A. Competition setup & objectives

1. What is the business objective of this Datathon?

Rewritten question: What is the main business goal of this Datathon, expressed in one clear sentence?

Answer: The business objective is to **forecast monthly sales volumes for branded drugs after generic entry over a 24-month horizon, with special focus on high-erosion brands, so that Novartis can better anticipate revenue loss, plan post-patent strategies, and optimize product and country-level decisions in the post-LoE (loss-of-exclusivity) period.**

---

2. How are the winners decided?

Rewritten question: Is the competition decided only by a numeric leaderboard, or by a combination of metrics and jury evaluation?

Answer: The winners are decided by a **two-step combination**:

1. **Phase 1 – Numeric metrics only**
   - **Scenario 1 (Phase 1A)**: All teams are ranked on the Scenario 1 metric (no post-entry actuals). The **top 10 teams** move forward.
   - **Scenario 2 (Phase 1B)**: Only these 10 teams are evaluated on the Scenario 2 metric (6 months of post-entry actuals). The **top 5 teams** move to the final phase.
2. **Phase 2 – Jury evaluation among the Top 5**
   - The **5 finalist teams** present their approach, exploratory analysis, modeling choices, and business interpretation to a mixed technical/business jury.
   - The jury then selects the **top 3 winners** based on methodology, interpretability, quality of insights, and business relevance.

So: **metrics decide who reaches the final**, and **the jury decides the final ranking among the 5 finalists**.

---

3. Do we understand the evaluation flow and the relative roles of metrics and jury?

Rewritten question: Do we clearly understand how Phase 1 and Phase 2 work, and how the metric and the jury each influence the final outcome?

Answer: Yes, the evaluation flow is:

- **Phase 1A – Scenario 1 metric**
  - Evaluate all teams using **Metric 1** (Scenario 1: predictions from month 0 to 23 with no post-entry actuals).
  - Select **Top 10 teams** by lowest prediction error.
- **Phase 1B – Scenario 2 metric**

    - On those 10 teams, evaluate using **Metric 2** (Scenario 2: predictions from month 6 to 23 with actuals available for months 0–5).
    - Select **Top 5 teams** by lowest prediction error.

- **Phase 2 – Jury**

    - The 5 finalists present their work.

    - The **jury** (technical + business experts) chooses the **Top 3** based on:

        - soundness of methodology,
        - clarity and depth of EDA & feature engineering,
        - interpretability of the models (especially for high-erosion Bucket 1),
        - quality of business story and implications.

The Datathon brief does **not** give a numeric weight (e.g., 50% metric, 50% jury) inside Phase 2. Practically:

- **Metrics fully determine who is in the Top 5.**
- Among those 5, **the jury's qualitative evaluation fully determines the final ranking** (1st–3rd).

---

4. What are the required deliverables at each stage?

Rewritten question: What exactly must we deliver as files during the competition, and who has to deliver what?

Answer:

- For **all teams**:

    1. **Prediction files (submissions)** on the test set:

        - In the format of `submission_template.csv` : `country, brand_name, months_postgx, volume`

        - Covering all required `(country, brand, months_postgx)` combinations for:

            - **Scenario 1** (months 0–23),
            - **Scenario 2** (months 6–23).

        - Uploaded via the **submission platform**, where metrics are computed.

- For **Top-5 finalists**: 2. **Slide deck (presentation)**:

    - Prepared using the template provided in **Microsoft Teams → Novartis Datathon → Files**.

    - Must summarize:

        - data understanding and preprocessing (especially high-erosion Bucket 1),

        - modeling choices and validation strategy,

        - key insights and business interpretation,

- limitations and possible next steps.

3. **Code package**:

   - The code used to generate the **final selected submission** (the one marked as final in the platform between 9:30 and 10:30 on Sunday).
   - Uploaded to the private mentoring channel (Teams) by the deadline, following the naming rules provided.

So: **everyone** submits prediction CSVs; **Top-5** also submit **slides + code**.

---

5. What constraints do we have (data, tools, submissions, compute)?

Rewritten question: What explicit or implicit constraints apply to data usage, tools, submissions, and computing resources?

Answer:

From the brief and supporting files we know **explicitly**:

- **Submission limits:**
  - The platform allows **up to 3 submissions every 8 hours**.
  - This forces you to be strategic and not spam submissions.
- **Submission format:**
  - Must exactly follow `submission_template.csv`:
    - Columns: `country, brand_name, months_postgx, volume`.
    - All required rows present, no extras, no missing keys.
- **Metrics:**
  - Must respect the official metrics in `metric_calculation.py`, including:
    - Use of `avg_vol` and `bucket` from `auxiliar_metric_computation.csv`.
    - Bucket 1 being weighted double.
- **Test split for leaderboard:**
  - The test set is split into:
    - 30% **public** part (used for live leaderboard),
    - 70% **private** part (used for final scoring).
  - You will not see performance on the private part until the end.

What is **not explicitly specified**, but we can sensibly assume:

- **External data / internet:**
  - The documentation does not explicitly forbid external data, but:

        - The task and metrics are designed around the **provided datasets** (`df_volume_*` , `df_generics_*` , `df_medicine_info_*`).
        - Any external data would likely not be available to the organizers when evaluating, so you should be careful not to rely on unavailable or unverifiable sources.
    - Good practice: **focus on the provided data** and do not use external proprietary data.

- **Libraries / frameworks:**
    - No specific limitations are mentioned.
    - In practice, typical Datathon expectations:
        - Use standard open-source Python/R libraries (e.g., pandas, numpy, scikit-learn, XGBoost, LightGBM, CatBoost, basic deep learning if you want).
        - No need for commercial/paid packages.

- **Compute assumptions:**
    - No formal limits are given, but:
        - The solution should be **trainable and re-runnable** on a typical laptop or standard cloud VM within reasonable time.
        - Extremely heavy models requiring days of training or specialized hardware are risky and unnecessary.

So the **hard constraints** you must strictly respect are:

- submission frequency,
- file format,
- correct use of metrics and buckets.

---

6. How should we allocate our time during the competition?

Rewritten question: Given the schedule (kick-off, working time, and Sunday deadlines for final submission, slides, and presentations), how do we split our effort across understanding, modeling, and storytelling?

Answer (practical plan):

You can think of the work in **four blocks**, roughly:

1. **Understanding + EDA (Day 1 and early Day 2)**
    - Clarify business problem (generic erosion, Bucket 1 focus).
    - Explore:
        - how `volume` behaves pre/post generic entry,
        - how erosion differs by `bucket` , `ther_area` , `biological` , `hospital_rate` , `n_gxs` ,
        - distribution of `months_postgx` for train and test.

- Output: clear mental model of erosion patterns + some key plots.

2. **Modeling + validation (mainly Day 2 and part of Day 3)**

    - Design:

        - a consistent **feature pipeline** using `df_volume_train`, `df_generics_train`, `df_medicine_info_train`,
        - **time-aware validation** aligned with Scenario 1 and 2.

    - Train and iterate:

        - baseline models (naive, simple regression),
        - hero models (e.g., gradient boosting on engineered time-series features).

    - Use `metric_calculation.py` locally to mimic Phase 1A/1B.

3. **Business narrative + visualizations (late Day 3)**

    - Build story around:

        - high-erosion brands (Bucket 1),
        - which product characteristics drive faster erosion,
        - practical implications for post-LoE strategy.

    - Create a few strong, clean plots:

        - volume vs `months_postgx` for typical high- and low-erosion examples,
        - differences by therapeutic area, biological vs small molecule, etc.

4. **Final polishing, submissions, and slides (Sunday morning)**

    - Use the submission platform to:

        - run a **final clean training + inference** run,
        - generate **final submission file**, and mark **final option** between 9:30–10:30.

    - Prepare and polish slides:

        - follow the Teams template,
        - ensure coherence: Problem → Data → Model → Results → Business impact → Limitations.

    - Package **code** for upload by 12:00.

In other words: **front-load understanding and validation**, then **solidify one or two robust models**, and end with **a clear story and a clean submission**, not endless small tweaks.

---

7. Are there official baselines or starter notebooks?

Rewritten question: Do we have any official baseline models or notebooks, and do we know what performance we need above baseline to be competitive?

Answer:

From the material you provided:

- There is **no explicit mention** of:
  - official baseline models (e.g., "naive last value", "simple exponential smoothing"), or
  - starter notebooks with prebuilt models.
- What **is** provided is:
  - the **metric code** (`metric_calculation.py`),
  - the data files and submission examples,
  - clear instructions for how to locally compute the official metrics.

So:

- We should assume **no official baseline** is given.
- We must construct our own internal baselines, for example:
  - naive forecast (e.g., repeat last pre-LoE volume or a simple decay),
  - simple linear model on a few lags and `months_postgx` .

Regarding "how much above baseline is competitive":

- There is no numeric guidance in the brief.
- A practical approach:
  - Define an internal baseline (naive model).
  - Aim for a **significant reduction in the PE** (Metric 1/2) – e.g., at least **10–20% improvement** over naive in both buckets, especially Bucket 1.
  - Focus not only on absolute score but also on **stability across folds** and a strong story for Bucket 1.

---

8. Are there reference solutions from past Datathons?

Rewritten question: Do we have example submissions, code, or winning slide decks from previous Novartis / Barcelona Digital Finance Hub Datathons?

Answer:

In the material you shared, there is:

- No direct link to **past winning code**,
- No examples of **past slide decks**,
- No example of a **complete prior solution**.

The only examples we have are:

- `submission_example.csv` (just a structural example with zeros),
- `submission_template.csv` (format to follow),
- `auxiliar_metric_computation_example.csv` (toy auxiliary file with `avg_vol` and `bucket`).

So:

- We **do not** have official reference solutions from previous editions in this package.
- We must infer expected standards from:
    - the level of detail in the metric script and documentation,
    - the instructions about EDA, visualization, and focus on high-erosion brands.
- Good working assumption:
    - The jury expects **clean, well-structured code**,
    - a **coherent narrative** connecting data patterns to business decisions,
    - and **clear, uncluttered plots** that highlight erosion dynamics.

---

9. What are the rules around explainability, ethics, and generative AI?

Rewritten question: What do we know (or reasonably infer) about requirements for model explainability, responsible AI, and the use of tools like ChatGPT/Copilot?

Answer:

From the Datathon brief:

- There are **no explicit paragraphs** detailing:
    - explainability requirements,
    - ethics guidelines,
    - or explicit rules about generative AI tools.

However, the design of the challenge implies:

- **Explainability:**
    - The brief explicitly asks for:
        - a “deep exploratory analysis”,
        - a **business-oriented** and interpretable narrative,
        - special focus on **high-erosion Bucket 1** brands.
    - This strongly suggests that:
        - Black-box models are **allowed**, but you must be able to **explain their behavior**:
            - Which features drive erosion predictions?
            - How do predictions differ by bucket, therapeutic area, biological vs small molecule?
        - The jury will value models whose decisions can be **reasoned about and challenged**, not just raw scores.
- **Ethics & responsible AI:**
    - Data: commercial/volume data in healthcare context, not patient-level data.

  - Still, you should:
    - avoid overly strong or causal claims (“this model proves X causes Y”),
    - frame results as **decision support**, not as deterministic truth,
    - be careful about potential biases across countries or therapeutic areas.

- **Generative AI tools:**
  - No prohibition is stated.
  - Common Datathon practice (unless explicitly banned) is:
    - You may use tools like ChatGPT/Copilot for **coding help, documentation, and conceptual support**.
    - You must **not** use external data to “peek” at hidden labels or future volumes (which is impossible anyway here).
    - Final decisions and numbers must come from **running models on the provided datasets**.

So our safe working assumption:

- Use generative AI to **accelerate coding and documentation**, but:
  - be explicit in slides that the **core modeling, validation, and conclusions are derived from the provided data and metrics**,
  - ensure full **reproducibility** without external online dependencies.

---

10. Are there bonus elements the jury typically values, and what should we prioritize?

Rewritten question: Beyond the core metric and slides, what extra elements could make our solution stand out, and which are realistic to prioritize?

Answer:

The brief does not explicitly define “bonus points”, but based on how the evaluation is described (especially Phase 2), it is reasonable to assume the jury will value:

1. **Concrete decision-support framing**
   - For example:
     - “Here is how a country brand team could use our erosion forecasts to:
       - plan promotional spend after LoE,
       - adjust inventory planning,
       - prioritize which brands to monitor more closely.”
   - Showing **one or two realistic scenarios** (e.g., a high-erosion brand vs a stable brand) with clear explanations of the decisions that could be taken is very powerful.
2. **Simple dashboard or visualization concept**
   - Even a **mock-up** (e.g., a slide with a dashboard sketch or simple notebook screenshots) that:

        - plots erosion curves by `country, brand`,
        - highlights Bucket 1 brands,
        - shows differences by therapeutic area or `hospital_rate`,
        - gives a ranked list of “brands at highest risk”.
    - You do not need a fully deployed web app; a **clean concept** is enough.

3. **High-level deployment / MLOps sketch**
    - A slide that briefly describes:
        - data flow (**volume** → **features** → **model** → **dashboard**),
        - how often the model could be retrained,
        - what monitoring would be used (e.g., track error per bucket and per brand over time).

Given limited time, you should **prioritize**:

1. **Decision-support scenario + business story**
    - This directly aligns with the Datathon’s business motivation and will strongly influence the jury.
2. **Clear, focused visualizations**
    - A few high-quality plots showing:
        - typical high-vs-low erosion,
        - feature effects (e.g., impact of `n_gxs`, biological vs small molecule),
        - prediction vs actual for representative brands.
3. **A compact deployment sketch (if time remains)**
    - One slide is enough to show that the solution could realistically be integrated into Novartis’ planning processes.

A full interactive dashboard is **not required** to win, but a good business story and interpretable visualizations almost certainly are.

B. Problem, stakeholder, and domain (11–20)

11. What real-world problem are we solving?

Rewritten question: In everyday language, what is the real business problem this Datathon is trying to address?

Answer: The real-world problem is: **Novartis wants to anticipate how much sales of a branded drug will drop after generic competitors enter the market, so that finance and brand teams can better plan revenues, budgets, and strategies once the patent expires.**

In other words, we help them **see the shape and speed of generic erosion** brand by brand and country by country, instead of being surprised by sudden sales drops.

---

12. Who is the main stakeholder for our solution?

Rewritten question: Who mainly benefits from our forecasts and will actually use them?

Answer: There are three key groups, but the "core" stakeholder is:

1. **Global/Regional Finance Leadership**
    - They need accurate **revenue forecasts** and **budget planning** across countries and brands.
2. **Digital Finance Hub Analysts**
    - They are the **builders and maintainers** of analytical tools.
    - They will use our approach as a **template** or **prototype** for future models.
3. **Country-level Brand Managers**
    - They care about **what will happen to their specific brand** post-LoE.
    - They use erosion forecasts to adjust **marketing, pricing, and resource allocation**.

Practically, the **primary stakeholder** for the Datathon solution is the **Digital Finance Hub + Finance leadership**, because they will integrate this into forecasting processes. But the **end beneficiaries** also include **brand managers**, who use these insights to craft post-patent strategies.

---

13. What decisions should our forecasts support?

Rewritten question: Which concrete business decisions can people make using our erosion forecasts?

Answer: Our forecasts should help answer questions like:

1. **Revenue and budget planning**
    - "How much will this brand's volume drop in the first 2 years after generics enter?"
    - "How should we adjust our **P&L projections** and **country budgets**?"
2. **Post-LoE strategy for brands**
    - "Should we **invest more, maintain, or cut** marketing spend after LoE?"
    - "Is this brand still worth supporting, or should resources move to other products?"
3. **Risk management for high-erosion brands (Bucket 1)**
    - "Which brands are in **Bucket 1** (high erosion, mean erosion 0–0.25) and at risk of a big volume collapse?"
    - "Which brands deserve **extra attention, mitigation plans, or alternative strategies** (e.g., line extensions, new formulations, or portfolio shifts)?"
4. **Scenario comparison**
    - "What do we expect **right at generic entry** (Scenario 1) vs. **after 6 months of data** (Scenario 2)?"
    - "Does real performance confirm or contradict our expectations?"

So, the forecasts are not just numbers; they are **inputs to planning, budgeting, and strategic choices** around each brand's post-LoE life.

---

14. What is the current status quo without this solution?

Rewritten question: How are these decisions typically made today, without the Datathon model?

Answer: Without a structured erosion forecasting model, the situation likely looks like this:

1. **Heuristics and rough rules**
   - Teams use "typical erosion curves" from past LoE cases:
     - e.g., "oncology brands usually drop fast; cardiovascular brands a bit slower"
   - This is **experience-based**, not systematically quantified.
2. **Manual Excel-based analysis**
   - Finance and brand teams:
     - export historical volumes to Excel,
     - fit simple trends or compare with a few past similar brands,
     - then manually adjust forecasts.
3. **Very simple statistical forecasts**
   - Univariate time-series models or rules like:
     - "take last year's volume and subtract X% after LoE."
   - These often **ignore important drivers** like:
     - number and timing of generic entries (`n_gxs`),
     - therapeutic area (`ther_area`),
     - biological vs small molecule,
     - hospital vs retail (`hospital_rate`).

In short, the status quo is **fragmented, manual, and heuristic**, with limited ability to consistently quantify and compare erosion patterns across hundreds of country–brand combinations.

---

15. How should we view the technical task?

Rewritten question: From a modeling perspective, what exactly is the technical problem we're solving?

Answer: Technically, the problem is best seen as:

1. **Panel time-series forecasting**
   - We forecast **monthly volume** for each `(country, brand)` around the **generic entry date**, over a horizon of **24 months after entry**.

  - Each time series is aligned using `months_postgx` (negative months before entry, 0 at entry, positive after).

2. **Two forecasting regimes (scenarios)**
    - **Scenario 1:**
        - Forecast months 0–23 with **no post-entry actuals** available.
        - Purely based on pre-LoE history + drug and generics characteristics.
    - **Scenario 2:**
        - Forecast months 6–23, when months 0–5 actuals are **already known**.
        - Model must adapt to new information and refine predictions.
3. **Bucket structure (erosion severity)**
    - Brands are classified into two **erosion buckets** using mean normalized erosion:
        - **Bucket 1:** high erosion (0–0.25).
        - **Bucket 2:** medium/low erosion (>0.25–1).
    - Metrics **weight Bucket 1 twice as much**, so we implicitly have a form of **stratification** where high-erosion cases are more important.

So, the core technical task is a **time-series/panel forecasting problem with aligned timelines, two information regimes, and a weighted focus on high-erosion brands.**

---

16. What exactly is the target, and how do we interpret volume levels?

Rewritten question: What exactly are we predicting, and how should we interpret high, low, or zero values after generic entry?

Answer:

- **Target variable:**
    - `volume` in `df_volume_*` = **monthly sales volume (units sold)** for each `(country, brand_name, month, months_postgx)`.
- **Interpretation:**
    - **High volume after generic entry**:
        - The brand is relatively **resilient**.
        - Erosion is **low or moderate**; brand may retain strong loyalty, pricing power, or differentiation.
    - **Low volume after generic entry**:
        - The brand is **strongly eroded** by generics.
        - This is typical for **Bucket 1** cases (mean normalized erosion near 0).

- **Zeros or near-zero volumes:**
    - These likely correspond to:
        - **Full erosion** (brand essentially replaced by generics),
        - Possible **discontinuation** of the brand in that country.
    - For modeling:
        - They are valid outcomes—**not errors**—and represent "worst-case" erosion.
        - They must be handled carefully to avoid numerical instability (log transforms, etc.) but conceptually they are just **extreme erosion**.

So we are predicting **actual unit volumes**, which then drive the computation of normalized erosion and bucket behavior.

---

17. Which domain constraints affect how forecasts are used?

Rewritten question: What real-world pharma/finance constraints might shape how these forecasts can and should be used?

Answer:

Several domain constraints matter:

1. **Risk and compliance policies**
    - In a pharmaceutical company:
        - forecasts feed into **official financial planning**, which is scrutinized internally and externally.
    - Models must be **traceable and defensible**:
        - Finance and compliance teams may challenge large changes in forecasts.
2. **Regulatory and market access environment**
    - In some countries, pricing and reimbursement are heavily regulated.
    - Model outputs cannot directly drive:
        - "automatic" pricing decisions or policy actions, without human review.
3. **Need for interpretability**
    - When volumes change sharply (e.g., high-erosion brand in Bucket 1), leadership will ask:
        - "Why does the model predict such a drop?"
        - "Is it because of more generics, therapeutic area, hospital rate, or something else?"
    - A completely opaque model is risky; we need:
        - **clear explanations** and **drivers of erosion**.

4. **Use as decision support, not replacement**
    - The model should be seen as a tool to **support** decisions, not to **replace** judgment.
    - It should be combined with:
        - local market knowledge,
        - upcoming events (new indications, competitor launches, policy changes).

These constraints imply we should build **interpretable, robust models** and communicate: “This is a **supporting tool** that quantifies erosion patterns, not an automatic policy engine.”

---

18. What happens if our predictions are wrong?

Rewritten question: In business terms, what are the consequences of over- or under-predicting post-LoE volume?

Answer:

1. **Overestimating post-LoE volume (underestimating erosion)**
    - We think the brand will retain more volume than it actually does.
    - Consequences:
        - **Overly optimistic revenue forecasts** → budgets based on income that will not materialize.
        - **Inventory risk** → overstock, potential write-offs, waste.
        - Possible **over-investment** in marketing/sales for a brand that is already collapsing.
2. **Underestimating post-LoE volume (overestimating erosion)**
    - We expect a huge drop, but the brand holds better than expected.
    - Consequences:
        - **Missed opportunity**:
            - Less marketing support than needed,
            - Under-allocation of resources to a brand that could still perform well.
        - **Conservative revenue forecasts**:
            - Could protect against disappointment, but may lead to:
                - too cautious investment,
                - losing ground to competitors who support their brands better.
3. **Unstable forecasts**
    - If predictions swing a lot when updated (especially between Scenario 1 and Scenario 2), planning becomes:

    - **volatile and hard to trust**.
  - Stakeholders may:
    - stop relying on the model,
    - revert to manual heuristics.

So accuracy matters, but **stability and interpretability** are just as important for building **trust** and enabling consistent planning.

---

19. Which domain patterns should we focus on in EDA?

Rewritten question: What specific erosion-related patterns in the data should we deliberately explore and visualize?

Answer:

We should explicitly study:

1. **Differences between Bucket 1 and Bucket 2**
   - Compare average erosion curves (normalized volume vs `months_postgx`):
     - Bucket 1: mean erosion 0–0.25 (sharp drops).
     - Bucket 2: mean erosion >0.25–1 (more stable).
   - Look at:
     - how quickly volumes fall in each bucket,
     - how early the main drop occurs (first 6–12 months vs later).
2. **Life-cycle trajectory**
   - For each brand:
     - **Pre-entry**: (months_postgx < 0) growth or stabilization phase.
     - **At entry**: month 0 — inflection point.
     - **Post-entry**: (months_postgx ≥ 0) erosion phase.
   - Plot:
     - typical shapes (growth → plateau → drop),
     - variations by therapeutic area or region.
3. **Impact of `n_gxs` (number_of_gx)**
   - Explore how:
     - the **timing** of the first generic,
     - the **speed** at which `n_gxs` rises (0 → 1 → 2 → ...),
     - the **level** of competition (e.g., 1 vs 5 generics)
   - affects erosion patterns.

4. **Drug characteristics from `df_medicine_info_*`**
    - E.g.:
        - **Therapeutic area** (`ther_area`):
            - Do oncology drugs erode differently from cardiovascular or anti-infectives?
        - **Biological vs small_molecule**:
            - Biologicals may erode more slowly due to complexity, regulation, or substitution issues.
        - **Hospital_rate**:
            - High hospital_rate may indicate different prescribing and tender dynamics compared to retail.
        - **Main_package**:
            - Packaging format may correlate with usage patterns and substitution ease.

These EDA insights are crucial for both:

- **feature engineering** (what to feed into the model), and
- **business narrative** (what drives erosion and how we explain model behavior).

---

20. How would senior stakeholders define "success"?

Rewritten question: In non-technical words, how would a CFO, finance director, or brand lead describe a successful outcome of this Datathon?

Answer:

They would likely describe success along these lines:

1. **More reliable and stable erosion forecasts**
    - "Our forecasts of post-LoE sales for each brand and country are **more accurate and consistent** than before."
    - "We no longer get surprised by sudden drops after generics enter."
2. **Better anticipation of revenue drops**
    - "We can **see revenue decline coming** well in advance and adjust budgets accordingly."
    - "We can plan for **inventory, staffing, and investments** with more confidence."
3. **Clear prioritization of high-risk brands**
    - "We have a clear list of **high-erosion brands (Bucket 1)** by country that need attention."
    - "We know where to **focus mitigation efforts** and where we can afford to reduce investment."
4. **Actionable, understandable insights**

- “We understand **why** some brands erode faster—number of generics, therapeutic area, biological vs chemical, hospital share, etc.”
- “We can **explain these patterns** to local teams, not just show a black-box score.”

In simple terms, success is:

> “We trust these forecasts enough to **use them in real planning** and they help us **avoid unpleasant surprises** and **prioritize our efforts** after patents expire.”

C. Metric, risk tolerance, and constraints (21–30)

21. Have we correctly implemented the official scoring rules in code?

Rewritten question: Have we fully captured the official scoring rules (Phase 1A and Phase 1B, plus bucket weights) in our code, and do we understand how they work?

Answer: Yes. The provided `metric_calculation.py` **exactly encodes** the Datathon metrics:

- **Phase 1A (Scenario 1 – no post-entry actuals)** Implemented in `_compute_pe_phase1a` and `_metric1`:
    - Uses a **normalized prediction error** based on:
        - `sum_abs_diff(0–23)` → monthly absolute error across all 24 months, weighted 0.2.
        - `abs_sum_diff(0–5)` → absolute error on **total** volume in months 0–5, weighted 0.5.
        - `abs_sum_diff(6–11)` → absolute error on total volume in months 6–11, weighted 0.2.
        - `abs_sum_diff(12–23)` → absolute error on total volume in months 12–23, weighted 0.1.
    - Each term is normalized by `avg_vol` and the number of months in that window.
    - It then:
        - Groups by `(country, brand_name, bucket)`,
        - Filters series whose **start month** is 0 (Scenario 1),
        - Computes a **PE (prediction error)** per series,
        - Aggregates by buckets with:
            - **Bucket 1 weighted 2×**, Bucket 2 weighted 1×,
            - and each bucket normalized by the number of series in that bucket.
- **Phase 1B (Scenario 2 – 6 months of actuals)** Implemented in `_compute_pe_phase1b` and `_metric2`:
    - Uses a similar idea, but only from month 6 onwards:
        - `sum_abs_diff(6–23)` → monthly absolute error, weighted 0.2.
        - `abs_sum_diff(6–11)` → total error in early post-entry period, weighted 0.5.
        - `abs_sum_diff(12–23)` → total error in later period, weighted 0.3.

- Again normalized by `avg_vol` and number of months.
- It:
  - Groups by `(country, brand_name, bucket)`,
  - Filters those whose **start month** is 6 (Scenario 2),
  - Aggregates with the same **2× weight for Bucket 1** and **1× for Bucket 2**.

- **Bucket & avg_vol information** comes from `auxiliar_metric_computation_example.csv` (or its real counterpart), which provides:
  - `country`, `brand_name`,
  - `avg_vol` (average pre-entry monthly volume over 12 months),
  - `bucket` (1 or 2 based on mean normalized erosion).

So yes: **the official scoring rules are fully translated into code**. We can use `compute_metric1` and `compute_metric2` locally to estimate our performance before submitting.

---

22. Does the metric treat over- and under-prediction symmetrically?

Rewritten question: Does the metric punish over- and under-predictions in the same way, or is one direction implicitly more costly?

Answer: Mathematically, the metric is:

- Based on **absolute differences**:
  - `sum(|actual - pred|)` (monthly),
  - `|sum(actual) - sum(pred)|` (aggregated windows).
- This is **symmetric**: over-predicting by +X or under-predicting by –X contributes the same amount to the error.

However, there are two important nuances:

1. **Early months are weighted more heavily**:
   - Phase 1A:
     - Months 0–5 total error has weight 0.5 (largest).
     - Middle months (6–11) and late months (12–23) have smaller weights.
   - Phase 1B:
     - Months 6–11 total error has weight 0.5 (largest),
     - Months 12–23 get 0.3, and monthly error 6–23 gets 0.2.
   - So **errors in the early post-entry period are much more penalized**, regardless of direction.
2. **High-volume brands are scaled by `avg_vol`**:
   - Errors are normalized by `avg_vol` (average volume pre-entry).

  - This reduces the dominance of large brands, but **relative errors** still matter more for brands with larger, stable histories.

In summary:

- **Over- and under-prediction are symmetric in magnitude** (absolute error),
- but **errors in early months after entry are structurally more costly** in the metric, which mirrors the business importance of the early erosion period.

---

23. How well does the metric align with business costs of errors?

Rewritten question: Does the way the metric is built match the financial/business impact of errors? Should we further emphasize some cases internally?

Answer: The metric is **well-aligned with business concerns**:

- It **emphasizes early post-entry months**:
  - In Phase 1A: months 0–5 (where the sharp drop usually happens) receive the **highest weight (0.5)**.
  - In Phase 1B: months 6–11 carry the highest weight.
  - This reflects the reality that **early erosion** is critical for:
    - adjusting revenue forecasts,
    - inventory planning,
    - deciding whether to maintain or cut investments.
- It **double-weights Bucket 1** (high-erosion brands):
  - Bucket 1 PE is multiplied by 2, Bucket 2 by 1.
  - This matches the business fact that **high-erosion brands are the riskiest** and most critical to get right.

Internally, we might still choose to:

- Perform **dedicated error analysis on Bucket 1**:
  - E.g., track separate metrics for Bucket 1 and try to minimize them.
- Prioritize **scenario 1 performance for Bucket 1**:
  - The riskiest situation is “no post-entry data yet + high erosion”.

So yes, the metric largely reflects **business cost structure**, and internally we can **reinforce focus on Bucket 1** and early months in our model design and monitoring.

---

24. What secondary metrics should we track internally?

Rewritten question: Besides the official metric, which other metrics should we compute to better understand our model?

Answer: In addition to the official PE metrics (Metric 1 and 2), we should track:

1. **Normalized MAE per scenario and bucket**
    - E.g., MAE / avg_vol:
        - For Scenario 1 and Scenario 2 separately,
        - For Bucket 1 vs Bucket 2.
2. **MAPE-like measures**, where feasible
    - Careful with very small volumes, but where volume is reasonable:
        - MAPE by bucket, scenario, and therapeutic area.
3. **Bucket-specific error tracking**
    - Separate dashboards for:
        - Bucket 1 error distribution (critical),
        - Bucket 2 error distribution (secondary but still relevant).
4. **Segmented errors by drug characteristics**
    - Track errors per:
        - `ther_area` (e.g., oncology vs cardiovascular vs anti-infectives),
        - `biological` vs `small_molecule`,
        - `hospital_rate` bands (e.g., mostly hospital vs mostly retail).
5. **Time-profile errors**
    - Plot average error per `months_postgx`:
        - Are we systematically over- or under-predicting around month 0–6?
        - Are errors higher later (12–23)?

These secondary metrics help us **debug and interpret** the model, even though the competition is judged only on Metric 1 and Metric 2.

---

25. Do we understand the public/private test split and its implications?

Rewritten question: How does the hidden test set work (public vs private), and what does that mean for our strategy?

Answer: Yes, the process is:

- The **test set** is split into:
    - **Public test (30%)** → used for **online leaderboard** during the Datathon.

  - **Private test (70%)** → used only **after submissions close** for the final evaluation.

Implications:

- The **Leaderboard** only reflects performance on the **public 30%**.
- Final ranking (top 10, top 5, winners) is based on the **full test set** (public + private).
- Overfitting to the leaderboard is risky:
  - If we tune heavily to squeeze tiny improvements on the public 30%, we might degrade performance on the hidden 70%.
- Therefore, we must:
  - rely more on our **local validation (time-based splits)**,
  - use the leaderboard as a **signal, not absolute truth**.

---

26. Are there practical constraints on latency, model size, or training time?

Rewritten question: What operational constraints do we face regarding how big/slow our model can be?

Answer: Explicitly, the Datathon rules mainly constrain **submissions**, not model size:

- **Latency**:
  - Predictions are batch-generated (24 months per series); latency is **not critical** for the competition.
- **Model size/memory**:
  - There is no explicit limit, but:
    - Participants run on their **own machines** (laptops/VMs),
    - Very large deep models or heavy hyperparameter searches may be impractical.
- **Training time**:
  - We may need to retrain models **several times** during the weekend.
  - Therefore:
    - Training should be **reasonably fast** (e.g., minutes, not many hours).
    - This favors **gradient boosting, classical TS models, or light deep architectures** over huge networks.

In practice, we should design models that:

- Fit comfortably in memory,
- Train within our time budget,
- Allow for **rapid iteration** and **frequent retraining** when we adjust features or validation strategy.

---

27. Are black-box models acceptable if we explain them?

Rewritten question: Can we use complex models (e.g., boosted trees, ensembles) as long as we make them understandable to the jury?

Answer: Yes, nothing in the rules forbids **black-box models**. The key expectations are:

- **Acceptable models**:
    - Gradient-boosted trees (XGBoost, LightGBM, CatBoost),
    - Ensembles of multiple models,
    - Possibly neural networks, if justified.
- **But we must provide explanations**, particularly for Bucket 1 / high-erosion cases:
    - **Global level**:
        - Feature importance (e.g., Shapley values, gain, split importance),
        - Typical erosion curves by bucket, therapeutic area, etc.
    - **Local level**:
        - For a few representative brands:
            - Explain why the model predicts strong erosion or resilience (role of `n_gxs`, pre-entry trend, therapeutic_area, etc.).

Thus, black-box models are acceptable **as long as we treat explainability as a first-class requirement** in our slides and discussion with the jury.

---

28. Are there imbalance/scale issues we must consider?

Rewritten question: Given big differences in volume across brands and countries, how do scale/imbalance issues affect evaluation and our internal modeling?

Answer: Yes, there are strong **scale differences**:

- Some brands/countries have very high pre-entry volumes, others are very small.
- The metric already addresses this somewhat by **normalizing error by** `avg_vol`:
    - This keeps each series' contribution comparable.

Still, we should:

1. **Normalize or standardize internally**
    - Many approaches benefit from:
        - log-transforming volume,
        - working with **normalized volume** (e.g., divided by pre-entry average).
2. **Check imbalance in buckets and segments**

   - Bucket 1 vs Bucket 2 counts,
   - Differences by therapeutic area, country, etc.

3. **Inspect error distributions**

   - Ensure that large brands are not implicitly dominating training or internal validation in ways that the official metric does not.

So yes, scale is a real issue, but the official metric is already **per-series normalized**, and we should mirror that normalization in our internal model design and evaluation.

---

29. Should we overlay threshold-based business rules on top of forecasts?

Rewritten question: Even if not required, are there simple business rules we could define using our forecasts to make them more actionable?

Answer: Yes, and this can **strengthen our business story**:

1. **Critical brand flagging**

   - Define:

     - "Critical brands" = those predicted to have mean normalized erosion **below 0.25** in the first 24 months (i.e., clearly in **Bucket 1**).

   - Use this to:

     - create a **watchlist of high-risk brands** per country.

2. **Deviation from historical pattern**

   - If predicted post-entry volume is:

     - very different from simple heuristics (e.g., naive continuation or average life-cycle shape),

   - we can trigger:

     - a **"deep dive required" flag** → recommended human review.

3. **Alert on sudden changes between Scenario 1 and Scenario 2**

   - If adding the first 6 months of actuals massively changes the forecast:

     - highlight these brands as **unstable** and needing **manual inspection**.

These rules are not needed for the numeric score, but including them in the slides can show how the model becomes a **practical decision-support tool**, not just an abstract forecaster.

---

30. What minimum improvement over a baseline is "meaningful"?

Rewritten question: Relative to a simple baseline, how much metric improvement do we need to justify extra model complexity?

Answer: We do not have official baseline scores, but we can define our own internal standard:

- **Simple baselines** could be:
  - Naive flat forecast (e.g., repeat last pre-LoE volume),
  - Average erosion curve per bucket or per therapeutic area,
  - Simple linear trend extrapolation.
- For a Datathon like this, a **meaningful improvement** is typically:
  - Around **10–20% reduction** in the official PE metrics (Metric 1 and 2) vs a naive baseline.

Practical rule for us:

- If a more complex model (extra features, heavy tuning, or ensembles) does **not** give at least ~10–15% improvement in **both**:
  - our **local validation metric**, and
  - the **public leaderboard score** (on average over a few attempts),
- then its additional complexity is probably **not justified** under time pressure.

D. Data source, structure, and semantics (31–40)

31. What is the origin and nature of the data?

Rewritten question: What kind of dataset are we working with, and what does it describe in the real world?

Answer: We are working with a **pharmaceutical commercial dataset** that:

- Tracks **monthly units sold** (`volume`) for branded drugs.
- Is organized by:
  - **Country** (`country`),
  - **Brand** (`brand_name`),
  - **Time** (calendar `month` and relative `months_postgx`).
- Focuses specifically on the period **before and after the first entry of generics** (“generic entry” or `gx`).

The goal is to model the **evolution of sales volume around the patent expiry / generic entry event**.

---

32. What is the row-level granularity?

Rewritten question: What does each row in the main volume dataset represent?

Answer: Each row in `df_volume_*` represents:

- A single monthly observation for a specific **(country, brand)** pair, with:
  - `country` – the market,

- `brand_name` – the branded drug,
- `month` – calendar month,
- `months_postgx` – how many months before/after generic entry:
  - Negative values: months **before** generic entry,
  - 0: **month of generic entry**,
  - Positive values: months **after** generic entry.
- `volume` – units sold in that month.

So, the granularity is clearly: **one row = one brand in one country in one month**.

---

33. How do the three datasets relate?

Rewritten question: How are `df_volume`, `df_generics`, and `df_medicine_info` connected to each other?

Answer:

1. **Sales Volume dataset (`df_volume_train.csv`, `df_volume_test.csv`)**
   - Fact table with target:
     - `country`, `brand_name`, `month`, `months_postgx`, `volume`.
2. **Generics dataset (`df_generics_train.csv`, `df_generics_test.csv`)**
   - Time-varying competitive context:
     - `country`, `brand_name`, `months_postgx`, `n_gxs`.
   - For each `(country, brand_name, months_postgx)` we know **how many generics** are on the market.
3. **Drug characteristics dataset (`df_medicine_info_train.csv`, `df_medicine_info_test.csv`)**
   - Static product attributes:
     - `country`, `brand_name`,
     - `ther_area` (therapeutic area),
     - `hospital_rate` (% units via hospitals),
     - `main_package` (EYE DROP, PILL, INJECTION, etc.),
     - `biological` (True/False),
     - `small_molecule` (True/False).

Key relationships:

- The **primary key** for time-series rows is:
  - `(country, brand_name, months_postgx)`.

- We can join:
  - `df_volume` ↔ `df_generics` on `(country, brand_name, months_postgx)`,
  - `df_volume` ↔ `df_medicine_info` on `(country, brand_name)`.

Thus, the modeling table is effectively built by **joining these three sources** on those keys.

---

34. What are the temporal fields and how do they work?

Rewritten question: What time-related fields do we have, and how should we interpret/use them?

Answer:

We have two key temporal elements:

1. `month` (calendar month)
   - Example: `Jul`, `Aug`, `Sept` etc. (possibly with year in full dataset).
   - Represents the actual calendar time.
   - Useful for:
     - **Seasonality** (e.g., month-of-year effects),
     - Aligning with external events (if allowed).
2. `months_postgx`
   - A **relative time index** centered on generic entry:
     - `-24 … -1` → 24 to 1 months **before** entry,
     - `0` → month of **generic entry**,
     - `1 … 24` → 1 to 24 months **after** entry.
   - Used to:
     - Align all brands around the **same "time zero" event** (generic entry),
     - Compute metrics and scenarios:
       - Scenario 1: months 0–23,
       - Scenario 2: months 6–23 (knowing 0–5).

For modeling:

- At prediction time:
  - For Scenario 1 (month 0), we can only use features built from **months with** `months_postgx` **< 0** and **static info**.
  - For Scenario 2 (month 6+), we can also use actual volumes and generics info up to `months_postgx = 5`.

---

35. Which identifiers exist and how will we use them?

Rewritten question: What ID fields do we have, and are they just for grouping or also features?

Answer:

Main identifiers:

- `country`:
  - Identifies the **market**,
  - Used as:
    - Group key for times series,
    - Potential categorical feature (encoded via one-hot, target encoding, or similar),
    - Dimension for reporting and visualizations.
- `brand_name`:
  - Identifies the **branded product** within a country,
  - Used mainly as:
    - Group key for each time series,
    - Key to join with `df_medicine_info` and `df_generics`,
    - Typically **not** directly used as a feature, but could be encoded if helpful (at risk of overfitting).

In practice, we will:

- Treat `(country, brand_name)` as the **series identifier**.
- Use `country` as a feature if we want to capture **systematic country-level differences**.
- Keep `brand_name` mostly for grouping and merging, not as a raw feature (unless we have a robust encoding strategy).

---

36. How big is the dataset and what does that mean for modeling?

Rewritten question: Roughly how many time series and rows do we have in train and test, and does this limit model choices?

Answer:

From the brief:

- **Training set**:
  - **1,953** `(country, brand)` series.
  - Each has **up to** 24 months before and 24 months after generic entry:
    - Maximum ~48 rows per series.

  - So order of magnitude:
    - Raw training rows ≈ 1,953 × (up to ~48) ≈ 90k rows (roughly).
- **Test set**:
  - **340** series:
    - 228 Scenario 1,
    - 112 Scenario 2.
  - Each with relevant pre/post data for the scenario.

Implications:

- Data size is **moderate**, not huge.
- This comfortably supports:
  - Gradient-boosted trees,
  - Panel TS models,
  - Even some fairly flexible models without massive compute cost.
- We can afford:
  - Multiple validation splits,
  - Reasonable hyperparameter tuning,
  - Without hitting serious scalability issues.

---

37. What are the data types of each column?

Rewritten question: How are columns typed (numeric, categorical, boolean, datetime), and are any tricky?

Answer:

Based on the samples:

- **Numeric:**
  - `volume` (float),
  - `months_postgx` (integer),
  - `n_gxs` (float; conceptually an integer count),
  - `hospital_rate` (float percentage).
- **Categorical (string):**
  - `country` (e.g., `COUNTRY_0024`),
  - `brand_name` (e.g., `BRAND_1143`),
  - `ther_area` (e.g., `Nervous_system` , `Antineoplastic_and_immunology`),
  - `main_package` (e.g., `PILL` , `INJECTION` , `EYE DROP` , `Others`).
- **Boolean:**

  - `biological` (True/False),
  - `small_molecule` (True/False).

- **Temporal:**

  - `month` (string in the snippet, but in full data likely a datetime or a string representing a month/year).

Potential issues:

- Some missing values (e.g., `hospital_rate` is blank for some rows).
- `n_gxs` is stored as float (e.g., `4.0`), but semantically is a **count**.

We should:

- Cast types properly in code,
- Impute or handle missing values,
- Ensure `month` is treated either as true datetime or transformed into meaningful time features (month index, month-of-year, etc.).

---

38. Are there metadata columns we should ignore?

Rewritten question: Are there any purely technical columns we should drop when modeling?

Answer:

From the provided fragments, the CSVs look fairly clean:

- Columns shown:

  - `country`, `brand_name`, `months_postgx`, `n_gxs`,
  - `ther_area`, `hospital_rate`, `main_package`, `biological`, `small_molecule`,
  - `month`, `volume`.

We don’t see:

- Row indices,
- Internal IDs,
- File names, etc.

If in the actual files we see columns like:

- `Unnamed: 0`,
- `index`,
- or any constant fields,

these should be **ignored** in modeling. But based on the description, the main distributed files are already curated and don’t include obvious metadata noise.

---

39. Are there pre-engineered features that could cause leakage?

Rewritten question: Does the dataset contain any columns that already encode erosion or future information and could leak the target?

Answer:

From the description and samples, we do **not** see:

- Pre-computed erosion scores,
- Normalized volumes,
- Post-hoc classifications.

The only potentially “derived” fields are:

- `months_postgx` → relative time index (safe, needed).
- `n_gxs` → count of generics at that month (safe as long as we respect the time index).
- `hospital_rate`, `ther_area`, `main_package`, `biological`, `small_molecule` → all **static drug attributes** (safe).

The only source that could encode bucket labels and average volumes is:

- `auxiliar_metric_computation_example.csv`:
  - Contains `avg_vol` and `bucket`.
  - This file is **meant only for metric computation**, not as a feature.
  - Using `bucket` as a feature would be a form of **label leakage**, because it’s defined using post-entry behavior.

So:

- The main dataset is safe.
- We must **not use `bucket` or any erosion label derived from test sets as an input feature**.

---

40. Can we infer semantics confidently without a full data dictionary?

Rewritten question: If we don’t have a long formal data dictionary, can we still trust our understanding of each field?

Answer: Yes, we can, because:

- Column names are **descriptive**:
  - `hospital_rate` → % of units delivered via hospitals.
  - `ther_area` → therapeutic area of the drug.
  - `main_package` → main dosage form/package type.
  - `biological` and `small_molecule` → clear yes/no flags.
  - `n_gxs` → number of generic competitors.
  - `months_postgx` → months relative to generic entry.
- The Datathon brief explicitly explained:
  - Meaning of `volume`, `months_postgx`, `n_gxs`,

- Role of each dataset (`volume` , `generics` , `medicine_info`).

- In EDA, we can:

  - Check ranges and distributions (e.g., `hospital_rate` between 0 and 100),
  - Confirm that `n_gxs` increases on/after entry, etc.

As long as we:

- Use only **pre-entry information** when simulating Scenario 1,
- Respect time causality when building features,
- Avoid using metric-specific files like `auxiliar_metric_computation_example.csv` as input features,

we can confidently use the provided fields without misinterpreting their semantics.

E. Data quality, missingness, leakage, and sampling bias (41–50)

41. What is the missingness profile in our data?

Rewritten question: For each dataset and column, what does missingness look like? Which columns have a lot of missing values and may need special handling?

Answer: From the problem description and the samples:

- **`df_volume_*` (train/test)**

  - Columns: `country` , `brand_name` , `month` , `months_postgx` , `volume` .
  - We expect **very few or no missing values** in `volume` : the metric and task require observed time series around generic entry, so missing target values would undermine the setup.
  - Some series may not have the **full 24 pre + 24 post months**, but that is not NaN, it just means **shorter time series** (fewer rows).

- **`df_generics_*` (train/test)**

  - Columns: `country` , `brand_name` , `months_postgx` , `n_gxs` .

  - Conceptually, we expect:

    - Rows mostly for **months ≥ 0** (post generic entry), because before entry `n_gxs` is often 0 or not meaningful.
    - Where rows exist, `n_gxs` should usually be present (0, 1, 2, ...).

  - Any true missing `n_gxs` (NaN) would need:

    - Either **imputation** (e.g., forward-fill, backward-fill, or set to 0 if consistent), or
    - Removal if very rare and not critical.

- **`df_medicine_info_*` (train/test)**

  - Columns: `country` , `brand_name` , `ther_area` , `hospital_rate` , `main_package` , `biological` , `small_molecule` .

- From the sample:
    - `hospital_rate` is **missing for at least some brands** (blank entry).
    - We may also see:
        - Occasional missing `ther_area` or `main_package` (e.g., for edge cases),
        - `biological` / `small_molecule` should mostly be complete (True/False), but we should still check.
- This means:
    - We will need a clear **imputation strategy** for `hospital_rate` (e.g., median per therapeutic area, global median, or a “missing” flag).
    - For categorical attributes with rare missingness, we can create an **explicit “Unknown” category**.

There is no evidence of a column with **extreme missingness (>80%)** from the description, but we must confirm via EDA. If any such column appears, we would likely **drop it or use it only for analysis**, not as a core feature.

---

42. Is missingness systematic and potentially informative?

Rewritten question: Do missing values occur in specific patterns (e.g., certain months, countries, or brands) that might themselves be informative?

Answer: We expect missingness (or absence of rows) to be **structured rather than random**:

- **Generics data (`df_generics`)**
    - Likely **present only from** `months_postgx ≥ 0`:
        - Before generic entry, generics either do not exist or are irrelevant.
        - So we will see no rows (not NaNs) for negative `months_postgx`.
    - This “missing” before 0 is **structural**, not noise.
- **Short or incomplete time series**
    - Some `(country, brand)` series may not have:
        - full 24 months before entry,
        - or full 24 months after entry (e.g., if the brand disappears early).
    - This truncation is also **informative**:
        - A brand that disappears quickly after generic entry (early near-zero volumes) indicates **strong erosion** or discontinuation.
- **Drug-info missingness (e.g., `hospital_rate`)**
    - If missing values cluster:

        - in specific countries,
        - or in specific therapeutic areas,
    - that pattern may reflect genuine **documentation gaps** or differences in how products are distributed (e.g., older brands or legacy systems).

In summary, most missingness is likely **structured, not random**, and we should:

- Avoid blindly dropping rows,
- Consider whether patterns of missingness (short histories, absent generics rows, missing hospital data) signal **specific business realities** (recent launches, niche products, etc.).

---

43. Are there duplicates we need to deduplicate?

Rewritten question: Do we have duplicate rows (same `country`, `brand_name`, `months_postgx`) in the volume or generics data, and what should we do if we find them?

Answer: By design, we expect **one row per (country, brand_name, months_postgx)** in:

- `df_volume_*` (volume time series),
- `df_generics_*` (generics counts).

From the small samples, we don't see duplicates, but in real EDA we should check:

- For each dataset:
    - Count rows versus count of **unique** (`country`, `brand_name`, `months_postgx`).
    - If counts differ, identify duplicates.

If duplicates exist:

- In **volume**:
    - Multiple entries for the same series and month could represent:
        - multiple channels or partial shipments aggregated later,
        - or **data duplication errors**.
    - Our default handling:
        - **Aggregate** by summing `volume` (total units sold in that month),
        - Ensure just **one row per time step** remains.
- In **generics**:
    - Multiple rows for the same key would likely be an error.
    - We would:
        - Check consistency of `n_gxs`,
        - If consistent, keep one row,
        - If inconsistent, resolve by domain logic or averaging (but this should be rare).

We should **deduplicate early** so that subsequent joins and modeling are consistent.

---

44. Do we see impossible or suspicious values?

Rewritten question: Do we observe values that make no sense (negative volume, absurd spikes, inconsistent `n_gxs`), and how do we plan to handle them?

Answer: We need to explicitly check for:

- **Impossible volumes**:
  - Negative `volume` → should not exist for units sold.
  - Zero `volume`:
    - Could be valid (brand inactive that month),
    - But a long run of zeros before entry would be suspicious (e.g., misalignment of dates).
- **Extreme spikes**:
  - Very large isolated `volume` spikes (10× normal levels) may be:
    - Real events (tender wins, stockpiling before price changes),
    - Or data errors (duplicate loading, mis-scaled units).
- **Inconsistent `n_gxs`**:
  - `n_gxs` < 0 → impossible.
  - `n_gxs` decreasing over time in strange ways:
    - In practice, generics can **enter and exit** (e.g., market withdrawals), so some decreases may be real.
    - But very erratic patterns (e.g., 0 → 5 → 0 → 3) could suggest coding issues.

Our plan:

- Use EDA to:
  - Plot volumes over time for sample brands,
  - Summarize min/max values by brand and overall,
  - Check `n_gxs` trajectories for a few brands.
- Treat clearly impossible values as **data cleaning targets**:
  - Negative values → set to NaN and decide to drop or impute (rare) or exclude the series if too problematic.
  - Extremely large spikes → consider **capping** or **log-transforming** volumes to reduce impact, while documenting decisions.

We must be careful not to remove genuine business events, but also not to let obvious data errors distort the model.

---

45. How do we treat outliers in volume?

Rewritten question: When we see extreme volume values, are they real business events or likely noise, and what should we do with them?

Answer: Outliers in this context often have a **business interpretation**:

- Possible real events:
  - A **stockpiling** spike before expected generic entry or price increase,
  - A **tender win** or big contract,
  - A temporary **supply recovery** after a shortage,
  - A **launch** or relaunch event.
- Possible data issues:
  - Duplication of data,
  - Mis-scaled units (e.g., mixing packs vs units),
  - Reporting anomalies.

Our approach:

1. **Flag outliers statistically**:
   - e.g., volumes above a certain multiple of typical series-level mean/median.
2. **Visually inspect a small subset**:
   - Check context around the spike: does it look like part of a pattern or a one-off glitch?
3. **Treatment**:
   - If plausible business events:
     - **Keep them**, but maybe:
       - use **robust models** or log-transformations,
       - ensure they don't dominate training.
   - If likely errors:
     - **Cap** at a reasonable threshold,
     - or replace with local median/mean,
     - documenting every cleaning rule we apply.

Given this is a forecasting competition, we lean towards **conservative cleaning** (fix only clearly impossible cases) and favor modeling techniques robust to heavy tails.

---

46. Could any features leak future information?

Rewritten question: Which features risk using information from the future (relative to the prediction month), and how do we avoid this, especially for Scenario 1 vs Scenario 2?

Answer: Yes, leakage is a serious risk, especially because:

- We have **full post-entry history** in train,
- But we must simulate having **no post-entry data** (Scenario 1) or only 6 months (Scenario 2) at prediction time.

Key rules:

- **Scenario 1 (right after generic entry, month 0):**
    - At prediction time (t = 0), we must **only use**:
        - Pre-entry volumes: months with `months_postgx < 0`,
        - Static attributes (`ther_area`, `hospital_rate`, etc.),
        - Generics info up to `months_postgx ≤ 0` (e.g., whether generics are already present in month 0, if the dataset encodes that).
    - We must **not** use:
        - Any variable derived from **months_postgx > 0** (e.g., future volumes, future `n_gxs`),
        - Any "average erosion" computed using post-entry data.
- **Scenario 2 (month 6 with 6 months of post-entry actuals):**
    - At prediction time (starting at `months_postgx = 6`), we can use:
        - Volumes for `months_postgx ≤ 5`,
        - `n_gxs` for `months_postgx ≤ 5`,
        - Static drug info.
    - We must not use:
        - Any information from `months_postgx > predict_horizon` (i.e., 24),
        - Any features computed using future months beyond the forecast point.

Implementation tip:

- When building features, **truncate** each series up to the relevant cut-off (0 for Scenario 1, 6 for Scenario 2), and compute all features using **only data up to that cut-off**.

This way, our training setup **matches the real evaluation scenario**, and we avoid leakage via "peek into future months."

---

47. Are there target-like columns that must be excluded?

Rewritten question: Do we have any columns that are just aggregates or labels derived from the volume itself, which would leak target information if used as features?

Answer: In the **core train/test datasets** (`df_volume`, `df_generics`, `df_medicine_info`) we don't see any explicit target-like columns.

The main risk comes from **auxiliary / metric-related data**:

- `auxiliar_metric_computation_example.csv` (or its real equivalent) contains:
  - `avg_vol` (average pre-entry volume),
  - `bucket` (1 or 2, high vs mid/low erosion).
- These are **computed using the time series, including post-entry behavior**, and are meant **only for metric computation**.

Therefore:

- We **must not** use:
  - `bucket` as an input feature (it encodes the outcome we're trying to predict/classify),
  - `avg_vol` from the auxiliary metric file as a precomputed feature (we can recompute pre-entry averages for train series, but we must not compute them for test in a way that uses future info).

If we derive our own:

- Pre-entry averages,
- Pre-entry volatility,
- Pre-entry trend,

we must calculate them **only using pre-entry months** and treat them as legitimate features, separated from the metric-specific auxiliary file.

---

48. How were the 2,293 country–brand combinations selected, and what biases might exist?

Rewritten question: Is this dataset a complete universe of Novartis brands with generics, or a selected sample, and what sampling bias could that imply?

Answer: We don't have a full internal sampling protocol, but we can infer:

- The 2,293 observations correspond to **country–brand combinations that experienced a generic entry** and have enough data (pre- and post-entry) to compute the metric.
- This likely implies some **inclusion criteria**, e.g.:
  - Minimum pre-entry history (e.g., at least 12 months),
  - Minimum post-entry history (enough months to observe erosion),
  - Availability of generics info (`n_gxs`) and drug characteristics.

Potential biases:

- Under-representation of:
  - Very small brands with short or noisy histories,
  - Very new products or those with incomplete data,
  - Some countries with poor data coverage.
- Over-representation of:
  - Brands that are **commercially significant** (larger volumes),

  - Therapeutic areas where generics and erosion dynamics are well-studied.

For the competition:

- This is fine: the dataset represents **the part of the portfolio where erosion modeling makes sense**.
- But in any business discussion, we should note that our model is trained on a **specific subpopulation** and may not generalize to all possible brands (e.g., ultra-niche or newly launched products).

---

49. Do we see distribution shift between train and test?

Rewritten question: Are training and test distributions aligned, or do we see differences in erosion buckets, product types, or time ranges?

Answer: The organizers have explicitly tried to keep a **consistent structure** between train and test:

- The test split (340 series) is designed to:
  - Cover both **Scenario 1 and Scenario 2**,
  - Maintain the **same proportions of Bucket 1 and Bucket 2** as in the broader population.

However, we should still check for:

- **Bucket proportions**:
  - Compare distribution of `bucket` in training vs. test (if available for train; for test we may not see bucket, but we can approximate via train-like heuristics on a validation split).
- **Product mix**:
  - Therapeutic areas (`ther_area`) distribution in train vs. test.
  - Biological vs small_molecule distribution.
- **Country mix**:
  - Are some countries only in the test set? Are some low-frequency countries more prominent in test?
- **Temporal coverage**:
  - Are test time series from later or earlier calendar years than train?

We should assume that organizers **tried** to avoid severe shift, but we cannot rely on that blindly. Our internal validation strategy (using time-based splits) should mirror the **expected test conditions**, not random splits that might not represent them.

---

50. Are there macro events that break stationarity (e.g., COVID)?

Rewritten question: Could global events (like the COVID-19 pandemic) or structural changes create sudden shifts in volume over calendar time?

Answer: It is very plausible:

- The data likely spans **multiple years**, potentially including:
    - The **COVID-19 period**, with:
        - Reduced outpatient visits,
        - Elective procedure delays,
        - Changed prescription patterns.
    - Other macro events (economic crises, policy changes, pricing reforms) in different countries.

These can cause **non-stationarities**:

- Sudden drops or increases in volume around specific calendar years/months not directly related to generic entry.
- Different impacts by therapeutic area (e.g., elective vs chronic vs oncology).

Implications:

- When we do EDA, we should:
    - Plot volumes over time (using `month`) for a subset of brands,
    - Look for **global patterns** (e.g., drop in 2020 for many countries) separate from generic entry effects.
- For modeling, we might:
    - Introduce **calendar-related features** (year, month, or period indicators),
    - Or at least be ready to explain:
        - “Part of the variation might be driven by macro events not explicitly modeled.”

In our final presentation, we can acknowledge:

- The model captures erosion patterns **on top of** whatever other macro shocks are present in the data,
- It is not guaranteed to generalize to entirely new structural breaks unless retrained with new data.

F. Splitting strategy, validation design, and EDA focus (51–60)

51. What is a realistic validation strategy for this panel time-series setup?

Rewritten question: Given that we have time-series by (country, brand_name), how should we design validation to realistically simulate Scenario 1 (no post-entry data) and Scenario 2 (6 months of post-entry data)?

Answer: A realistic strategy is:

- Work **at series level** (country–brand), not row level.
- For local validation, **hold out some full series** as a “pseudo-test”:
    - For each selected validation series, you keep all its data but artificially “pretend you don’t know” post-entry volumes when building features.

- For each validation series:
  - **Scenario 1 validation:**
    - Use only data with `months_postgx < 0` (pre-entry) + static info (`ther_area` , `hospital_rate` , `biological` , etc.) to train/fit.
    - Predict `volume` for `months_postgx = 0,…,23` .
    - Compare predictions to actuals using `compute_metric1` (Metric 1), with `avg_vol` and `bucket` computed only from that series' pre-entry data.
  - **Scenario 2 validation:**
    - Now assume months 0–5 post-entry are known.
    - Use `months_postgx <= 5` (plus pre-entry) as features.
    - Predict `volume` for `months_postgx = 6,…,23` .
    - Evaluate with `compute_metric2` (Metric 2).

This approach:

- Respects the **temporal direction** (we never use future months to predict past),
- Directly mirrors the **competition evaluation** (scenario structure + metrics),
- Lets us tune and select models based on the same logic as the leaderboard.

---

52. Why should we avoid random row-level splits?

Rewritten question: Why is a random split of rows (mixing different months of the same series into train and validation) a bad idea here?

Answer: Random row-level splits would:

- Put **past and future months of the same (country, brand)** into both train and validation,
- Let the model see **post-entry behavior** in training while we are supposedly validating "as of" month 0 or month 6,
- Produce **artificially optimistic metrics** because future patterns leak backward.

In practice:

- The model would learn from volumes at `months_postgx > 0` when validating "Scenario 1" (which in reality has no post-entry data), which is exactly the kind of **leakage** we must avoid.

Therefore:

- We must **not** use simple random K-fold on individual rows.
- Splits must respect:
  - **Time** (train on earlier part, validate on later),
  - **Group identity** (country–brand), especially when simulating the two scenarios.

---

53. Do we need some stratification in validation?

Rewritten question: Should we ensure that our validation set has a representative mix of erosion buckets and product types?

Answer: Yes, some **light stratification** is very useful.

Key aspects:

- The business and the metric both **emphasize Bucket 1** (high erosion) and also use Bucket-1 vs Bucket-2 weighting.
- If our validation series accidentally contained mostly low-erosion brands, we would underestimate how hard the competition really is.

Practical approach:

- When selecting a subset of country–brand series for validation:
  - Ensure a **balanced mix of buckets**:
    - e.g., validation share of Bucket 1 ≈ training share of Bucket 1.
  - Also try to cover:
    - A range of **therapeutic areas**,
    - Both **biological** and **small_molecule** products,
    - Diverse `hospital_rate` profiles (hospital-heavy vs retail-heavy).
- Technically:
  - We can pre-compute `bucket` and some category stats on the **train** data and then sample validation series **stratified by bucket + perhaps therapeutic area**.

This makes our local validation more representative of what the **hidden test set** and final business use cases will look like.

---

54. Is a rolling time-series split better than standard K-fold?

Rewritten question: Should we use rolling time-based splits instead of standard K-fold to capture temporal dependencies?

Answer: For this challenge, we don't need "multi-origin" rolling splits inside each series as in classic time-series forecasting, because:

- The **natural origin** for the forecasting task is fixed:
  - Scenario 1: origin at `months_postgx = 0`,
  - Scenario 2: origin at `months_postgx = 6`.

What we do need is:

- **Time-aware series selection**:
  - Train models on many series across all months (with appropriate feature windows),

  - Then, for validation series, ensure we only use data up to the relevant cut-off (0 or 6) to simulate each scenario.

So:

- **Standard K-fold over series** (not rows), with each fold:
  - Choosing a subset of **country–brand** pairs as validation,
  - Respecting the cut-offs for 0/6 months,
- Is usually enough and more practical than complex rolling window schemes.

If time and compute allow, we can adopt a **2–3 fold series-level CV** (different sets of brands as validation folds), but always with **temporal truncation** inside each series to avoid leakage.

---

55. How many validation folds can we afford?

Rewritten question: Given the limited time and hardware, how many validation splits are realistic?

Answer: Given:

- It's a **weekend Datathon**,
- We may iterate multiple times on:
  - Feature engineering,
  - Models,
  - Hyperparameters,

a pragmatic choice is:

- **1 strong hold-out split** + optionally **1 additional fold**.

Concretely:

1. **Main validation split**
   - Select ~20–30% of country–brand series as "validation series" (stratified by bucket, etc.),
   - Use this split consistently to compare different model versions.
2. **Optional second split**
   - After we have a "final candidate model," re-check on another small subset of series for robustness.

More folds (e.g., 5-fold CV) might be ideal in theory, but in practice:

- They are **expensive** in time/compute,
- Hard to manage under the Datathon deadline,
- Offer diminishing returns versus doing **good EDA + 1–2 well-designed validation splits**.

---

56. What target distribution comparisons should we do in EDA?

Rewritten question: When comparing train, validation, and (where possible) test-like splits, what aspects of the target distribution should we inspect?

Answer: For the target (`volume` and derived erosion measures), we should:

- **Across train vs validation (and possibly a pseudo-test split)**:
  - Compare **basic stats**:
    - Mean, median, standard deviation,
    - Key quantiles (e.g., 10th, 50th, 90th) of volume.
  - Inspect **distribution of pre-entry average volume** (avg_vol) for both Bucket 1 and Bucket 2.
  - Look at **mean normalized erosion** across buckets in our train/validation splits.
- **By key segments**:
  - Compare distributions by:
    - `ther_area` ,
    - `biological` vs `small_molecule` ,
    - high vs low `hospital_rate` .
- **Check for alignment**:
  - Ensure validation's volume ranges and erosion patterns look **similar** to the training population.
  - If the validation series are systematically smaller or larger, we interpret metrics accordingly.

This helps us verify:

- Our splits are **representative**,
- Our models are not calibrated on a dataset that looks very different from what they will see in evaluation.

---

57. Which univariate plots are most useful initially?

Rewritten question: What simple one-variable plots will quickly give us a sense of the data?

Answer: Useful univariate plots include:

- **For the target and erosion:**
  - Histogram of **pre-entry average volume** (avg_vol),
  - Histogram of **normalized post-entry volume** (e.g., volume / pre-entry avg),
  - Histogram of **mean normalized erosion** over 24 months (for Bucket 1 vs Bucket 2 separately).
- **For generics:**
  - Histogram of `n_gxs` overall and for selected months (`months_postgx` ranges),
  - Distribution of `months_postgx` values in train vs validation.
- **For drug characteristics:**

- Histogram of `hospital_rate`,
- Bar chart of `ther_area` frequencies,
- Counts of `biological` vs `small_molecule`,
- Distribution of `main_package` (PILL, INJECTION, EYE DROP, etc.).

These quick visualizations give us:

- The **scale and skewness** of volumes,
- How common different **product types** and **therapeutic areas** are,
- How generics typically evolve in count.

---

58. Which bivariate or temporal plots will teach us the most?

Rewritten question: What two-variable or time-series plots are most informative to understand erosion behavior?

Answer: Key bivariate/temporal plots:

- **Erosion curves:**
  - Plot `volume` vs `months_postgx` for:
    - Representative **Bucket 1** series (sharp erosion),
    - Representative **Bucket 2** series (moderate/low erosion),
  - Optionally normalized by pre-entry average volume to compare shapes.
- **Number of generics vs erosion:**
  - Plot `volume` and `n_gxs` over `months_postgx` on the same chart (dual axis) for selected brands,
  - Investigate whether volume drops more strongly as `n_gxs` increases.
- **Segmented erosion:**
  - Plot average normalized volume vs `months_postgx` grouped by:
    - `ther_area`,
    - `biological` vs `small_molecule`,
    - High vs low `hospital_rate`.

These plots allow us to:

- Visually separate **classic erosion patterns**,
- See how **product type, therapeutic area, and generic competition** shape the erosion curves,
- Identify any **non-intuitive behaviors** early.

---

59. Do we see non-linearities and interactions suggesting model choices?

Rewritten question: From EDA, do we expect simple linear models to be enough, or do we see interactions and non-linear patterns that call for tree-based or segmented models?

Answer: We can anticipate several **non-linear and interaction** effects:

- **Biological vs small_molecule:**
  - Erosion for **biologics** is often slower and less complete than for small molecules due to biosimilar dynamics and regulation.
- **Hospital_rate:**
  - Brands heavily used in hospitals may show **different erosion shapes** (e.g., tender-driven, stepwise) vs pure retail brands.
- **Therapeutic_area:**
  - Oncology vs cardiovascular vs CNS may have **distinct erosion profiles** due to replacement options, clinical guidelines, and payer behavior.
- **n_gxs trajectory:**
  - The effect of the 1st, 2nd, 3rd generic may not be linear:
    - The first few entrants could cause a **disproportionate drop**,
    - Additional generics may have diminishing incremental impact.

These patterns suggest:

- Tree-based models (e.g. LightGBM, XGBoost, CatBoost) or models that can handle interactions are **strong candidates**.
- We may also consider **segmented approaches**, such as:
  - Separate models for Bucket 1 vs Bucket 2,
  - or for biologics vs small molecules.

Linear baselines are still useful for sanity checks, but likely **not sufficient** to capture all important interactions.

---

60. Which features look most promising after initial EDA?

Rewritten question: After a first pass of EDA, which 5–10 features or transformations should we focus on, and which seem low-value or noisy?

Answer: Promising features / transformations:

1. **Pre-entry average volume (avg_vol)**
   - Over the last 12 months before `months_postgx = 0`; a key scale parameter and used in metric normalization.
2. **Pre-entry trend / slope**
   - Linear trend of volume over pre-entry months (growth vs decline before generics).
3. **Lag features on volume**

   - Recent volumes: `volume` at `t-1`, `t-2`, `t-3` (in `months_postgx` terms),
   - Possibly rolling averages (3, 6, 12 months).

4. **Time since generic entry**

   - `months_postgx` itself (and, if helpful, squared or piecewise, because erosion shape is not linear over time).

5. **Generics-related features**

   - `n_gxs` at month t,
   - Cumulative max `n_gxs` up to t,
   - Indicators like "first generic just entered" (e.g. `n_gxs` changes from 0 to >0).

6. **Drug characteristics**

   - `ther_area`,
   - `biological` vs `small_molecule`,
   - `hospital_rate` (possibly binned into low/medium/high),
   - `main_package`.

7. **Seasonality / calendar features** (if relevant)

   - Month of year from `month` (to capture seasonal prescribing patterns).

Features likely to be low-value or noisy:

- Very fine-grained or rare categories in `main_package` or `ther_area` (unless we aggregate categories),

- Any raw identifiers (`brand_name` as text, country string) without appropriate encoding; better to use them as:

  - Categorical features with proper encoding,
  - Or to derive higher-level groupings.

The core idea is to build a feature set that:

- Captures **pre-entry level and trend**,
- Encodes **generic competition dynamics**,
- Includes **product type context**,
- And respects the **information availability** constraints for Scenario 1 and 2.

Great, let's design the feature side clearly. For each point I'll restate the question in plain form and then answer it for *this* Datathon.

G. Feature engineering, preprocessing, and transformations (61–70)

61. How do we handle missing values in these datasets?

Rewritten question: Given our three main tables (volume, generics, medicine info), how should we treat missing values for volume, number_of_gx, and static categorical/numeric features?

Answer:

- **Volume (`volume` in df_volume_*)**
  - In practice, the dataset comes as **explicit rows with numeric volume**; a "missing month" usually means **no row**, not `NaN`.
  - If we do encounter `NaN` volume inside the table (rare):
    - Do **not** automatically set it to zero, because zero = "no units sold" (strong business meaning).
    - Prefer:
      - Either **drop** that single row, or
      - Impute with a **local time-based estimate** (e.g. mean of neighboring months) and add a **flag** (`volume_imputed = 1`).
  - In most cases, volume is complete, so no heavy imputation strategy is needed.
- **Number of generics (`n_gxs` in df_generics_*)**
  - `0` is a perfectly valid value = "no generic on the market yet".
  - If we see missing (`NaN`) in `n_gxs` for months where the brand clearly exists:
    - Treat missing as:
      - **Forward-fill** from the closest previous month if available, because the generic count usually changes piecewise in time.
      - If no previous data (e.g. very early months), we can **safely interpret missing as 0**, plus a flag (`n_gxs_missing_flag = 1`).
  - This preserves the business meaning: absence of known generics vs real competition.
- **Static / categorical features (in df_medicine_info_*)**
  - For fields like `ther_area` , `main_package` :
    - Use a special **"Missing" category** if NaNs appear.
  - For `hospital_rate` :
    - If missing, impute with a **reasonable statistic within the same therapeutic_area** (e.g. median hospital_rate for that ther_area),
    - And add a **flag** (`hospital_rate_missing = 1`).
  - For `biological` , `small_molecule` :
    - Normally these should be True/False. If missing, treat as an **explicit category** (e.g. `biological_unknown`) or impute based on product context if clearly deducible.

Overall, the rule is:

- Don't silently drop important series.
- When we impute, we **flag** it so the model can learn that this data point was uncertain.

---

62. Do we need to transform skewed numeric features?

Rewritten question: Are variables like volume, avg_vol, and number_of_gx skewed enough to benefit from log or other transformations?

Answer:

- **Volume & avg_vol**:
  - These are typically **right-skewed**: a few huge brands, many smaller ones.
  - For modeling:
    - It is often useful to work with **log-transformed** targets or features, e.g. `log1p(volume)` or `log1p(avg_vol)` .
    - This stabilizes variance across large and small brands and helps some models (especially linear ones).
  - For metric computation:
    - We must **keep the original scale** (volume in units), because the official metric uses raw volume and avg_vol—so we transform only inside the model, then invert the transform for predictions.
- **n_gxs (number_of_gx)**:
  - Takes small integer values (0, 1, 2, ... up to a modest max).
  - Usually **does not require log-transform**; we can keep it as integer.
  - We might still use:
    - Indicators: `has_generic` (n_gxs > 0),
    - or thresholds: `n_gxs >= 3` etc.
- **hospital_rate**:
  - A rate (often between 0 and 100) and can be skewed (e.g. many near 0, some near 100).
  - We can:
    - Use it as-is for tree-based models, or
    - Bin it (see Q68) for interpretability.
- **Scaling/standardization**:
  - For **tree-based models**, scaling is not necessary.

    - For **linear/NN models**, we may standardize log-transformed volume/avg_vol; but this is optional if we focus on GBM-style models.

---

63. How do we encode categorical variables?

Rewritten question: What encoding strategies should we use for country, brand_name, therapeutic_area, main_package, etc.?

Answer:

- **High-cardinality IDs: `country`, `brand_name`**
    - They are primarily **series identifiers**.
    - For global models:
        - We can use **target encoding / frequency encoding** or let models like CatBoost treat them as categorical directly.
        - Or we may avoid using `brand_name` as a feature and rely instead on:
            - Its **aggregated properties** (avg_vol, pre-entry trend, erosion category).
    - For simpler and safer modeling:
        - Use `country` as a categorical feature (one-hot or CatBoost category),
        - Treat `brand_name` mostly as a **group key**, not as a direct predictor (or apply careful target encoding to avoid leakage).
- **Domain-level categories: `ther_area`, `main_package`**
    - These have **manageable cardinality**.
    - Encoding options:
        - **One-hot** encoding (for tree or linear models),
        - Or pass them as **categorical features** to CatBoost / LightGBM with categorical support.
- **Booleans: `biological`, `small_molecule`**
    - Keep as **0/1** or bool type.
    - For CatBoost, we can keep them as numeric (0/1) or categorical; both work.

In summary:

- For a GBM approach:
    - Use **categorical support** where available,
    - Otherwise one-hot encode `ther_area`, `main_package`, and maybe `country`,
    - Use `brand_name` mostly as an ID for grouping / time-series, not a direct feature unless carefully encoded.

---

64. Which interaction features and ratios are likely to help?

Rewritten question: What derived ratios or interactions should we engineer to capture erosion and context?

Answer:

Useful derived features include:

1. **Normalized volume (erosion-style)**
   - `norm_volume_t = volume_t / avg_vol_pre_entry`
   - This matches the metric's logic and helps the model learn erosion behavior independent of brand scale.
2. **Hospital focus vs retail**
   - Use `hospital_rate` directly and/or as bins (see Q68).
   - Interaction ideas:
     - `norm_volume_t × hospital_rate_bin`, to capture that heavy hospital brands may erode differently.
3. **Bucket-aware interactions (for EDA / explanation)**
   - While `bucket` is defined from post-entry data and should not be used as an input feature, we can:
     - Analyze interactions by bucket (e.g. mean erosion curve by ther_area within Bucket 1 vs Bucket 2),
     - Use that to decide whether to **train separate models** or adjust hyperparameters.
4. **Generic competition interactions**
   - `norm_volume_t × n_gxs`,
   - `has_generic = (n_gxs > 0)`,
   - Step indicators: e.g. `first_generic_month_flag`, `second_generic_or_more_flag`.
5. **Country-level context**
   - If desired, simple encodings of `country` combined with erosion, e.g. mean erosion by country (computed only from train and used cautiously).

These features help the model capture:

- How erosion dynamics differ by **brand scale**, **hospital vs retail mix**, and **generic competition intensity**.

---

65. What time-series features should we derive?

Rewritten question: Which temporal features (lags, trends, time indices) should we build from volume and generics?

Answer:

Core time-series features:

1. **Lags of volume**
    - Pre-entry:
        - For Scenario 1, use lags from `months_postgx = -1, -2, …` as allowed by the history.
    - Early post-entry (for Scenario 2):
        - For predictions at `months_postgx ≥ 6`, we can use `volume` at months 0–5 as features.
    - Typical choice: lags 1, 2, 3 and maybe 6 months.
2. **Rolling statistics over pre-entry**
    - Rolling mean and std of volume over:
        - last 3 months,
        - last 6 or 12 months.
    - Simple linear **trend/slope** over the last 6–12 pre-entry months.
3. **Time since generic entry**
    - `months_postgx` itself is an important driver:
        - Negative values (pre-entry), 0 at entry, positive afterward.
    - We might use:
        - `months_postgx` directly, and optionally:
        - piecewise features (e.g. indicators for "0–5", "6–11", "12–23").
4. **Generic competition dynamics over time**
    - At each month t:
        - Current `n_gxs_t`,
        - Cumulative max `max_n_gxs_up_to_t`,
        - Month index at which the first generic appeared.
5. **Calendar month or seasonality**
    - From `month` (calendar), extract **month of year** (1–12).
    - This can capture seasonal patterns (e.g., certain therapeutic areas have seasonal usage).

All these features must respect:

- Scenario 1: use only **pre-entry months** to construct features,
- Scenario 2: also include months 0–5 in the feature window, but **never use future months** beyond the prediction horizon.

66. What if we had free text fields?

Rewritten question: If we had free text like long product descriptions, how would we use them under time constraints?

Answer:

In the provided specification, we **do not** see free-text fields; all key inputs are structured: country, brand_name, ther_area, etc.

If some additional free-text field exists (e.g., "product_description"):

- Given the Datathon time limit, the simplest approach would be:
  - Either **ignore it**, or
  - Map it to a few **hand-crafted categories** if obvious patterns exist.
- We should **avoid heavy NLP** (embeddings, transformers) unless:
  - We're sure it brings clear value,
  - And we have enough time and compute to integrate it correctly.

So practically: ignore free-text fields or reduce them to **very simple tags**, focusing our effort on structured features that directly drive the metric.

---

67. How do we engineer features from DF_Generics?

Rewritten question: What aggregations and transformations over time should we build from `n_gxs` ?

Answer:

From the DF_Generics table (country, brand_name, months_postgx, n_gxs), we can derive:

1. **Current generic competition intensity at month t**
   - `n_gxs_t` itself.
2. **Cumulative maximum up to month t**
   - `max_n_gxs_up_to_t` = max of `n_gxs` for months ≤ t.
   - Captures whether the brand has ever faced intense competition.
3. **First generic entry timing**
   - `first_generic_month` = smallest `months_postgx` where `n_gxs > 0`.
   - For Scenario 1, this will usually be 0 (they define 0 as the month of entry),
   - But it's still useful to identify early vs delayed arrivals if anomalies exist.
4. **Speed of generic build-up**
   - Time from `n_gxs = 0` to first time `n_gxs ≥ k` (e.g. ≥ 2 or ≥ 3).
   - Or simply the **change in generics** over early post-entry months (for Scenario 2).

5. **Binary indicators**

    - `has_generic_t = (n_gxs_t > 0)`,
    - `multiple_generics_t = (n_gxs_t >= 2)`.

At prediction time:

- For Scenario 1 at t = 0:
    - `n_gxs` will be 0 or a small value; features using future `n_gxs` must **not** be used.
- For Scenario 2 at t ≥ 6:
    - We can use the **history of n_gxs up to month 5** (and maybe up to current t if the design allows incremental predictions), but not beyond the prediction horizon.

---

68. Which features should we bin?

Rewritten question: Are there numeric features we should discretize into bins for robustness and interpretability?

Answer:

Yes, some variables are well-suited to binning:

- **hospital_rate** (0–100)
    - We can define bins such as:
        - 0–10% → "mostly retail",
        - 10–50% → "mixed channel",
        - 50–100% → "mostly hospital".
    - This helps:
        - Simplify patterns for the model,
        - Explain results to business stakeholders ("hospital-heavy brands behave like this…").
- **Pre-entry average volume (avg_vol)**
    - Bins like:
        - Small brands,
        - Medium brands,
        - Large brands.
    - These categories provide clear business interpretation: "large brands with high hospital_rate and high n_gxs erode more/less".
- **n_gxs**
    - Instead of using raw counts only, we can add:

        - 0 generics,
        - 1 generic,
        - 2+ generics.

Binning:

- Can reduce overfitting,
- Produces **more stable, explainable segments** for the slides and discussions,
- Is particularly useful if we plan to show **summary tables** (“Bucket 1 brands with high hospital_rate tend to follow pattern X”).

---

69. How do we avoid target leakage in features?

Rewritten question: How do we guarantee that our features don’t accidentally use future information (especially for Scenario 1 and 2)?

Answer:

We enforce **strict time boundaries**:

- **Scenario 1 (right after generic entry)**:
    - At prediction time (month 0), we can only use:
        - Pre-entry volume (`months_postgx < 0`),
        - Static features (ther_area, biological, etc.),
        - Any generic info up to month 0, but typically 0 generics before that.
    - We must **not** use:
        - Any volume or n_gxs for `months_postgx > 0` .
- **Scenario 2 (6 months after entry)**:
    - At prediction time (month 6), we can use:
        - All pre-entry data (`months_postgx < 0`),
        - Months 0–5 volume and `n_gxs` ,
        - Static features.
    - We must **not** use:
        - Volume or `n_gxs` for months > 5 when predicting months 6–23.

Implementation-wise:

- When building features, we explicitly:
    - Filter rows by `months_postgx` up to the scenario-specific cut-off,
    - Compute lags, rolling stats, and generic dynamics **only within that window**.
- We also **exclude** any derived variables that were computed over the full 24-month horizon (e.g., final mean erosion, bucket) from model inputs—they are used only for analysis and metrics, not training.

70. How do we make the feature pipeline reproducible?

Rewritten question: How can we implement feature engineering as a clean, reusable pipeline that we can apply consistently to train, validation, and test?

Answer:

We should structure our code around **scenario-aware feature functions**, such as:

- `prepare_base_panel()` :
  - Joins:
    - `df_volume_*` ,
    - `df_generics_*` ,
    - `df_medicine_info_*` ,
  - Ensures we have a unified panel keyed by `(country, brand_name, months_postgx)` with all raw columns.
- `make_features(panel_df, scenario)` :
  - Inputs:
    - `panel_df` (full joined data),
    - `scenario` ∈ {"scenario1", "scenario2"}.
  - Inside:
    - Filters rows to keep only the **allowed history** per scenario,
    - Computes:
      - lags, rolling stats,
      - normalized volume,
      - generic dynamics features,
      - binned features (hospital_rate, avg_vol categories),
    - Applies missing value treatment and encoding.
- `make_train_val_split()` :
  - Takes the full panel,
  - Selects which series are train vs validation,
  - Applies `make_features` consistently to both, respecting each scenario’s rules.
- `prepare_submission_features()` :
  - Takes the **test** panel,
  - Applies the same `make_features(...)` logic,
  - Ensures columns and preprocessing steps exactly match those used in training.

We keep:

- All parameters (e.g. which lags, which bins, etc.) **centralized** (e.g. a config dict),
- So that training, validation, and final submission are **fully aligned**.

This way:

- Our pipeline is **reproducible** (same steps every time),
- Easy to re-run if we change models,
- And easy to explain to the jury: “We have a single feature function that is applied identically to train, validation, and test, with scenario-specific cut-offs for information availability.”

H. Model selection, training, hyperparameters, and experimentation (71–80)

71. What is our simplest baseline for each scenario, and can we score it with the official metric?

Rewritten question: What very simple forecasts will we use as baselines for Scenario 1 and Scenario 2, and can we evaluate them with `metric_calculation.py`?

Answer:

- **Scenario 1 (0–23 months, no post-entry data)** Baseline idea:

  1. Compute **pre-entry average volume** per (country, brand):

     - `avg_pre = mean(volume) over months_postgx in [-12, -1]` (or all available pre-entry months if fewer).

  2. Predict **constant volume** for all post-entry months:

     - For months_postgx = 0…23, set `volume_pred = avg_pre`.

  3. This is equivalent to saying “no erosion; the brand continues at its recent average level.”

  Optional slightly richer baseline (if we have time):

  - Compute a **global average erosion curve** in normalized terms `E[m] = mean( volume_t / avg_pre ) over all brands for each months_postgx = m`, then for each brand predict: `volume_pred(m) = avg_pre_brand × E[m]`.

- **Scenario 2 (6–23 months, with 0–5 actuals known)** Baseline idea:

  1. Fit a **simple linear trend** on months_postgx = 0…5 for each series:

     - Regress `volume` on `months_postgx` in `[0, 5]`.

  2. Extrapolate this line to months 6–23.

  3. If data is too noisy, fall back to:

     - **Last observed value baseline**: `volume_pred(m) = volume at month 5` for m ≥ 6.

- **Scoring baselines**

- We can run these baselines on a **local train/validation split**:
    - Use part of train as “fake test”, generate predictions,
    - Feed them into `compute_metric1` (Scenario 1) and `compute_metric2` (Scenario 2) from `metric_calculation.py`.
- These scores become our **reference floor**; any serious model must clearly beat them.

---

72. What is our primary “hero model”, and why is it a good fit?

Rewritten question: Which main model family will we rely on, and why does it fit this structured Datathon problem?

Answer:

Our **hero model** will be a **gradient boosting decision tree** model on engineered panel features, e.g.:

- **CatBoost**, **LightGBM**, or **XGBoost**, trained on a table where each row is:
    - (country, brand_name, months_postgx) with:
        - Target: `volume`,
        - Features: lags, rolling stats, normalized volume, n_gxs dynamics, ther_area, hospital_rate, etc.

Why this is a good fit:

- The data is **medium-sized structured tabular**:
    - ~2k series × up to 48 months → on the order of **tens of thousands of rows**, not millions.
- We have **mixed feature types**:
    - Numeric (volume, n_gxs, hospital_rate), categorical (ther_area, main_package, country), booleans (biological, small_molecule).
- Gradient boosting:
    - Captures **non-linearities** and interactions (e.g. erosion vs number_of_gx × ther_area),
    - Trains **fast enough** to run many experiments over a weekend,
    - Handles missing values reasonably well.

We will likely train **separate models per scenario**:

- One model optimized to predict volumes for Scenario 1 (given only pre-entry info),
- One model optimized for Scenario 2 (using pre-entry + 0–5 months).

---

73. Which hyperparameters matter most for our hero model in this context?

Rewritten question: For a gradient boosting model on this data, which hyperparameters should we focus on tuning?

Answer:

Most influential hyperparameters (conceptually, independent of specific library):

- **Model capacity / complexity**
    - `max_depth` (or `depth` in CatBoost): controls tree depth.
    - `num_leaves` (in LightGBM): controls leaf count, strongly tied to complexity.
    - `min_data_in_leaf` / `min_child_samples`: prevents tiny leaves and overfitting.
- **Learning dynamics**
    - `learning_rate`: smaller = more stable but requires more trees.
    - `n_estimators` / `iterations`: number of trees; combined with early stopping.
- **Regularization**
    - `lambda_l1`, `lambda_l2` (L1/L2 penalties),
    - `min_split_gain` / `min_gain_to_split` (minimum gain threshold),
    - `max_bin` (for LightGBM) if using many continuous features.
- **Stochasticity / robustness**
    - `feature_fraction` / `colsample_bytree`: random subset of features per tree.
    - `bagging_fraction` / `subsample`: random subset of rows per iteration.
    - `bagging_freq`: how often subsampling is applied.

Given we have a **limited number of series**, we must balance:

- Enough flexibility to model **different erosion patterns**,
- Strong regularization and subsampling to **avoid overfitting** to a few large brands.

---

74. What tuning strategy is realistic during the Datathon?

Rewritten question: How should we tune hyperparameters without wasting time on heavy optimization?

Answer:

We will use a **lightweight, pragmatic tuning strategy**:

1. Start from **sane defaults**:
    - Moderate depth (e.g., 4–7),
    - Learning rate around 0.03–0.1,
    - Reasonable regularization.
2. For each scenario (1 and 2):
    - Use a **single time-based validation split** (or at most 2 splits).
    - Perform:
        - A **small random search** over a short list of candidate values for:

            - depth / max_depth,
            - num_leaves or equivalent,
            - learning_rate,
            - L2 regularization,
            - subsampling parameters.

        - Or a **tiny grid search** (like 2–3 values per important parameter).

3. Use **early stopping** on the official metric (or a close proxy) computed on the validation set.

We **avoid**:

- Heavy Bayesian optimization,
- Dozens of long-running configurations.

Goal:

- Get to a **strong, robust configuration** quickly,
- Spend time on **feature engineering, validation quality, and analysis**.

---

75. How do we limit overfitting?

Rewritten question: What concrete steps will we take to reduce overfitting risk on this relatively small, structured dataset?

Answer:

We will control overfitting at three levels:

1. **Data & validation design**

    - Use **time-consistent splits** that mimic Scenario 1 and 2:

        - Train on earlier months/brands, validate on later months/held-out series.

    - Never mix future months into training for the same series.

2. **Model configuration**

    - Limit tree depth (no very deep trees),
    - Use **min_data_in_leaf** to avoid tiny leaves,
    - Use **L2 regularization**,
    - Apply **column and row subsampling** (feature_fraction, bagging_fraction).

3. **Training process**

    - Use **early stopping**:

        - Stop adding trees when validation metric stops improving.

    - Monitor the gap between train and validation:

        - Large gaps → reduce capacity or increase regularization.

We will also regularly inspect:

- Performance per **bucket** and per **therapeutic area**,
- To ensure we're not just memorizing a few high-volume series.

---

76. How will we log and compare experiments?

Rewritten question: How do we keep track of which model/setting produced which result so we can compare and pick the best?

Answer:

We will set up a **minimal but disciplined experiment log**:

- A simple **CSV or Google Sheet** with columns like:
  - `timestamp`,
  - `scenario` (1 or 2),
  - `model_type` (e.g., CatBoost_v1, LGBM_v2),
  - `features_version` (e.g., FE_v1, FE_v2),
  - Key hyperparameters (depth, learning_rate, n_estimators, regularization),
  - Validation **Metric1/Metric2** scores,
  - Optional notes ("added hospital_rate bins", "removed outliers").

Optionally, if time permits:

- Use **MLflow** or similar to log runs automatically.

But even with a manual table:

- We can compare **experiments side-by-side**,
- Avoid repeating old mistakes,
- And trace back how we achieved the final submission.

---

77. What baseline and target metric scores are we aiming for?

Rewritten question: How do we think about "good enough" in metric terms, relative to simple baselines?

Answer:

Because the official metric is custom and data-dependent, we won't know exact numbers upfront. Our strategy:

1. **Compute baseline scores**:
   - Implement the naive baselines from Q71 (flat avg, linear trend),
   - Evaluate them with `compute_metric1` and `compute_metric2` on a local validation split,
   - Call this our **baseline error** for each scenario.
2. **Define meaningful improvement**:

    - A **small improvement** (e.g., 5%) might be within noise and not worth a lot of complexity.
    - A **clearly meaningful improvement**:
        - Aim for at least **10–20% reduction** in the metric vs baseline,
        - Especially for **Bucket 1** (since it is business-critical and double-weighted).

3. **Competitive target**:
    - In many datathons, the winning models often achieve:
        - Noticeable improvement over simple baselines (sometimes >20–30%),
        - But the exact margin is unpredictable.
    - Our aim:
        - Achieve a **robust, significant reduction** vs baseline,
        - Confirm that improvement is **consistent across both scenarios and buckets**.

---

78. Will we try more than one model family or segmentation strategy?

Rewritten question: Besides the main GBM model, what other models or segmentations might we explore?

Answer:

Yes, we will explore a **small set of complementary approaches**, but in a controlled way:

1. **Multiple model families**
    - Primary: **GBM** (CatBoost / LightGBM).
    - Secondary:
        - A simple **regularized linear model** on normalized volume,
        - Or a very simple **time-series model** (e.g., per-series ARIMA/ETS for a small subset) to check consistency.
2. **Segmentation / specialized models**
    - If EDA suggests very different behaviors, we may consider:
        - Separate models for **Bucket 1 vs Bucket 2** (only for training/analysis, not using bucket as input),
        - Or separate models for **biological vs small_molecule**,
        - Or at least add interaction features capturing these differences.
3. **Ensembling simple variants**
    - If two reasonably strong models behave differently on validation:
        - We may average their predictions,
        - As long as the ensemble is still **easy to explain**.

We will **not** explode into dozens of unrelated models; every extra model must justify itself via:

- Clear metric gain, or
- Clear interpretability/business insight.

---

79. When do we stop creating new models and focus on robustness and storytelling?

Rewritten question: At what point in the weekend do we decide "this model is good enough" and shift energy to analysis and presentation?

Answer:

We will set a **practical decision point**, e.g.:

- By **late Saturday** (or equivalent time before the Sunday deadlines):
    - We expect to have:
        - At least one **strong, validated GBM** per scenario,
        - Baseline vs best-model comparisons,
        - Reasonable confidence in the feature set.

We stop adding new models when:

- Additional model variants only improve the metric by **very small margins** (e.g., <2–3%),
- Or improvements are **not stable** across validation splits.

After that, we will invest our time in:

1. **Robustness checks**:
    - Validate on a different time split or subset of brands,
    - Inspect errors per bucket, ther_area, etc.
2. **Error analysis & insights**:
    - Study high-erosion brands where we underperform,
    - Understand key patterns and failure modes.
3. **Business narrative & visualizations**:
    - Build clear plots of erosion curves, feature importance,
    - Prepare slides that explain **how this helps finance and brand teams**.

This shift ensures we are competitive both on the **metric** and in the **jury evaluation**.

---

80. How do we ensure the final model is stable and not just "lucky"?

Rewritten question: What checks will we perform to confirm our chosen model is robust across splits, seeds, and segments?

Answer:

We will check stability along three axes:

1. **Across validation splits**
    - For each scenario, run the final model on:
        - The main validation split,
        - At least one **alternative time split** (e.g., different cutoff or different set of brands).
    - Confirm that:
        - Metric scores are **similar**, not wildly different.
2. **Across random seeds**
    - For GBM models, many parameters are stochastic (subsampling, initialization).
    - We will:
        - Train the final configuration with **2–3 different seeds**,
        - Check that performance variation is **small** (e.g., within ± a few percent).
    - If performance is unstable:
        - Increase regularization or adjust subsampling.
3. **Across buckets and subgroups**
    - Evaluate:
        - Errors for **Bucket 1 vs Bucket 2**,
        - Errors by ther_area, biological vs small_molecule, maybe by country.
    - Ensure no segment is **catastrophically worse** than others:
        - Especially Bucket 1, since it is double-weighted and business-critical.

If the model passes these checks:

- We can confidently say it is **robust and reliable**,
- Not dependent on a lucky split or a specific random seed,
- And thus suitable as our **final submission model** and the backbone of our presentation.

I. Ensembles, calibration, robustness, fairness, and ethics (81–90)

81. Does a simple ensemble improve error and stability, especially for Bucket 1?

Rewritten question: Should we try a simple ensemble of 2–3 strong models, and does it help overall error, Bucket 1 performance, and stability across splits?

Answer: Yes, we should test a **very simple ensemble**:

- Take **2–3 best-performing models** (for example):

    - GBM with Feature Set A,
    - GBM with slightly different features/regularization (Feature Set B),
    - Possibly a simpler linear/TS model.

- Ensemble = **simple average** (or weighted average) of their predictions per (country, brand, months_postgx).

We will:

- Compare **Metric1/Metric2**:
    - vs each single model,
    - **per bucket**, with special attention to **Bucket 1**.
- Check **stability across validation splits**:
    - If the ensemble gives similar or better performance and reduces variance between splits, we keep it.

If the ensemble:

- Improves Bucket 1 error and
- Makes validation more stable,

then it is worth using as our **final forecasting approach**.

---

82. Can we keep the ensemble simple enough to explain?

Rewritten question: If we use an ensemble, can we explain it to the jury in simple business language?

Answer: Yes. We will deliberately keep the ensemble **very simple and transparent**. For example:

- "Our final forecast is the **average of two complementary models**:
    1. A model focused on **time dynamics** (lags, erosion pattern),
    2. A model focused on **product and market characteristics** (therapeutic area, biological vs small_molecule, hospital_rate, number_of_gx)."

In the presentation we can show:

- A simple diagram: **Model A predictions** + **Model B predictions** → **Average** → **Final forecast**.
- We avoid complicated stacking/metalearners that are hard to explain. The jury just needs to understand that:

> We blended two good but slightly different perspectives to get a more stable, robust forecast.

---

83. Is uncertainty quantification useful for the story?

Rewritten question: Should we try to add prediction intervals or uncertainty estimates around erosion forecasts, and how would we do it?

Answer: Uncertainty is **not required for the metric**, but it can be **very valuable for the business story**:

- Finance and brand teams care about **risk ranges**, not just point forecasts.

Within our time constraints, we can consider **lightweight approaches**:

- **Quantile models**:
  - Train a model to predict not only median volume but also, for example, the **10th and 90th percentile** (if supported by the library).
- Or **bootstrap-style uncertainty**:
  - Train the same model with different random seeds or slightly different samples,
  - Use the spread between predictions as a **rough interval**.

We would then show in slides:

- For selected high-erosion brands:
  - Erosion curve + **shaded band** representing uncertainty.
- Interpretation:
  - "The band indicates plausible ranges of volume; wider bands mean greater uncertainty/risk."

If time is tight, we can at least **discuss uncertainty qualitatively** and highlight this as a natural **next step**.

---

84. How sensitive is our model to splits, outliers, and noise?

Rewritten question: How will we check the model's sensitivity to changes in validation split, outlier handling, and small perturbations in inputs?

Answer: We will explicitly test **robustness** along three dimensions:

1. **Different validation splits**
   - Change the time cutoff or the set of series in validation,
   - Recompute Metric1/Metric2,
   - Check whether performance is **consistent** (not changing wildly).
2. **Outlier handling**
   - Run at least two versions:
     - Raw data (with all outliers),
     - A version with **capped or winsorized** extreme volumes.
   - If performance and key patterns remain similar, the model is robust to outliers.
3. **Small noise in features**
   - Optionally, add small random noise to some numeric features in validation (e.g., ±1–2%),
   - Check if predictions and error change only slightly.

From these checks, we can say:

- “Our solution is **not overly fragile** to how we split data or to individual spikes. It behaves stably under reasonable perturbations.”

---

85. How will we perform targeted error analysis?

Rewritten question: How do we analyze where the model performs poorly and why?

Answer: We will implement **structured error analysis**:

1. **Worst series inspection**
    - Compute per-series prediction error for each (country, brand),
    - Rank series by error (especially in **Bucket 1**),
    - Select the **top worst-predicted series**.
2. **Visual inspection**
    - For those series, plot:
        - actual vs predicted volume over months_postgx,
        - highlight early months vs later months.
    - Look for patterns:
        - abrupt policy shocks,
        - strange data (missing, huge spikes),
        - behaviors the model doesn’t capture (e.g., delayed erosion).
3. **Bucket comparison**
    - Compare average error for **Bucket 1 vs Bucket 2**,
    - Check whether high-erosion brands are systematically harder.
4. **Phase-focused analysis**
    - Compare errors by **time window** (0–5, 6–11, 12–23):
        - e.g., “Most of our error comes from months 0–5, where erosion is steepest.”

These insights go directly into the **slides** to demonstrate that we understand:

- Where the model works well,
- Where it struggles,
- What this means for business decisions.

---

86. Do the top features make pharma/commercial sense?

Rewritten question: Do the model’s most important features align with domain intuition about what drives erosion?

Answer: We will verify that **top features** are **domain-plausible**. Expected key drivers:

- **Pre-entry volume and growth:**
    - Average volume in the last 6–12 months before generic entry,
    - Recent growth trend pre-entry.
- **Early post-entry dynamics (Scenario 2):**
    - Volume trend in months 0–5 (how fast the decline starts).
- **Generics competition:**
    - `number_of_gx` and its **evolution over time** (e.g., speed of increase).
- **Product characteristics:**
    - `therapeutic_area`,
    - `biological` vs `small_molecule`,
    - `hospital_rate` (hospital vs retail exposure),
    - `main_package` (pill vs injection, etc.).

We will use:

- Built-in **feature importance** from the GBM,
- Possibly **SHAP** or partial dependence plots for a few key features.

We will check that:

- Features like **pre-entry volume**, **n_gxs**, and **early trend** appear near the top,
- No strange “technical” features dominate (e.g., pure IDs).

If the feature ranking matches business intuition, we can confidently tell the jury:

> “The model is using exactly the kinds of drivers that commercial and medical teams already consider important.”

---

87. Does performance differ by subgroup (country, area, drug type)?

Rewritten question: Do we see big differences in model performance across countries, therapeutic areas, or drug types, and how do we handle that?

Answer: We will compute **segment-wise metrics**:

- For each major subgroup, compute Metric1/Metric2 or a simpler error proxy:
    - By **country** or region (aggregated),
    - By **therapeutic_area** (e.g., oncology vs CNS vs anti-infectives),
    - By **biological vs small_molecule**,
    - By **hospital_rate bands** (low, medium, high).

We will then:

- Look for **patterns**:
    - Example: worse performance for **rare/highly specialized oncology** brands,
    - Or for **high-hospital-rate** drugs where tender dynamics are complex.
- Highlight **weak spots** in the presentation:
    - “Performance is strong and stable for most areas, but less reliable for [X], where behavior is more irregular.”

If time allows, we might:

- Slightly adjust the model or features for clearly problematic segments,
- Or propose **segment-specific refinements** as **future work**.

---

88. Are there sensitive or regulatory aspects we must frame carefully?

Rewritten question: Are there aspects of region or system differences that require careful wording so our model is not misinterpreted?

Answer: Yes. This is a **commercial/healthcare** context, not a purely financial one. We should be careful that our solution is **not presented as**:

- A **pricing** or **access** recommendation tool,
- A direct prescription behavior manipulator.

Key points to emphasize:

- The model operates on **aggregated sales volumes** at country–brand level, not on patient-level data.
- Its goal is **forecasting volume erosion** to improve:
    - **Revenue planning**,
    - **Budget allocation**,
    - **Operational readiness** (e.g., supply chain).

We should explicitly say:

- It is **not** a tool for:
    - Setting drug prices,
    - Making patient-level clinical decisions,
    - Overriding regulatory or ethical constraints in any market.

This framing keeps the solution aligned with **finance and planning** use cases and avoids sensitive interpretations.

---

89. What caveats and disclaimers must we include?

Rewritten question: What limitations and warnings should we clearly state about our model?

Answer: We will clearly state **several caveats** in the final slides and report:

1. **Predictive, not causal**
    - The model identifies **associations** in historical data,
    - It does **not infer causal effects** (e.g., "if we change X, Y will change by Z").
2. **Historical pattern dependence**
    - Results are based on **past behavior** of volumes, competition, and market conditions,
    - They may not hold under:
        - New regulations,
        - Major policy changes,
        - Unseen macro shocks.
3. **Decision-support, not decision-replacement**
    - Forecasts should be used to **inform** human decisions,
    - Final decisions must remain with finance, brand, and leadership teams who can integrate:
        - Local knowledge,
        - Regulatory constraints,
        - Strategic priorities.
4. **Data scope limitations**
    - Only covers brands and markets in the provided dataset,
    - Behavior may differ in **unrepresented countries** or **new product classes**.

By stating these explicitly, we show we are using AI **responsibly** and do not oversell what the model can do.

---

90. When do we explicitly recommend human override?

Rewritten question: In which situations should human experts override or heavily qualify our model's predictions?

Answer: We will clearly indicate that **human override is essential** in several situations:

1. **Very short or unusual history**
    - New or niche products with limited pre-entry data,
    - Brands with atypical launch or promotion patterns.
2. **Markets under structural change**
    - Countries undergoing:
        - Major regulatory reforms,
        - Reimbursement or tender changes,
        - Macroeconomic crises.

- In such contexts, historical patterns may be **poor guides**.

3. **Extreme outlier behavior**
    - Series where our own error analysis shows:
        - Very poor fit,
        - Highly unstable predictions across versions,
    - These should be flagged for **manual review**.
4. **Critical strategic decisions**
    - When decisions involve:
        - Large capital commitments,
        - Major headcount or supply chain shifts,
    - Our forecasts should be one of several inputs, not the sole driver.

We can summarize in the presentation:

> "Our model is a **strong assistant** for planning, especially for typical erosion cases. For unusual or high-stakes situations, we explicitly recommend **human expert review and override**."

J. Implementation, reproducibility, presentation, and future work (91–100)

91. Can someone **reproduce our entire pipeline**—from raw CSVs to final prediction file—using a small number of clear steps or a single main script/notebook?
92. Is our **project structure** clear:
    - Separate folders for raw data (as provided), processed data, notebooks, `src/`, results, and slides,
    - Easy for a mentor or judge to navigate?
93. Do we have a concise **README**:
    - Explaining environment and dependencies,
    - How to run EDA, training, and generate the final submission,
    - Any scenario-specific commands (Scenario 1 vs Scenario 2)?
94. Are key **configuration values** (paths, random seed, hyperparameters, feature sets) centralized in a small number of files or cells and not scattered?
95. Have we cleaned final notebooks and scripts of **dead code and noisy output**, leaving:
    - A clear narrative of **the chosen solution**,
    - Minimal friction for reviewers?
96. Can we explain our solution in **one minute** in business language:

  - “We forecast volume erosion after generics enter, using historical volume, generic competition, and drug characteristics, focusing on high-erosion brands to help finance and brand teams plan revenue and strategy”?

97. Do our **plots and tables** for the final presentation:

    - Clearly show typical erosion patterns and bucket differences,
    - Illustrate how our forecasts compare to actuals on key examples,
    - Use readable labels and avoid visual clutter?

98. Have we explicitly documented:

    - Our **baseline approach and score**,
    - Our **best model and score** on validation,
    - The **incremental improvements** and why they matter practically (e.g., better early-month accuracy, improved Bucket 1 performance)?

99. Have we listed the main **limitations**:

    - Data coverage (only 24 months before/after),
    - Possible distribution shifts,
    - Limited modeling time; and at least 2–3 realistic **next steps** (e.g., integrating pricing data, richer market covariates, more advanced uncertainty modeling)?

100. If we had **1–2 extra weeks** after the Datathon, what would be our top three priorities to turn this into a **production-ready tool**:

     - E.g., robust retraining pipeline, integration with internal finance dashboards, better calibration of risk/uncertainty, and automated reporting for brand and country teams?