Human-Centered Agentic Framework for Machine Learning Modeling in Finance

1st Izunna Okpala
Emerging Capabilities Research Group
Discover Financial Services Inc.
okpalaiu@mail.uc.edu

2nd Ashkan Golgoon
Emerging Capabilities Research Group
Discover Financial Services Inc.
agolgoon3@gatech.edu

3rd Arjun Ravi Kannan
Emerging Capabilities Research Group
Discover Financial Services Inc.
arjun.kannan@gmail.com

***Abstract*—The advent of large language models has ushered in a new era of agentic systems, where artificial intelligence programs exhibit remarkable autonomous decision-making capabilities across diverse domains. This paper explores agentic system workflows in the financial services industry. In particular, we build agentic crews with human-in-the-loop orchestrator that can effectively collaborate to perform complex machine learning modeling tasks. The modeling crew consists of a judge agent and multiple agents who perform specific tasks such as exploratory data analysis, feature engineering, model selection/hyperparameter tuning, model training, model evaluation, and writing documentation. We demonstrate the effectiveness and robustness of modeling crews by presenting a comparative experiment applied to the detection of credit card fraud and card portfolio credit risk. Our fraud detection experiment achieved a recall of 81.6% and an F1-score of 88.9%, outperforming AutoML's 70.4% and 82.1%, respectively.**

***Index Terms*—Large Language Models (LLMs), Multi-Agent Systems, Human-in-the-Loop (HITL), Agentic Systems, Multi-Agent Debate, Multi-Agent Collaboration**

I. Introduction

This paper isolates and elaborates on the modeling aspect of the broader framework introduced in [1], offering a more detailed analysis. Agentic programming has emerged as a powerful framework in natural language processing, capable of generating and understanding textual instructions and performing tasks to mimic human behavior. One of the interesting applications of agentic systems is their ability to engage in role-playing, where they can simulate various personas, perspectives, or even multiple roles within a conversation [2], [3].

Recent research in LLM-based multi-agent systems has demonstrated considerable potential by equipping collaborative agents with specialized tools, resulting in advanced problem-solving skills in different domains [4], [5].

We propose a human-centered approach that enables direct orchestration by a human expert - requiring minimal effort for machine learning modeling in finance. This framework addresses potential concerns that may arise when agents operates autonomously—despite its ability to generate accurate results. This is particularly critical in financial services, where decision-making must be based on accurate and trustworthy data. We integrate **agents** with large language models (LLM) as their knowledge-bank, enable implicit prompts (**goals**) that enhance agent's functionality including the definition of its *persona*, *memory stack*, and *tools*. Each agent is connected to a specific **task**, independently defined to outline in detail the various functions needed to be performed. The **tools** handle dynamic requests, including actions that require internal and external services; allowing agents to interact with various data sources, analytical procedures, and computational resources.

The proposed framework facilitates a hierarchical arrangement of events across agents, tasks, and tools using an important component called the **planning module** (see Figure 1). This organization enables agents to produce results, receive feedback, think through instructions, track changes, and make informed decisions based on a comprehensive stack of interconnected operators. The experimentation in §IV-D closely matches traditional machine learning tasks, tailored to specific stakeholder needs. This approach offers a scalable and cost-effective way for machine learning (ML) modeling and testing. As illustrated in Figure 5, we introduced a judge agent whose primary role is to function as a "third eye." This agent serves as an additional layer of oversight, ensuring that critical aspects of effective machine learning modeling are not overlooked by the human expert.

II. Related Works

Key research areas utilizing agentic systems in finance include trading and investment agents [6]–[8], the simulation of market and economic activity [9], [10], financial sentiment analysis [11], audit and compliance automation [12], anomaly detection [13], and stock predictions [14]. TradingGPT [6] introduced a multi-agent system with layered memories used for stock trading. In this framework, an agent assigns objects to *long-term*, *middle-term*, or *short-term* memory layers. Improving on [3]'s metrics for *recency*, *relevancy*, and *importance*, [6] models a hierarchical arrangement of events within each memory layer and within an agent's memory. The way they handle memory enables the agents to effectively debate, form strategies, track financial changes, and make informed investment decisions based on their individual risk appetite.

The safety of multi-agent systems is important, especially when it involves the financial services industry. As [15] would put it, harms in agentic systems lead to systemic and long-range impacts, as well as undermining collective decision-making power. They noted the research on FATE (Fairness, Accountability, Transparency, and Ethics) [16], [17], which suggests that as programmable systems become more agentic, they may amplify biases and inequities, particularly for marginalized groups. Some other challenges, not necessarily related to harm and safety, were elucidated by [18]. These include, but are not limited to, the following: optimizing task planning, managing complex context information, and improving memory management. Identifying the challenges of LLM-powered multi-agent systems and their potential solutions is of paramount importance in ensuring the safety and compliance of agentic systems. Some of these challenges include error handling techniques and/or system failures leading to unpredictable behavior like hallucination, lack of control and oversight over the input variables that shape the decision of the system, induced preference or bias, toxic degeneration, and difficulty assessing how agents reach their conclusions.

One of the ways to tackle this issue is the introduction of *human-in-the-loop* (see §IV-A). This is the ability for humans to intervene or be a part of the processing capabilities of agentic systems. This approach, when applied properly, protects end-users from biased or incorrect results [19]. Another way to tackle the issue of safety and harm in agentic systems is the use of other *guardrail* strategies. Guardrails are a set of rules that ensure operational safety and ethical practices in machine learning applications. Their implementation can be in the form of a layered protection model, system prompts, retrieval-augmented generation (RAG) architectures [20], and other techniques that minimize bias and protect privacy [21]. Guardrails reduce the likelihood of issues like bias, potential for unsafe actions, dataset poisoning, lack of explainability, hallucinations, and non-reproducibility [21]. [19] also argue in favor of guardrail inclusion in LLM-based systems in addition to their position on transparency to ensure the their runtime behavior is safe and responsible. Ultimately, guardrails are a crucial aspect in harnessing the immense potential of LLM while minimizing harm and ensuring their alignment with human values.

The subsequent sections are organized as follows: In §III, we provide a brief overview of the components of agentic systems and their collaborative strategies. The applications of agentic systems in financial services are discussed in §IV with agentic systems for the modeling given in §IV-C.

III. Agentic systems architectures

In this section, we briefly review some important elements of the agentic system architecture and its general components (see [22]–[24] for more details).

A. Agentic Systems Components

Agentic systems utilize LLM as their knowledge-bank and are equipped with predefined functions to create a plan for a given task, collaborate with one another, and leverage a wide variety of tools to execute the plan [22]. An agent is typically characterized by the following general components, namely the agent core, memory module, tools, and planning module, though it should be noted that there is no general consensus on the definition of agent components in the literature. (see Figure 1).

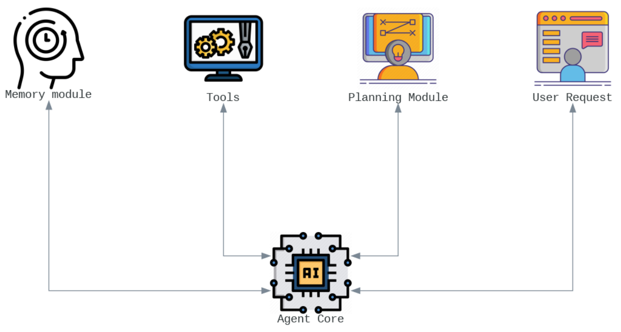

Fig. 1: General components of an LLM-based agent (adapted from [22])

The *agent core* contains information about the core NLP engine (such as GPT), the agent's goals, tools, memory, and persona.

The *memory module* consists of short-term and long-term memories. The short-term memory tracks immediate context and actions, while the long-term memory stores information across multiple prior sessions, enabling more personalized interaction to be provided by the agent [22].

Tools are external systems and workflows, APIs, and specialized functions that agents can leverage to perform tasks. These tools allow the agent to interact with the outside world, access real-time data, perform computations, or control systems. Some examples of agent tools are retrieval-augmented generation (RAG) tools [20] to enable extracting contextually relevant information (context), web browsing and scraping tools, third-party integration tools (e.g. weather, finance or social media APIs), computation, code execution, and interpreter tools, etc. (see [24]).

In agentic systems, a *planning module* is responsible for managing the decision-making and task execution process by breaking down complex tasks into manageable steps. In other words, the planning module acts as a task orchestration engine that manages how an agent handles multistep, goal-oriented tasks. In doing so, a combination of two techniques, namely task (and question) decomposition, as well as reflection (or critique), is used [22]. Task decomposition is used to break down a complex task into smaller (more manageable) subtasks. The reflection or critic mechanism plays a key role in improving the agent's decision-making, planning, and reasoning processes. Several techniques like *ReAct* [25], *Reflexion* [26], *Chain of Thought* [27], and *Graph of Thought* [28] have emerged as methods for augmenting the planning process by introducing reflective or evidence-based approaches. These techniques enhance the agent's reasoning capabilities by enabling it to reflect on its own actions, evaluate possible outcomes, and refine its execution plans, resulting in handling tasks with greater accuracy and efficiency.

Next, we focus on CrewAI [24], a multi-agent orchestration framework used for defining agent components. The key components include *Role Playing*, *Focus*, *Tools*, *Cooperation*, *Guardrails*, and *Memory*.

The memory helps agents recall, reason, and effectively learn from past events and interactions [24]. The memory system consists of *short-term* memory, *long-term* memory, *entity* memory, and *contextual* memory. *Role playing* is a specific identity assigned to an agent within the CrewAI system. This provides context and direction, influencing how the agent interacts with other agents and tools. The *Focus* component gives the agent the ability to concentrate on its assigned tasks without being distracted by irrelevant information or activities. The agent is thus able to execute its prompts, enabling the prioritization of its efforts on specific tasks. It connects to the role-playing component, which streamlines the agent to a particular function irrespective of the prompts within the agent's construct.

For agents to work effectively, especially when there are specialized actions like exploratory data analysis that need to be performed, *tools* are used [24]. *Tools* are the capabilities that agents can utilize to accomplish specific tasks. The selection of appropriate tools is vital, as providing agents with too many options can lead to confusion and inefficiency. *Guardrails* are safety measures and protocols implemented to ensure that agents operate reliably and ethically. These guidelines help prevent issues such as hallucinations (incorrect output) and ensure that agents adhere to best practices during their interactions. One of the components that drives this action is the LLM temperature. A temperature setting of '1' allows the LLM greater freedom to generate creative or less accurate responses, while a temperature of '0' restricts it to deterministic outputs, eliminating such flexibility. It is always a good practice to evaluate trade-offs to ensure agents perform optimally. In our case, we selected a moderate temperature of '0.2' to balance creativity and precision. Text generation spans the entire vocabulary without the temperature parameter. For instance, Llama3 can go through all 128,256 tokens, Deepseek-R1 spans 130,000 tokens, and GPT-3.5 Turbo reaches 100,256 tokens. A higher or no temperature allows for more variability. Using a low temperature setting means selecting tokens with the highest softmax scores. The temperature parameter controls the impact of this sampling. Specifically, it modifies the softmax function to adjust the distribution of token selection. In Equation III.1, x_i is the raw score (logit) for class i, T is the temperature parameter, and j is the index that iterates over all classes in the set of possible classes.

$$Temperature - adjusted\ softmax \ = \frac{e^{x_i/T}}{\sum e^{x_j/T}} \qquad \text{(III.1)}$$

For additional context on our guardrail strategy, refer to §IV-A. The *Cooperation* component is arguably one function that makes CrewAI unique. It involves the collaborative efforts of multiple agents working together to achieve common goals. Agents can share information, delegate tasks, and provide feedback to one another, enhancing the overall effectiveness of the system [24].

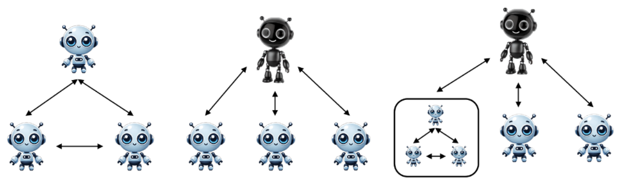
Fig. 2: Agentic system collaboration structure: Horizontal Collaboration (*left*), Hierarchical Collaboration (*middle*), Nested Collaboration (*right*) (adapted from [18])

B. Collaboration Strategies in Agentic Systems

Collaboration in agentic systems enables agents to assist one another by sharing information and integrating their skills. In CrewAI [24], this collaboration is realized utilizing *information sharing*, *task assistance*, and *resource allocation*. Effective information sharing is essential to ensure that all agents can communicate their findings and stay well-informed. Task assistance provides the opportunity for agents to ask for help from other agents that possess specialized skills for a task. Finally, resource allocation is responsible for the efficient allocation of computational resources among agents to optimize task execution.

There are multiple collaboration structures in a multi-agent framework based on agent functionality and their interactions, such as *equi-level or horizontal collaboration*, *hierarchical or vertical collaboration* and *hybrid or nested collaboration* [4], [18] (see Figure 2). In horizontal collaboration, each agent has its own role and strategy, with no agents having a hierarchical advantage over the others. Agents with similar goals collaborate, while agents with opposing goals negotiate or debate to collectively make decisions and complete the task [18]. In a hierarchical structure, a leader agent guides the follower agents to execute its instructions [18]. When both horizontal and vertical structures are present, a nested structure (or hybrid) is formed. Finally, the state of multi-agent systems, their collaboration strategy, agent roles, the number of agents, and their relations may evolve [18]. This scenario leads to *dynamic structures* in which agents may possess dynamically evolving configurations in order to adaptively react to external factors or dynamic conditions [18], [23].

IV. Applications to Financial Services

In this section, we provide an end-to-end agentic system implementation for credit card fraud and portfolio credit risk use cases. In particular, we build *modeling* crews and illustrate how the agents collaborate to perform their specialized tasks collectively.

The financial services industry is highly dependent on accurate modeling procedures for its predictive and decision-making capabilities. The goal is to streamline the modeling workflow and effectively manage dependencies, as well as collaboration among agents.

The system architecture for financial crews with human in-the-loop integration, memory property, and role-playing is discussed in §IV-A and §IV-B. Agentic workflows for modeling are discussed in §IV-C. We provide two modeling use cases to highlight the kind of task that can be accomplished with agentic systems *-credit card fraud detection and portofolio credit risk* in §IV-D. Our agents were powered by Llama3, Deepseek-R1, and GPT-3.5 Turbo. CrewAI served as the foundational framework for building and testing the agentic system, as illustrated in §IV-C and §IV-D.

A. Agentic Human-in-the-Loop (A-HITL)

The mind map presented in Figure 3 illustrates a human expert as the system orchestrator. The tree shows a human expert overseeing the modeling crew and providing instructions to help them achieve their respective objectives. The expert provides feedback to agents in addition to their predefined description, in a case where the agent cannot figure out how to solve the problem. Agentic systems without human orchestration can exhibit abnormal behavior due to the inherent challenges associated with large language models (LLMs), particularly when handling complex tasks. Issues such as hallucination and a tendency to focus on isolated segments of text (often referred to as text chunking) can impede overall context.

To address these challenges, we first adopt a minimal temperature parameter at the LLM level, so that text generation focuses on tokens with higher softmax (see Equation III.1). We also make use of the methodical guardrail strategy in CrewAI, which establishes a clear distinction between Agents and Tasks. Each Agent has a defined persona and is accompanied by a backstory that details its specialization. The Task module specifies the actions to be executed and outlines the expected output, thereby minimizing deviations from the primary goal defined within the Agent's construct. This expected output not only serves as a benchmark, but also prompts the Agent to refine its response, ensuring that the initial output is not treated as final. Given that our program utilizes code execution, we have implemented a custom tool to manage standard output from these executions and log all actions, including errors and completions. This enables us to verify that any generated code runs without problems. However, there remains a 1-10% chance that the output may not align with expectations, which highlights the importance of incorporating a human-in-the-loop. In equation IV.1, we demonstrate the procedure that requires human feedback $h(p)$. The original prompt within the agent's construct is represented as $u(p)$, $x(p_i)$ are the inputs, while $y(p)$ is the output.

$$y(p) \ = f(x(p),\ u(p),\ h(p)) \tag{IV.1}$$

Considering the sensitivity of financial modeling and the need for thorough quality assurance, we propose that a human expert assume the role of system orchestrator rather than relying on an agentic orchestrator (see Figure 5). In this framework, the human expert oversees task delegation, guides Agents in correcting errors when they fail to do so autonomously, and provides additional suggestions to enhance the results produced by the Agents. Additionally, we introduced a judge Agent to review the actions performed by other Agents, offering insights and recommendations to the human expert. This allows the human expert to orchestrate the entire process effectively, ensuring timely feedback and corrections when necessary.

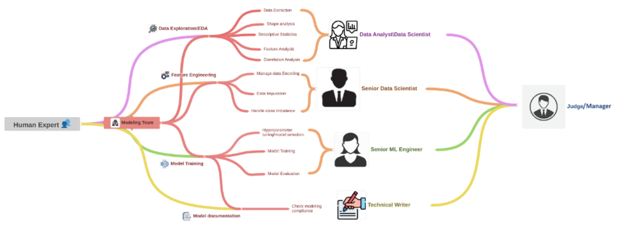

Fig. 3: Mind-map demo of the agentic system

B. Properties of our agentic system architecture

The proposed system comprises of several autonomous agents, each responsible for distinct tasks within the pipeline. The architecture is designed to promote modularity, allowing agents to operate independently and collaboratively. The modeling agents include:

- Data Extraction
- Exploratory Data Analysis (EDA)
- Feature Engineering
- Meta Tuning
- Model Training
- Model Evaluation
- Documentation Writer
- Manager/Judge

1) Memory, delegation and information retrieval: The memory property is most effective when individual agents store their inputs and outputs in memory, allowing for effective transfer of knowledge to the other agents. Since we have employed a human to oversee all processes, we introduced knowledge transfer into the Human-in-the-Loop (HITL) module using the context parameter inherited from the Task module. This allows each agent to access additional information, helping them understand previous actions in relation to their current tasks. These interactions, including inputs and outputs, are visually represented in Figure 4.

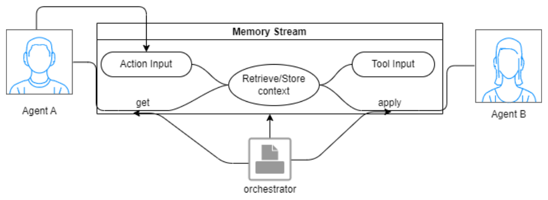

Fig. 4: Memory, delegation and information retrieval [1]

The memory stream is an object with a specific capacity and can hold task delegations in natural language, task execution timestamps, and the information needed by the collaborating agent. The core attribute of the memory object is the storage of interconnected interactions from different agents.

2) Role playing properties of the system: The "role-playing feature" makes the solution more intuitive, since specific roles or personas are assigned to each agent, guiding their behavior and decision-making within a collaborative task.

These roles represent specific job functions, such as data engineering or machine learning engineering, based on the requirements and objectives of the initialized agents. The first agent, for example, is a data analyst tasked with data extraction and splitting to avoid data leakage (see §IV-C for more details on the actions performed by this agent). We briefly touched on the role of the data scientist in §IV-B1. There are two data scientists: The goal of the first one is to conduct an in-depth exploratory analysis of the data provided. The second prioritizes feature engineering, with a particular emphasis on the creation of a preprocessor pipeline based on the extracted data properties. The interdependent functions of data scientists and other agents illustrate the need for collaboration and the value of clear and designated roles. The importance of the memory stream is emphasized here; different results are chained together to achieve the common goal of the entire crew.

C. Detailed description of agents

The agentic system, introduced in §I, features a modular architecture that takes advantage of the strengths of individual components to achieve a unified goal. Figure 5 provides a clear description of the systems and how tasks are segmented based on expertise. In this section, we discuss the workflow of our proposed agentic system applied to financial modeling. The distinct roles facilitate specialization (see §IV-B2). We provide detailed descriptions of these agents, their assigned functions, and the methods used to prompt them.

1) Data Extraction Agent: This agent functions as a data analyst, tasked with extracting data from external sources such as Kaggle or GitHub. Its primary responsibility is to split the data into training and testing sets to prevent data leakage. Additionally, it performs a mini-task of subsampling the training set for use by the meta-tuning agent in hyperparameter tuning and model selection. The agent utilizes a specialized tool, the "code execution tool," and is powered by GPT-3.5 Turbo.
2) EDA Agent: This agent specializes in exploratory data analysis, assuming the "role" of a Data Scientist. It employs the EDA Tool, designed to capture nuances that basic data exploration may overlook. It utilizes the Llama3 model as its LLM engine, enabling it to perform in-depth analyses. The agent's procedures include identifying missing values, detecting class imbalances, analyzing categorical variables, and pinpointing outliers. In addition, it provides insights into the data distribution of all features, capturing skewness and correlations.

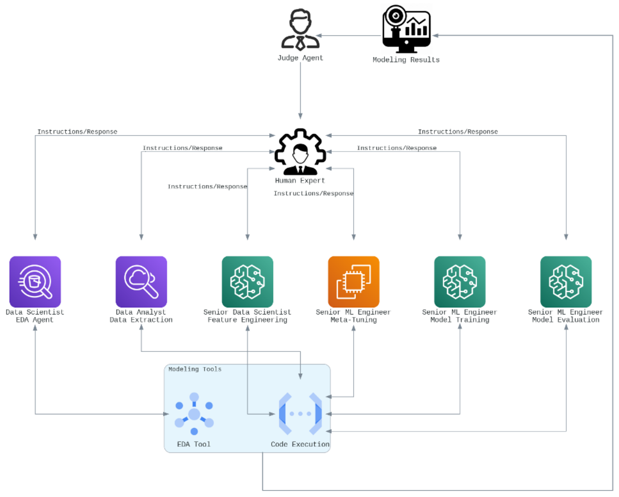

Fig. 5: Visual representation of the modeling crew

3) Feature Engineering Agent: This agent is responsible for creating a preprocessor pipeline based on the characteristics of the data. The preprocessing steps include KNN imputation for missing numeric features, feature normalization, ordinal encoding for categorical variables, and custom imputation for missing categorical variables. The LLM that powers the code generation for this agent is GPT-3.5 Turbo, and it assumes the role of "Senior Data Scientist." The feature engineering agent effectively manages missing values, encodes categorical variables, and normalizes numeric features.
4) Meta-Tuning Agent: This agent is responsible for identifying the optimal hyperparameters and performing model selection using the "code execution tool." It is configured to utilize the preprocessor created by the feature engineering agent to perform grid search on a select hyperparameters inputs. The subsample of the training data generated by the data extraction agent is used to improve processing speed. The agent assumes the "role" of a "Senior Machine Learning Engineer," and makes use of GPT-3.5 Turbo for its operations.
5) Model Training Agent: This agent is responsible for training the selected machine learning model and its optimal hyperparameters using the "code execution tool." It uses GPT-3.5 Turbo because of its code generation capability. There is also a provision within the prompt for saving the trained model to a pre-defined directory. The agent takes the same persona of "Senior Machine Learning Engineer" as the meta-tuning agent.
6) Model Evaluation Agent: This agent is responsible for evaluating the trained model using the test data. It uses the same LLM engine as the meta-tuning and model training agent. The generated code calculates the accuracy score, f1-score, precision, recall, and auc of the trained model using the test data from the "data extraction agent." This agent takes on the persona of a "Senior Machine Learning

Engineer."

7) Judge Agent: The judge takes the role of a "Manager," with the goal of examining how well its coworkers performed. The LLM that powers the judge task is the DeepSeek-R1 model because of its reasoning capabilities. It acts as a layer of security to support the human expert in making informed decision.
8) Documentation Writer Agent: This agent is responsible for creating technical documentation for all tasks performed by the modeling agents. With expertise in technical writing and a deep understanding of data science workflows, it utilizes the output instance from the HITL module to collect results from individual agents. This allows the agent to compile a comprehensive summary of all tasks within the agentic ecosystem, leveraging the capabilities of DeepSeek-R1.

Algorithm 1 Modeling Crew

```
Human → Agent:
Provide instructions to agent in [agents]

Agent Initialization:
  role ← [Data Scientist, ML Engineer...]
  goal ← Perform given functions
  tool ← [code_executor, eda_tool...]
  memory ← [short,long...]

Judge:
  role ← Manager
  goal ← Evaluate agent actions
  If execution completed:
    feedback ← assess(actions, codebase, results)
    feedback → human
```

D. Experiments and Results

We conducted an experiment with two practical use cases relevant to the financial services industry using the framework discussed in §IV-A. Figure 6 illustrates the process flow; how the agentic system operates in conjunction with various tasks and tools. The process, outlined earlier in Algorithm 1, demonstrates a hierarchical process. This gives the human expert the ability to delegate and manage the agents and tasks available to them for various functions. The available tools can be utilized by any agent or task based on the agent's persona and human expert directives.

1) Credit Card Fraud Detection Dataset: We present the performance metrics derived from parsing the credit card fraud detection dataset [29] through the agentic system and highlight several subtleties involved. The dataset contains 284, 807 rows and 31 columns. The column "Class," serves as the target variable for binary classification. This column identifies whether a transaction is fraudulent (represented by 1) or non-fraudulent (represented by 0). A notable characteristic of the dataset is its class imbalance, as 99.83% of the transactions were non-fraudulent. Apart from the target feature, the dataset consists entirely of 29 numeric features and a date feature. The numeric feature includes the "Amount" feature, representing the monetary value of each transaction. The remaining 28 features were anonymized, labeled "V1" to "V28." There were no missing values, categorical or text-based features. The hand-off and execution details for data extraction agent can be found in Log 1.

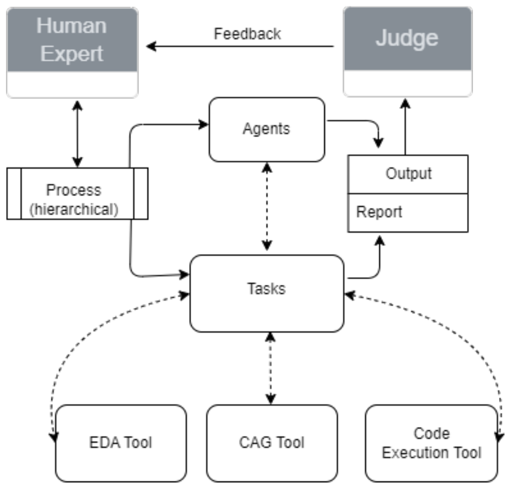

Fig. 6: Pictorial representation of the Human-Agent-Task-Tool integration [1]

Log 1: Human - Data Extraction - EDA Agent interaction

```
Human Interface
Task: Extract data from external source
Select the Agent to handle this:
1. Data Extraction Agent
.....
Agent number: 1

Working Agent: Data Analyst
Starting Task: Extract the data in the Kaggle - 'mlg
    ↪ -ulb/creditcardfraud.'
Human Feedback: Drop the Time variable, and split
    ↪ the original dataset using the 80/20 rule

Thought: I need to use the Code Executor tool.

Action: Python Code Executor Tool
Action Input: {"generated_code": "###"}

Final Answer:
The data has been successfully extracted and the
    ↪ data split into train and test sets.....

Human Interface
Human: Provide Feedback (type 'end' to stop): end
.....
```

The model trained using the agentic system presents good performance compared to the results obtained from H2O AutoML. A direct comparison with the most upvoted solution on Kaggle [30] was not feasible due to methodological flaws. Specifically, the Kaggle solution applied SMOTE and random undersampling prior to the train-test split, resulting in data leakage and compromising the integrity of the final results.

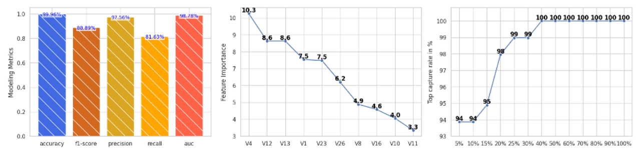

Fig. 7: Performance report on the Credit Card Fraud Detection dataset

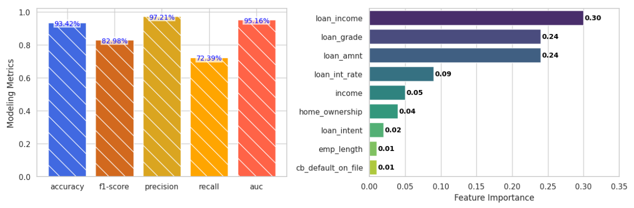

Fig. 8: Performance report for portfolio credit risk

The AutoML solution utilized several models, including XGBoost, Generalized Linear Model (GLM), Gradient Boosting Machine (GBM), and Distributed Random Forest (DRF), along with an additional model selected automatically by the framework. This approach was designed to align closely with the options provided to the meta-tuning agent. Although CatBoost and AdaBoost were not directly available within the H2O framework, XGBoost, GBM, and DRF effectively filled the gap.

The best-performing AutoML model was XGBoost, achieving the following metrics: accuracy of 99.9%, recall of 70.4% and F1 score of 82.1%. In contrast, the CatBoost model, selected as the optimal model by the agentic framework, demonstrated good performance with an accuracy of 99.9%, recall of 81.6%, and f1 score of 88.9%. This indicates that the model selected using agentic programming outperformed the XGBoost model in terms of recall and F1 score, suggesting that it is better suited for scenarios where capturing true positives are crucial, or when presented with a highly imbalanced dataset such as in fraud detection. The top five performing features are "V4, V12, V13, V1 and V23."

2) *Portfolio Credit Risk Dataset:* The result of using agentic programming on the portfolio credit risk dataset [31] underscores the importance of a human expert in agent-based modeling. The dataset contains 32,581 data points and 12 features, with the target feature named "loan_status." The "person_age" feature indicates the age of the borrower, while "person_income" represents their annual income. The "person_home_ownership" feature describes the borrower's home ownership status, which can impact their creditworthiness. Additionally, "person_emp_length" reflects the length of employment in years. Loan characteristics are detailed through the "loan_intent" and "loan_grade" features, which outline the purpose of the loan and its associated grading. The "loan_amnt" specifies the total amount borrowed, and "loan_int_rate" provides the interest rate applicable to the loan. The target variable, "loan_status", indicates whether the loan has defaulted (1) or remained non-default (0). The "loan_percent_income" shows the proportion of income allocated to loan repayments, while "cb_person_def ault_on_file" reveals historical default records. The "cb_person_cred_hist_length" measures the length of the borrower's credit history, providing insights into their borrowing behavior.

The credit risk dataset had an issue of class imbalance, with 78.18% of instances belonging to the majority class ('loan_status' = 1). Missing values were identified in two features: 'person_emp_length' (2.75% missing) and 'loan_int_rate' (9.56% missing). Preliminary performance metrics indicate the robustness of the model trained through agentic collaborative, achieving an accuracy of 93.4%. This result is slightly lower than the accuracy recorded by the upvoted solution on Kaggle [32], primarily because the methodology for feature selection differed. The preprocessing steps applied by the Kaggle solution include creating new features such as income group, loan amount group, loan-to-income ratio, and interest rate-to-loan amount ratio. They also utilized OneHotEncoder for categorical variable encoding. Among their top-performing models was CatBoost, which achieved an accuracy of 93.72%, with a recall of 72.68%, and an f1-score of 83.37%. LightGBM also performed well, reaching an accuracy of 93.54%. In contrast, the model selected at the meta-tuning stage of agentic programming was XGBoost classifier, achieving an accuracy of 93.4%, recall of 72.4%, and an f1-score of 82.9%. The feature engineering agent specifically applied only ordinal encoding, differing from the step employed by the Kaggle solution. The AutoML solution produced comparable metrics to both the agentic system and the Kaggle solution, achieving an accuracy of 92.9%, recall of 72.4%, and an f1-score of 81.9% using Distributed Random Forest (DRF). The top five performing features were "loan_percent_income, loan_grade, loan_amt, loan_int_rate, and person_income."

3) *Human verification of results - Reliability check:* The complexity of the agentic system calls for human evaluation. This was done to make sure that the agents do not produce unreliable outputs and also to determine the usage of the provided dataset. The authors meticulously verified each output and examined the various codes and inferences generated by the agents. This human-centered validation process serves as a crucial safeguard against potential flaws or discrepancies that may have gone undetected by the agents themselves, the judge and human expert. This reduces the likelihood of biases, blind spots, or unforeseen edge cases that only a human observer can identify.

V. Conclusion/Future Directions

This study proposes the human-in-the- loop architecture for developing agentic systems specifically for financial modeling. This is particularly important given that LLMs are prone to hallucination and/or producing results that may not address the exact predefined prompts. We employed four guardrail strategies: implementing a 20% trade-off for LLM temperature (see equation III.1), utilizing the built-in CrewAI feature that separates **agents** from **tasks** (refer to §III-A), integrating a human-in-the-loop framework for oversight (see §IV-A), and introducing a judge that autonomously checks instructions and response. We view this paper as a foundational and pragmatic effort to harness the power of agentic systems for tasks relevant to the financial services industry. Our framework achieved better performance compared to traditional approaches to fraud detection modeling (see §IV-D1), and a comparable result to traditional credit risk modeling (see §IV-D2). Looking ahead, research in this area should focus on self-improvement agents through reinforcement learning.

Acknowledgement

We would like to thank Sharon O'Shea Greenbach (Sr. Counsel & Director, Regulatory Policy at DFS) for her constructive comments relevant to compliance challenges that emerge within the financial services sector and for carefully reviewing the manuscript. Ashkan Golgoon benefited from stimulating discussions with Amirhossein Tajdini and Khashayar Filom. Izunna Okpala benefited from technical discussions with Ryan Franks and Carlos Cardona Amparan. The opinions expressed herein are solely our own and do not reflect the views or opinions of our employer. The authors have not used any customer data or internal data from Discover Financial Services for their experiments.

References

[1] I. Okpala, A. Golgoon, and A. R. Kannan, "Agentic ai systems applied to tasks in financial services: Modeling and model risk management crews," *arXiv preprint arXiv:2502.05439*, 2025.

[2] C. Qian, X. Cong, C. Yang, W. Chen, Y. Su, J. Xu, Z. Liu, and M. Sun, "Communicative agents for software development," *arXiv preprint arXiv:2307.07924*, vol. 6, 2023.

[3] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proceedings of the 36th annual acm symposium on user interface software and technology*, 2023, pp. 1–22.

[4] M. Wu, Y. Yuan, G. Haffari, and L. Wang, "(perhaps) beyond human translation: Harnessing multi-agent collaboration for translating ultra-long literary texts," *arXiv preprint arXiv:2405.11804*, 2024.

[5] C. Li, R. Yang, T. Li, M. Bafarassat, K. Sharifi, D. Bergemann, and Z. Yang, "Stride: A tool-assisted llm agent framework for strategic and interactive decision-making," *arXiv preprint arXiv:2405.16376*, 2024.

[6] Y. Li, Y. Yu, H. Li, Z. Chen, and K. Khashanah, "Tradinggpt: Multi-agent system with layered memory and distinct characters for enhanced financial trading performance," *arXiv preprint arXiv:2309.03736*, 2023.

[7] Y. Yu, H. Li, Z. Chen, Y. Jiang, Y. Li, D. Zhang, R. Liu, J. W. Suchow, and K. Khashanah, "Finmem: A performance-enhanced llm trading agent with layered memory and character design," in *Proceedings of the AAAI Symposium Series*, vol. 3, 2024, pp. 595–597.

[8] Y. Huang, C. Zhou, K. Cui, and X. Lu, "A multi-agent reinforcement learning framework for optimizing financial trading strategies based on timesnet," *Expert Systems with Applications*, vol. 237, p. 121502, 2024.

[9] N. Li, C. Gao, M. Li, Y. Li, and Q. Liao, "Econagent: Large language model-empowered agents for simulating macroeconomic activities," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 15 523–15 536.

[10] N. Vadori, L. Ardon, S. Ganesh, T. Spooner, S. Amrouni, J. Vann, M. Xu, Z. Zheng, T. Balch, and M. Veloso, "Towards multi-agent reinforcement learning-driven over-the-counter market simulations," *Mathematical Finance*, vol. 34, no. 2, pp. 262–347, 2024.

[11] F. Xing, "Designing heterogeneous llm agents for financial sentiment analysis," *ACM Transactions on Management Information Systems*, vol. 16, no. 1, 2025.

[12] H. Jingrong, H. Shan, C. Zhaobin, L. Yu, L. Yingying *et al.*, "Ai-driven digital transformation in banking: A new perspective on operational efficiency and risk management," *Information Systems and Economics*, vol. 5, no. 1, pp. 82–90, 2024.

[13] T. Park, "Enhancing anomaly detection in financial markets with an llm-based multi-agent framework," *arXiv preprint arXiv:2403.19735*, 2024.

[14] K. J. Koa, Y. Ma, R. Ng, and T.-S. Chua, "Learning to generate explainable stock predictions using self-reflective large language models," in *Proceedings of the ACM on Web Conference 2024*, 2024, pp. 4304–4315.

[15] A. Chan, R. Salganik, A. Markelius, C. Pang, N. Rajkumar, D. Krasheninnikov, L. Langosco, Z. He, Y. Duan, M. Carroll *et al.*, "Harms from increasingly agentic algorithmic systems," in *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, 2023, pp. 651–666.

[16] A. Abid, M. Farooqi, and J. Zou, "Persistent anti-muslim bias in large language models," in *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, 2021, pp. 298–306.

[17] L. Weidinger, J. Uesato, M. Rauh, C. Griffin, P.-S. Huang, J. Mellor, A. Glaese, M. Cheng, B. Balle, A. Kasirzadeh *et al.*, "Taxonomy of risks posed by language models," in *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, 2022, pp. 214–229.

[18] S. Han, Q. Zhang, Y. Yao, W. Jin, Z. Xu, and C. He, "Llm multi-agent systems: Challenges and open problems," *arXiv preprint arXiv:2402.03578*, 2024.

[19] M. Shamsujjoha, Q. Lu, D. Zhao, and L. Zhu, "Towards ai-safety-by-design: A taxonomy of runtime guardrails in foundation model based systems," *arXiv preprint arXiv:2408.02205*, 2024.

[20] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

[21] S. G. Ayyamperumal and L. Ge, "Current state of llm risks and ai guardrails," *arXiv preprint arXiv:2406.12934*, 2024.

[22] T. Varshney, "NVIDIA Generaitve AI Technical Blog: Introduction to LLM Agents," https://developer.nvidia.com/blog/introduction-to-llm-agents/, 2023.

[23] Y. Talebirad and A. Nadiri, "Multi-agent collaboration: Harnessing the power of intelligent llm agents," *arXiv preprint arXiv:2306.03314*, 2023.

[24] crewAIInc, "crewAI: Cutting-edge framework for orchestrating role-playing, autonomous AI agents." https://github.com/crewAIInc/crewAI/, 2024.

[25] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *11th International Conference on Learning Representations, ICLR 2023*, 2023.

[26] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[27] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[28] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, M. Podstawski, L. Gianinazzi, J. Gajda, T. Lehmann, H. Niewiadomski, P. Nyczyk *et al.*, "Graph of thoughts: Solving elaborate problems with large language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, 2024, pp. 17 682–17 690.

[29] Kaggle, "Credit Card Fraud Detection Dataset," https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/data, Mar. 2005, [Online; accessed 3. Oct. 2024].

[30] J. M. Bachmann, "Credit Fraud, Dealing with Imbalanced Datasets," Mar. 2005, [Online; accessed 22. Jan. 2025]. [Online]. Available: https://www.kaggle.com/code/janiobachmann/credit-fraud-dealing-with-imbalanced-datasets

[31] Kaggle, "Credit Risk Dataset," https://www.kaggle.com/datasets/laotse/credit-risk-dataset/data, 2020, [Online; accessed 3. Oct. 2024].

[32] A. Tanwar, "Credit Risk Prediction Training and EDA," https://www.kaggle.com/code/anshtanwar/credit-risk-prediction-training-and-eda, 2024, [Online; accessed 22. Jan. 2025].