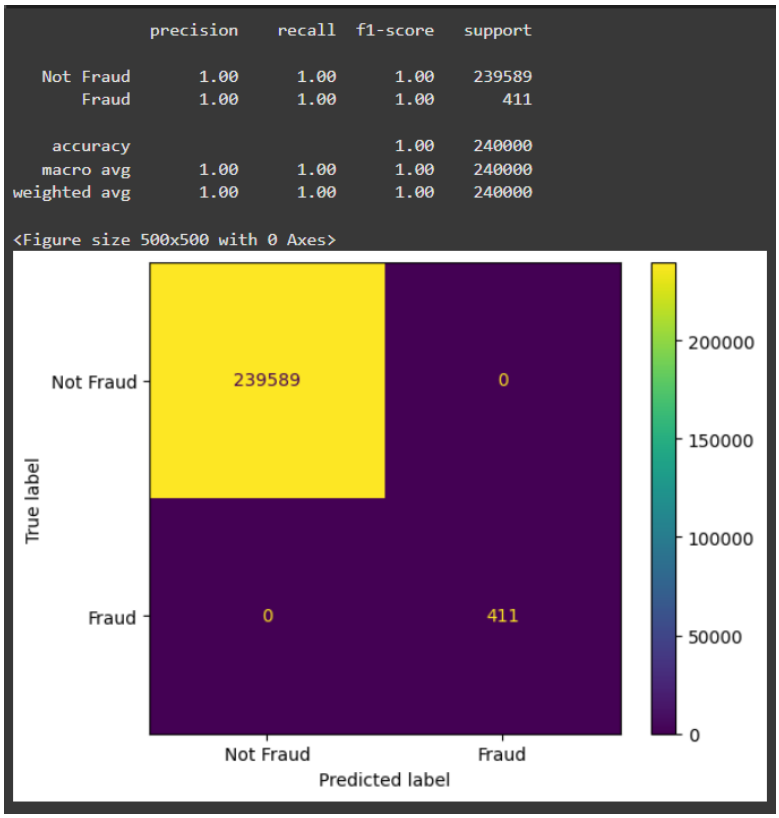

Figure 4.5: confusion matrix

Interpretation:

These results indicate that the Random Forest model achieved perfect performance on the training data. It correctly identified every single non-fraudulent transaction and every single fraudulent transaction in the training set.

While this might seem ideal, it's important to consider this in the context of model evaluation:

- **Expected for Tree-Based Models:** Tree-based models like Random Forests can easily achieve perfect or near-perfect scores on the training data, especially with sufficient depth and complexity. They essentially "memorize" the training examples.
- **Risk of Overfitting:** Perfect performance on the training data is a strong indicator of overfitting. Overfitting occurs when a model learns the training data too well, including its noise and peculiarities, to the detriment of its ability to generalize to new, unseen data.

The perfect scores on the training set confirm that the model has learned the training data completely. However, the true test of a model's performance is how well it performs on unseen data (like your validation and test sets). The classification report you shared earlier for the test set provides a more realistic picture of the model's ability to generalize to new transactions. Comparing the training set performance to the test set performance helps identify if overfitting is occurring. In this case, the slight drop in precision and recall for the 'Fraud' class on the test set compared to the perfect training set scores suggests some degree of overfitting might be present.

B. Testing Data Ouput (Random Forest)

The accuracy of testing data is 100%, the recall value is 1.00 for Not Fraud and 0.76 for fraud transactions (100% for Not Fraud and 76% for fraud). The precision value of Not Fraud transaction is 100% and for fraudulent transaction is 97%. F1-score for Not Fraud transaction is 100% and for fraud transaction is 85%.

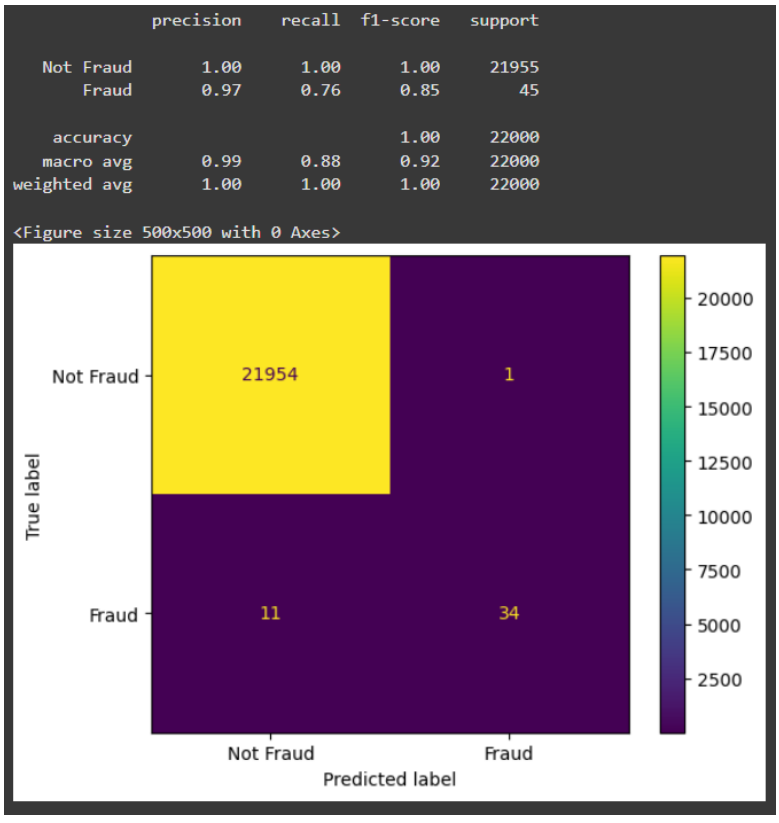

Figure 4.6: confusion matrix

The Random Forest model performs exceptionally well on the majority class ("Not Fraud"). For the minority class ("Fraud"), the model achieves high precision (97%), meaning when it flags something as fraud, it's very likely to be correct. However, its recall (76%) is lower, indicating that it misses a significant portion (24%) of the actual fraudulent transactions.

4.4.4 Rationale for Using NSGA-II in Hyperparameter Tuning

The Random Forest model performs exceptionally well on the majority class ("Not Fraud"). For the minority class ("Fraud"), it achieves a high precision of 97%, meaning that when the model predicts fraud, it is highly likely to be correct. However, the recall stands at only 76%, indicating that 24% of actual fraud cases go undetected. This trade-off is a common challenge in imbalanced classification problems like fraud detection, where finding the right balance between false positives and false negatives is critical. A major contributor to this imbalance in performance often lies in the choice of hyperparameters. Random Forest has several key hyperparameters (e.g., number of trees, max depth, feature sampling strategy) that can significantly influence both recall and precision. However, tuning them manually or through single-objective optimization may fail to capture the optimal trade-offs.

This challenge motivates our approach: we frame hyperparameter tuning as a multi-objective optimization problem and use the NSGA-II algorithm to explore diverse combinations of hyperparameters. This allows us to generate a Pareto front of non-dominated solutions, offering multiple high-performing models that each balance recall and precision differently empowering us to make more informed choices tailored to specific risk and cost considerations.

4.5 Mathematical Formulation of the Hyperparameter Optimization Problem

We formulate the task of optimizing the hyperparameters of a Random Forest classifier as a multi-objective optimization problem. The goal is to find a set of hyperparameters that simultaneously:

- Maximize the **Recall**: the ability to identify actual frauds.
- Maximize the **Precision**: the accuracy of the fraud predictions made.

Let the vector of hyperparameters be defined as:

$$\mathbf{x} = [n_{\text{estimators}},\ \text{max\_depth},\ \text{min\_samples\_split},\ \text{max\_features},\ \text{bootstrap}]$$

The optimization problem can be written as:

$$\begin{aligned} &\underset{\mathbf{x}\in\mathcal{H}}{\text{maximize}} \quad [f_1(\mathbf{x}) = \text{Recall}(\mathbf{x}),\quad f_2(\mathbf{x}) = \text{Precision}(\mathbf{x})] \\ &\text{subject to} \quad \mathbf{x} \in \mathcal{H} \end{aligned}$$

Where $\mathcal{H}$ is the hyperparameter search space:

$$\begin{aligned} n_{\text{estimators}} &\in [10, 150] \subset \mathbb{N} \\ \text{max\_depth} &\in [3, 20] \subset \mathbb{N} \\ \text{min\_samples\_split} &\in [2, 10] \subset \mathbb{N} \\ \text{max\_features} &\in \{\texttt{'sqrt'}, \texttt{'log2'}\} \\ \text{bootstrap} &\in \{\texttt{True}, \texttt{False}\} \end{aligned}$$

The NSGA-II algorithm is then used to explore this space and identify the set of non-dominated solutions, forming the Pareto front that balances the trade-off between recall and precision.

4.5.1 NSGA-II Hyperparameter Optimization with DEAP: Architecture Overview

The optimization process for tuning Random Forest hyperparameters using the **NSGA-II algorithm** relies on various components from the **DEAP (Distributed Evolutionary Algorithms in Python)** framework. Below is the structured breakdown of the tools and their roles:

1. Library Importation

```
from deap import base, creator, tools, algorithms
```

Purpose: Load the core DEAP components used to define individuals, evolutionary operators, and the evolutionary algorithm.

2. creator Module —Define Types

Used to create customized classes for individuals and fitness:

- **FitnessMulti**:
  - Represents **multi-objective fitness**.
  - `weights=(1.0, 1.0)` implies **maximization** of two objectives (e.g., **Recall** and **Precision**).
- **Individual**:
  - Defines an individual solution (a **dictionary of hyperparameters**).
  - Linked to `FitnessMulti` to carry fitness scores.

```
creator.create("FitnessMulti", base.Fitness, weights=(1.0, 1.0))
creator.create("Individual", dict, fitness=creator.FitnessMulti)
```

3. base.Toolbox —Register Core Functions

Acts as a **function registry** for the evolutionary process:

- `attr_dict`: Generates a **random hyperparameter dictionary**.
- `individual`: Builds an `Individual` using `attr_dict`.
- `population`: Creates a list of individuals (population).
- `evaluate`: Evaluates a solution's performance using **cross-validation** to compute **Recall** and **Precision**.
- `mate`: Crossover function (e.g., `cxUniform`).
- `mutate`: Mutation function (custom or default).
- `select`: Uses `selNSGA2`, the NSGA-II selection strategy.

```
toolbox = base.Toolbox()
toolbox.register("individual", tools.initIterate, creator.Individual, attr\_dict)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
toolbox.register("evaluate", evaluate)
toolbox.register("mate", tools.cxUniform, indpb=0.5)
toolbox.register("mutate", mutate\_individual)
toolbox.register("select", tools.selNSGA2)
```

4. tools Module —Evolutionary Operators

Provides **ready-to-use genetic operators**:

- `initIterate`, `initRepeat`: Initialize individuals and population.
- `cxUniform`: Uniform crossover (used in mating).
- `mutShuffleIndexes`: Optional mutation utilities.
- `selNSGA2`: Core **NSGA-II selection operator**.
- `HallOfFame`: Stores **non-dominated solutions** found during evolution.

5. algorithms Module —Main Loop

Implements the **evolutionary algorithm process**:

- algorithms.eaSimple: Executes the **main genetic algorithm loop**, which includes:
  Selection → Crossover → Mutation → Evaluation
- Although simple, when combined with tools.selNSGA2, it effectively implements the logic of **NSGA-II**.

Summary Table

Component	Role
creator	Defines `FitnessMulti` (objective scores) and `Individual` (solution)
toolbox	Registers genetic operators, evaluation, and generation functions
tools	Provides selection, mutation, initialization, and storage utilities
algorithms	Executes the evolutionary loop (`eaSimple`) with NSGA-II logic

4.4.2 Non-dominated Solutions found by NSGA-II

The output you provided has two main parts: **the evolutionary process log** and the **list of non-dominated solutions.**

```
Running NSGA-II for hyperparameter optimization...
gen     nevals
0       50
1       39
2       32
3       38
4       39
5       35
```

Figure 4.7: Genetic Algorithm Progress Log

1. Genetic Algorithm Progress Log The table columns `gen` and `nevals` provide insight into the progress of the genetic algorithm during the optimization process.

- **gen**: This column represents the generation number of the genetic algorithm. The process begins at generation 0 (initial random population) and runs for *n_gen* generations (in this case, set to 5 in the `optimize_rf_hyperparameters_nsga2` function).
- **nevals**: This column indicates the number of individuals whose fitness (i.e., Recall and Precision) was evaluated during each generation.
  - In generation 0 we have 50 individuals were evaluated, corresponding to the initial population size (`pop_size`).
  - In subsequent generations (1 through 5), the number of evaluations is slightly less than the population size (typically between 30 and 40). This is because NSGA-II, like many evolutionary algorithms, evaluates the fitness of newly generated offspring created via crossover and mutation. These offspring then replace part of the current population based on selection criteria involving both fitness and diversity.
  - The exact number of evaluations per generation may vary slightly depending on implementation details and replacement strategies.

This log thus provides a compact summary of how the algorithm evolves the population over successive generations.

Table 4.1: Non-dominated Hyperparameter Solutions Found by NSGA-II

ID	Recall	Precision	n_estimators	max_depth	min_samples_split	max_features
1	0.7778	0.9222	24	8	2	log2
2	0.7778	0.8949	84	13	3	sqrt
3	0.7778	0.8949	145	11	2	sqrt
4	0.7778	0.8949	84	18	3	sqrt
5	0.7556	0.9095	55	10	2	log2
6	0.7556	0.8949	104	18	7	sqrt
7	0.7556	0.8949	136	18	4	sqrt
8	0.7556	0.8949	73	15	6	sqrt
9	0.7556	0.8949	73	18	6	sqrt
10	0.7556	0.8949	88	19	7	sqrt

2. Non-dominated Solutions found by NSGA-II (Recall, Precision):

Interpretation of the Non-dominated Solutions: These 10 solutions represent different trade-offs between Recall and Precision found by the genetic algorithm within the defined hyperparameter search space.

- Notice that solutions 1 through 4 all achieve the highest Recall (0.7778) but have slightly varying Precision values (0.9222 and 0.8949). Solution 1 has the highest Precision (0.9222) among those with the top Recall.
- Solutions 5 through 10 have a slightly lower Recall (0.7556) but also have high Precision (0.9095 and 0.8949).

We would typically choose one of these non-dominated solutions as my "optimized" hyperparameters based on which trade-off between Recall and Precision is most desirable for your specific application (e.g., is it more important to catch every possible fraud, even if it means more false alarms, or is it more important to minimize false alarms, even if some fraud is missed?).

I choose the second set of hyperparameters (n_estimators= 84, max_depth= 13, min_samples_split= 3, max_features= ('sqrt'), bootstrap= False) because it catch an other fraud .

4.5.3 Results of RFC with NSGA-II

After training the random forest model with the optimized hyperparameters we have: The accuracy of testing data is 100%, the recall value is 1.00 for Not Fraud and 0.76 for fraud transactions (100% for Not Fraud and 78% for fraud). The precision value of Not Fraud transaction is 100% and for fraudulent transaction is 97%. F1-score for Not Fraud transaction is 100% and for fraud transaction is 86%.

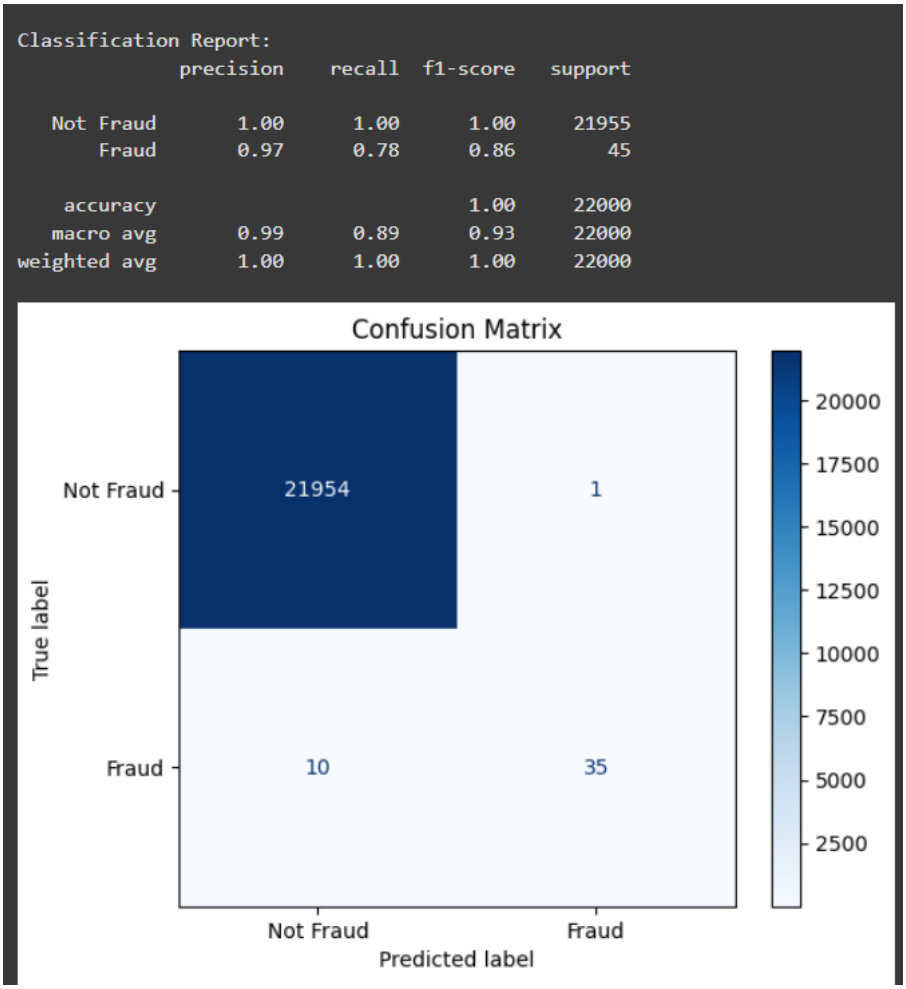

Figure 4.8: Matrix confusion

4.6 Results analysis

Before applying NSGA-II, the Random Forest model achieved excellent metrics on the *Not Fraud* class but only a **76% recall** on the *Fraud* class—meaning that **24% of actual fraud cases were**

missed. To address this, I decided to use the **NSGA-II algorithm** to simultaneously maximize both *recall* and *precision* by tuning the model's hyperparameters. This decision was also driven by the observation that Random Forest results varied across runs due to its inherent randomness.

However, during the implementation of the evolutionary NSGA-II algorithm, I faced significant **execution time** and **computational resource** constraints. To manage this, I imposed several limitations:

- I reduced the training set to only **22,000 labeled transactions**, which represents **12.95% of the full dataset**,
- I capped the number of trees (`n_estimators`) at **150**,
- I limited the number of generations to **5** and the population size to **50**.

These constraints helped reduce the runtime but also limited the quality of the resulting *Pareto front*. At the end of the optimization process, NSGA-II provided **ten non-dominated hyperparameter sets**. I then trained a Random Forest model with each of these configurations and evaluated them on the **training data**. As expected, the results were modest, mainly due to the fact that only **12.95% of the dataset** had been used during optimization.

Surprisingly, when evaluating on the **test data**, the results were **notably improved**, as illustrated in 4.9 and 4.10. Compared to findings in the literature, such as [13], we can reasonably assume that if NSGA-II were given more freedom (e.g., **80% of the dataset**, **20 generations**, and a **population size of 200**), the performance gains would likely be much more significant than a mere **2% difference**, potentially achieving even better precision–recall trade-offs.

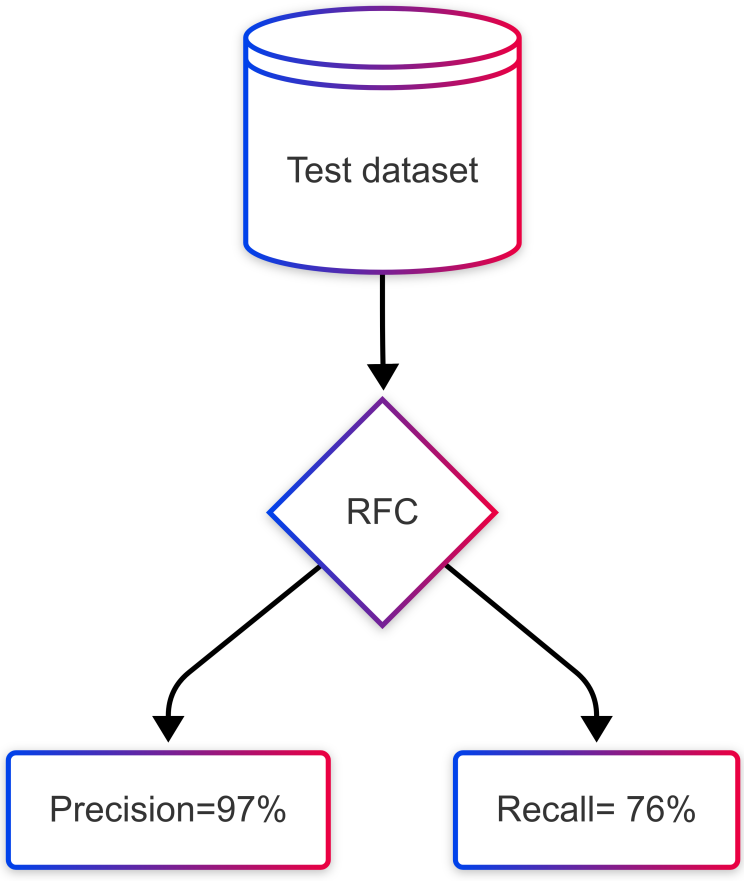

Figure 4.9: Model Performences before NSGA-II

After applying NSGA-II to optimize hyperparameters, recall for fraud improved to 78%, while precision remained high at 97%. This balanced trade-off was achieved by exploring non-dominated solutions.

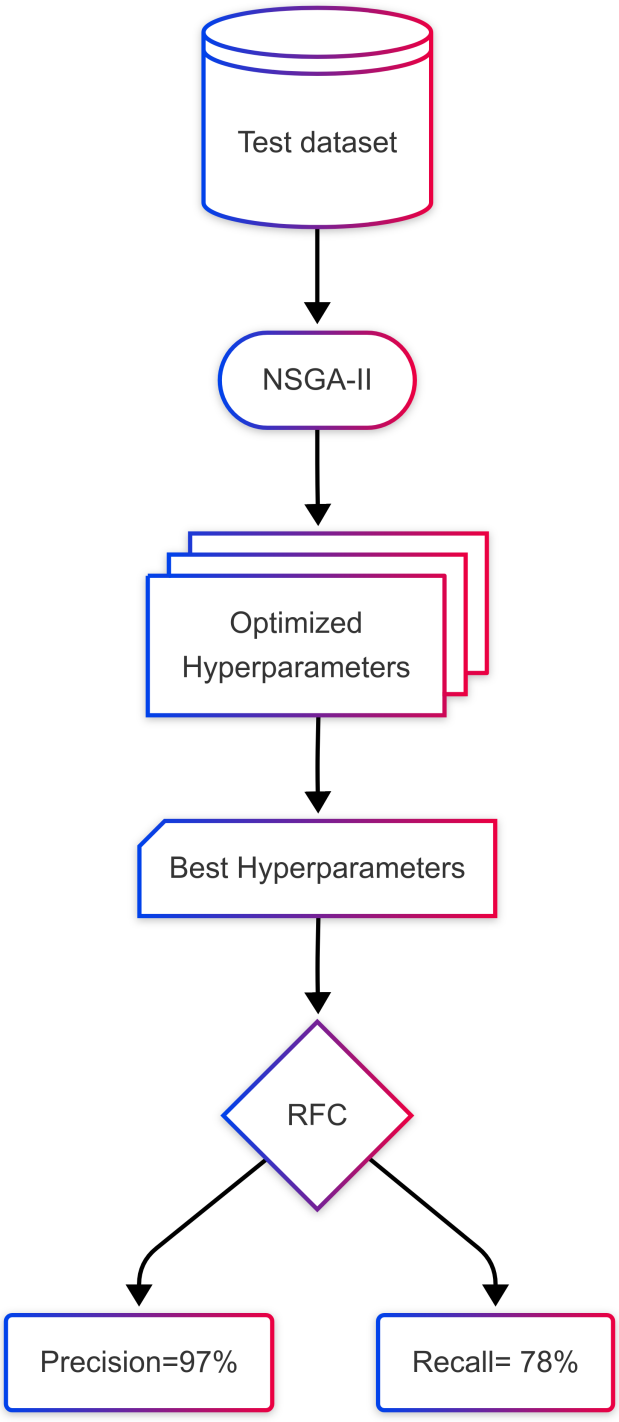

Figure 4.10: Model Performences after NSGA-II

But, when evaluating on the **data validation**, the results were the same even using the only RFC or RFC with NSGA-II as illustrated in 4.11.

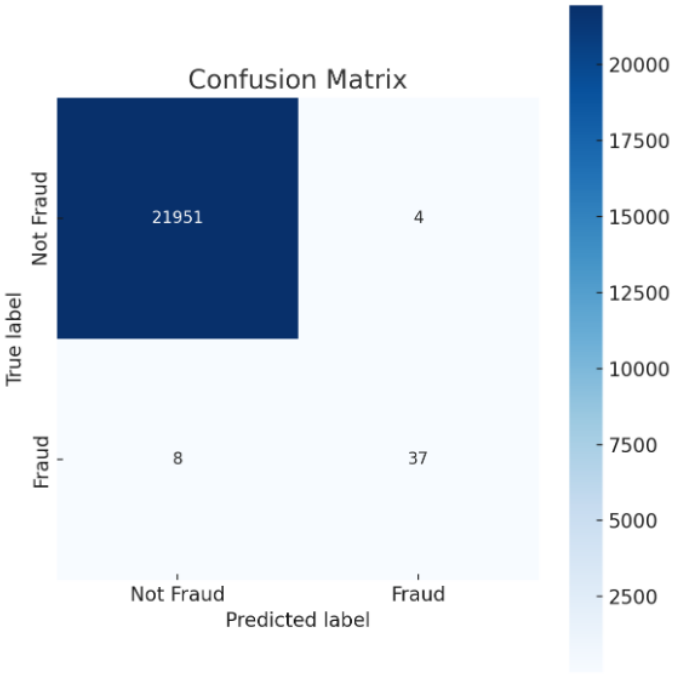

Figure 4.11: Data validation

Conclusion

We have successfully loaded and preprocessed the credit card transaction data, addressing the scaling of the *Amount* and *Time* features. We then employed a multi-objective genetic algorithm (NSGA-II) to optimize the hyperparameters of a Random Forest Classifier, aiming to balance the trade-off between precision and recall for fraud detection on an imbalanced dataset.

The optimization process yielded several non-dominated solutions representing different performance trade-offs. Using one of these optimized hyperparameter sets, we trained the Random Forest model and evaluated its performance on a separate test set. The evaluation results demonstrate that the model achieves high precision in identifying fraudulent transactions on unseen data, minimizing false alarms, while also maintaining a respectable recall rate. This establishes a solid baseline for credit card fraud detection using a genetically optimized Random Forest, providing a set of hyperparameters that offer a favorable balance between catching fraud and minimizing misclassifications of legitimate transactions.

General conclusion and future work

This thesis has addressed the critical challenge of fraud detection in financial transactions, where class imbalance, complex fraud patterns, and real-time decision-making constraints render traditional approaches insufficient. The main objective was to enhance detection performance—particularly recall and precision—by optimizing the hyperparameters of a Random Forest model using the NSGA-II multi-objective evolutionary algorithm.

We successfully preprocessed a highly imbalanced real-world dataset of credit card transactions. A baseline Random Forest classifier showed excellent metrics for the "Not Fraud" class but only 76% recall on fraudulent cases, indicating a substantial risk of undetected fraud. Through NSGA-II optimization, we generated a set of non-dominated solutions representing different trade-offs between precision and recall. Despite hardware constraints that limited the population size and number of generations, the optimized models demonstrated improved and more stable results when tested on unseen data.Notably, we reached a recall rate of **78%**, meaning we successfully improved fraud detection accuracy—even under constrained conditions.

The central research question was whether a multi-objective optimization approach could improve fraud detection performance without compromising overall model robustness. The experimental results indicate that NSGA-II effectively enhances the balance between false positives and false negatives, confirming that this approach offers a viable answer to the problem.

Our findings show the practical potential of combining Random Forest with evolutionary optimization for fraud detection. Financial institutions can leverage such strategies to better adapt their models to specific datasets and operational contexts, thus increasing detection rates while controlling the cost of false alarms.

This work contributes to the theoretical understanding of how evolutionary algorithms—traditionally used in optimization—can support machine learning in imbalanced classification tasks. It also underscores the relevance of exploring the Pareto front instead of focusing solely on a single optimal solution.

This research was subject to several constraints. Only 12.95% of the dataset was used in the optimization phase due to computational limitations, potentially limiting the diversity of the explored solutions. Moreover, the number of generations and population size in NSGA-II were reduced to ensure feasibility. Lastly, only the Random Forest model was studied, without benchmarking against other machine learning algorithms.

Future studies could extend this work by granting NSGA-II more computational freedom (more data, generations, and population size). Exploring alternative optimization algorithms such as SPEA2 or MOEA/D could yield even more competitive results. Additionally, this approach could be tested on other classifiers such as XGBoost, LightGBM, or neural networks. Incorporating online learning and streaming data techniques would be particularly relevant for real-time fraud detection environments.

In conclusion, this thesis demonstrates the relevance and effectiveness of hybrid approaches that combine artificial intelligence with evolutionary optimization to tackle the persistent challenge of financial fraud. Despite certain technical limitations, the outcomes are promising and lay a strong foundation for future work in a field where precision, adaptability, and speed remain essential to risk management.

Bibliography

[1] Basel Committee on Banking Supervision. *International Convergence of Capital Measurement and Capital Standards: A Revised Framework*. Bank for International Settlements, Basel, Switzerland, 2004.

[2] Basel Committee on Banking Supervision. Basel iii: Finalising post-crisis reforms (also known as basel iv), 2017. Available at https://www.bis.org/bcbs/publ/d424.htm.

[3] Bernard, S. *Random Forests: From Analysis of Functioning Mechanisms to Dynamic Construction*. PhD thesis, University of Lyon, December 2009. PhD Thesis.

[4] Blank, J. pymoo: Nsga-ii implementation, 2020. Multi-objective optimization in Python.

[5] Breiman, L. Bagging predictors. *Machine Learning 24*, 2 (1996), 123–140.

[6] Breiman, L. Random forests. *Machine Learning 45*, 1 (2001), 5–32.

[7] Bylander, T. Estimating generalization error on two-class datasets using out-of-bag estimates. *Machine Learning 48*, 1-3 (2002), 287–297.

[8] Caron, S. An introduction to decision trees. In *CHES: Conference on Health, Environment and Society* (2011).

[9] Cohen, B., and Goldstein, M. Developing risk taxonomies to improve risk identification. *Journal of Risk Research 23*, 4 (2020), 456–472.

[10] Collette, Y., and Siarry, P. Multiobjective optimization using evolutionary algorithms: A comparative case study. *IEEE Transactions on Evolutionary Computation 6*, 2 (2002), 117–131.

[11] Company, M. . Reducing false positives in fraud detection. *McKinsey Insights* (2023).

[12] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation 6*, 2 (2002), 182–197.

[13] Deepika, K., Nagenddra, M. P. S., Ganesh, M. V., and Naresh, N. Implementation of credit card fraud detection using random forest algorithm. *International Journal for Research in Applied Science & Engineering Technology (IJRASET) 10*, III (Mar 2022), 797. IC Value: 45.98; SJ Impact Factor: 7.538; ISRA Journal Impact Factor: 7.894.

[14] Dietterich, T. G. Ensemble methods in machine learning. In *Multiple Classifier Systems* (2000), vol. 1857 of *Lecture Notes in Computer Science*, Springer, pp. 1–15.

[15] Frank, A., and Asuncion, A. UCI machine learning repository. https://archive.ics.uci.edu/, 2010. University of California, Irvine, School of Information and Computer Sciences.

[16] Gartner. Continuous authentication and behavioral biometrics, 2024.

[17] Genuer, R. *Random Forests: Theoretical Aspects, Variable Selection and Applications*. PhD thesis, University Paris-Sud, 2010. PhD Thesis.

[18] Géron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 3rd ed. O'Reilly Media, 2022.

[19] Javelin Strategy & Research. 2024 identity fraud study: The evolving threat landscape, 2024.

[20] Katona, A. Nsga-2 tutorial. https://github.com/adam-katona/NSGA_2_tutorial/blob/master/NSGA_2_tutorial.ipynb, 2023. Consulté le [Date de consultation].

[21] Krebs, B. The poly network hack: Anatomy of a $37 million fraud, 2023. Krebs on Security.

[22] Leeson, N. *Rogue Trader: How I Brought Down Barings Bank*. Penguin, 1995.

[23] Louppe, G. *Understanding Random Forests: From Theory to Practice*. PhD thesis, University of Liège, 2014.

[24] Mastercard. Securing digital payments with token vaults. Tech. rep., Mastercard White Paper, 2022.

[25] Merchant Risk Council. Global false positives impact study, 2024.

[26] MIT Technology Review. Federated learning for financial fraud detection. *MIT Technology Review* (2024).

[27] Motimo, P. Random forests: Mathematical foundations and applications. https://perso.math.univ-toulouse.fr/motimo/files/2013/07/random-forest.pdf, 2013. Document académique, Université Toulouse. Consulté le 17 juin 2025.

[28] Probst, P., Wright, M. N., and Boulesteix, A.-L. Hyperparameters and tuning strategies for random forest. *arXiv preprint arXiv:1802.09596* (2019).

[29] Ramdani, Z. Cours sur optimisation multiobjectifs. Master 1 Course Notes, 2023. Faculté des Mathématiques et Informatique.

[30] Review, H. B. The financial impact of security breaches on public companies. *Harvard Business Review* (2024).

[31] scikit-learn Developers. *scikit-learn: RandomForestClassifier Documentation*, 2023. Version 1.2.0.

[32] Scornet, E. Learning and random forests. *Knowledge-Based Systems 44* (2012), 48–56. 2011-2012.

[33] Seshadri, A. Nsga-ii: A multi-objective optimization algorithm, 2025. Retrieved June 17, 2025.

[34] Smith, J. The wells fargo scandal and its financial consequences. *Journal of Banking Ethics 18*, 2 (2021), 134–145.

[35] Tarantino, A. Governance, risk, and compliance handbook: Technology, finance, environmental, and international guidance and best practices.

[36] Team, M. Supervised vs unsupervised learning: Key differences, 2023. Blog post.

[37] Times, F. Aftermath of the poly network exploit: Industry-wide changes. *Financial Times* (2024).

[38] University of Wolverhampton. Data mining: Process, techniques and applications. Online course guide, 2023. Accessed: 17 June 2025.

Résumé & Abstract

Résumé

Ce mémoire s'inscrit dans le domaine de la détection de la fraude financière, un enjeu majeur des systèmes bancaires modernes. L'objectif est de développer un modèle efficace basé sur l'apprentissage automatique, capable d'identifier les transactions frauduleuses dans un environnement déséquilibré.

Des données réelles de paiements ont été prétraitées puis analysées à l'aide de modèles supervisés comme les arbres de décision et les forêts aléatoires. Une optimisation multi-objectifs par l'algorithme NSGA-II a permis d'améliorer le compromis entre précision et rappel.

Les résultats montrent que le modèle Random Forest optimisé atteint un rappel de 78% sur l'ensemble de validation, avec un faible taux de fausses alertes. Ce travail renforce l'efficacité des systèmes de détection automatisée et ouvre la voie à des recherches futures intégrant des approches hybrides ou en temps réel.

Mots-clés : détection de fraude, apprentissage automatique, forêts aléatoires, NSGA-II, données déséquilibrées.

Abstract

This thesis focuses on financial fraud detection, a major challenge for modern banking systems. The goal is to build an efficient machine learning model that can identify fraudulent transactions in imbalanced datasets.

Real-world payment data was preprocessed and analyzed using supervised models such as Decision Trees and Random Forests. Multi-objective optimization with the NSGA-II algorithm was applied to enhance the balance between precision and recall.

Results show that the optimized Random Forest model achieves 78% recall on the validation set while keeping false positives low. This work contributes to improving automated fraud detection systems and lays the groundwork for future hybrid or real-time approaches.

Keywords: fraud detection, machine learning, random forest, NSGA-II, imbalanced data.