Fraud Detection in Digital Transactions: Stage 3 Report

Tom Delahaye, Gabriel Carlotti, Pierre Briand, Kentin Guillemot

December 8, 2024

Abstract

This report presents the third stage of a multi-phase project focused on fraud detection in digital payments. In this stage, we implemented a deep learning model optimized with **Optuna** and explored an **ensemble learning approach with stacking** using the best models from previous stages: Logistic Regression, KNN, Random Forest, and XGBoost. The results highlighted the challenges of overfitting in stacking models and the importance of hyperparameter tuning and regularization techniques to achieve optimal performance.

Contents

1 Problem Formalisation and Methods

1.1 Algorithm Description

In this stage, we implemented the following models:

- **Deep Learning Model:** A feedforward neural network optimized using Optuna for hyperparameter tuning.
- **Stacking Model:** An ensemble model where the base learners consisted of Logistic Regression, KNN, Random Forest, and XGBoost, with a deep learning model acting as the meta-learner.

1.2 Limitations

The initial stacking approach exhibited overfitting issues due to the complexity of the deep learning model. These issues were mitigated by introducing dropout regularization, batch normalization, and early stopping.

2 Methodology

2.1 Data Preprocessing

Step 1: Simple Neural Network Model For the initial neural network model, we opted for a straightforward preprocessing approach to evaluate the model's performance without introducing unnecessary complexity. The following preprocessing techniques were applied:

- **Standardization:** We standardized the features to have a mean of 0 and a standard deviation of 1. This helps the neural network converge faster and achieve better performance.
- **No SMOTE:** We did not apply SMOTE with neural networks because it can introduce synthetic examples that add noise and redundancy, which may lead to overfitting. Neural networks already have a high learning capacity, making SMOTE unnecessary and potentially harmful to generalization.
- **No Outlier Removal:** We chose not to remove outliers. Neural networks are capable of learning complex patterns and can handle outliers effectively. Removing outliers could result in losing valuable information that the model might learn from.

Step 2: Stacking Model with Machine Learning Base Models For the stacking model, we combined the predictions from the models used in the previous stages: Logistic Regression, KNN, Random Forest, and XGBoost. We applied the same preprocessing techniques as in the previous stages to ensure consistency and optimal performance for these base models:

- **Outlier Removal:** We removed outliers using the Z-score method to prevent extreme values from negatively affecting the performance of the base models.
- **SMOTE:** We applied SMOTE to balance the dataset, addressing the class imbalance and enhancing the models' ability to detect fraudulent transactions.
- **Standardization:** The features were standardized to ensure the models perform well, particularly for distance-based models like KNN and gradient-based models like Logistic Regression.

The meta-learner for the stacking model was a deep learning model optimized with Optuna.

—

2.2 Neural Network Optimization

We optimized the hyperparameters of both the initial neural network and the meta-learner of the stacking model using **Optuna** with the objective of maximizing recall. The key hyperparameters tuned were:

- **Number of Layers:** Between 2 and 5 hidden layers.
- **Number of Units per Layer:** The number of neurons in each hidden layer.
- **Dropout Rates:** To prevent overfitting by randomly deactivating neurons during training.
- **Learning Rate:** The learning rate for the Adam optimizer.
- **Batch Size:** The size of the mini-batches used during training.

—

2.3 Addressing Overfitting in the Stacking Model

The results of the stacking model revealed slight overfitting, as indicated by the near-zero training loss and a high training recall compared to the validation recall. To mitigate this, we made the following adjustments:

- **L2 Regularization:** Added to the dense layers to penalize large weights and promote simpler models.
- **Increased Dropout Rates:** Increased the dropout rates to further improve regularization by randomly deactivating neurons during training.
- **Early Stopping Adjustment:** Reduced the early stopping patience to 3 epochs to halt training sooner when the validation loss stops improving, preventing unnecessary training and overfitting.

These adjustments helped improve the model's generalization performance while maintaining a high recall.

3 Results and Evaluation

3.1 Performance Metrics

We evaluated the models using the following metrics:

- **Recall**
- **F2-Score**

3.2 Neural Network Model Results

- The neural network model optimized with Optuna showed strong recall but exhibited signs of overfitting.
- Adjustments such as increased dropout rates and early stopping helped mitigate overfitting.

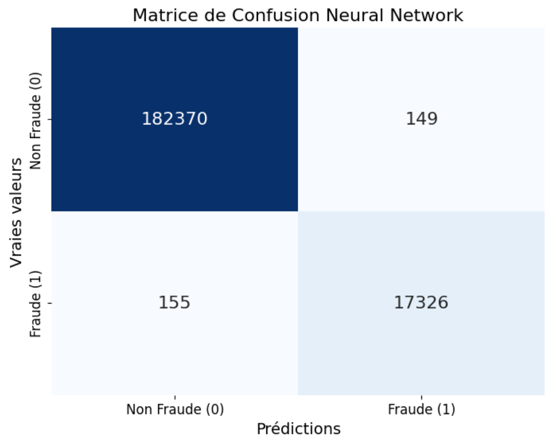

Figure 1: Confusion Matrix for the Neural Network Model

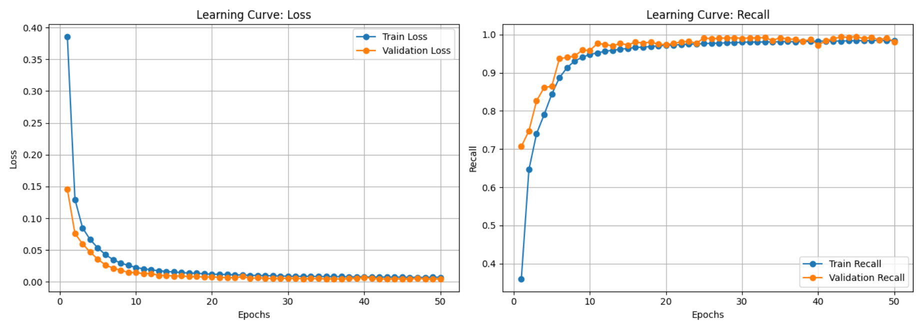

Figure 2: Learning Curves for the Neural Network Model (Loss and Recall)

3.3 First Stacking Model Results

- The initial stacking model combined Logistic Regression, KNN, Random Forest, and XGBoost with a deep learning meta-learner.
- This model initially faced slight overfitting, with a low training loss and near-perfect training recall.

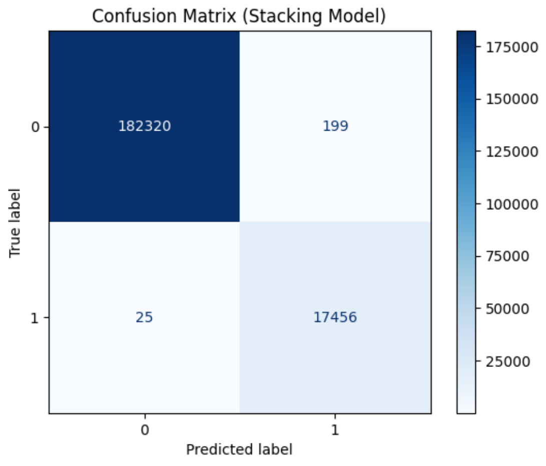

Figure 3: Confusion Matrix for the First Stacking Model

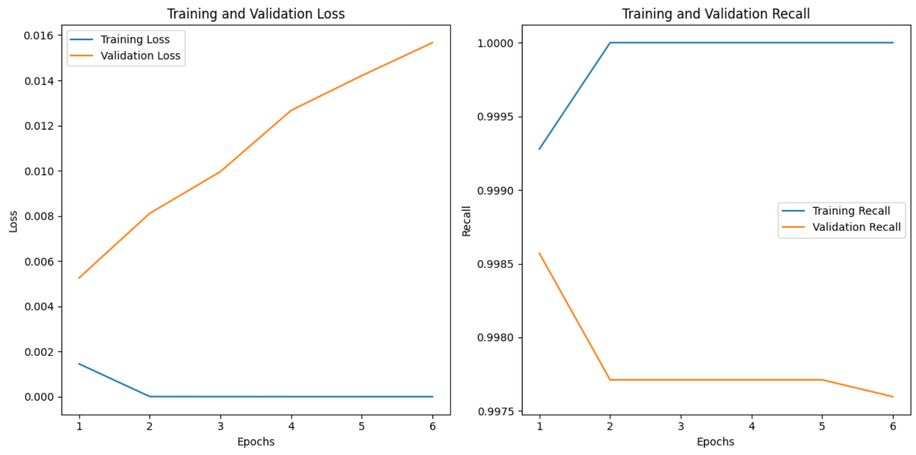

Figure 4: Learning Curves for the First Stacking Model (Loss and Recall)

3.4 Second Stacking Model Results

- In the second stacking approach, additional regularization and hyperparameter tuning were applied to address overfitting.
- This led to improved generalization, reducing both false positives and false negatives while maintaining high recall.

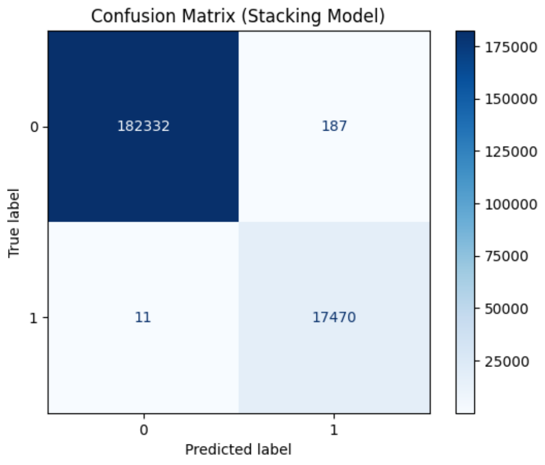

Figure 5: Confusion Matrix for the Second Stacking Model

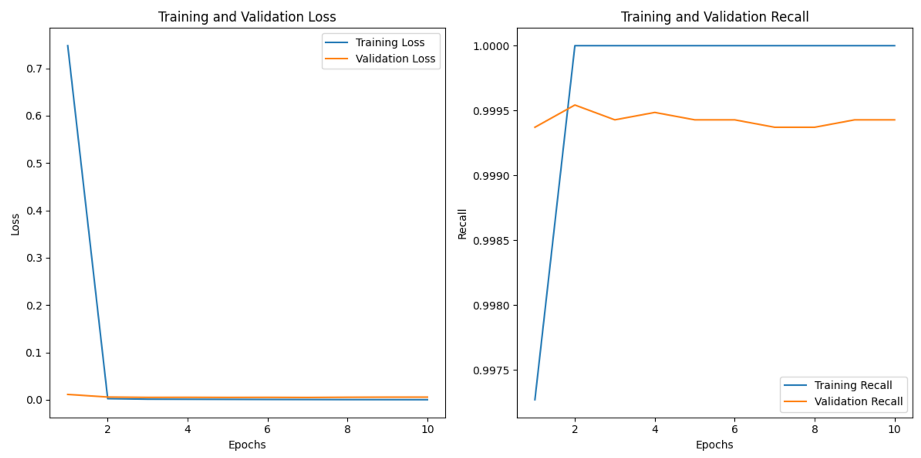

Figure 6: Learning Curves for the Second Stacking Model (Loss and Recall)

4 Discussion and Conclusion

In this stage, we implemented and optimized a deep learning model and explored an ensemble learning approach using stacking. The key results and insights are summarized below:

- The **Neural Network Model** demonstrated strong recall and performed consistently well on both the training and validation sets. The use of dropout layers, batch normalization, and early stopping contributed to maintaining good generalization.
- The **First Stacking Model** combined Logistic Regression, KNN, Random Forest, and XGBoost with a deep learning meta-learner. This model experienced slight overfitting due to the complexity of the meta-learner.
- The **Second Stacking Model** improved upon the first by applying additional regularization and hyperparameter tuning, resulting in better generalization and reduced overfitting.
- Despite these improvements, the **Random Forest** and the **Stacking Model from Stage 2** remained the best-performing models, achieving the best balance between precision and recall.

This stage marks the conclusion of our exploration into fraud detection using machine learning and deep learning techniques. The insights gained highlight the strengths and limitations of each approach. While deep learning and stacking methods offer promising results, traditional models such as Random Forest continue to provide reliable performance with balanced metrics.

Our work underscores the importance of careful preprocessing, hyperparameter tuning, and regularization in developing effective fraud detection models.