EHRFlow: A Large Language Model-Driven Iterative Multi-Agent Electronic Health Record Data Analysis Workflow

Hao Wu*
Peking University
Beijing, China
xhawborn@gmail.com

Yinghao Zhu*
Peking University
Beijing, China
yhzhu99@gmail.com

Zixiang Wang
Peking University
Beijing, China
zixiangwang@buaa.edu.cn

Xiaochen Zheng
ETH Zürich
Zürich, Switzerland
xzheng@ethz.ch

Ling Wang
Affiliated Xuzhou Municipal Hospital of Xuzhou Medical University
Xuzhou, Jiangsu, China

Wen Tang
Peking University Third Hospital
Beijing, China

Yasha Wang
Peking University
Beijing, China
wangyasha@pku.edu.cn

Chengwei Pan
Beihang University
Beijing, China
pancw@buaa.edu.cn

Ewen M. Harrison
University of Edinburgh
Edinburgh, Scotland, UK
ewen.harrison@ed.ac.uk

Junyi Gao†
University of Edinburgh
Health Data Research UK
Edinburgh, Scotland, UK
junyi.gao@ed.ac.uk

Liantao Ma†
Peking University
Beijing, China
malt@pku.edu.cn

ABSTRACT

`EHRFlow` is a large language model-driven platform that simplifies electronic health record (EHR) data analysis for physicians through natural language interactions, eliminating the need for complex coding. `EHRFlow` integrates four key agents: PlanAgent, ToolAgent, CodeAgent, and ReviewAgent in a dual-loop mechanism that efficiently decomposes and executes tasks via PlanAgent, from coarse-grained conceptual planning to fine-grained task execution, leveraging a healthcare tool bank and the code generation and execution capabilities of ToolAgent and CodeAgent. The ReviewAgent ensures the safety and quality of code through iterative feedback and refinement. `EHRFlow` offers a user-friendly solution for physicians' customized needs while preserving sensitive data's privacy by executing code operations locally. Demonstrated through case studies, `EHRFlow` showcases its ability to coordinate agents in resolving complex medical data analysis tasks, highlighting its flexibility, scalability, and efficiency. The code for `EHRFlow` is publicly released at https://github.com/PKU-AICare/EHRFlow.

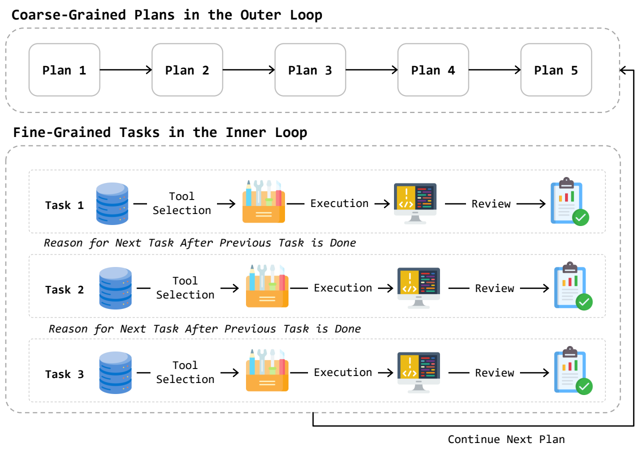

Coarse-Grained Plans in the Outer Loop
Plan 1
Plan 2
Plan 3
Plan 4
Plan 5
Fine-Grained Tasks in the Inner Loop
Task 1
Tool Selection
Execution
Review
Reason for Next Task After Previous Task is Done
Task 2
Tool Selection
Execution
Review
Reason for Next Task After Previous Task is Done
Task 3
Tool Selection
Execution
Review
Continue Next Plan

Figure 1: Illustration of plan and task decomposition and execution pipeline in the loop.

1 INTRODUCTION

Electronic health records (EHRs) are comprehensive digital repositories of health information [1, 6, 12]. Leveraging machine learning and deep learning approaches, valuable knowledge can be mined from EHR data to aid physicians in analyzing patients' health conditions. These approaches support healthcare in various ways, including personalized treatments, disease prediction, clinical research, public health surveillance, and resource allocation [5, 23, 24]. Despite the potential benefits, the development of AI models and the analysis of health data are typically conducted by health data scientists with a computer science background. Physicians, on the other hand, often have limited skills in complex EHR data analysis. Consequently, effective EHR data analysis requires close collaboration and frequent communication between physicians and AI experts. Physicians have expressed significant interest in analyzing their specific domain EHR data to enhance their clinical decision-making processes [11]. However, the skill gap between physicians and data scientists presents a challenge.

This gap motivates us to develop a platform that allows physicians to enter their data analysis requirements in natural language. We envision that this platform could interpret these requirements and execute the corresponding data analysis tasks automatically, without human intervention. As indicated in [4], physicians expect the digital agents to integrate into their clinical workflows through interactive conversations, expecting these AI tools to enhance their work efficiency and effectiveness.

*Equal contribution.
†Corresponding author.

The recent advent of large language models (LLMs) [14] with their instruction-following, reasoning, and function-calling capacities, has made implementing such a platform possible. Specifically, building upon the LLM's capacities, the LLM agent could form an autonomous proxy system that integrates other tools and technologies to enhance model performance and execute tasks efficiently[19]. For example, AutoGPT [7] showcases its capability in calling functions and executing code to achieve users' software development use cases, such as building a front-end website. However, such LLM agent systems cannot be directly applied in the medical domain where physicians have much more customized needs and privacy concerns. The challenges include: (1) For customized and professional use cases, LLM cannot understand the inherent processes without prompting the LLM with domain-specific knowledge. (2) Healthcare tasks are sometimes quite complicated and require precise and rigorous execution results, while conventional agent systems are prone to collapse. (3) Privacy matters in healthcare data, posing challenges in designing a privacy-preserved approach so that no patient data is uploaded when calling LLM's online APIs.

Aiming to address these challenges, we introduce `EHRFlow`, an iterative multi-agent system supported by LLMs, designed to serve for physicians' diverse EHR analysis requirements in their clinical workflows. The web interface is demonstrated in Figure 5. As illustrated in Figure 2, `EHRFlow` consists of four main agents: PlanAgent, ToolAgent, CodeAgent, and ReviewAgent. Inspired by XAgent [17], we refine the dual-loop mechanism of task decomposition and dynamic execution in the healthcare domain. The outer loop conceptualizes complex tasks into plans, and each plan is further split into lower-level tasks by the PlanAgent one by one. The inner loop refines and executes the task using tools retrieved from the healthcare tool bank, leveraging ToolAgent's reasoning and CodeAgent's code generation and execution capabilities. After finishing the task, PlanAgent continues to reason for the next task, as demonstrated in Figure 1. Meanwhile, the ReviewAgent examines the execution results and provides reflection and feedback to the PlanAgent and ToolAgent. Moreover, `EHRFlow` protects patients' privacy by only providing data schema, and all sensitive data-related operations are executed locally.

`EHRFlow`'s contributions can be summarized as follows:

(1) We develop and deploy `EHRFlow` to assist physicians' data analysis requirements through simple natural language interactions, lowering the threshold with no need to write code. `EHRFlow` has a user-friendly interface, meets the needs of professional clinical use cases, and is scalable for more customized use cases by registering tool APIs in the tool bank and providing usage illustrations.
(2) `EHRFlow` tailors a multi-agent dual-loop mechanism of coarse-grained and fine-grained task decomposition specifically designed for the healthcare domain. This mechanism ensures effective management and optimization of complex medical tasks with privacy protection through the concept decomposition and sub-plan generation of both loops, as well as reviewing, reflection, and feedback correction within the each loop.
(3) Code for the tool banks and `EHRFlow` has been publicly released, offering transparent and reliable resources for the health data science community.

2 RELATED WORK

2.1 Healthcare Data Analytics Toolkits

Open-source projects like PyHealth [20] and AutoPrognosis [10] are two Python toolkits specifically designed for EHR data, simplifies deep learning integration for healthcare data analysis, supporting various data modalities and models. PyHealth focuses on clinical predictive modeling, mainly designed for both ML researchers and medical practitioners, while AutoPrognosis additionally offers more customized model explanations and cohort analysis features.

However, the gap remains between physicians and code-based data analysis workflow, a seamless data analysis and physician-driven data decisions require further development, a lower-threshold workflow is still to be implemented [4].

2.2 LLM-Driven Agent-Based Data Analytics

As Large Language Models (LLMs) have demonstrated significant intelligence, leveraging them as planning modules for autonomous agents has gained increased attention [9]. This paradigm involves generating both reasoning traces and task-specific actions in an interleaved manner, allowing for greater synergy between the two. Reasoning traces help the model induce, track, and update action plans as well as handle exceptions, while actions enable it to interface with external sources, such as knowledge bases or environments, to gather additional information [21].

Building on this foundation, multiple agents in a collaborative environment, each with distinct attributes and roles, can handle larger-scale and more complex tasks [16]. For example, TaskWeaver [13] introduces stateful execution and code verification to validate generated code before execution, thereby preventing potential issues. Similarly, XAgent incorporates reflection to provide feedback to other agents, which is useful for identifying bugs and making refinements [17].

In the healthcare domain, EHRAgent [15] automates EHR multitable inference tasks from natural language instructions, simplifying interactions for healthcare workers with EHR systems. Additionally, Cui et al. [2] explore the feasibility of LLMs in few-shot EHR disease prediction, proposing frameworks with predictive and critical agents for narrative generation and error feedback to improve accuracy. However, existing LLM agent work in EHR data has limitations, particularly in tool usage, as it primarily relies on the LLM's own reasoning capabilities.

3 METHODOLOGY

Figure 2 illustrates the overall framework of `EHRFlow`, showcasing its hierarchical dual-loop processes. The outer loop is responsible for coarse-grained planning, while the inner loop focuses on fine-grained task execution.

3.1 Conceptual Decomposition-Centric Reasoning, Planning and Code Execution

User requirements typically involve multiple interrelated entities and concepts, forming a complex semantic concept network. Although large language models have high capabilities in understanding intent and reasoning, they still face challenges in dealing with these complex coupled conceptual systems. At the understanding

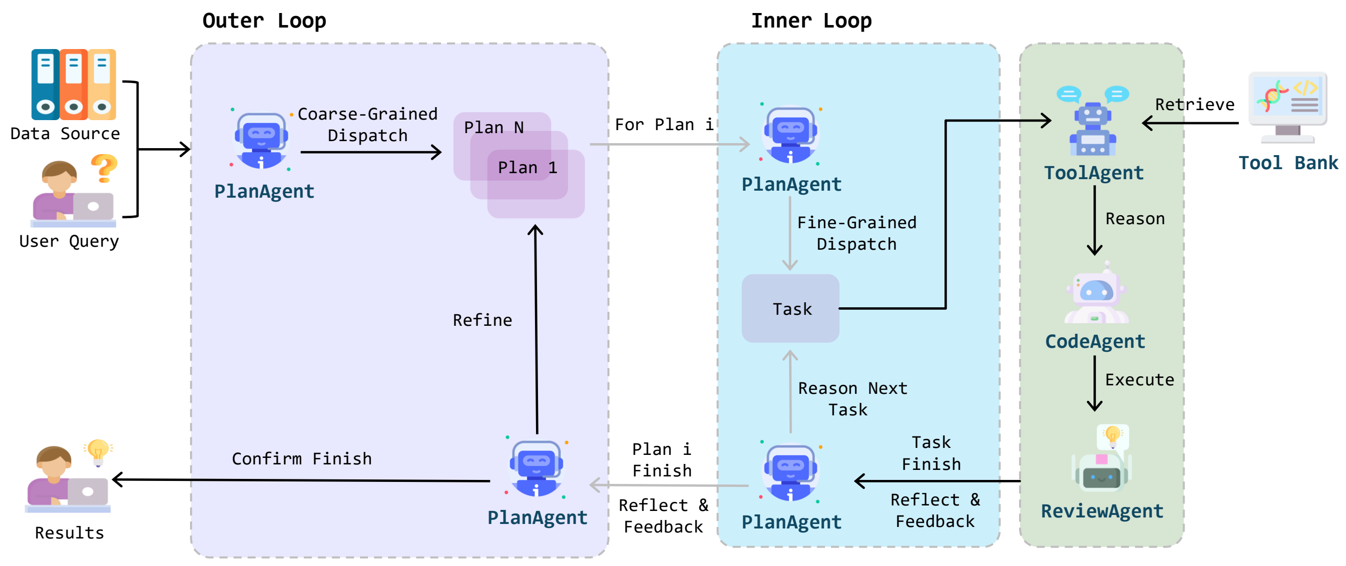


Figure 2: Overall framework of EHRFlow. EHRFlow accepts user's natural language queries, and consists of the hierarchical dual-loop processes: the outer loop for coarse-grained planning and the inner loop mainly for fine-grained task executing.

level, the complexity of coupled concepts may make it difficult for the model to quickly identify relationships and meanings between entities, especially when the requirements are unclear or the concept boundaries are fuzzy. At the execution level, transforming abstract concepts into concrete solutions is a challenge, especially in scenarios requiring the integration of cross-domain knowledge and multi-step logical reasoning.

Inspired by the Chain of Thought (CoT) [18] and Rephrase and Refine (RaR) [3] prompt strategies, targeted prompts are designed to guide the model in presenting its reasoning overview. This process includes steps such as question restatement, key concept identification, conceptual decomposition, reflection, deep thinking, reasoning, and action planning. EHRFlow adapts the decomposition of medical tasks into dual granularity levels from coarse in the outer loop to fine in the inner loop. As shown in Figure 1, the PlanAgent decomposes the user query instruction into multiple coarse-grained plans in the outer loop. Within the inner loop of each plan, the PlanAgent further decomposes into the fine-grained executable task, where ToolAgent reasons for choosing required tool APIs from the tool bank and the CodeAgent generates glue code and executes the code given the function calling of healthcare tool interfaces. After completing code execution, ReviewAgent checks the current status, reflects, and provides feedback to PlanAgent. If everything is on track, then PlanAgent continues to reason for the next task towards the target of the current plan.

During the above process, the prompt for each component also follows the strategy of concept identification and decomposition, reflection, reasoning, etc., and is adjusted appropriately based on the complexity of the tasks to support customized diverse medical task decomposition and arrangement. Our elaborately designed prompts ensure that the system can flexibly handle tasks of various complexities, optimizing the entire decision-making process from macro planning to micro execution.

Note that during the execution of tasks, the CodeAgent continuously maintains the code execution status throughout the dialogue process, similar to the programming mode in Jupyter Notebook [13], ensuring the coherence and consistency of code execution.

Furthermore, due to the sensitive nature of healthcare data, during the agent reasoning process, the context only includes the data descriptions, i.e., the column names indicating all features, and the data recorded format. The large language model has the ability to generate the correct code with these data descriptions as background for data analysis operations.

3.2 Reflection and Feedback-Based Dynamic Execution Mechanism

To address the challenges posed by the complexity of the environment and the potential issues of safety, harmlessness, and executability in the code generated by large language models for EHR data analysis, EHRFlow introduces a ReviewAgent to establish a reflection and feedback phase. This mechanism is designed to monitor each step's execution results and dynamically adjust based on the differences between actual and expected outcomes.

Speficically, the PlanAgent in the outer loop acts as a project manager, reviewing the execution results of sub-tasks and adjusting planned steps based on feedback. This aims to address issues and anomalies during execution, avoid fault loops, and seek alternative solutions, thereby facilitating the smooth progress of tasks. Additionally, the ReviewAgent conducts a comprehensive review after code generation, screening for harmful information, verifying the code's executability, evaluating its alignment with user requirements, reviewing code structure and style, and checking for exception handling and error handling mechanisms.

In the event that the code fails the review, the ReviewAgent provides clear review results and feedback to the large language model, initiating a new cycle of code generation and review. This iterative process continues until the code fully meets the required standards, aiming to improve code quality, enhance trustworthiness, and provide doctors with a more reliable data analysis tool.

By obtaining perceptual information through real-time interaction and combining it with actual trial results, EHRFlow's feedback loop is designed to enable the workflow to quickly self-correct when facing inefficient planning or continuous errors. This transition from relying on preset perceptual information to real-time dynamic adjustment seeks to enhance the comprehensiveness and effectiveness of planning, strengthen adaptability and robustness in dynamic environments, and ultimately enable large language models to make more effective decisions and responses for new complex tasks.

3.3 Retrieval-Enhanced Healthcare Analysis Tool Bank Design

In the process of using EHR datasets to assist medical decision-making, we face challenges due to the inherent complexity of the data and the need for in-depth analysis. Therefore, meticulous and customized preprocessing of EHR data is essential to ensure that the data quality meets the standards for advanced analysis. Although EHR data is commonly stored in CSV file format, basic processing of CSV data alone is insufficient to meet the requirements of complex analysis. This approach can lead to misunderstandings and misjudgments of the information. Hence, a series of steps including data cleaning, transformation, integration, and advanced analysis are necessary to accurately extract and present the key information in EHR data. Additionally, existing large language models show limitations in handling EHR data. While these models can handle some basic algorithms, they currently do not provide sufficient support for the complex predictive analysis required for EHR data.

To address the challenges of medical data analysis, we have adopted a functional decoupling strategy, separating the key functions of EHR data analysis from the system and designing the API standard within the tool bank specifically for the core steps and requirements of EHR data analysis with built-in basic data analysis knowledge and functions. On this basis, we build the specialized agent ToolAgent to understand which tools in the tool bank are helpful for targeted solutions. CodeAgent calls these functions with glue code to smoothly execute them.

In addition, to meet professional physicians' customized advanced EHR data analysis requirements, besides various built-in medical data analysis tools and APIs in the tool bank, it is also easy to register customized use cases into the tool bank by providing related code and instructions for their usage.

4 CASE STUDIES

We provide two case studies: Figure 3 showcases how EHRFlow successfully calculates the male and female ratio in a specific dataset. Figure 4 illustrates a more advanced scenario that plots the histogram of patient feature importance, which includes the training of a complex deep learning model. These two cases demonstrate EHRFlow's superior task-solving ability in complicated tasks, while only natural language is input to EHRFlow. The detailed version of the case studies is provided in the Appendix.

5 FUTURE WORK

This study provides a preliminary exploration of the application of large language models in the analysis of structured medical data, focusing on EHR data and prognostic prediction tasks of interest to doctors, and systematically designs and implements relevant technologies. The research not only provides a new perspective for the field of medical data analysis but also demonstrates the potential of large language models in processing medical data. However, given the complexity of the field and the continuous advancement of technology, future research needs to delve deeper into and refine aspects such as application-oriented process decomposition, user interaction experience optimization, security and privacy protection, and multimodal data processing. This includes clarifying the boundaries of technical capabilities, expanding system functions, optimizing user engagement processes, innovating user interface design, strengthening data security and privacy protection measures, and developing algorithms and models that integrate multimodal data to enhance the overall performance and clinical application value of the system.

6 CONCLUSIONS

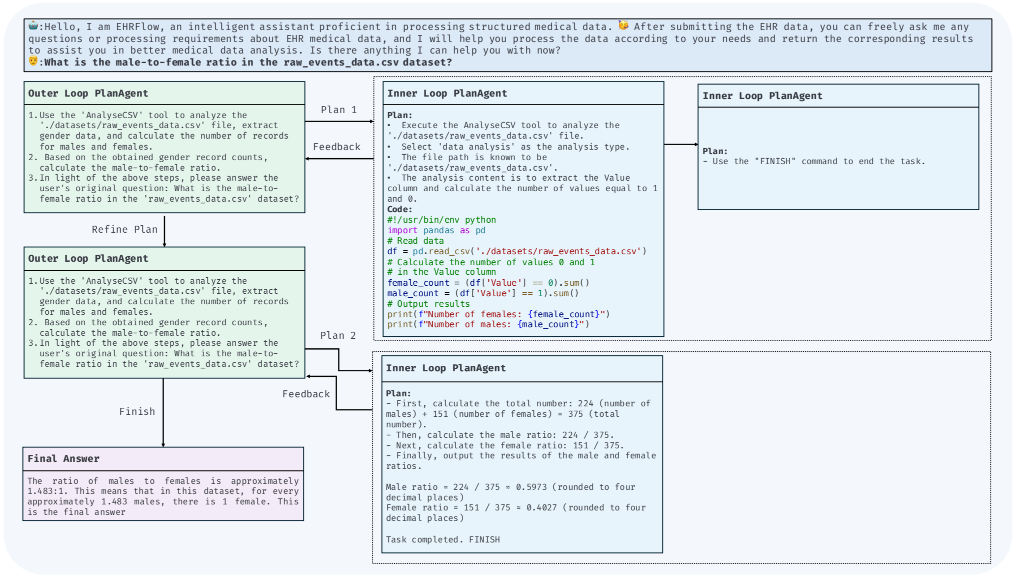
This study develops a user-friendly electronic health record (EHR) data analysis platform powered by a large language model. With only natural language interaction, physicians could conduct complex data analysis tasks in code-free manner. EHRFlow applies a multi-agent collaborative mechanism, achieving fine-grained task decomposition and dynamic execution based on user needs, as well as deep customization and optimization of EHR data. The system design is user-oriented, allowing users to express personalized needs through natural language interaction and provide users with an intuitive and flexible data analysis experience. EHRFlow showcases the application potential and practical value of large language models in the field of medical data analysis.

7 DISCUSSION ON ALIGNING EHRFLOW WITH AIDSH WORKSHOP SCOPE

The "Artificial Intelligence and Data Science for Healthcare (AIDSH)" workshop is at the forefront of exploring the integration of AI and DS within the healthcare sector, with a particular focus on bridging the gap between data-centric AI and people-centric healthcare (PCHC). The workshop is dedicated to fostering innovative applications of AI and DS that not only advance healthcare practices but also actively engage individuals in managing their health data.

Our paper, EHRFlow, aligns with the AIDSH workshop's research scope in several ways:

- **Enhancing Clinical Decision-Making**: EHRFlow is designed to empower physicians with an intuitive platform that enables them to harness the power of AI through natural language interactions. By facilitating the analysis of EHR data, EHRFlow supports personalized treatments, disease prediction, and clinical research, which are in line with the workshop's goals of driving healthcare innovation with AI and data science.
- **Advancing People-Centric Healthcare (PCHC)**: EHRFlow embodies the PCHC philosophy by integrating digital agents into clinical workflows through interactive conversations. It addresses the skill gap between physicians and data scientists by



:Hello, I am EHRFlow, an intelligent assistant proficient in processing structured medical data. After submitting the EHR data, you can freely ask me any questions or processing requirements about EHR medical data, and I will help you process the data according to your needs and return the corresponding results to assist you in better medical data analysis. Is there anything I can help you with now?
:What is the male-to-female ratio in the raw_events_data.csv dataset?

Outer Loop PlanAgent

1. Use the 'AnalyseCSV' tool to analyze the './datasets/raw_events_data.csv' file, extract gender data, and calculate the number of records for males and females.
2. Based on the obtained gender record counts, calculate the male-to-female ratio.
3. In light of the above steps, please answer the user's original question: What is the male-to-female ratio in the 'raw_events_data.csv' dataset?

Inner Loop PlanAgent

Plan:
- Execute the AnalyseCSV tool to analyze the './datasets/raw_events_data.csv' file.
- Select 'data analysis' as the analysis type.
- The file path is known to be './datasets/raw_events_data.csv'.
- The analysis content is to extract the Value column and calculate the number of values equal to 1 and 0.

Code:
```python
#!/usr/bin/env python
import pandas as pd
# Read data
df = pd.read_csv('./datasets/raw_events_data.csv')
# Calculate the number of values 0 and 1
# in the Value column
female_count = (df['Value'] == 0).sum()
male_count = (df['Value'] == 1).sum()
# Output results
print(f"Number of females: {female_count}")
print(f"Number of males: {male_count}")
```

Inner Loop PlanAgent

Plan:
- Use the "FINISH" command to end the task.

Outer Loop PlanAgent

1. Use the 'AnalyseCSV' tool to analyze the './datasets/raw_events_data.csv' file, extract gender data, and calculate the number of records for males and females.
2. Based on the obtained gender record counts, calculate the male-to-female ratio.
3. In light of the above steps, please answer the user's original question: What is the male-to-female ratio in the 'raw_events_data.csv' dataset?

Inner Loop PlanAgent

Plan:
- First, calculate the total number: 224 (number of males) + 151 (number of females) = 375 (total number).
- Then, calculate the male ratio: 224 / 375.
- Next, calculate the female ratio: 151 / 375.
- Finally, output the results of the male and female ratios.

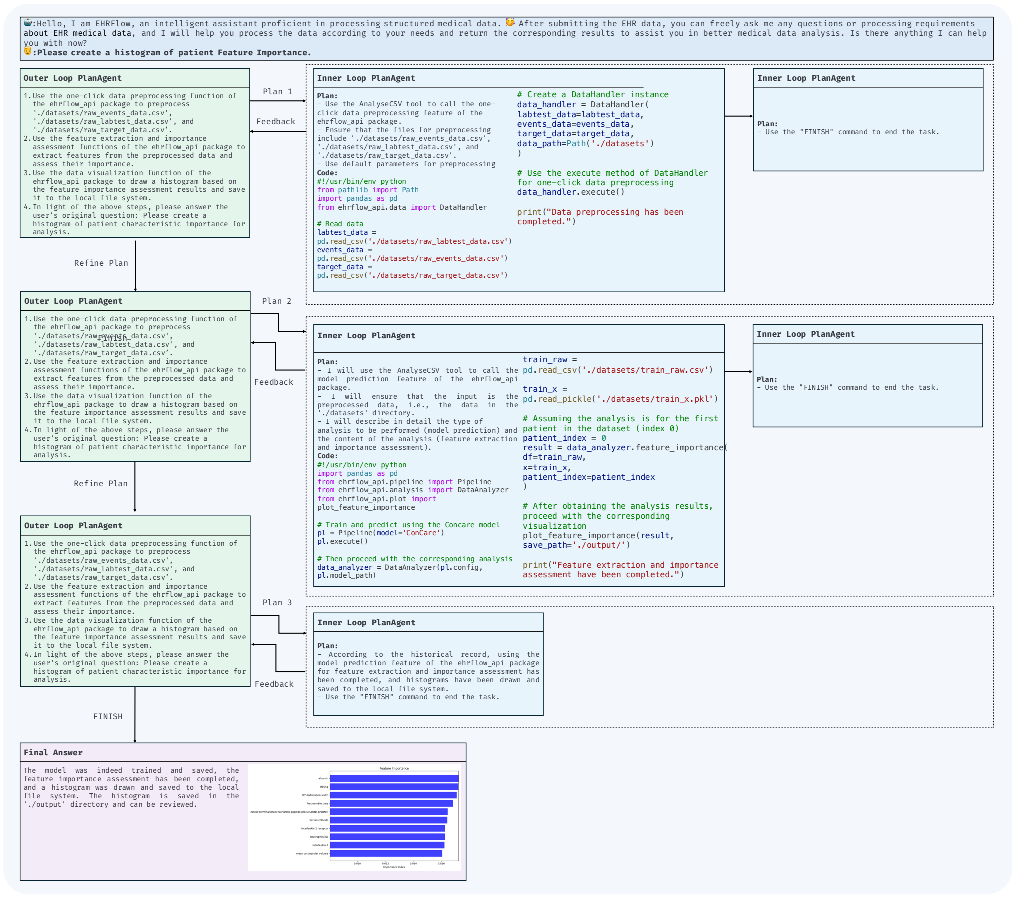
Male ratio = 224 / 375 = 0.5973 (rounded to four decimal places)
Female ratio = 151 / 375 = 0.4027 (rounded to four decimal places)

Task completed. FINISH

Final Answer

The ratio of males to females is approximately 1.483:1. This means that in this dataset, for every approximately 1.483 males, there is 1 female. This is the final answer

Figure 3: EHRFlow successfully calculates the male and female ratio in a specific dataset.



:Hello, I am EHRFlow, an intelligent assistant proficient in processing structured medical data. After submitting the EHR data, you can freely ask me any questions or processing requirements about EHR medical data, and I will help you process the data according to your needs and return the corresponding results to assist you in better medical data analysis. Is there anything I can help you with now?
:Please create a histogram of patient Feature Importance.

Outer Loop PlanAgent

1. Use the one-click data preprocessing function of the ehrflow_api package to preprocess './datasets/raw_events_data.csv', './datasets/raw_labtest_data.csv', and './datasets/raw_target_data.csv'.
2. Use the feature extraction and importance assessment functions of the ehrflow_api package to extract features from the preprocessed data and assess their importance.
3. Use the data visualization function of the ehrflow_api package to draw a histogram based on the feature importance assessment results and save it to the local file system.
4. In light of the above steps, please answer the user's original question: Please create a histogram of patient characteristic importance for analysis.

Inner Loop PlanAgent

Plan:
- Use the AnalyseCSV tool to call the one-click data preprocessing feature of the ehrflow_api package.
- Ensure that the files for preprocessing include './datasets/raw_events_data.csv', './datasets/raw_labtest_data.csv', and './datasets/raw_target_data.csv'.
- Use default parameters for preprocessing

Code:
```python
#!/usr/bin/env python
from pathlib import Path
import pandas as pd
from ehrflow_api.data import DataHandler

# Read data
labtest_data =
pd.read_csv('./datasets/raw_labtest_data.csv')
events_data =
pd.read_csv('./datasets/raw_events_data.csv')
target_data =
pd.read_csv('./datasets/raw_target_data.csv')

# Create a DataHandler instance
data_handler = DataHandler(
labtest_data=labtest_data,
events_data=events_data,
target_data=target_data,
data_path=Path('./datasets')
)

# Use the execute method of DataHandler
for one-click data preprocessing
data_handler.execute()

print("Data preprocessing has been
completed.")
```

Inner Loop PlanAgent

Plan:
- Use the "FINISH" command to end the task.

Outer Loop PlanAgent

1. Use the one-click data preprocessing function of the ehrflow_api package to preprocess './datasets/raw_events_data.csv', './datasets/raw_labtest_data.csv', and './datasets/raw_target_data.csv'.
2. Use the feature extraction and importance assessment functions of the ehrflow_api package to extract features from the preprocessed data and assess their importance.
3. Use the data visualization function of the ehrflow_api package to draw a histogram based on the feature importance assessment results and save it to the local file system.
4. In light of the above steps, please answer the user's original question: Please create a histogram of patient characteristic importance for analysis.

Inner Loop PlanAgent

Plan:
- I will use the AnalyseCSV tool to call the model prediction feature of the ehrflow_api package.
- I will ensure that the input is the preprocessed data, i.e., the data in the './datasets' directory.
- I will describe in detail the type of analysis to be performed (model prediction) and the content of the analysis (feature extraction and importance assessment).

Code:
```python
#!/usr/bin/env python
import pandas as pd
from ehrflow_api.pipeline import Pipeline
from ehrflow_api.analysis import DataAnalyzer
from ehrflow_api.plot import
plot_feature_importance

# Train and predict using the Concare model
pl = Pipeline(model='ConCare')
pl.execute()

# Then proceed with the corresponding analysis
data_analyzer = DataAnalyzer(pl.config,
pl.model_path)

train_raw =
pd.read_csv('./datasets/train_raw.csv')

train_x =
pd.read_pickle('./datasets/train_x.pkl')

# Assuming the analysis is for the first
patient in the dataset (index 0)
patient_index = 0
result = data_analyzer.feature_importance(
df=train_raw,
x=train_x,
patient_index=patient_index
)

# After obtaining the analysis results,
proceed with the corresponding
visualization
plot_feature_importance(result,
save_path='./output/')

print("Feature extraction and importance
assessment have been completed.")
```

Inner Loop PlanAgent

Plan:
- Use the "FINISH" command to end the task.

Outer Loop PlanAgent

1. Use the one-click data preprocessing function of the ehrflow_api package to preprocess './datasets/raw_events_data.csv', './datasets/raw_labtest_data.csv', and './datasets/raw_target_data.csv'.
2. Use the feature extraction and importance assessment functions of the ehrflow_api package to extract features from the preprocessed data and assess their importance.
3. Use the data visualization function of the ehrflow_api package to draw a histogram based on the feature importance assessment results and save it to the local file system.
4. In light of the above steps, please answer the user's original question: Please create a histogram of patient characteristic importance for analysis.

Inner Loop PlanAgent

Plan:
- According to the historical record, using the model prediction feature of the ehrflow_api package for feature extraction and importance assessment has been completed, and histograms have been drawn and saved to the local file system.
- Use the "FINISH" command to end the task.

Final Answer

The model was indeed trained and saved, the feature importance assessment has been completed, and a histogram was drawn and saved to the local file system. The histogram is saved in the './output' directory and can be reviewed.

Figure 4: EHRFlow successfully plots the histogram of patient feature importance.

providing a platform that requires no coding knowledge. This user-friendly interface encourages collaboration by enabling physicians to specify their data analysis needs directly, thereby promoting active participation in the clinical process.

- **Education Support**: By making the code for the tool banks and `EHRFlow` publicly available, we provide transparent and reliable resources used for educational purposes. This aligns with the workshop's emphasis on data education and support, empowering healthcare professionals to leverage AI and data science in practices.
- **Privacy Protection**: `EHRFlow`'s approach to privacy protection, where only data schema are provided and all sensitive operations are executed locally, aligns with the growing concern for data security in healthcare. This commitment to privacy is a critical aspect of the workshop's focus on responsible AI applications.

In summary, `EHRFlow` not only contributes to the advancement of EHR data analysis but also fits seamlessly within the research scope of the AIDSH workshop. It represents a significant step towards realizing the vision of data-centric AI that is also people-centric, providing a powerful, accessible, and privacy-conscious tool to enhance clinical workflows and decision-making.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (U23A20468), and Xuzhou Scientific Technological Projects (KC23143). Junyi Gao acknowledges the receipt of studentship awards from the Health Data Research UK-The Alan Turing Institute Wellcome PhD Programme in Health Data Science (grant 218529/Z/19/Z).

REFERENCES

[1] Euiyul Choi, Mohammad Taha Bahadori, Jimeng Sun, John Kulas, Andrew Schuetz, and Walter Stewart. 2016. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. *Advances in neural information processing systems* 29 (2016).

[2] H Cui, Z Shen, J Zhang, and et al. 2024. LLMs-based Few-Shot Disease Predictions using EHR: A Novel Approach Combining Predictive Agent Reasoning and Critical Agent Instruction. *arXiv preprint arXiv:2403.15464* (2024). [Online; accessed <date of access>].

[3] Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. 2023. Rephrase and respond: Let large language models ask better questions for themselves. *arXiv preprint arXiv:2311.04205* (2023).

[4] Andri Färber, Christiane Schwabe, Philipp H Stalder, Mateusz Dolata, and Gerhard Schwabe. 2024. Physicians' and Patients' Expectations From Digital Agents for Consultations: Interview Study Among Physicians and Patients. *JMIR Human Factors* 11 (2024), e49647.

[5] Jing Gao, Chen Xiao, Yifan Wang, Lee M Glass, and Jimeng Sun. 2020. Stagenet: Stage-aware neural networks for health risk prediction. In *Proceedings of The Web Conference 2020*. 530–540.

[6] Junyi Gao, Yinghao Zhu, Wenqing Wang, Zixiang Wang, Guiying Dong, Wen Tang, Hao Wang, Yasha Wang, Ewen M. Harrison, and Liantao Ma. 2024. A comprehensive benchmark for COVID-19 predictive modeling using electronic health records in intensive care. *Patterns* (2024), 100951. https://doi.org/10.1016/j.patter.2024.100951

[7] Significant Gravitas. 2024. *AutoGPT*. https://agpt.co A collection of tools and experimental open-source attempts to make GPT-4 fully autonomous..

[8] Mehak Gupta, Brennan Gallamoza, Nicolas Cutrona, Pranjal Dhakal, Raphael Poulain, and Rahmatollah Beheshti. 2022. An extensive data processing pipeline for mimic-iv. In *Machine Learning for Health*. PMLR, 311–325.

[9] Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the planning of LLM agents: A survey. *arXiv preprint arXiv:2402.02716* (2024).

[10] Fergus Imrie, Bogdan Cebere, Eoin F McKinney, and Mihaela van der Schaar. 2023. AutoPrognosis 2.0: Democratizing diagnostic and prognostic modeling in healthcare with automated machine learning. *PLOS Digital Health* 2, 6 (2023), e0000276.

[11] X Liu, D McDuff, G Kovacs, et al. 2023. Large language models are few-shot health learners. *arXiv preprint arXiv:2305.15525* (2023).

[12] Liantao Ma, Chaohe Zhang, Junyi Gao, Xianfeng Jiao, Zhihao Yu, Yinghao Zhu, Tianlong Wang, Xinyu Ma, Yasha Wang, Wen Tang, Xinju Zhao, Wenjie Ruan, and Tao Wang. 2023. Mortality prediction with adaptive feature importance recalibration for peritoneal dialysis patients. *Patterns* 4, 12 (2023), 100892. https://doi.org/10.1016/j.patter.2023.100892

[13] Bo Qiao, Liqun Li, Xu Zhang, Shilin He, Yu Kang, Chaoyun Zhang, Fangkai Yang, Hang Dong, Jue Zhang, Lu Wang, et al. 2023. Taskweaver: A code-first agent framework. *arXiv preprint arXiv:2311.17541* (2023).

[14] P P Ray. 2023. ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems* (2023).

[15] Wenqi Shi, Ran Xu, Yuchen Zhuang, Yue Yu, Jieyu Zhang, Hang Wu, Yuanda Zhu, Joyce C Ho, Carl Yang, and May Dongmei Wang. 2024. EHRAgent: Code Empowers Large Language Models for Few-shot Complex Tabular Reasoning on Electronic Health Records. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

[16] Yashar Talebirad and Amirhossein Nadiri. 2023. Multi-agent collaboration: Harnessing the power of intelligent llm agents. *arXiv preprint arXiv:2306.03314* (2023).

[17] XAgent Team. 2023. XAgent: An Autonomous LLM Agent for Complex Task Solving. https://github.com/OpenBMB/XAgent. (Accessed: June 4, 2024).

[18] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* 35 (2022), 24824–24837.

[19] Z Xi, W Chen, X Guo, et al. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864* (2023).

[20] Chaoqi Yang, Zhenbang Wu, Patrick Jiang, Zhen Lin, Junyi Gao, Benjamin P Danek, and Jimeng Sun. 2023. Pyhealth: A deep learning toolkit for healthcare applications. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5788–5789.

[21] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629* (2022).

[22] Hugo Yèche, Rita Kuznetsova, Marc Zimmermann, Matthias Hüser, Xinrui Lyu, Martin Faltys, and Gunnar Rätsch. 2021. HiRID-ICU-Benchmark–A Comprehensive Machine Learning Benchmark on High-resolution ICU Data. *arXiv preprint arXiv:2111.08536* (2021).

[23] Chaohe Zhang, Xu Chu, Liantao Ma, Yinghao Zhu, Yasha Wang, Jiangtao Wang, and Junfeng Zhao. 2022. M3Care: Learning with Missing Modalities in Multimodal Healthcare Data. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 2418–2428. https://doi.org/10.1145/3534678.3539388

[24] Yinghao Zhu, Zixiang Wang, Long He, Shiyun Xie, Liantao Ma, and Chengwei Pan. 2024. PRISM: Leveraging Prototype Patient Representations with Feature-Missing-Aware Calibration for EHR Data Sparsity Mitigation. arXiv:2309.04160 [cs.LG]

A WEB INTERFACE OF EHRFLOW

The web interface of EHRFlow is designed to be intuitive and user-friendly, allowing users to seamlessly interact with the model. It features a straightforward data upload mechanism and an input box where users can specify their requirements, offering a user experience akin to GPT-4 (see Figure 5).

B DETAILED VERSION OF CASE STUDIES

We provide a comprehensive version of case studies using EHRFlow. These detailed and complete versions delve deeper into the EHRFlow's inner loop and outer loop process, offering a richer understanding of EHRFlow (see Figures 6 and 7).

C DETAILS OF THE TOOL BANK

C.1 Development of EHRFlowAPI

We first designed the EHRFlowAPI, a set of customized interfaces tailored to the specific needs and processes of medical data analysis, aimed at providing bespoke solutions for EHR data processing and analysis. The modular design of EHRFlowAPI allows each module to focus on specific tasks within the data processing and analysis workflow, covering the entire process from data preprocessing, feature extraction, model training and prediction, to data analysis interpretation and result visualization [8, 22]. The development of EHRFlowAPI emphasizes simplifying the process of invoking complex models, particularly to meet the needs of core user groups such as frontline doctors who typically lack professional data analysis and programming knowledge. Therefore, ease of use is a key focus in the API design, with default values set for interface parameters to ensure smooth operation even if users do not provide detailed data analysis parameters. Additionally, the careful design and integration of the interfaces ensure continuity in the data analysis process, allowing users to complete the entire workflow smoothly even with incomplete parameters.

To further enhance operational convenience, we have developed a one-click execution interface that allows users to automatically complete a series of fundamental steps, including data preprocessing and analysis, with a single operation. This greatly lowers the technical barrier and improves the efficiency of data analysis. These design philosophies collectively ensure that users without a technical background can easily use the system for data analysis, while reducing the workload on the model during interface calls, significantly enhancing the user experience.

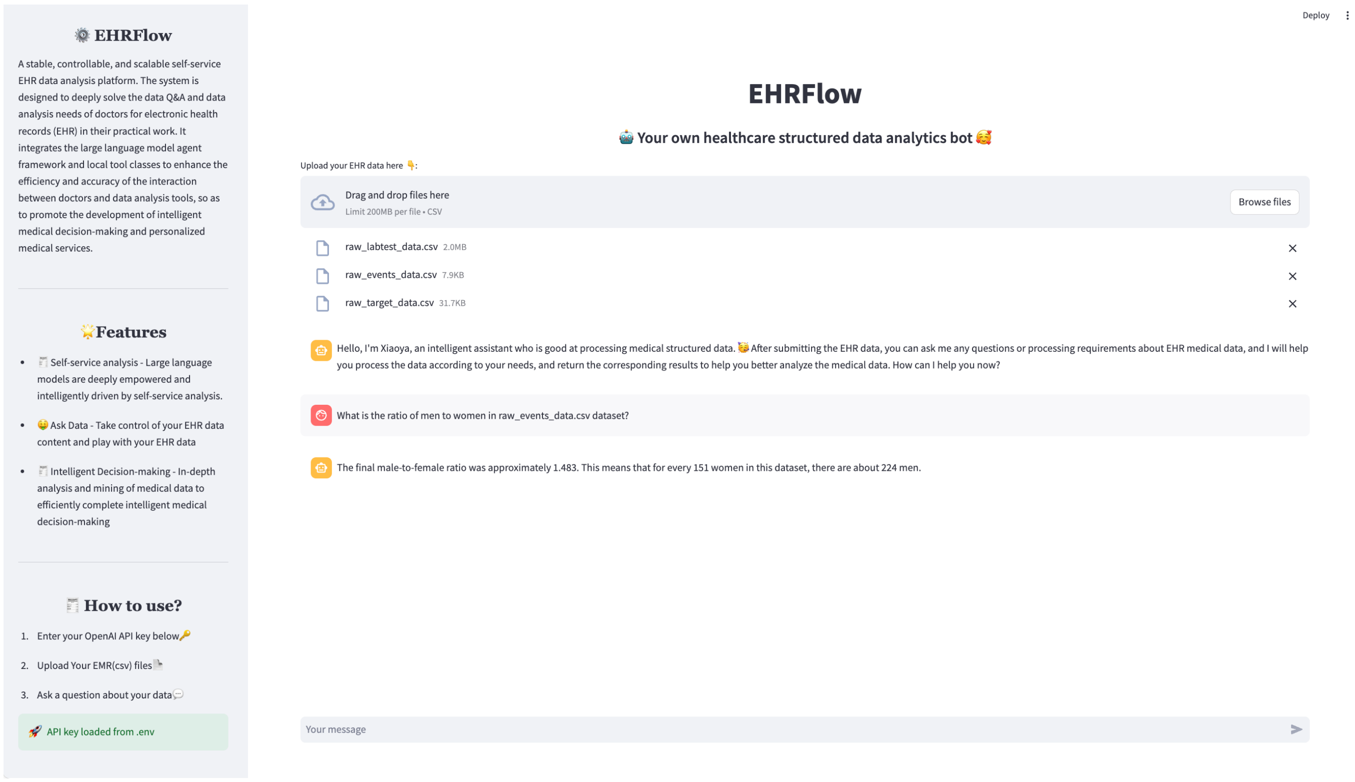
Through EHRFlowAPI, the role of ToolAgent shifts from generating complete analysis and prediction code to invoking these interfaces. This transition not only reduces the complexity of tasks handled by the large model but also improves the efficiency and executability of code generation, thereby optimizing the overall data analysis process.

C.2 Construction of the ToolAgent

After developing APIs that cater to the needs of medical data analysis and the characteristics of EHR data, we faced the challenge of enabling large language models to effectively understand and utilize these APIs. Directly inputting code into the model not only occupies a substantial amount of token space but also risks introducing unnecessary redundant information, hindering effective API invocation. To address this issue, we adopted an innovative approach: using natural language to describe the functionality, parameter requirements, and expected return results of the API in detail, instead of directly providing code. Utilizing the GPT-4 model, we generated natural language descriptions of the APIs, emphasizing clearly defining the role and parameter options of each method to ensure that the large language model accurately understands the purpose and operation of the code. Furthermore, we introduced a manual review process to verify the accuracy of the natural language descriptions, eliminating ambiguities and ensuring the model can precisely invoke the API interfaces, thereby enhancing the system's accuracy, reliability, applicability, and safety.

The EHRFlowAPI is designed to meet the core needs of EHR analysis and offers a rich set of interfaces. Despite the exhaustive natural language descriptions of these interfaces, the volume of textual information remains substantial. To manage this, after developing the EHRFlowAPI functionality prompt library based on GPT-4 generation and manual review, we implemented a filtering strategy, providing the model only with descriptions directly relevant to the current task. This ensures that the model efficiently and accurately processes the necessary information. In constructing the EHRFlow system, we established a standardized data analysis workflow: "data preprocessing—model prediction—data analysis—result visualization" to ensure effective API invocation and data processing accuracy. We also built a case library of medical data analysis, containing validated typical cases that cover various aspects of medical data analysis, ensuring the standardization and completeness of the code. ToolAgent can retrieve and integrate professional medical information based on user requirements, execute case and API information, and generate corresponding code solutions, enhancing the coherence of the analysis and execution process and the accuracy of code generation.

Addressing the limitations of large language models in handling long texts and maintaining contextual coherence, we developed a Python code interpreter tool integrated with Jupyter Notebook's interactive design. This tool supports seamless linkage and effective reuse of data and intermediate variables across steps while maintaining context history, significantly improving the precision and efficiency of analysis results. Through these comprehensive measures, we leverage the advantages of large language models in natural language processing while ensuring their efficiency and robustness in practical applications.



⚙️ EHRFlow

A stable, controllable, and scalable self-service EHR data analysis platform. The system is designed to deeply solve the data Q&A and data analysis needs of doctors for electronic health records (EHR) in their practical work. It integrates the large language model agent framework and local tool classes to enhance the efficiency and accuracy of the interaction between doctors and data analysis tools, so as to promote the development of intelligent medical decision-making and personalized medical services.

🌟Features

- 📊 Self-service analysis - Large language models are deeply empowered and intelligently driven by self-service analysis.
- 😜 Ask Data - Take control of your EHR data content and play with your EHR data
- 📊 Intelligent Decision-making - In-depth analysis and mining of medical data to efficiently complete intelligent medical decision-making

📊 How to use?

1. Enter your OpenAI API key below🔑
2. Upload Your EMR(csv) files📄
3. Ask a question about your data💬

🚀 API key loaded from .env

Deploy ⋮

EHRFlow

🤖 Your own healthcare structured data analytics bot 🥰

Upload your EHR data here 👇:

Drag and drop files here
Limit 200MB per file • CSV
Browse files

raw_labtest_data.csv 2.0MB ✕
raw_events_data.csv 7.9KB ✕
raw_target_data.csv 31.7KB ✕

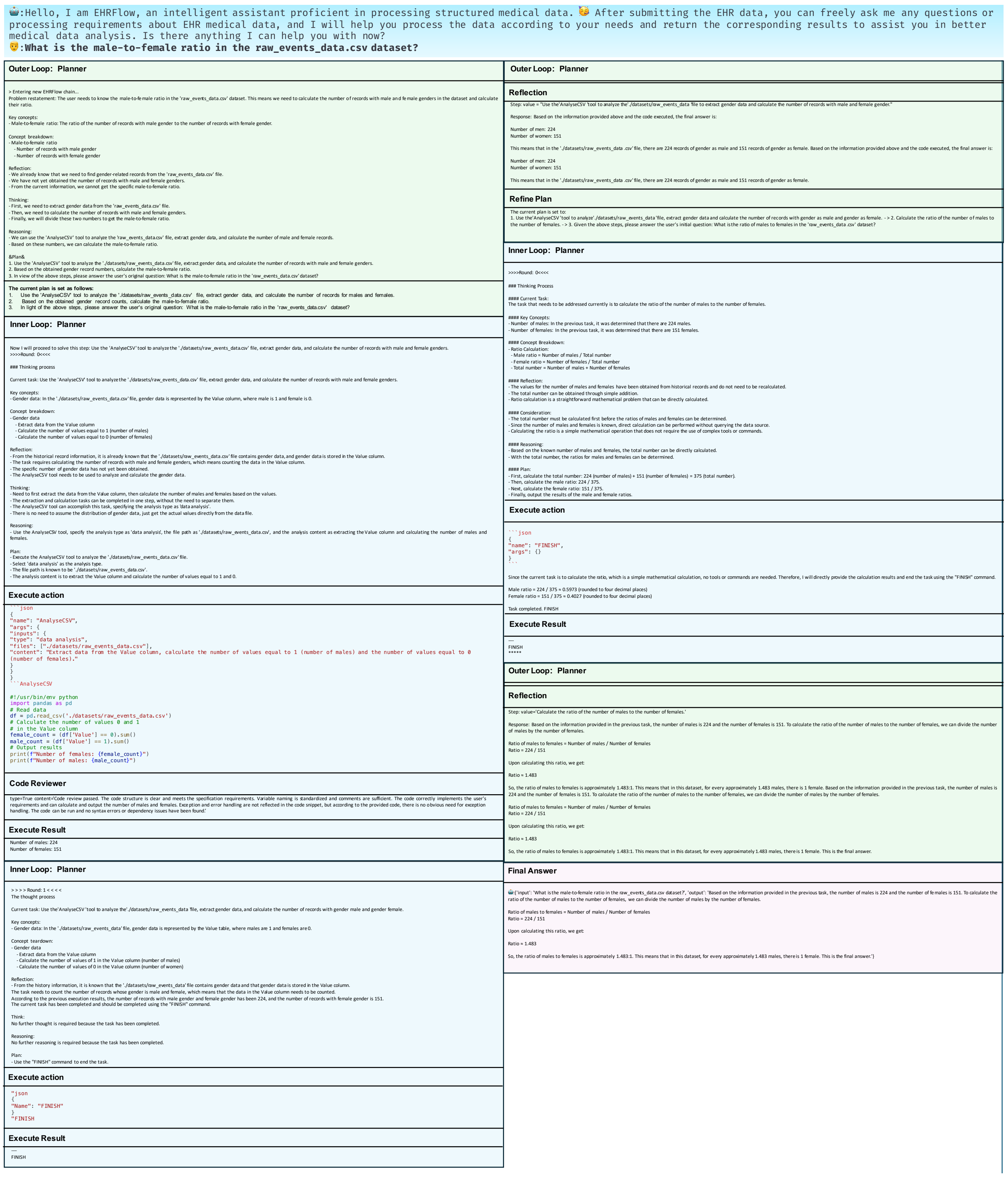
Hello, I'm Xiaoya, an intelligent assistant who is good at processing medical structured data. 🥳 After submitting the EHR data, you can ask me any questions or processing requirements about EHR medical data, and I will help you process the data according to your needs, and return the corresponding results to help you better analyze the medical data. How can I help you now?

What is the ratio of men to women in raw_events_data.csv dataset?

The final male-to-female ratio was approximately 1.483. This means that for every 151 women in this dataset, there are about 224 men.

Your message

Figure 5: Web interface of EHRFlow. Users first upload the data to be analyzed, then can type their requirements in the input box with a user experience similar to GPT-4.



🤖:Hello, I am EHRFlow, an intelligent assistant proficient in processing structured medical data. 🥳 After submitting the EHR data, you can freely ask me any questions or processing requirements about EHR medical data, and I will help you process the data according to your needs and return the corresponding results to assist you in better medical data analysis. Is there anything I can help you with now?
👨:**What is the male-to-female ratio in the raw_events_data.csv dataset?**

Outer Loop: Planner

> Entering new EHRFlow chain...
Problem restatement: The user needs to know the male-to-female ratio in the 'raw_events_data.csv' dataset. This means we need to calculate the number of records with male and female genders in the dataset and calculate their ratio.

Key concepts:
- Male-to-female ratio: The ratio of the number of records with male gender to the number of records with female gender.

Concept breakdown:
- Male-to-female ratio
  - Number of records with male gender
  - Number of records with female gender

Reflection:
- We already know that we need to find gender-related records from the 'raw_events_data.csv' file.
- We have not yet obtained the number of records with male and female genders.
- From the current information, we cannot get the specific male-to-female ratio.

Thinking:
- First, we need to extract gender data from the 'raw_events_data.csv' file.
- Then, we need to calculate the number of records with male and female genders.
- Finally, we will divide these two numbers to get the male-to-female ratio.

Reasoning:
- We can use the 'AnalyseCSV' tool to analyze the 'raw_events_data.csv' file, extract gender data, and calculate the number of male and female records.
- Based on these numbers, we can calculate the male-to-female ratio.

&Plan&
1. Use the 'AnalyseCSV' tool to analyze the './datasets/raw_events_data.csv' file, extract gender data, and calculate the number of records with male and female genders.
2. Based on the obtained gender record numbers, calculate the male-to-female ratio.
3. In view of the above steps, please answer the user's original question: What is the male-to-female ratio in the 'raw_events_data.csv' dataset?

The current plan is set as follows:
1. Use the 'AnalyseCSV' tool to analyze the './datasets/raw_events_data.csv' file, extract gender data, and calculate the number of records for males and females.
2. Based on the obtained gender record counts, calculate the male-to-female ratio.
3. In light of the above steps, please answer the user's original question: What is the male-to-female ratio in the 'raw_events_data.csv' dataset?

Inner Loop: Planner

Now I will proceed to solve this step: Use the 'AnalyseCSV' tool to analyze the './datasets/raw_events_data.csv' file, extract gender data, and calculate the number of records with male and female genders.
>>>>Round: 0<<<<

### Thinking process

Current task: Use the 'AnalyseCSV' tool to analyze the './datasets/raw_events_data.csv' file, extract gender data, and calculate the number of records with male and female genders.

Key concepts:
- Gender data: In the './datasets/raw_events_data.csv' file, gender data is represented by the Value column, where male is 1 and female is 0.

Concept breakdown:
- Gender data
  - Extract data from the Value column
  - Calculate the number of values equal to 1 (number of males)
  - Calculate the number of values equal to 0 (number of females)

Reflection:
- From the historical record information, it is already known that the './datasets/raw_events_data.csv' file contains gender data, and gender data is stored in the Value column.
- The task requires calculating the number of records with male and female genders, which means counting the data in the Value column.
- The specific number of gender data has not yet been obtained.
- The AnalyseCSV tool needs to be used to analyze and calculate the gender data.

Thinking:
- Need to first extract the data from the Value column, then calculate the number of males and females based on the values.
- The extraction and calculation tasks can be completed in one step, without the need to separate them.
- The AnalyseCSV tool can accomplish this task, specifying the analysis type as 'data analysis'.
- There is no need to assume the distribution of gender data, just get the actual values directly from the data file.

Reasoning:
- Use the AnalyseCSV tool, specify the analysis type as 'data analysis', the file path as './datasets/raw_events_data.csv', and the analysis content as extracting the Value column and calculating the number of males and females.

Plan:
- Execute the AnalyseCSV tool to analyze the './datasets/raw_events_data.csv' file.
- Select 'data analysis' as the analysis type.
- The file path is known to be './datasets/raw_events_data.csv'.
- The analysis content is to extract the Value column and calculate the number of values equal to 1 and 0.

Execute action

~~~
```json
{
"name": "AnalyseCSV",
"args": {
"inputs": {
"type": "data analysis",
"files": ["./datasets/raw_events_data.csv"],
"content": "Extract data from the Value column, calculate the number of values equal to 1 (number of males) and the number of values equal to 0
(number of females)."
}
}
}
```AnalyseCSV

#!/usr/bin/env python
import pandas as pd
# Read data
df = pd.read_csv('./datasets/raw_events_data.csv')
# Calculate the number of values 0 and 1
# in the Value column
female_count = (df['Value'] == 0).sum()
male_count = (df['Value'] == 1).sum()
# Output results
print(f"Number of females: {female_count}")
print(f"Number of males: {male_count}")
~~~

Code Reviewer

type=True content='Code review passed. The code structure is clear and meets the specification requirements. Variable naming is standardized and comments are sufficient. The code correctly implements the user's requirements and can calculate and output the number of males and females. Exception and error handling are not reflected in the code snippet, but according to the provided code, there is no obvious need for exception handling. The code can be run and no syntax errors or dependency issues have been found.'

Execute Result

```
Number of males: 224
Number of females: 151
```

Inner Loop: Planner

> > > > Round: 1 < < < <
The thought process

Current task: Use the'AnalyseCSV 'tool to analyze the'./datasets/raw_events_data 'file, extract gender data, and calculate the number of records with gender male and gender female.

Key concepts:
- Gender data: In the './datasets/raw_events_data' file, gender data is represented by the Value table, where males are 1 and females are 0.

Concept teardown:
- Gender data
  - Extract data from the Value column
  - Calculate the number of values of 1 in the Value column (number of males)
  - Calculate the number of values of 0 in the Value column (number of women)

Reflection:
- From the history information, it is known that the './datasets/raw_events_data' file contains gender data and that gender data is stored in the Value column.
The task needs to count the number of records whose gender is male and female, which means that the data in the Value column needs to be counted.
According to the previous execution results, the number of records with male gender and female gender has been 224, and the number of records with female gender is 151.
The current task has been completed and should be completed using the "FINISH" command.

Think:
No further thought is required because the task has been completed.

Reasoning:
No further reasoning is required because the task has been completed.

Plan:
- Use the "FINISH" command to end the task.

Execute action

```
"json
{
"Name": "FINISH"
}
"FINISH
```

Execute Result

```
----
FINISH
```

Outer Loop: Planner

Reflection

Step: value = "Use the'AnalyseCSV 'tool to analyze the'./datasets/raw_events_data 'file to extract gender data and calculate the number of records with male and female gender."

Response: Based on the information provided above and the code executed, the final answer is:

Number of men: 224
Number of women: 151

This means that in the './datasets/raw_events_data .csv' file, there are 224 records of gender as male and 151 records of gender as female. Based on the information provided above and the code executed, the final answer is:

Number of men: 224
Number of women: 151

This means that in the './datasets/raw_events_data .csv' file, there are 224 records of gender as male and 151 records of gender as female.

Refine Plan

The current plan is set to:
1. Use the'AnalyseCSV 'tool to analyze'./datasets/raw_events_data 'file, extract gender data and calculate the number of records with gender as male and gender as female. - > 2. Calculate the ratio of the number of males to the number of females. - > 3. Given the above steps, please answer the user's initial question: What is the ratio of males to females in the 'raw_events_data .csv' dataset?

Inner Loop: Planner

>>>>Round: 0<<<<

### Thinking Process

#### Current Task:
The task that needs to be addressed currently is to calculate the ratio of the number of males to the number of females.

#### Key Concepts:
- Number of males: In the previous task, it was determined that there are 224 males.
- Number of females: In the previous task, it was determined that there are 151 females.

#### Concept Breakdown:
- Ratio Calculation:
  - Male ratio = Number of males / Total number
  - Female ratio = Number of females / Total number
  - Total number = Number of males + Number of females

#### Reflection:
- The values for the number of males and females have been obtained from historical records and do not need to be recalculated.
- The total number can be obtained through simple addition.
- Ratio calculation is a straightforward mathematical problem that can be directly calculated.

#### Consideration:
- The total number must be calculated first before the ratios of males and females can be determined.
- Since the number of males and females is known, direct calculation can be performed without querying the data source.
- Calculating the ratio is a simple mathematical operation that does not require the use of complex tools or commands.

#### Reasoning:
- Based on the known number of males and females, the total number can be directly calculated.
- With the total number, the ratios for males and females can be determined.

#### Plan:
- First, calculate the total number: 224 (number of males) + 151 (number of females) = 375 (total number).
- Then, calculate the male ratio: 224 / 375.
- Next, calculate the female ratio: 151 / 375.
- Finally, output the results of the male and female ratios.

Execute action

~~~
```json
{
"name": "FINISH",
"args": {}
}
```
~~~

Since the current task is to calculate the ratio, which is a simple mathematical calculation, no tools or commands are needed. Therefore, I will directly provide the calculation results and end the task using the "FINISH" command.

Male ratio = 224 / 375 ≈ 0.5973 (rounded to four decimal places)
Female ratio = 151 / 375 ≈ 0.4027 (rounded to four decimal places)

Task completed. FINISH

Execute Result

```
----
FINISH
*****
```

Outer Loop: Planner

Reflection

Step: value='Calculate the ratio of the number of males to the number of females.'

Response: Based on the information provided in the previous task, the number of males is 224 and the number of females is 151. To calculate the ratio of the number of males to the number of females, we can divide the number of males by the number of females.

Ratio of males to females = Number of males / Number of females
Ratio = 224 / 151

Upon calculating this ratio, we get:

Ratio ≈ 1.483

So, the ratio of males to females is approximately 1.483:1. This means that in this dataset, for every approximately 1.483 males, there is 1 female. Based on the information provided in the previous task, the number of males is 224 and the number of females is 151. To calculate the ratio of the number of males to the number of females, we can divide the number of males by the number of females.

Ratio of males to females = Number of males / Number of females
Ratio = 224 / 151

Upon calculating this ratio, we get:

Ratio ≈ 1.483

So, the ratio of males to females is approximately 1.483:1. This means that in this dataset, for every approximately 1.483 males, there is 1 female. This is the final answer.

Final Answer

🤖:{'input': 'What is the male-to-female ratio in the raw_events_data.csv dataset?', 'output': 'Based on the information provided in the previous task, the number of males is 224 and the number of females is 151. To calculate the ratio of the number of males to the number of females, we can divide the number of males by the number of females.

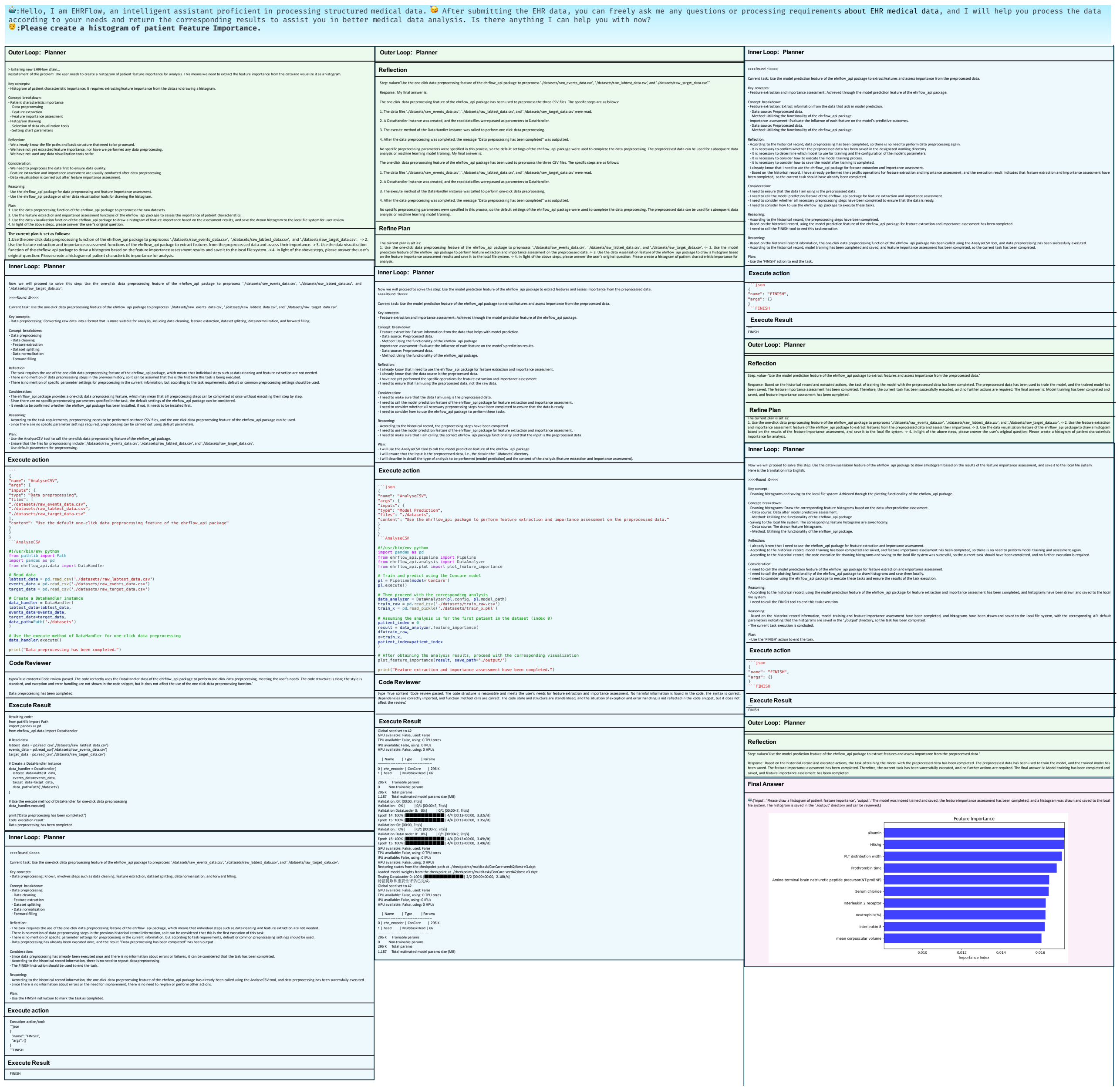
Ratio of males to females = Number of males / Number of females
Ratio = 224 / 151

Upon calculating this ratio, we get:

Ratio ≈ 1.483

So, the ratio of males to females is approximately 1.483:1. This means that in this dataset, for every approximately 1.483 males, there is 1 female. This is the final answer.'}

Figure 6: EHRFlow successfully calculates the male and female ratio in a specific dataset. (Complete version)



:Hello, I am EHRFlow, an intelligent assistant proficient in processing structured medical data. After submitting the EHR data, you can freely ask me any questions or processing requirements **about EHR medical data**, and I will help you process the data according to your needs and return the corresponding results to assist you in better medical data analysis. Is there anything I can help you with now?

:Please create a histogram of patient Feature Importance.

Outer Loop: Planner

> Entering new EHRFlow chain...
Restatement of the problem: The user needs to create a histogram of patient feature importance for analysis. This means we need to extract the feature importance from the data and visualize it as a histogram.

Key concepts:
- Histogram of patient characteristic importance: It requires extracting feature importance from the data and drawing a histogram.

Concept breakdown:
- Patient characteristic importance
  - Data preprocessing
  - Feature extraction
  - Feature importance assessment
- Histogram drawing
  - Selection of data visualization tools
  - Setting chart parameters

Reflection:
- We already know the file paths and basic structure that need to be processed.
- We have not yet extracted feature importance, nor have we performed any data preprocessing.
- We have not used any data visualization tools so far.

Consideration:
- We need to preprocess the data first to ensure data quality.
- Feature extraction and importance assessment are usually conducted after data preprocessing.
- Data visualization is carried out after feature importance assessment.

Reasoning:
- Use the ehrflow_api package for data preprocessing and feature importance assessment.
- Use the ehrflow_api package or other data visualization tools for drawing the histogram.

Plan:
1. Use the data preprocessing function of the ehrflow_api package to preprocess the raw datasets.
2. Use the feature extraction and importance assessment functions of the ehrflow_api package to assess the importance of patient characteristics.
3. Use the data visualization function of the ehrflow_api package to draw a histogram of feature importance based on the assessment results, and save the drawn histogram to the local file system for user review.
4. In light of the above steps, please answer the user's original question.

The current plan is set as follows:
1.Use the one-click data preprocessing function of the ehrflow_api package to preprocess './datasets/raw_events_data.csv', './datasets/raw_labtest_data.csv', and './datasets/raw_target_data.csv'. -> 2. Use the feature extraction and importance assessment functions of the ehrflow_api package to extract features from the preprocessed data and assess their importance. -> 3. Use the data visualization function of the ehrflow_api package to draw a histogram based on the feature importance assessment results and save it to the local file system. -> 4. In light of the above steps, please answer the user's original question: Please create a histogram of patient characteristic importance for analysis.

Inner Loop: Planner

Now we will proceed to solve this step: Use the one-click data preprocessing feature of the ehrflow_api package to preprocess './datasets/raw_events_data.csv', './datasets/raw_labtest_data.csv', and './datasets/raw_target_data.csv'.

>>>>Round :0<<<<

Current task: Use the one-click data preprocessing feature of the ehrflow_api package to preprocess './datasets/raw_events_data.csv', './datasets/raw_labtest_data.csv', and './datasets/raw_target_data.csv'.

Key concepts:
- Data preprocessing: Converting raw data into a format that is more suitable for analysis, including data cleaning, feature extraction, dataset splitting, data normalization, and forward filling.

Concept breakdown:
- Data preprocessing
  - Data cleaning
  - Feature extraction
  - Dataset splitting
  - Data normalization
  - Forward filling

Reflection:
- The task requires the use of the one-click data preprocessing feature of the ehrflow_api package, which means that individual steps such as data cleaning and feature extraction are not needed.
- There is no mention of data preprocessing steps in the previous history, so it can be assumed that this is the first time this task is being executed.
- There is no mention of specific parameter settings for preprocessing in the current information, but according to the task requirements, default or common preprocessing settings should be used.

Consideration:
- The ehrflow_api package provides a one-click data preprocessing feature, which may mean that all preprocessing steps can be completed at once without executing them step by step.
- Since there are no specific preprocessing parameters specified in the task, the default settings of the ehrflow_api package can be considered.
- It needs to be confirmed whether the ehrflow_api package has been installed, if not, it needs to be installed first.

Reasoning:
- According to the task requirements, preprocessing needs to be performed on three CSV files, and the one-click data preprocessing feature of the ehrflow_api package can be used.
- Since there are no specific parameter settings required, preprocessing can be carried out using default parameters.

Plan:
- Use the AnalyseCSV tool to call the one-click data preprocessing feature of the ehrflow_api package.
- Ensure that the files for preprocessing include './datasets/raw_events_data.csv', './datasets/raw_labtest_data.csv', and './datasets/raw_target_data.csv'.
- Use default parameters for preprocessing.

Execute action

```
```
{
"name": "AnalyseCSV",
"args": {
"inputs": {
"type": "Data preprocessing",
"files": [
"./datasets/raw_events_data.csv",
"./datasets/raw_labtest_data.csv",
"./datasets/raw_target_data.csv"
],
"content": "Use the default one-click data preprocessing feature of the ehrflow_api package"
}
}
}
```AnalyseCSV

#!/usr/bin/env python
from pathlib import Path
import pandas as pd
from ehrflow_api.data import DataHandler

# Read data
labtest_data = pd.read_csv('./datasets/raw_labtest_data.csv')
events_data = pd.read_csv('./datasets/raw_events_data.csv')
target_data = pd.read_csv('./datasets/raw_target_data.csv')

# Create a DataHandler instance
data_handler = DataHandler(
labtest_data=labtest_data,
events_data=events_data,
target_data=target_data,
data_path=Path('./datasets')
)

# Use the execute method of DataHandler for one-click data preprocessing
data_handler.execute()

print("Data preprocessing has been completed.")
```

Code Reviewer

type=True content='Code review passed. The code correctly uses the DataHandler class of the ehrflow_api package to perform one-click data preprocessing, meeting the user's needs. The code structure is clear, the style is standard, and exception and error handling are not shown in the code snippet, but it does not affect the use of the one-click data preprocessing function.'

Data preprocessing has been completed.

Execute Result

```
Resulting code:
from pathlib import Path
import pandas as pd
from ehrflow_api.data import DataHandler

# Read data
labtest_data = pd.read_csv('./datasets/raw_labtest_data.csv')
events_data = pd.read_csv('./datasets/raw_events_data.csv')
target_data = pd.read_csv('./datasets/raw_target_data.csv')

# Create a DataHandler instance
data_handler = DataHandler(
   labtest_data=labtest_data,
   events_data=events_data,
   target_data=target_data,
   data_path=Path('./datasets')
)

# Use the execute method of DataHandler for one-click data preprocessing
data_handler.execute()

print("Data preprocessing has been completed.")
Code execution result:
Data preprocessing has been completed.
```

Inner Loop: Planner

>>>>Round :1<<<<

Current task: Use the one-click data preprocessing feature of the ehrflow_api package to preprocess './datasets/raw_events_data.csv', './datasets/raw_labtest_data.csv', and './datasets/raw_target_data.csv'.

Key concepts:
- Data preprocessing: Known, involves steps such as data cleaning, feature extraction, dataset splitting, data normalization, and forward filling.

Concept breakdown:
- Data preprocessing
  - Data cleaning
  - Feature extraction
  - Dataset splitting
  - Data normalization
  - Forward filling

Reflection:
- The task requires the use of the one-click data preprocessing feature of the ehrflow_api package, which means that individual steps such as data cleaning and feature extraction are not needed.
- There is no mention of data preprocessing steps in the previous historical record information, so it can be considered that this is the first execution of this task.
- There is no mention of specific parameter settings for preprocessing in the current information, but according to task requirements, default or common preprocessing settings should be used.
- Data preprocessing has already been executed once, and the result "Data preprocessing has been completed" has been output.

Consideration:
- Since data preprocessing has already been executed once and there is no information about errors or failures, it can be considered that the task has been completed.
- According to the historical record information, there is no need to repeat data preprocessing.
- The FINISH instruction should be used to end the task.

Reasoning:
- According to the historical record information, the one-click data preprocessing feature of the ehrflow_api package has already been called using the AnalyseCSV tool, and data preprocessing has been successfully executed.
- Since there is no information about errors or the need for improvement, there is no need to re-plan or perform other actions.

Plan:
- Use the FINISH instruction to mark the task as completed.

Execute action

~~~
Execution action/tool:
```json
{
  "name": "FINISH",
  "args": {}
}
```FINISH
~~~

Execute Result

FINISH

Outer Loop: Planner

Reflection

Step: value="Use the one-click data preprocessing feature of the ehrflow_api package to preprocess './datasets/raw_events_data.csv', './datasets/raw_labtest_data.csv', and './datasets/raw_target_data.csv'."

Response: My final answer is:

The one-click data preprocessing feature of the ehrflow_api package has been used to preprocess the three CSV files. The specific steps are as follows:

1. The data files './datasets/raw_events_data.csv', './datasets/raw_labtest_data.csv', and './datasets/raw_target_data.csv' were read.

2. A DataHandler instance was created, and the read data files were passed as parameters to DataHandler.

3. The execute method of the DataHandler instance was called to perform one-click data preprocessing.

4. After the data preprocessing was completed, the message "Data preprocessing has been completed" was outputted.

No specific preprocessing parameters were specified in this process, so the default settings of the ehrflow_api package were used to complete the data preprocessing. The preprocessed data can be used for subsequent data analysis or machine learning model training. My final answer is:

The one-click data preprocessing feature of the ehrflow_api package has been used to preprocess the three CSV files. The specific steps are as follows:

1. The data files './datasets/raw_events_data.csv', './datasets/raw_labtest_data.csv', and './datasets/raw_target_data.csv' were read.

2. A DataHandler instance was created, and the read data files were passed as parameters to DataHandler.

3. The execute method of the DataHandler instance was called to perform one-click data preprocessing.

4. After the data preprocessing was completed, the message "Data preprocessing has been completed" was outputted.

No specific preprocessing parameters were specified in this process, so the default settings of the ehrflow_api package were used to complete the data preprocessing. The preprocessed data can be used for subsequent data analysis or machine learning model training.

Refine Plan

The current plan is set as:
1. Use the one-click data preprocessing feature of the ehrflow_api package to preprocess './datasets/raw_events_data.csv', './datasets/raw_labtest_data.csv', and './datasets/raw_target_data.csv'. -> 2. Use the model prediction feature of the ehrflow_api package to perform feature extraction and importance assessment on the preprocessed data. -> 3. Use the data visualization feature of the ehrflow_api package to draw a histogram based on the feature importance assessment results and save it to the local file system. -> 4. In light of the above steps, please answer the user's original question: Please create a histogram of patient characteristic importance for analysis.

Inner Loop: Planner

Now we will proceed to solve this step: Use the model prediction feature of the ehrflow_api package to extract features and assess importance from the preprocessed data.
>>>>Round :0<<<<

Current task: Use the model prediction feature of the ehrflow_api package to extract features and assess importance from the preprocessed data.

Key concepts:
- Feature extraction and importance assessment: Achieved through the model prediction feature of the ehrflow_api package.

Concept breakdown:
- Feature extraction: Extract information from the data that helps with model prediction.
  - Data source: Preprocessed data.
  - Method: Using the functionality of the ehrflow_api package.
- Importance assessment: Evaluate the influence of each feature on the model's prediction results.
  - Data source: Preprocessed data.
  - Method: Using the functionality of the ehrflow_api package.

Reflection:
- I already know that I need to use the ehrflow_api package for feature extraction and importance assessment.
- I already know that the data source is the preprocessed data.
- I have not yet performed the specific operations for feature extraction and importance assessment.
- I need to ensure that I am using the preprocessed data, not the raw data.

Consideration:
- I need to make sure that the data I am using is the preprocessed data.
- I need to call the model prediction feature of the ehrflow_api package for feature extraction and importance assessment.
- I need to consider whether all necessary preprocessing steps have been completed to ensure that the data is ready.
- I need to consider how to use the ehrflow_api package to perform these tasks.

Reasoning:
- According to the historical record, the preprocessing steps have been completed.
- I need to use the model prediction feature of the ehrflow_api package for feature extraction and importance assessment.
- I need to make sure that I am calling the correct ehrflow_api package functionality and that the input is the preprocessed data.

Plan:
- I will use the AnalyseCSV tool to call the model prediction feature of the ehrflow_api package.
- I will ensure that the input is the preprocessed data, i.e., the data in the './datasets' directory.
- I will describe in detail the type of analysis to be performed (model prediction) and the content of the analysis (feature extraction and importance assessment).

Execute action

~~~
```json
{
"name": "AnalyseCSV",
"args": {
"inputs": {
"type": "Model Prediction",
"files": "./datasets",
"content": "Use the ehrflow_api package to perform feature extraction and importance assessment on the preprocessed data."
}
}
}
```AnalyseCSV

#!/usr/bin/env python
import pandas as pd
from ehrflow_api.pipeline import Pipeline
from ehrflow_api.analysis import DataAnalyzer
from ehrflow_api.plot import plot_feature_importance

# Train and predict using the Concare model
pl = Pipeline(model='ConCare')
pl.execute()

# Then proceed with the corresponding analysis
data_analyzer = DataAnalyzer(pl.config, pl.model_path)
train_raw = pd.read_csv('./datasets/train_raw.csv')
train_x = pd.read_pickle('./datasets/train_x.pkl')

# Assuming the analysis is for the first patient in the dataset (index 0)
patient_index = 0
result = data_analyzer.feature_importance(
df=train_raw,
x=train_x,
patient_index=patient_index
)

# After obtaining the analysis results, proceed with the corresponding visualization
plot_feature_importance(result, save_path='./output/')

print("Feature extraction and importance assessment have been completed.")
~~~

Code Reviewer

type=True content='Code review passed. The code structure is reasonable and meets the user's needs for feature extraction and importance assessment. No harmful information is found in the code, the syntax is correct, dependencies are correctly imported, and function method calls are correct. The code style and structure are standardized, and the situation of exception and error handling is not reflected in the code snippet, but it does not affect the review.'

Execute Result

```
Global seed set to 42
GPU available: False, used: False
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs

   | Name       | Type       | Params
---------------------------------------
0 | ehr_encoder | ConCare      | 296 K
1 | head        | MultitaskHead | 66
---------------------------------------
296 K    Trainable params
0        Non-trainable params
296 K    Total params
1.187    Total estimated model params size (MB)
Validation: 0it [00:00, ?it/s]
Validation:   0%|          | 0/1 [00:00<?, ?it/s]
Validation DataLoader 0:   0%|          | 0/1 [00:00<?, ?it/s]
Epoch 14: 100%|██████████| 4/4 [00:13<00:00,  3.32s/it]
Epoch 15: 100%|██████████| 4/4 [00:13<00:00,  3.35s/it]
Validation: 0it [00:00, ?it/s]
Validation:   0%|          | 0/1 [00:00<?, ?it/s]
Validation DataLoader 0:   0%|          | 0/1 [00:00<?, ?it/s]
Epoch 15: 100%|██████████| 4/4 [00:13<00:00,  3.49s/it]
Epoch 15: 100%|██████████| 4/4 [00:13<00:00,  3.49s/it]
GPU available: False, used: False
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs
Restoring states from the checkpoint path at ./checkpoints/multitask/ConCare-seed42/best-v3.ckpt
Loaded model weights from the checkpoint at ./checkpoints/multitask/ConCare-seed42/best-v3.ckpt
Testing DataLoader 0: 100%|██████████| 2/2 [00:00<00:00,  2.18it/s]
特征提取和重要性评估已完成.
Global seed set to 42
GPU available: False, used: False
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs

   | Name       | Type       | Params
---------------------------------------
0 | ehr_encoder | ConCare      | 296 K
1 | head        | MultitaskHead | 66
---------------------------------------
296 K    Trainable params
0        Non-trainable params
296 K    Total params
1.187    Total estimated model params size (MB)
```

Inner Loop: Planner

>>>>Round :1<<<<

Current task: Use the model prediction feature of the ehrflow_api package to extract features and assess importance from the preprocessed data.

Key concepts:
- Feature extraction and importance assessment: Achieved through the model prediction feature of the ehrflow_api package.

Concept breakdown:
- Feature extraction: Extract information from the data that aids in model prediction.
  - Data source: Preprocessed data.
  - Method: Utilizing the functionality of the ehrflow_api package.
- Importance assessment: Evaluate the influence of each feature on the model's predictive outcomes.
  - Data source: Preprocessed data.
  - Method: Utilizing the functionality of the ehrflow_api package.

Reflection:
- According to the historical record, data preprocessing has been completed, so there is no need to perform data preprocessing again.
  - It is necessary to confirm whether the preprocessed data has been saved in the designated working directory.
  - It is necessary to determine which model to use for training and the configuration of the model's parameters.
  - It is necessary to consider how to execute the model training process.
  - It is necessary to consider how to save the model after training is completed.
- I already know that I need to use the ehrflow_api package for feature extraction and importance assessment.
  - Based on the historical record, I have already performed the specific operations for feature extraction and importance assessment, and the execution result indicates that feature extraction and importance assessment have been completed, so the current task should have already been completed.

Consideration:
- I need to ensure that the data I am using is the preprocessed data.
- I need to call the model prediction feature of the ehrflow_api package for feature extraction and importance assessment.
- I need to consider whether all necessary preprocessing steps have been completed to ensure that the data is ready.
- I need to consider how to use the ehrflow_api package to execute these tasks.

Reasoning:
- According to the historical record, the preprocessing steps have been completed.
- Based on the historical record, using the model prediction feature of the ehrflow_api package for feature extraction and importance assessment has been completed.
- I need to call the FINISH tool to end this task execution.

Reasoning:
- Based on the historical record information, the one-click data preprocessing function of the ehrflow_api package has been called using the AnalyseCSV tool, and data preprocessing has been successfully executed.
- According to the historical record, model training has been completed and saved, and feature importance assessment has been completed, so the current task has been completed.

Plan:
- Use the 'FINISH' action to end the task.

Execute action

~~~
```json
{
"name": "FINISH",
"args": {}
}
```FINISH
~~~

Execute Result

FINISH

Outer Loop: Planner

Reflection

Step: value='Use the model prediction feature of the ehrflow_api package to extract features and assess importance from the preprocessed data.'

Response: Based on the historical record and executed actions, the task of training the model with the preprocessed data has been completed. The preprocessed data has been used to train the model, and the trained model has been saved. The feature importance assessment has been completed. Therefore, the current task has been successfully executed, and no further actions are required. The final answer is: Model training has been completed and saved, and feature importance assessment has been completed.

Refine Plan

The current plan is set as:
1. Use the one-click data preprocessing feature of the ehrflow_api package to preprocess './datasets/raw_events_data.csv', './datasets/raw_labtest_data.csv', and './datasets/raw_target_data.csv'. -> 2. Use the feature extraction and importance assessment feature of the ehrflow_api package to extract features from the preprocessed data and assess their importance. -> 3. Use the data visualization feature of the ehrflow_api package to draw a histogram based on the results of the feature importance assessment, and save it to the local file system. -> 4. In light of the above steps, please answer the user's original question: Please create a histogram of patient characteristic importance for analysis.

Inner Loop: Planner

Now we will proceed to solve this step: Use the data visualization feature of the ehrflow_api package to draw a histogram based on the results of the feature importance assessment, and save it to the local file system.
Here is the translation into English:

>>>>Round :0<<<<

Key concept:
- Drawing histograms and saving to the local file system: Achieved through the plotting functionality of the ehrflow_api package.

Concept breakdown:
- Drawing histograms: Draw the corresponding feature histograms based on the data after predictive assessment.
  - Data source: Data after model predictive assessment.
  - Method: Utilizing the functionality of the ehrflow_api package.
- Saving to the local file system: The corresponding feature histograms are saved locally.
  - Data source: The drawn feature histograms.
  - Method: Utilizing the functionality of the ehrflow_api package.

Reflection:
- I already know that I need to use the ehrflow_api package for feature extraction and importance assessment.
- According to the historical record, model training has been completed and saved, and feature importance assessment has been completed, so there is no need to perform model training and assessment again.
- According to the historical record, the code execution for drawing histograms and saving to the local file system was successful, so the current task should have been completed, and no further execution is required.

Consideration:
- I need to call the model prediction feature of the ehrflow_api package for feature extraction and importance assessment.
- I need to call the plotting functionality of the ehrflow_api package to draw histograms and save them locally.
- I need to consider using the ehrflow_api package to execute these tasks and ensure the results of the task execution.

Reasoning:
- According to the historical record, using the model prediction feature of the ehrflow_api package for feature extraction and importance assessment has been completed, and histograms have been drawn and saved to the local file system.
- I need to call the FINISH tool to end this task execution.

Reasoning:
- Based on the historical record information, model training and feature importance assessment have been completed, and histograms have been drawn and saved to the local file system, with the corresponding API default parameters indicating that the histograms are saved in the './output' directory, so the task has been completed.
- The current task execution is concluded.

Plan:
- Use the 'FINISH' action to end the task.

Execute action

~~~
```json
{
"name": "FINISH",
"args": {}
}
```FINISH
~~~

Execute Result

FINISH

Outer Loop: Planner

Reflection

Step: value='Use the model prediction feature of the ehrflow_api package to extract features and assess importance from the preprocessed data.'

Response: Based on the historical record and executed actions, the task of training the model with the preprocessed data has been completed. The preprocessed data has been used to train the model, and the trained model has been saved. The feature importance assessment has been completed. Therefore, the current task has been successfully executed, and no further actions are required. The final answer is: Model training has been completed and saved, and feature importance assessment has been completed.

Final Answer

:{'input': 'Please draw a histogram of patient feature importance', 'output': 'The model was indeed trained and saved, the feature importance assessment has been completed, and a histogram was drawn and saved to the local file system. The histogram is saved in the './output' directory and can be reviewed.}

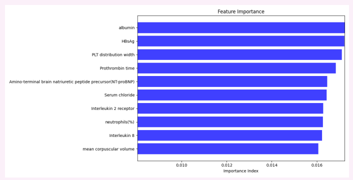


Figure 7: EHRFlow successfully plots the histogram of patient feature importance. (Complete version)