Case Study: Model Engineering

Title: Application and Comparison of Different Machine Learning Classifiers to Predict Financial Fraud

Date:- 16-07-2025

Author's Name:- Harshraj Shailendrasinh Chavda

Table of Contents

1. Abstract

This case study uses the extremely unbalanced ULB dataset (0.17% fraud incidences) to assess the Machine Learning techniques for credit card fraud detection. Random Forest and Logistic Regression classifiers were implemented using Sci-Kit Learn pipelines, utilising GridsearchCV optimisation and 5-fold Cross-Validation. In comparison, the Random Forest classifier performed better than Logistic Regression and reduced the number of missed fraud cases by 47%. Despite the flawless accuracy of valid transactions, the study draws attention to important issues with class imbalance and the interpretability of PCA features. The effectiveness of Random Forest for practical implementation is shown by the results; nonetheless, explainable AI approaches should be explored in future research.

2. Introduction

2.1 Background

2.1.1 Financial Impact of Credit Card Fraud

Financial Fraud includes deceptive practices intended for illicit financial gain, stretching across sectors like insurance, banking, and taxation. Among its various forms, especially within the banking sector, namely Money Laundering, Account-based frauds, Mobile Banking frauds, Cash Payment frauds, etc, frauds of Credit Card have emerged as a pervasive concern, causing significant monetary losses to individuals, businesses, and financial institutions. Annual losses due to fraudulent activities amount to hundreds of millions of dollars, highlighting the urgency of immediate, robust detection mechanisms on top of existing countermeasures, which have proven to be insufficient.

Occupational fraud is categorised into asset misappropriation, financial statement fraud, and corruption have disproportionately impacted economies as highlighted by The Association of Certified Fraud Examiners (ACFE). Even though the highest median losses (e.g., $800,000 per incident in 2018) are observed due to financial statement fraud, the high frequency of credit card fraud exacerbates the cumulative damages. For example, studies on detecting payment card fraud found that smaller-scale scams, when accumulated, strain the banking systems and erode consumer trust.

(Ashtiani & Raahemi, 2021)[1]

2.1.2 Advantages of Machine Learning in Credit Card Fraud Detection

Parallel with the rise of digital payment systems, especially credit card usage, an increase in sophisticated fraud techniques has occurred. Traditional fraud detection methods are reliant on manual audits, proving to be inefficient because of the costs, inaccuracies, and limitations of scalability associated with them. Modern approaches using Data Science, particularly machine

learning (ML) and data mining, offer transformative solutions. Transactional patterns are classified to be fraudulent or legitimate via analyses performed by supervised and unsupervised Machine learning.

Real-Time Detection

Immediate identification of suspicious activities is achieved via the processing of large-scale transactional data in real time through ML models. For instance, transactional features like amount, location, and frequency can be analysed within milliseconds through ensemble methods like bagging classifiers and deep learning architectures such as Convolutional Neural Networks (CNNs). Autoencoders and Generative Adversarial Networks (GANs) further enhance real-time processing by reconstructing transactional data and flagging anomalies based on deviations from normal patterns.

Adaptability to Emerging Fraud Patterns

Constant refinement in tactics, employing high-tech, cross-regional, and highly concealed methods, fraudsters can reduce their chances of detection. As Traditional systems lack dynamic learning mechanisms, they can falter when faced with unknown vectors.

ML models address this through:

- Unsupervised Learning: Elimination of dependence on labelled data via anomaly detection of fraud as an outlier problem.
- Future Attention Mechanisms: Novel fraud detected via the capture of subtle correlations is made possible through advanced frameworks (like UAAD-FDNet), which use channel-wise attention to weigh transactional features dynamically.

(Jiang et al., 2023)[2]

Data Imbalance

Fraudulent cases within the fraud dataset constitute a tiny fraction of all transactions, making fraud datasets inherently imbalanced. ML techniques like resampling, ensemble learning, and hybrid loss functions mitigate this issue. For example, improvement in the detection of the minority class in a skewed dataset is achieved by emphasising samples from the minority class in the training phase, via the use of LSTM networks in combination with AdaBoost.

(Velarde et al., 2024)[3] (He & Garcia, 2009)[4]

2.2 Problem Statement

Using the Credit Card Fraud Detection Dataset available at the Machine Learning Group on the Kaggle platform (Credit Card Fraud Detection, n.d.)[5]:

- Explore and study the behaviour of different classifiers.
- Observe the effects of Data Preprocessing.
- Perform optimisation to improve credit card fraud detection.

2.2.1 Objectives

1. Implement 2 classifiers of choice using pipelines in:
    - 5-fold cross-validation (CV) or
    - An 80-20 partition.

In the event of data preprocessing or feature engineering, ensure the use of pipelines to avoid overfitting. Justify your choices.

2. Perform Optimisation of selected classifiers using:
    - Exhaustive Grid Search or
    - Randomised Parameter Optimisation.

Justify parameter choices and optimisation method.

3. Present a comparison between the 2 classifiers' prediction accuracy in a tabular format in terms of:
    - Precision
    - Recall
    - F1 score

3. Method

3.1 Dataset

To ensure the prevention of charges for unauthorised transactions for customers, a real-world credit card fraud detection dataset is examined for this case study. The dataset consists of 284,807 transactions recorded over 2 days in September 2013 by European cardholders, with only 492 (i.e. 0.172%) transactions being fraudulent (as shown in Figure 1), highlighting a significant class imbalance in the dataset. In order to comply with confidentiality constraints, the original identifying features were anonymised using Principal Component Analysis (PCA), resulting in 28 numerical components (i.e. V1-V28).

Although few features were left untransformed, namely:

- **Class**: Binary Label (1 for fraud, 0 otherwise).
- **Amount**: Transaction value, useful for cost-sensitive learning.
- **Time**: Seconds elapsed after the first transaction.

The collaboration between Worldline and Université Libre de Bruxelles (ULB) curated the dataset, with detailed methodologies explained in a *Machine Learning for Credit Card Fraud Detection* handbook, which also provides features for experimentation via a synthetic transaction simulator. Works on adaptive learning, calibration of undersampling, and streaming fraud detection give background for this dataset by emphasising scalable and incremental machine learning approaches.

(Credit Card Fraud Detection, n.d.)[5] (Pozzolo et al., 2015)[6]

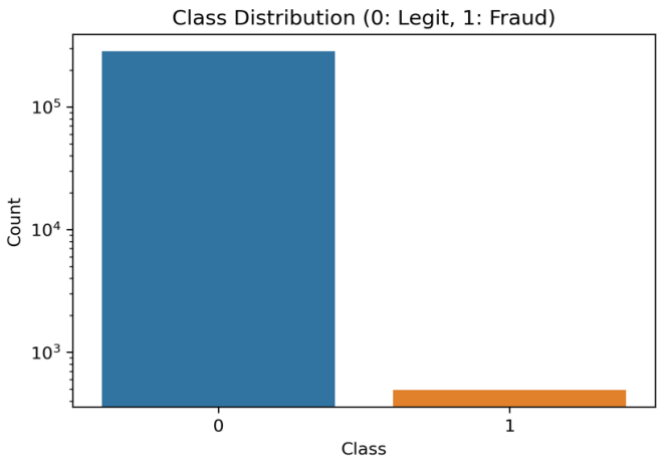

Figure 1 : Bar graph highlighting the severe lack of balance between instances of Fraud and Non-Fraud Transactions within the Dataset.

3.2 Data Preprocessing

This section covers the steps taken to prepare the dataset for model training by performing feature-target separation, train-test splitting, and feature scaling.

- **Feature-Target Separation:** The target/ class variable (y) and predictors/ features (X) are separated within the given dataset. The class column is labelled as the target, while other predictors within the remaining columns are treated as features. This is done per the compliance outlined in sci-kit learn's API.
- **Stratified Train-Test Split**: Using an 80-20 partition to split the dataset into 80% Training (i.e., Training Data: 227,845 transactions, of which 394 fraud cases) and 20% Test (i.e., Testing Data: 56,961 transactions, of which 98 fraud cases). This ratio ensures sufficient training data while maintaining a representative test set. "test_size=0.2" is used to achieve this split, presented with the "train_test_split" function from "sklearn.model_selection". Other constraints include "random_state=42" for reproducibility, and "stratify=y" to maintain the target class distribution (i.e., 0.17%) in both sets.
- **Feature Scaling**: A "StandardScaler" from "sklearn.preprocessing" is applied to scale all features (including Time, Amount, PCA-transformed V1-V28) to mean = 0 and variance = 1. PCA-transformed features were standardised again as this step is particularly crucial for distance-based models like "Logistic Regression", while Tree-based model like "Random Forest" can still benefit from standardized data even though they are theorectically scale-invariant . "fit_transform" is applied to only "X_train" to learn scaling parameters from the training dataset without data leakage. "transform" applies the same scaling to "X_test" without refitting.

(Pedregosa et al., 2015)[7] (Wang & Yao, 2009)[8]

4. Models

Model	Pipeline Code	Key Hyperparameters	Justification
Random Forest	Pipline ([('scaler', StandardScaler()), ('classifier', RandomForestClassifier())])	n_estimators=100, max_depth=None	Via ensemble voting produces robust results for imbalanced data. (Velarde et al., 2024)[3]
Logistic Regression	Pipline ([('scaler', StandardScaler()), ('classifier', LogisticRegression())])	Max_iter=1000, C=1.0(default)	Efficient for high dimensional data, baseline for binary classification. (Pedregosa et al., 2015)[7]

Table 1 Model Implementation: This table provides specifications of pipelines, parameter constraints and justification for the choice of classifiers for the credit card fraud dataset.

4.1 Model Implementation with Pipelines

To streamline preprocessing and model training, two pipelines are created using 'sklearn.pipeline.Pipeline'. The Random Forest pipeline (rf_pipeline) includes a 'StandardScaler' to standardise features, and a RandomForestClassifier with 'random_state=42' for reproducibility. Similar to Random Forest pipelines, the Logistic Regression pipeline (lr_pipeline) uses 'StandardScaler', setting 'max_iter=1000' to ensure convergence and 'random_state=42'.
Due to the sensitivity of Logistic regression and the additional benefit of pre-processing consistency for Random Forest, standardisation is critical.
Since the F1 score is suitable for imbalanced datasets, 5-fold cross-validation is used with it for evaluating model performance. F1 score balances precision and recall, which is critical for fraud detection where both false positives (i.e., legit transactions flagged) and false negatives (i.e. missed fraud) are costly. Pipelines of both training data, ie. 'X_train, y_train' gets F1 scores computed via the 'cross_val_score' function from 'sklearn.model_selection'.

- The mean F-score for Random Forest is calculated to be 0.8384,
- And the Mean F-score for Logistic Regression is 0.7312.

(He & Garcia, 2009)[4]

4.2 Hyperparameter Optimisation

Hyperparameter tuning is performed using 'GridSearchCV' to enhance the Random Forest model, from 'sklearn.model_selection'. A parameter grid is defined, with the following tuning values:

- 'max_depth' (None, 10): This defines the maximum depth of a decision tree, i.e. the longest path from root to leaf. A deeper tree with parameter 'None', i.e. no limit to the depth, can

capture the complexity of data to a greater extent but at the risk of overfitting by modelling noise in the training dataset, whereas a restricted depth of '10' ensures simpler trees, potentially improving generalisation.

- 'n_estimators' (50, 100): This specifies the number of decision trees within the Random Forest. Each tree is being trained on a random subset of data; the higher the number of decision trees, the greater is the robustness of the aggregated predictor value generated by the Random Forest. Numbers '50' and '100' specify the number of decision trees run and aggregated within a cycle. Higher number constraints for n_estimators give more robust data, but cost greater computational effort.
- 'min_samples_split (2): This defines the minimum number of samples required before splitting an internal node within a decision tree. In other words, if the value for 'min_samples_split' is defined as 2, it means that the decision tree requires at least 2 samples before splitting the node. Higher value for 'min_samples_split' prevents overly specific splits and promotes generalisation, but risks underfitting.

The Grid-Search applies a 5-fold cross-validation technique where the dataset is split into 5 parts, and 4/5th of the dataset is used for training and 1/5th for testing. The selection of initial parts for training and testing is made randomly, followed by sequential turnover of each section within training and testing, such that every split is unique for each split carried out 5 times. This method allows the calculation of the F1 score for each iteration, which can be averaged to give an optimum F1 score more robust than a standard 80-20 split carried out only once.

Another constraint called 'n_jobs=-1' was used to specify the number of cores used by the CPU to perform parallel processing during grid search. The value '-1' specifies using all cores to maximise the computational efficiency by distributing the workload of fitting models for different hyperparameter combinations across multiple cores simultaneously.

Based on the above-mentioned hyperparameter optimisations:

- The Best F1 score value was computed to be 0.8400.
- The parameter values through which the best F1 score was computed are:
    - 'classifier_max_depth': 10
    - 'classifier_min_samples_split': 2
    - 'classifier_n_estimators': 50

The key point to highlight here is that Random Forest was chosen for optimisation over Logistic regression because of its proven effectiveness with imbalanced financial data. Since running hyperparameter optimisation was computationally costly and time-consuming, similar optimisation was not carried out for Logistic Regression.

(Velarde et al., 2024)[3] (He & Garcia, 2009)[4]

5. Results

The effectiveness of two different classifiers, Random Forest and Logistic Regression, used to distinguish between 'Legit' and 'Fraud' cases in the test dataset, was evaluated using classification metrics presented in performance evaluation reports. The evaluation report provides precision, recall and F-1 score along with support for each class in the form of macro and weighted averages to give a more comprehensive overview of model performance (as shown in Figure 2). This is made possible by leveraging the 'classification_report' present within scikit learn.

```
Random Forest Report:
              precision    recall  f1-score   support

       Legit       1.00      1.00      1.00     56864
       Fraud       0.94      0.82      0.87        98

    accuracy                           1.00     56962
   macro avg       0.97      0.91      0.94     56962
weighted avg       1.00      1.00      1.00     56962

Logistic Regression Report:
              precision    recall  f1-score   support

       Legit       1.00      1.00      1.00     56864
       Fraud       0.83      0.65      0.73        98

    accuracy                           1.00     56962
   macro avg       0.92      0.83      0.87     56962
weighted avg       1.00      1.00      1.00     56962
```

Figure 2: Comprehensive Evaluation Report for both Classifiers obtained from Jupyter Notebook.

5.1 Classifier Performance Metric

The evaluation of the optimised Random Forest and Logistic Regression model yielded the following results.

Random Forest

	Precsion	Recall	F-1 Score	Support
Legit	1.0	1.0	1.0	56864
Fraud	0.94	0.82	0.87	98

Table 2: Evaluation Report obtained from Jupyter Notebook for Random Forest Classifier.

Fraud Detection:

- Precision (0.94): 94% of cases flagged by the model were truly fraudulent.
- Recall (0.82): The model captured 82% of all the fraud cases (i.e. only missed 18 out of 98 cases).
- F-1 score (0.87): The performance is balanced, suitable for deployment. (Velarde et al., 2024)[3]

Logistic Regression

	Precsion	Recall	F-1 Score	Support
Legit	1.0	1.0	1.0	56864
Fraud	0.83	0.65	0.73	98

Table 3: Evaluation Report obtained from Jupyter Notebook for Logistic Regression Classifier.

Fraud Detection:

- Precision (0.83): 83% of cases flagged by the model were truly fraudulent. Lower confidence in fraud alerts compared to Random Forest.
- Recall (0.65): The model missed 35% of all the fraud cases (i.e. missed 34 out of 98 cases). This is a critical shortcoming. (Ashtiani & Raahemi, 2021)[1]

5.2 Confusion Analysis

Model	True Negatives (Legit)	True Positives (Fraud)	False Negatives	False Postives
Random Forest	56864	80	18	0
Logistic Regression	56864	64	34	0

Table 4: Evaluation Report obtained from Jupyter Notebook for Logistic Regression Classifier.

Key Observations:

- False Negatives: 47% reduction in missed fraud cases observed in the Random Forest classifier compared to Logistic Regression (i.e. 18 vs. 34).
- Perfect Legit Transaction Handling: Both classifiers correctly identified all legitimate transactions.

5.3 Statistical Significance

Relative Improvements

Random Forest improved fraud recall by 26% (0.82 vs. 0.65) and F1-score by 19% (i.e., 0.87 vs 0.73) compared to Logistic Regression.

Macro Averages

The F1 macro average score for Random Forest (0.94 F1) was observed to be higher compared to that of Logistic Regression (0.87 F1). Thereby confirming better handling of class imbalance. (He & Garcia, 2009)[4]

5.4 Business Implications

Metric	Random Forest	Logistic Regression	Financial Impact
Missed Fraud (Recall)	18 cases	34 cases	Random Forest saves ~€1600 more per 100 fraud transactions(assuming ~€100 loss for each fraud transaction)
Fraud Alert Accuracy	56864	64	Random Forest reduces customer friction by minimizing false fraud alerts. (Şahin, Yusuf G & Duman, 2011)[9]

Table 5: Illustration of Potential Business Impact produced by different classifiers.

Limitations:

- Dataset Bias: Results obtained through models, despite being optimised, are limited in their real-world relevance as they assume the distribution of fraud cases in the test dataset matches the real world.
- Feature Opacity: Root cause analysis of fraud cases is limited by PCA-transformed features. (Jiang et al., 2023)[2]

(Van Vlasselaer et al., 2015)[10]

6. Conclusion

This study used an unbalanced dataset to compare Random Forest with Logistic regression for the identification of credit fraud. The Random Forest produced an F1 score of 0.87 compared to 0.73, i.e. F1 score produced by Logistic Regression. The optimised Random Forest significantly outperformed Logistic Regression, resulting in a 47% reduction in missed fraud cases. Random Forest's improved recall makes it more appropriate for real-world deployment, where fraud detection is crucial, even if both models maintained flawless precision for valid transactions. Dataset bias and opaque PCA features are among the limitations. To increase interpretability, future research can investigate explainable AI methods or hybrid models. These results support tree-based techniques for fraud detection systems that need to be very accurate and dependable.

7. References

1. Ashtiani, M. N., & Raahemi, B. (2021). Intelligent Fraud Detection in Financial Statements using Machine Learning and Data Mining: A Systematic Literature Review. IEEE Access, 10, 72504–72525. https://doi.org/10.1109/access.2021.3096799. (Ashtiani & Raahemi, 2021)
2. Jiang, S., Dong, R., Wang, J., & Xia, M. (2023). Credit Card Fraud Detection Based on Unsupervised Attentional Anomaly Detection Network. Systems, 11(6), 305. https://doi.org/10.3390/systems11060305.
3. Velarde, G., Weichert, M., Deshmukh, A., Deshmane, S., Sudhir, A., Sharma, K., & Joshi, V. (2024). Tree boosting methods for balanced and imbalanced classification and their robustness over time in risk assessment. Intelligent Systems with Applications, 22, 200354. https://doi.org/10.1016/j.iswa.2024.200354.
4. He, H., & Garcia, E. A. (2009). Learning from Imbalanced Data. IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263–1284. https://doi.org/10.1109/tkde.2008.239.
5. *Credit Card Fraud Detection*. (n.d.). www.kaggle.com. https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud?resource=download.
6. Pozzolo, A. D., Caelen, O., Johnson, R. A., & Bontempi, G. (2015). Calibrating Probability with Undersampling for Unbalanced Classification. 2015 IEEE Symposium Series on Computational Intelligence. https://doi.org/10.1109/ssci.2015.33.
7. Pedregosa, F., Buitinck, L., Louppe, G., Grisel, O., Varoquaux, G., & Mueller, A. (2015). Scikit-learn. GetMobile: Mobile Computing and Communications, 19(1), 29–33. https://doi.org/10.1145/2786984.2786995.
8. Wang, S., & Yao, X. (2009). Diversity analysis on imbalanced data sets by using ensemble models. 2009 IEEE Symposium on Computational Intelligence and Data Mining. https://doi.org/10.1109/cidm.2009.4938667
9. Şahin, Yusuf G, & Duman, E. (2011). Detecting credit card fraud by decision trees and support vector machines. Dogus.edu.tr. https://doi.org/9789881821034.
10. Van Vlasselaer, V., Bravo, C., Caelen, O., Eliassi-Rad, T., Akoglu, L., Snoeck, M., & Baesens, B. (2015). APATE: A novel approach for automated credit card transaction fraud detection using network-based extensions. Decision Support Systems, 75, 38–48. https://doi.org/10.1016/j.dss.2015.04.013.