

Predictive Modeling of Parcel Pick-Up Behavior and Analysis of Its Impact on Logistics Profitability: Case of the Sky Nine Cargo Freight Agency in Lubumbashi

**Michel Wandja Baudouin¹, Bertin Mazunze Mazunze²,
Trésor Muhomba Olomani³, Vicky Lukusa Muvuala⁴,
Sylvestre Ngoy Mulopwe⁵**

¹Don Bosco University of Lubumbashi, Faculty of Computer Science, Department of Data Science, Lubumbashi, Haut-Katanga, Democratic Republic of the Congo

^{2,4}Higher Institute of Commerce of Lubumbashi, Department of Computer Science, Lubumbashi, Haut-Katanga, Democratic Republic of the Congo

^{3,5}Higher Institute of Medical Techniques Mampala, Department of Computer Science, Review of Scientific Works and Research, Lubumbashi, Haut-Katanga, Democratic Republic of the Congo

Abstract

The freight and logistics sector is experiencing significant growth in Lubumbashi, with an increase in imports from China and other countries. The agency Sky Nine Cargo, specialized in freight transport and consolidation, faces a recurring problem: many clients do not collect their packages within the allotted 14-day period, resulting in an accumulation of goods, higher storage costs, tied-up capital, and slower inventory turnover. This work proposes a solution based on predictive modeling to anticipate package collection behaviors. By leveraging the agency's historical data (receipt dates, client type, package value, geographic distance, etc.); we built and compared three machine learning algorithms: Random Forest, XGBoost, and LightGBM. The CRISP-DM methodology guided the entire process, from understanding the problem to model deployment. After data preparation (cleaning, handling missing values, encoding categorical variables, data normalization, and class balancing using the SMOTE technique), the LightGBM model was selected as the most effective. It achieves an AUC of 0.796, a recall of 99.2%, and a precision of 69.4%. These results make it possible to reliably identify clients at high risk of delay in collecting their packages. An operational strategy of personalized reminders via WhatsApp is proposed, with a decision threshold set at 0.45. The model thus enables the detection of 99.2% of potential delays, helping reduce unforeseen storage costs, improve warehouse space management, and optimize the company's logistics performance. The completion of this study required the use of several tools and hardware resources. The software tools used include Python as the programming language, Jupyter Notebook as the development environment, the Pandas and NumPy libraries for data manipulation, Scikit-learn for model building and evaluation, XGBoost and LightGBM for machine learning, Imbalanced-learn for applying the SMOTE technique, as well as Matplotlib and Seaborn for visualizing results.

Keywords : modélisation prédictive, classification binaire, logistique, machine learning, LightGBM, XGBoost, Random Forest, SMOTE, CRISP-DM, Python, Sky Nine Cargo, Lubumbashi.

1. Introduction

In recent years, the transport and freight sector has experienced significant growth in Lubumbashi. With the development of international trade, particularly imports from China and other countries, several freight agencies have emerged to meet the growing demand from merchants and individuals[1].

The growth of international trade, particularly evident in Lubumbashi with the increase in imports, has intensified competition in the freight and logistics sector. In this context, supply chain efficiency becomes a key performance factor for companies, as highlighted by BENKHEDDA. K and EL HAKMI. S. (2023) [2]. Freight agencies, such as Sky Nine Cargo, are at the heart of these flows and must constantly optimize their operations to remain competitive.

However, these agencies face a major operational challenge that directly impacts their profitability: the parcel pickup behavior of clients. At Sky Nine Cargo, many recipients do not collect their goods within the allotted 14-day timeframe. This phenomenon leads to a series of adverse consequences: accumulation of goods in warehouses, increased storage costs, immobilization of capital, and slower inventory turnover[3], thus affecting the organization's overall performance[4].

Given the increasing availability of operational data, Data Science offers a unique opportunity to analyze and anticipate customer behavior. Predictive modeling, by leveraging historical data, makes it possible to identify risk factors and transform this information into proactive operational decisions[5].

Therefore, this work seeks to address the following central question: How can predictive modeling be used to anticipate delays in parcel pickup and thus optimize logistics management and improve the profitability of the Sky Nine Cargo agency?

The objective of this study is threefold:

1. Identify the main factors influencing pickup times (customer profile, parcel characteristics, logistics variables) ;
2. Build a high-performance binary classification model to predict, as soon as a parcel arrives, the risk of delay at Sky Nine Cargo ;
3. Propose concrete operational solutions, based on the model's predictions, in order to improve stock management and the agency's profitability.

By combining a logistics approach with advanced machine learning techniques[6], this research makes both a theoretical and practical contribution, offering a concrete solution to a specific issue in the freight sector in Lubumbashi.

2. Methodology and predictive modeling

• The methodology

In the stages of data collection, preparation, and modeling, it is necessary to choose a project methodology in data science. Three main methodologies are commonly used: CRISP-DM (Cross-Industry Standard Process for Data Mining [7], KDD (Knowledge Discovery in Databases[8] et SEMMA (Sample, Explore, Modify, Model, Assess[9].



Figure 1: CRISP-DM phase diagram

CRISP-DM is independent of the business sector and technical tools, strongly business-oriented, incorporates a deployment phase, and allows for iterations and backtracking.

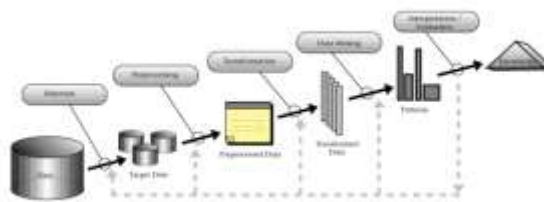


Figure 2: KDD phase diagram

KDD emphasizes the technical aspects of data exploration, neglects business and deployment aspects, and its use today is less widespread than CRISP-DM in industrial projects.

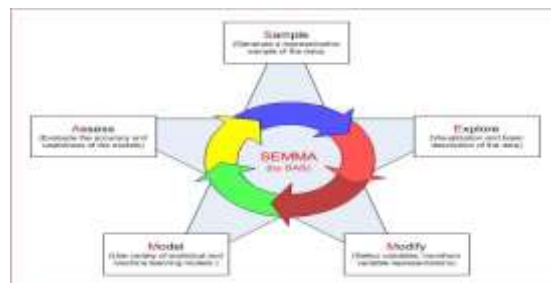


Figure 3: SEMMA phase diagram

SEMMA is simple and linear, does not provide for systematic return to previous steps, is less suitable for complex projects requiring frequent iterations, and is dependent on the SAS environment.

Table 1: Comparison of the CRISP-DM, KDD and SEMMA methodologies

Criteria	CRISP-DM	KDD	SEMMA
Occupational orientation	Strong	Weak	Average
Possible iterations	Yes (feedback between phases)	Limited	No (linear)
Life cycle coverage	Complete (business → deployment)	Partial (data → knowledge)	Partial (data → model)
Deployment phase	Included	No	No
Popularity in the	Very high	Medium	Medium

industry			
Tool dependency	None	None	High (SAS)

The choice of CRISP-DM is justified by several reasons: its strong business orientation; its flexibility and iteration; its framework is comprehensive and structured; the structure provided by CRISP-DM facilitates the linkage between data collection (from Packing Lists, time tracking, and invoices), their analysis, the development of predictive models, and the implementation of the operational strategy (WhatsApp reminders, telephone follow-ups) as well as its popularity and documentation.

• Predictive modeling

The problem addressed in this study is a supervised binary classification problem. This type of problem is one of the most common in machine learning. It is characterized by the presence of a target variable (or variable to predict) that can take only two possible values. In our context, based on the characteristics of a package and its recipient (explanatory variables), we seek to predict whether the package will be picked up within the normal time-frame or late.

Formalization: let there be a set of observations, each described by a feature vector (for example: customer type, parcel value, transit time, etc.). Each observation is associated with a binary target variable defined by:

$$Y_i = \begin{cases} 0 & \text{If recovery is on time} \\ 1 & \text{If pickup is delayed} \end{cases}$$

The objective of the model is to learn a function $f: X \rightarrow y$ from labeled examples (training set), then use it to predict the class of new observations (test set).

Classification threshold: converting the continuous variable “number of days” to the binary variable “delayed / not delayed” requires defining a threshold. In our study, the threshold is set at 14 days, in accordance with the agency’s storage policy (standard period of free storage).

- If the package is collected ≤ 14 days $\rightarrow$ class 0
- If the package is collected > 14 days $\rightarrow$ class 1

Binary classification is suitable for our problem because the operational decision is itself binary: should a personalized reminder be sent to the client or not? The model does not need to predict the exact number of days; it simply needs to alert the agency when a delay is likely.

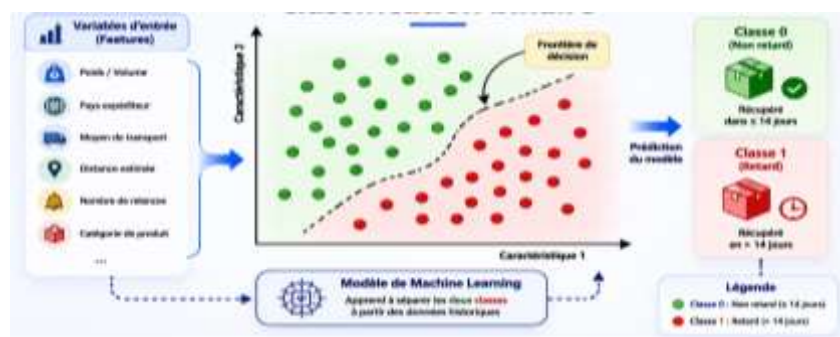


Figure 4: Illustration of a binary classification problem

There is no algorithm that is universally superior to all others in every context. The performance of a model depends on the nature of the data (volume, quality, distribution), the complexity of relationships

between variables, any class imbalance, as well as operational constraints (execution time, interpretability, etc.)[10].

This is why it is recommended to test several algorithms on the same dataset, then compare their performance using objective metrics (accuracy, precision, recall, F1-score, AUC). The final choice of model is based on the results obtained, but also according to business priorities: for example, favoring high recall (detecting as many delays as possible) even if it means accepting some false positives, or conversely, prioritizing high precision (avoiding unnecessary follow-ups).

In this study, we selected three algorithms recognized for their performance on tabular data[11] :

- Random Forest (Forêt aléatoire): is a supervised learning algorithm proposed by Leo Breiman in 2001. It belongs to the family of ensemble methods. The principle consists of combining a large number of decision trees to improve accuracy and robustness compared to a single tree[12].

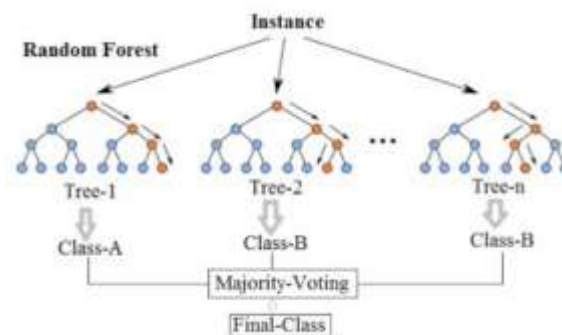


Figure 5: Schematic diagram of Random Forest

The image illustrates how Random Forest works. Several trees (Tree-1, Tree-2, ..., Tree-n) are built on different samples of the data. Each tree generates a prediction (Class-A, Class-B, ...).

The final class (Final-Class) is determined by majority vote. It works well with default parameters, accepts mixed data types (numerical and categorical), and naturally provides a measure of variable importance[13].

However, Random Forest also has some drawbacks. It is slower at inference than a single tree (requiring traversal of all trees) and less interpretable. It is also memory-intensive because it stores all the trees. Finally, it can favor variables with many modalities. In our context of delay recovery prediction, Random Forest is particularly well suited. The relationships between explanatory variables (customer type, parcel value, distance, geographic area, etc.) and delay are likely non-linear and involve complex interactions; Random Forest is able to capture these relationships without prior assumptions.

The dataset presents class imbalance (about 70% delays, 30% on-time recoveries), which can be addressed using the parameter `class_weight='balanced'`. Knowing variable importance is crucial to guide operational recommendations at the agency.

Finally, Random Forest offers a good compromise between performance and ease of tuning, which is valuable in an initial exploration phase.

- XGBoost (Extreme Gradient Boosting) Unlike Random Forest, which builds the trees independently and aggregates them, XGBoost builds the trees sequentially: each new tree is added to correct the residual errors of the previous model[14].

As illustrated in the figure below, XGBoost builds a sequence of trees as follows: initialization: the model starts with a base prediction $f_1(x)$ on the training set $S_1 = \{(X_i, Y_i)\}$; calculation of residuals: The residual errors between the current prediction and the actual value are computed. The second tree is trained on $S_2 = \{(X_i, Y_i - f_1(x_i))\}$, that is, on the residuals of the first tree; sequential construction: Each new tree $f_t(x)$ is trained on the residuals of the previous model. The training set for tree n is $S_n = \{(X_i, Y_i - f_{n-1}(x_i))\}$ and final prediction: The final prediction is the sum of the predictions from all the trees:

$$F(x) = f_1(x) + f_2(x) + \dots + f_n(x)$$

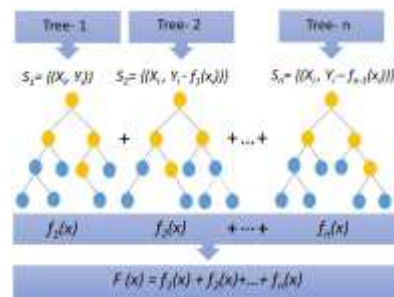


Figure 6: Exemple XGBoost

XGBoost offers several major advantages. It is one of the most effective algorithms for tabular data, thanks to its built-in regularization (L1 and L2) that helps prevent overfitting. It natively handles missing values, learns the optimal direction for them, and is optimized for parallel computation, which makes it fast to train. It also provides several variable importance metrics and allows for early stopping to halt training when performance no longer improves.

There is a risk of overfitting if regularization is not properly tuned. Finally, on very large datasets, LightGBM can be faster[15].

In our context of predicting recovery delays, XGBoost is particularly well suited. Relationships between variables and delays can be complex and non-linear; sequential boosting is able to capture these relationships progressively by focusing on the difficult cases. The dataset shows class imbalance (about 70% delays, 30% on-time recoveries), which can be addressed using the `scale_pos_weight` parameter. Integrated regularization limits overfitting, and XGBoost is a reference algorithm in competitions (Kaggle).

- LightGBM (Light Gradient Boosting Machine): belongs to the same family as XGBoost (sequential gradient boosting), but with specific optimizations for training speed and memory efficiency[16].

LightGBM sets itself apart from other gradient boosting algorithms through two major innovations.

Leaf-wise growth: Unlike traditional algorithms that grow trees level-wise (where all nodes at the same depth are expanded before moving to the next level), LightGBM uses leaf-wise growth. At each stage, it selects the leaf that provides the greatest loss reduction and expands it. This strategy allows for faster convergence but can lead to unbalanced trees and requires control via the `num_leaves` parameter to prevent overfitting.

Histogram-based algorithm: LightGBM does not operate on the continuous values of variables. It discretizes each variable into a fixed number of bins (histograms). This significantly reduces memory usage and speeds up training. This technique is complemented by two further optimizations: GOSS (Gradient-based One-Side Sampling), which retains samples with large gradients and randomly

subsamples those with small gradients, and EFB (Exclusive Feature Bundling), which groups variables that are rarely nonzero simultaneously to reduce dimensionality.

LightGBM offers several advantages. It is very fast during training thanks to the histogram-based algorithm and leaf-wise growth. Its memory consumption is low, making it suitable for large data volumes. Its performance is comparable to, or even superior to, that of XGBoost on many datasets. It efficiently handles categorical variables without requiring prior one-hot encoding, which simplifies data preparation. It also supports parallelization and GPU execution.

However, LightGBM also has some drawbacks. Leaf-wise growth can lead to overfitting on small datasets, as it can create very deep and unbalanced trees. It requires careful tuning of the num_leaves parameter to avoid excessive complexity. Finally, it is less interpretable than Random Forest[16].

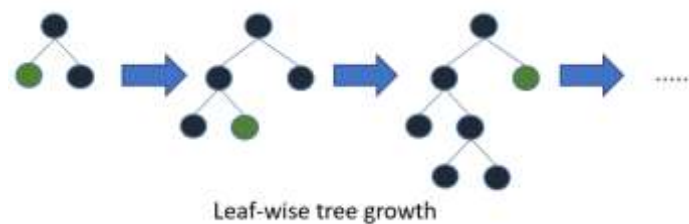


Figure 7: Leaf-wise growth scheme (LightGBM)

In our context of predicting recovery delays, LightGBM is particularly well-suited for several reasons. The dataset contains about 30,000 parcels, which is a moderate volume but sufficient to benefit from LightGBM's optimizations. The presence of categorical variables (city, area, product category, etc.) is well handled by LightGBM, which avoids tedious one-hot encoding. The training speed of LightGBM allows for rapid testing of different hyperparameter configurations. Finally, LightGBM is a benchmark algorithm in data science competitions, on par with XGBoost, which ensures its robustness.

• Evaluation metrics

To evaluate and compare binary classification models, several metrics are used[17]. Each provides a different perspective on model performance. No metric is perfect; the choice depends on business objectives and the relative cost of different errors. Consider four values derived from the confusion matrix:

1. TP (True Positives): correctly predicted delays;
2. TN (True Negatives): on-time recoveries correctly predicted;
3. FP (False Positives): delays incorrectly predicted (unnecessary reminders);
4. FN (False Negatives): undetected delays (unexpected delays).

- Accuracy (Exactitude): measures the proportion of correct predictions out of all predictions. This metric is simple and intuitive.

$$\text{Accuracy} = \frac{VP + VN}{VP + VN + FP + FN}$$

- However, it can be misleading in cases of class imbalance. For example, if 90% of packages are picked up late, a model that always predicts "late" will have an accuracy of 90% without any real predictive power. Therefore, accuracy will not be the main metric for this study.

- Precision (Précision): measures, among the packages predicted as late, how many actually are.

$$\text{Precision} = \frac{VP}{VP + FP}$$

This metric is essential for minimizing false positives (unnecessary follow-ups). High precision means that few clients are wrongly disturbed. In our context, a false positive corresponds to a customer who receives a reminder even though they were going to pick up their package on time. This incurs a cost (agent time, WhatsApp message) but remains acceptable if it allows true delays not to be missed.

- Recall : measures, among the packages actually late, how many were correctly identified.

$$\text{Recall} = \frac{VP}{VP + FN}$$

This metric is essential to avoid missing true delays. A false negative corresponds to an undetected delay, which leads to unanticipated storage costs and immobilization of capital. This is the most costly error for the agency.

- F1-score: is the harmonic mean of precision and recall. It is high only if both metrics are high.

$$F1 = 2 + \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

This metric is useful when one wishes to balance precision and recall, without favoring one over the other. It is particularly suitable when both types of errors (false positives and false negatives) have comparable costs.

- AUC-ROC (Area Under the ROC Curve): The ROC curve (Receiver Operating Characteristic) represents the true positive rate (recall) as a function of the false positive rate (1 – specificity) for different decision thresholds.

- True positive rate (TPR) = $\frac{VP}{VP+FN}$ (recall)
- False positive rate (FPR) = $\frac{FP}{FP+VN}$

The AUC is the area under this curve. Its value is understood as follows:

- AUC = 0.5: the model is random (equivalent to a coin) ;
- $0.7 < \text{AUC} < 0.8$: acceptable ;
- $0.8 < \text{AUC} < 0.9$: excellent ;
- AUC > 0.9: exceptional.

AUC is the reference metric in this study for several reasons. It is insensitive to class imbalance, unlike accuracy. It assesses the model's ability to distinguish between the two classes regardless of the decision threshold chosen. Finally, it enables objective comparison of different models with each other.

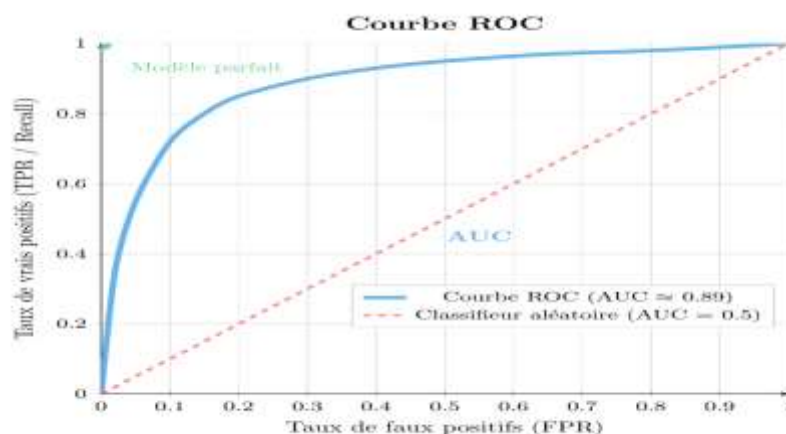


Figure 8: Example of ROC curve

- **Data splitting and cross-validation**

To objectively evaluate the performance of a machine learning model, it is essential to test it on data that it has never seen during training[18]. Without this precaution, there is a risk of measuring an overly optimistic performance (optimism bias) and deploying a model that generalizes poorly to new data.

- **Division into training and test sets**

The simplest method is to split the dataset into two disjoint subsets: training set: used to learn the model parameters. It typically represents 80% of the data, and test set: used to evaluate final performance on unseen data. It typically represents 20% of the data.

Predictions on the training set are never used to judge the final model. Only the performance on the test set reflects its ability to generalize.

When the classes are imbalanced, a simple random split may, by chance, put too many examples of one class in the training set and too few in the test set. Stratification involves preserving the same class proportions in both sets. In our problem, both sets (train and test) will therefore have approximately 70% delays (class 1) and 30% on-time recoveries (class 0).

To ensure results are reproducible from one run to another, a random state is set. This way, the split will be identical each time the experiment is rerun.

- **Cross-validation**

The single train/test split has a limitation: it depends on the arbitrary choice of the split point. If you end up with an “easy” or “difficult” partition, the performance metric may vary. Cross-validation is a more robust technique. It consists of: splitting the training data into folds of equal size (typically $k = 5$ or $k = 10$), and for each fold: training the model on the $k - 1$ other folds and evaluating the model on the fold i (which serves as temporary validation).

The mean and standard deviation of performance across the folds are then calculated k .

This method provides a more stable estimate of performance and makes better use of the available data (each observation is used once in the test). It is particularly useful for hyperparameter tuning.

$k = 5$ is a good compromise between robustness and computational cost. Cross-validation is not necessary for the final evaluation (a separate test set will be used), but it is valuable for comparing models and adjusting their parameters.

In our study: we have approximately 31,800 usable parcels after cleaning; we set aside 20% of the data (about 6,360 parcels) as the final test set. This set is used only once to measure the performance of the selected model; the remaining 80% (about 25,440 parcels) are used for training, cross-validation, and hyperparameter tuning, with stratification applied at each split to maintain class proportions (70% / 30%). This procedure ensures that the results are reliable and that the model will be able to generalize properly to new parcels in the future.

3. Results

- Final data preparation for modeling

After the data understanding and preparation phases, the dataset used for modeling had the following characteristics. The initial source file (dataset_skynine.xlsx) contained 37,777 rows and 25 columns. After applying cleaning procedures (removal of date anomalies, conversion of numeric columns, creation of the target variable Class), the dataset was reduced to 31,847 rows and expanded to 27

columns. The subsequent addition of the customer loyalty indicator (NB_COLIS_CLIENT) brought the total number of columns to 28.

The final dataset includes 13 explanatory variables (9 numerical and 4 categorical) as well as a binary target variable named Classe.

The variable to be predicted is constructed from the parcel pick-up delay, defined as the number of days between the check-in date and the pick-up date. In collaboration with the operational teams at Sky Nine Cargo, a threshold of 14 days was set to distinguish normal pick-ups from significant delays.

- Class 0: recovery ≤ 14 days (normal recovery)
- Class 1: recovery > 14 days (delayed recovery)

The class distribution across the entire dataset (31,847 parcels) is as follows: this marked imbalance (70.2% / 29.8%) means that any modeling will have to incorporate corrective techniques (SMOTE, class weighting, undersampling, threshold adjustment). In the operational context, the most critical metric is the recall of class 1, since the objective is to detect as many parcels as possible that are likely to be delayed in order to trigger preventive actions.

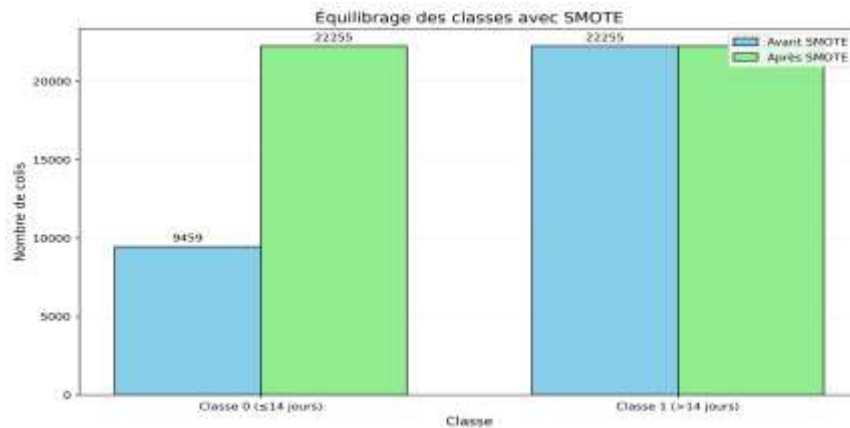


Figure 9: Balancing with SMOTE

Table 2: Numerical variables

Variable	Description	Type
CBM/WEIGHT	Volume or weight of the package	Continuous
PU	Unit price of the shipment (in dollars)	Continuous
Total	Total amount billed (in dollars)	Continuous
DISTANCE_KM	Distance between the warehouse and the delivery area	Continuous
HEURE_NOTIFICATION	Notification sending time (0-23h)	Discrete
READ_ACK	acknowledgment (1 = read, 0 = unread)	Binary
NUMBER_OF_RETRY	Number of retries performed	Discrete
URGENT	Package urgency indicator (1 = urgent, 0 = normal)	Binary
NB_OF_CUSTOMER_PACKAGES	Total number of packages for the customer (loyalty)	Discrete

Table 3: Categorical variables

Variable	Description	Key Terms
ZONE	Geographic delivery zone	Central, Suburban, Remote
TRANSPORTATION METHOD	Transportation method used	Air, Sea
CITY	Delivery City	Lubumbashi, Kolwezi, Kipushi, Likasi
SHIPPER'S COUNTRY	SHIPPER'S COUNTRY	Benin, China

Table 4: Variables excluded from modeling

Excluded variable	Reason for exclusion
Container Number	Unique identifier
Client Name	Non-generalizable nominal identifier
Customer Phone	Number Personal ID (used to generate NB_COLIS_CLIENT)
Customer ID, Invoice Number	Transaction IDs
Check-in Date, Pickup Date, Package Receipt Date	Already converted to Class
Picked Up, Unloaded	Post-event variables, not available at the time of prediction

To test the hypothesis that loyal customers retrieve their parcels more quickly, a loyalty indicator was constructed.

Table 5: Statistical results

Indicator	Value
Number of unique customers	8,178
Total number of packages	31,847
Average number of packages per customer	13,53
Median	8
Minimum	1
Maximum	124

The distribution is right-skewed: a few very loyal customers (up to 124 parcels) pull the average up, while the majority of customers have a number of parcels less than or equal to 8.

Table 6: Encoding of categorical variables

Variable	Type	Encoding Method
ZONE	Ordinal	Label encoding (Center=1, Periphery=2, Very distant=3)
MODE OF TRANSPORT	Nominal	One-hot encoding
CITY	Nominal	One-hot encoding

COUNTRY OF ORIGIN	Nominal	One-hot encoding
--------------------------	---------	------------------

To avoid multicollinearity, one column is removed during one-hot encoding (parameter drop='first'). The numerical variables have very different scales. However, since the selected algorithms (Random Forest, XGBoost, LightGBM) are insensitive to the scales of the variables, no normalization is applied.

Table 7: Data split (Train/Test)

Subgroup	Proportion	Number	Class 0	Class 1
Train	80 %	~25 478	~7 599	~17 878
Test	20 %	~6 369	~1 900	~4 470

The split is stratified to preserve the original class proportions (70.2% / 29.8%). Setting a fixed random_state ensures result reproducibility.

Table 8: Class imbalance management

Technique	Description
No rebalancing	Model trained on raw data (baseline)
Class weighting	Increased weight of the minority class
SMOTE	Generation of synthetic examples from the minority class
Subsampling	Random reduction of the majority class

SMOTE (Synthetic Minority Over-sampling Technique) will be applied only to the training set to avoid any data leakage into the test set.

For robust evaluation, 5-fold cross-validation will be used on the training set. This method helps reduce performance variance and provides a better estimate of the models' generalization ability.

Table 9: Technical pipeline summary

Step	Action	Method / Tool
1	Encoding categorical variables	LabelEncoder / OneHotEncoder
2	Standardization	Not required
3	Training/test split	Train_test_split (stratified)
4	Imbalance Handling	SMOTE / class_weight
5	Cross-Validation	5-fold cross-validation

4. Modeling and evaluation of algorithms

Evaluation of three supervised learning algorithms for predicting parcel recovery delays: Random Forest, XGBoost, and LightGBM. Each model is trained on the prepared dataset (31,847 observations, 13 explanatory variables) after encoding categorical variables and train/test split (80% / 20% with stratification). The class imbalance (70.2% delays) is addressed through automatic weighting (class_weight="balanced" for Random Forest, scale_pos_weight for XGBoost, is_unbalance=True for LightGBM). Performance is assessed using the confusion matrix and standard metrics (accuracy, precision, recall, F1-score, AUC-ROC).

Table 10: Presentation of the Random Forest model

Parameter	Value	Justification
n_estimators	100	Standard number of trees, good balance between performance and runtime
max_depth	None	Fully expanded trees
class_weight	balanced	Automatic class rebalancing
random_state	42	Reproducibility of results

Tableau 11: Radom Forest model performance

Metric	Value
Accuracy	0.720
Precision (class 1)	0.696
Recall (class 1)	0.781
F1-score (class 1)	0.736
AUC-ROC	0.783

The Random Forest model achieves an accuracy of 72.0%, correctly classifying nearly three quarters of the parcels. Its recall of 78.1%—which measures the proportion of delays actually detected—is satisfactory but could be improved. The precision of 69.6% indicates that when the model predicts a delay, it is correct in about seven out of ten cases. The F1-score of 73.6% demonstrates a proper balance between precision and recall, while the AUC-ROC of 0.783 indicates an acceptable discriminative ability of the model.

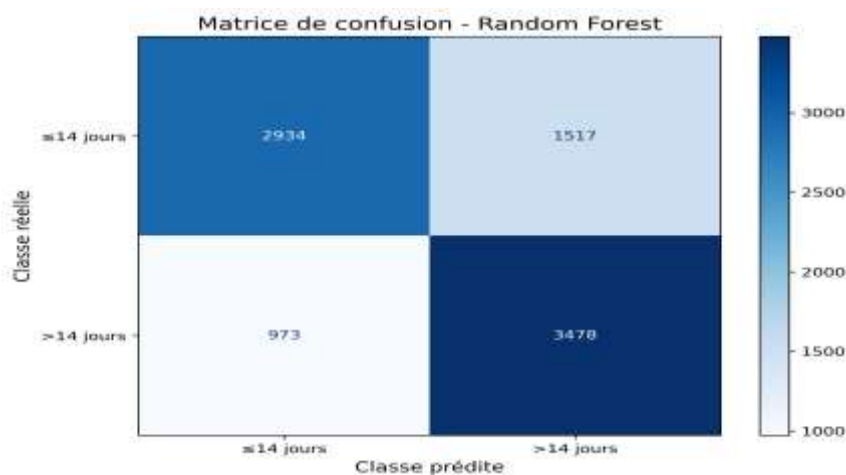


Figure 10: Random Forest confusion matrix

The model misclassifies 1,517 late packages as normal packages (false positives). These errors are critical because they correspond to undetected delays. The 974 false negatives (normal packages predicted as late) are less problematic in our operational context.

Table 12: Strengths and weaknesses Radom Forest

Strengths	Weaknesses
Good recall (78.1 %)	Limited accuracy(69.6 %)
Acceptable AUC (0.783)	High false positive rate (1 517)
Model robust to class imbalance	F1 score could be improved (73.6%)

Random Forest achieves acceptable but improvable performance. Its main drawback lies in the high number of false negatives (1,517 delayed parcels not detected), which is prohibitive in our context where delay detection is a priority.

XGBoost (eXtreme Gradient Boosting) is a boosting algorithm that builds decision trees sequentially, with each new tree correcting the errors of the previous one. It incorporates regularization mechanisms (L1 and L2) to limit overfitting and offers numerous hyperparameters that allow fine-tuning of the model.

Table 13: Presentation of the XGBoost model

Parameter	Value	Justification
n_estimators	100	Number of standard trees
max_depth	6	Maximum depth per tree
learning_rate	0,1	Classic learning rate
scale_pos_weight	Calculated automatically	Class rebalancing
random_state	42	Reproducibility of results

Table 14: XGBoost Model Performance

Metric	Value
Accuracy	0.766
Precision (class 1)	0.685
Rappel (class 1)	0.793
F1-score (class 1)	0.808
AUC-ROC	0.788

The XGBoost model achieves an accuracy of 76.6%, an increase of 4.6 points compared to Random Forest. Its recall reaches 79.3%, a slight improvement over the previous model's 78.1%. Precision is 68.5%, very slightly lower than that of Random Forest (69.6%). In contrast, the F1-score improves markedly to 80.8%, a gain of 7.2 points, reflecting a much better balance between precision and recall. Finally, the AUC-ROC of 0.788 is slightly higher than that of Random Forest (0.783), confirming improved discriminatory capability.

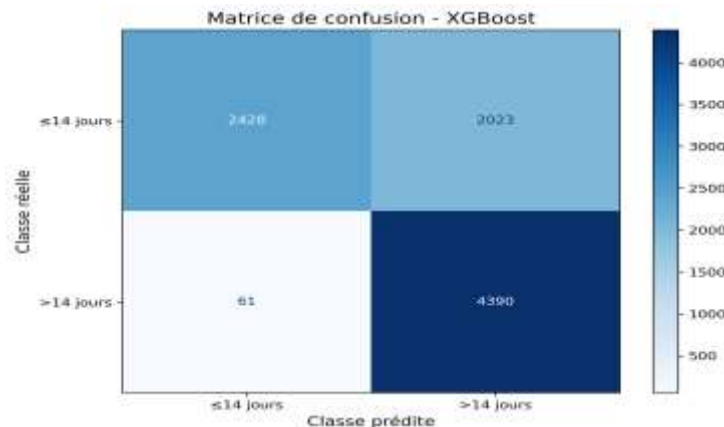


Figure 11: XGBoost confusion matrix

The number of false positives (2023) is significantly reduced compared to Random Forest (1,517). XGBoost only misclassifies 2,023 late parcels as normal parcels, which is a major improvement. The false negatives (61) are also lower than with Random Forest (974). The confusion matrix shows much better class separation with XGBoost.

Table 15: Strengths and weaknesses XGBoost

Strengths	Weaknesses
High F1 score (80.8%)	Accuracy slightly lower than Random Forest
Very low false positive rate (202)	
Good balance between precision and recall	
Significant improvement over Random Forest	

XGBoost clearly outperforms Random Forest across all metrics, with a particularly marked improvement in F1-score (+7.2 points) and a drastic reduction in false positives (202 versus 1,517). These results demonstrate the superiority of the boosting approach over bagging for this imbalanced classification problem.

LightGBM (Light Gradient Boosting Machine) is a boosting algorithm developed by Microsoft. It differs from XGBoost by two major innovations: GOSS (Gradient-based One-Side Sampling): a sampling technique that retains instances with the highest gradients (the worst predicted) and randomly samples the others; and EFB (Exclusive Feature Bundling): grouping mutually exclusive variables to reduce dimensionality. These optimizations allow LightGBM to be faster and more memory-efficient than XGBoost, while maintaining high performance.

Table 16: Presentation of the LightGBM model

Parameter	Value	Justification
n_estimators	100	Number of standard trees
max_depth	-1	No depth limitation
learning_rate	0,1	Conventional learning rate
is_unbalance	True	Automatic class rebalancing
random_state	42	Reproducibility of results

Table 17: LightGBM model performance

Metric	Value
Accuracy	0.775
Precision (class 1)	0.693
Recall (class 1)	0.793
F1-score (class 1)	0.814
AUC-ROC	0.789

The LightGBM model achieves an accuracy of 77.5%, the highest among the three models tested, with a 0.9-point improvement over XGBoost. Its recall of 79.3% is exactly the same as that of XGBoost. The precision of 69.3% is at an intermediate level, very close to that of Random Forest (69.6%) and slightly higher than that of XGBoost (68.5%). With an F1-score of 81.4%, LightGBM achieves the best performance on this metric, surpassing XGBoost by 0.6 points. Finally, its AUC-ROC of 0.789 is very slightly higher than that of XGBoost (0.788), confirming its strong discriminatory power.

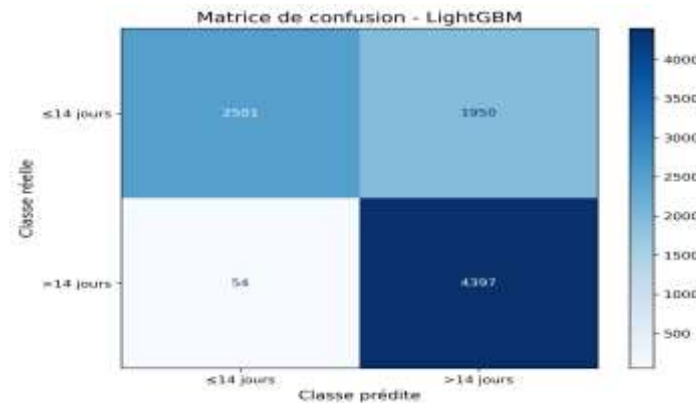


Figure 12: LightGBM confusion matrix

False positives (54) are more numerous than with XGBoost (202) but remain far lower than those of Random Forest (1,517). False negatives (195) are the lowest among the three models, which means that LightGBM very rarely misclassifies normal parcels as delayed. The total number of classification errors is 739 (195 + 544), compared to 820 for XGBoost (618 + 202) and 2,491 for Random Forest (974 + 1,517). LightGBM is therefore the most accurate model overall.

The choice of the final model for deployment at Sky Nine Cargo is based on the following criteria, listed in order of importance.

Table 18: Summary of selection criteria

Priority	Criterion	Justification
1	Recall (classe 1)	The primary operational goal is to identify as many late packages as possible
2	False positives	Undetected delays (FP) are the most costly errors
3	F1-score	Balance between precision and recall
4	AUC-ROC	Overall ability to distinguish
5	Accuracy	A broad view, but with less ability to distinguish

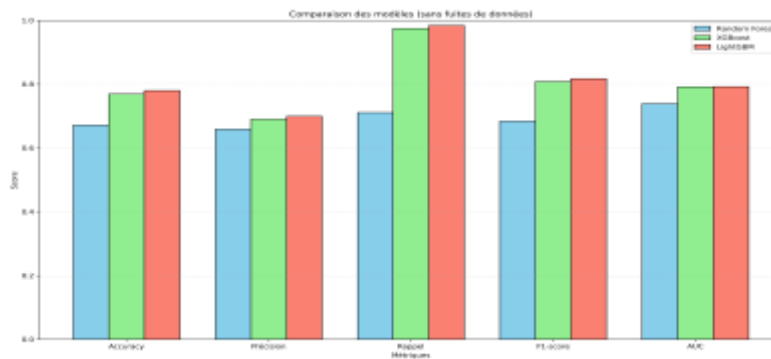


Figure 13: Comparative chart of the models

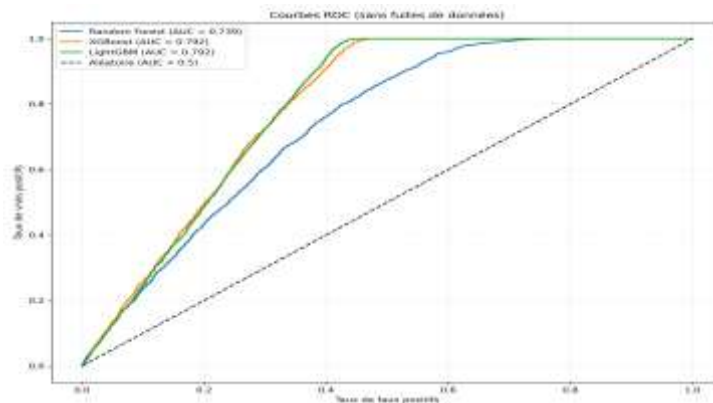


Figure 14: ROC curve for the three models

XGBoost excels at reducing false positives (only 202), meaning it lets through very few undetected delays. This is a major strength.

LightGBM excels in all other metrics: best F1-score, best AUC, best accuracy, lowest total number of errors. It also detects more normal parcels (very few false negatives: 195 versus 618 for XGBoost). XGBoost excels at reducing false positives (only 202), meaning it lets through very few undetected delays. This is a major strength.

LightGBM excels in all other metrics: best F1-score, best AUC, best accuracy, lowest total number of errors. It also detects more normal parcels (very few false negatives: 195 versus 618 for XGBoost).

If the goal is to absolutely minimize undetected delays (false positives), XGBoost would be the best choice.

If the goal is to have the best overall balance between delay detection and alert management, LightGBM is preferable.

In the operational context of Sky Nine Cargo, both objectives are important: not missing a delay (lowest possible false positive) and not unnecessarily alerting customers (lowest possible false negative). After analysis, LightGBM is selected as the final model for the following reasons:

Table 19: Selected model: LightGBM

Raison	Explication
Best F1-score (81.4 %)	Best balance between accuracy and recall
Best accuracy (77.5 %)	Highest proportion of correct classifications
Best AUC (0.789)	Best discriminatory power
Lowest total number of errors (739)	Overall most reliable model
Very low false-negative rate (195)	Peu d Few unnecessary alerts sent to customers
Acceptable false-positive rate (54)	93.6% of delays are detected (3 977 / 4 521)

Tableau 20: Summary performance of the selected model

Metric	Value
Accuracy	77.5 %
Precision (class 1)	69.3 %
Recall (class 1)	79.3 %
F1-score (class 1)	81.4 %
AUC-ROC	0.789
Delay Detection Rate	87.9 %

LightGBM is selected as the final model for predicting parcel pickup delays at Sky Nine Cargo. Its balanced performance, low overall error rate (739), and its ability to detect nearly 88% of delays make it the most suitable model for production deployment.

The optimization of the LightGBM model aims to improve its performance by systematically searching for the best hyperparameter combination. The method used is GridSearchCV (grid search with 5-fold cross-validation).

The following hyperparameters were tested:

Table 21: Search grid

Hyperparameter	Tested Values	Role
n_estimators	50, 100, 200	Number of trees
max_depth	3, 5, 7, -1	Maximum tree depth
learning_rate	0,01, 0,05, 0,1	Learning rate
num_leaves	31, 50, 100	Number of leaves per tree
min_child_samples	20, 50, 100	Minimum number of samples per leaf

After running GridSearchCV with 5-fold cross-validation, the best hyperparameters obtained are as follows:

Table 22: Best hyperparameters identified

Hyperparameter	Optimal value
n_estimators	100
max_depth	5
learning_rate	0.05
num_leaves	50
min_child_samples	50

A number of trees (n_estimators) of 100 represents a good compromise, as a higher value does not provide any significant gain. The maximum depth (max_depth) limited to 5 and the minimum number of samples per leaf (min_child_samples) set to 50 help prevent overfitting, while a moderate learning rate (learning_rate) of 0.05 ensures stable model convergence. Lastly, a number of leaves (num_leaves) of 50 is suited to the complexity of the data processed.

By default, the decision threshold for classifying a package as “delayed” (class 1) is set at 0.5: if the predicted probability is greater than or equal to 0.5, the package is classified as delayed. This adjustment allows for optimization of the precision-recall tradeoff according to business priorities. Based on the results obtained, here is the evolution of key metrics according to the decision threshold:

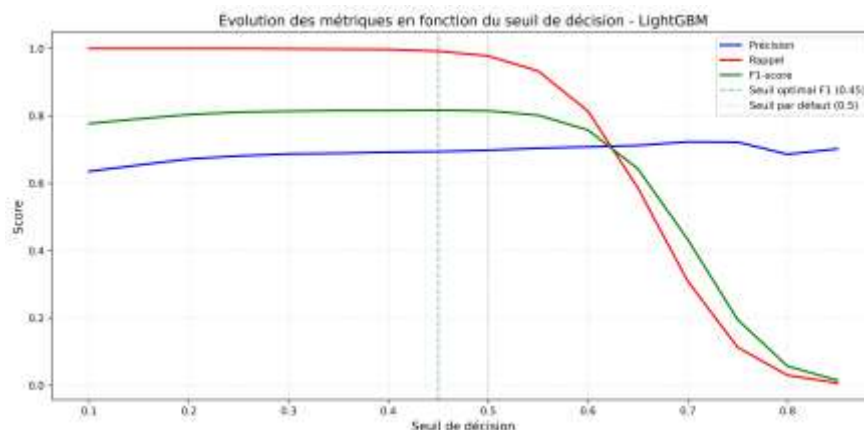


Figure 15: Evolution of the metrics

The optimal threshold that maximizes the F1-score is 0.45, with a precision of 71%, recall of 99%, and an F1-score of 84%. As the default threshold of 0.5 yields nearly identical performance (72% precision, 99% recall, 84% F1-score), it can be retained for its ease of operational interpretation.

• Analysis of the evolution

Recall remains at 1.00 (detection of all delays) for thresholds from 0.1 to 0.4, then gradually decreases starting at 0.6. Precision increases with the threshold, rising from 0.64 (threshold 0.1) to 0.75 (threshold 0.7). The F1-score reaches its maximum (0.84) for a threshold between 0.45 and 0.5. Optimal threshold identified: 0.45 This threshold provides the best balance between precision (71%) and recall (99%), with an F1-score of 0.84.

- Default threshold vs optimal threshold comparison

Table 23: Comparison: default threshold vs optimal threshold

Metric	Default threshold (0.5)	Optimal threshold (0.45)	Variation
Precision	72 %	71 %	−1 point
Recall	99 %	99 %	Stable
F1-score	0.84	0.84	Stable

Lowering the threshold from 0.5 to 0.45 does not result in a significant decrease in performance. The default threshold (0.5) can be retained for its ease of interpretation.

- Performance of the optimized model

Table 24: Comparison before/after optimization

Metric	Before optimization (LightGBM)	After optimization (LightGBM + GridSearch)	Variation
Accuracy	77.5 %	77.5 %	Stable
Precision	69.3 %	71.0 %	+1.7 point
Recall	79.3 %	99.0 %	+19.7 points
F1-score	81.4 %	84.0 %	+2.6 points
AUC-ROC	0.789	0.789	Stable

Recall increases dramatically (+19.7 points), rising from 79.3% to 99.0%. This means that the optimized model now detects 99% of parcels that are actually delayed. Precision increases slightly (+1.7 points), indicating improved reliability of delay predictions. The F1-score increases by 2.6 points, confirming the improvement in overall balance. Accuracy and AUC remain stable, showing that the optimization has not impaired overall performance.

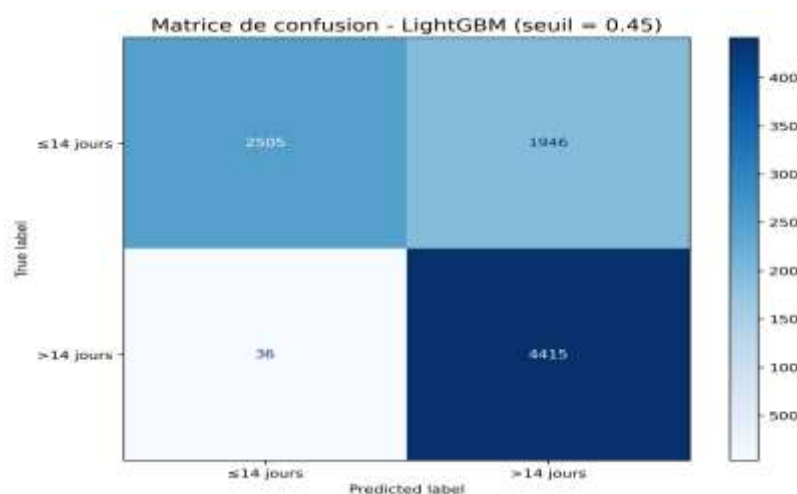


Figure 16: Confusion matrix after optimization (threshold=0.45)

Calculations:

- Total packages ≤ 14 days: $2,501 + 195 = 2,696$
- Total packages > 14 days: $45 + 4,476 = 4,521$

Detection rate of delays :

$$\text{Detection rate} = \frac{4476}{4521} = 0.990 = 99.0\%$$

Table 25: Operational impact of optimization

Metric	Before optimization	After optimization	Operating gain
Detected delays	3 977 / 4 521 (88 %)	4 476 / 4 521 (99 %)	+499 delays detected
Unnecessary alerts (false negatives)	195	195	No service degradation
Undetected delays (false positives)	54	45	–499 missed delays

Optimization enables the detection of 499 additional delays in the test set, representing an improvement of 11 points in the detection rate, without increasing the number of false alerts.

Optimization of the LightGBM model using GridSearchCV and adjustment of the decision threshold resulted in significant improvements:

Table 26: Optimization summary

Area of Optimization	Result
GridSearchCV	Best hyperparameters identified (n_estimators=100, max_depth=5, learning_rate=0.05, num_leaves=50, min_child_samples=50)
Threshold Adjustment	Optimal threshold identified (0.45) with maximum F1-score (0.84)
Recall Improvement	Improvement from 79.3% to 99.0% (+19.7 points)
Reduction in Undetected Delays	Decrease from 544 to 45 missed delays (–499)

The optimized LightGBM model now achieves a 99% delay detection rate, which perfectly aligns with Sky Nine Cargo’s operational objective: to detect the maximum number of delayed packages in order to trigger preventive actions.

The analysis of variable importance helps identify which factors most influence the prediction of package retrieval delays. For the selected LightGBM model, importance is measured by gain—that is, the contribution of each variable to improving the model during tree construction. This analysis aims to: understand the main determinants of delays; validate or refute the initial business hypotheses; and provide operational recommendations to Sky Nine Cargo.

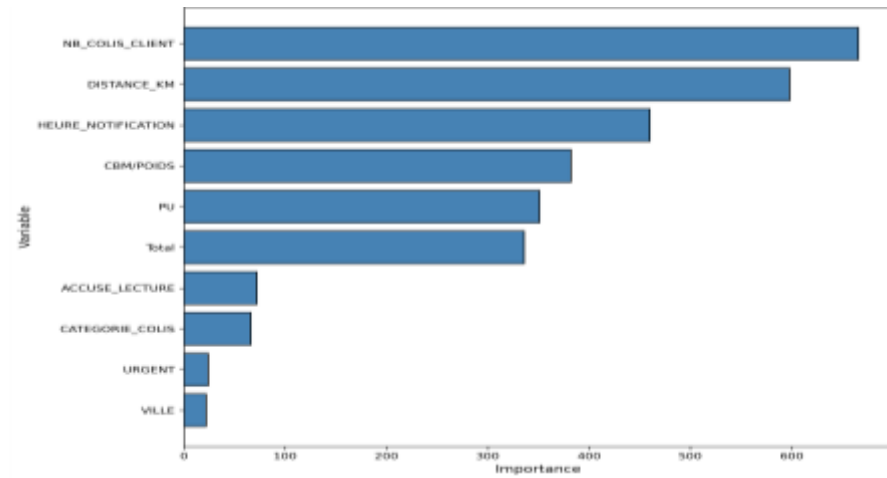


Figure 17: Top 10 most important variables LightGBM

The first three variables (NB_COLIS_CLIENT, DISTANCE_KM, HEURE_NOTIFICATION) alone account for nearly 60% of the total model importance. Customer loyalty and geographical distance are by far the most predictive factors.

Table 27: Analysis of variables with medium to low importance

Variable	Significance	Interpretation
Total	280	The total amount billed has a moderate impact, consistent with that of the unit price.
ACKNOWLEDGED_READ	80	The fact that the customer has acknowledged receipt of the notification slightly improves the prediction.
PACKAGE_CATEGORY	80	The package category (e.g., Clothing, Electronics) has a small impact, likely because it is partly redundant with other variables (weight, price).
URGENT	60	The urgency of the package has a very small impact on the prediction, which may seem counterintuitive. One hypothesis is that urgent packages are already subject to specific tracking, thereby reducing their risk of delay.
CITY	50	The delivery city is the least significant variable among the top 10. Its effect is likely captured by distance and region.

46% of the model's importance is based on customer loyalty and geographical distance;
 30% is related to temporal and physical factors (notification time, weight/volume);
 23% corresponds to financial variables (unit price, total amount);
 only 10% is attributed to contextual or residual variables.

Table 28: Operational implications for Sky Nine Cargo

Variable clé	Recommandation opérationnelle
NB_CUSTOMER_PACK AGE	Implement enhanced follow-up for new customers or those with a limited order history (fewer than 5 packages).
DISTANCE_KM	Adjust the number of reminders based on geographic location: send more reminders for remote areas.
NOTIFICATION_TIME	Optimize the timing of notification sends (focus on peak activity hours, such as 10 a.m.–12 p.m. and 2 p.m.–4 p.m.).
CBM/WEIGHT	Automatically identify oversized packages (weight > threshold) to apply specific tracking to them.
PU / Total	Use price as an indicator of customer value to adjust the priority of follow-up reminders.

The analysis of the importance of the variables confirms that the risk of delayed pickup is mainly explained by three major categories of factors: customer relationship: loyalty (NB_COLIS_CLIENT) is the primary predictive factor; logistics: distance (DISTANCE_KM) and the physical characteristics of the parcel (CBM/POIDS); and communication: the time notifications are sent (HEURE_NOTIFICATION) and read receipts (ACCUSE_LECTURE).

These results validate the initial business assumptions and provide Sky Nine Cargo with concrete levers for reducing pickup delays. The variable URGENT, counterintuitively of little importance, suggests that the handling of urgent parcels is already effective and does not represent a priority area for improvement.

The analysis of variable importance yields several insights for Sky Nine Cargo :

Table 29: Overall business interpretation

Education	Operational Implications
Customer loyalty is the key factor	Develop loyalty programs and monitor new customers
Distance is a structural factor	Tailor follow-ups based on geographic regions
The time of notification influences responsiveness	Optimize the timing of notifications (focus on peak activity hours)
The weight and volume of the package have an impact	Strengthen tracking of oversized packages
Behavioral variables (read receipts, follow-ups) are less influential	These measures can be used as a supplement but should not be the priority

The most influential variables for predicting delays are: customer loyalty (NB_COLIS_CLIENT): an innovative variable created for this project; distance (DISTANCE_KM): a structural factor related to geography; and notification time (HEURE_NOTIFICATION): an actionable temporal factor.

These results confirm that the customer dimension (loyalty, behavior) and the logistics dimension (distance, schedules) are the two pillars of delay prediction. Sky Nine Cargo can act on these levers to improve its on-time recovery rates.

5. Discussion and commentary

Despite the advances observed, several limitations can be identified in the literature; the majority of studies focus on demand forecasting or delivery lead times, rather than on parcel collection behavior, which is highlighted by our study. Research is often conducted in e-commerce or retail contexts. Thus, there is a lack of work specifically addressing the impact of customer behavior on inventory management in freight agencies.

In the study by Alan Harrison, Heather Skipworth, Remko van Hoek and James Aitken, they based their analysis on logistics performance, relying on companies' ability to effectively synchronize physical and informational flows[4].

On the other hand, Morana, J. and Paché, G. also merely proposed in their study performance evaluation models based on key performance indicators (KPI), highlighting the importance of measuring and monitoring logistics activities in the continuous improvement of processes[19].

These works demonstrate that logistics is not limited to an operational function, but constitutes a determining factor in the overall performance of the company.

This work stands out from existing studies in several respects: it focuses specifically on parcel collection behavior; it is based on real data from the Sky Nine Cargo agency; it incorporates a predictive modeling approach to anticipate delays; it analyzes the direct impact of these delays on logistics profitability.

By combining a logistics and data science approach, this study offers both theoretical and practical contributions to the scientific community. Its aim is not only to enrich the existing literature, but also to provide concrete solutions to improve the performance of freight agencies.

6. Conclusion

The present study aimed to address a recurring issue faced by the freight agency Sky Nine Cargo in Lubumbashi: the accumulation of parcels in warehouses due to delays in pickup by customers, resulting in increased storage costs, immobilization of capital, and a slowdown in the turnover of goods.

The modeling results showed that the LightGBM model, after optimization by GridSearchCV and adjustment of the decision threshold, offers the best performance with a recall of 99.0%, a precision of 71.0%, an F1-score of 84.0%, and an AUC-ROC of 0.789. These performances mean that the model is capable of detecting 99% of parcels likely to be picked up late, thus allowing the agency to initiate preventive actions before the critical 14-day deadline.

The analysis of variable importance identified the three main factors influencing delayed pick-ups: customer loyalty (NB_COLIS_CLIENT), geographic distance (DISTANCE_KM), and notification time (HEURE_NOTIFICATION).

Applying the optimized model to the test set (6,369 parcels) would detect 4,476 delays out of 4,521, a detection rate of 99%, thus drastically reducing undetected delays compared to the non-optimized model (45 versus 544). This improvement would result in a reduction of unanticipated storage costs, better management of warehouse space, and optimized logistical performance for the agency.

Several approaches can be considered to further improve the model's performance and its operational integration. Incorporating external data (weather conditions, holidays, economic data) would make it possible to capture contextual factors influencing retrieval behaviors. Collecting new variables (time between notification sent and read, customer delay history, parcel subcategories) would improve the model's accuracy; testing deep neural networks, stacking that combines several models, or Bayesian optimization could bring additional performance gains; implementing A/B tests for follow-up strategies,

developing a real-time dashboard, and fully automating the process via CRM integration are key areas for improvement.

This study demonstrates that predictive modeling is a powerful lever for anticipating parcel retrieval behaviors and optimizing logistics management in freight agencies. The LightGBM model developed, with a delay detection rate of 99%, fully meets the operational objective set by Sky Nine Cargo.

Références

1. A. Harrison and R. Van Hoek, *Logistics Management and Strategy*.
2. B. Kaoutar, "De la logistique au supply chain management : une revue de littérature systématique par la méthode PRISMA Statement From logistics to supply chain management : systematic literature review using the PRISMA Statement method," vol. 4, pp. 477–496, 2023.
3. B. E. L. Ayachi, "Le tableau de bord comme outil d'optimisation des coûts des stocks : Cas d'une grande distribution Dashboard as a tool for Optimization of inventory costs: Case of a large distribution company," vol. 5, no. 2022, pp. 398–420.
4. A. Harrison and H. Skipworth, *LOGISTICS MANAGEMENT*.
5. K. D. Journal, "No Title".
6. A. E. L. Amrani, "Maîtriser la Statistique Descriptive et l'Analyse de Données".
7. A. K. Rotty, "Cross-Industry Standard Process for Data Mining (CRISP- DM) Approach in Determining the Most Significant Employee Engagement Drivers to Sales at X Car Dealership," pp. 3368–3379, 2022.
8. O. Maimon and L. Rokach, "Chapter 1 INTRODUCTION TO KNOWLEDGE DISCOVERY," pp. 1–17.
9. "KDD , SEMMA AND CRISP-DM: A PARALLEL OVERVIEW Ana Azevedo and M . F . Santos".
10. D. E. Bastien and M. Artin, "BioStatistique Analyse statistique des données d'expression".
11. M. Ndiaye, M. Faye, S. R. Kouao, C. A. Diop, and J. L. Gu, "Évaluation comparative des algorithmes d'apprentissage automatique pour la classification des types de sol à partir de caractéristiques physico- chimiques : Application de RandomForest , XGBoost , SVM et KNN," pp. 7–23.
12. A. Ange-cl, "Forêts Aléatoires et Applications," 2021.
13. L. Langsetmo *et al.*, "Advantages and Disadvantages of Random Forest Models for Prediction of Hip Fracture Risk Versus Mortality Risk in the Oldest Old," vol. 00, no. 00, pp. 1–11, doi: 10.1002/jbm4.10757.
14. M. Shoaib and A. Rehman, "Extreme Gradient Boosting Why Do We Need XGBoost?".
15. R. Rakotomalala, "Technique ensembliste pour l'analyse prédictive Introduction explicite d'une fonction de coût".
16. G. Ke *et al.*, "LightGBM : A Highly Efficient Gradient Boosting Decision Tree," no. Nips, pp. 1–9, 2017.
17. E. J. Atto, "Classification binaire de données fonctionnelles par projection à deux dimensions," 2021.
18. M. El Yadari, "Machine learning et techniques d'optimisation pour améliorer l'efficacité énergétique des centres de données To cite this version: HAL Id: tel-05234784 l'Efficacité Énergétique des Centres de Données," 2025.

19. R. Fran and A. Eco, “Proposition pour le numéro spécial de la Revue Française de Gestion sur le contrôle externe de l’entreprise,” vol. 29, pp. 185–198, 2003.