From Prediction to Decision

Data Science project: Applications to Real Cases

MSc Supply Chain and Operations Management – 2026-2027

Work	in pairs (a student alone is allowed)
Subject	**two situations of your choice** among the three below
Material	one Jupyter notebook, `project_notebook.ipynb`: it generates the data and contains the first lines of code
Deliverables	a **report** (PDF) and your completed **notebook**
Deadline	**Friday 29 January 2027, 23:59**. A dropbox will be created on Brightspace. One submission per group, you can update it until the deadline.

1 Spirit of the project

In class you learnt the main supervised-learning models (linear and logistic regression, k-NN, SVM, decision trees, random forests, AdaBoost, gradient boosting, neural networks), how to compare them honestly (train/test split, cross-validation, metrics) and how to turn a prediction into a decision (costs and thresholds, explanation of the model). The project asks you to apply all of this, **on your own**, to problems you are likely to meet in your first job.

Each situation describes a company, its data and a business question. There is no single right answer: we assess how you **frame** the problem, how **honestly** you compare the models, how you **decide** with the business costs and how **clearly** you explain your choices and their limits. A simple, well-argued model is worth more than a complex one you cannot explain.

The method: the 8 steps of the final project of the course notebook (Part 14)

1. **Explore**: distributions, rates by group, first intuitions, anything surprising.
2. **Prepare**: keep only what is known at the time of the decision (no leakage), handle missing values and outliers, split train/test and lock the test set away.
3. **Benchmark**: a naive baseline and *all* the models of the course, with the same cross-validation and the right metric.
4. **Improve the variables**: create one or two variables that make business sense.
5. **Tune** the one or two most promising models (`GridSearchCV` on the training set).
6. **Decide and test once**: choose the decision rule from the business costs, then evaluate it *once* on the test set.
7. **Explain**: which variables drive the predictions (permutation importance, partial dependence, rules of a small tree)?
8. **Conclude**: recommendation to the manager, expected gain, limits. At the very end, read the data generator and compare its hidden rules with what your models found.

2 The notebook

Everything is in one notebook. Open it in Jupyter or VS Code (the usual Anaconda installation of the course is enough) and run it from top to bottom.

- The cell **Data generators** creates the three data sets. They are *synthetic* but built to mimic real mechanisms, and they are **identical for everybody** (fixed random seed). Do not modify this cell and there is no need to spend much time on it since they will be explained later.
- For each situation, **EXAMPLE** cells start Steps 1, 2, 3 and 6 on a minimal case (a few variables, two models, the business-cost function). Extend them; everything else is yours. Reuse freely the code of the course notebook.
- The questions in italics in the notebook are there to guide you, not to limit you.

3 Situation 1 – Modalia: which online orders will be returned?

Context

Modalia (a fictitious company) sells clothes, shoes, home items, beauty products and small electronics online in France. About one order in three is returned. A return costs on average **16 €** (return shipping, handling, depreciation of the item). Before shipping, the customer-care team can trigger a **fit check** (size advice by chat, exchange offer): it costs **1.20 €** per order and reduces the probability that the order is returned by **30 %**. *Which orders should receive a fit check, and how much would Modalia save?*

Data `returns`: 24 000 orders, January 2025 – October 2026, one row per order.

`order_date`, `category`, `n_items`, `order_value_eur`, `discount_pct`	the order (date, main category, number of items, value, discount)
`n_size_duplicates`	number of items ordered in several sizes (same model)
`customer_tenure_months`, `customer_past_orders`, `customer_past_return_rate`	the customer's history (the return rate is empty for a first order)
`device`, `payment_method`, `loyalty_member`, `free_returns`, `viewed_size_guide`, `region`, `promised_delivery_days`	how and where the customer ordered
`delivery_delay_days`, `customer_service_contacts`	delay of the delivery, contacts with customer service
`returned`	**target**: 1 if the order was returned

Questions to guide the study.

- Which columns are really known when the decision is taken (before shipping)? What do the missing return rates mean?
- Modalia changed its return policy in March 2026. What is the impact on the data, and on the way you split train and test?
- Which model would you deploy, given that the customer-care manager wants to understand why an order is flagged?
- Compare your rule with "never" and "always" send a fit check: saving per 1 000 orders? How sensitive is it to the 30 % effect?

4 Situation 2 – Logivia: picking time and staffing of a warehouse

Context

Logivia (fictitious) runs a third-party logistics warehouse near Lyon. Every order (e-commerce parcel, store replenishment, B2B order) is prepared – *picked* – by one operator. Every evening the site manager decides how many pickers to plan for each shift of the next day, knowing the orders to prepare. A picker provides **420 productive minutes** per shift; a missing hour costs **38 €** (overtime, temporary staff), an idle hour **24 €**. Today the plan simply uses the average workload of each shift. *Can a model of the picking time of each order give a better plan?*

Data `picking`: 45 000 picking tasks, March – August 2026 (26 weeks, Monday to Friday), one row per task.

`date`, `shift`, `hour`	when the task was done (morning, afternoon, night)
`picker_id`, `picker_experience_months`	who picked it, and since when the picker works on the site
`order_type`, `n_lines`, `n_units`, `n_zones`, `total_weight_kg`	the order: type, number of lines and units, zones of the warehouse visited, weight
`cold_items`, `high_shelf_lines`, `equipment`	cold-room items, lines stored above 2 m, equipment used (cart, pallet jack, reach truck)
`warehouse_occupancy_pct`, `new_wms`	how full the warehouse is; 1 after the new warehouse-management software (June 2026)
`lines_per_minute`	productivity of the task
`pick_minutes`	**target**: picking time of the task (minutes)

Questions to guide the study.

- Which column cannot be used to predict the picking time, and why? What do you do with the very long tasks?
- How much slower is a newcomer? What did the new software change? Can you turn this into useful variables?
- From the predictions of the tasks to the plan of a shift: what is the cost of your plan compared with the current practice? Should you plan with a safety margin, given that a missing hour is more expensive than an idle one?

5 Situation 3 – Rhodanis Parcel Hub: which conveyor motors will fail?

Context

The Rhodanis parcel hub (fictitious) sorts parcels with 90 conveyor motors equipped with sensors (vibration, temperature, current). An unplanned failure stops a line: about **6 000 €** (repair, overtime, late parcels). Every Monday the maintenance team can **inspect 4 motors**; an inspection costs **300 €** and prevents a failure in the next 14 days with probability **0.85**. Today the team inspects the motors that vibrate the most. *Can a model choose better which 4 motors to inspect each week?*

Data `motors`: weekly summaries of the 90 motors, June 2025 – November 2026, one row per motor and week.

`machine_id`, `week`, `model`, `zone`, `age_months`	the motor (three models, three zones of the hub) and its age
`running_hours`, `avg_load_pct`, `ambient_temp_c`	how much and how hard it worked this week, temperature of the hall
`vibration_rms_mm_s`, `motor_temp_max_c`, `current_draw_a`, `micro_stops`	sensor summaries of the week
`hours_since_maintenance`	running hours since the last maintenance
`maintenance_ticket_next_week`	1 if a maintenance ticket was opened for the motor the following week
`failure_next_14d`	**target**: 1 if the motor failed within the next 14 days

Questions to guide the study.

- Failures are rare: which metric do you use? Why is a random split of the rows a bad idea when the same motor appears every week?
- Is a vibration of 4 mm/s worrying for every motor model? Can the history of a motor (trend over the last weeks) help?
- Build the weekly list of 4 motors from your model: net saving compared with the current rule? Would you recommend changing the capacity of 4 inspections?

6 Deliverables and assessment

Upload **one archive** `GroupXX_Name1_Name2.zip` before **Friday 29 January 2027, 23:59**, containing:

- the **report** (PDF, **12 pages maximum** excluding appendix, about 6 pages per situation), written for a manager who knows the course: question, data and traps found, models compared (one summary table), decision and expected gain, explanation, limits. Every number of the report must come from the notebook;
- the **notebook** `project_notebook.ipynb`, completed for your two situations, with its outputs; it must run from top to bottom (*Restart & Run All*). Delete the situation you did not choose.

Assessment (/20).

For each of your two situations (2×7 points)	
Exploration and preparation: data understood, traps detected (leakage, missing values, time order, outliers)	2
Comparison of the models of the course: baseline, same validation, right metric, tuning, summary table	2.5
Decision with the business costs, tested once, compared with the current practice	1.5
Explanation and recommendation, limits	1
Other points	
Report: clarity, structure, figures, concision	4
Notebook: runs from top to bottom, readable, numbers of the report reproducible	2

Both members of a pair receive the same mark.

Rules. You may use any public resource and AI assistants, provided you say where and how in an appendix of the report; you remain responsible for every line of code and every number. Do not share code or results between pairs.