Optimizing Hyperparameters and Feature Selection for Credit Card Fraud Detection

1st Ilya Grigorev
DS-01
Innopolis University
il.grigorev@innopolis.university

2nd Salavat Faizullin
DS-01
Innopolis University
s.faizullin@innopolis.university

I. Introduction

Credit card transactions are one of the financial instruments that are vulnerable to different types of fraud resulting in substantial financial losses [3]. Transactions are described by a wide range of information and are associated with many other attributes, such as identity information. Thus, fraud detection can be seen as distinguishing the usual transaction from rare based on data with a large number of features, which are difficult to process. Not all features have a significant impact on the investigation. Therefore, efficient selection of attributes is crucial for fast and reliable detection. In addition, the imbalance between fraudulent and normal transactions occurs when collecting data for analysis. Such an imbalance can lead to supervised machine learning (ML) models having poor performance [3]. Supervised machine learning is one of the powerful approaches for fraud detections, but other methods exist, such as outlier detection [4].

In this paper, we designed a machine learning training pipeline for fraud detection based on a public dataset from the competition platform Kaggle. The dataset is compirsed of more than 590,000 transaction with more than 300 features to analyze. To optimize the training, two processes are performed. First, feature selection is performed to find the optimal set of features for the model to effectively detect fraud. Among many methods, nature-inspired algorithms are known to provide suboptimal solutions and converge relatively fast [macinec]. Particularly, we implemented this step using genetic algorithms (GA). Second, since different ML models have specific sets of hyperparameters that require configuration for better performance [2], hyperparameter optimization is configured using another nature-inspired algorithm called Particle Swarm Optimization (PSO). This optimization is performed to tune the following models: Random forest, XGBoost, LightGBM, and dense neural network (DNN).

II. Related Work

Recent studies demonstrate the effectiveness of nature-inspired algorithms (NIAs) in machine learning pipeline optimization. Key contributions are summarized below.

P. Macinec and T. Zatko [1] investigated feature selection using multiple NIAs including Firefly Algorithm, Cuckoo Search, Bat Algorithm, Flower Pollination Algorithm, and Grey Wolf Optimizer. Their methodology addressed class imbalance through undersampling by matching minority and majority class sizes. While some algorithms exhibited convergence issues in feature selection scores, all NIAs collectively improved ROC-AUC performance compared to baseline methods. This work establishes NIAs as viable alternatives to conventional feature selection techniques.

S. Reddy et al. [2] achieved exceptional results (ROC-AUC: 0.997) using Elephant Herd Optimization for XGBoost hyperparameter tuning, though on synthetic data. Their findings suggest XGBoost and LightGBM as particularly suitable for NIA-enhanced optimization due to their parameter-sensitive performance characteristics. The study highlights the potential for combining gradient-boosted trees with biologically inspired optimization methods.

N. Mqadi et al. [3] proposed an alternative approach to class imbalance through oversampling using nearest-neighbor techniques. Their results showed significant improvement in positive class prediction accuracy, with Random Forest demonstrating particular robustness to the resampled data distribution. This complements earlier undersampling approaches while maintaining model performance stability.

The literature review yields three principal conclusions:

- Random Forest, XGBoost, and LightGBM emerge as preferred models for NIA-enhanced pipelines, with DNN included for comparative analysis
- Genetic Algorithms remain underutilized in feature selection compared to other NIAs, warranting investigation
- While modified PSO variants exist, the standard algorithm provides a robust baseline for hyperparameter optimization

Consistent with these studies, ROC-AUC serves as the primary optimization metric and evaluation criterion in the present work. This choice aligns with established practice in imbalanced classification tasks and facilitates direct comparison with referenced results. Additionally, we use undersampling technique for resolving dataset imbalance due to limited resources.

III. Methodology

Python served as the primary tool for data analysis and model development. The methodology consisted of dataset preprocessing, feature selection using a genetic algorithm

(GA), hyperparameter optimization via particle swarm optimization (PSO), and model evaluation.

A. Dataset Exploration and Preprocessing

The dataset preprocessing followed these steps:

1) Two CSV files were loaded and merged into a single pandas DataFrame.
2) Columns with missing values exceeding an 80% threshold were removed.
3) Missing values were imputed using the median for numerical features and the label 'MISSING' for categorical features.
4) Kernel density estimation (KDE) plots analyzed numerical feature distributions, while count plots with cumulative sums visualized categorical distributions.
5) Redundant features (`TransactionID`, `DeviceInfo`, `id31`) were excluded.
6) Categorical features were encoded: ordinal encoding for `P_emaildomain`, `R_emaildomain`, and M-series features (M1–M9), and one-hot encoding for remaining categories. Categories with frequencies below 3% were grouped as `RARE_CATEGORY`.
7) Features were normalized using `MinMaxScaler`.
8) The processed dataset was saved in parquet format.

B. Dataset Undersampling

Class imbalance was addressed through undersampling, following the pipeline:

1) Load parquet data with optional feature filtering
2) Split data into training (80%) and test sets (20%)
3) Apply undersampling to the training set
4) Further split the training set by a specified fraction (1.0 for GA, 0.3 for hyperparameter tuning) to reduce computational load

C. Feature Selection Using Genetic Algorithm

The DEAP framework implemented the GA with the configuration in Table I.

TABLE I
GENETIC ALGORITHM PARAMETERS

Parameter	Value
Population size (n_pop)	50
Generations (n_gen)	20
Crossover probability (cxbp)	0.5
Mutation probability (mutpb)	0.2

Each individual was represented as a binary feature mask. Fitness evaluation used a random forest classifier with parameters:

- `n_estimators=50`
- `max_depth=5`
- `class_weight='balanced_subsample'`
- `n_jobs=-1`
- `random_state=42`

The fitness score combined ROC-AUC performance (80% test set, 20% validation) with a penalty term: $0.1 \times \frac{\text{selected features}}{\text{total features}}$.

D. Hyperparameter Tuning Using PSO

Four models underwent PSO-based optimization with the parameters in Table II.

TABLE II
PSO CONFIGURATION

Parameter	Value
Particles (n_particles)	5
Iterations (iters)	10
Cognitive coefficient (c1)	0.5
Social coefficient (c2)	0.3
Inertia weight (w)	0.9
Early stopping	True
Patience	3

The search spaces for each model's hyperparameters are detailed in Table III.

TABLE III
HYPERPARAMETER SEARCH RANGES

Model	Parameters
Random Forest	`n_estimators`: [50,500], `max_depth`: [2,30]
	`min_samples_split`: [2,10], `max_features`: [1,30]
XGBoost	`learning_rate`: [0.01,0.3], `max_depth`: [3,10]
	`min_child_weight`: [0.1,10], `subsample`: [0.1,1]
	`colsample_bytree`: [0.1,1], `gamma`: [0, 5]
LightGBM	`learning_rate`: [0.01,0.3], `num_leaves`: [3,50]
	`min_data_in_leaf`: [0.1,100]
	`feature_fraction`: [0.1, 1], `bagging_fraction`: [0.1, 1]
	`bagging_freq`: [20, 50]
DNN	`learning_rate`: [0.0001,0.01], `layer1`: [16,256]
	`layer2`: [16,256], `dropout_rate`: [0.1,0.5]

The objective function measured the average ROC-AUC score from 3-fold cross-validation.

E. Evaluation Metrics

Each model was assessed with and without undersampling using:

- ROC-AUC and PR-AUC scores
- Precision, recall, and F1-score
- Feature importance (for tree-based models)
- Confusion matrices

F. Repository

The source code and the results reports can be found in the the project repository by following the link. The research steps are documented thoroughly in the jupyter notebooks in the notebooks directory.

IV. EXPERIMENTS AND EVALUATION

All experiments were conducted in a Google Colab environment with necessary libraries installed and data properly configured. Detailed pictures of the results as well as text reports can be found in the repository folder.

TABLE IV
GENETIC ALGORITHM FITNESS PROGRESSION

Generation	Evaluations	Avg Fitness	Min Fitness	Max Fitness
0	50	0.783033	0.768537	0.798525
10	26	0.804404	0.797343	0.806865
20	26	0.808826	0.800840	0.809949

A. Feature Selection

The genetic algorithm completed 20 generations with fitness values progressing as shown in Table IV. The convergence plot (Figure 1) demonstrates steady improvement in average fitness from 0.783 to 0.809.

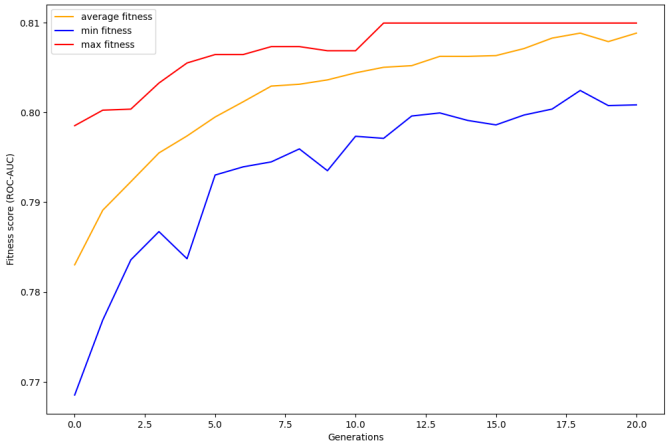

Fig. 1. Fitness Convergence by 20 Generations.

The optimal feature subset comprised 155 indices from the original (preprocessed) feature space. The full list is available in the repository.

B. Hyperparameter Optimization

Table V presents the best parameters discovered through PSO optimization for each model:

TABLE V
OPTIMIZED MODEL PARAMETERS

Model	Optimal Parameters
Random Forest	`n_estimators`: 478, `max_depth`: 26 `min_samples_split`: 5, `max_features`: 28
XGBoost	`learning_rate`: 0.284, `max_depth`: 7 `min_child_weight`: 6.172, `subsample`: 0.988 `colsample_bytree`: 0.888, `gamma`: 0.328
LightGBM	`learning_rate`: 0.219, `num_leaves`: 24 `min_data_in_leaf`: 37, `feature_fraction`: 0.425 `bagging_fraction`: 0.986, `bagging_freq`: 29
DNN	`learning_rate`: 0.0056, `layer1`: 210 `layer2`: 144, `dropout_rate`: 0.152

The optimization yielded ROC-AUC scores of 0.894 (Random Forest), 0.907 (XGBoost), 0.906 (LightGBM), and 0.873 (DNN) on validation data.

C. Model Evaluation

Table VI compares final model performance on test data, and Figures 2–5 demonstrate confusion matrices with feature analysis (for tree-based models):

TABLE VI
MODEL PERFORMANCE METRICS

Model	ROC-AUC	PR-AUC	F1 (Class 1)	Accuracy
Random Forest	0.9045	0.5389	0.27	0.84
XGBoost	0.9175	0.5607	0.27	0.83
LightGBM	0.9129	0.5578	0.30	0.86
DNN	0.7548	0.1599	0.10	0.45

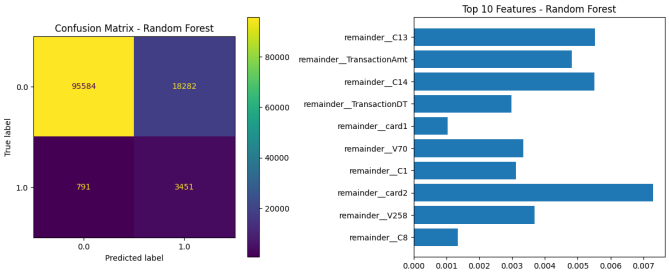

Fig. 2. Visualized performance for RandomForest

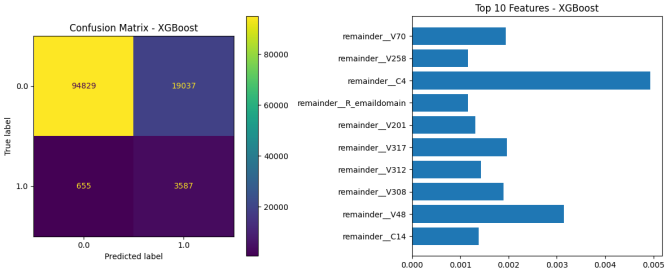

Fig. 3. Visualized performance for XGBoost

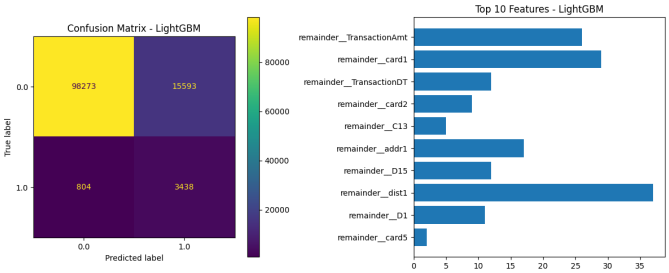

Fig. 4. Visualized performance for LightGBM

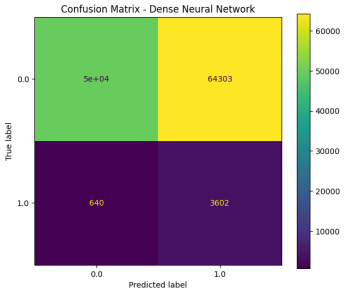

Fig. 5. Visualized performance for DNN

V. Analysis and Observations

Key observations:

- GA feature selection reduced the number of features by almost 60%.
- XGBoost achieved highest ROC-AUC (0.9175) and PR-AUC (0.5607).
- LightGBM showed best F1-score for minority class (0.30).
- DNN underperformed with 0.7548 ROC-AUC, suggesting architectural limitations.

Confusion matrices (Figures 2–5) reveal that tree-based models maintained high specificity (99% for class 0) while achieving 81-85% recall for class 1. Feature importance analysis (Figures 2–4) identified consistent predictive features across tree-based models.

Interestingly, most of the top features for all three models are distinct. This observation seems to be misleading for analysis of features because models with high performance should share some common top features. However, the reason for this outcome could be the limitation of the dataset size for model training.

Unfortunately, due to limited resources, the study lacks thorough investigation of results. Future research should focus more on testing different combinations of NIAs with feature selection similar to [1] and [2].

VI. Conclusion

Overall, this paper demonstrated the combination of two nature-inspired techniques for optimizing the performance of the four ML models for fraud transaction detection. The feature selection method using GA converged to the suboptimal subset of features reducing their number by more than 50%. Moreover, the hyperparameter search optimizer using PSO was used to configure hyperparameters for each model. The best model among suggested was XGBoost, while the worst performance demonstrated the dense neural network. Although the optimizations were performed the models did not score more than 90% accuracy on the test set, the reason for which could be the lack of data caused by the dataset undersampling. Future experimentations should perform other techniques like oversampling for achieving higher model performance with the use of NIAs.

References

[1] P. Macinec and T. Zatko, "Using Nature Inspired Algorithms for Feature Selection in Transaction Fraud Detection," GitHub repository.

[2] S. S. Reddy, K. Amrutha, V. M. Gupta, K. V. Murthy, and V. V. R. M. Rao, "Optimizing Hyperparameters for Credit Card Fraud Detection with Nature-Inspired Metaheuristic Algorithms in Machine Learning," *J. Inst. Eng. India Ser. B*, 2025, DOI: 10.1007/s40031-025-01207-2.

[3] N. Mqadi, N. Naicker, and T. Adeliyi, "A SMOTe based Oversampling Data-Point Approach to Solving the Credit Card Data Imbalance Problem in Financial Fraud Detection," *Int. J. Com. Dig. Sys.*, 2021, DOI: 10.12785/ijcds/100128.

[4] N. Malini and M. Pushpa, "Analysis on Credit Card Fraud Identification Techniques based on KNN and Outlier Detection," *Adv. Elect. Electron. Inf. Com. Bio-Informatics*, vol. 1, no. 1, pp 1-12, 2017. DOI: 10.1109/AEEICB.2017.7972424.