# Part 4: Strategic Scenarios & Reflection

## Question 1: Balancing Urgency vs. Quality

**Scenario:** Launch rating prediction in 2 weeks with 75% accuracy (Target: 85%).

### Recommendation: Launch with "Beta" Label + Improvement Plan

I recommend launching immediately with a visible "Beta" indicator, rather than delaying for perfection.

### Rationale (Bias for Action)

1. **Value over Perfection:** A 75% accurate prediction is still significantly more valuable to a user than no prediction at all (0% value).
2. **Data Flywheel:** We cannot improve from 75% to 85% efficiently in a lab vacuum. We need real user feedback and interaction data to identify *where* the model fails.
3. **Risk Mitigation:** The "Beta" label manages user expectations and protects the brand while allowing us to ship.

### Implementation Plan

- **Week 1 (Launch):** Deploy model with a UI tooltip: *"Beta Feature: Estimates are experimental."* Implement a "Thumps Up/Down" feedback mechanism next to the rating.
- **Week 2-3 (Iterate):** Analyze the "Thumbs Down" data. Are we failing on specific cuisines? Specific price ranges?
- **Week 4 (Update):** Retrain the model using the new feedback data to close the gap toward 85%.

---

## Question 2: Debugging Poor Search Results

**Scenario:** Search fails for "romantic date night restaurants."

### Systematic Debugging Process

**1. Reproduce & Quantify**

- **Action:** Run the query "romantic date night" against the current system.
- **Observation:** The system returns random restaurants or misses obvious candidates like "The Original French House."
- **Metric:** Current Recall is <30% (Semantic search misses keyword tags).

**2. Root Cause Analysis (The "Why")**

The issue is a **Semantic Gap** in the implementation.

- **Hypothesis:** The Vector Database finds restaurants based on *text description similarity*. If a description says "dim lighting" but lacks the specific word "romantic," the embedding distance might be too far.
- **Confirmation:** The metadata contains a field `attributes: "Romantic"`, but the initial RAG implementation relied purely on vector similarity and ignored this structured tag.

**3. Proposed Fixes**

- **Short-Term (Hotfix):** Implement **Hybrid Search**. If the query contains high-value keywords (Romantic, Outdoor, Business), force a Metadata Filter (`WHERE attributes CONTAINS 'Romantic'`) alongside the vector search.
- **Long-Term:** Implement **Query Expansion**. Use an LLM agent to rewrite "date night" into multiple search vectors: "romantic," "quiet atmosphere," "fine dining," "couples," effectively casting a wider net.

**4. Success Metrics**

- **Recall:** >90% of restaurants with the "Romantic" tag must appear in the top 10 results.
- **Precision:** >80% of results must be validated as relevant by human review.

---

## Question 3: Learning from Failure

**Scenario:** Describe a recent ML project failure/obstacle and how you applied the learning.

### The Context: Overfitting on Small Datasets (Current Project)

During the development of the **Task 2.1 Rating Prediction Model**, I encountered a significant obstacle.

- **The Setup:** I trained an XGBoost model on the provided dataset of 50 restaurants using 38 generated features.
- **The Failure:** The model achieved a **Training $R^2$ of 0.99** (near perfect) but performed poorly on unseen test data (Negative $R^2$ in initial splits).
- **The Diagnosis:** This was a classic case of **Overfitting**. With only 50 data points and 38 features, the model simply memorized the training data ("noise") rather than learning the underlying patterns ("signal").

### How I Applied the Learning (Disagree and Commit)

Instead of trying to force a complex model to work, I pivoted my strategy:

1. **Simplify:** I acknowledged that for n=50, a complex Neural Network or deep tree was the wrong tool.
2. **Regularize:** I heavily tuned the XGBoost hyperparameters (`max_depth=3`, `reg_alpha=0.1`) to penalize complexity.
3. **Validation:** I moved from a simple Train/Test split to **Cross-Validation**, which provided a more honest assessment of model performance on small data.

### The Outcome (Learn Eternally)

This experience reinforced the ML Engineering principle that **Data Quantity dictates Model Complexity**.

- **Application:** For the final submission, I documented this limitation transparently in the Technical Doc, explaining that while the *pipeline* is production-ready, the *model performance* is mathematically limited by the dataset size.
- **Future Strategy:** In production, I would prioritize setting up a data collection pipeline to increase sample size to 1,000+ before attempting to increase model complexity again.