# Agentic Medical Claim Analysis Tool

This Python script uses a multi-agent AI approach to analyze medical claims from an Excel file. It processes each claim using a set of predefined "agents," each with a specific task (e.g., checking for upcoding, matching diagnosis to services). The results from each agent are then written to a new output Excel file for easy review.

The tool supports multiple AI backends, allowing the user to choose between **Ollama** (for local models like Llama 3.1) and **OpenAI** (e.g., GPT-3.5-Turbo, GPT-4o-mini) at runtime.

## Features

* **Reads from Excel:** Ingests medical claim data directly from an .xlsx file.
* **Multi-Agent Analysis:** Applies multiple specialized AI agents to each claim.
* **Easily Extensible:** New agents can be added simply by defining a new system prompt in the AGENT\_PROMPTS dictionary within the script.
* **Dual AI Backend:** Supports both Ollama and OpenAI APIs.
* **Structured Output:** Saves all agent responses in new columns in a separate output Excel file, correlating results with the original claim data.

## Requirements

* Python 3.x
* The following Python libraries:
  + pandas
  + ollama
  + openai
  + openpyxl

## Installation

1. **Clone the repository (or download the script):**  
   git clone [https://your-repo-url.git](https://your-repo-url.git)  
   cd your-repo-folder
2. **Install the required Python libraries:**  
   pip install pandas ollama openai openpyxl
3. Install Ollama (if using):  
   If you plan to use the Ollama option, you must have the Ollama desktop application installed and running. You also need to have the model specified in the script pulled:  
   ollama pull llama3.1  
     
   (You can change the model by editing the OLLAMA\_MODEL constant in the script).

## Configuration

Before running the script, you may need to configure the following constants at the top of test.py:

### 1. Agent Prompts

The core logic is defined in the AGENT\_PROMPTS dictionary. You can add, remove, or edit agents here. The dictionary key (e.g., "Agent\_1\_Diagnosis\_Service\_Match") is used to name the output column.

AGENT\_PROMPTS = {  
 "Agent\_1\_Diagnosis\_Service\_Match": """  
You are an AI agent focused on the consistency between Diagnosis and Services Provided...  
Output Format:  
Score: [1-10]  
Rationale: [Your one-sentence explanation]  
""",  
 "Agent\_2\_Upcoding\_Check": """  
You are an AI agent specializing in identifying potential upcoding...  
Output Format:  
Upcoding Suspicion: [Low/Medium/High]  
Reasoning: [Your brief explanation]  
""",  
 # --- ADD YOUR NEW AGENTS HERE ---  
 "Agent\_3\_Your\_Check\_Name": """  
Your system prompt for the new agent goes here...  
"""  
}

### 2. AI Models

You can change the specific models to be used:

OLLAMA\_MODEL = "llama3.1" # Or "mistral", etc. (must be pulled in Ollama)  
OPENAI\_MODEL = "gpt-3.5-turbo" # Or "gpt-4o-mini", "gpt-4", etc.

### 3. Input/Output Files

Define your input and output Excel file names:

INPUT\_EXCEL\_FILE = "Multiple Claims.xlsx"  
OUTPUT\_EXCEL\_FILE = "Multiple Claims - Agentic Output.xlsx"  
EXCEL\_SHEET\_NAME = "Sheet1"

The INPUT\_EXCEL\_FILE is expected to have at least two columns:

* Claim ID: A unique identifier for the claim.
* Input Details: The text data for the claim (e.g., diagnosis, services provided) that will be fed to the AI agents.

### 4. OpenAI API Key (CRITICAL)

If you plan to use the **OpenAI** option, you **MUST** edit the main() function and replace the placeholder API key with your own.

Find this section in main():

if api\_choice == '2':  
 # V V V - EDIT THIS LINE - V V V  
 hardcoded\_api\_key = "keyyyyyyyy"   
 # ^ ^ ^ - EDIT THIS LINE - ^ ^ ^  
  
 if not hardcoded\_api\_key or hardcoded\_api\_key == "PASTE\_YOUR\_OPENAI\_API\_KEY\_HERE":  
 # ... (error handling)

**Note:** Hardcoding API keys is insecure. For production use, consider using environment variables (os.environ.get("OPENAI\_API\_KEY")) instead.

## Usage

1. Place your input data file (e.g., Multiple Claims.xlsx) in the same directory as the script.
2. Run the script from your terminal:  
   python test.py
3. You will be prompted to choose your AI backend:  
   Select the AI model/API to use:  
   1: Ollama (using llama3.1)  
   2: OpenAI (using gpt-3.5-turbo)  
   Enter your choice (1 or 2):
4. Enter 1 for Ollama or 2 for OpenAI and press Enter.
5. The script will process each row from the input file and print its progress to the console.
6. Once finished, the results will be saved to the file specified by OUTPUT\_EXCEL\_FILE (e.g., Multiple Claims - Agentic Output.xlsx). This new file will contain all the original data plus new columns for each agent's response (e.g., Agent\_1\_Diagnosis\_Service\_Match\_Response).

## License

This project is licensed under the MIT License. See the LICENSE file for details (or add your own).