1 Implementation Workflow

The project creates a workflow to finish the task on Titanic data train, validation, and test CSV files, can train a model by itself and generate two submission files (submission_val.csv, submission_test.csv) with validation-set accuracy no less than 75%.

Improvement tricks: 1. Add Constraints for LLM: Use only standard libraries along with pandas, numpy, and scikit-learn. Do not alter file paths. Always print key metrics to ensure informative observations. Define CSV format PassengerId,Survived

Component	**Key Implementation Notes**
DataScienceAgent	Maintains conversation state; deletes stale submission files; halts when accuracy ≥ 0.75 or after 10 iterations.
LLM Integration	Uses Qwen 2.5–Coder–32B–Instruct via DeepInfra API with temperature = 0 for determinism.
Code Execution	Runs LLM–generated Python in a sandbox and captures artefacts. Isolated interpreter exposes only `pandas`, `numpy`, `scikit-learn`, and standard libraries.
Evaluation System	Computes validation accuracy and enforces CSV schema.
Iterative Improvement	Feeds structured feedback back to the LLM for self–correction. Adds an Observation message each loop containing metrics or tracebacks to guide the next code patch.

2 R1-like RL Training Strategy

R1 Methodology: DeepSeek-R1 uses a rule-based reinforcement learning strategy - Group Relative Policy Optimization (GRPO) to enhance reasoning capabilities in large language models (LLMs). It utilizes verifiable rewards—binary signals to train models effectively even without extensive SFT.

Stage 1: Pretraining with Verifiable Rewards In the initial stage, the model is trained using a rule-based reward system. Use maybe thouds of warm-up data to do supervised fine-tuning. This method focuses on verifiable, interpretable rewards to guide the learning process. The combined reward is calculated as a weighted sum:

- **Correctness Reward**: Evaluates the correctness of the model's output. In coding tasks, code can be executed against predefined test cases.
- **Format Reward**: Format_ok = 1 if the agent returns files that exactly match the required CSV schema and obey internal style rules (e.g. reasoning in <think> ... </think> blocks, code in <code> ... </code> tags).
- **Execution Reward**: No_error = 1 when the generated code executes without exceptions and all unit tests pass.

Stage 2: Reinforcement Learning like Group Relative Policy Optimization (GRPO)

- **Further RL training with Design Verifiable Rewards**: After establishing a foundational reasoning capability, employ reinforcement learning like GRPO to fine-tune the model's behavior further, ensuring alignment with specific objectives and preferences. Use the accuracy rewards, format reward and execution reward.
- **Iterative Feedback Loop and Monitor for Reward Hacking**: Continuously sample outputs, evaluate them using the defined reward functions, and update the model's policy accordingly. Ensure that the model isn't exploiting loopholes in the reward functions to achieve high scores without genuinely solving the tasks.

3 Case study & Qualitative Analysis - Prompt Engineering Effects

3.1 Experiment Setting

A DataScienceAgent uses a ReAct prompt to coordinate Qwen2.5-Coder-32B-Instruct in solving the classic Titanic Kaggle task. Print the "Thought", "Action" and "Observation". Two runs are compared:

Run ID	Prompt Fragment	Val Accuracy	Test Accuracy
A	"Train a model" (RF clause removed)	0.832 ± 0.0	0.821
B	"Train a model like random forest"	0.811 ± 0.0	0.860

Table 1: Validation and Test Accuracy with Different Prompt Variants

Run A Iteration 1 built an end-to-end **ColumnTransformer +RandomForest** pipeline:

- **Pre-processing**: median/mode imputation; one-hot encoding of six categorical columns; standardisation of four numeric columns.
- **Model**: RF with 100 trees, default hyper-parameters.
- **Metrics**: Validation Accuracy = 0.8322, Test Accuracy = 0.8212.
- The success threshold was met immediately; the loop stopped after a single turn.

Run B With the RF cue, the LLM simplified the pipeline:

- Dropped high-cardinality columns (`Name`, `Ticket`, `Cabin`).
- Replaced the transformer with manual median/mode fills and `pd.get_dummies` on `Sex`, `Embarked`.
- Same RF size (100 trees).
- **Metrics**: Validation Accuracy = 0.8112, but **Test Accuracy jumped to 0.8603**.

3.2 Qualitative Analysis

- **Prompt Sensitivity.** A single line—"Train the model like random forest"—altered the LLM's entire feature pipeline, showing the agent's readiness to reinterpret vague instructions and an outsized prompt–performance coupling.
- **Convergence Behaviour.** Both episodes converged in a single iteration; thereafter the agent merely reported metrics. This might suggest adding a secondary reward for "metric improvement" could spark further exploration once the accuracy floor is reached.
- **Reasoning Process.**In both runs the *Thought* outlined a step-wise plan before emitting code—useful for auditability. Run A's plan explicitly separated numerical vs categorical pipelines, whereas Run B condensed preprocessing into a utility function—less modular but more readable.
- **Generalisation vs. Fit.** The richer Run A pipeline over-fit slightly: it out-performed on the *seen* validation split yet under-performed on the *held-out* test set. Run B's leaner feature set sacrificed 2% on val but gained 4% on test, echoing a bias–variance trade-off.

4 Future Improvement Proposal

- **Try More Models**: Strategic hints (here, "Random Forest") can steer the LLM towards simpler, more generalisable data pipelines. Try more models to get batter results.
- **Memory of Feature Engineering**: Log feature-importances on each run; comparing them would have explained why the simpler pipeline in Run B generalised better.
- **Reward Design**: Introduce a delta-reward or diversity-reward for accuracy improvement to encourage at least one refinement pass before early termination, which could prevent post-success stagnation.

5 Ablation study on the adding tool module

I try to add tool module but uncovered failure modes that masked these gains:

- A positional-argument error during early runs
- Stray markdown fences triggering SyntaxError
- Missing submission_*.csv artefacts when exceptions short-circuited the pipeline.