International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

A Product Owner's Guide to Machine Learning: Case Study - Credit Card Fraud Detection

Sachin Gadiyar[1], Archana Umakanth[2]

Email Address: [1]gadiyar.sachin@gmail.com, [2]archana.umakanth@gmail.com

Abstract
Machine Learning (ML) models are increasingly being integrated into the financial industry to enable development of intelligent, data driven products. For Product Owners to successfully guide teams and drive the strategic use of ML in their products, it is crucial to understand the end-to-end process of ML model development. This paper provides an accessible framework for product owners, using an example of Credit Card Fraud Detection, to grasp the key stages involved in building ML models. It covers the foundational concepts, including data preprocessing, model selection, training and evaluation. Additionally, it explores critical topics such as model performance metrics, collaboration with data science and engineering teams, and the importance of continual monitoring and iteration. By demystifying the ML process, this paper equips product owners with the knowledge to effectively prioritize features, manage expectations, ensure ethical practices, and leverage ML to drive product innovation. The goal is to empower product owners to make informed decisions, communicate effectively with technical teams, and ultimately deliver impactful ML-powered products that meet both business objectives and customer needs.

Keywords: Artificial Intelligence, Machine Learning, Credit Cards, Fintech, Data Science, Analytics, Fraud Detection, Random Forest, XGBoost, ANN

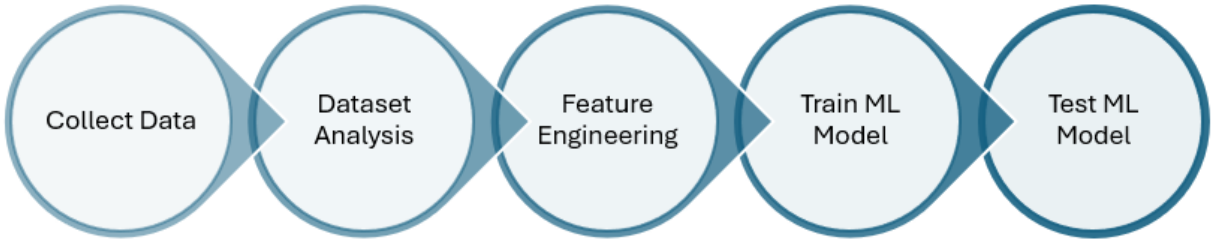
1. Introduction

A **Machine Learning (ML) model** is a mathematical model or algorithm trained to recognize patterns in data and make predictions or decisions on new, unseen data. In essence, an ML model is a function that learns from data and improves its predictions or decisions over time. ML models can be broadly classified into the following categories:

- **Supervised Learning:** Such models are trained on labeled data, meaning the input data comes with corresponding output labels pre-defined. The goal is to learn mapping from inputs to outputs. (e.g. 'Classification' models that predict if a transaction is fraudulent or not)
- **Unsupervised Learning:** In unsupervised learning, the model is trained on unlabeled data. The goal is to find hidden patterns or structures within the data without predefined output labels. (e.g. 'Clustering' models that group data points into clusters based on similarities.)
- **Reinforcement Learning:** In reinforcement learning, the learning 'agent' learns by interacting with its environment and receiving feedback in the form of rewards or penalties. The goal is to learn the best actions to maximize cumulative reward.
- **Semi-Supervised Learning**: Uses a mix of labeled and unlabeled data.
- **Self-Supervised Learning:** Self-supervised learning involves creating labels automatically from the data itself. The model learns by predicting parts of the input data (e.g., predicting missing words in a sentence or missing pixels in an image).
- **Ensemble Learning:** Ensemble methods combine multiple individual models (often weaker models) to improve the overall performance. These methods aim to reduce errors and variance.
- **Deep Learning**: Deep learning refers to neural networks with many layers (hence "deep"). These models are capable of learning complex representations of data and are particularly

useful for tasks like image recognition, natural language processing (NLP), and speech recognition.

ML model creation begins with collecting the dataset. In dataset analysis, the data is then carefully examining for bias, outliers etc. and 'cleaned up' to ensure there is no invalid data to interfere with results. In some cases, normalization may be needed to get all data to similar scale. Feature Engineering is the next step, which is a process of carefully determining which inputs or 'features' are to be fed into the model. The dataset is then split into two - train and test components. The model is 'trained' or 'fitted' using the train dataset and is then 'tested' against the test dataset. Often multiple models are used and a comparison between the performance of the models is made to determine which is the best fit for a given use case. Figure below shows a typical workflow for developing an AI/ML model.



2. Development of an ML model: Case Study – Credit Card Fraud Detection

This paper explores the development of a **Supervised Learning Classification** model designed to predict fraudulent credit card transactions. The dataset used here was collected and analyzed as part of a research collaboration between Worldline and the Machine Learning Group of ULB (Université Libre de Bruxelles), focusing on big data mining and fraud detection. The model was developed using Python, chosen for its simplicity, readability and extensive libraries.

2.1 Exploratory Data Analysis (EDA)

The first, and perhaps the most crucial, step in developing an ML model is understanding the dataset. Optimizing the dataset is crucial to achieving a high performing model. Issues such as missing data, outliers and inconsistencies can significantly impact model performance. Additionally, any biases present in the dataset can be amplified during model training, leading to undesirable consequences. Key objectives of EDA are:

- **Understanding the Structure of the Data**: Dimensions (number of rows and columns), types of variables (numerical, categorical), and data types.
- **Identifying Patterns and Relationships:** Find correlations, trends, or groups in the data that can provide insights into the underlying phenomena.
- **Spot Data Issues:** Detect missing values, outliers, or inconsistencies that could affect modeling.

Data Visualization tools such as histograms, boxplots, heatmaps etc. help uncover inherent patterns and relationships.

Functions like 'Correlation Matrix' can provide the correlation between different attributes in dataset. Figure below shows a sample correlation matrix for the fraud dataset; cool/blue hues showing a negative correlation and red/hot hues indicating a positive correlation. From the figure, it is easily evident that the output variable 'Class' has a positive correlation with V11, meaning as V11 value increases, the class value also tends to be higher.

	Amount	Class	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17
Amount	1.000000	0.005632	-0.227709	-0.531409	-0.210880	0.098732	-0.386356	0.215981	0.397311	-0.103079	-0.044246	-0.101502	0.000104	-0.009542	0.005293	0.033751	-0.002986	-0.003910	0.007309
Class	0.005632	1.000000	-0.101347	0.091289	-0.192961	0.133447	-0.094974	-0.043643	-0.187257	0.019875	-0.097733	-0.216883	0.154876	-0.260593	-0.004570	-0.302544	-0.004223	-0.196539	-0.326481
V1	-0.227709	-0.101347	1.000000	0.000000	-0.000000	-0.000000	0.000000	-0.000000	-0.000000	-0.000000	-0.000000	0.000000	0.000000	0.000000	-0.000000	-0.000000	0.000000	0.000000	-0.000000
V2	-0.531409	0.091289	0.000000	1.000000	0.000000	-0.000000	0.000000	0.000000	0.000000	-0.000000	0.000000	-0.000000	0.000000	-0.000000	0.000000	-0.000000	-0.000000	0.000000	-0.000000
V3	-0.210880	-0.192961	-0.000000	0.000000	1.000000	0.000000	-0.000000	0.000000	0.000000	-0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
V4	0.098732	0.133447	-0.000000	-0.000000	0.000000	1.000000	-0.000000	-0.000000	-0.000000	0.000000	0.000000	0.000000	0.000000	-0.000000	0.000000	0.000000	0.000000	-0.000000	-0.000000
V5	-0.386356	-0.094974	0.000000	0.000000	-0.000000	-0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	-0.000000	0.000000	0.000000	0.000000	0.000000	-0.000000	0.000000	0.000000
V6	0.215981	-0.043643	-0.000000	0.000000	0.000000	-0.000000	0.000000	1.000000	0.000000	-0.000000	0.000000	0.000000	0.000000	0.000000	-0.000000	0.000000	-0.000000	0.000000	0.000000
V7	0.397311	-0.187257	-0.000000	0.000000	0.000000	-0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	-0.000000	0.000000	-0.000000	0.000000	0.000000	-0.000000	0.000000	0.000000
V8	-0.103079	0.019875	-0.000000	-0.000000	-0.000000	0.000000	0.000000	-0.000000	0.000000	1.000000	0.000000	-0.000000	0.000000	0.000000	-0.000000	-0.000000	0.000000	-0.000000	-0.000000
V9	-0.044246	-0.097733	-0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	-0.000000	0.000000	-0.000000	0.000000	0.000000	-0.000000	-0.000000	0.000000
V10	-0.101502	-0.216883	0.000000	-0.000000	0.000000	0.000000	-0.000000	0.000000	-0.000000	-0.000000	-0.000000	1.000000	-0.000000	0.000000	-0.000000	0.000000	0.000000	0.000000	0.000000
V11	0.000104	0.154876	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
V12	-0.009542	-0.260593	0.000000	-0.000000	0.000000	-0.000000	0.000000	0.000000	-0.000000	0.000000	-0.000000	0.000000	0.000000	1.000000	-0.000000	0.000000	-0.000000	0.000000	0.000000
V13	0.005293	-0.004570	-0.000000	0.000000	0.000000	0.000000	0.000000	-0.000000	0.000000	-0.000000	0.000000	-0.000000	0.000000	-0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
V14	0.033751	-0.302544	-0.000000	-0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	-0.000000	-0.000000	0.000000
V15	-0.002986	-0.004223	0.000000	-0.000000	0.000000	0.000000	-0.000000	-0.000000	-0.000000	0.000000	-0.000000	0.000000	0.000000	-0.000000	0.000000	-0.000000	1.000000	0.000000	0.000000
V16	-0.003910	-0.196539	0.000000	0.000000	0.000000	-0.000000	0.000000	0.000000	0.000000	-0.000000	-0.000000	0.000000	0.000000	0.000000	0.000000	-0.000000	0.000000	1.000000	0.000000
V17	0.007309	-0.326481	-0.000000	-0.000000	0.000000	-0.000000	0.000000	0.000000	0.000000	-0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000

2.2 Feature Engineering

This process involves careful examination of 'features' or inputs that are to be used in the model. Often, transforming features (creating new features, combining existing features, scaling and normalizing etc.) may be needed to improve data quality.

2.2.1 Feature Transformation:

A real-world credit card transaction dataset may include several attributes including potential customer personal information. It may include transaction details like Credit Card Number, Transaction Date and Time, Merchant information, Transaction Amount, Transaction Unique Identifier, Merchant location (latitude/longitude) etc. as well as customer information like Customer Name, Address, Location, Date of Birth etc. While this type of data may intuitively make sense to humans, for ML models, the data must be numerical and ordered. Therefore, some transformation techniques need to be employed.

The dataset used in this paper contains only numerical input variables which are the result of such a transformation on real world data. As illustrated in the figure below, all attributes, except the transaction amount and time, are transformed values. The transformation technique employed by ULB to generate this dataset is called 'Principal Component Analysis' (PCA).

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0

Column	Description
V1, V2, ...V28	Transformed transaction attributes
Time	Time (in seconds) elapsed between each transaction and the first transaction in dataset
Class	Indicates if transaction is Fraudulent or not; 0=Not Fraudulent, 1=Fraudulent

2.2.2 Imbalance in Data

The preliminary analysis of the dataset revealed a significant imbalance in the output variable. Specifically, only 0.2% of data pertains to actual fraudulent data while 98.8% pertains to non-fraudulent data, which is quite typical of real-life transactions.

To address this imbalance, sampling technique was employed. Oversampling generates new records of minority class (fraud transactions), and undersampling drops some records from majority class (non-fraudulent transactions) to achieve a more balanced dataset. While Python libraries provide multiple sampling techniques, for this paper, the 'Random Sampling' function was used.

Figure below show the change in imbalance in training data before and after applying the sampling technique.

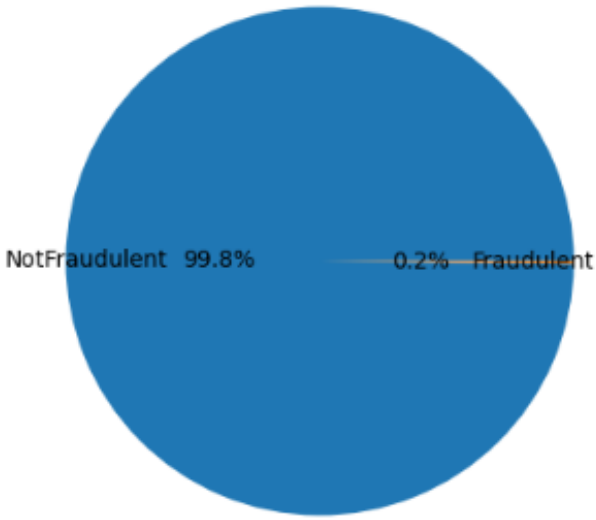


Fig 1: Data imbalance before sampling

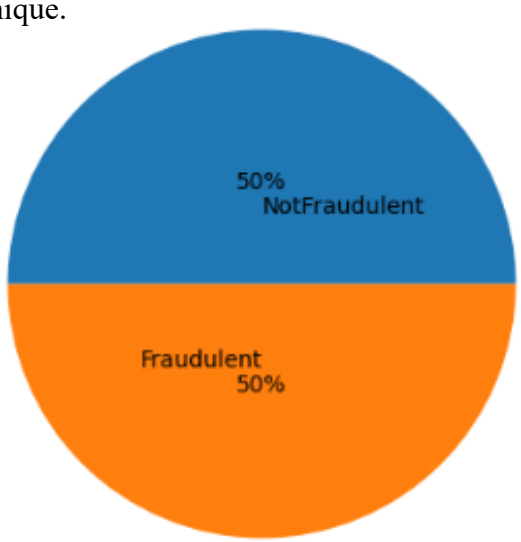


Fig 2: Data balanced after sampling

It is important to note that sampling must be done on model training data only and never on model testing data. Moreover, sampling is not necessary in all cases of imbalanced data and must be evaluated based on the use case under context.

2.3 Interpreting Model Performance Results

In this case study model performance will be evaluated using Accuracy, Precision and Recall values.

- **Accuracy** is the ratio of correctly predicted instances (both true positives and true negatives) to the total number of instances.
- **Precision** (also known as positive predictive value) measures the accuracy of the positive predictions. When the model predicts a positive instance, precision measures what proportion of those predictions are correct.
- **Recall** (also known as sensitivity or true positive rate) measures the ability of a model to identify all relevant instances. i.e. It measures the proportion of actual positive instances that are correctly predicted, reflecting the model's ability to capture all relevant cases
- **F1 Score** provides a single measure of a model's performance by balancing the trade-off between precision and recall; Mathematically, it is the harmonic mean of precision and recall values.

For highly imbalanced datasets, accuracy alone is not a measure of model performance. In this case where the data is 99.8% non-fraudulent, the model can simply predict all transactions as non-fraudulent and achieve an accuracy of 99.8%. Hence careful examination of Precision and Recall values is also important. A model with high recall and low precision indicates presence of many false positives; On the other hand, a model with low recall and high precision would indicate that the fraudulent transaction identified are accurate, but model may not identify all the fraudulent population.

2.4 Model development

There are a variety of models available in Python libraries, like Decision trees, Linear models, Naïve Bayes and ANN to name a few. When developing a ML model for a use case, it is common to experiment with and compare multiple models to identify the best performing model. For predictive models, like in the context of this paper, **Decision Tree** based models are known to perform extremely well. Decision Trees are a fundamental machine learning algorithm used for both classification and regression tasks. They model decisions and their possible consequences, including outcomes, costs, and benefits, in the form of a tree-like structure. The decision tree-based models also perform better on imbalanced data. For this use case, the following models were developed and the performance compared:

- Random Forest
- Extreme Gradient Boosting (XGBoost)
- Artificial Neural Network (ANN).

The dataset was randomly split into 80% / 20% train/test categories. Models were fitted using training data and evaluated against the test data.

2.4.1 Random Forest

Random Forest is an ensemble method that combines the predictions of multiple decision trees to produce a more robust and accurate model. This approach helps mitigate the limitations of individual decision trees as well as avoids potential over-fitting issues.

The Random Forest Model resulted in a training accuracy of 99.28% and a test accuracy of 99.84%.

Classification Report:

	Precision	Recall	f1-score
Not Fraudulent	1.000	0.999	0.999
Fraudulent	0.519	0.847	0.643
Accuracy			0.998

The classification report indicates that while the model is good at detecting fraudulent transactions (high recall), it also misclassifies a significant number of non-fraudulent transactions as fraudulent (low precision). Despite the model having a very high accuracy, the performance of the minority class (Fraudulent) is poor.

2.4.2 XGBoost (Extreme Gradient Boosting)

XGBoost (Extreme Gradient Boosting) is another ensemble method that is based on decision trees. It is a highly efficient, scalable, and widely used machine learning algorithm that is part of the gradient boosting family. It is particularly popular in classification, regression, and ranking tasks, and has gained prominence in both research and industry due to its performance and flexibility.

The Random Forest Model gave a train accuracy of 100% and a test accuracy of 99.94%.

Classification Report:

	Precision	Recall	f1-score
Not Fraudulent	1.000	1.000	1.000
Fraudulent	0.875	0.786	0.828
Accuracy			0.999

With the XGBoost model, there is certainly an improvement in performance for the minority class compared to the previous model. The model performs very well for predicting **Not Fraudulent** transactions and is relatively accurate (Precision 0.88) for predicting **Fraudulent** transactions (although some non-fraudulent transactions are still predicted as fraudulent).

2.4.3 Artificial Neural Network (ANN)

Artificial Neural Networks (ANNs) are a class of machine learning models inspired by the structure and function of the human brain. ANNs consist of layers of interconnected nodes called neurons. The basic structure includes an **Input Layer**: The first layer that receives the input data, **Hidden Layers**: One or more layers where the actual processing happens through weighted connections and **Output Layer**: The final layer that produces the output, whether it be class labels for classification or continuous values for regression. ANNs are trained using a process called backpropagation. When ANNs have multiple hidden layers (typically more than two), they are often referred to as Deep Neural Networks (DNNs).

The ANN Model resulted in a train accuracy of 95% and a test accuracy of 98.9%.

Classification report:

	Precision	Recall	f1-score
Not Fraudulent	1.000	0.989	0.994
Fraudulent	0.118	0.857	0.207
Accuracy			0.989

Classification report suggests the model struggles significantly to predict Fraudulent transactions, as reflected by a very low precision (0.04) and F1-score (0.08). While recall for Fraudulent transactions is higher (0.87), the low precision indicates that many non-fraudulent transactions are being incorrectly predicted as fraudulent.

2.5 Model Evaluation Results

Among the three models compared, the XGBoost model outperformed the others in terms of accuracy, precision, recall, and F1 score. In contrast, the ANN model exhibited significantly lower performance, likely due to the original data being skewed and the oversampling process resulting in duplicate entries from the minority class. While Random Forest performs adequately, it has some weaknesses, particularly in misidentifying non fraudulent transactions and missing.

Model	Test Accuracy %	Transaction Type	Precision	Recall	F1 Score
Random Forest	99.84	Not Fraudulent	1.00	1.00	1.00
		Fraudulent	0.52	0.85	0.64
XGBoost	99.94	Not Fraudulent	1.00	1.00	1.00
		Fraudulent	0.88	0.79	0.83
Artificial Neural Network	98	Not Fraudulent	1.00	0.98	0.99
		Fraudulent	0.06	0.87	0.12

The XGBoost model can be improved further to increase the recall rate, since misidentifying non-fraudulent transactions as fraudulent can cause bad customer experience. Concepts like 'Cost Sensitive Learning' can be explored to penalize false positives and improve recall rate.

2.6 Significance of Product Perspective in ML Model Development

In large organizations, model development is often led by Data Science and Technology teams that work separately from the Product teams. This separation can result in a gap of product perspective in the model development process. Data teams may lack the domain knowledge and product-specific insights that are embedded within Product teams. Incorporating this expertise—particularly during stages like data analysis and feature engineering—can have a significant positive impact on model performance.

Moreover, Product teams often have a deeper understanding of compliance and regulatory requirements and can help identify potential biases in the model. For example, models that use customer demographics for predictions may unintentionally introduce racial or socioeconomic biases, which can be mitigated with input from the Product teams.

3. Conclusion

Machine Learning (ML) models are increasingly being integrated into financial industry products. Often these models are developed in silo by the modeling team which operates separately from the product team. This means the ML models often appear as a 'black box' to Product Owners who rely solely on the model outcomes to drive the product decisions. Understanding how ML models are developed can help Product Owners make informed decisions about the product features, design and road map. Additionally, this knowledge also empowers Product Owners to collaborate and share valuable domain and product insights with the modelling team, ultimately improving both model performance and product outcomes.

References

1. M. L. G. ULB, "Credit Card Fraud Detection," Kaggle, [Online]. Available: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud. [Accessed: 3-Dec-2024].
2. M. L. G. ULB, "Machine Learning Group (MLG)," [Online]. Available: http://mlg.ulb.ac.be. [Accessed: 6-Dec-2024].
3. T. F. L. I. Handbook, "Fraud Detection Handbook: Foreword," [Online]. Available: https://fraud-detection-handbook.github.io/fraud-detection-handbook/Foreword.html. [Accessed: 7-Dec-2024].
4. Scikit-learn developers, "Principal Component Analysis (PCA) — scikit-learn 1.5.2 documentation," [Online]. Available: https://scikit-learn.org/dev/modules/generated/sklearn.decomposition.PCA.html. [Accessed: 10-Dec-2024].
5. Scikit-learn developers, "Random Forest Classifier — scikit-learn 1.5.2 documentation," [Online]. Available: https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Accessed: 11-Dec-2024].
6. XGBoost developers, "XGBoost Documentation," [Online]. Available: https://xgboost.readthedocs.io/en/latest/python/index.html. [Accessed: 11-Dec-2024].
7. Keras developers, "Keras Documentation," [Online]. Available: https://keras.io/api/. [Accessed: 17-Dec-2024].