République Tunisienne
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Carthage - Ecole Supérieure des Statistiques et de l'Analyse de l'Information

---

Presented By

Oumayma Bejaoui

Modeling Inter-Transaction Times with Explainable AI

Supervised by:

MR. Ahmed Meddah — BIAT

Mr. Nader — BIAT

Abstract

In the modern banking landscape, predicting customer behavior is essential for enhancing customer engagement, optimizing resource allocation, and proactively managing client relationships. This internship project focuses on developing an end-to-end machine learning pipeline to predict the number of days until a customer's next banking transaction, using historical transaction data.

The project began with extensive data preprocessing, including cleaning, merging, and transforming timestamped financial data. A robust feature engineering phase followed, generating time-based, behavioral, and aggregated variables. Clients were segmented using a KMeans clustering approach based on RFM (Recency, Frequency, Monetary) indicators to enhance model expressiveness.

The predictive model was built using CatBoost, leveraging ordered boosting and native categorical handling. We augmented features with Fourier terms to encode weekly, monthly, and annual seasonality, and optimized hyperparameters with Optuna. Cross-validation used GroupKFold at the client level to prevent leakage, with a time-ordered holdout for early stopping. The model was trained on a log-transformed target for stability and evaluated on the real-day scale; SHAP values provided interpretability by highlighting the most influential drivers.

Finally, the solution was deployed in a production-ready environment using Docker, FastAPI, and PostgreSQL, with a user-facing dashboard enabling real-time interaction with the model outputs. This project demonstrates the full lifecycle of a machine learning solution—from raw data to deployment—within a real-world financial services context.

Introduction

In the context of BIAT's Private Corporate Banking (PCB) clients, transactions represent more than routine financial activity—they reflect strategic interactions between high-value clients and the bank. These clients typically approach the bank for financing needs, which may involve negotiating preferential exchange rates or requesting specific funding amounts to support their operations. Each transaction is therefore the result of a tailored process, shaped by the client's profile and the bank's financial strategy. Understanding the timing and frequency of such transactions provides critical insight into client behavior, enabling the bank to anticipate needs, strengthen relationships, and optimize resource allocation.

The data used in this project was provided by the Banque Internationale Arabe de Tunisie (BIAT) covering client profiles, employee information, organizational structures, and detailed transaction logs for the years 2021 to 2023. This raw data served as the foundation for a complete machine learning pipeline that integrates preprocessing, feature engineering, behavioral clustering, predictive modeling, model explainability, and deployment.

The project encompassed a complete machine learning pipeline, including:

- Data ingestion and preprocessing of timestamped banking records,
- Feature engineering with behavioral, temporal, and aggregated metrics,
- Client segmentation through unsupervised learning (KMeans),
- Predictive modeling using CatBoost with hyperparameter optimization (Optuna),
- Model validation using cross-validation strategies that prevent data leakage,
- Interpretability analysis using SHAP (SHapley Additive exPlanations),
- And finally, deployment of the solution using FastAPI, PostgreSQL, and Docker within an interactive dashboard environment.

One of the main challenges of this project was to preserve the temporal integrity and client-based isolation of the data across training and validation phases, while creating features that are both robust and informative.

Chapter 1

Problem Statement

The core problem addressed in this project can be formulated as follows:

Given a history of transactional data for a bank's clients, predict the number of days until each client performs their next transaction.

This problem presents several technical and business complexities:

1. From a technical perspective:

    - Target Definition: The "next transaction date" is not directly present in the data; it must be engineered through temporal differencing.
    - Temporal Nature: The data is sequential and time-dependent, requiring careful handling to prevent information leakage.
    - Missingness and Irregularity: Clients have irregular transaction patterns, leading to sparse and unevenly spaced data.
    - Categorical Variables: Several features are categorical (e.g., client type, direction, segment) and require encoding for machine learning models.
    - Feature Engineering: The predictive power depends heavily on creating insightful features from dates, amounts, and client history.

2. From a business perspective:

    - Client Segmentation: Not all clients behave the same; clustering may help differentiate strategies.
    - Deployment Readiness: The model must eventually integrate into the bank's IT systems through APIs and databases.
    - Interpretability: Business stakeholders need to understand why the model makes a prediction, especially in sensitive contexts like client contact strategies.

The solution must therefore strike a balance between model complexity, predictive performance, and business usability.

Chapter 2

Data description

The dataset used in this project was provided by BIAT (Banque Internationale Arabe de Tunisie). It reflects real, anonymized operational data collected internally by the bank over a three-year period (2021–2023). The data was delivered in multiple Excel files grouped by entity (clients, employees, directions, groups) and transaction type (negotiated vs. entrusted operations).

This rich dataset captures a wide range of variables related to customer behavior, transactional volume, currency types, booking channels, and client hierarchy. It serves as the foundation for building predictive models aimed at forecasting the delay until a customer's next banking transaction.

The dataset comprises the following Excel files:

File Name	**Description**
`table client.xlsx`	Client metadata
`table employes.xlsx`	Assigned bank employees
`table direction.xlsx`	Organizational hierarchy and direction metadata
`table groupe.xlsx`	Client group affiliation and structure
`table négocié - 2021 1 1.xlsx`	Negotiated transactions from early 2021
`table négocié - 2021 1 2.xlsx`	Continuation of 2021 negotiated data
`table négocié - 2022 partie 1.xlsx`	First batch of 2022 negotiated transactions
`table négocié - 2022 partie 2.xlsx`	Second batch of 2022 negotiated transactions
`table négocié - 2022 partie 3.xlsx`	Third batch of 2022 negotiated transactions
`table négocié - 2023 - 1.xlsx`	Negotiated operations, 2023 part 1
`table négocié - 2023 - 2.xlsx`	Negotiated operations, 2023 part 2
`table confi - 2021.xlsx`	Entrusted transactions for 2021
`table confié - 2022.xlsx`	Entrusted transactions for 2022
`table confi - 2023 1 1.xlsx`	First batch of 2023 entrusted transactions
`table confi - 2023 1 2.xlsx`	Second batch of 2023 entrusted transactions
`table confi - 2023 3.xlsx`	Third batch of 2023 entrusted transactions
`table confi - 2023 4.xlsx`	Fourth batch of 2023 entrusted transactions

2. Data Structure and Integration

The overall dataset used in this work is the result of a structured integration process combining client master data, organizational metadata, and transactional flows. The initial sources are heterogeneous: the `client` table contains information about individual clients and their segmentation attributes, while the `employe` and `direction` tables describe the internal organizational structure by linking employees to specific business directions and poles. In addition, the `groupe` table provides a higher-level mapping of clients into groups. These sources represent the institutional and organizational backbone of the dataset. On the other side, the `confi` and `negoci` tables provide the transactional layer, each recording foreign exchange and negotiation operations with detailed information on reference numbers, dates, amounts in both foreign and local currencies, exchange rates, and applied margins.

A cleaning phase was necessary before integration. Transactions from multiple files and time periods were concatenated into two unified tables (`confi` and `negoci`), where rows containing only missing values were removed and duplicates were dropped. Both sources were then harmonized by selecting a set of common columns (`REFERENCE_OPERATION_T24`, `CLIENT_ID`, `BOOKING_DATE`, `DESCRIPTION_OPERATION`, `EXCHANGE_RATE`, `AMOUNT_FCY`, `AMOUNT_LCY`, `SENS`, `CURRENCY`, `MT_TND`, `MT_DEVISE`, `COURS_MARCHE`, `NATURE_CLIENT`, `MARGE_PNDR`), resulting in a coherent transactional dataset.

Client information was enriched in successive steps. First, the `employe` table was joined with `direction` to inherit the pole and label of each direction. Second, the `client` table was linked to its responsible employee through the `MATRICULE_CHARGE` key, thus attaching staff names and direction metadata to each client. Third, client affiliation to groups was ensured by normalizing the `ID_BCT` field and joining with `groupe`, which added the group name to the client profile. The result of these steps is the `client_enriched` table, a master view that consolidates client identifiers, socio-economic attributes, responsible employee information, and group membership.

Finally, transactional data from `df_client` was merged with `client_enriched`. This produced the final `data` table, which combines two levels of information: transactional attributes (amounts, currencies, exchange rates, and booking dates) and client attributes (identity, sector, group, employee, direction). This integration ensures that each transaction can be analyzed not only in its financial context but also within the organizational and client dimension. The resulting schema, summarized in Figure 3, provides a unified analytical dataset.

adjustbox

.

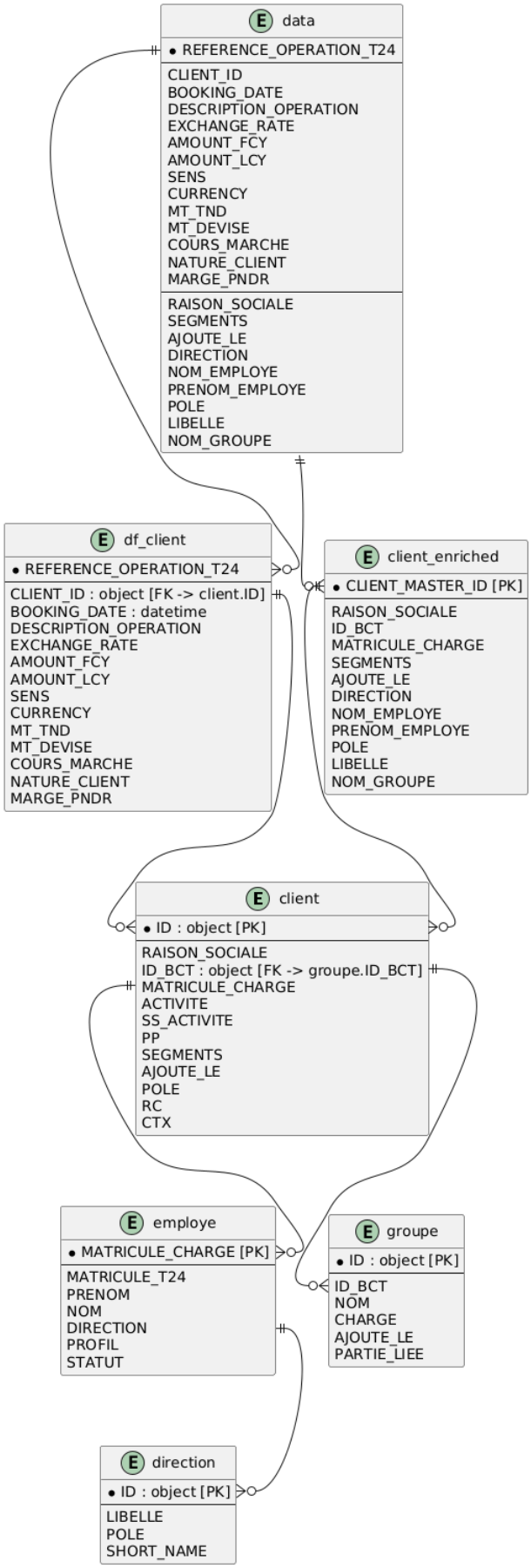


Figure 2.1: UML Schema of Data Enrichment Process

Chapter 3

Data Cleaning and Preprocessing

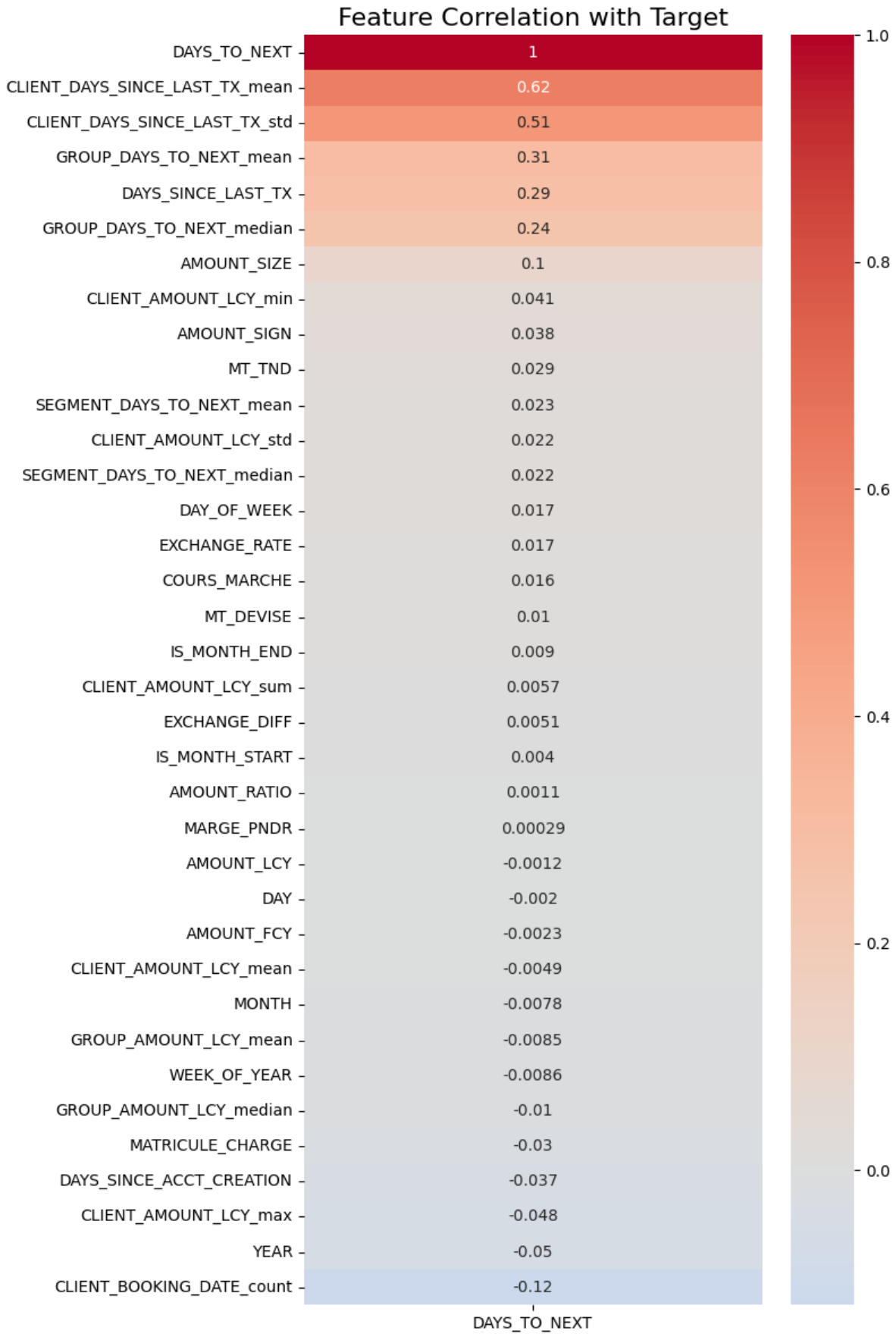
Data Cleaning

Before integration, a systematic data cleaning phase was carried out to ensure consistency and reliability of the dataset. First, rows containing only missing values were removed, and duplicate records across different files and periods were dropped. This step was essential since the raw `confi` and `negoci` sources were constructed from multiple yearly and quarterly extracts, which naturally introduced redundancy. Next, missing values in numerical columns such as transaction amounts, exchange rates, and margins were imputed according to the statistical distribution of each variable. For variables with approximately symmetric distributions, the mean was used, while for skewed variables (e.g., transaction amounts), the median was preferred, as it is more robust to extreme outliers. Categorical variables with missing values were handled either by introducing an explicit "Unknown" category or by propagating the most frequent category depending on the analytical relevance. All join keys such as client identifiers and group codes were normalized to a consistent string format (trimmed and uppercased) to prevent mismatches caused by whitespace or case differences. Finally, the `BOOKING_DATE` field was standardized to a consistent datetime format, ensuring its usability as a temporal anchor for subsequent analyses. These cleaning procedures established a coherent and consistent base, making the dataset suitable for enrichment and downstream modeling.

Feature Engineering

Feature engineering was designed to transform raw transactional and client-level data into informative variables suitable for predictive modeling. Temporal attributes were extracted from the `BOOKING_DATE`, including year, month, day, day of week, week of year, quarter, and flags for month start or end, which capture seasonality and calendar effects. Additional time-based features were created, such as the number of days since the account creation (`DAYS_SINCE_ACCT_CREATION`) and the days since the previous transaction (`DAYS_SINCE_LAST_TX`), providing a notion of recency and client activity dynamics. Financial attributes were engineered by deriving ratios and transforma-

tions: the relative ratio between foreign and local currency amounts (AMOUNT_RATIO), the deviation between recorded and market exchange rates (EXCHANGE_DIFF), the transaction sign (AMOUNT_SIGN), and the logarithmic transformation of transaction amounts (AMOUNT_LCY_LOG) to reduce skewness. At the aggregation level, client-wise statistics (mean, standard deviation, minimum, maximum, count, and total of transaction amounts, as well as recency-related measures) were computed to capture behavioral patterns across a client's history. Group-level aggregates by NOM_GROUPE were also derived to embed contextual information about collective financial activity. Transaction sequencing was encoded with features such as transaction order number and reverse ranking to reflect positional effects within a client's history. Missing values in numerical aggregates were imputed with median values for stability, while remaining gaps were replaced with zeros. Finally, categorical variables were label-encoded to ensure numerical compatibility with machine learning algorithms. Importantly, the prediction target was reformulated: rather than using the transaction date directly, the problem was cast as a regression task by defining DAYS_TO_NEXT (the gap to the next transaction) as the target variable, with a logarithmic transformation (DAYS_TO_NEXT_LOG) applied to stabilize variance and reduce the impact of extreme outliers.


Feature Correlation with Target
DAYS_TO_NEXT
CLIENT_DAYS_SINCE_LAST_TX_mean
CLIENT_DAYS_SINCE_LAST_TX_std
GROUP_DAYS_TO_NEXT_mean
DAYS_SINCE_LAST_TX
GROUP_DAYS_TO_NEXT_median
AMOUNT_SIZE
CLIENT_AMOUNT_LCY_min
AMOUNT_SIGN
MT_TND
SEGMENT_DAYS_TO_NEXT_mean
CLIENT_AMOUNT_LCY_std
SEGMENT_DAYS_TO_NEXT_median
DAY_OF_WEEK
EXCHANGE_RATE
COURS_MARCHE
MT_DEVISE
IS_MONTH_END
CLIENT_AMOUNT_LCY_sum
EXCHANGE_DIFF
IS_MONTH_START
AMOUNT_RATIO
MARGE_PNDR
AMOUNT_LCY
DAY
AMOUNT_FCY
CLIENT_AMOUNT_LCY_mean
MONTH
GROUP_AMOUNT_LCY_mean
WEEK_OF_YEAR
GROUP_AMOUNT_LCY_median
MATRICULE_CHARGE
DAYS_SINCE_ACCT_CREATION
CLIENT_AMOUNT_LCY_max
YEAR
CLIENT_BOOKING_DATE_count
1
0.62
0.51
0.31
0.29
0.24
0.1
0.041
0.038
0.029
0.023
0.022
0.022
0.017
0.017
0.016
0.01
0.009
0.0057
0.0051
0.004
0.0011
0.00029
-0.0012
-0.002
-0.0023
-0.0049
-0.0078
-0.0085
-0.0086
-0.01
-0.03
-0.037
-0.048
-0.05
-0.12
DAYS_TO_NEXT
1.0
0.8
0.6
0.4
0.2
0.0

Chapter 4

Data Understanding

To better interpret the dataset and prepare it for modeling, an exploratory data analysis (EDA) was conducted focusing on transaction dynamics, client behavior, and organizational structure. The analysis combines descriptive statistics with visualizations to highlight patterns, seasonality, and distributions across different attributes.

First, transaction activity was analyzed along the temporal axis. By breaking down operations per day of the week and per month, we observe strong cyclical patterns that align with business days and seasonal peaks. Figure 4.7 illustrates the variation in the average number of transactions per day-of-week, differentiated by client type, while Figure **??** highlights month-of-year fluctuations. Year-over-year comparisons further emphasize growth trends and disruptions, with 2022 showing an exceptional increase in volumes compared to 2021 and 2023 (Figure **??**).

In terms of transaction values, large disparities exist between client types. Aggregated measures of daily transaction volumes (`MT_TND`) reveal that negotiated clients tend to generate significantly higher amounts compared to entrusted clients, even though their transaction frequencies are more balanced (Figure **??**). This suggests structural differences in client portfolios that are crucial for downstream modeling.

Client-level characteristics were also examined. The distribution of client age at transaction (measured in days since account creation) displays multiple peaks (Figure 4.6), reflecting different cohorts of onboarding over time. Similarly, pie charts of categorical variables such as transaction description, sense (`SENS`), client segments, and direction assignments reveal the concentration of activity in a few dominant categories (Figure **??**). For example, a large share of operations falls under "Grandes Entreprises," and the currency distribution is heavily dominated by EUR and USD.

Finally, temporal dynamics of transaction frequency were studied through rolling averages and peak detection. This analysis highlights both short-term fluctuations and long-term seasonal patterns, confirming that transaction behavior is not stationary and requires time-aware modeling.

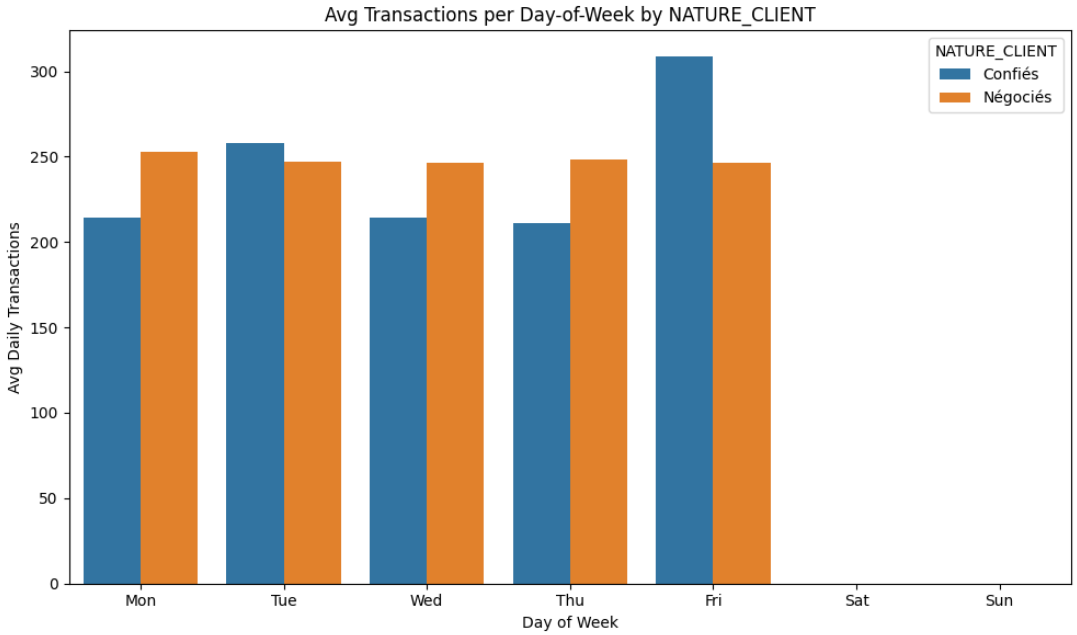


Figure 4.1: Average transactions per day of week by client type

These bar charts compare daily transaction activity between Confiés and Négociés clients. They show how transaction volume fluctuates throughout the week, with Fridays typically peaking for Confiés. This highlights behavioral differences in how client segments interact with the bank over time.

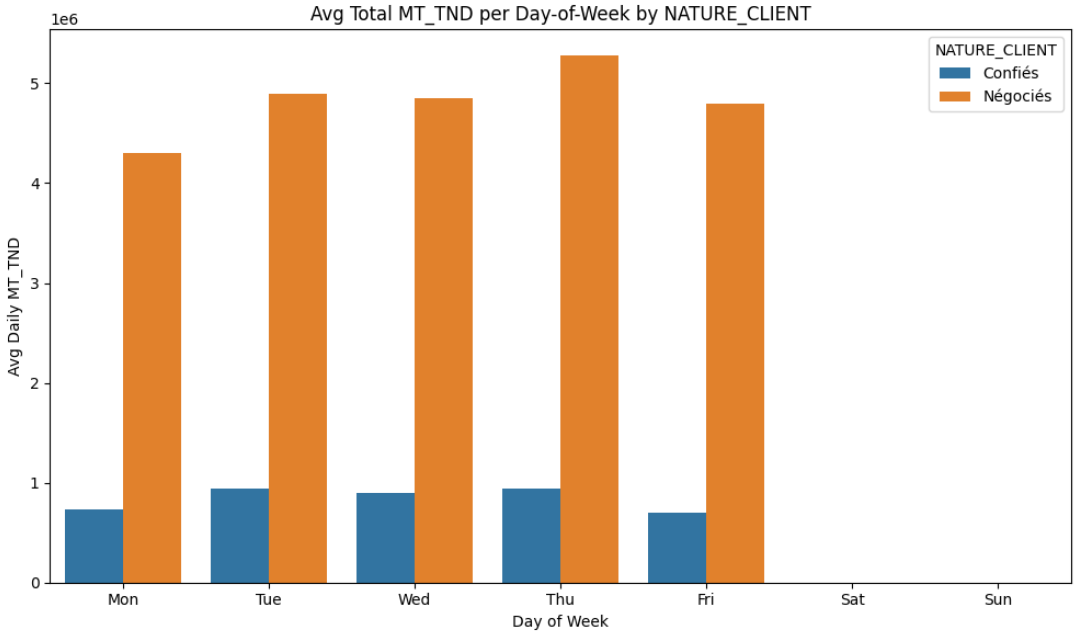


Figure 4.2: transaction values per day of week by client type

This chart highlights the difference in transaction values between Confiés and Négociés clients across weekdays. While Confiés generate more frequent transactions, their amounts remain relatively modest. In contrast, Négociés dominate in terms of value, with significantly higher average daily totals throughout the week. This confirms the structural distinction between high-frequency/low-value and low-frequency/high-value clients.

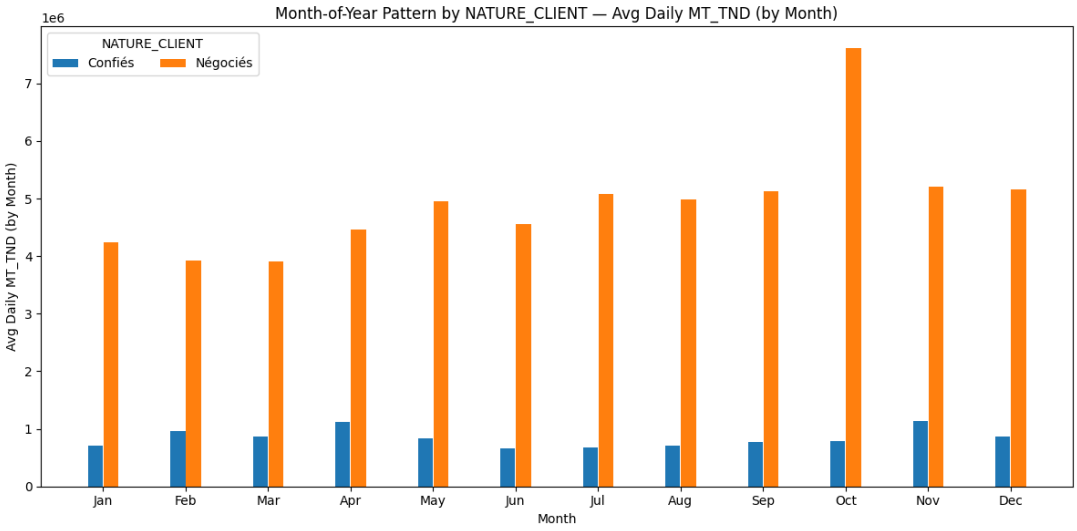


Figure 4.3: month-of-year pattern by client type

This chart illustrates the seasonality of transaction values across months. While Négociés consistently transact at higher volumes, both groups exhibit noticeable spikes in certain months, suggesting cyclical financial activities such as year-end or quarterly operations.

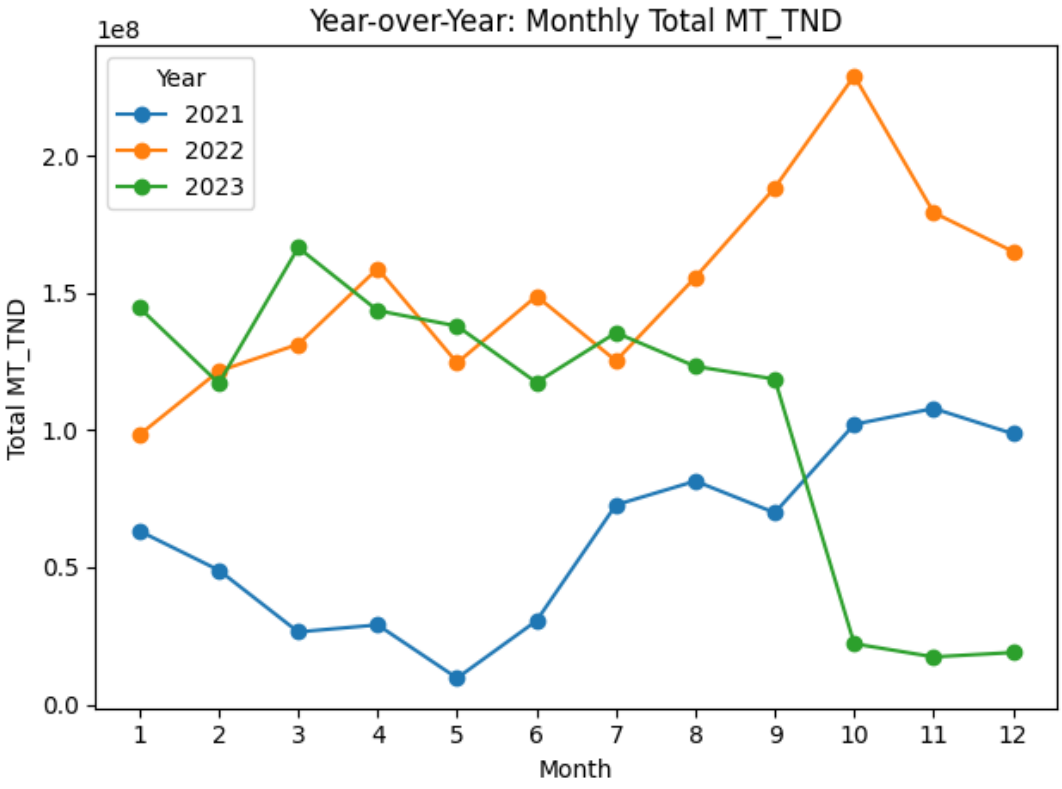


Figure 4.4: year-over-year evolution

This line chart compares total transaction values across 2021, 2022, and 2023. The modest rebound observed in 2022 can be linked to post-COVID recovery, as banking activity normalized following the disruptions of 2020–2021. However, the sharp decline in 2023 suggests a slowdown possibly tied to lingering macroeconomic effects of the pan-

demic, shifts in client behavior, or internal policy adjustments at the bank.

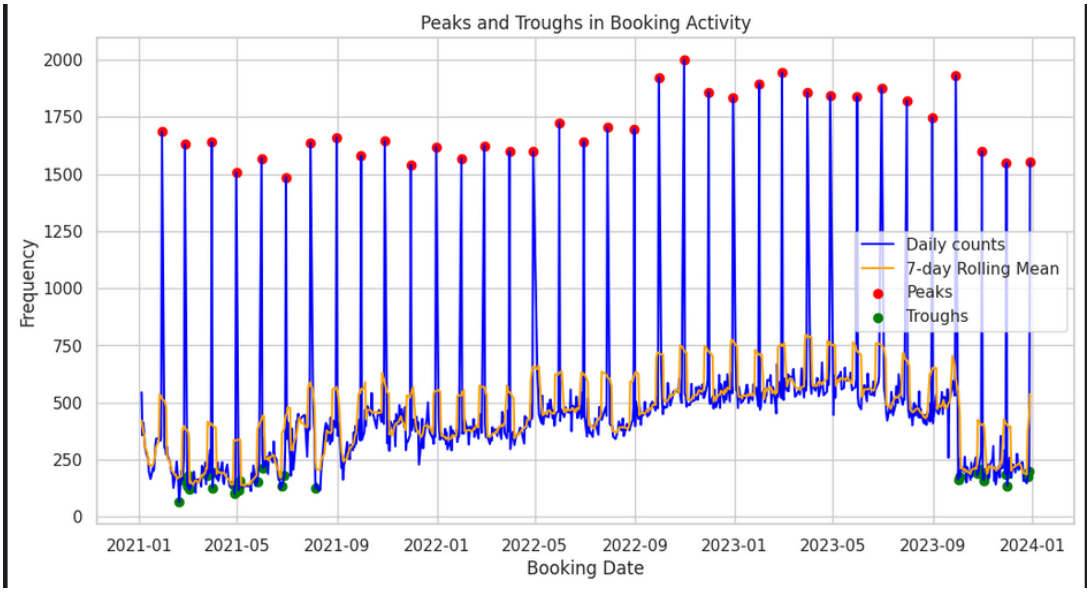


Figure 4.5: peak/trough analysis

This time-series plot highlights peaks and troughs in booking activity. The use of rolling averages smooths the series, while peak markers reveal periods of intense activity, contrasted with troughs of reduced engagement. This temporal insight is key for detecting seasonal cycles and stress periods.

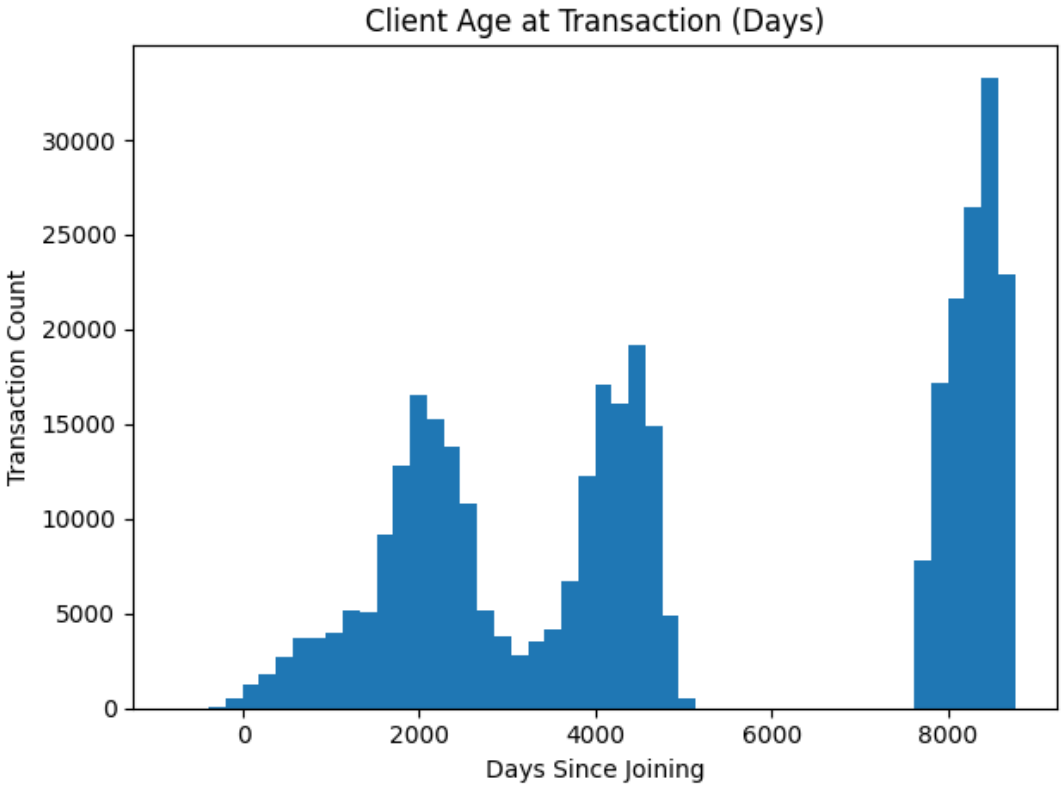


Figure 4.6: Client age distribution

This histogram shows the distribution of client "age" in terms of days since account creation. The multimodal structure indicates distinct cohorts of clients who joined at

different times. An important observation is that older clients (with longer tenure) tend to record more transactions overall. This suggests a loyalty or familiarity effect, where clients who have been with the bank longer are more actively engaged with its services.

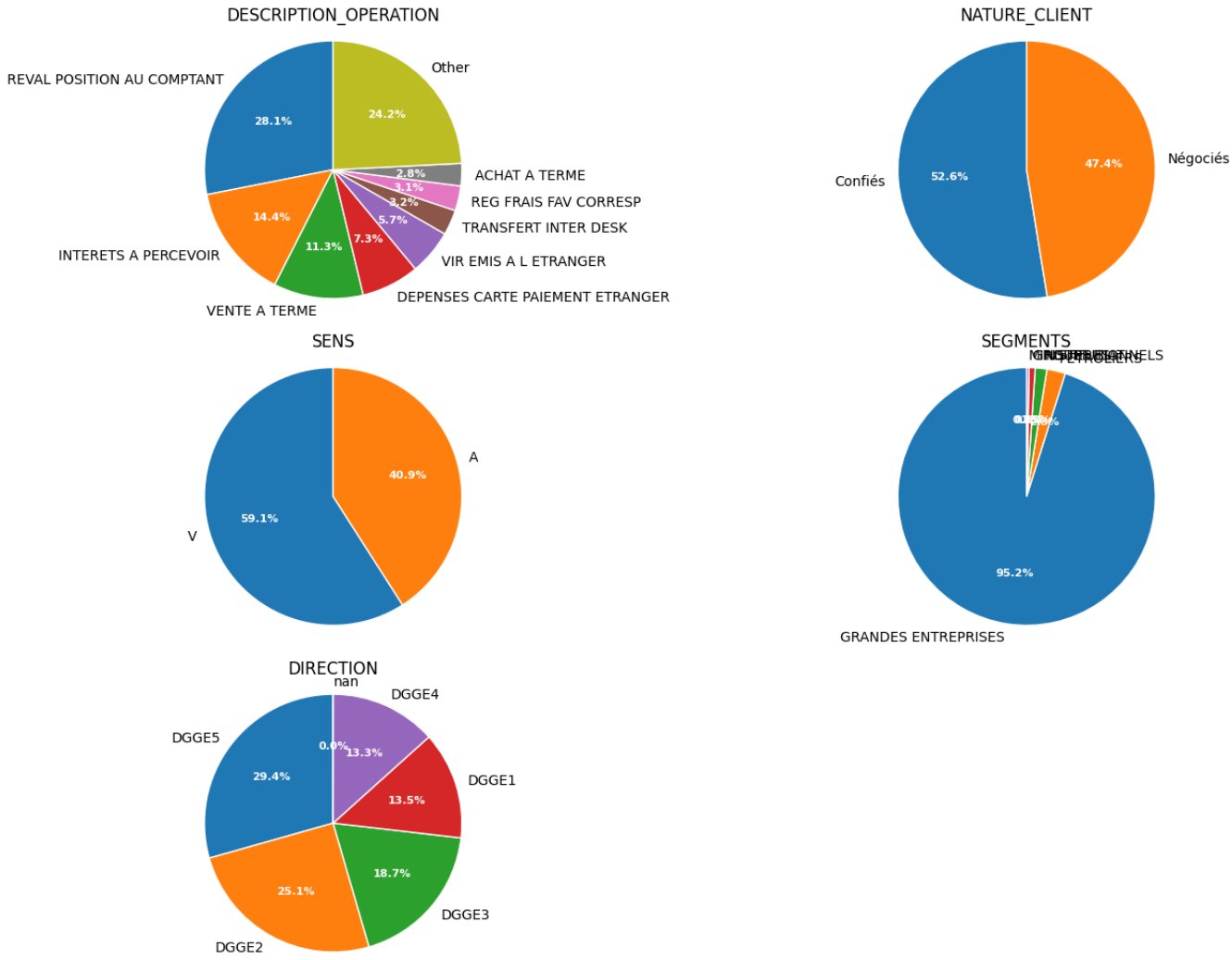


Figure 4.7: Currency and categorical pie charts

The categorical pie charts reveal a similar concentration dynamic: a handful of categories absorb most of the transactional activity, while others represent niche segments. This skew indicates that client activity is not evenly distributed — some groups consistently drive the bank's revenue while others contribute marginally. For example, clients in business-oriented or corporate categories are likely to appear in the "heavyweight" slices, as their operations generate recurring and higher transaction volumes. This distribution also sheds light on client lifecycle behavior: older clients are disproportionately represented in the high-volume categories, which explains why they tend to accumulate more transactions over time. Their loyalty and established financial patterns create a stable backbone of activity for the bank, while newer clients, still in the exploratory phase, appear in the lighter categories with fewer interactions.

Chapter 5

Modeling

The modeling task was framed as a supervised regression problem, where the objective was to predict the number of days until a client's next transaction (`DAYS_TO_NEXT`). A log transformation (`DAYS_TO_NEXT_LOG`) was also used to stabilize variance and handle skewness in inter-transaction intervals. Both global models and cluster-specific models were evaluated, with emphasis on performance, interpretability, and computational efficiency.

5.0.1 Problem framing

We predict the time-to-next transaction at the *transaction level*. For each transaction i by client c, with booking date t_i, the target is

$$y_i \;=\; \text{days\_to\_next}_i \;=\; (t_{i+1} - t_i) \text{ in days.}$$

The distribution of y is non-negative, right-skewed, and heavy-tailed, so we train on a stabilized target

$$z_i \;=\; \log(1 + y_i),$$

and invert at inference time by $\hat{y}_i = \exp(\hat{z}_i) - 1$.

We segment clients into two clusters using FRM features (Frequency, average Monetary value, Recency). Cluster 0 (high-frequency, low-value) is modeled with CatBoost plus explicit seasonality. Cluster 1 (low-frequency, high-value) is modeled with Ridge on the same log target. This chapter focuses on Cluster 0.

5.1 Data Splitting and Validation

Time-aware cross-validation was adopted to respect the temporal ordering of financial transactions. Clients were grouped so that no client appeared in both training and validation sets, avoiding leakage of client-specific transaction histories. This ensured that evaluation reflected real deployment scenarios where models must generalize to unseen future periods.

5.2 Baseline Models

Linear and Ridge regressions served as baselines. Their performance was moderate: although they achieved relatively high R^2 due to heterogeneity between clients (high-frequency vs. low-frequency traders), error metrics such as RMSE indicated limited ability to capture fine-grained transaction dynamics. These models, however, were extremely fast to train and provided a useful reference point. For example, in Cluster 1 (low-frequency, high-value clients), Ridge regression achieved an RMSE of 1.68 and $R^2 = 0.32$, showing that linear methods can capture some variability in sparse transaction patterns.

5.3 Modeling Cluster 0 with CatBoost and Fourier Seasonality

5.3.1 Why CatBoost for Cluster 0

Cluster 0 contains many events per client and rich categorical structure. Requirements:

- capture nonlinear interactions between amounts, recency, sequence indices, and categorical context;
- handle many categorical features without manual target encoding leakage;
- remain robust to class imbalance and heavy tails after log transform;
- support early stopping and calibrated validation.

CatBoost meets these requirements through ordered boosting and target statistics that avoid prediction shift. It trains symmetric oblivious trees that split on the same feature across each level, which yields strong regularization and fast inference.

5.3.2 CatBoost mechanics in brief

CatBoost builds an ensemble of trees $\{T_m\}_{m=1}^{M}$ with learning rate η, prediction

$$\hat{z}(x) \;=\; \sum_{m=1}^{M} \eta\, T_m(x).$$

Key ideas:

- **Ordered boosting**. Samples are processed in random permutations. Gradient estimates and target statistics for categoricals use only earlier items in the permutation, which removes the target leakage that appears in naive target encoding.

- **Categorical handling**. For a categorical feature $x^{(j)}$, CatBoost constructs ordered target statistics such as

$$\mathrm{CTR}(x^{(j)}) \;=\; \frac{\sum_{k\in\mathcal{P}(i)} y_k + a\cdot p}{|\mathcal{P}(i)| + a},$$

  where $\mathcal{P}(i)$ is the prefix set in the permutation, a is a prior strength, and p is a prior target. This is done per split, per permutation, with additional combinations of categorical features.

- **Symmetric trees**. Each tree is a balanced structure where each level uses the same split feature, which reduces overfitting and increases speed.

5.3.3 Seasonality with Fourier features

Calendar variables like day of week and month capture stepwise patterns only. We add smooth seasonal signals via Fourier terms. For a seasonality with period P days and order K, define for transaction time t (in days since a reference date t_0)

$$\forall k \in \{1,\dots,K\}: \quad \sin\big(2\pi k\, t/P\big),\;\; \cos\big(2\pi k\, t/P\big).$$

We use:

$$\begin{aligned} \text{weekly:}\;& P = 7,\; K = 3,\\ \text{monthly:}\;& P \approx 30.4375,\; K = 2,\\ \text{yearly:}\;& P \approx 365.25,\; K = 3. \end{aligned}$$

These smooth bases let CatBoost model cyclical effects and interactions with tabular features, for example higher activity on specific weekdays for specific segments.

5.3.4 Features

The design matrix includes:

- recency and sequence: `DAYS_SINCE_LAST_TX`, `TX_NUM`, `TX_REVERSE_NUM`, `DAYS_SINCE_ACCT_CREATION`
- amounts and ratios: `AMOUNT_LCY`, `AMOUNT_LCY_LOG`, `AMOUNT_RATIO`, `EXCHANGE_DIFF`;
- client and group aggregates: means, std, counts on amounts and gaps;
- calendar: `DAY_OF_WEEK`, `WEEK_OF_YEAR`, `QUARTER`;
- categoricals: `SENS`, `CURRENCY`, `NATURE_CLIENT`, `DIRECTION`, `SEGMENTS`, plus encoded text categories `DESCRIPTION_OPERATION_ENCODED`, `NOM_GROUPE_ENCODED` kept as pandas `category`;
- Fourier terms for weekly, monthly, yearly cycles as above.

5.3.5 Training protocol

Target. Train on $z = \log(1 + y)$ where y is days to next transaction.

Validation. We sort by `BOOKING_DATE` and use the last 20% as a time holdout. For hyperparameter search we use GroupKFold on the remaining 80% with groups set to `CLIENT_ID` to avoid client leakage across folds.

Objective and metrics. CatBoost is trained with RMSE on the log target. Reporting uses real days:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_i \big(y_i - \hat{y}_i\big)^2}, \quad \text{MAE} = \frac{1}{n}\sum_i \big|y_i - \hat{y}_i\big|, \quad R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}.$$

5.3.6 Hyperparameter optimization

We used Optuna on the 80% training split with client-based folds and pruning. The most impactful hyperparameters for CatBoost were:

- `iterations` with early stopping,
- `learning_rate`,
- tree `depth`,
- `l2_leaf_reg`,
- `bagging_temperature`, `random_strength`,
- `border_count`, and optionally `rsm` and `subsample`.

Note on baseline search. For completeness we record the best Optuna parameters from an earlier LightGBM quantile baseline:

```
{'learning_rate': 0.003687076821356058,
 'num_leaves': 151,
 'max_depth': 12,
 'feature_fraction': 0.8294958941197903,
 'bagging_fraction': 0.8758590995393195,
 'bagging_freq': 4,
 'min_child_samples': 59}
```

These were used for the LGBM quantile experiment and are not the CatBoost hyperparameters. The CatBoost search selected a different configuration and then early stopped at the iteration reported below.

5.3.7 Training trace

CatBoost training on the time-ordered split produced the following log excerpt:

```
0:   learn: 0.8507493  test: 0.8104946  best: 0.8104946 (0)  total: 182ms  remaining:
200: learn: 0.6655562  test: 0.6561258  best: 0.6560508 (191) total: 29.5s  remaining
400: learn: 0.6550555  test: 0.6612123  best: 0.6552821 (225) total: 58.9s  remaining
Stopped by overfitting detector (300 iterations wait)

bestTest = 0.6552820962
bestIteration = 225
Shrink model to first 226 iterations.
```

The *overfitting detector* halted training when the validation metric did not improve for 300 rounds. The model was shrunk to the best iteration.

5.3.8 Results on time holdout (real days)

$$\begin{aligned} \text{RMSE} &= 5.4053 \text{ days}, \\ \text{MAE} &= 1.6957 \text{ days}, \\ R^2 &= 0.2442. \end{aligned}$$

5.3.9 Inference

Given a feature vector x and booking date t, the model outputs $\hat{z}(x)$ on the log scale. We return

$$\hat{y}(x) \;=\; \exp\big(\hat{z}(x)\big) - 1, \quad \hat{t}_{\text{next}} \;=\; t + \lceil \hat{y}(x) \rceil \text{ days}.$$

Rounding up is safer for date arithmetic and avoids predicting a next date in the past.

5.3.10 Why Fourier helped

Fourier bases approximate periodic functions via sines and cosines. They efficiently encode:

- weekly cycles (business days vs weekends),
- monthly cycles (beginning or end of month effects),
- annual cycles (seasonal activity).

CatBoost learns nonlinear interactions between these smooth periodic signals and transactional context, for example different weekly patterns by `SEGMENTS` or `CURRENCY`. This improves stability compared to only using discrete calendar one-hots.

5.3.11 Ablations and practical notes

- Training on $\log(1+y)$ reduces variance and helps the trees fit proportional rather than absolute errors. Back-transform for metrics.
- Keep categorical columns as `category` dtype and pass them to CatBoost. Align category vocabularies across train and validation to avoid unseen category issues.
- Use time-based holdout for final reporting, group folds by client during search to prevent leakage.
- Early stopping is essential. The best iteration occurred far before the maximum allowed iterations.

5.4 Modeling Cluster 1 with Ridge regression.

For the low-frequency, high-value segment (Cluster 1), we adopted a Ridge regression on the stabilized target $z = \log(1+y)$ (with y the days-to-next transaction). Ridge is well-suited here because the sample is relatively small and noisy, and the ℓ_2 penalty improves generalization by shrinking coefficients toward zero. Concretely, we solve

$$\min_{\mathbf{w},b} \ \|\mathbf{z} - X\mathbf{w} - b\mathbf{1}\|_2^2 \ + \ \alpha\|\mathbf{w}\|_2^2,$$

with $\alpha = 1.0$. All numeric predictors were standardized and categoricals label-encoded; predictions are reported on the original scale via $\hat{y} = \exp(\hat{z}) - 1$.

5.4.1 Results

On the time-ordered holdout, Ridge achieved

$$\text{RMSE} = 1.6807 \text{ days},$$
$$R^2 = 0.3238.$$

indicating a strong linear signal for this cluster while keeping the model simple and auditable.

Limitations. CatBoost gives strong point estimates but not a full generative model of interarrival times.Long-horizon predictions remain inherently uncertain;

Bibliography

1. Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). *CatBoost: Unbiased boosting with categorical features*. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*.

2. De Livera, A. M., Hyndman, R. J., & Snyder, R. D. (2011). *Forecasting time series with complex seasonal patterns using exponential smoothing*. *Journal of the American Statistical Association*, 106(496), 1513–1527.

3. Taylor, S. J., & Letham, B. (2018). *Forecasting at scale*. *The American Statistician*, 72(1), 37–45.

4. Lundberg, S. M., & Lee, S.-I. (2017). *A unified approach to interpreting model predictions*. In *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*.

5. Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). *Optuna: A next-generation hyperparameter optimization framework*. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*, 2623–2631.

6. Pedregosa, F., Varoquaux, G., Gramfort, A., *et al.* (2011). *Scikit-learn: Machine learning in Python*. *Journal of Machine Learning Research*, 12, 2825–2830.

7. Ke, G., Meng, Q., Finley, T., *et al.* (2017). *LightGBM: A highly efficient gradient boosting decision tree*. In *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*.

8. McKinney, W. (2010). *Data structures for statistical computing in Python*. In *Proceedings of the 9th Python in Science Conference (SciPy 2010)*, 51–56.

9. Harris, C. R., Millman, K. J., van der Walt, S. J., *et al.* (2020). *Array programming with NumPy*. *Nature*, 585, 357–362.

10. Angelopoulos, A. N., & Bates, S. (2021). *A gentle introduction to conformal prediction and distribution-free uncertainty quantification*. *arXiv preprint* arXiv:2107.07511.

11. Lloyd, S. (1982). *Least squares quantization in PCM*. *IEEE Transactions on Information Theory*, 28(2), 129–137.