Assignment 2: The Multi-Agent AutoML Team

Overview

In this assignment, you will not build a single AI script. You will build an **autonomous AI Data Science Team**.

Your system will consist of **three distinct LLM Agents** that collaborate to process a dataset. Unlike standard AutoML that simply runs a grid search, your agents will reason about the data, make semantic decisions (e.g., "This column looks like a date, I should extract the month"), and write their own code to train models.

The Architecture: A Sequential Pipeline

Your system must implement a sequential handoff workflow. Information flows from one agent to the next via a **Structured Report**.

The pipeline consists of three specialists:

1. **The Data Cleaner:** Audits quality and handles missing values/outliers.
2. **The Feature Engineer:** Creates new variables and selects the most relevant features.
3. **The Model Trainer:** Generates Python code to train, evaluate, and iterate on an XGBoost model.

Detailed Agent Specifications

Agent 1: The Data Cleaner (" The Auditor")

- **Goal:** Ensure the dataset is technically sound and ready for logic operations.
- **Behavior:** It initiates the process by inspecting the raw CSV. It must identify missing values, wrong data types, or high cardinality columns.
- **Tools:**
  - `inspect_metadata(df)`: Returns shape, data types, and null counts.
  - `get_column_stats(df, col)`: Returns distribution or unique values.
  - `impute_missing(df, col, strategy)`: Fills NaNs (mean, median, mode).
  - `drop_column(df, col)`: Removes unusable columns.
- **The Handoff:** Once satisfied, this agent saves a `clean_data.csv` and generates a text summary (e.g., *"I dropped ID because it was unique, and imputed Age with the median"*).

Agent 2: The Feature Engineer ("The Architect")

- **Goal:** Maximize information density. This agent receives the `clean_data.csv` and the summary from Agent 1.
- **Behavior:** It uses domain logic to create new features. It must perform feature selection to remove redundancy.
- **Tools:**
  - `create_interaction(df, expression)`: Creates a new column via math (e.g., `df['income_per_age'] = df['income'] / df['age']`).
  - `encode_categorical(df, col)`: Applies One-Hot or Label encoding.
  - `correlation_analysis(df, target)`: Checks how features relate to the label.
  - `select_top_features(df, k)`: Keeps only the `k` most predictive features.
- **The Handoff:** Saves `engineered_data.csv` and generates a text summary explaining its "strategy" (e.g., *"I created a ratio feature and removed collinear variables"*).

Agent 3: The Model Trainer ("The Coder")

- **Goal:** Train a robust XGBoost model.
- **Behavior:** This agent is unique. It reads the engineered data and **generates executable Python code** to train the model.
- **The Feedback Loop (Critical Requirement):**
  1. The Agent generates code to train a baseline XGBoost model.
  2. It executes the code and reads the performance metrics (Accuracy/Recall/F1).
  3. **Decision Step:** The Agent must decide: *Is this good enough?*
     - **If No:** It generates *new* code with different hyperparameters (e.g., learning rate, max_depth) and retrains.
     - **If Yes:** It outputs the final metrics and terminates.
- **Tools:**
  - `execute_python_code(code_string)`: Runs the code generated by the LLM and returns the stdout/stderr (logs and metrics).

The "Creative Control" Requirement

The LLM must be the one making the decisions.

- **Do not** hardcode "Always fill missing values with 0."
- **Do not** hardcode "Always run XGBoost with 100 trees."

You must prompt the LLM to **look** at the data using its tools and **decide** the best course of action.

Deliverables

1. **The System Code:** Python scripts implementing the 3-Agent flow.
2. **Execution Logs:** A log showing the "conversation" or thought process of the agents (e.g., *"Agent 1: I see 20% missing values in 'Age', I will impute."*).

3. **Final Report:** A Markdown file generated by the system summarizing the transformation from Raw Data → Final Model Metrics.

Student Checklist

- [ ] Does Agent 1 successfully clean the data and pass it to Agent 2?
- [ ] Does Agent 2 add at least one new feature based on logic?
- [ ] Does Agent 3 successfully generate and execute Python code?
- [ ] Does Agent 3 react to the training results (the feedback loop)?

Due Date: 14 days from assignment date