**Advanced Programming 2025**

# A Comparative Study of Machine Learning Methods for Credit Card Fraud Detection

Final Project Report

Luca Benkoussa Buquet
luca.benkoussa-buquet@unil.ch
Student ID: 1287515

January 3, 2026

**Abstract**

Credit card fraud detection is a major challenge faced by financial institutions due to the large volume of transactions and the significant imbalance between legitimate and fraudulent operations. In this project, we address the problem of classifying credit card transactions as fraudulent or non-fraudulent based on various features related to the payment process. To solve this binary classification task, we evaluate several machine learning models, including logistic regression, decision trees, and ensemble methods such as bagging, in order to compare their effectiveness and limitations. The models are trained and tested on a real-world dataset containing 284,807 transactions, of which only 0.172% correspond to fraud cases. Our experiments show that the strong class imbalance makes some classification methods difficult to interpret and less reliable when using standard performance metrics. However, we demonstrate that certain models remain robust and achieve satisfactory performance in detecting fraudulent transactions. These results highlight the importance of model selection and evaluation strategies when dealing with highly imbalanced datasets in fraud detection tasks.

**Keywords:** Data Science, Machine Learning, Binary Classification, Fraud Detection, Imbalanced Dataset, Supervised Learning

# Contents

# 1 Introduction

The widespread adoption of digital payment systems has profoundly transformed the way financial transactions are conducted. With the increasing use of credit cards and contactless payment solutions such as Apple Pay and Google Pay, financial operations have become faster and more convenient. However, this growing reliance on electronic payments has also led to a rise in fraudulent activities, making fraud detection a critical issue for banks and financial institutions. Ensuring secure transactions while maintaining a seamless user experience has therefore become a major challenge in modern financial systems.

In this context, credit card fraud detection represents a fundamental problem in a society where payments are increasingly dematerialized. Fraud detection systems must be capable of identifying suspicious transactions in real time in order to protect users and limit financial losses. At the same time, these systems must minimize false positives, as incorrectly blocking a legitimate transaction can negatively impact customer trust and satisfaction. This trade-off between security and usability makes fraud detection a particularly complex and important classification task.

The objective of this project is to identify a robust and reliable classification method for credit card fraud detection. To this end, several supervised machine learning models are implemented and compared in order to evaluate their performance on a highly imbalanced dataset. The goal is to determine which approaches provide solid and consistent results, while remaining effective in detecting fraudulent transactions in realistic conditions.

The remainder of this report is organized as follows. Section 2 reviews related work and existing approaches in credit card fraud detection. Section 3 presents the methodology adopted in this project, including the data description, the proposed approach, and the implementation details. The experimental results are reported in Section 4, which covers the experimental setup, performance evaluation, and result visualizations. Section 5 provides a discussion of the obtained results. Finally, Section 6 concludes the report and outlines possible directions for future work.

# 2 Literature Review / Related Work

Credit card fraud detection has been extensively studied in the field of data science and machine learning due to its significant economic impact and the challenges posed by real-world financial data [2]. Early approaches to fraud detection relied on rule-based systems and expert-defined heuristics designed to identify suspicious transactions based on predefined patterns. While these methods were simple to implement, they lacked adaptability and often failed to cope with evolving fraud strategies.

With the rise of machine learning, fraud detection has increasingly been formulated as a supervised classification problem. Traditional algorithms such as logistic regression and decision trees have been widely used due to their interpretability and relatively low computational cost. Logistic regression is commonly considered a strong baseline model in fraud detection tasks, as it provides probabilistic outputs and allows for straightforward analysis of feature importance. Decision trees, on the other hand, are capable of modeling non-linear relationships between features and have therefore been frequently applied in this context [1].

More recent studies highlight the effectiveness of ensemble methods, such as bagging and random forests, for fraud detection problems [3]. By combining multiple classifiers, these approaches improve robustness and reduce variance, which is particularly beneficial in complex and noisy financial datasets. However, their increased complexity can reduce interpretability compared to simpler models.

A major challenge emphasized in the literature is the extreme class imbalance present in credit card fraud datasets, where fraudulent transactions typically account for less than 1% of all operations [2]. As a consequence, standard accuracy metrics are often misleading, and al-

ternative evaluation measures such as precision, recall, and the area under the ROC curve are widely recommended [6]. Publicly available datasets, such as anonymized credit card transaction datasets released for research purposes, are commonly used to benchmark fraud detection methods under highly imbalanced conditions [5].

# 3 Methodology

## 3.1 Data Description

The dataset used in this project is a publicly available credit card transaction dataset provided on Kaggle[1]. It contains transactions made by European cardholders in September 2013 and was collected during a research collaboration between Worldline and the Machine Learning Group (MLG) of Université Libre de Bruxelles (ULB) [3, 4]. The dataset has been widely used as a benchmark in the fraud detection literature.

The dataset includes a total of 284,807 transactions recorded over a two-day period, among which 492 are labeled as fraudulent. This corresponds to only 0.172% of the data, making the dataset highly imbalanced. Such an extreme class imbalance represents a major challenge for standard classification algorithms and requires careful model evaluation strategies.

All input variables are numerical and result from a Principal Component Analysis (PCA) transformation applied to the original features in order to preserve confidentiality. As a consequence, the original features and their semantic meaning are not available. The dataset contains 28 principal components, denoted as V1 to V28. In addition to these components, two features have not been transformed: *Time* and *Amount*. The *Time* feature represents the number of seconds elapsed between a given transaction and the first transaction in the dataset, while *Amount* corresponds to the transaction value and can be used for cost-sensitive learning approaches. The target variable, *Class*, indicates whether a transaction is fraudulent (1) or legitimate (0).

Due to the strong class imbalance, standard accuracy metrics based on confusion matrices are not suitable for evaluating model performance. Instead, performance measures such as precision, recall, and the Area Under the Precision-Recall Curve (AUPRC) are more appropriate and are commonly recommended in related studies [6]. These metrics provide a more meaningful assessment of a model's ability to detect rare fraudulent transactions.

## 3.2 Approach

This project adopts a comparative machine learning approach in order to evaluate the effectiveness of different classification algorithms for credit card fraud detection. Several supervised learning models are considered, including logistic regression, k-nearest neighbors (KNN), decision trees, random forests, bagging, AdaBoost, and a majority voting ensemble. This diverse set of algorithms allows us to analyze both linear and non-linear models, as well as ensemble-based approaches, in the context of a highly imbalanced classification problem.

Regarding data preprocessing, the dataset is split into training and testing sets using an 80% / 20% ratio. The split is performed chronologically in order to preserve the temporal structure of the transactions and avoid information leakage. Feature scaling is applied when required, notably for KNN and logistic regression models, using a pipeline-based approach to ensure that preprocessing steps are consistently applied during training and evaluation. To mitigate the strong class imbalance, class weights are set to `balanced` for models that support this option, allowing higher importance to be assigned to fraudulent transactions.

Rather than relying on a single monolithic model, the project is structured around modular model builders implemented in a dedicated `models.py` file. These builders return either complete pipelines, combining preprocessing and classification steps, or standalone estimators

[1] `https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud`

for models such as decision trees and random forests. The overall workflow consists of loading the dataset (`creditcard.csv`), performing a chronological split using a data loader, identifying numerical and categorical features, instantiating the selected model through the appropriate builder, training the model, and finally evaluating its performance. The trained models and evaluation results are saved for further analysis, with optional visualizations exported as image files.

Model performance is evaluated using metrics that are well suited to imbalanced classification problems. Precision, recall, and F1-score are reported to assess the trade-off between false positives and false negatives. In addition, the Area Under the Precision-Recall Curve (AUPRC) is used as a primary evaluation metric, as it provides a more informative measure than accuracy in highly imbalanced settings. Standard accuracy is intentionally excluded, since predicting all transactions as non-fraudulent would already yield an accuracy of approximately 99.83%, despite offering no practical fraud detection capability.

### 3.3 Implementation

The project is implemented entirely in Python. Data loading and preprocessing are mainly performed using the `pandas` library, while standard utilities such as file and path management rely on the built-in `os` module. Machine learning models, preprocessing pipelines, and evaluation tools are implemented using the `scikit-learn` library. Visualizations and performance plots are generated using `matplotlib`.

The overall architecture of the project and the organization of the source code are illustrated in Figure 1. The project follows a modular structure that separates data loading, model definition, evaluation, and result storage, ensuring clarity, maintainability, and reproducibility.

```
your-project/
├── README.md
├── PROPOSAL.md
├── environment.yml
├── requirements.txt
├── main.py
├── src/
│   ├── __init__.py
│   ├── data_loader.py
│   ├── models.py
│   └── evaluation.py
├── data/
│   └── raw/
├── results/
└── notebooks/
```

Figure 1: Project architecture and code organization

The project is structured around a modular architecture that separates data handling, model definition, and experimental execution, improving code clarity, reusability, and reproducibility. Core functionalities are organized within the `src` directory, which contains dedicated modules for data loading, model construction, and evaluation. The `data_loader.py` module is responsible for loading the dataset and performing the chronological train-test split; for this purpose, the dataset must be placed in the `data/raw` directory. Model construction is imple-

mented in `models.py` through reusable builder functions that return either complete preprocessing pipelines or standalone estimators. Model evaluation and result visualization are managed in the `evaluation.py` module.

Experimental scripts orchestrate the end-to-end workflow by combining data loading, model training, evaluation, and result storage. The project directory further distinguishes between raw data, source code, and outputs: original data files are stored in `data/raw`, while trained models, evaluation metrics, and generated figures are saved in the `results` directory. Additional configuration files and optional Jupyter notebooks are included to support reproducibility and exploratory analysis, following common best practices in applied machine learning projects.

```python
def load_and_split(
    path: str,
    target: str = 'Class',
    time_col: str = 'Time',
    test_size: float = 0.2
):
    df = load_raw(path)

    if time_col in df.columns:
        df = df.sort_values(time_col)
    else:
        df = df.reset_index(drop=True)

    if target not in df.columns:
        raise ValueError(f"Target column '{target}' not found")

    X = df.drop(columns=[target])
    y = df[target]

    n = len(df)
    train_end = int((1.0 - test_size) * n)

    X_train = X.iloc[:train_end]
    y_train = y.iloc[:train_end]
    X_test = X.iloc[train_end:]
    y_test = y.iloc[train_end:]

    return X_train, X_test, y_train, y_test
```

Listing 1: Chronological data loading and train-test split

```python
from sklearn.ensemble import RandomForestClassifier

def build_random_forest(
    n_estimators: int = 200,
    class_weight: str = 'balanced',
    random_state: int = 42,
    **kwargs
):
    return RandomForestClassifier(
        n_estimators=n_estimators,
        class_weight=class_weight,
        random_state=random_state,
        **kwargs
    )
```

Listing 2: Random Forest model builder

```python
from src.data_loader import load_and_split
from src.models import build_random_forest
from src.evaluation import print_scores
import joblib
```

```
5
6
7  def main():
8      X_train, X_test, y_train, y_test = load_and_split(
9          'data/raw/creditcard.csv'
10     )
11
12     rf = build_random_forest()
13     rf.fit(X_train, y_train)
14
15     y_pred = rf.predict(X_test)
16     y_proba = rf.predict_proba(X_test)[:, 1]
17
18     print_scores(y_test, y_pred, y_proba)
19     joblib.dump(rf, 'results/random_forest.joblib')
```

Listing 3: End-to-end training and evaluation pipeline

Together, these components illustrate the complete machine learning workflow, from raw data loading to model training, evaluation, and persistence, following a clear and modular design.

# 4 Results

In order to reproduce and inspect the experimental results, the different models can be trained and evaluated by running specific commands from the terminal. Each command executes the corresponding experiment independently, allowing a focused analysis of each classification method.

- `make logistic` runs the logistic regression experiment
- `make rf` runs the Random Forest experiment
- `make knn` runs the KNN experiment
- `make tree` runs the Decision Tree experiment
- `make adaboost` runs the AdaBoost experiment
- `make bagging` runs the Bagging experiment
- `make voting` runs the Voting ensemble experiment
- `make all` trains all default models (equivalent to `python -m src.train`)

## 4.1 Experimental Setup

All experiments were conducted on a personal computer equipped with an Apple M1 processor, using a CPU-based setup. No GPU acceleration or distributed computing was used, as the considered machine learning models can be efficiently trained on standard hardware.

The experimental environment was based on Python version 3.11. The software environment was managed using a Conda configuration to ensure consistency and reproducibility across experiments.

Unless otherwise specified, models were trained using the default hyperparameters provided by `scikit-learn`. To mitigate the strong class imbalance present in the dataset, the `class_weight='balanced'` option was enabled for all models that support it.

For ensemble methods, AdaBoost was trained using a base estimator with 30 boosting iterations, while the Bagging classifier used 10 estimators. The Random Forest and Decision Tree models were trained with the entropy criterion. The Voting classifier combined logistic regression and random forest models using soft voting. For K-nearest neighbors, the number of neighbors

was set to 5 with distance-based weighting. Logistic regression was trained with a maximum of 1000 iterations to ensure convergence.

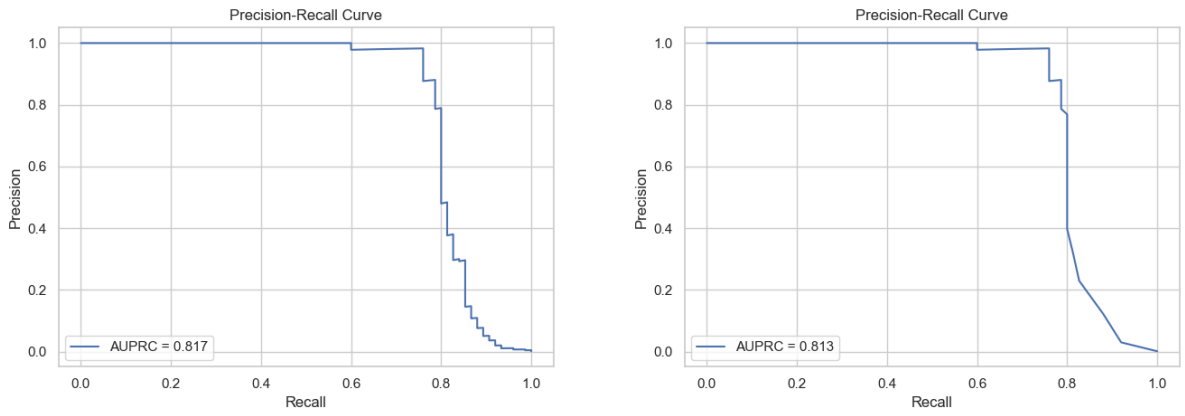
## 4.2 Performance Evaluation

Table 1: Performance comparison of classification models

| Model | Precision | Recall | F1-score | AUPRC |
|---|---|---|---|---|
| Logistic Regression | 0.071 | 0.893 | 0.131 | 0.762 |
| KNN | 0.964 | 0.707 | 0.815 | 0.754 |
| Decision Tree | 0.776 | 0.693 | 0.732 | 0.539 |
| Random Forest | 0.980 | 0.653 | 0.784 | 0.806 |
| AdaBoost | 0.791 | 0.453 | 0.576 | 0.563 |
| Bagging | 0.962 | 0.680 | 0.797 | 0.744 |
| Voting Ensemble | 0.550 | 0.800 | 0.652 | 0.817 |

Table 1 reports the performance of the different classification models using precision, recall, F1-score, and AUPRC. Given the strong class imbalance of the dataset, AUPRC is considered the most informative metric. The results highlight substantial performance differences across models, as well as clear trade-offs between precision and recall. Ensemble-based methods, such as Random Forest and Voting, achieve the highest AUPRC values, while simpler models exhibit more extreme behaviors, emphasizing the importance of using appropriate evaluation metrics in imbalanced classification settings.

## 4.3 Visualizations




(a) Voting ensemble (b) Random Forest

Figure 2: Precision–Recall curves for the best-performing models

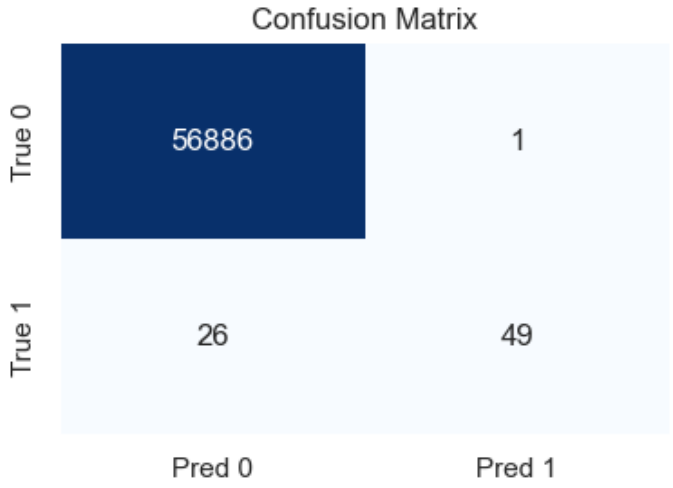



Figure 3: Confusion matrix for the Random Forest model

Figure 2 presents the Precision–Recall curves for the Voting ensemble and Random Forest models, which achieve the highest AUPRC values among the evaluated methods. Both curves remain close to the upper-right region of the plot, indicating strong performance in detecting fraudulent transactions under class imbalance. The Voting ensemble exhibits a slightly more stable precision across intermediate recall levels, while the Random Forest shows a comparable but marginally steeper precision drop at very high recall values. For both models, precision decreases significantly when recall exceeds approximately 0.8, illustrating the trade-off between maximizing fraud detection and limiting false alarms.

Figure 3 shows the confusion matrix obtained with the Random Forest model. The model correctly classifies the vast majority of legitimate transactions, with only one false positive, resulting in a very high precision. Among fraudulent transactions, 49 cases are correctly detected, while 26 frauds remain undetected. This behavior illustrates a conservative classification strategy that strongly limits false alarms at the cost of a moderate recall, which is consistent with the performance metrics reported in Table 1.

# 5 Discussion

The evaluated models exhibit different behaviors, with some prioritizing recall while others favor precision. A recall close to 1 indicates that only a small number of fraudulent transactions escape detection, whereas a precision close to 1 implies that the model generates very few false alarms.

## 5.1 What Worked Well

The experimental results show that ensemble-based methods perform particularly well in the context of credit card fraud detection. Models such as Random Forest, Bagging, and Voting Ensemble achieve the highest AUPRC values, indicating strong robustness when dealing with a highly imbalanced dataset. Among them, the Voting Ensemble model reaches the best overall AUPRC score, suggesting that combining complementary classifiers can lead to improved detection performance.

The Random Forest model also demonstrates excellent behavior, achieving a very high precision score. This indicates that the model generates very few false alarms, which is a critical requirement in real-world banking applications where unnecessary transaction blocking can negatively affect user experience. In addition, KNN and Bagging models provide a good balance between precision and recall, as reflected by their relatively high F1-scores.

### 5.2 Challenges Encountered

One of the main challenges of this project lies in the extreme class imbalance of the dataset, where fraudulent transactions represent only a very small fraction of the total data. This imbalance makes standard evaluation metrics such as accuracy misleading and complicates the training and assessment of classification models.

Several models exhibit strong trade-offs between precision and recall. For instance, logistic regression achieves a very high recall, meaning that most frauds are detected, but at the cost of extremely low precision, resulting in a large number of false alarms. This illustrates the inherent difficulty of simultaneously maximizing fraud detection while minimizing incorrect alerts.

### 5.3 Limitations of the Approach

Despite the encouraging results, several limitations must be acknowledged. First, the dataset has been anonymized using a PCA transformation, which prevents any meaningful interpretation of feature importance from a business or domain perspective. As a result, the models operate as black boxes with limited explainability.

Second, no advanced resampling techniques, such as synthetic oversampling or adaptive undersampling, were explored to further address class imbalance.

Finally, a fixed decision threshold is used for all models, which may not be optimal in practice. The financial costs associated with false positives and false negatives are also not explicitly modeled, although they play a crucial role in real-world deployment scenarios.

## 6 Conclusion and Future Work

### 6.1 Summary

This project demonstrates that effective binary classification is achievable even in highly imbalanced settings such as credit card fraud detection. Through a thorough evaluation of multiple machine learning models, we show that ensemble-based methods consistently provide the most robust performance under severe class imbalance. In particular, the Random Forest model proves to be especially relevant due to its ability to capture non-linear relationships while maintaining high precision and competitive overall performance.

### 6.2 Future Directions

As discussed above, further work could focus on exploring advanced techniques specifically designed to handle class imbalance, such as resampling methods (e.g., SMOTE), in order to assess whether simpler models like logistic regression can achieve improved performance. Additionally, more complex approaches, including neural networks, could be investigated to capture richer patterns in the data. Nevertheless, the results of this project suggest that ensemble-based methods, particularly Random Forests, already offer strong performance and could be realistically deployed in financial institutions to enhance fraud detection while preserving a satisfactory user experience.

## References

[1] Alejandro Correa Bahnsen, Djamila Aouada, and Bjorn Ottersten. "Cost-sensitive decision trees for fraud detection". In: *Expert Systems with Applications* (2015).

[2] Andrea Dal Pozzolo, Gianluca Bontempi, and Mathieu Snoeck. "Adversarial drift detection in credit card fraud". In: *IEEE Transactions on Neural Networks and Learning Systems* (2015).

[3] Andrea Dal Pozzolo et al. "Calibrating probability with undersampling for unbalanced classification". In: *IEEE Symposium Series on Computational Intelligence and Data Mining* (2015).

[4] Andrea Dal Pozzolo et al. "Learned lessons in credit card fraud detection from a practitioner perspective". In: *Expert Systems with Applications* 41.10 (2014), pp. 4915–4928.

[5] European cardholders. *Credit Card Fraud Detection Dataset*. Available on Kaggle: `https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud`. 2013.

[6] Tom Fawcett. "An introduction to ROC analysis". In: *Pattern Recognition Letters* (2006).

## A Code Repository

**GitHub Repository:** `https://github.com/Lucabb08/fraud_detection_project`
The structure of the project repository follows the organization presented in Figure 1, which illustrates the separation between data, source code, experiments, and results. The setup and usage instructions of the project are clearly documented in the `README.md` file.

## B Helper Tools

The following helper tools were used during the development of this project:

- **ChatGPT 5.2**: used as an assistance tool for code clarification, debugging support, and improvement of documentation and report structure. All algorithmic choices, model implementations, experimental results, and interpretations were designed and validated by the author.