Building an AI agent using **n8n** that can analyze, validate, and debug a **Jupyter notebook's code** to check for a **predefined expected output** involves integrating a few components:

**🧠 What We’re Building:**

An **automated n8n workflow** that:

1. Accepts or fetches a Jupyter Notebook (.ipynb).
2. Extracts Python code from it.
3. Executes the code in a secure sandbox (e.g. using an API or script).
4. Compares the actual output to expected output(s).
5. If the output is wrong, sends the code and error context to an LLM (like OpenAI or local model) to debug and suggest a fix.
6. Optionally, applies the fix and re-runs.

**✅ Step-by-Step Guide to Build This Agent with n8n**

**🔧 1. Set Up Input (Notebook File)**

* **Trigger node**: Webhook or Google Drive/Dropbox node to accept/upload the .ipynb file.
* Or fetch from a URL.

**📦 2. Parse the Notebook**

* Use a Code node (or Python in Docker/Command node) to run a Python script that:
  + Loads the .ipynb file.
  + Extracts code cells only.
  + Returns the code as a string.

**Python Code Example**:

import nbformat

with open("your\_notebook.ipynb", "r", encoding="utf-8") as f:

nb = nbformat.read(f, as\_version=4)

code\_cells = [cell['source'] for cell in nb.cells if cell['cell\_type'] == 'code']

all\_code = "\n\n".join(code\_cells)

return {"code": all\_code}

You can use this in a:

* Execute Command node (with Python installed in Docker or system)
* Or wrap it as a custom API you call from n8n

**🧪 3. Run and Validate Output**

You need to **safely execute** the extracted code and **capture its output**.

**✅ Option 1: External Sandbox**

Use a cloud-based service like:

* **Replit API**
* **Jupyter Kernel Gateway**
* A self-hosted Docker container with nbconvert or papermill
* Custom Flask API on a separate server

This helps you:

* Run the code securely
* Capture stdout, stderr, variables, and return values

Compare the **actual output** to **predefined expected values** (you can store them in n8n as parameters or in a database).

**Example Validation:**

expected = {"result": 42}

# After running notebook

if variables["result"] != expected["result"]:

status = "failed"

reason = f"Expected {expected['result']}, got {variables['result']}"

**🧠 4. Debug Using an AI Model**

If the result **doesn’t match**:

* Send the full code + expected output + actual output to **OpenAI GPT-4** or similar.
* Use HTTP Request node to call the model.

**Prompt Example**:

This is the Python code from a Jupyter notebook:

<INSERT CODE HERE>

Expected output: result = 42

Actual output: result = None

Please explain why the code doesn’t produce the expected result and suggest a fix.

Use the OpenAI node or HTTP Request node to send this.

**🔁 5. Optional: Apply Fix and Re-run**

If the AI suggests a fix, you can:

* Ask the AI to provide the **corrected code**
* Re-run the new code through the same validation process

**💡 Bonus (Optional Steps)**

* Log results to Airtable/Google Sheets
* Send summary report via Slack/Email
* Version control with GitHub API

**🧱 Required Tools & Setup**

| **Component** | **Tool/Node** | **Purpose** |
| --- | --- | --- |
| Notebook input | Webhook / Google Drive | Accept .ipynb |
| Code parsing | Python script (Execute command) | Extract code |
| Code execution | External service (Docker/Flask) | Secure execution |
| Output comparison | n8n IF node / custom code | Check output |
| AI Debugging | OpenAI / LLM API | Explain & fix |
| Notification | Slack / Email node | Alert or report |

**🔐 Security Warning**

⚠️ Executing arbitrary code is dangerous. Always isolate execution environments and **do not run untrusted code directly inside n8n or your main system**. Use containers or serverless sandboxes.

**🧪 Want a Working Example?**

If you'd like, I can generate:

* A sample n8n workflow export (in JSON)
* A Python script for the execution and parsing parts
* OpenAI prompt templates

Let me know which parts you'd like to implement first or try out!