Part 3: Creative Extension

1. **Approach 1: Teacher-Student Technique (Synthetic Dataset)**

1. How it works:

Since we don't actually have access to real-life examples of what might be characterized as manipulative reviews, let's generate those examples.

We would use a super smart AI model like GPT-4 for what I refer to below as a Teacher.

The prompt for the Teacher would simply read, 'Create 1,000 reviews that utilize scarcity tactics, guilt trips, or fake urgency.

- **Scarcity:** Lying that the store is almost out of the item.
- **Guilt:** Making the customer feel bad if they don't buy it.
- **Fake Urgency:** Rushing the customer (like saying "Sale ends in 1 minute!" when it doesn't).

After that, we will ask the AI to simply write 1,000 **nice** reviews.

Now that the Teacher has created this textbook of examples, we can use it to train our own small and inexpensive AI system (the "Student") on these examples to learn to pick out the patterns. Then we can use the Student on our servers for free.

2. Pros and Cons

- **Pros:**
    - **Cheap to run:** We don't pay for expensive API calls every time a user posts a review. The Student runs locally on our hardware.
    - **We control the rules:** If we want to catch "Hidden Fees," we just tell the Teacher to write examples about hidden fees.
- **Cons:**
    - **Robot bias:** The Student only knows what the Teacher wrote. If the Teacher writes very formal sentences, the Student might miss slang or messy typos that real humans use.

3. Expected Performance

- **Estimate: High (75% - 85% Accuracy)**
- **Reasoning:** Big AIs are amazing at role-playing. They can create very clear, obvious examples of manipulation. Our small model will easily pick up on these strong signals.

4. Validation Plan

- **The "Golden 50":** I will personally sit down and label just 50 real reviews from our database. We will test the Student model against these 50 real-world examples to make sure it isn't just memorizing robot text.

2. Approach 2: The "Logic Check" (Zero-Shot NLI)

1. How it works :

We will employ the same model from Part 2, which is the NLI model. The model knows logic and English relationships.

Therefore, we won't have to train it on any specific review.

All we have to do is turn the detection task into a True/False question.

For each new incoming review, the model seeks the answer to this question: "Does this text pressure the reader?" or the answer to this question: "Does this text make the reader feel guilty?"

If the answer is "Yes, that description fits," we mark the review as flagged. It doesn't use any trained specifics, just general knowledge.

2. Pros and Cons

- **Pros:**
    - **Instant Deployment:** We can have this running in 1 hour. It requires zero training data.
    - **Explainable:** The model tells us *why* it flagged a review (e.g., "I found 'Urgency'").
- **Cons:**
    - **Slow & Heavy:** It takes a lot of computing power to ask these questions for every single review.
    - **Too Literal:** It might flag innocent things. For example, *"Fire! Leave the building now!"* sounds like "Urgency," so the model might flag it even though it's a safety warning, not a scam.

3. Expected Performance

- **Estimate: Moderate (65% - 75% Accuracy)**
- **Reasoning:** It is a "Generalist," not a "Specialist." It will catch obvious manipulation, but it might struggle with nuances specific to our product.

4. Validation Plan

- **The "Temperature Check":** We run this on 1,000 unlabelled reviews and look at the scores. We will likely see a lot of noise, so we simply tune the "sensitivity knob" (threshold) until we stop flagging normal reviews.

Part 4 :

Q1: Latency Optimization (200ms → <50ms)

To deal with 10 million reviews per day, we have to be much quicker.

1. "Filter" Method (Cascade): First, use a super-fast and dumb model (Bi-Encoder). It culminates in discarding 90% of the so-called "safe" reviews. They can be distinguished easily. Send the puzzling 10% to the slow but smart system.
2. Shrink the Math (Quantization): We reduce the AI's complex math system from 32-bit numbers down to whole numbers with only 8-bits. It runs 3 times faster on regular computers with minimal loss of intelligence.
3. Cut the Fluff (Pruning): The AI does not have to read a 500-word story. We trim the text down to the essential first 100 words or key sentences. The less it reads, the faster it goes.

Q2: Adversarial Robustness (Stopping Bad Actors)

Spammers will try to outsmart your AI.

- Attack: "Leetspeak" (Typos): Writing "This is f@ke" or "G00d product" to fool filters.
- Defense: Text Cleaning One of the processes we use is a "spell check" program that cleans up these symbols - turns "@", for example, to "a" - before even letting the AI see the text.
- Attack: Invisible Text: Here, white text is hidden on a white background to trick the AI's context.
- Defense: Code Cleaning: All HTML and hidden formatting codes are removed. The AI reads only precisely what any human would see.

Q3: The Confidence Calibration Problem

Problem: The AI states that it is “95% sure!”, but it's correct only 70% of the time. It is arrogant.

Fix: Temperature Scaling.

This is a mathematical equation we use at the final stage. This equation compels our artificial intelligence to “relax” or reduce its marks.

Before: AI shouts "99% Sure!"

Since: Math adjusts it to "75% Sure."

Instead, the final decision (Fake vs. Real) remains the same, while the confidence score becomes honest.

Q4: Cold Start for New Product Categories

Problem: AI has learnt that "Fast = Good" (Laptops) but hasn't understood that "Tight = Bad" (Clothing). You have only 100 pieces of clothes' examples.

Strategy: Few-Shot Learning SetFit.

We don't retrain the whole brain, we take your 100 examples of outfits and run those through a tool that makes variations synonyms of those, so now you've got 500 examples. Then we use that small dataset to "fine-tune" the AI for only fashion logic, and it works amazingly well with very little data.

Q5: Ethical Consideration (The "Update" Edge Case)

Problem: A user writes: "Love it! ... Update: It broke." This is a contradiction, but it is honest (time passed).

Systematic Fix: Time Splitting.

We program the system to look for keywords like "Update:", "Edit:", or "Month 2:".

If found, we cut the text in half. We check Part 1 for contradictions, and we check Part 2 for contradictions. But we do not compare Part 1 against Part 2. Changing your mind over time is allowed.