An Autonomous Multi-Agent System for Machine Learning

Autonomous Data Scientist Agent (ADSA):

Outperforming 98% of Data Scientists in Data Science Competitions

Atenea Labs Research Team

led by Carlos Navarro Cabrera

info@atenea.dev

Technical Whitepaper - January 15, 2025

Index

EXECUTIVE SUMMARY

This whitepaper presents a **revolutionary methodology** developed by **Atenea Labs** that has proven to outperform **98% of data scientists** in global data science competitions. Through a **multi-agent AI system**, we have managed to automate the entire machine learning process, from code generation to validation and iterative model optimization. The result is an extremely versatile and scalable approach, capable of adapting to different domains and requirements, with a clear focus on continuous improvement.

2. CONTEXT AND CHALLENGE

2.1. The Spaceship Titanic Challenge

To validate our multi-agent system, we participated in the Kaggle "Spaceship Titanic" competition[1]. The scenario consists of predicting which passengers were transported to an alternate dimension after the ship's collision with a spacetime anomaly. Records are available for approximately 13,000 passengers (8,700 for training and 4,300 for testing), including demographic information, travel variables, and on-board purchase data. The ultimate goal is to generate a Boolean column (True/False) that indicates whether the passenger was transported

2.2. Competition Rules and Evaluation

The competition is mainly governed by:

- Evaluation Metrics: Accuracy.
- Submission Format: CSV with "PassengerId" and "Transported".
- Limitations: Up to 10 submissions per day and dynamic 2-month leaderboard windows.
- Objective: Maximize accuracy and stay at the top of the leaderboard for as long as possible.

Relevance and Applications

The Spaceship Titanic challenge is an ideal case study to demonstrate the capability of our system, due to:

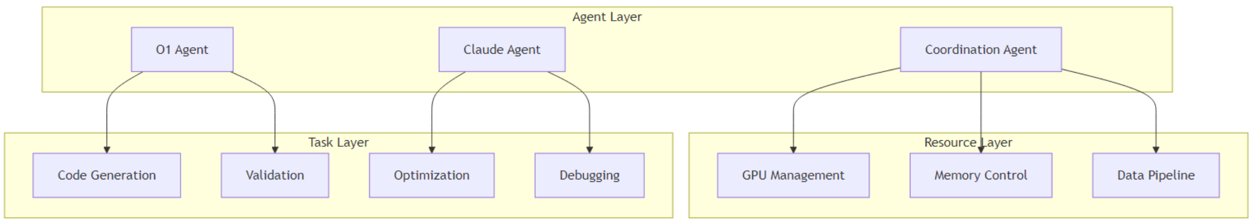
- Complexity of the problem: Missing data, noise and multitude of variables.
- Analogous real applications: Anomaly detection, user segmentation and recommender systems.
- Challenging metrics: Accuracy > 0.81 (Top 2%) is expected.

[1] Kaggle Competition: https://www.kaggle.com/competitions/spaceship-titanic/overview

3. AI AGENT ECOSYSTEM

3.1. Multi-Agent Architecture

The ecosystem of our **Autonomous Data Scientist** is composed of three main agents, orchestrated by a coordination layer and a resource layer that guarantees computational efficiency.



3.2. Roles and Responsibilities of the Agents

o1 Agent (Principal Agent)

- **Specialization**: ML Model Generation and Optimization
- **Functions**: Automatic selection of architectures and features
- **Key metric**: Model accuracy and efficiency

Claude Agent (Support Agent)

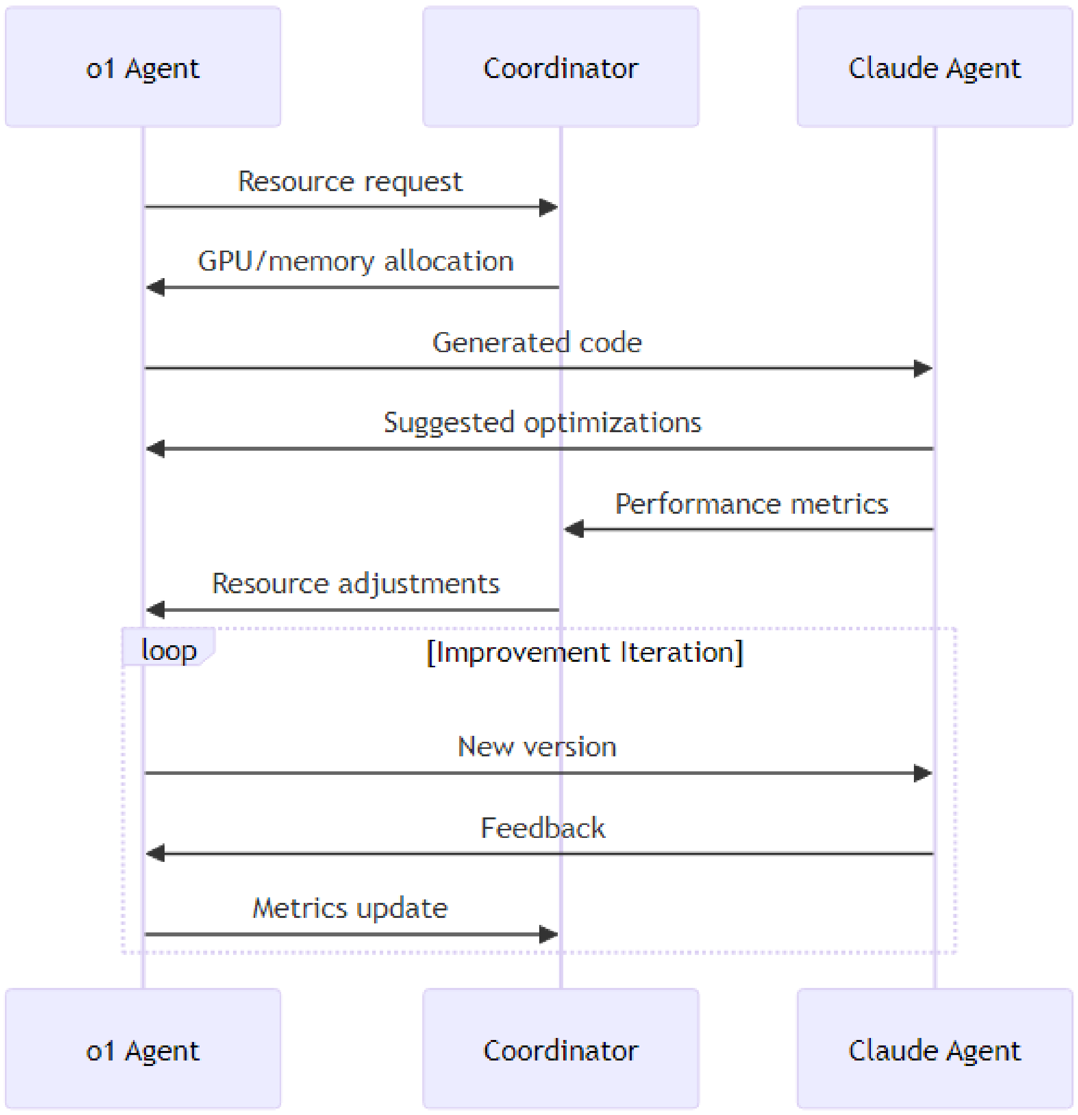
- **Specialization**: Performance Optimization and Debugging
- **Functions**: Hyperparameter tuning and error correction
- **Key metrics**: Error reduction and efficiency improvements

Coordination Agent

- **Specialization**: Orchestration and Resource Management
- **Duties**: Task distribution and resource control
- **Key metrics**: Resource utilization and scalability

3.3. Information Flow between Agents

Working together is done through a **continuous improvement loop**:



3.4. Advantages of the Multi-Agent System

- **Specialization**: Each agent masters specific tasks.
- **Redundancy and Reliability**: Multiple layers of validation and failover.
- **Modular Scalability**: Possibility of adding agents with different specializations without affecting the core.

3.5. The Coordinator Agent

In the proposed ecosystem, the Coordinator Agent plays a key role in orchestrating processes and resources within the overall workflow. Unlike the other two agents (o1 and Claude), the Coordinator Agent is neither a language model nor an AI system focused on code analysis or correction; its main function consists of:

1. **Supervision and Execution Control** - Triggers code generation, iteration and validation cycles. - Monitors execution times, detecting and managing possible timeouts. - Controls the number of submissions (results) and generated files (solution.py, results.csv, progress_report.json).

2. **Error and Feedback Management** - Listens to the output of each execution for critical errors (Traceback, TypeError, etc.). - Requests corrections from Agent Claude when bugs are detected that prevent proper execution. - Decides to apply performance improvements when training exceeds the time limit.

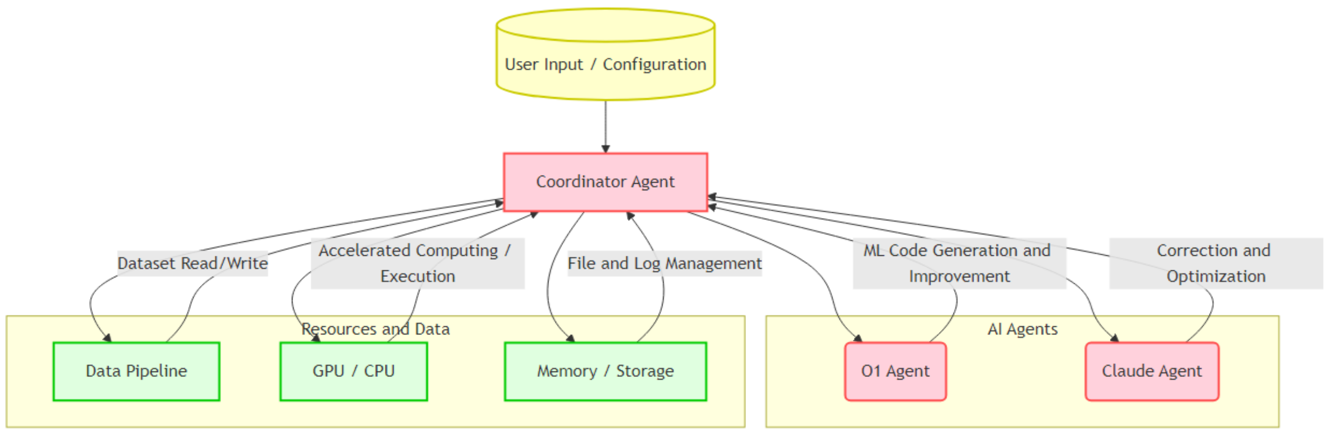
3. **Resource Allocation and Optimization** - Coordinates GPU and CPU usage, as well as available memory. - Manages intermediate files and reports to maintain a clear record of iterations. - Allows adding new resource gateways or services (e.g., distributed storage or multiple GPUs) without altering the central flow.

4. **Iteration Logic** - Determines how many times the code should be regenerated or refined (e.g. 50 iterations). - Uses previous results information (progress_report.json) to request improvements from Agent o1. - Structure the working folder, moving older versions of the code to specific directories (e.g., older_solutions).

This coordination layer is often implemented as a set of routines or scripts that:

- They call language models (Azure OpenAI, Anthropic Claude) to generate, correct or refine the code.
- They execute the solutions (solution.py) and capture outputs and errors in real time (subprocess.Popen).
- They detect bottlenecks (e.g. prolonged training) and ask for optimization suggestions.

The following diagram illustrates how the Coordinating Agent is placed in the flow:



With this approach, the Coordinating Agent need not itself be an AI system; it is sufficient for it to act as a "workflow manager" that communicates with the two specialized agents (O1 and Claude), sending them requests and taking advantage of their responses to direct execution automatically. Much of this control is reflected in the code:

- Define environment variables and prepare credentials for AI APIs (Azure OpenAI, Anthropic).
- It controls the main for cycle with a configurable number of iterations, where ML solutions are generated and refined.
- It handles the files (solution.py, submission.csv, etc.) and logs all output to execution_logs.txt.
- Invoke get_claude_fix when detecting specific errors, and get_timeout_improvement when it exceeds the previous time.

In this way, the Coordinating Agent becomes the key that enables seamless interaction between the code generators and the model training/validation machinery, ensuring an iterative process that is continuously improved without requiring manual intervention.

Interaction between Agents o1 and Claude Sonnet 3.5

In our multi-agent system - composed of o1 Agent (hereinafter "o1") and Claude Sonnet 3.5 (hereinafter "Claude"), coordinated by the Coordinator Agent - the workflow is based on an iterative loop involving the following stages:

1. **Initial Code Generation (o1)** - o1 analyzes the problem description, data sample and performance specifications to produce a preliminary machine learning script ("solution.py"). - This phase includes configuration and basic data preprocessing, declaration of the ML model and generation of the essential files (submission.csv, progress_report.json).

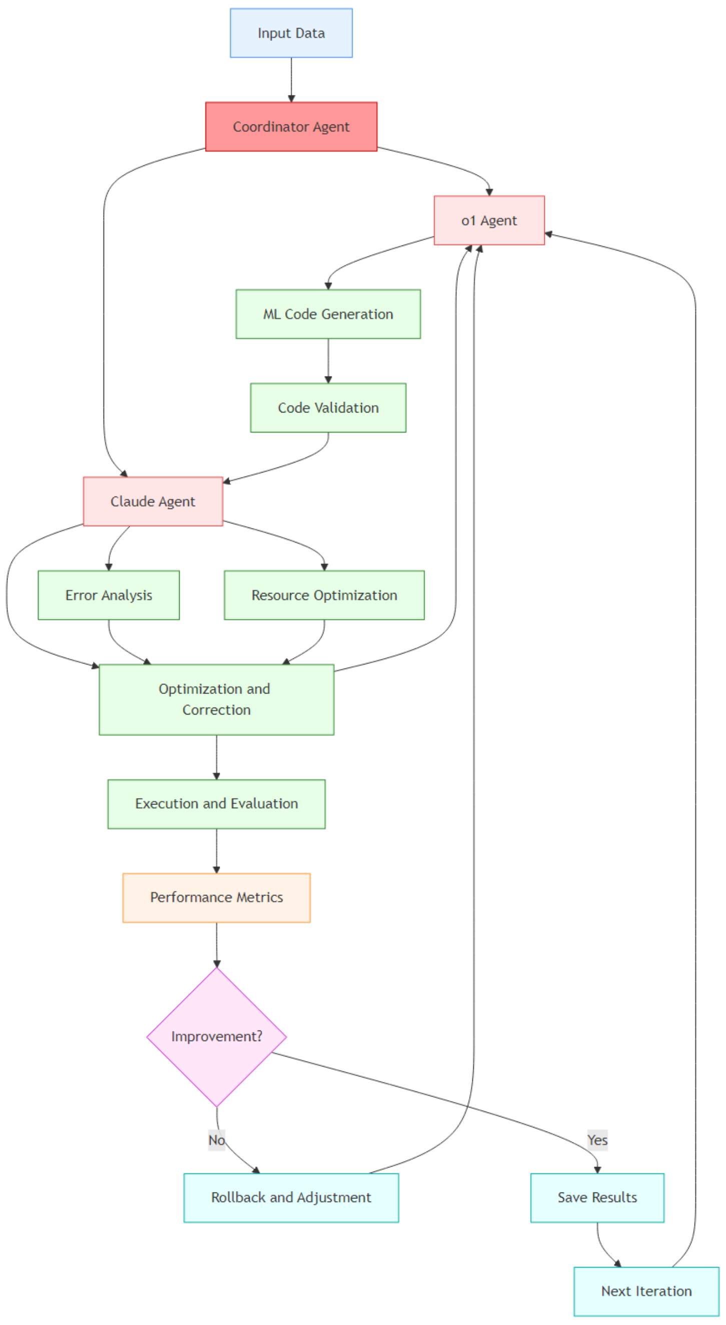
2. **Evaluation of the Script by the Coordinating Agent.** - The Coordinating Agent executes the script with a set time limit (e.g. 30 minutes). - Standard output (stdout) and possible errors (stderr) are captured. - If the script is successfully executed and generates results, they are stored in files as progress_report.json.

3. **Corrections and Adjustments (Claude)** - When critical errors or warnings appear that prevent proper completion (e.g. NameError, ValueError, etc.), the Coordinating Agent calls Claude. - Claude examines the code block and proposes timely fixes (e.g., adding imports or modifying hyperparameters). - If the execution exceeds the timeout, Claude may also suggest performance optimizations to reduce model complexity, improve data handling, etc.

4. **Iteration and Feedback** - With each iteration, the Coordinating Agent updates the history, moving the files to an "older_solutions" directory and repeats the cycle, asking o1 to refine the model based on previous results (e.g. new cross-validation metrics). - Claude intervenes again if new errors arise.

This flow ensures that o1 and Claude collaborate synergistically, where:

- o1 concentrates on the "creativity" of the model and the overall code generation.
- Claude acts as a "bug checker" and "optimizer" of the solution, correcting details and proposing micro-optimizations.



In this diagram, it can be seen:

- The Coordinator Agent as the central hub that obtains user or system input (e.g., number of iterations, maximum execution time) and distributes the tasks to the specialized agents.

- o1 Agent produces the machine learning code base at every step, leveraging data and hardware information.

- Claude Sonnet 3.5 needs to see the code and error messages to provide immediate corrections, ensuring that the system recovers from missing dependencies, syntax errors, etc.

In section 5 (Architecture), this section is complemented by including the global diagram showing the three agents (o1, Claude and Coordinator), the resource layer and the interaction with the produced files. In this way, the information trajectory is clear:

1. The Coordinator sends a request to O1 to produce or update the script.
2. The script is run in the local or remote environment (GPU/CPU, data).
3. In case of error, Claude receives the code along with the error message and provides a corrected version.
4. The Coordinator, satisfied with the execution, monitors the results and repeats the cycle if necessary.

In short, **this multi-agent system allows to iterate and improve machine learning solutions almost autonomously**, validating different approaches and fine-tuning the configuration until high levels of performance are achieved in a fully automated cycle.

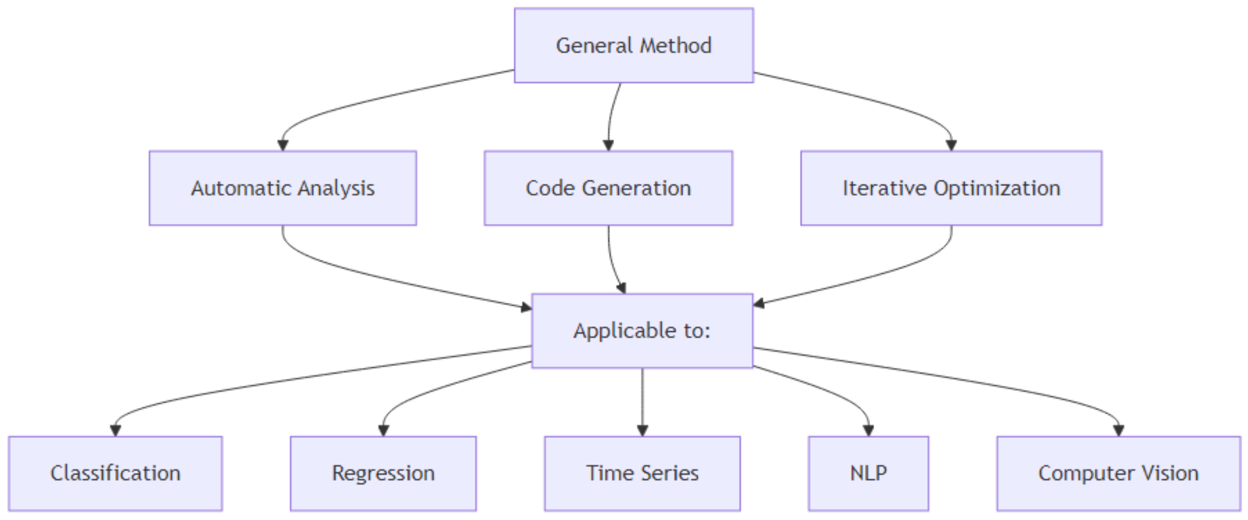
4. METHODOLOGY

4.1. Generalization of the Method

Our multi-agent approach can be applied to different types of machine learning problems (classification, regression, time series, NLP, computer vision, etc.) thanks to its ability to:

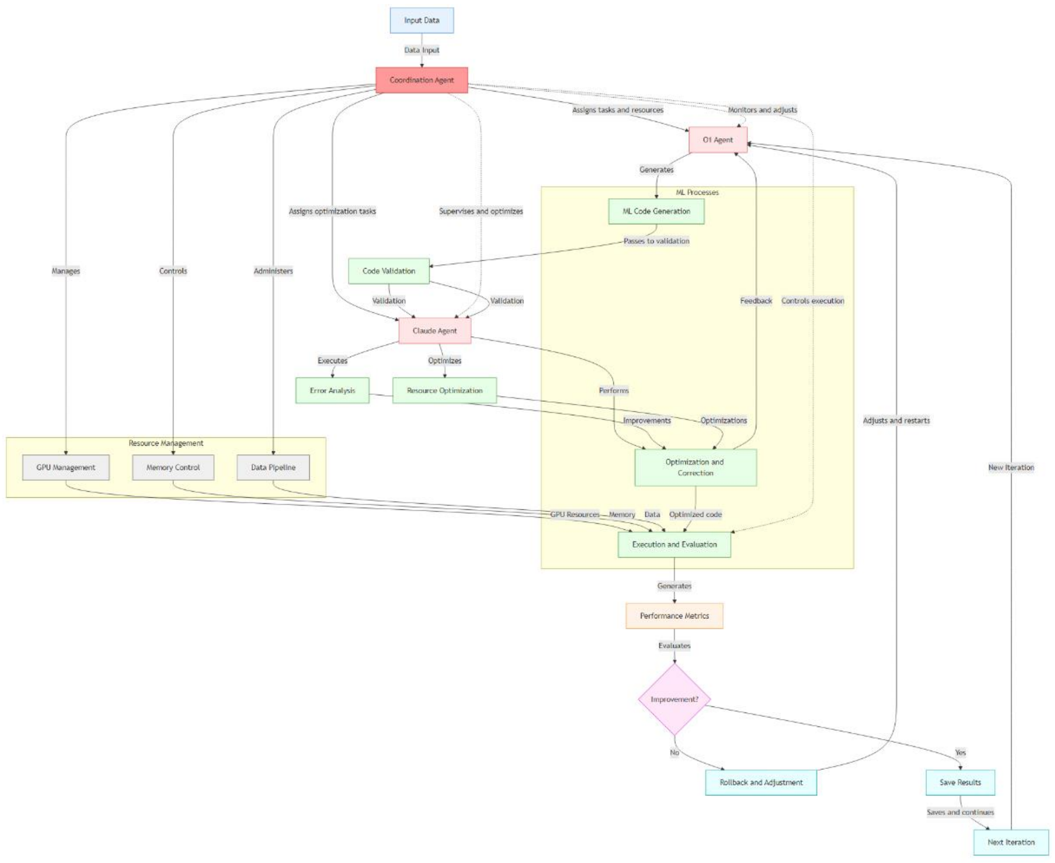
- **Automatic Analysis**: Identify the relevant characteristics and the type of model to be used.
- **Code Generation**: Create training pipelines and scripts autonomously.
- **Iterative Optimization**: Adjust hyperparameters and architecture continuously.

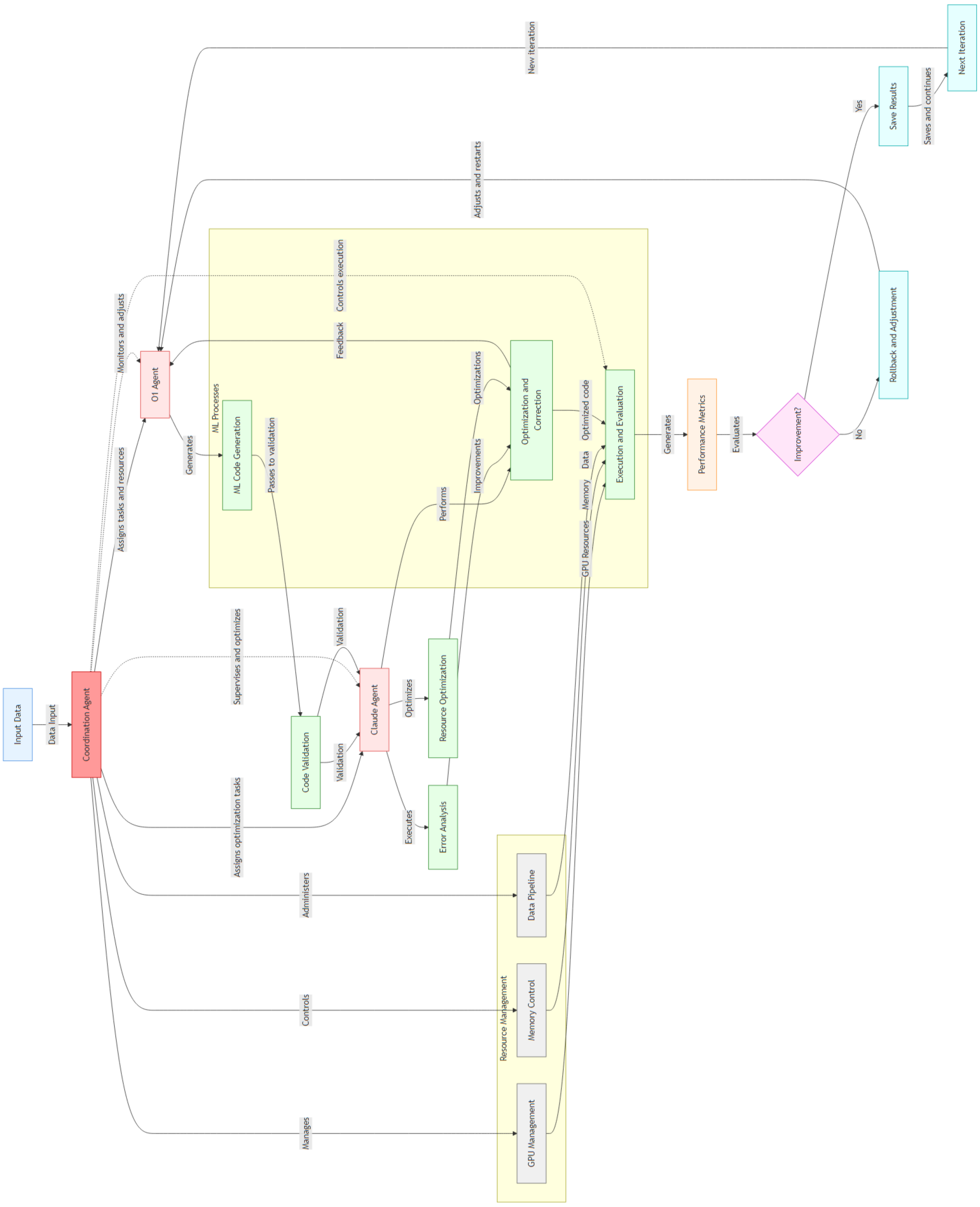
The following representation illustrates the versatility of our method:



5. SYSTEM ARCHITECTURE

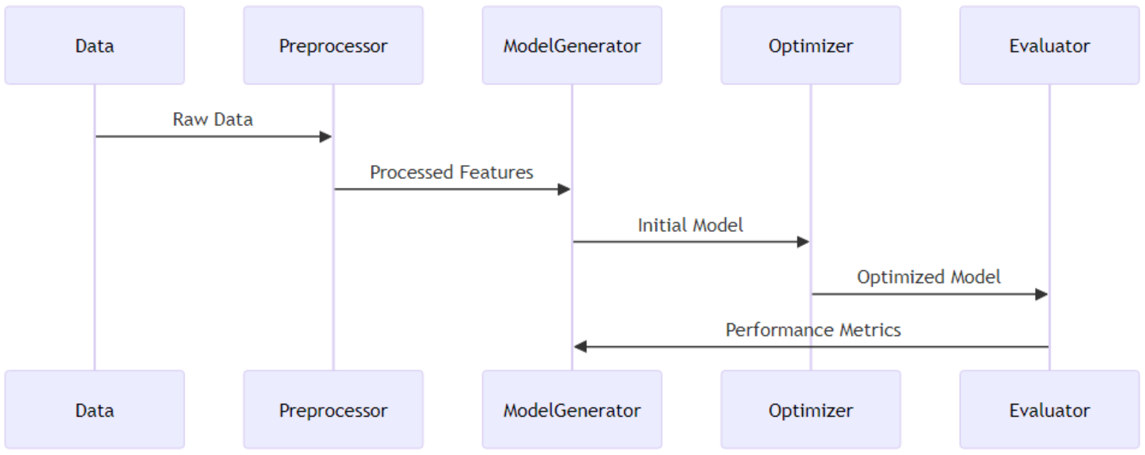
The overall architecture combines the multi-agent ecosystem with a staged workflow of preprocessing, training, validation and optimization:




Input Data
Data Input
Coordination Agent
Assigns tasks and resources
Monitors and adjusts
O1 Agent
Generates
ML Processes
ML Code Generation
Passes to validation
Controls execution
Feedback
Optimizations
Improvements
Performs
Optimization and Correction
Optimized code
Data
Memory
GPU Resources
Execution and Evaluation
Generates
Performance Metrics
Evaluates
Improvement?
Yes
No
Save Results
Saves and continues
Next Iteration
New iteration
Rollback and Adjustment
Adjusts and restarts
Supervises and optimizes
Assigns optimization tasks
Code Validation
Validation
Validation
Claude Agent
Optimizes
Executes
Resource Optimization
Error Analysis
Administers
Controls
Manages
Resource Management
Data Pipeline
Memory Control
GPU Management

6. MAIN COMPONENTS

- **Multi-Agent System**: - o1 Agent, Claude Agent and Coordinator working together.
- **Processing Pipeline**:



- **Automated Optimization**: - Model generation with dynamic cross-validation. - Resource optimization (GPU, memory). - Continuous monitoring of performance metrics.

7. RESULTS AND VALIDATION

7.1. Analysis of Prediction Distributions

Evolution of Predictions:

Submission	True	False	Observation
1	49.87%	50.13%	Near-perfect balance
2	51.11%	48.89%	Slight positive bias
3	53.43%	46.57%	Increasing bias toward True
4	52.26%	47.74%	Moderate bias correction
5	52.15%	47.85%	Final stabilization

7.2. Performance and Efficiency Analysis

Key Performance Metrics:

- **Total Execution Time**: ~15 minutes
- **Number of Optimum Iterations**: 5
- **Position in Kaggle**: Top 2% (>98% of competitors). Top 10% at first iteration.

Cross Validation Results (5 folds):

Fold	Accuracy
1	80.16%
2	80.34%
3	80.67%
4	81.07%
5	81.60%

Iteration Efficiency Analysis:

Iteration	Time (min)	Complexity	Relative Improvement
1	1.0	Download	Baseline
2	1.5	Media	+1.2%
3	2.0	Media	+0.8%
4	3.5	High	+0.3%

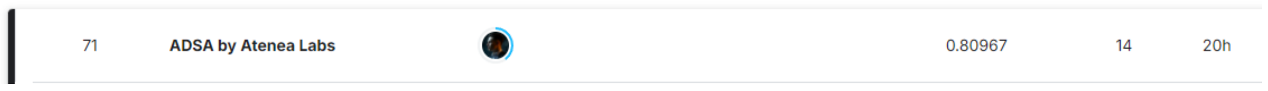
5	7.0	High	+0.1%

Observations on the Limit of Iterations:

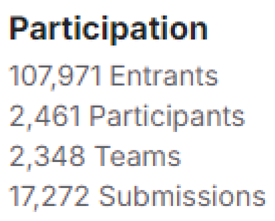
- Computational complexity increases significantly after the 5th iteration.
- Marginal return on improvement decreases drastically
- An optimal balance between time and performance is observed.
- In the future, the system will be able to iterate more times and with better results, since **we are using the first generation of reasoning models**.

7.3. Result in the competition

With submission number 5, **we achieved 71st place in the competition out of 107,971 registrants and 2,461 active participants during the last two months.**



71	ADSA by Atenea Labs	0.80967	14	20h



Participation
107,971 Entrants
2,461 Participants
2,348 Teams
17,272 Submissions

This means that ADSA, in its simplest version, is in the top 2.8% of data scientists in the period November 15, 2024 - January 15, 2025.

7.4. Efficiency and Scalability Analysis

Key Efficiency Factors:

- ✓ Early feature optimization
- ✓ Efficient process parallelization.
- ✓ Algorithmic complexity control.

Limitations Identified:

1. Exponential increase in post-iteration complexity 5
2. Non-linear processing time
3. Consumption of computational resources

Optimization Strategy Adopted:

- Focus on high quality early iterations
- Prioritization of significant improvements
- Balance between execution time and performance

8. TECHNICAL INNOVATIONS

Multi-target optimization

- **Accuracy**: Maximizing performance metrics
- **Efficiency**: Optimization of computational resources
- **Generalization**: Ability to adapt to new data

Auto-Correction System

- **Detection**: Continuous error monitoring
- **Correction**: Automatic adjustments in real time
- **Validation**: Verification of implemented changes

9. INFRASTRUCTURE AND OPTIMIZATION

Hardware Used

Component	**Specification**
GPU	NVIDIA GeForce RTX 3080 (12GB VRAM)
CPU	AMD Ryzen 5600
Memory	32GB DDR4 RAM
Storage	NVMe SSD

Performance Metrics

- **Throughput**: ~1000 samples/second
- **Time per iteration**: <30 minutes
- **Standard deviation**: ~1% in folds

10. APPLICATIONS AND FUTURE

Current Use Cases

1. **ML Competitions**: High Performance in Kaggle
2. **Business Models**: Optimization of large volumes
3. **Research**: Automatic architecture search

Upcoming Developments

- **Meta-learning**: Rapid adaptation to new domains
- **Cloud Scaling**: AWS, GCP, Azure
- **Deep Learning**: Transformers and generative models

11. ETHICAL AND LEGAL CONSIDERATIONS

Fundamental Principles

- **Privacy**: GDPR compliance and anonymization
- **Transparency**: Detailed documentation
- **Responsibility**: Prevention of bias
- **Supervision**: Periodic audits

12. CONCLUSIONS

Atenea Labs' ADSA multi-agent system represents a **revolutionary breakthrough** in the automation of machine learning projects, standing out for its ability to achieve exceptional results in remarkably reduced times:

Major Achievements

- Positioning in the **top 2%** of Spaceship Titanic in only **15 minutes** of autonomous execution.
- Top 10% at the first iteration in 1 minute.
- Consistent accuracy **>80%** in cross-validation
- **Modular** and **adaptive** architecture
- **Continuous and autonomous** improvement system

Efficiency and Optimization

- **Development Time**: Reduction from weeks to minutes
- **Optimal Iterations**: 5 iterations balancing performance and efficiency
- **Automation**: Completely autonomous process from analysis to implementation

Industry Impact

- **Democratization of ML**: Access to high-level solutions for more professionals
- **Raising the Standard**: Overall improvement in the quality of ML solutions
- **Operational Efficiency**: Significant reduction in development times

Future Economic, Social and Technological Impact

The success of the system suggests a future where autonomous AI tools will raise the overall level of data science. While this could lead to increased competition on platforms such as Kaggle, the real benefit lies in **democratizing access to high-quality ML solutions**, enabling more organizations and professionals to harness the power of machine learning effectively and efficiently.

These conclusions are intended to go beyond a simple technical closure and place this multi-agent system at the crossroads of the economic, scientific and social future in which humanity finds itself.

The advances described here not only delineate a new frontier in machine learning automation, but also envision a paradigm shift in the way of conceiving data science and its impact on humanity. One of the most striking observations of this multi-agent system is the existence of a saturation point: after the fifth iteration, the increase in complexity does not provide tangible improvements in performance, but it does generate a sharp increase in computational cost. **We should be aware that we are dealing with the first generation of reasoning models**, and in spite of this, with only five iterations - performed in approximately fifteen minutes - the system manages to consistently reach a place within the top 2% in the competition analyzed, **demonstrating a ratio between efficiency and effectiveness that is difficult to match with other current methods**.

Now, the possibility exists to further refine the model through more iterations or more complex architectures; however, there is currently a trade-off where the marginal benefits do not outweigh the increase in time and resources. Put another way, **the clear dominance of the initial phase (those first five iterations) demonstrates how valuable a fast generation-validation-tuning cycle is and teaches that, in many cases, well-targeted simplicity can overcome uncontrolled complexity.**

On the other hand, **we cannot ignore the democratizing imprint that this system embodies**. Staying in the top 2% is a remarkable achievement, but **if this technology were to become commonplace for the data science community, the gap between the top teams and the rest would tend to narrow. A "leveling up" would then take place, where excellence would become the norm rather than the exception. Far from threatening competitiveness, this dynamic would encourage innovation: large corporations and small startups alike would have cutting-edge tools at their disposal to create disruptive solutions.**

In terms of economic and social impact, the potential is difficult to overestimate. Tools like this could reduce barriers to entry for new developers and lower the costs of large-scale research and development, acting as catalysts for new business opportunities. With them, **humanity is heading towards an era of hyperproductivity, where the convergence of human talent and autonomous systems lays the foundation for exponential and sustained economic growth.** The future lies in the massive and responsible adoption of these technologies, in the training of professionals capable of understanding their limits and opportunities, and in the institutionalization of ethical and collaborative practices that ensure that their benefits are extended to the greatest possible part of society.

The race to squeeze the full potential of artificial intelligence has only just begun. On the horizon, an increasingly competitive and sophisticated ecosystem is on the horizon, where the best results would be a matter of smarter iterations, not always longer. As similar systems are appropriated by multiple sectors, **the speed and quality of innovation will be exponentially enhanced, lifting humanity to heights of scientific and economic prosperity unimaginable until just a few years ago.**

In short, these five iterations, seemingly modest but executed in mere minutes, represent an emblem of the power and agility of modern AI: a prelude to the transformations to come.