NTNU | Norwegian University of Science and Technology

Machine Learning Project Report

TDT4173 - Modern Machine Learning in Practice

Group: 34 [Nan Squad]

Authors: Anass Chebbaki, Silvia Bortoluzzi, Alessandro Rimondi

Table of Content

1. Introduction

This project addresses a supply chain forecasting challenge for **Hydro ASA**: predicting cumulative raw material receipts for the first five months of 2025. The business context requires a careful balance between inventory optimization and risk mitigation, where over-prediction carries higher costs than under-prediction. The problem is formally framed as a quantile regression task with $\alpha = 0.2$, directly optimizing the Pinball Loss that penalizes overestimation more than underestimation. This setup aligns with Hydro's operational priorities, promoting conservative forecasts that slightly under-predict to minimize stock-out risks while maintaining reasonable inventory costs.

The analysis utilizes four primary datasets that provide complementary perspectives on Hydro's supply chain operations:

- **Receivals**: Contains historical records of actual material deliveries, including timestamps, quantities and material identifiers. This dataset serves as the foundation for understanding historical patterns and serves as the target variable for model training.
- **Purchase Orders**: Comprises planned procurement activities with scheduled delivery dates and quantities. This forward-looking information provides signals about future material flows and serves as a key predictive input.
- **Materials**: Provides relational linkages between different product identification systems, enabling consistent categorization across datasets and facilitating data integration.
- **Transportation**: Provides logistical details for material deliveries, including anonymized transporter names , vehicle identifiers and shipment status (such as "loaded or in-transit"). This data offers insights into the delivery process and specific carrier activities. We decided not to use this dataset.

2. Exploratory Data Analysis

The exploratory data analysis (EDA) represented a crucial step to understand the structure of the dataset and to guide the subsequent feature engineering choices in a motivated way. Through a set of descriptive analyses, temporal visualizations and comparisons across suppliers and materials, several patterns, anomalies and discontinuities were identified, all of which directly influenced the construction of the model's features.

2.1 Dataset Structure and Unit of Measure Anomalies

One of the first verification steps focused on checking the consistency of measurement units in the purchase order tables.
The analysis revealed that, while most records were expressed in kilograms (unit_id = 40), a small subset of orders (7) was recorded in pounds (unit_id = 43).
Although limited in number, this inconsistency could have introduced systematic errors in total quantity calculations and temporal aggregations if not corrected.

To ensure uniformity across the dataset, all quantities expressed in pounds were converted to kilograms using the standard conversion factor (1 lb = 0.453592 kg).
The corresponding records were then updated with unit_id = 40 and unit = 'kg', aligning them with the unit of measure used in the rest of the dataset.

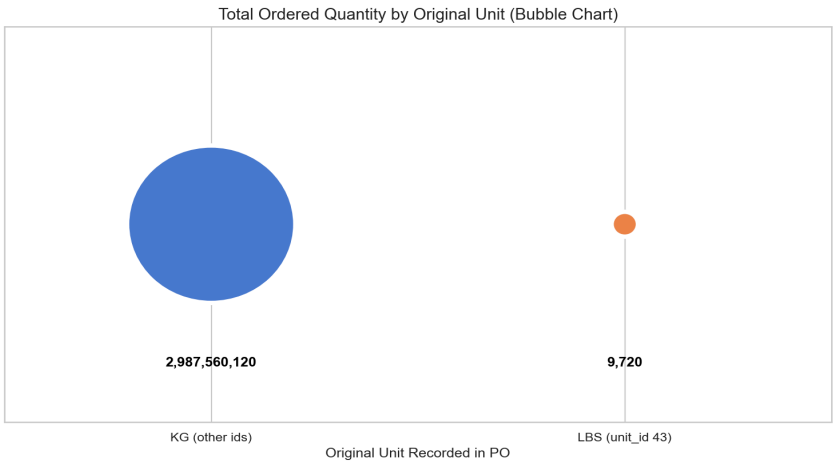


Figure 2.1.1 – Distribution of measurement units (kg vs lbs) in purchase orders

2.2 Macroscopic Analysis: Trends and Structural Anomalies

The analysis of the temporal evolution of aggregated receivals (receivals.csv) was fundamental in defining our data sampling strategy.
A plot of the total received weight per year (figure 2.2.1) reveals a generally stable growth trend from 2004 to 2019. Before plotting the data, we hypothesized that this pattern experienced a sharp disruption starting in 2020, coinciding with the onset of the COVID period.

However, the plot revealed that the anomaly was not a simple "collapse." Instead, it represented a period of extreme volatility: 2020 showed a temporary stagnation with a slight decline in volumes, which was immediately followed by an all-time peak in receivals in 2021, and then by a new contraction in 2022.

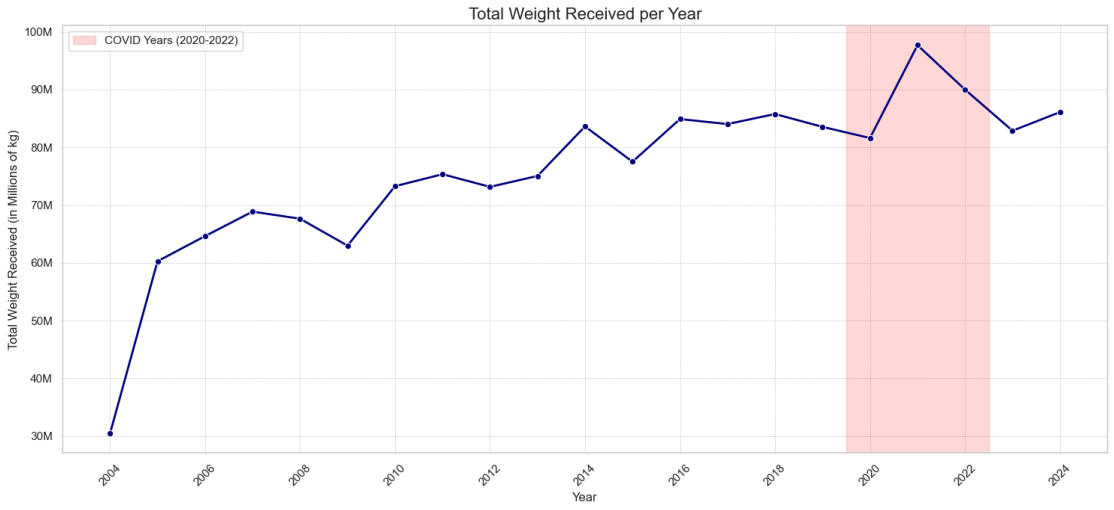


Figure 2.2.1 – Total received weight per year (2004–2024)

The COVID-19 period (2020–2022) therefore represents a clear discontinuity in the long-term trend of the supply system, characterized by strong volatility and atypical behavior. From 2023 onward, the data suggest a return to regular operational levels, indicating a re-stabilization of the procurement process after the disruptions of the pandemic years.

2.3 Pareto Analysis of Materials, Suppliers and Periods

The Pareto analysis represents a crucial preliminary step to understand the distribution of supply volumes and to identify the materials, suppliers and time periods that contribute most significantly to the total received weight.

In the context of inventory management, this analysis allows us to focus attention on the small set of critical elements that explain the majority of overall variability, consistent with the 80/20 principle typical of ABC analysis.
A small number of materials or suppliers account for most of the handled weight and, consequently, for most of the operational and economic impact of the supply chain.
Applying Pareto analysis to the received material volumes (net_weight) revealed that only 16 materials (out of 203 total) account for approximately 80% of the total received volume. Among these, the most relevant are rm_id = 2130, 1903, 2160, 2140 and 1909, which together represent over half of the total handled weight.
This concentration highlights how the dynamics of a few key materials dominate the entire supply system.

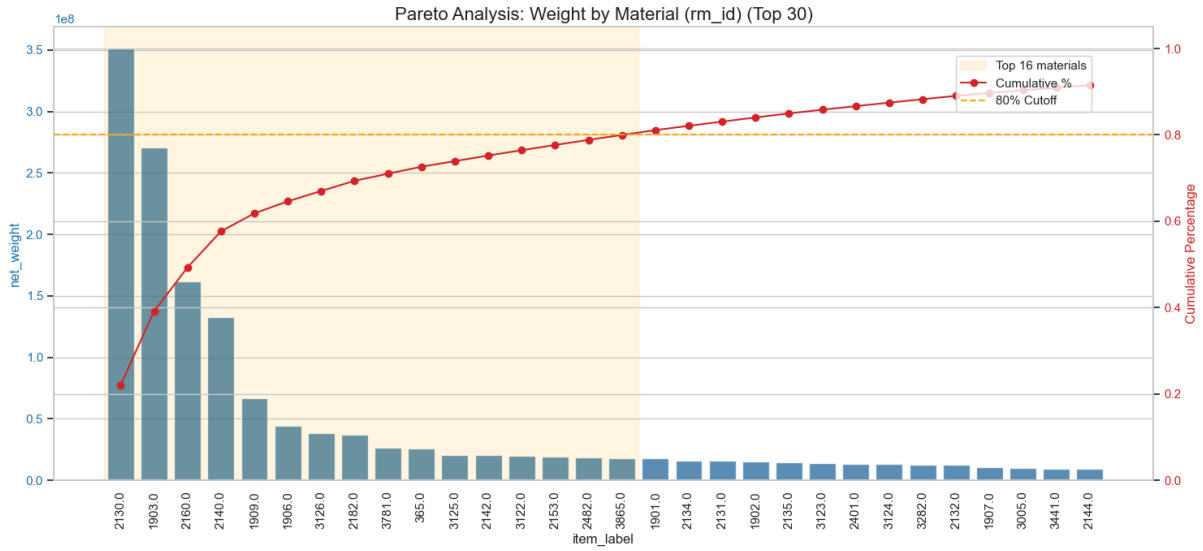


Figure 2.3.1 – Pareto Analysis: Weight by Material

The analysis was further extended to examine the distribution of weight by month and by supplier, with two specific objectives:

- **Weight by Month**: to assess the seasonality of material flows and to identify the periods of the year with higher supply intensity. This is particularly relevant in our case, since the predictive model's final goal is to estimate the cumulative received weight over the first five months of the year. The plot confirms that most materials tend to be received in the months preceding major holiday or shutdown periods, ensuring adequate stock levels before production slowdowns. Conversely, January, August, and December, which coincide with holiday breaks and reduced industrial activity, show the lowest inbound volumes, reflecting the typical seasonal rhythm of the supply chain.

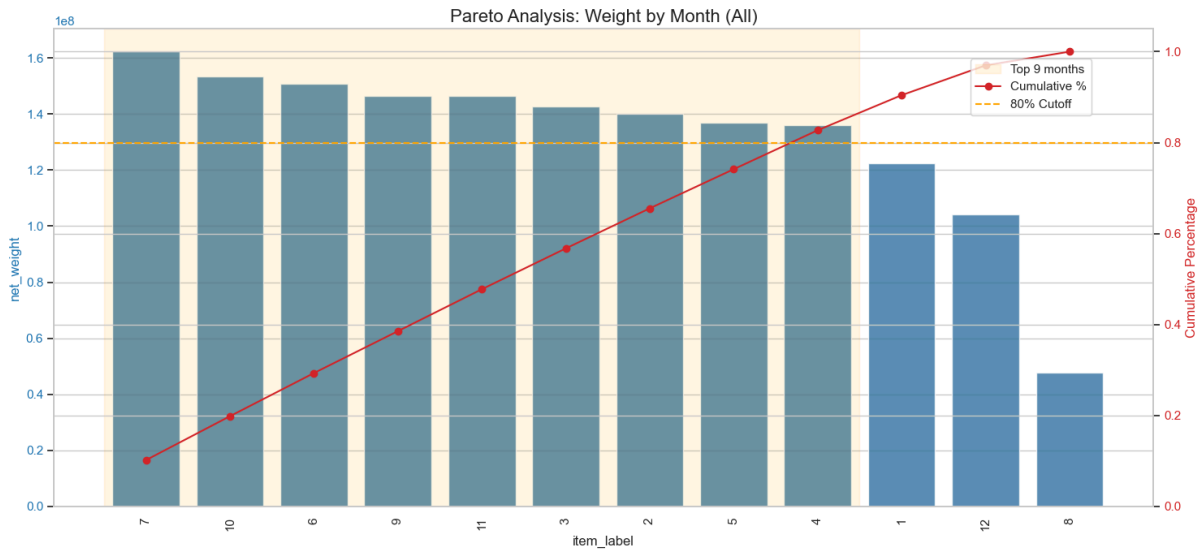


Figure 2.3.2 – Pareto Analysis: Weight by Month

- **Weight by Supplier:** to understand the distribution of volumes across suppliers and their percentage contribution to the total. The resulting plot reveals a clear *long-tail effect*: a small group of suppliers accounts for half of the total received weight, while a large number of smaller suppliers contribute only marginally. Specifically, it takes more than the top 30 suppliers to reach 80% of the cumulative volume, indicating a highly unbalanced distribution. This pattern suggests that the supply network is diversified but dominated by a few key

partners that ensure the main material flow, while many secondary suppliers provide niche or low-frequency materials.

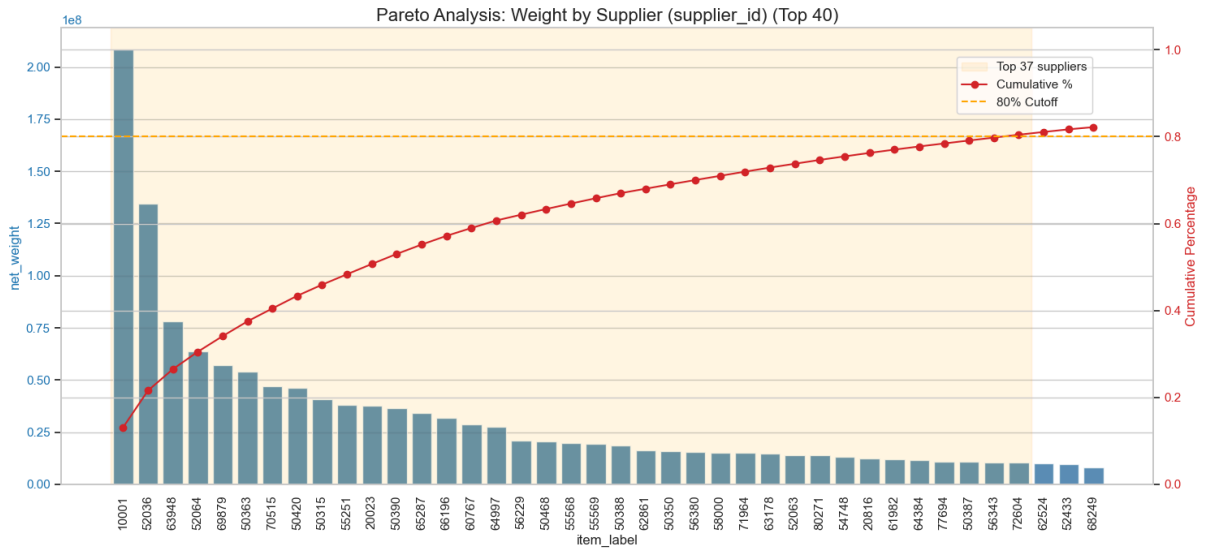


Figure 2.3.3 – Pareto Analysis: Weight by Supplier (Top 40)

2.4 Temporal Analysis

The temporal analysis, together with the Pareto analysis, was one of the most important components in identifying effective predictive features.
Its goal was to investigate the dynamic structure of the receivals data and to assess the presence of seasonal patterns, long-term trends and temporal dependencies.

To better understand the relevance of temporal variables in modeling, the correlation between the main historical metrics and the target variable was computed.
Figure 2.4.1 shows the linear correlation between the main historical metrics and the target value (i.e., the total received weight within the forecast window).

The objective of this analysis was to quantitatively evaluate how informative past temporal windows (7, 30 and 90 days, or the same period from the previous year) are for predicting future receival behavior.

The chart highlights that the variables with the strongest positive correlation with the target are:

- Total Receivals (Same Window, Last Year)
- Total Receivals (Last 90 Days)
- Total Receivals (Last 30 Days)
- Mean Receivals (Last 30 Days)

These results confirm that recent historical patterns over short and medium horizons (1–3 months) have strong predictive power; in other words, materials that showed high and regular volumes in recent weeks are highly likely to maintain similar behavior in the following periods.

The strong correlation with the same time window from the previous year further indicates the presence of a significant seasonal component.
Additionally, features related to recent variability in receivals (e.g., Standard Deviation of Receivals over the Last 30 or 90 Days) show a moderately positive correlation, suggesting that materials subject to recent fluctuations often continue to exhibit unstable behavior, an insight useful for modeling target variance.

On the other hand, the Sparsity feature (Zero Days, Last 30d), which measures the frequency of days without receivals, shows a negative correlation, consistent with the notion that a higher number of “empty” days corresponds to lower overall received volumes.

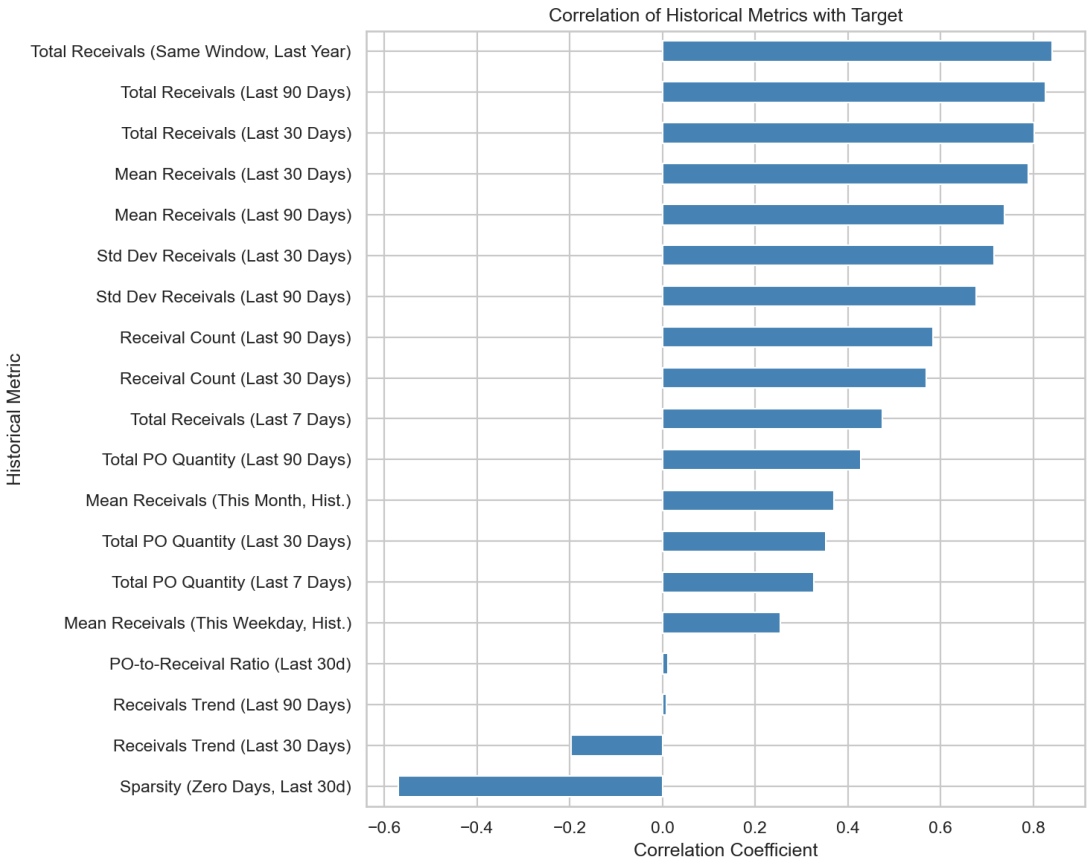


Figure 2.4.1 – Correlation between Historical Metrics and target variable

In summary, the analysis confirms that receivals exhibit a complex temporal structure, characterized by both seasonality and dependence on recent historical values.

2.5 Daily and Weekly Patterns

In addition to long-term dynamics, an analysis of holidays, intra-month and intra-week receival patterns was conducted to identify operational regularities linked to production and logistics cycles.

Figure 2.5.1 compares the average total weight of receivals on regular working days and on national holidays (according to the Norwegian calendar), clearly highlighting the impact of holidays on inbound logistics.

As shown in the chart, the average received volume on holidays is roughly half of that on

regular working days.

This difference reflects the operational slowdown in transportation and unloading activities that typically occurs during holiday periods.

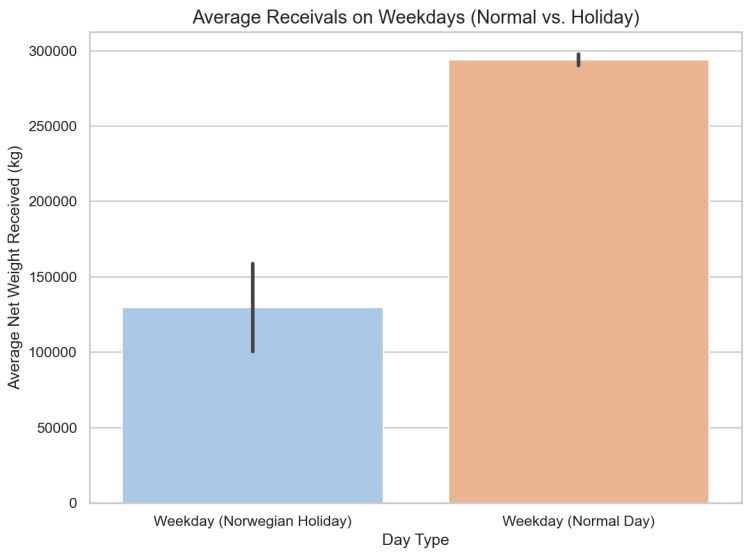


Figure 2.5.1 – Average receivals on working days vs national holidays

Figure 2.5.2 shows the trend of total received weight as a function of the day of the month. A significant increase in volumes can be observed during the first few days of the month (days 1–3), followed by a more stable central phase and a sharp decline toward the end of the month (days 28–31). This behavior is consistent with the monthly closing of procurement cycles and the reduction of logistics activity near month-end.

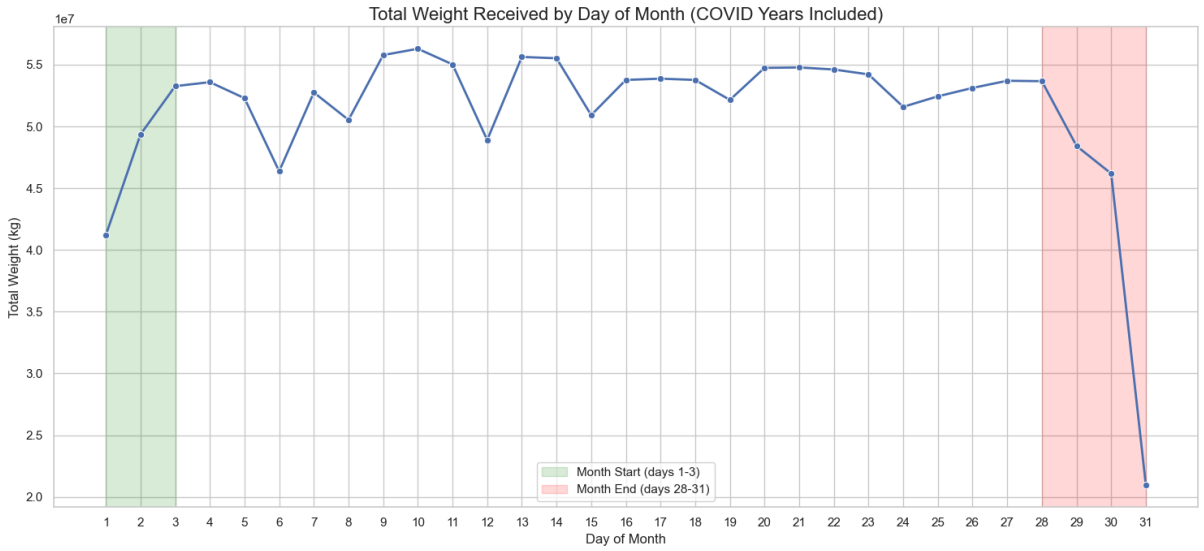


Figure 2.5.2 – Total Received Weight by Day of the Month

Figure 2.5.3 instead analyzes the distribution of total received weight by day of the week. The results show that receivals are concentrated on weekdays, with a clear peak between Tuesday and Wednesday and a drastic reduction over the weekend. Mondays and Fridays exhibit slightly lower values, indicating reduced logistical activity near the weekend. This regular weekly pattern is typical of industrial supply chains and highlights the importance of including day-of-week features and weekend indicators for improved predictive accuracy.

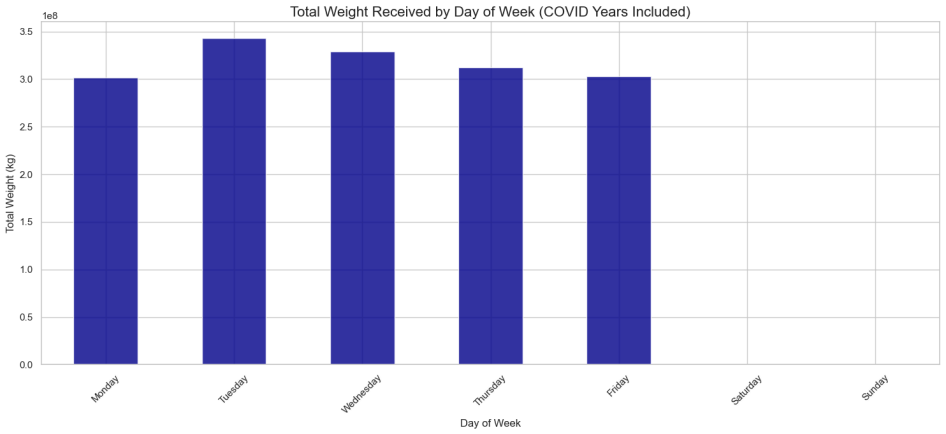


Figure 2.5.3 – Total Received Weight by Day of the Week

2.6 Supplier Concentration and Reliability

Then we moved to the analysis of supply concentration, represented through the supplier share heatmap, which revealed a strong heterogeneity in the sourcing structure.

Some materials, such as rm_id = 1909 and rm_id = 2140, appear to be almost entirely dependent on a single supplier (100% share), while others such as 2130, 1903 and 2160, show a more balanced distribution of volume across multiple suppliers.

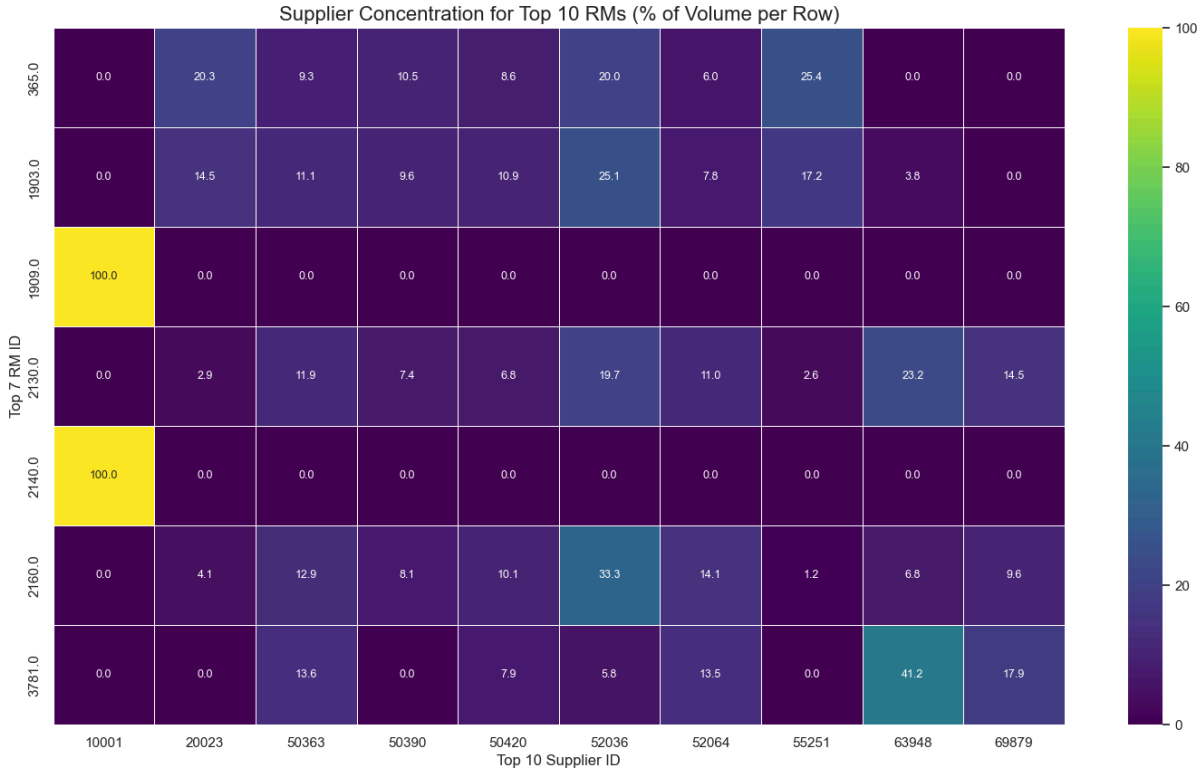


Figure 2.6.1 – Supplier share heatmap by material

Based on this observation, a quantitative analysis of delivery punctuality and variability was conducted on the materials which together account for approximately 80% of the total received volume.

For each of these key materials, the mean, median, standard deviation, and outlier count of the actual lead time (difference between arrival date and planned date) were calculated.

The results showed that the supply system, overall, tends to receive deliveries earlier than the scheduled date, although with a non-negligible degree of variability. Specifically:

- The only material with a positive median delay is rm_id = 2182, with a median of +4 days and an average delay of +10 days, representing the only case of systematic lateness.
- The two materials strongly dependent on a single supplier (2140 and 1909) show more stable lead times and median delays close to zero, suggesting lower flexibility but greater short-term predictability.
- The remaining materials exhibit negative median delays, indicating that deliveries generally occur ahead of the scheduled date, with median delays between –10 and –30 days, values consistent with operational practices that favor a safety margin in delivery timing.

The analysis of lead time data reveals two distinct types of unpredictability. Materials 1906.0 (IQR 23.0), 2482.0 (IQR 20.0), and 3126.0 (IQR 20.0) demonstrate the most consistently unpredictable performance, having the highest Interquartile Range (IQR); this indicates their core 50% of deliveries are the most widely spread, making their standard lead time inherently unreliable. A separate issue is seen with materials prone to extreme, one-off events. Specifically, 2130.0 (212 outliers), 1903.0 (164 outliers), and 2140.0 (111 outliers) experience the highest number of outlier deliveries. This means that while their typical delivery window may seem narrow (e.g., 2140.0), they are highly susceptible to significant disruptions.

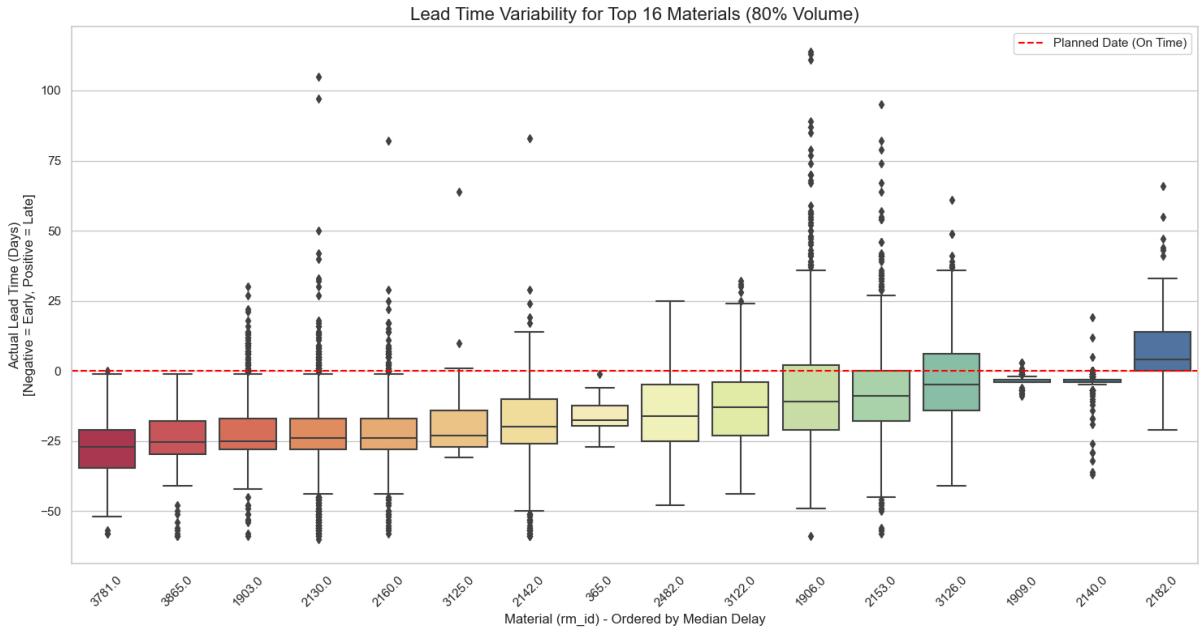


Figure 2.6.2 – Distribution of actual lead times by material

In summary, the quantitative analysis confirms that most materials are delivered ahead of schedule on average, with variability and the presence of outliers being highest among the most impactful materials. Furthermore, dependence on single suppliers tends to reduce the average delivery advance while simultaneously making deliveries more regular and predictable.

2.7 Dynamic analysis of materials over time

Since the cumulative analysis alone is not sufficient to describe the temporal evolution of each material's contribution, a dynamic analysis of the monthly trends of the main materials over time was carried out, as shown in Figures 2.7.1 and 2.7.2.

Figure 2.7.1, focused on recent years (2016–2024), confirms that materials 2130, 2160 and 2140 have been the major contributors to total volumes in recent periods. In particular, material 2130 clearly dominates the series, showing regular peaks and annual oscillations that suggest a strongly seasonal and stable pattern over time.

A marked decline can be observed between 2023 and 2024, during which material 3781 temporarily takes a leading role, before 2130 regains dominance in the second half of 2024. This behavior may indicate a temporary supply substitution or a production adjustment. On the other hand, material 2140 shows higher variability and an overall

downward trend toward 2024, suggesting a possible structural reduction in its demand or a progressive change in the composition of raw materials used.

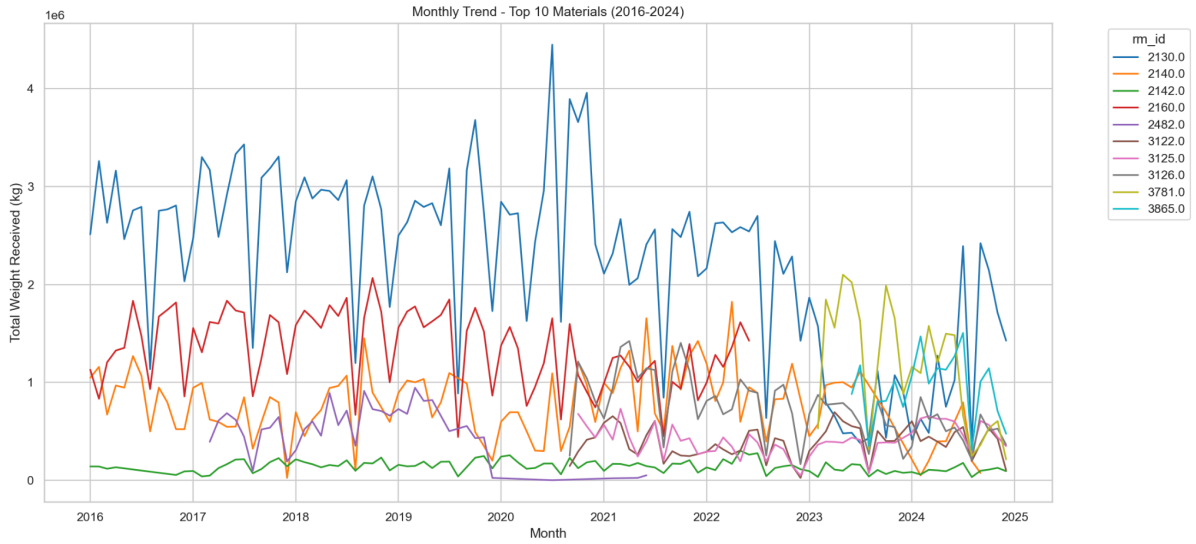


Figure 2.7.1 – Monthly weight trends of top materials (2016–2024)

Figure 2.7.2, which extends the analysis to the entire historical period (2004–2024), provides a broader long-term perspective. New materials such as 3126, 3865 and 3781 clearly emerge, appearing only after 2020 and highlighting the evolution of the material portfolio over time.

A particularly notable case is material 365, which was active only in the initial period (2004–2006) and subsequently disappeared, indicating a complete phase-out or replacement by a functionally equivalent material. Similarly, material 1903, which up to 2011 represented the main contributor to overall volumes, ceased to appear in receivals starting from 2012, marking a rapid exit from the supply cycle. An intermediate case is material 2160, introduced at the end of 2011 and discontinued in 2022, likely due to a gradual substitution or changes in production processes.

These instances of complete or gradual disappearance of historical materials reinforce the evidence that the dataset captures real-world dynamics of logistical and industrial obsolescence. In this context, it becomes particularly relevant to distinguish between

active and discontinued materials, thus preventing forecast overestimations and improving the model's ability to adapt to the real evolution of the supply chain.

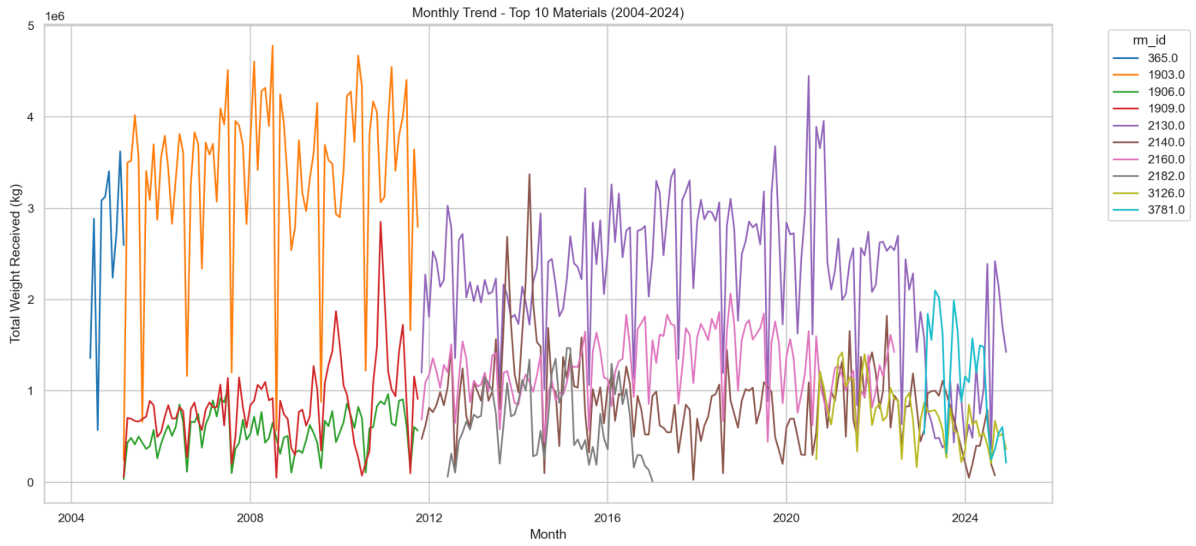


Figure 2.7.2 – Monthly weight trends of top materials (2004–2024)

3. Models

This chapter details the machine learning models developed to address the forecasting task. We intentionally did not experiment with deep learning architectures, such as Convolutional Neural Networks (CNNs), as it was unclear if they would be accepted as standard machine learning models for this project.
Our initial approach involved an exploratory phase where we benchmarked several different models to identify the most promising candidates. The models tested include AutoGluon, Prophet, CatBoost, LightGBM and XGBoost.
From this preliminary analysis, **Prophet** and **XGBoost** consistently yielded the best initial results. Therefore, we concentrated our efforts on refining and tuning these two models, which will be explained in detail in the following sections.

3.1 Setup and Execution Constraints

3.1.1 Hardware Configuration Requirements

All experiments and final submissions were executed on a strictly controlled hardware environment with the following specifications:

- **Model**: CPU
- **CPU Cores**: 4
- **Clock Speed**: 2.25 GHz
- **Memory**: 16 GB RAM

3.1.2 Reproducibility Constraint

The scores and predictions submitted for evaluation were obtained exclusively using the CPU-only configuration specified above. During development, we observed that executing the same notebook on different hardware configurations, particularly when using GPU acceleration or alternative CPU specifications, produced non-identical results and performance metrics. This variation stems from the stochastic nature of

optimization algorithms, which exhibit sensitivity to numerical precision and parallel execution strategies that differ between hardware architectures.

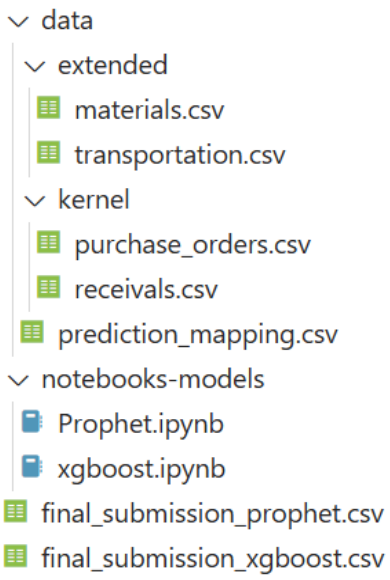
3.1.3 Software Dependencies

Beyond hardware constraints, exact reproducibility requires matching the software environment. Below we list all the requirements needed:

- numpy
- pandas
- prophet
- matplotlib
- seaborn
- xgboost
- optuna

3.1.4 Required Directory Structure

The notebook assumes a specific project directory structure for data loading and output generation. All scripts must be executed from within the designated notebook directory, with data files organized as shown below:



3.2 Prophet Approach: Implementation and Interpretation

For this forecasting task, we decided to submit a notebook that uses Facebook Prophet (*Short_notebook_1.ipynb*), a time series forecasting framework specifically designed to handle the complexities inherent in real-world business data. Prophet was selected over alternatives like ARIMA or LSTM networks due to its robust handling of several critical characteristics present in supply chain logistics data: strong seasonal patterns, irregular holiday effects, abrupt trend changes and sparse or missing observations. Unlike classical statistical models that struggle with irregularity, Prophet's additive decomposition framework (trend + seasonality + holidays + error) provides explicit, interpretable components that align well with the logistics processes we aimed to model.

Below we will describe different decisions we took along the way.

3.2.1 Target Variable

We decided to not implement a target variable from daily receival weights (daily_weight), but instead using cumulative weights (cum_weight). The raw daily arrivals exhibit extreme sparsity:most days record zero arrivals for any given raw material, interspersed with occasional large shipments. This creates a zero-inflated, highly intermittent signal that is difficult for trend-based models to learn from. By computing the cumulative sum of weights over time, we reframed the problem from forecasting sporadic events to modeling a smooth, monotonically increasing growth curve. This transformation serves two purposes: it converts noise into signal by accumulating information over time and it aligns perfectly with Prophet's strength in modeling growth trajectories. The model forecasts the cumulative curve's absolute growth trajectory. This forecast is then post-processed to normalize it, showing the total weight accumulated relative to the start of the forecast period.

3.2.2 Per- Rm_Id Modeling Strategy

Rather than building a single global model across all materials, we implemented a "per-item" approach by training independent Prophet models for each unique rm_id (raw material identifier). This architectural choice is justified by the heterogeneity of demand patterns across materials that we discovered in the EDA part: each item has distinct procurement cycles, supplier lead times, seasonal demand drivers and lifecycle stages. Each material's unique temporal signature is captured without interference from unrelated items. While this approach increases computational cost (29 models in the final implementation), it significantly improves forecast accuracy by allowing each model to specialize on its specific item's behavior rather than averaging across diverse patterns.

3.2.3 Preprocessing and Training Dataset Construction

Three filters were applied to better define our task. First, we restricted training to materials with at least one receival after October 1, 2024 (considered “active”). This was done to address a problem identified during our exploratory data analysis (EDA): many historical rm_id entries represent discontinued products. Specifically, after sorting all materials by their most recent arrival date, we observed that only about 60 rm_id’s had any receivals during 2024. Forecasting the thousands of inactive items would waste computational resources and introduce significant noise, so this "active" filter was critical.

Second, we excluded materials with insufficient historical data, requiring at least 10 recorded arrivals to train a model. This threshold ensures statistical sufficiency and that Prophet has enough observations to estimate seasonality, trend changepoints and uncertainty intervals reliably. Items below this threshold would produce unreliable, high-variance forecasts so they are not taken into consideration.

Additionally, for each material, we restricted training data to start only after its first non-zero arrival. This prevents the model from learning patterns from a long "dead period" before a material becomes active.

Historical data was taken from January 1, 2009, providing up to 16 years of context. The earlier years (2004-2008) were found in the EDA part as following a different pattern. Also, we followed the belief that recent years are more informative than older years. This long historical window enables Prophet to capture long-term trends and multi-year seasonal cycles. However, the raw data contains gaps: not every material has arrivals on every day. To create a continuous time series required by Prophet, we constructed a complete date grid, then reindexed the data to fill missing days with zero weights. This ensures Prophet receives a regular, uninterrupted daily time series from 2009 to January 1, 2025, with explicit zeros representing days without arrivals rather than missing data.

3.2.4 Feature Engineering

Weekly Seasonality: Enabled by default, this captures the strong day-of-week patterns in logistics operations, shipments typically arrive on weekdays, with weekends showing significantly lower activity. This seasonality is estimated via Fourier series terms with automatic regularization.

Holiday Effects: We incorporated Norwegian national holidays as a custom holidays parameter. As seen on EDA, logistics operations are heavily impacted by holidays, when ports, warehouses and transportation networks operate at reduced capacity or close entirely.

Working Days Regressor: Beyond built-in seasonality, we added a custom binary regressor that flags weekdays (Monday–Friday) as 1 and weekends as 0. While this overlaps with weekly seasonality, the explicit regressor provides a more direct, interpretable coefficient for the "weekday effect," reinforcing the model's ability to distinguish operational days.

Additive Seasonality Mode: We used an additive seasonality mode,meaning seasonal effects are added to the trend rather than multiplied. Seasonal fluctuations do not scale proportionally with the trend magnitude so it is a reasonable assumption for cumulative weight, where weekly patterns remain relatively constant even as the cumulative total grows.

3.2.5 Hyperparameter Tuning

Two critical hyperparameters were tuned:

Changepoint Prior Scale: This controls the flexibility of the piecewise linear trend. Prophet automatically detects potential "changepoints" where the trend rate might shift (e.g., a supplier change, demand spike). A lower prior scale applies stronger regularization, making the model more conservative about fitting these changepoints. The chosen value of 0.008 (well below the default of 0.05) reflects a deliberate decision to prioritize smooth, stable long-term trends over fitting short-term volatility. This prevents overfitting to transient fluctuations in the historical data, which is critical for producing reliable forecasts in a real-world logistics context where short-term noise should not dictate long-term projections and critical for the submission requirements too.

Seasonality Prior Scale: This parameter controls the strength of seasonal patterns, where a higher value allows more flexibility in fitting seasonal effects. We retained the default (10), enabling the model to capture strong weekly patterns without over-smoothing.

An iterative hyperparameter search (with first rankings documented in the table below) tested different configurations, varying changepoint_prior_scale, start_date, activity filter, and shrinkage factors.
Configuration #2 (changepoint_prior_scale=0.008, start_date=2009, shrink factors 0.6/0.95, October 2024 as activity filter) was selected as the final model. Although configuration #1 achieved marginally better local performance, #2 was deemed more robust: its lower changepoint prior and broader shrinkage range produce more conservative, generalizable forecasts, reducing the risk of overfitting to the specific validation data.

Rank (Best to Worst)	Changepoint Prior Scale	Start Date	Shrink Factors (Min/Max)	Activity Filter (Since)
1	0.01	2010	0.6 / 1.0	October 2024
2	0.008	2009	0.6 / 0.95	October 2024
3	0.008	2016	0.6 / 0.95	October 2024
4	0.01	2009	0.6 / 0.95	October 2024
5	0.01	2009	0.6 / 1.0	October 2024
6	0.01	2014	0.1 / 0.6	October 2024
7	0.001	2010	0.6 / 0.9	October 2024
8	0.008	2009	0.6 / 0.95	September 2024
9	0.008	2009	0.6 / 0.95	January 2024

3.2.6 Forecast Generation

For each trained model, we generated forecasts for the next 150 days (as the task requires). This forecast horizon must include the same features as training data, so we extended the working_days regressor into the future. Additionally, we forward-filled the quantity_year feature (annual order quantity for that material) with the last observed value, assuming the most recent annual demand level will persist. Prophet then produces a forecast dataframe containing yhat (point forecast), yhat_lower and yhat_upper (uncertainty bounds).

3.2.7 Post-Processing

Prophet's raw forecasts for cumulative weight (yhat) occasionally exhibit small decreases due to model uncertainty or seasonal adjustments. Since cumulative sums cannot decrease by definition, we enforced a monotonic constraint to ensure that each predicted value is at least as large as the previous one, correcting any non-physical decreases while preserving the overall trend shape.

To convert the cumulative forecast (yhat_corr) into a value relative to the forecast period, we normalize the series. For each 150-day forecast block, we set a base value equal to the first predicted value in the block (preds[start]). We then subtract this single base value from all 150 predictions in that block. This yields the predicted_weight: a new cumulative series that represents the total weight accumulated since the start of the forecast period (i.e., it starts at 0 on the first day).

Another post-processing adjustment was applied to incorporate domain knowledge about demand uncertainty that we discovered through EDA. High-volume materials (so, those with large quantity_year values) tend to have more stable, predictable receival patterns compared to low-volume materials that exhibit higher variability and uncertainty. To encode this, we computed a "shrink factor" for each prediction based on the material's annual order quantity. First, we normalized quantity_year, then we linearly interpolated a shrinkage multiplier using the custom variables min_factor and max_factor, where low-volume items receive the minimum factor and high-volume items receive the maximum.
This was done to reduce overconfidence in predictions for uncertain, low-volume items while preserving the model's confidence in high-volume materials. The specific range

(60%–95%) was determined empirically through validation experiments, balancing conservatism with forecast utility.

3.2.8 Why Prophet?

The final model produces item-specific, day-level forecasts of raw material arrivals 150 days into the future. Each forecast reflects learned patterns of trend (long-term procurement growth or decline), weekly seasonality, Norway holiday impacts and working-day effects. The shrinkage adjustment introduces a layer of risk management.

This model was chosen because it is a robust forecasting methodology designed to overcome the core challenge of sparse, zero-inflated logistical data. The starting insight was to shift the problem from forecasting intermittent daily arrivals to modeling the absolute cumulative growth curve using a per-material Prophet implementation.

3.3 XGBoost Approach: Implementation and Interpretation

An XGBoost (Extreme Gradient Boosting) model was implemented to assess its capabilities for this task (*Short_notebook_2.ipynb*), driven by its well-documented success in similar domains. The resulting performance was promising; however, it did not surpass the predictive accuracy previously established by the Prophet model. The primary motivations for testing the algorithm were:

1. **Superior Performance on Tabular Data**: XGBoost consistently demonstrates state-of-the-art performance on structured, tabular datasets, making it ideal for our feature-rich dataset.
2. **Native Handling of Missing Values**: The algorithm efficiently handles missing data through its built-in split-finding mechanism, eliminating the need for complex imputation strategies.
3. **Robustness to Overfitting**: When properly tuned, XGBoost's regularization parameters and tree-pruning mechanisms provide strong safeguards against overfitting.
4. **Computational Efficiency**: The hist tree method with QuantileDMatrix enables efficient training on large datasets with millions of samples.
5. **Model Interpretability**: Feature importance metrics (gain, cover, weight) allow for transparent understanding of predictive drivers.

While the choice of XGBoost provides a robust algorithmic foundation, the true success of this model resides in a comprehensive and theoretically motivated feature engineering process. The model's predictive power emerges not merely from algorithmic sophistication, but from carefully crafted features that capture the underlying dynamics of the supply chain system: temporal patterns, supplier reliability, demand intermittency and logistics constraints.

3.3.1 Data Preparation and Training Set Construction

The raw data comprises two primary sources: receivals.csv and purchase_orders.csv. The function aggregates_3df_daily transforms these transaction-level datasets into daily aggregates grouped by rm_id. This aggregation serves two purposes:

1. **Dimensionality Reduction**: Reduces millions of individual transactions to a manageable daily panel structure.
2. **Alignment of Time Granularity**: Ensures both receivals and purchase orders are represented at the same temporal resolution.

During data preparation, key cleaning steps were taken to ensure data integrity. First, unit standardization was applied to all purchase orders, converting any entries recorded in pounds (unit_id = 43) to kilograms using the conversion factor (0.453592). Following this, to prevent noise in the model, any records with zero or negative weights or quantities were filtered out of the dataset.

The training_data function implements an approach to generate training samples. Rather than treating the problem as a simple regression with fixed windows, the model learns across all possible forecast horizons:

```
for rm_id in all_rm_ids:
    for year in year_for_training:
        start_date = date(year, 1, 1)
        end_max = date(year, 5, 31)
        max_horizon = (end_max - start_date).days

        for h in range(0, max_horizon + 1):
            end_date = start_date + timedelta(days=h)
            windows_for_train.append({
                'rm_id': rm_id,
                'forecast_start_date': start_date,
                'forecast_end_date': end_date
            })
```

The model's design is centered on horizon-agnostic learning, which is achieved by training it on windows that vary from 1 day up to 151 days. This method allows the model to learn how cumulative patterns evolve over different time scales, enabling accurate predictions regardless of the specific forecast horizon requested. To implement this, a fixed start date (January 1st) is used for all windows within a given year. This choice ensures consistency in how historical features are computed. Furthermore, the design provides comprehensive temporal coverage by using every possible day within the training period as an endpoint. This strategy creates a rich dataset that captures the continuous evolution of cumulative weights, ultimately generating approximately N_rm_ids * N_years * 151 days training samples and resulting in a dataset with several million observations.

One of the most impactful decisions in the modeling pipeline was the explicit exclusion of 2020, 2021 and 2022 from the training data. As documented in the Exploratory Data Analysis (*Chapter 2*), the supply chain exhibited severe disruptions during the COVID-19 pandemic. These events introduced significant volatility and anomalous data patterns, which were deemed unrepresentative of standard operational conditions.

3.3.2 The Core of the Model: Feature Engineering

The predictive performance of this model is fundamentally driven by a meticulously designed feature space comprising over 30 engineered variables. These features are organized into several conceptual categories, each addressing specific aspects of the forecasting challenge.

Calendar-Based and Window Features

- **window_length**: The number of days between forecast_start_date and forecast_end_date (inclusive). Longer windows naturally accumulate more weight.
- **business_days_in_window**: Computed using NumPy's busday_count, this captures the number of working days, as deliveries typically occur on weekdays.
- **weekends_in_window:** Derived as window_length - business_days_in_window, accounting for periods with reduced logistics activity.
- **end_dayofweek:** The day of the week (0=Monday, 6=Sunday) of the forecast end date. Deliveries exhibit weekly patterns, with fewer arrivals on Mondays and Fridays.
- **end_days_to_month_end:** Days remaining until the end of the month, capturing end-of-month rush behaviors in procurement.
- **month_sin** and **month_cos:** Cyclical encoding of the month using sine and cosine transformations. This preserves the circular nature of calendar months while avoiding ordinality assumptions.

Purchase Order Features:

- **po_in_window**: The total quantity (in kg) of purchase orders scheduled for delivery within the forecast window. Computed using vectorized cumulative sum logic with binary search (np.searchsorted) for computational efficiency.
- **daily_avg_po_in_window**: Normalizes po_in_window by window_length, providing a rate measure that is comparable across different horizon lengths.

Historical Aggregate Features

- **hist_rec_7d, hist_rec_30d, hist_rec_90d**: Total weight received in the 7, 30 and 90 days immediately preceding forecast_start_date.
- **hist_po_7d, hist_po_30d, hist_po_90d**: Analogous aggregates for purchase orders.

By including multiple horizons, the model can adaptively weight the relevant timescale for each rm_id.

Rolling Statistical Features

- **rec_mean_30d, rec_mean_90d:** Average daily receival weight over 30 and 90 days. This smooths out day-to-day noise.
- **rec_std_30d, rec_std_90d**: Standard deviation of daily receivals. High volatility may indicate unreliable supply or seasonal surges.
- **rec_trend_30d, rec_trend_90d:** Linear trend coefficient computed via np.polyfit. A positive trend suggests increasing receival activity, while a negative trend may signal declining demand or supply issues.
- **rec_count_30d, rec_count_90d:** Number of distinct days with non-zero receivals. This measures supply frequency.
- **rec_cv_30d:** Coefficient of variation (std / mean), normalized measure of volatility.
- **rec_zero_ratio_30d:** Proportion of days with zero receivals (1 - rec_count_30d / 30), quantifying demand intermittency.
- **po_to_rec_ratio_30d:** Ratio of planned to actual supply, revealing systematic over- or under-ordering patterns.

Logistics and Operational Features

These features encode physical and operational constraints of the supply chain.

- **typical_load_rm:** Median weight of a single delivery for each raw material, computed from historical single-day receivals. This feature captures the characteristic "truckload" size, which varies significantly across materials.
- **expected_truck_count:** Estimated number of deliveries, computed as po_in_window / typical_load_rm. This transforms a quantity forecast into a logistics planning metric.

Lead Time and Supplier Reliability Features

These features capture the temporal lag between order placement and delivery, as well as supplier-specific performance metrics.

- **lead_time_mean_rm:** Average lead time for each raw material.
- **lead_time_std_rm:** Variability in lead time, indicating reliability.
- **lead_time_median_rm:** Robust central measure, less sensitive to outliers.
- **sup_lead_mean, sup_lead_std, sup_lead_median:** Supplier-specific lead time statistics.
- **supplier_reliability_score:** A composite metric combining speed and consistency. This score rewards suppliers with short, consistent lead times. The negative signs ensure higher scores indicate better reliability.

Seasonal Prior Features

These features encode learned seasonal patterns from the entire historical dataset. These priors function as regularization anchors. For materials with sparse recent history, the model can fall back on these long-term averages, preventing overconfident extrapolation from limited data.

- **rm_month_mean_prior:** Average daily receival weight for each (rm_id, month) combination across all historical years.
- **rm_dow_mean_prior:** Average daily weight for each (rm_id, day_of_week) combination.

- **global_month_mean_prior:** Average daily weight across all materials for each month.
- **rm_month_vs_global:** Difference between the material-specific and global priors, capturing how much a given material deviates from the typical seasonal pattern.

Year-over-Year (YoY) Feature

- **last_year_weight**: The actual weight received during the same calendar window exactly one year prior. This feature captures annual periodicity that may not be fully represented by monthly or day-of-week effects.

3.3.3 Croston Intermittent Demand Forecast as a Feature

Standard forecasting methods (moving averages, exponential smoothing) perform poorly on such data, as they are designed for continuous, regular demand.
Croston's method, developed specifically for intermittent demand, decomposes the series into two components:

1. **Demand Size** (z): The magnitude of non-zero events
2. **Inter-arrival Interval** (p): The time between non-zero events

The forecast rate is then computed as: rate = z / p
Both are updated using exponential smoothing:

```
for k in range(1, demand.size):
    z = alpha * demand[k]   + (1 - alpha) * z
for k in range(1, intervals.size):
    p = alpha * intervals[k] + (1 - alpha) * p

rate = z / max(p, 1e-6)
return np.full(h, rate, dtype=float)
```

Feature Construction

The build_intermittent_baseline function constructs a daily time series for each rm_id from historical receivals, then applies Croston forecasting to generate a daily rate for the entire forecast horizon and finally sums these daily rates over each specific forecast window to produce baseline_sum_in_window.
This feature provides XGBoost with a sophisticated, domain-appropriate baseline forecast. Rather than learning intermittency patterns from scratch (which is difficult for tree-based models), XGBoost can treat baseline_sum_in_window as a strong prior and learn residual adjustments based on PO signals, supplier reliability and temporal factors.

3.3.4 Training and Validation Strategy

The test set is explicitly defined by prediction_mapping.csv, which contains the exact (rm_id, forecast_start_date, forecast_end_date) tuples for which predictions are required. Features are generated for these rows using the same create_features pipeline applied to training data.

We adopted of a chronological split rather than random sampling (as required by forecasting tasks):

```
val_idx = X_train.groupby('rm_id').tail(150).index
```

For each rm_id, the last 150 forecast windows (corresponding to the most recent temporal endpoints in the training data) are reserved for validation.

This was done for three reasons:

1. **Temporal Realism**: The model will always predict the future based on past data. A chronological split faithfully simulates this scenario, preventing "data leakage" from future to past.
2. **Per-Material Validation**: By splitting within each rm_id, we ensure that validation performance is evaluated across all materials, preventing biases that could arise if certain materials were entirely excluded.
3. **Robustness to Distribution Shift**: The validation set represents the "near future" relative to the training set, making it a reliable proxy for performance on the actual 2025 test data.

Cross-Validation: Considered but Not Implemented

Cross-validation (e.g., TimeSeriesSplit or grouped K-Fold by rm_id) is a standard best practice for robust model evaluation, as it provides multiple independent estimates of generalization performance and reduces variance in score estimates.

Despite its theoretical appeal, cross-validation was not adopted due to these motivations:

1. **Computational Constraints**: The training dataset comprises several million samples. Training K separate models (e.g., K=5) would require approximately 5× the computational budget, translating to multiple days of training time even on high-performance hardware.
2. **Marginal Empirical Gains**: Preliminary experiments with 3-fold cross-validation on a subset of the data showed **negligible improvements** in final Kaggle scores compared to the single chronological split.
3. **Sufficiency of Validation Set**: The validation set contains 150 windows × N rm_ids, resulting in tens of thousands of validation samples. This large, diverse validation set provides a **stable and representative estimate** of out-of-sample performance.
4. **Hyperparameter Tuning Considerations**: Optuna's Bayesian optimization is already computationally intensive *(20 trials × ~1-2 hours per trial*). Nested cross-validation would multiply this cost by K, rendering hyperparameter search impractical.

The single, well-designed chronological split proved sufficient as a proxy for generalization, balancing statistical rigor with practical feasibility.

3.3.5 Hyperparameter Optimization and Final Model Training

Preprocessing for XGBoost

XGBoost requires numerical inputs. The function _encode_rm_id_as_codes transforms rm_id into categorical codes:

```
X_train_full['rm_id'] = X_train_full['rm_id'].astype('category')
rm_categories = X_train_full['rm_id'].cat.categories.tolist()
X_train_full['rm_id'] = X_train_full['rm_id'].cat.codes.astype('int32')
```

It is critical to ensure consistency in the category mapping across the training, validation, and test sets. If rm_id values appear in the test data that were not seen during training, they are mapped to -1 to denote a missing category, an outcome that XGBoost handles effectively. Furthermore, it is important to consider how the model interprets these codes: although the IDs are integers (and thus appear ordinal), XGBoost uses them as categorical boundaries for tree splits. This means the model does not impose a sequential or smooth relationship between numerically adjacent IDs.

Hyperparameter Tuning with Optuna

Hyperparameter tuning is conducted using Optuna, a Bayesian optimization framework that uses Tree-structured Parzen Estimator (TPE) sampling to intelligently explore the hyperparameter space:

```
study = optuna.create_study(direction="minimize", sampler=TPESampler(seed=42))
study.optimize(objective, n_trials=HYPEROPT_N_TRIALS, timeout=None)
```

The following hyperparameters are tuned:

- **eta** (learning rate): [0.001, 0.3], log-uniform
- **max_depth:** [3, 12], integer
- **min_child_weight**: [0.01, 64.0], log-uniform
- **subsample**: [0.5, 1.0], uniform
- **colsample_bytree**: [0.5, 1.0], uniform

- **reg_alpha** (L1 regularization): [1e-8, 10.0], log-uniform
- **reg_lambda** (L2 regularization): [1e-8, 10.0], log-uniform
- **gamma** (min split loss): [0.0, 10.0], uniform

Each trial trains a model with early stopping (200 rounds patience) and returns the validation quantile loss.

Justification for 20 Trials and Seed Selection

A critical aspect of the tuning process was determining the optimal number of Optuna trials. While a larger search space can theoretically find better parameters, it also increases computational cost and the risk of overfitting to the validation set.

Through iterative experimentation and submissions to the Kaggle leaderboard, we determined that 20 trials provided the optimal balance. This configuration was sufficient to find a robust set of hyperparameters, yielding our best public leaderboard score of 5136 without tailoring the model too closely to our specific validation data.

Further experiments with a higher trial count (e.g., 50 or 100) confirmed this dynamic. While local validation scores sometimes improved, the Kaggle performance consistently degraded. This indicated that with more freedom, Optuna began finding parameters that exploited idiosyncrasies of the validation set (a form of hyperparameter overfitting).

Similarly, the SEED = 42 was selected not just for reproducibility, but also after testing adjacent values (e.g., 41, 43). The value 42 provided the most stable and consistent performance between our local validation score and the public leaderboard.

Final Model Training

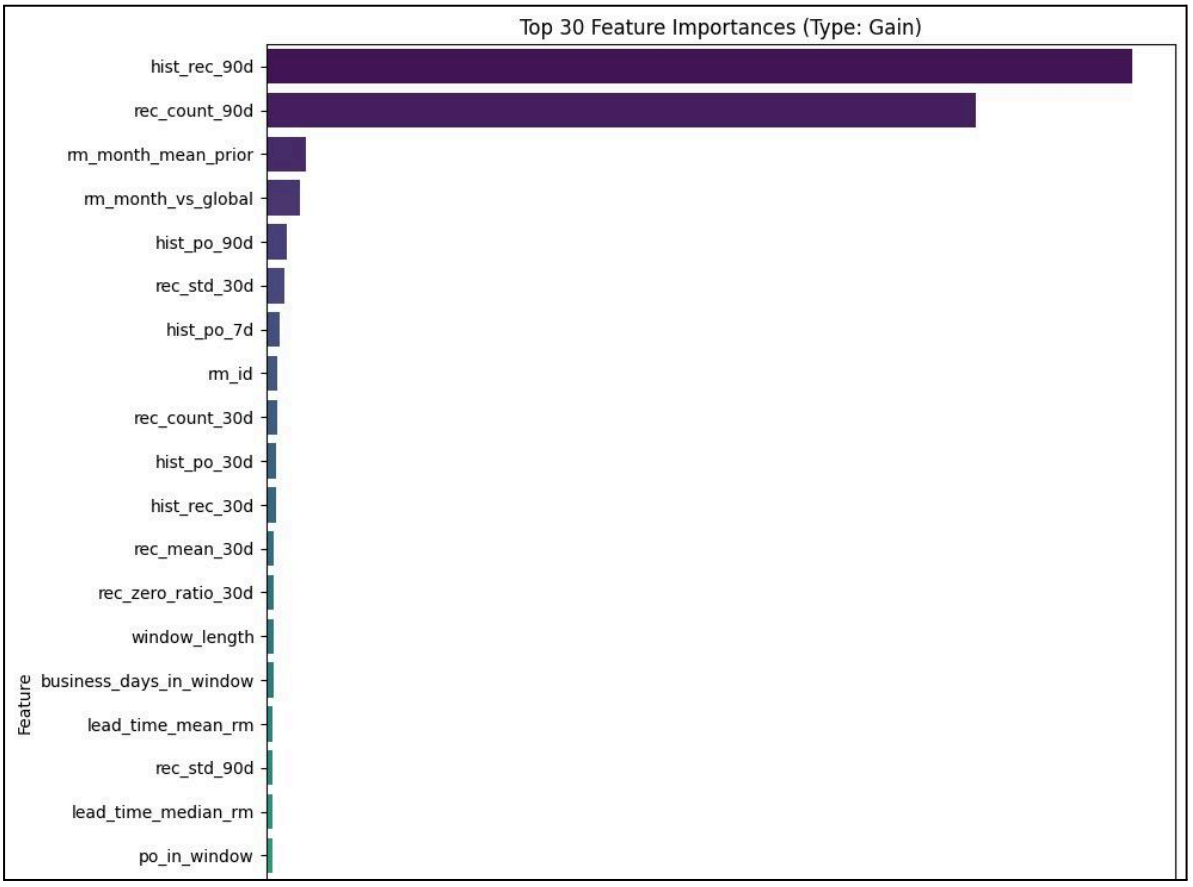
Once optimal hyperparameters are identified, a final model is trained on the combined training and validation sets. After hyperparameter selection, the validation set is no longer needed for model selection.

The num_boost_round is set to best_iter + 1 (the iteration that achieved minimum validation loss during tuning), avoiding further overfitting.

3.3.6 Interpretation and Model Persistence

Feature Importance Analysis

To understand the model's predictive logic, a feature importance analysis was conducted on the final trained booster (booster_final). The “gain” metric was selected as the importance type, as it provides a robust measure of each feature's average contribution to the model's performance. The analysis maps the internal XGBoost feature indices back to their descriptive names and visualizes the top 30 most impactful features. This step is critical for validating that the model has learned logical patterns from the data and for identifying the key predictive drivers.



Selection of some selected features

The chart clearly highlights a steep drop-off in predictive power, with a long tail of features contributing very little gain. We retained this large feature set, including an exhaustive set of rolling statistics and seasonal priors, knowing that XGBoost is

inherently robust to non-informative inputs. The algorithm effectively performs its own feature selection by learning to ignore variables that do not offer predictive value, thus avoiding the risks of premature manual feature removal.

Model Persistence

For reproducibility and future inference, the final model and its associated metadata were persisted to disk. The trained XGBoost model (booster_final) was saved in JSON format as best_xgb_model.json. A separate best_xgb_model_meta.json file was saved, storing the metadata. This file is crucial as it stores the exact list of feature_columns, the learned categorical_rm_id_categories (for encoding rm_id), the quantile_alpha (0.2), and the best_params dictionary found by Optuna. Key metrics from the run, such as the best_validation_score, best_iteration, and RUN_NOTES, were appended to a persistent CSV log (score_log.csv) for longitudinal tracking of model experiments. Example Log Entry:

timestamp	best_validation_score	best_iteration	n_trials	notes	features_used
2025-11-09 14:32:15	1234.56	847	20	-	rm_id|..|..

4. Conclusion

This project was not only a modeling exercise but an extensive process of understanding, both of the data and of the operational system it represents. One of the greatest challenges was to make sense of the heterogeneous datasets and uncover their underlying logic. The data did not reveal its structure immediately: units, timestamps, supplier relationships, and material lifecycles all required careful inspection and contextual interpretation.
Translating these discoveries from the EDA into meaningful predictive features proved equally complex. Each analytical finding, from temporal correlations and supplier dependencies to seasonality and discontinuities, had to be reformulated into variables that a model could interpret.

Prophet was the best public leaderboard scoring solution for this task. Its per-material cumulative modeling captured long-term trends, weekly patterns and holiday effects while remaining robust to sparse data. XGBoost, by contrast, required a richer and more feature-dense representation of the system to reach comparable accuracy. Its performance improved markedly only after incorporating many engineered variables derived from the EDA, such as supplier reliability, delivery frequency, and historical demand patterns.

Overall, the final models produce conservative and stable predictions, respecting the business requirement of prioritizing under-forecasting over over-forecasting. The work lays a solid foundation for future improvements, such as model ensembling, while already delivering a solution for real-world supply chain planning.