Machine Learning for Cybersecurity

Credit Card Fraud Detection

Thibaud RIMBERT, Eliott HANGARD, Arsène GALLIEZ, Hugo DOS ANJOS DOS SANTOS

IPSA

Table des matières

1. Introduction 1
2. Context of the project 1
3. Exploratory data analysis 2
4. Methodology and results 6
4.1. Models Selection 6
4.2. Decision Tree 7
4.3. Random Forest Classifier 8
4.4. XGB classifier 9
4.5. Deep Neural Network 12
5. Conclusion 14

1. Introduction

Detecting fraud has become a top priority for financial institutions, especially in datasets where fraudulent cases are extremely rare. This rarity creates major challenges for machine learning algorithms, which must detect anomalies without being overwhelmed by false alarms. In this report, we focus on how fraud detection can be enhanced using machine learning, with particular emphasis on maximizing recall — since missing a fraud is costlier than a false positive. We begin by explaining the context and nature of the dataset, followed by a detailed analysis of various models including decision trees, random forests, XGBoost, and deep neural networks. Each method is tested and evaluated based on performance in a highly imbalanced setting, leading to a conclusion about the most effective strategy for real-world application.

2. Context of the project

Credit card fraud poses a significant threat to financial institutions and consumers alike, leading to substantial financial losses and undermining trust in digital payment systems. To put this into perspective, "worldwide gross card fraud losses reached $33.83 billion in 2023" [Nilson Report, 2024], highlighting just how massive this problem has become. The challenge in detecting fraudulent transactions lies in their rarity and the ever-evolving tactics employed by fraudsters. Beyond the direct financial impact, fraud also damages the relationship between customers and their banks. Research shows that "more than half of scam victims consider switching financial institutions after their experience, and 30% actually do so" [PYMNTS, 2024]. This erosion of trust demonstrates that the stakes go far beyond just monetary losses, it's about maintaining customer confidence in an increasingly digital financial landscape.
Fortunately, machine learning offers promising solutions to this problem by identifying patterns and anomalies within transaction data that may indicate fraudulent activity. The banking

industry is taking notice: "seven in 10 banking executives say AI will be the most critical technology over the next decade for fraud prevention" [Laura Fitzgerald, 2024]. What makes machine learning particularly powerful in this context is its speed and adaptability. These systems can operate at incredible scales, they "can handle billions of transactions and respond in real-time with accuracy." [Fraud.com].
Perhaps most importantly, unlike traditional rule-based systems that quickly become outdated, "machine learning systems never stop learning and can detect new fraud scenarios and patterns quickly and accurately, helping organizations stay up to date with evolving threats" [Fraud.com]. This continuous learning capability is crucial when dealing with fraudsters who constantly adapt their tactics to bypass security measures.

For this project, we utilize the "Credit Card Fraud Detection" dataset provided by the Machine Learning Group of Université Libre de Bruxelles (ULB) and available on Kaggle. This dataset contains transactions made by European cardholders over two days in September 2013. It contains 284,807 transactions, with only 492 (approximately 0.172%) identified as fraudulent, highlighting the dataset's significant class imbalance.

3. Exploratory data analysis

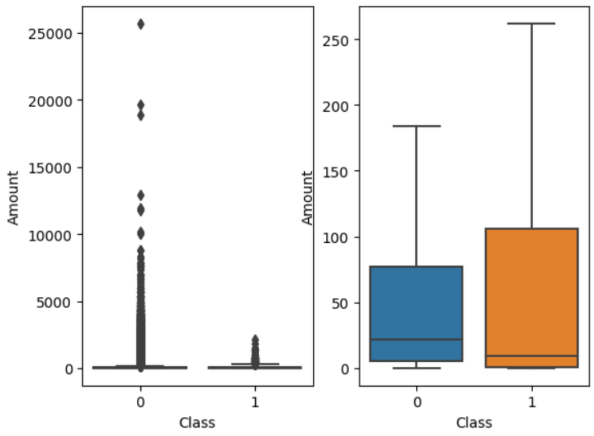
A deep exploration of the dataset is always useful to better understand the decision made by machine learning models. In this part we will try to find correlations or hidden features that could help to detect frauds.

Let’s begin our analysis by looking at the distributions of the variables :

	Time	V1	V2	V3	V4	...
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	...
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	...
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	...
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	...
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	...
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	...
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	...
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	...

As we can see with this table, the dataset is already preprocessed and cleaned, no missing values were found, and therefore no imputation is necessary. For features V1 to V28, the mean is around zero and the standard deviation around one, this means that those features are already normalized.
However, we can notice that for the features : Time and Amount, a scaler is necessary to normalize the data.

Then, a feature that needs to be analyzed deeply, is the amount of the transactions. We want to know if we can find hidden patterns by looking at the distribution :



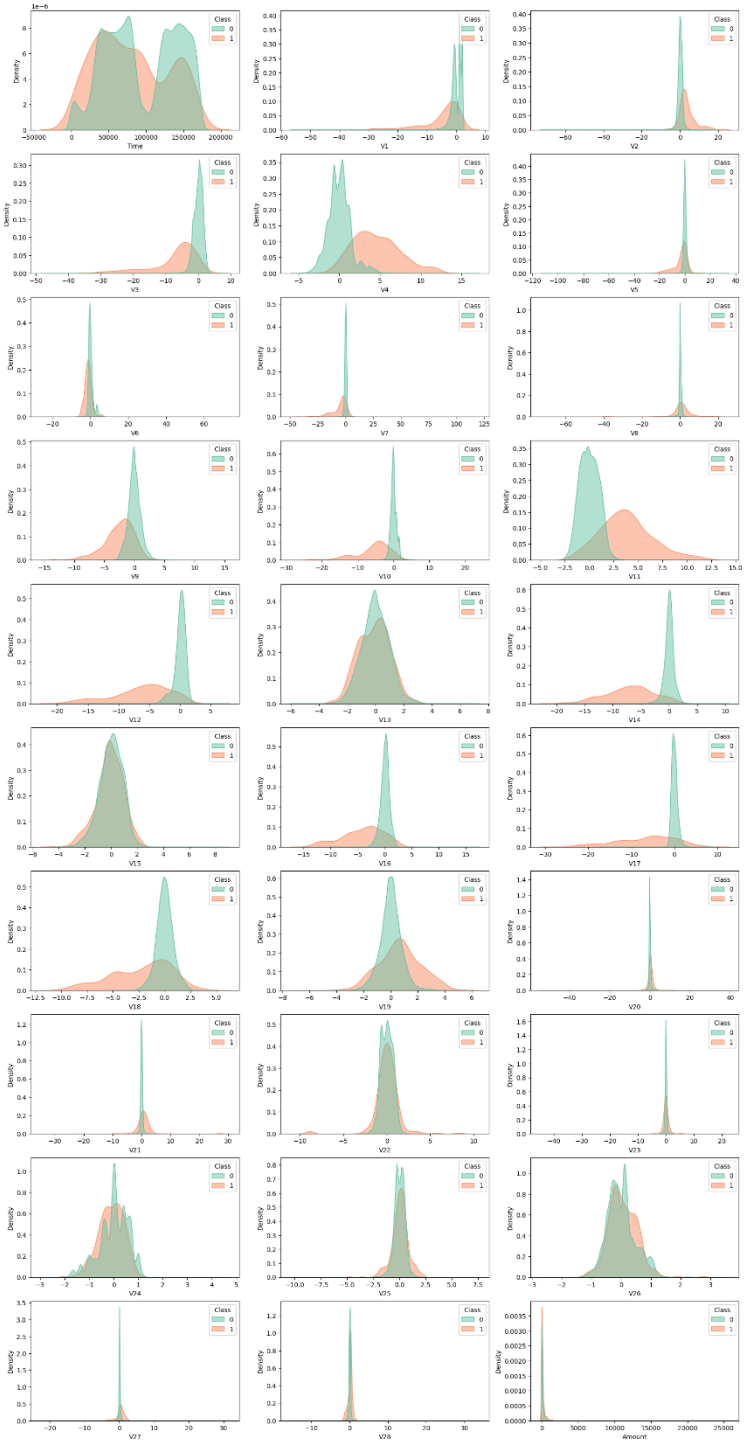
Amount		
	Normal	**Fraudulent**
Mean	88.3	122.2
Std	250.1	256.6

As we can see, fraudulent transactions (class 1) tend to have lower medians and fewer high-value outliers, whereas legitimate transactions exhibit a wider range, with higher median values and more extreme amounts. However, fraudulent transactions have a higher average amount compared to normal transactions, indicating that fraudulent activities tend to involve slightly larger sums on average.

To better understand the distributions of the dataset, we examined the histograms of each features to see if there is a difference of density depending on the class

Amount & Similar Features

The Amount feature, along with V13, V15, V22, V24, V25, V26, and V28, exhibits nearly identical distributions for both classes. This similarity indicates that these features hold low predictive power for classification tasks. When the density plots of two classes overlap significantly, it becomes challenging to differentiate between them based on these features alone. Consequently, they contribute minimally to the effectiveness of a classification model.



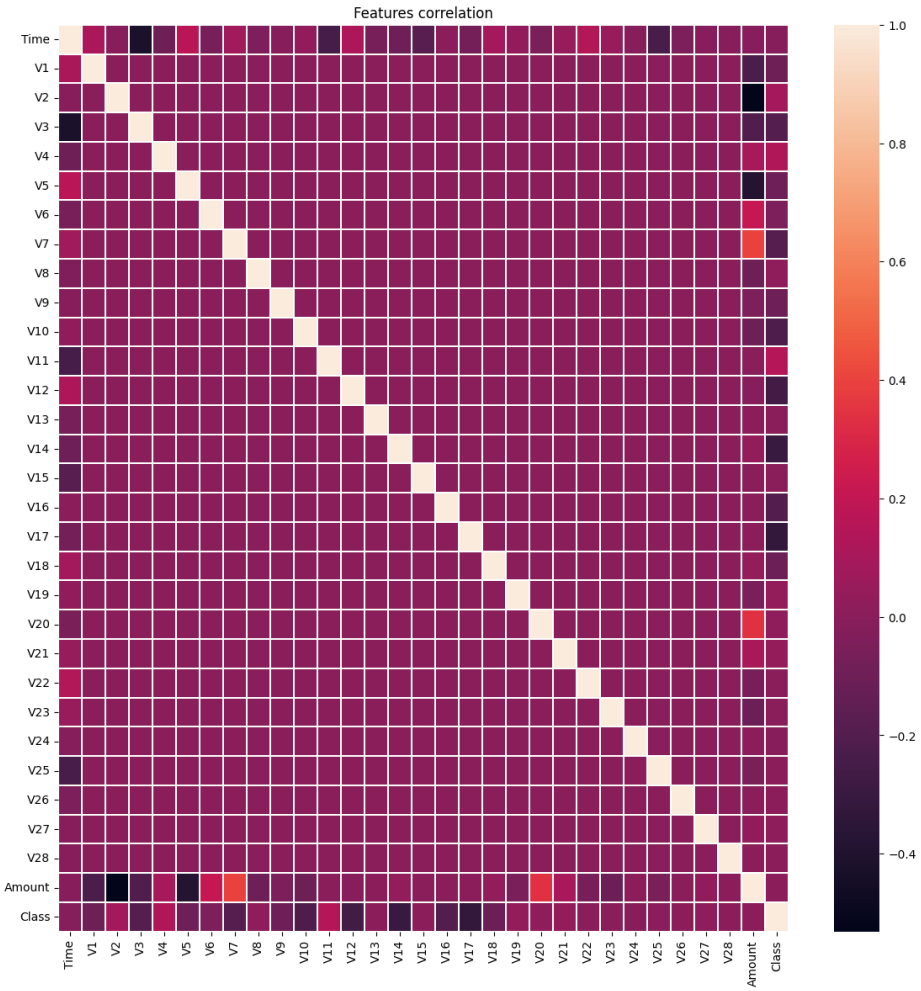
Long-Tailed Features

Features such as V1, V2, V5, V6, V7, V8, V20, V21, V23, and V27 display a distinct long tail in the distribution for class 0, which is absent in class 1. This characteristic suggests some potential for classification, as the difference in distribution shape could help distinguish between the two classes. However, despite this potential, there remains a notable overlap between the distributions, which may limit their effectiveness in achieving high classification accuracy.

Highly Distinct Distributions

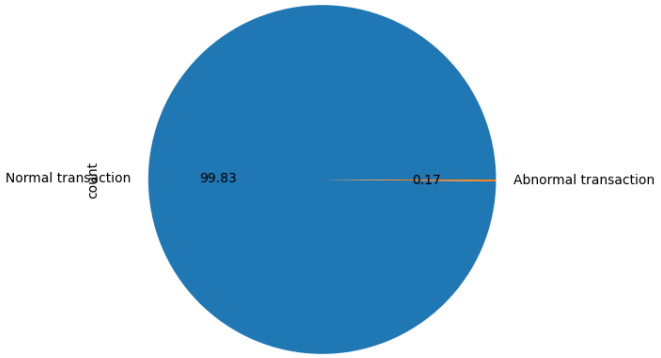
The most promising features for classification are V3, V4, V9, V10, V11, V12, V14, V16, V17, and V18. These features demonstrate **highly distinct distributions** between the two classes, characterized by **different means** and **larger standard deviations**. Such differences make them **strong candidates** for predicting class membership. The clear separation in their distributions allows for a more reliable distinction between classes, enhancing the performance of classification models.

To better understand the structure of the dataset, we examined the correlation matrix of all numerical features.



The features V1 to V28 are largely uncorrelated with each other, which is expected since they are principal components resulting from PCA. The feature Amount shows moderate correlation with V7 and V20, and inverse correlation with V2 and V5. As for our main target, Class, no strong correlations were observed with any single feature. There is only a slight inverse correlation with V12, V14, and V17, but the coefficients are too weak to be used as reliable predictors on their own.

One of the core difficulties in working with this dataset is the extreme class imbalance. Standard classification algorithms tend to favor the majority class, which in this case is legitimate transactions. As a result, models may achieve high overall accuracy while completely failing to detect fraud.



This imbalance could strongly impact the performance of classifiers, particularly those that are not explicitly designed to handle it. To mitigate this, techniques such as oversampling the minority class (e.g., using SMOTE or RandomOverSampler), under sampling the majority class, or using anomaly detection models are commonly applied. It can help rebalance the dataset and enabling the model to better learn the characteristics of fraudulent behavior.
Cost-sensitive learning, where higher penalties are assigned to false negatives (i.e., missed frauds), is also an effective strategy.

Finally, due to the dataset's imbalance, evaluation metrics such as accuracy become misleading. For example, a model that always predicts "not fraud" would achieve over 99.8% accuracy while being entirely useless. Therefore, we place greater emphasis on recall the proportion of actual frauds correctly identified as missing a fraudulent transaction is more costly than investigating a false alarm. Precision, F1-score, and the ROC-AUC are also considered to provide a balanced assessment of model performance.

This section sets the foundation for the methods explored in the following parts, where we compare several machine learning models and evaluate their ability to detect fraud under these real-world constraints.

4. Methodology and results

4.1. Models Selection

Before we get started with the model training it should be mentioned that the data has been split into a training and testing set (80% to 20%) and scaled with the StandardScaler function. Furthermore, in order to deal with the data imbalance we will try to do data augmentation by using a RandomOverSampler. We will evaluate a classifier with the original dataset and then with the augmented dataset to see if the results are improved or not.

When tackling credit card fraud detection with severely imbalanced datasets, several methodological approaches exist, each with distinct advantages and limitations. Traditional machine learning methods like Decision Trees offer interpretability and fast training times, making them ideal for initial exploration and understanding feature importance. However, they tend to overfit on minority classes and may not capture complex non-linear patterns effectively. Ensemble methods such as Random Forests and XGBoost address some of these limitations by combining multiple learners, which improves robustness and generalization. On the other hand, Deep Neural Networks can model highly complex, non-linear relationships and have shown remarkable performance in pattern recognition tasks. Their main drawback is the need for significant computational resources and longer training times, plus they can be harder to interpret compared to tree-based models. Other ML algorithms could have been considered but were less appropriate for our specific problem. Logistic Regression, while being simple, fast, and highly interpretable, struggles with the non-linear relationships present in our PCA-transformed features and would likely underperform compared to more complex models. Support Vector Machines with non-linear kernels could theoretically capture complex patterns,

but they scale poorly with large datasets like ours (284,807 transactions), and the computational cost of training becomes huge. K-Nearest Neighbors was also considered but rejected because it is highly sensitive to the curse of dimensionality, with 30 features, the notion of "nearness" becomes less meaningful, and the algorithm would likely be biased toward the majority class unless we applied complex distance weighting schemes.

Therefore, we decided to train a Decision Tree model, a Random Forest, XGBoost and a Deep Neural Network. This diverse selection allowed us to evaluate both simpler, interpretable models and more sophisticated, high-performance algorithms that are appropriate for this task.

4.2. Decision Tree

We begin our experiments with a simple Decision Tree classifier. The Decision Tree algorithm works by recursively splitting the feature space based on information gain or Gini impurity, creating a tree-like structure where each node represents a decision based on a feature threshold. Its main strength is interpretability, we can easily visualize and understand every decision the model makes. However, single trees are prone to overfitting, especially with imbalanced data.

As our main objective is to detect fraudulent transactions, we will prioritize ***recall*** over other metrics. Missing a fraud is far more critical than wrongly flagging a legitimate transaction. Some false positives are acceptable as long as their number remains low.
We first train the model using the original dataset (without augmentation). Below is the confusion matrix:

Actual \ Predicted	*Positive*	*Negative*
Positive	56842	22
Negative	24	74

```
Classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.77      0.76      0.76        98

    accuracy                           1.00     56962
   macro avg       0.89      0.88      0.88     56962
weighted avg       1.00      1.00      1.00     56962


AUC score:
 0.8773575764014102
```

These results are promising for an initial model without any tuning. However, the recall on fraud detection is still slightly below expectations.

Next, we apply the same model to the augmented dataset to evaluate whether data augmentation improves performance:

Actual \ Predicted	*Positive*	*Negative*
Positive	56839	25
Negative	27	71

```
Classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.74      0.72      0.73        98

    accuracy                           1.00     56962
   macro avg       0.87      0.86      0.87     56962
weighted avg       1.00      1.00      1.00     56962


AUC score:
 0.8620250752242372
```

As we can see with the AUC score, the results are a bit lower than before. It means that using an augmented dataset won't be useful because it will slow the execution and not give better results.

4.3. Random Forest Classifier

We now move on to the Random Forest model, which is known for its robustness and ability to handle complex data patterns. This algorithm is training multiple decision trees on random subsets of features and data, then aggregating their predictions through voting. This bagging approach reduces variance and makes the model more robust, though it sacrifices some interpretability and increases computational cost.

We will use GridSearch to find the best set of hyperparameters. To keep execution time manageable, we restrict our hyperparameter tuning to the following values:

- n_estimators ∈ {50, 100}
- max_depth ∈ {3, 10}

The best configuration found through grid search is:

- n_estimators = 100
- max_depth = 10

We evaluate the model under two conditions: first with default settings, and then with a manually adjusted threshold to improve recall.

Actual \ Predicted	*Positive*	*Negative*
Positive	56862	2
Negative	22	76

Without threshold tweaking

Actual \ Predicted	*Positive*	*Negative*
Positive	56834	30
Negative	14	84

With threshold tweaking

The results are better with threshold tweaking. Here is its classification report and the AUC score to visualize model performance:

```
Classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.74      0.86      0.79        98

    accuracy                           1.00     56962
   macro avg       0.87      0.93      0.90     56962
weighted avg       1.00      1.00      1.00     56962


AUC score:
 0.9283076412894928
```

We got a recall of 86% by keeping a precision of 74% which is really good. The AUC score is also better than before. This classifier is really promising.

4.4. XGB classifier

Next, we worked on the XGBoost model, a high-performance boosting algorithm for unbalanced data. XGBoost takes a different approach using gradient boosting, where trees are built sequentially, each one trying to correct the errors of the previous trees. It's incredibly powerful for structured data and includes several features that make it perfect for fraud detection.

We now apply a grid search to optimize the hyperparameters of the XGBoost model. The best configuration obtained is:

- n_estimators = 200
- subsample = 0.5

Then we set the following main parameters:

- scale_pos_weight = 5: This parameter increases the weight assigned to the minority class, in this case fraud.
- random_state = 42: This parameter allows for reproducible results by setting the random generation seed.
- use_label_encoder = False: This parameter disables the old label encoding.
- eval_metric = 'logloss': This metric measures the classification error based on predicted probabilities; the lower the logloss, the better the model.

We then applied a decision threshold adjustment strategy.

Indeed, by default, a transaction is considered fraudulent if the probability is greater than 0.5. However, by lowering this threshold to 0.03, the model becomes more sensitive to fraud. And we obtain the following Confusion Matrix.

Actual \ Predicted	*Positive*	*Negative*
Positive	56696	168
Negative	10	88

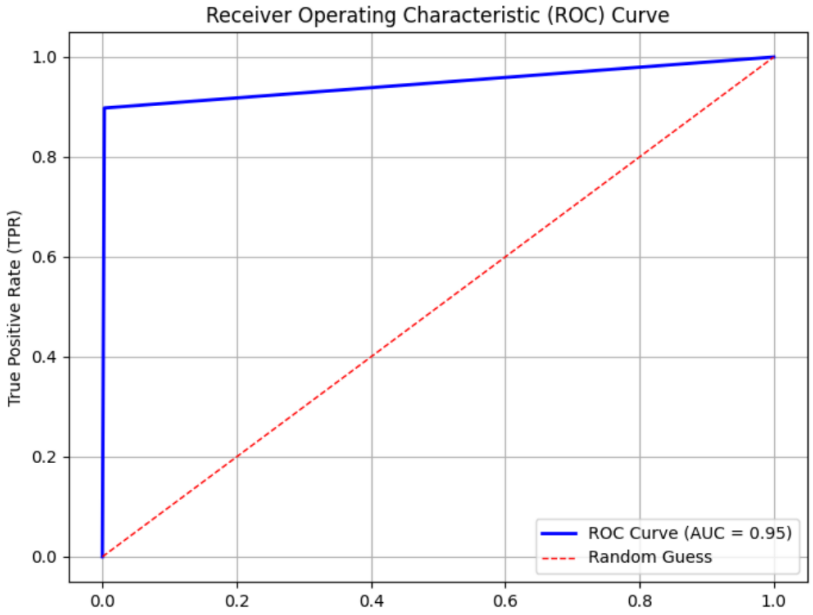
We also evaluated the model with the classification report :

```
Classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.34      0.90      0.50        98

    accuracy                           1.00     56962
   macro avg       0.67      0.95      0.75     56962
weighted avg       1.00      1.00      1.00     56962
```

By setting a threshold we have now reached our target values, a recall of 90% and a precision of 34%. And we also got an AUC score of 0.9475 which is pretty good.



We can see with the confusion matrix that even if the precision is low, the number of normal transactions that were missclassified as fraudulent is really low compared to the total number of normal transactions. Also, the number of fraudulent transactions that were missclassified as normal is quite low, so those results are really good for our application.

Although this accuracy is low, this compromise is acceptable in banking context. It's better to check a false alarm than to miss a real fraud. Therefore, XGBoost has proven to be an excellent model, outperforming decision trees and random forests.

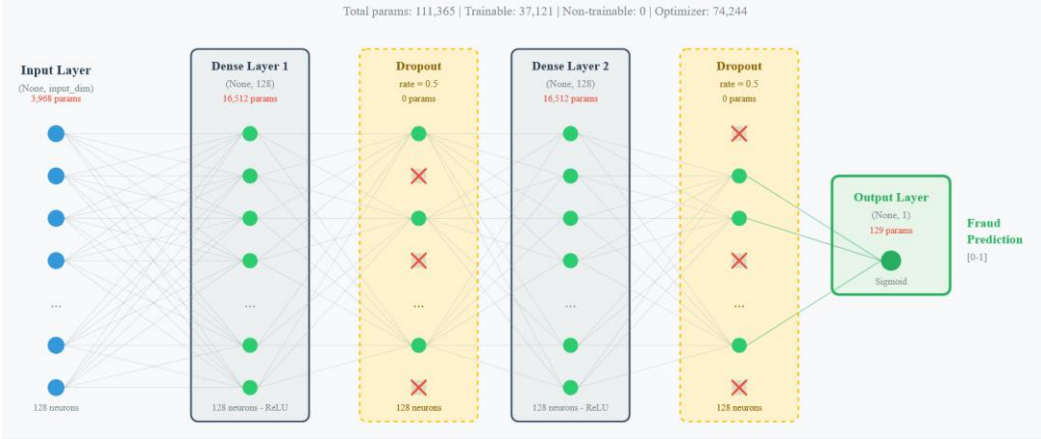
4.5. Deep Neural Network

Even if this course mostly focus on machine learning algorithm, we decided to train a deep neural network to compare the results and assess whether it can improve our results further.

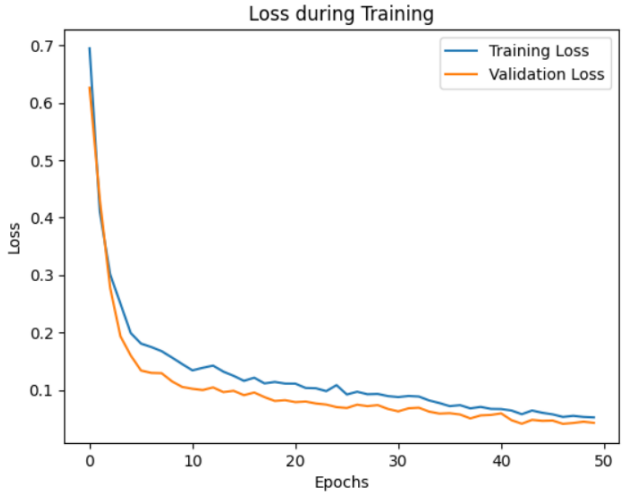
We tried different architectures of DNN until we found the one with the best results. Our final architecture is the following :

Our model is a sequential neural network composed of:

- Three hidden layers, each with 128 neurons
- ReLU activation functions
- Dropout between layers to prevent overfitting
- Adam optimizer
- A learning rate of 10^{-4}
- One sigmoid-activated output neuron for binary classification
- To account for class imbalance, we compute class weights and integrate them into the model during training.

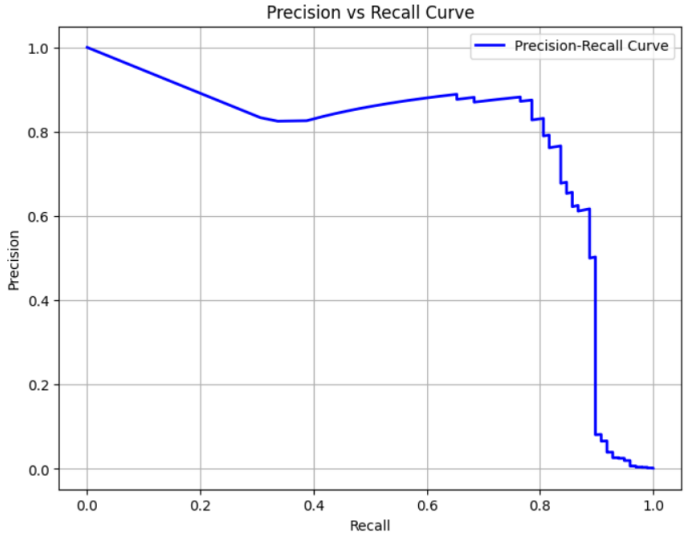


We trained the model for 50 epochs using a batch size of 2048 samples, and applied callbacks for performance monitoring. The following plot shows the evolution of the loss during training:



We can observe that the curves decrease rapidly at the beginning and continue to decrease as the epochs progress. This proves to us that there is no sign of overfitting.

To evaluate classification performance, we analyze the precision-recall curve :



Based on that curve we can expect really good results by using a threshold, so let's predict our test set and put a threshold to get our target recall. We can plot the confusion matrix, the classification report and the AUC score to visualize model performance:

Actual \ Predicted	*Positive*	*Negative*
Positive	56719	145
Negative	10	88

```
Classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.38      0.90      0.53        98

    accuracy                           1.00     56962
   macro avg       0.69      0.95      0.77     56962
weighted avg       1.00      1.00      1.00     56962


AUC score:
 0.9477046199740448
```

We reached our target values and the AUC score is really good. This was our best results so far so this model is really efficient.

5. Conclusion

In this project, we set out to explore how machine learning can be effectively applied to credit card fraud detection, a task made particularly difficult by the extreme imbalance of legitimate and fraudulent transactions. By carefully selecting evaluation metrics such as recall, precision, and AUC score, and by applying resampling techniques to mitigate class imbalance, we created a consistent and fair evaluation pipeline for several models.

We implemented and assessed a variety of models, ranging from traditional classifiers like Decision Trees and Random Forests to more advanced approaches such as XGBoost and Deep Neural Networks (DNN). Each model was evaluated on both the original dataset and an augmented version using RandomOverSampler, to measure their ability to identify rare fraudulent cases.

Among all the models tested, the Deep Neural Network consistently delivered the best performance. Thanks to its ability to model non-linear relationships and capture subtle patterns in the data, the DNN achieved superior results in key metrics such as recall and AUC, while maintaining solid precision. Its robustness to the class imbalance and generalization to unseen data made it the most effective solution for our problem.

Nonetheless, other models also demonstrated strong performance. In particular, the XGBoost classifier achieved excellent results with significantly shorter training times, making it a highly practical alternative in resource-constrained or latency-sensitive environments.

In summary, our work highlights the importance of model selection, evaluation strategies, and preprocessing in solving real-world problems with imbalanced data. By combining sound machine learning practices with thoughtful metric design, we successfully identified models capable of detecting fraud with high reliability a critical need in the financial sector.