DrugAgent: Automating AI-aided Drug Discovery Programming through LLM Multi-Agent Collaboration

Anonymous ACL submission

Abstract

Recent progress in Large Language Models (LLMs) has drawn attention to their potential for accelerating drug discovery. However, a central problem remains: translating theoretical ideas into robust implementations in the highly specialized context of pharmaceutical research. This limitation prevents practitioners from making full use of the latest AI developments in drug discovery. To address this challenge, we introduce DrugAgent, a multi-agent framework that automates machine learning (ML) programming for drug discovery tasks. DrugAgent employs an *LLM Planner* that formulates high-level ideas and an *LLM Instructor* that identifies and integrates domain knowledge when implementing those ideas. We present case studies on three representative drug discovery tasks. Our results show that DrugAgent consistently outperforms leading baselines, including a relative improvement of 4.92% in ROC-AUC compared to ReAct for drug-target interaction (DTI). DrugAgent is publicly available at the anonymous link `https://anonymous.4open.science/r/drugagent-5C42/`.

1 Introduction and Related Work

Artificial intelligence (AI) is changing many aspects of drug discovery (Huang et al., 2022). Since experimental measurements of drug properties are costly and time-consuming, researchers have turned to automated approaches for diverse stages of drug development (Pushpakom et al., 2019). AI-ready datasets and benchmarks, such as ADMET prediction, drug-target interaction, and high-throughput screening, are now widely accessible (Huang et al., 2021; Chen et al., 2024a; Wang et al., 2024c). Meanwhile, deep learning has shown promise in lead optimization and drug-target interaction prediction (Huang et al., 2020a), pointing toward possible reductions in the resources required for traditional experimentation.

Yet building machine learning (ML) pipelines for drug discovery is challenging, given that it involves biology, chemistry, pharmaceutical science, and computer science (Huang et al., 2022). While Large Language Models (LLMs) offer automated reasoning and coding assistance, domain-specific subtleties remain difficult to handle in standard frameworks. General-purpose agent-based systems for ML, such as MLAgentBench (Huang et al., 2024a) and AI-Scientist (Lu et al., 2024a), have been proposed for end-to-end ML programming, but they lack expert-level knowledge of drug discovery workflows. Small mistakes, such as using the wrong domain-specific library or misinterpreting biological data types, can be difficult to debug in specialized projects. In contrast, frameworks like ChemCrow (M. Bran et al., 2024) and MultiTool-CoT (Chain of Thought) (Inaba et al., 2023) include chemical tools but offer limited support for larger-scale ML tasks. This highlights the need for *an ML-focused system with domain awareness*, spanning data preprocessing through model evaluation.

Present Work: DrugAgent. We introduce DrugAgent, a multi-agent LLM framework that unifies ML programming with biomedical expertise to address the demands of modern drug discovery. First, DrugAgent systematically checks where domain knowledge is required, then deploys specialized resources before proceeding with coding. Second, it uses a dynamic approach to manage ML ideas, creating diverse options early on and refining them based on empirical results. Third, DrugAgent features a carefully curated library of domain-specific documentation covering data acquisition, data transformation, and advanced model design, supporting critical tasks in drug discovery. We evaluate DrugAgent on three representative tasks and find that it exceeds the performance of general-purpose baselines and matches or surpasses expert-written methods. **Our key contributions include**: **(1)** a systematic workflow that

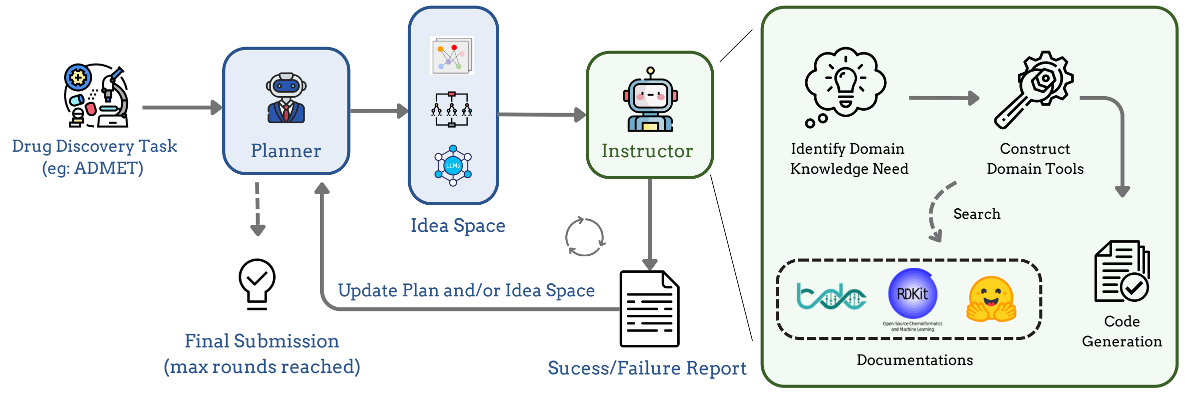


Figure 1: Overview of the DrugAgent framework. Given a drug discovery task described in natural language (i.e., user's input, e.g., design an AI model to predict Absorption (one of the ADMET properties) using the PAMPA dataset (Siramshetty et al., 2021)), the LLM Planner collaborates with the LLM Instructor to iteratively search for actionable, high-performing solutions.

emphasizes when and how to incorporate domain knowledge for ML-driven drug discovery, **(2)** an iterative planning strategy guided by experimental observations, and **(3)** a broad set of specialized tools and documentation for biological data processing and modeling. A detailed comparison with existing approaches is in Appendix A.

2 Methodology

We present DrugAgent, a multi-agent LLM framework designed to handle the specialized challenges of AI-driven drug discovery. As illustrated in Figure 1, DrugAgent integrates two primary agents: (1) an LLM **Planner**, which manages the high-level generation and refinement of solution ideas, and (2) an LLM **Instructor**, which translates these ideas into concrete code, drawing on domain-specific knowledge to address the complex needs of drug discovery tasks.

Problem Formulation. Following Huang et al. (2024a), an ML programming task consists of three components: (1) a *Task Description*, which specifies the objectives and constraints in natural language, (2) *Starter Files*, which provide initial resources like datasets or code templates, and (3) *Evaluator*, which is a performance metric function used to assess the output quality.

LLM Planner: Idea Space Management. Open-ended ML tasks in drug discovery can be approached by multiple strategies with no single deterministic solution, and single-agent systems risk missing promising alternatives (Wang et al., 2024a). Additionally, LLMs sometimes make impractical suggestions if they lack specific domain expertise or rely on hallucinated information. To address these concerns, the Planner operates in two phases: (1) *Idea Generation*, where it derives K candidate solutions from the task description, and (2) *Exploration*, where it selects one idea and sends it to the Instructor for experimental evaluation. Based on success or failure reports, it revises the idea set, discarding those that underperform or are not feasible. The process repeats until a maximum iteration limit is reached, after which the highest-performing idea is submitted as the final solution.

LLM Instructor: Domain-specific Knowledge and Tool Preparation. Drug discovery depends on specialized workflows, e.g., the correct handling of SMILES strings and tailored data preprocessing. When standard code-generation approaches ignore this domain requirements (Huang et al., 2023, 2024b), the resulting errors are hard to debug.

Within DrugAgent, the Instructor incorporates domain knowledge at every step of the coding process. It can execute standard ML actions (e.g., reading or editing scripts, running code; see Appendix B) and references a set of targeted documents to build or refine specialized tools. The Instructor then generates a performance report—if critical functionalities are absent, it returns a failure report instead. Specifically, the Instructor relies on three curated types of documentation:

- **Raw Data Acquisition:** Methods for retrieving and preprocessing biological data.

- **Featurizing Biological Data:** Techniques for encoding molecules and proteins (e.g., fingerprints, graph-based representations).
- **Domain-Specific Models:** Pretrained foundation models such as ChemBERTa (che) (small molecules) and ESM (Evolutionary Scale Modeling for protein sequence) (Lin et al., 2022).

Further details about these resources appear in Appendix C. By explicitly integrating domain guidance into the coding workflow, DrugAgent aims to reduce errors that arise from incomplete or incorrect handling of drug discovery subtleties.

3 Experiment

3.1 Experimental Setup

AI-solvable Drug Discovery Tasks. We propose three representative AI-solvable drug discovery tasks as a proof-of-concept to validate the effectiveness of DrugAgent. **ADMET** prediction forecasts pharmacokinetic properties (Absorption, Distribution, Metabolism, Excretion, and Toxicity) from a drug's molecular structure, crucial for assessing a drug's efficacy and safety (Niu et al., 2024; Lu et al., 2024b; Chen et al., 2021, 2024b). **High-throughput screening (HTS)** leverages ML models to predict assay outcomes based on molecular structure, improving the efficiency and reducing the cost of evaluating the biological activity of large chemical libraries (Pham et al., 2021). **Drug-target interaction (DTI)** prediction forecasts the binding affinity between drugs and proteins using compound structures and amino acid sequences, supporting virtual screening, drug repurposing, and side effect prediction (Liu et al., 2024). All these problems are binary classification tasks.

Dataset. We select one dataset for each task: **PAMPA** (Siramshetty et al., 2021) for ADMET prediction, **DAVIS** (Davis et al., 2011) for DTI prediction, and **HIV** (Wu et al., 2018) for HTS. Appendix D provides details on the dataset description, the rationale behind dataset selection, and the data splitting methods.

Baselines. We compare DrugAgent against four AI-based methods and one human baseline. We use `GPT-4o-2024-08-06` as the underlying language model for all AI methods, as it is one of the state-of-the-art models for ML coding according to previous benchmarks (Chan et al., 2024).

CoT (Chain of Thought) is a simple baseline where the agent generates a solution by breaking the problem into substeps (Wei et al., 2022). **ReAct** follows an interleaved reasoning and action approach, enabling interactive analysis and execution (Yao et al., 2023). **ResearchAgent** is designed for ML tasks, maintaining a research plan and executing key actions such as file understanding, script editing, and task reflection (Huang et al., 2024a). **ChemCrow** is a chemistry-focused LLM agent that augments LLM with 18 expert-designed tools to enable automated planning and execution across tasks such as synthesis, drug discovery, and materials design (M. Bran et al., 2024). The **Human** baseline relies on model choices reported as effective in the literature and selected by experts, with details provided in Appendix E.

These baselines are compared with two variants of DrugAgent: **DrugAgent@Idea1**, where the agent selects the best ideas based on validation results, and **DrugAgent@Idea3**, where the agent submits the top three ideas based on validation results, and reports the best test set outcome. Other methods do not include an idea search mechanism like DrugAgent, so only a single result is reported for each. Detailed experimental settings and implementation details for DrugAgent, including hyperparameters and prompt examples, are provided in Appendix F.

Evaluation Metrics. We conduct eight independent runs for each AI-based method. A submission is considered valid if (1) the generated code is free of bugs and, when executed, produces a submission file, (2) the submission file adheres to our format requirements, and (3) the performance does not fall more than 10% below the human baseline. The average metric (ROC-AUC) across all valid submissions is reported. If all eight submissions are invalid, the results are marked as N/A.

3.2 Quantitative Results

Table 1 reports the performance across all datasets. DrugAgent achieves the highest ROC-AUC and Valid Rate among all AI-based methods, performing comparably to baselines selected by human experts. Notably, it outperforms ReAct in the DTI task, achieving a relative improvement of 4.92% in ROC-AUC. We also observe that DrugAgent@Idea3 surpasses DrugAgent@Idea1 in the ADMET and HTS tasks. This suggests that validation set performance does not always strongly correlate with test set performance, sometimes leading the agent to select a suboptimal idea for final

	ADMET		HTS		DTI	
Method	ROC-AUC (↑)	Valid Rate (↑)	ROC-AUC (↑)	Valid Rate (↑)	ROC-AUC (↑)	Valid Rate (↑)
Human	0.8173	—	**0.8305**	—	0.8940	—
CoT	0.7599	62.5%	0.7524	50.0%	N/A	0.0%
React	0.7385	87.5%	0.7653	75.0%	0.8530	50.0%
ChemCrow	0.7860	25.0%	0.7663	25.0%	0.8862	75.0%
ResearchAgent	0.7957	100.0%	0.7913	100.0%	0.8793	75.0%
DrugAgent@Top1	0.7667	100.0%	0.7919	100.0%	0.8950	87.5%
DrugAgent@Top3	**0.8206**	100.0%	0.8257	100.0%	**0.8950**	87.5%

Table 1: ROC-AUC and Valid Rate for **PAMPA** (ADMET), **HIV** (HTS), and **DAVIS** (DTI) datasets.

Method	ROC-AUC (↑)	Valid Rate (↑)
DrugAgent	0.8950	87.5%
DrugAgent w/o Planner	0.8845	87.5%
DrugAgent w/o Instructor	0.8770	75.0%

Table 2: Ablation study on the DAVIS (DTI) task, demonstrating how removing the Planner or Instructor from DrugAgent affects ROC-AUC and Valid Rate. Results are averaged across runs.

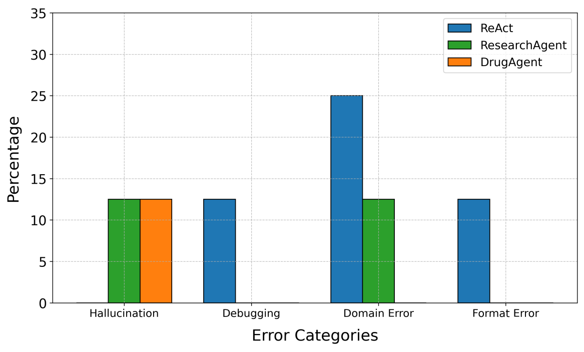


Figure 2: Percentage of runs over **DAVIS** (DTI) dataset that falls into different error modes.

submission. However, considering multiple submissions can help mitigate this problem. Furthermore, we find that domain-specialized agents such as ChemCrow do not outperform general-purpose agents. This is likely because ChemCrow's toolset is designed for chemical reasoning, which offers limited benefit for ML coding tasks.

Table 2 highlights the importance of each agent in our framework, demonstrating that both the Planner and Instructor contribute significantly to overall performance. Additional ablation studies, including a qualitative analysis of each agent's role, the effect of using alternative LLMs, and the impact of execution rounds, are provided in Appendix G.

3.3 Case Study

Comparing DrugAgent with ReAct. We conduct a case study to compare our framework with ReAct (see Appendix H for detailed traces and analysis). The results highlight our framework's effectiveness in diversifying ideas, accurately integrating domain knowledge, and learning from failures.

Trace Analysis. To further assess the agent's reasoning and decision-making process, we analyze the traces of all runs for the DTI task and categorize the top four error types. A detailed description of each failure type is provided in Appendix I. Figure 2 illustrates that for general agent frameworks like ReAct and ResearchAgent, most errors occur due to poor performance caused by incorrect operations in steps requiring domain knowledge. In contrast, DrugAgent exhibits no errors in this category and achieves the lowest overall error rate, highlighting the effectiveness of our framework in utilizing domain knowledge.

4 Conclusion

In this paper, we have introduced DrugAgent, a multi-agent framework that marks a significant advancement in leveraging large language models for automating critical aspects of drug discovery. Through case studies in three drug discovery tasks, DrugAgent demonstrates remarkable improvements over general-purpose agent frameworks, such as ReAct and ResearchAgent. This can largely be attributed to the planner agent, which effectively generates and searches for ideas, and the instructor agent, which ensures reliable implementation by integrating a specialized toolset. Together, these agents enable DrugAgent to bridge the gap between generalized AI capabilities and the nuanced demands of pharmaceutical research. We believe this work opens exciting new avenues for research and collaboration, pushing the boundaries of AI-driven drug discovery.

Limitations

This study has several limitations. First, we evaluate the performance of DrugAgent on three case study tasks. However, these tasks are not sufficient for a comprehensive evaluation, and there is a need for more extensive benchmarks to assess machine learning programming tasks in drug discovery settings. Second, although DrugAgent can generate solutions comparable to human baselines, it is still limited to classic state-of-the-art baselines rather than the latest cutting-edge methods. Advancing agent capabilities in this domain will require significant research efforts. Third, the current documentation for DrugAgent is relatively basic and could be expanded in the future to cover additional aspects of the drug discovery process. Lastly, the agent framework has the potential to incorporate a 'human-in-the-loop' approach, which would enhance its usability for scientists working on real-world drug discovery tasks.

Ethics Statement

We do not foresee any immediate ethical or societal concerns arising from our work. However, we acknowledge that, due to challenges like hallucination, the current version of DrugAgent is not yet ready for direct deployment in the drug discovery pipeline. For instance, errors such as fabricating results could lead to inaccurate predictions, which might waste resources in the wet lab verification process or misguide the drug discovery direction. As a result, further safety checks and human oversight are essential. Moreover, as AI agents advance, there is potential for them to replace human engineers in ML programming tasks within drug discovery. This highlights the need for human workers to learn how to effectively collaborate with the agent and understand its underlying implementation. By fostering this collaboration, AI can enhance and complement professional expertise rather than replace it.

References

CBDD Group. 2020. Pybiomed: A toolkit for calculating molecular descriptors and analyzing biological molecules. Accessed: February 13, 2025.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. 2024. Mle-bench: Evaluating machine learning agents on machine learning engineering.

Jintai Chen, Yaojun Hu, Yue Wang, Xu Cao, Miao Lin, Hongxia Xu, Jian Wu, Cao Xiao, Jimeng Sun, et al. 2024a. Trialbench: Multi-modal artificial intelligence-ready clinical trial datasets. *arXiv:2407.00631*.

Lulu Chen, Yingzhou Lu, Chiung-Ting Wu, Robert Clarke, Guoqiang Yu, Jennifer E Van Eyk, David M Herrington, and Yue Wang. 2021. Data-driven detection of subtype-specific differentially expressed genes. *Scientific reports*, 11(1):332.

Tianyi Chen, Nan Hao, and Capucine Van Rechem. 2024b. Uncertainty quantification on clinical trial outcome prediction. *Preprint*, arXiv:2401.03482.

Mindy I Davis, Jeremy P Hunt, Sanna Herrgard, Pietro Ciceri, Lisa M Wodicka, Gabriel Pallares, Michael Hocker, Daniel K Treiber, and Patrick P Zarrinkar. 2011. Comprehensive analysis of kinase inhibitor selectivity. *Nature biotechnology*, 29(11):1046–1051.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *Preprint*, arXiv:2402.01680.

Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W. Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. 2021. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. *Advances in Neural Information Processing Systems*.

Kexin Huang, Tianfan Fu, Wenhao Gao, Yue Zhao, Yusuf Roohani, Jure Leskovec, Connor W Coley, Cao Xiao, Jimeng Sun, and Marinka Zitnik. 2022. Artificial intelligence foundation for therapeutic science. *Nature Chemical Biology*, 18:1033.

Kexin Huang, Tianfan Fu, Lucas M Glass, Marinka Zitnik, Cao Xiao, and Jimeng Sun. 2020a. DeepPurpose: a deep learning library for drug–target interaction prediction. *Bioinformatics*, 36(22-23):5545–5547.

Kexin Huang, Cao Xiao, Lucas M Glass, and Jimeng Sun. 2020b. Moltrans: Molecular interaction transformer for drug–target interaction prediction. *Bioinformatics*, 37(6):830–836.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*. Work in progress; 49 pages.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. 2024a. Mlagentbench: Evaluating language agents on machine learning experimentation. In *Thirty-eighth Conference on Neural Information Processing Systems*.

Yue Huang, Lichao Sun, Haoran Wang, Siyuan Wu, Qihui Zhang, Yuan Li, Chujie Gao, Yixin Huang, Wenhan Lyu, Yixuan Zhang, et al. 2024b. Position: Trustllm: Trustworthiness in large language models. In *International Conference on Machine Learning*, pages 20166–20270. PMLR.

Tatsuro Inaba, Hirokazu Kiyomaru, Fei Cheng, and Sadao Kurohashi. 2023. MultiTool-CoT: GPT-3 can use multiple external tools with chain of thought prompting. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1522–1532, Toronto, Canada. Association for Computational Linguistics.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2024. A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*.

Greg Landrum. 2023. Rdkit: Open-source cheminformatics.

Binxu Li, Tiankai Yan, Yuanting Pan, Jie Luo, Ruiyang Ji, Jiayuan Ding, Zhe Xu, Shilong Liu, Haoyu Dong, Zihao Lin, and Yixin Wang. 2024a. Mmedagent: Learning to use medical tools with multi-modal agent. *arXiv preprint arXiv:2407.02483*. Accepted at EMNLP 2024.

Mufei Li, Jinjing Zhou, Jiajing Hu, Wenxuan Fan, Yangkang Zhang, Yaxin Gu, and George Karypis. 2021. Dgl-lifesci: An open-source toolkit for deep learning on graphs in life science. *ACS Omega*, 6(41):27233–27238.

Ziming Li, Qianbo Zang, David Ma, Jiawei Guo, Tuney Zheng, Minghao Liu, Xinyao Niu, Yue Wang, Jian Yang, Jiaheng Liu, Wanjun Zhong, Wangchunshu Zhou, Wenhao Huang, and Ge Zhang. 2024b. Autokaggle: A multi-agent framework for autonomous data science competitions. *arXiv preprint arXiv:2410.20424*.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, et al. 2022. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *bioRxiv*.

Sizhe Liu, Yuchen Liu, Haofeng Xu, Jun Xia, and Stan Z. Li. 2025. Sp-dti: Subpocket-informed transformer for drug-target interaction prediction. *Bioinformatics*, btaf011.

Sizhe Liu, Jun Xia, Lecheng Zhang, Yuchen Liu, Yue Liu, Wenjie Du, Zhangyang Gao, Bozhen Hu, Cheng Tan, Hongxin Xiang, and Stan Z. Li. 2024. Flexmol: A flexible toolkit for benchmarking molecular relational learning. In *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS)*.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024a. The AI Scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*.

Yingzhou Lu, Tianyi Chen, Nan Hao, Capucine Van Rechem, Jintai Chen, and Tianfan Fu. 2024b. Uncertainty quantification and interpretability for clinical trial approval prediction. *Health Data Science*, 4:0126.

Jakub Lála, Odhran O'Donoghue, Aleksandar Shtedritski, Sam Cox, Samuel G. Rodriques, and Andrew D. White. 2023. Paperqa: Retrieval-augmented generative agent for scientific research. *arXiv preprint arXiv:2312.07559*.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. 2024. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6(5):525–535.

Zhangming Niu, Xianglu Xiao, Wenfan Wu, Qiwei Cai, Yinghui Jiang, Wangzhen Jin, Minhao Wang, Guojian Yang, Lingkang Kong, Xurui Jin, Guang Yang, and Hongming Chen. 2024. Pharmabench: Enhancing admet benchmarks with large language models. *Scientific Data*, 11(985).

Thai-Hoang Pham, Yue Qiu, Jucheng Zeng, Lei Xie, and Ping Zhang. 2021. A deep learning framework for high-throughput mechanism-driven phenotype compound screening and its application to covid-19 drug repurposing. *Nature Machine Intelligence*, 3(3):247–257.

Sudeep Pushpakom, Francesco Iorio, Patrick A Eyers, K Jane Escott, Shirley Hopper, Andrew Wells, Andrew Doig, Tim Guilliams, Joanna Latimer, Christine McNamee, et al. 2019. Drug repurposing: progress, challenges and recommendations. *Nature Reviews Drug Discovery*, 18(1):41–58.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. 2023. Tool learning with foundation models. *Preprint*, arXiv:2304.08354.

Chidaksh Ravuru, Sagar Srinivas Sakhinana, and Venkataramana Runkana. 2024. Agentic retrieval-augmented generation for time series analysis. In *Proceedings of the Undergraduate Consortium at ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.

Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. 2019. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *PNAS*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Preprint*, arXiv:2302.04761.

V. B. Siramshetty, P. Shah, et al. 2021. Validating adme qsar models using marketed drugs. *SLAS Discovery*, 26(10):1326–1336.

Evan Wang, Federico Cassano, Catherine Wu, Yunfeng Bai, Will Song, Vaskar Nath, Ziwen Han, Sean Hendryx, Summer Yue, and Hugh Zhang. 2024a. Planning in natural language improves llm search for code generation. *arXiv preprint arXiv:2409.03733*.

Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, et al. 2023. Scientific discovery in the age of artificial intelligence. *Nature*, 620:47–60.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and et al. 2024b. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6).

Yue Wang, Tianfan Fu, Yinlong Xu, Zihan Ma, Hongxia Xu, Bang Du, Honghao Gao, Jian Wu, and Jintai Chen. 2024c. Twin-gpt: Digital twins for clinical trials via large language model. *ACM Transactions on Multimedia Computing, Communications and Applications*.

WecoAI. 2024. Aide: The machine learning engineer agent.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Ajay S Pappu, Karl Leswing, and Vijay Pande. 2018. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530.

Jun Xia, Lecheng Zhang, Xiao Zhu, Yue Liu, Zhangyang Gao, Bozhen Hu, Cheng Tan, Jiangbin Zheng, Siyuan Li, and Stan Z. Li. 2023. Understanding the limitations of deep models for molecular property prediction: Insights and solutions. In *NeurIPS 2023*. Last Modified: 03 Nov 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Sion Yoon, Tae Eun Kim, and Yoo Jung Oh. 2024. Designing and evaluating multi-chatbot interface for human-ai communication: Preliminary findings from a persuasion task. *arXiv preprint arXiv:2406.19648*.

Ling Yue, Sixue Xing, Jintai Chen, and Tianfan Fu. 2024. Clinicalagent: Clinical trial multi-agent with large language model-based reasoning. *arXiv preprint arXiv:2404.14777*.

Yizhen Zheng, Huan Yee Koh, Maddie Yang, Li Li, Lauren T. May, Geoffrey I. Webb, Shirui Pan, and George Church. 2024. Large language models in drug discovery and development: From disease mechanisms to clinical trials. *arXiv preprint arXiv:2409.04481*.

Hakime Öztürk, Arzucan Özgür, and Elif Ozkirimli. 2018. Deepdta: Deep drug–target binding affinity prediction. *Bioinformatics*, 34(17):i821–i829.

A Related Work

This section provides a more detailed overview of related work on LLM agents and their applications in ML programming and biomedical discovery.

LLM Agents An LLM agent is a system that uses large language models to interact with users or other systems, perform tasks, and make decisions autonomously. Empowered by LLMs, LLM agents have the capability to perform multi-step reasoning, planning, and action execution beyond static text generation (Wang et al., 2024b). Previous works have equipped LLM agents with modules to dynamically interact with external tools, retrieve information, and adapt based on real-time feedback (Schick et al., 2023; Yoon et al., 2024; Qin et al., 2023; Ravuru et al., 2024; Lála et al., 2023). This allows them to solve complex, evolving tasks such as code writing, long-term reasoning, and decision-making in various contexts (Guo et al., 2024; Jiang et al., 2024). In this work, we tailor the LLM multi-agent framework to drug discovery tasks.

Table 3: **Key differences between DrugAgent and existing agent methods.** DrugAgent stands out by: 1) interacting with the environment, 2) specializing in ML programming, 3) incorporating domain knowledge specific to drug discovery, and 4) planning at the idea space level.

	Interaction with Env	ML Specialization	Domain Knowledge	Idea Space Planning
ReAct (Yao et al., 2023)	✔	✗	✗	✗
ResearchAgent (Huang et al., 2024a)	✔	✔	✗	✗
ChemCrow (M. Bran et al., 2024)	✔	✗	✔	✗
DrugAgent (Ours)	✔	✔	✔	✔

LLM for ML Programming Recent work has focused on accelerating traditionally manual research processes by automating ML programming. AIDE acts as a data science agent, exploring a vast solution space and iteratively refining its approach to reach optimal solutions (WecoAI, 2024). AutoKaggle introduces a specialized multi-agent framework for Kaggle data science competitions (Li et al., 2024b). AI-Scientist enables LLMs to conduct research autonomously, from idea generation to paper drafting, focusing on ML-related topics (Lu et al., 2024a). In parallel, benchmarks have been developed that provide a suite of 13 tasks to evaluate LLMs' capabilities in conducting ML programming (Huang et al., 2024a). However, existing works cannot handle domain-specific ML tasks requiring complex domain knowledge, e.g., AI-aided drug discovery. To address this, we design workflows to insert domain knowledge and call domain-specific tools automatically.

LLM for Biomedical Discovery Many studies have highlighted the applications of LLMs in biomedical discovery, particularly when integrated with domain-specific tools. For instance, ChemCrow demonstrates the potential of LLM agents in organic synthesis, drug discovery, and material design (M. Bran et al., 2024). Similarly, MMedAgent is a multimodal medical agent designed to handle complex language and multimodal tasks, demonstrating LLM versatility in medical applications (Li et al., 2024a). The multi-agent approach is exemplified by ClinicalAgent (Yue et al., 2024), which introduces a framework for clinical trial outcome prediction by decomposing it into subproblems, allowing individual agents to collaborate and generate a comprehensive outcome. Existing ML biomedical agents, however, generally lack the ML-specific expertise required to perform end-to-end programming.

B Action

Below is a set of machine learning (ML)-related actions available to the instructor: List Files, Read File, Write File, Append File, Copy File, Inspect Script Lines, Undo Edit Script, Execute Script, Final Answer, Understand File, Edit Script, and Edit Script Segment. Since these actions are commonly used across general ML agents, we recommend referring to MLAgentBench (Huang et al., 2024a) for a detailed explanation of each action.

C Documentation

Raw Data Preprocessing: We compiled documentation from the TDC library (Huang et al., 2021), which includes 66 AI/ML-ready datasets for drug discovery.

Drug Preprocessing: We documented seven molecular fingerprinting methods, two molecular graph construction methods, and one one-hot encoding method, using a combination of the TDC (Huang et al., 2021), DGL-LifeSci (Li et al., 2021), and RDKit (Landrum, 2023) libraries.

Protein Preprocessing: We documented three protein fingerprinting methods and one one-hot encoding method, utilizing the PyBioMed (CBDD Group, 2020) library.

Domain-Specific Models: We documented the ChemBERTa (che) and ESM (Rives et al., 2019) models, using the Transformers library (Wolf et al., 2020).

The complete documentation, along with the code for our framework, is available at `https://anonymous.4open.science/r/drugagent-5C42/`. It is important to note that this documentation can be easily extended based on specific needs and available resources.

D Dataset Description

Table 4 provides an overview of the selected drug discovery tasks and datasets used in our case study.

DAVIS: This dataset contains 68 drugs and

	ADMET Prediction	**HTS Prediction**	**DTI Prediction**
Type	Single-instance prediction	Single-instance prediction	Multi-instance prediction
Input	SMILES string	SMILES string	SMILES string and protein amino acid sequence
Impact	Prevents clinical trial failures through early and accurate ADMET profiling	Reduces experimental screening costs by predicting assay outcomes	Reduces experimental screening needs by prioritizing drug candidates with high binding affinity
Dataset (Case Study)	PAMPA (Siramshetty et al., 2021)	HIV (Wu et al., 2018)	DAVIS (Davis et al., 2011)

Table 4: Task overview: ADMET, HTS, and DTI. In this paper, we focus on small-molecule drugs, which constitute over 90% of all approved drugs. Small molecules are represented as SMILES strings, a compact ASCII notation describing chemical structures.

379 proteins, with 2086, 3006, and 6011 samples allocated for training, validation, and testing, respectively. A detailed description of the dataset and preprocessing methods can be found in MolTrans (Huang et al., 2020b). The dataset is available at https://github.com/kexinhuang12345/moltrans.

PAMPA: This dataset includes 1424 training samples, 203 validation samples, and 407 test samples. The data is split using the TDC random split strategy. More details can be found on the TDC website: https://tdcommons.ai/single_pred_tasks/adme.

HIV: This dataset consists of 28,789 training samples, 4,113 validation samples, and 8,225 test samples. The split follows the TDC random split strategy. Further information is available on the TDC website: https://tdcommons.ai/single_pred_tasks/hts.

D.1 Rationale for Task and Dataset Selection

These tasks are identified in recent surveys and reviews as representative machine learning problems in drug discovery (Wang et al., 2023; Zheng et al., 2024). Together, they span key decision points across the Hit Identification, Hit-to-Lead, and Lead Optimization stages (Zheng et al., 2024). All three are essential for selecting compounds with desirable properties—whether related to biological activity, screening outcomes, or pharmacokinetic behavior—thereby ensuring that only the most promising candidates progress through the drug discovery pipeline.

These tasks have also been chosen by domain experts as representative benchmarks, including their official designation by the Therapeutics Data Commons (TDC) team (Huang et al., 2021).

The datasets used for each task are the official benchmark datasets provided by the TDC team. These datasets were curated to reflect realistic experimental settings and are widely adopted in the field as standardized benchmarks for evaluating predictive models.

E Human Baseline

Previous research (Xia et al., 2023) has shown that for ADMET and HTS tasks, tree-based models consistently outperform other approaches such as GCN, DNN, SVM, CNN, RNN, and MPNN. These models serve as a simple yet strong baseline that is difficult to beat. Therefore, we use a random forest model combined with Morgan fingerprinting as the human baseline for these two tasks.

For the DTI task, DeepDTA (Öztürk et al., 2018), which employs two CNN encoders for drug and protein representations, is a well-established deep learning baseline. It is widely adopted as a SOTA baseline in DTI studies (Huang et al., 2020b; Liu et al., 2024, 2025) and is considered the human baseline for this task.

F Settings

For all agent frameworks, we allow a maximum of 100 actions. For a detailed definition of an action, refer to Appendix B and the MLAgent-Bench (Huang et al., 2024a) paper. For the ResearchAgent baseline, we made the following adjustments to improve performance:

- For the `understand_file` action, we process only the first 3 blocks to save resources in case the file is too large (e.g., when understanding a CSV file).
- We also print error messages in the observation to assist the agent with debugging.

For DrugAgent, we set the maximum number of ideas explored to 5, based on the ablation study results in Appendix G. Prompt examples and implementation details for reproducibility are provided in Appendix J.

G Ablation Study

G.1 Without Instructor

We found that although exploring multiple ideas improves the overall performance compared to the original ReAct framework, the results are still not satisfactory. The primary reason is that the model sometimes encodes molecules in an ineffective manner. Below is an example of code generated by the ReAct Agent that naively encodes a protein, leading to poor results despite a promising idea.

```python
def protein_to_features(protein_sequence):
    # Convert amino acid sequence into a feature vector of fixed length 1024
    features = np.zeros(1024, dtype=int)  # fixed length vector
    for i, c in enumerate(protein_sequence[:1024]):
        features[i] = ord(c)
    return features
```

G.2 Without Planner

We found that even when prompted to iteratively experiment with different models, the agent fails to sufficiently diversify its approach, often focusing on variations of similar ideas. For example, it may compare logistic regression with logistic regression incorporating feature engineering, which limits its ability to explore more optimal approaches.

Model	RESEARCHAGENT	DRUGAGENT
GPT-4o	0.8793	0.8950
GPT-4o-mini	0.8772	0.8785
LLaMA-70B	0.8102	0.8367
GPT-3.5	N/A	0.8084

Table 5: ROC-AUC of ResearchAgent and DrugAgent under different LLMs on the DAVIS (DTI) task. N/A indicates an invalid submission.

Round Number	ROC-AUC (↑)
1	0.8433
3	0.8824
5	0.8950
10	0.8962

Table 6: ROC-AUC of DrugAgent with different round numbers on the DAVIS (DTI) task.

G.3 Alternative LLMs

We compare DrugAgent and ResearchAgent using four different LLMs. As shown in Table 5, DrugAgent consistently outperforms ResearchAgent across all settings, demonstrating the robustness of our method to the choice of underlying LLM.

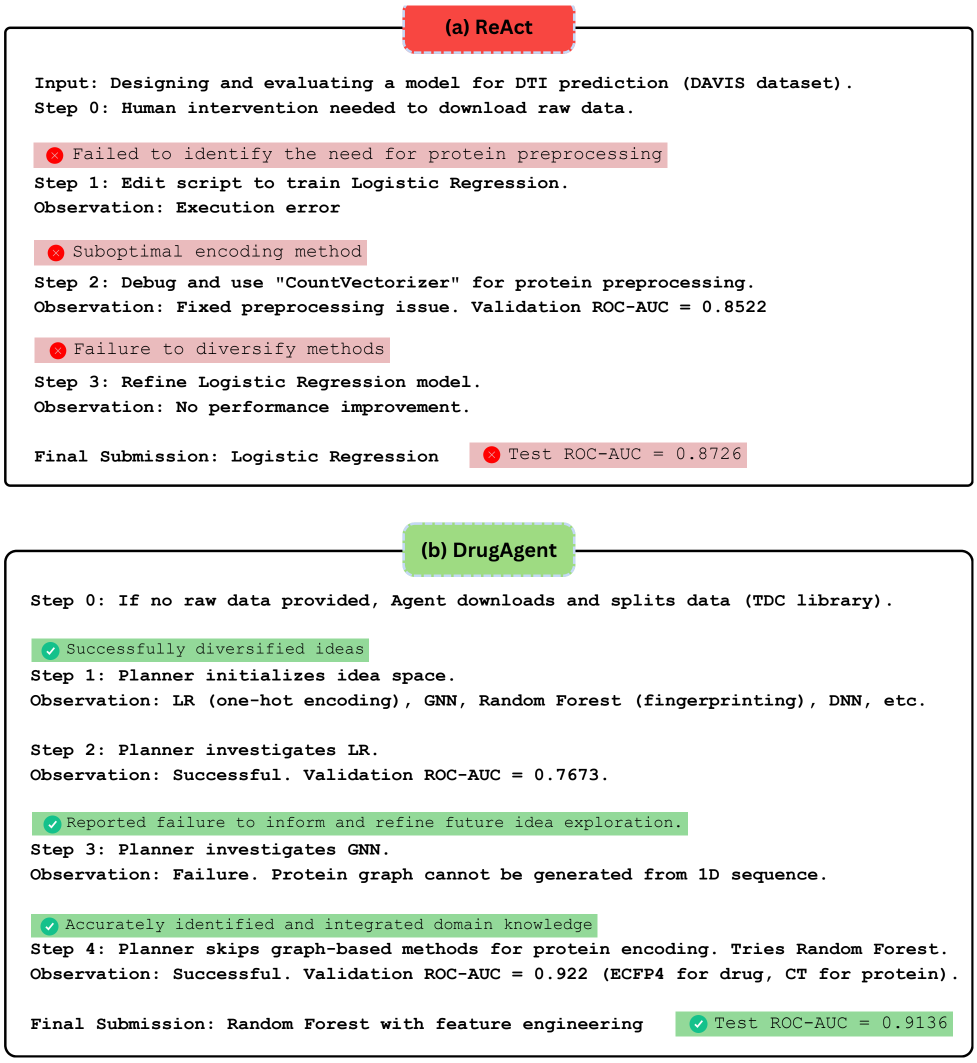
G.4 Number of Planning Rounds

The maximum number of planning rounds is a user-defined hyperparameter, with each round generating a distinct idea. We set the value to 5 in our main experiments, as it provides a good balance between computational cost and performance. We performed an ablation study on the DTI task to evaluate the effect of different round numbers. The results are shown in Table 6. Performance improves steadily up to 5 rounds, with only a small gain when increasing from 5 to 10. Notably, The performance of DrugAgent with round 1 is lower than that of DrugAgent without the planner. This is because the planner introduces more diversity in the idea space, which can make the quality of the first idea more variable.

H Comparing DrugAgent with ReAct

To demonstrate the effectiveness of DrugAgent, we conducted a case study on a DTI prediction task and compared its performance to ReAct, as illustrated in Fig. 3. This comparison underscores the challenges LLMs face in domain-specific tasks and highlights how DrugAgent overcomes these limitations.

First, while ReAct (Yao et al., 2023) is prompted



(a) ReAct

```
Input: Designing and evaluating a model for DTI prediction (DAVIS dataset).
Step 0: Human intervention needed to download raw data.

✗ Failed to identify the need for protein preprocessing
Step 1: Edit script to train Logistic Regression.
Observation: Execution error

✗ Suboptimal encoding method
Step 2: Debug and use "CountVectorizer" for protein preprocessing.
Observation: Fixed preprocessing issue. Validation ROC-AUC = 0.8522

✗ Failure to diversify methods
Step 3: Refine Logistic Regression model.
Observation: No performance improvement.

Final Submission: Logistic Regression    ✗ Test ROC-AUC = 0.8726
```

(b) DrugAgent

```
Step 0: If no raw data provided, Agent downloads and splits data (TDC library).

✓ Successfully diversified ideas
Step 1: Planner initializes idea space.
Observation: LR (one-hot encoding), GNN, Random Forest (fingerprinting), DNN, etc.

Step 2: Planner investigates LR.
Observation: Successful. Validation ROC-AUC = 0.7673.

✓ Reported failure to inform and refine future idea exploration.
Step 3: Planner investigates GNN.
Observation: Failure. Protein graph cannot be generated from 1D sequence.

✓ Accurately identified and integrated domain knowledge
Step 4: Planner skips graph-based methods for protein encoding. Tries Random Forest.
Observation: Successful. Validation ROC-AUC = 0.922 (ECFP4 for drug, CT for protein).

Final Submission: Random Forest with feature engineering    ✓ Test ROC-AUC = 0.9136
```

Figure 3: Comparison of ReAct and DrugAgent on a DTI task. (a) ReAct, a general-purpose framework, delivers lower performance due to a lack of idea diversification and failure to recognize and incorporate domain knowledge. (b) DrugAgent systematically explores a variety of approaches, successfully identifying optimal models and preprocessing methods to achieve strong performance.

to iteratively select the best model, it lacks a high-level planning mechanism, instead focusing on implementing and refining a single approach. In contrast, DrugAgent leverages a planner agent to diversify ideas and systematically identify the most effective approaches.

Second, ReAct fails to recognize the need for preprocessing protein data early in the process, resulting in wasted time during debugging and the subsequent selection of suboptimal methods for molecular data encoding. By comparison, DrugAgent correctly identifies substeps that require domain-specific tools and successfully integrates fingerprint-based encoding methods for biological data, delivering a bug-free model on the first attempt.

Third, ReAct is more prone to failure and struggles to recover from bad planning. DrugAgent, on the other hand, learns from observations to guide future idea selection and avoids repeated failures. For instance, in the case study, DrugAgent identifies the limitations of graph-based methods for protein encoding and avoids further exploration of those approaches. These findings highlight DrugAgent's ability to automate domain-specific machine learning tasks while systematically selecting and refining the most effective approaches for the problem at hand.

H.1 Sample Code by DrugAgent

```python
import pandas as pd
import numpy as np
```

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score
from agent_tools import DrugFeaturizer
from agent_tools import ProteinFeaturizer

# Load train, validation, and test datasets
train_data = pd.read_csv("train.csv")
val_data = pd.read_csv("val.csv")
test_data = pd.read_csv("test.csv")

print(f"Loaded {len(train_data)} training samples, {len(val_data)} validation samples, and {len(test_data)} test samples.")

print(train_data.head())

X_train_drug = train_data['Drug']
X_train_protein = train_data['Protein']
y_train = train_data['Y']

X_val_drug = val_data['Drug']
X_val_protein = val_data['Protein']
y_val = val_data['Y']

# ***********************************************
# In this part of the code, write and train the model on the above dataset to perform the task.
# This part should populate the variable train_roc_auc and valid_roc_auc on the model selected
# ***********************************************

# Feature extraction
drug_featurizer = DrugFeaturizer(method='ECFP4')
protein_featurizer = ProteinFeaturizer(method='ct')

# Transformations
X_train_drug_features = drug_featurizer(X_train_drug.tolist())
X_train_protein_features = protein_featurizer(X_train_protein.tolist())

X_val_drug_features = drug_featurizer(X_val_drug.tolist())
X_val_protein_features = protein_featurizer(X_val_protein.tolist())

# Concatenate drug and protein features
X_train_features = np.concatenate([X_train_drug_features, X_train_protein_features], axis=1)
X_val_features = np.concatenate([X_val_drug_features, X_val_protein_features], axis=1)

# Train Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_features, y_train)

# Predict probabilities
train_preds = rf_model.predict_proba(X_train_features)[:, 1]
val_preds = rf_model.predict_proba(X_val_features)[:, 1]

# Compute ROC AUC scores
train_roc_auc = roc_auc_score(y_train, train_preds)
valid_roc_auc = roc_auc_score(y_val, val_preds)

# ***********************************************
# End of the main training module
# ***********************************************

print("Train ROC AUC Score: " + str(train_roc_auc))
print("Validation ROC AUC Score: " + str(valid_roc_auc))

X_test_drug = test_data['Drug']
X_test_protein = test_data['Protein']

# Transformations for test set
X_test_drug_features = drug_featurizer(X_test_drug.tolist())
X_test_protein_features = protein_featurizer(X_test_protein.tolist())
X_test_features = np.concatenate([X_test_drug_features, X_test_protein_features], axis=1)

# Replace with actual predictions
test_preds = rf_model.predict_proba(X_test_features)[:, 1]

test_data['Predicted'] = test_preds

output_file = "submission.csv" #do not change submission file name
test_data.to_csv(output_file, index=False)

print(f"Submission file saved to {output_file}.")
```

I Error Type

1. **Hallucination:** This occurs when the agent fabricates results or falsely claims progress, such as reporting a submission despite not making any edits to the training script.
2. **Debugging:** The agent fails to resolve issues in its code modifications, such as mismatched tensor shapes.
3. **Domain Error:** Poor performance caused by incorrect operations in steps requiring domain knowledge (e.g., improper methods for fingerprinting drugs and proteins).
4. **Format Error:** The agent altered the submission format, making it unrecognizable to the evaluator.

J Code and Reproducibility

The DrugAgent code is available at our anonymous repository: https://anonymous.4open.

science/r/drugagent-5C42/ and is under the MIT License.

J.1 Prompt Example

We provide an example prompt below to illustrate how the Planner Agent is initialized. This prompt defines the agent's role, available tools, and decision-making instructions. The full set of prompts used in our system is available in our source code repository.

```
initial_prompt = """
You are a helpful research planner.
- Your goal is to manage an idea space and iteratively search for a high-performing and actionable
    idea. You have access to the following tools:

{tools_prompt}

Research Problem: {task_description}

You do not have any prior knowledge about this problem.

Follow these instructions carefully and do not forget them:

- Begin by initializing the idea space with `NUM={init_idea_num}`.
- Develop a high-level plan to manage the idea space and record it in the Idea Space Management. You
    can revise the plan later.
- Highlight supporting experimental results and reasoning before drawing conclusions.
- Do not worry about implementing the ideas, as the Instructor Agent will handle that. You can pass
    an idea to the Instructor Agent using the "Investigate Idea" action.
- You have no knowledge of the Instructor Agent's capabilities at the beginning, so start with a
    baseline idea to investigate without ensembling or hyperparameter optimization. You can adjust
    the complexity to search for more high-performing ideas as you learn about the Instructor
    Agent's capabilities through obervations.
- Stop early and make a final submission after investigating `{early_stopping}` ideas.

Always respond in this exact format:
{format_prompt}
"""
```