Evaluating Prompt Strategies on LLM-Based Assessment Performance of AI-Generated Clinical Support Information in Critical Care

Additional File 1

Content

Supplementary Table S1. Complete IMPACT rubric (domains and subitems).

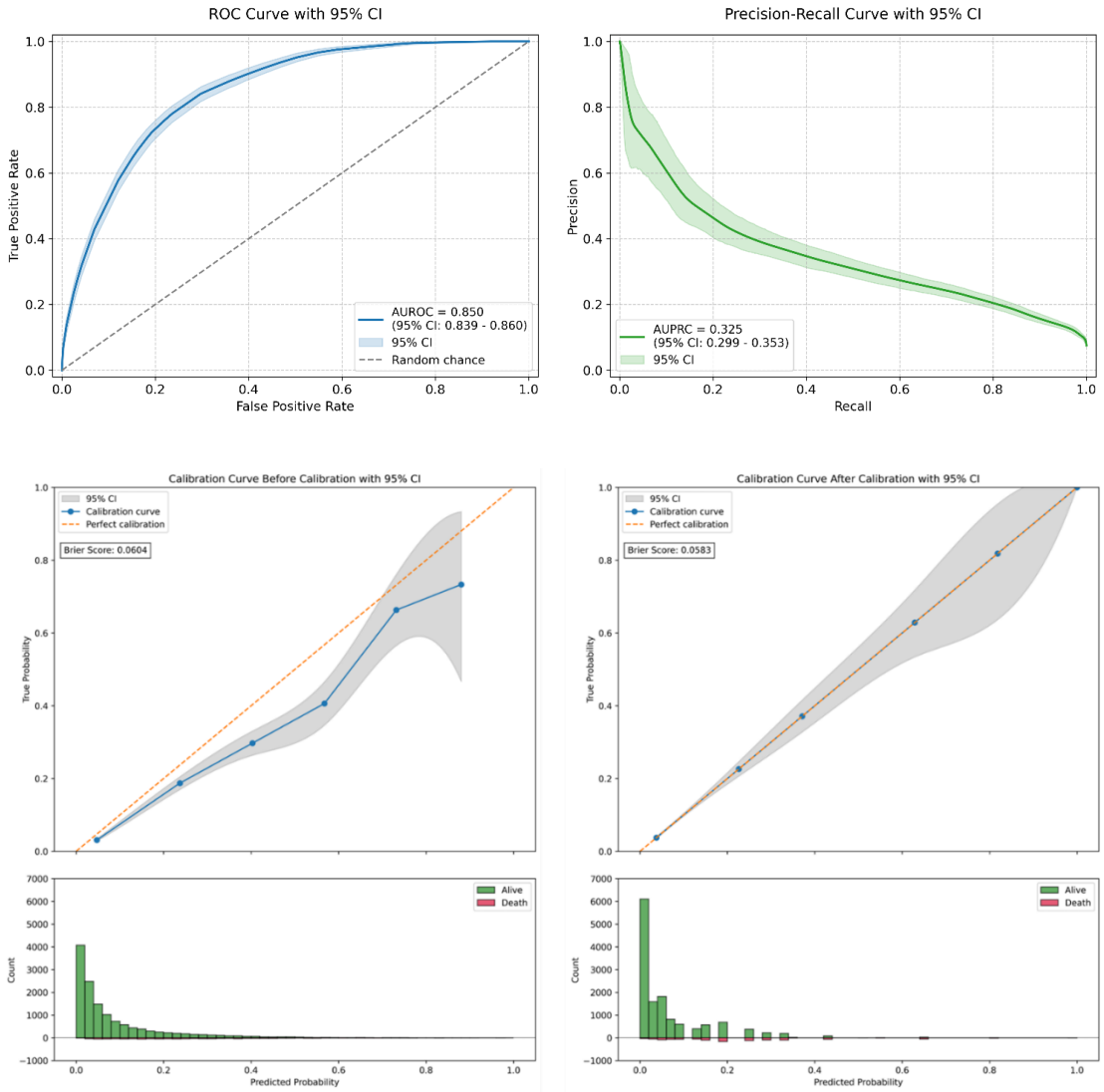
Domains	Subitems	Description
Integration	Clear Goals	Evaluate whether GAI explicitly identifies clear clinical goals or objectives of care
	Clinical Relevance	Assess if GAI response is patient-specific, context-aware, and not merely generic
	Evidence-Based Embedding	Check if GAI incorporates evidence-based knowledge, guidelines, and standard clinical protocols
	Clarity and Consistence	Appraise response quality for clarity, logical organization, and internal consistency
Mastery	Correct Clinical Reasoning	Assess pathophysiological reasoning, clinical priorities, and medically valid cause-and-effect conclusions
	Transparency of Uncertainty	Determine if GAI acknowledges uncertainty, limitations, and knowledge gaps, avoiding false certainty
	Adherence to Ethical Standards	Verify that GAI upholds patient autonomy, rights, safety, fairness, and professional conduct
Precision	Accuracy of Content	Verify whether the GAI provides factually accurate information consistent with established clinical evidence
	Up-to-Date Knowledge	Check whether the GAI uses current medical knowledge aligned with recent evidence and guidelines
	Specificity of Recommendations	Assess whether GAI delivers patient-specific recommendations with actionable steps and clinical parameters
	Bias Assessment	Evaluate whether the GAI identifies or avoids bias, unsupported assumptions, or imbalanced clinical perspectives
Applicability	Actionable Implementation	Determine whether GAI provides clear, practical clinical steps that can be immediately implemented
	Achievable Feasibility	Assess whether recommendations are realistically achievable given patient condition and available resources
	Appropriate Setting	Evaluate whether recommendations match ICU monitoring level, care intensity, and operational context
Comprehensiveness	Full Scenario Scope	Determine if GAI anticipates deterioration risks and addresses the full clinical scenario comprehensively
	Multidomain Coverage	Evaluate whether GAI covers relevant physiological systems, diagnostics, and management domains appropriately
	Benefits and Harms	Review how GAI balances therapeutic benefits with risks, complications, and potential patient harms
	Patient-Centered Care	Verify if GAI incorporates patient values, goals of care, and shared decisions appropriately
Timeliness	Urgency-Based Triage	Assess whether GAI recognizes clinical urgency and identifies conditions requiring immediate attention
	Priority Sequencing	Evaluate whether GAI sequences actions by clinical importance, risk level, and therapeutic priority
	Timing and Intervals	Examine whether GAI adjusts timing and reassessment intervals based on evolving patient physiology

Supplementary Table S2. Compute and inference details of o3-mini evaluation by generation prompt

Prompt	Description	Eval prompt tokens	Eval completion tokens	Eval total tokens	Time (sec)
P1	Zero-shot baseline	8,650	7,150	15,800	61
P2	Expanded structured instruction	9,450	7,100	16,550	62
P3	Conversational	13,100	7,100	19,650	55

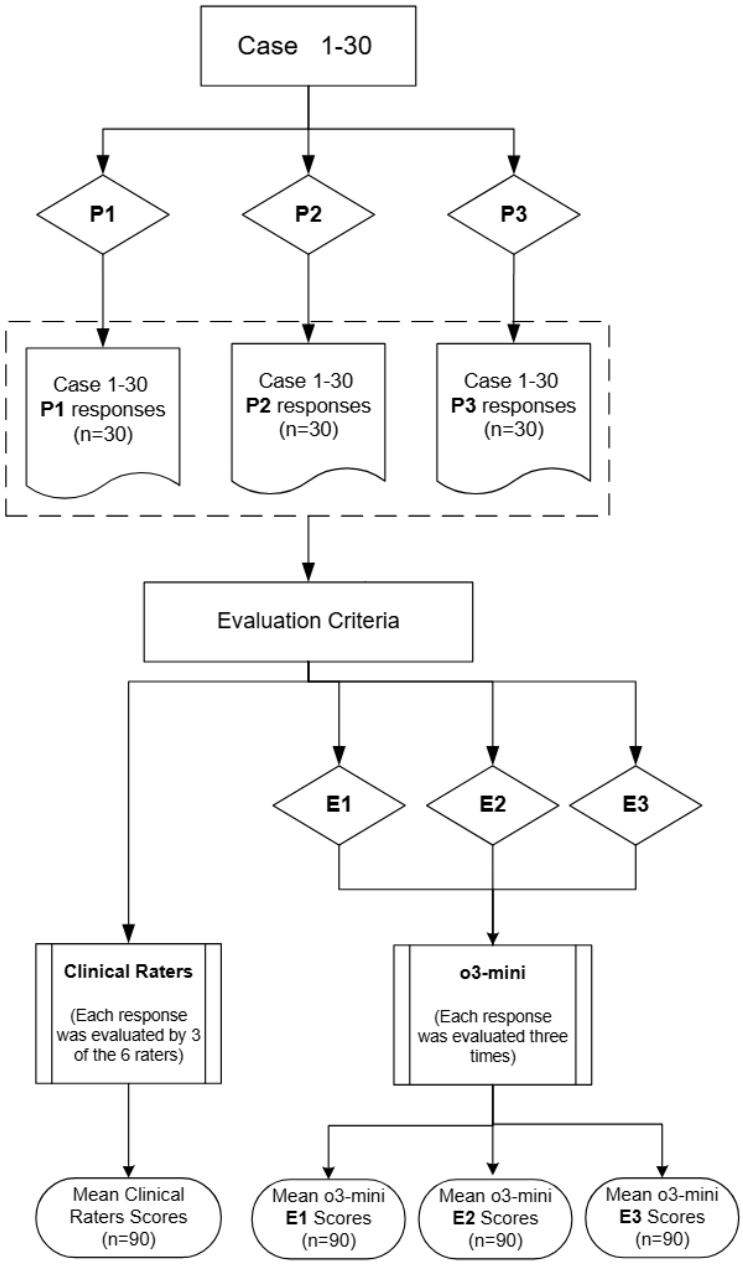
Token counts and times shown are per single o3-mini evaluation of one response used E1 with P1, P2, and P3. Data from the parent study. Total cost across 270 evaluation cycles (30 cases × 3 evaluation strategies × 3 repetitions) was approximately $34 USD, based on Azure OpenAI pricing ($1.10 per 1M input tokens; $4.40 per 1M output tokens).

Supplementary Figure S1. XGBoost model performance and calibration curve for the validation dataset



AUPRC, area under the precision-recall curve; AUROC, area under the receiver operating characteristic curve; CI, confident interval; XGBoost, Extreme Gradient Boosting

Supplementary Figure S2. Flowchart of AI-gerenatied eesponses and evaluation.



The 30 cases were sampled from the validation dataset of the parent study using stratified random sampling across three risk groups. Each case was processed by GPT-4o using three generation prompts (P1, P2, P3), producing 90 AI-generated clinical support responses. Each response was evaluated by three clinical raters and by the o3-mini automated evaluator under three evaluation strategies (E1, E2, E3). Abbreviations: P1, zero-shot baseline prompt; P2, expanded structured instruction prompt; P3, five-turn conversational prompt; E1, standard evaluation prompt; E2, top-down decremental evaluation prompt; E3, bottom-up incremental evaluation prompt.

1. TRIPOD-LLM reporting compliance checklist

Research Design

Under the TRIPOD-LLM modular framework (Gallifant et al., Nat Med 2025), the research design selection determines which items are required. This study falls under:

Research design	LLM methods (M): methodological investigation of prompt-engineering techniques for LLM evaluators.

Scope of Required Items

TRIPOD-LLM contains 19 main items and 50 subitems in total. Of these, 14 main items and 32 subitems apply to all research designs and tasks; the remainder are required only for specific designs or tasks. Under the LLM methods (M) design, 41 items are required and are listed in the checklist below.

The rationale for selecting M as the sole design is that the study evaluates how different prompt architectures affect the behavior of an LLM-based evaluator. The workflow itself is not a pure classification, question answering, or summarization task, so no single LLM task category adequately captures it; task-specific items are therefore omitted.

Items not required under the M design are not listed. These include items required only for LLM evaluation (E) or LLM evaluation in healthcare settings (H) designs (items 2b, 2l, 3b, 7b, 7c, 14c, 14d, 15, 16a-16d, and 19c-19f) and items that depend on a specific LLM task category (items 6e, 7a, and 10).

Legend

OK	Item reported adequately in the manuscript.
Partial	Item addressed in part; some required elements present but not fully detailed.
N/A	Item applies to this study design but the underlying condition does not apply (for example, no data splits exist, no fine-tuning performed, no probability thresholds used).

Compliance Checklist

Item	Section	Topic	Status	Compliance Statement
1	Title	Study identification, task, population, outcome	**OK**	Title identifies the study as evaluating prompt strategies on LLM-based assessment performance of AI-generated clinical support information in critical care.
2a	Abstract	Title identification	**OK**	Resolved via the revised title, which identifies the study, task, and critical care context.
2c	Abstract	Objectives	**OK**	Objectives clearly stated in the abstract: examining whether prompt architecture influences scoring patterns and concordance with clinical raters.
2d	Abstract	Study setting	**OK**	Post-hoc analysis design stated in the abstract Methods.
2e	Abstract	Data and splits	**N/A**	No training, tuning, or evaluation splits exist in this post-hoc analysis; the 90-response dataset is fully described in the abstract Methods.
2f	Abstract	LLM name and version	**OK**	GPT-4o (Azure AI, version 2024-11-20) and o3-mini (Azure AI, version 2025-01-31).
2g	Abstract	LLM-building steps	**N/A**	No fine-tuning, reward modeling, or RLHF was performed.
2h	Abstract	Specific tasks	**OK**	Generation of clinical support outputs and structured IMPACT scoring clearly described.
2i	Abstract	Evaluation datasets, endpoint, measures	**OK**	ICC and Fisher's z comparisons stated.
2j	Abstract	Results	**OK**	Results summarised with ICC values and score deviations.
2k	Abstract	Broader implications	**OK**	Broader implications addressed in the closing sentence.
3a	Introduction	Healthcare context and rationale	**OK**	Introduction paragraphs 1 and 2 cover critical care context, LLM-as-judge paradigm, RLHF, and prior validation work with appropriate references.
4	Introduction	Study objectives	**OK**	Hypothesis statement at end of Introduction satisfies this item.

Item	Section	Topic	Status	Compliance Statement
5a	Methods	Data sources	**OK**	MIMIC-IV v3.1 source identified; outputs from prior study used as test materials.
5b	Methods	Data points and distribution	**OK**	Patient characteristics presented in Table 1.
5c	Methods	Date of oldest and newest text	**OK**	90 GPT-4o responses generated between February 2025 and March 2025; MIMIC-IV v3.1 admissions span 2008 to 2019.
5d	Methods	Preprocessing and quality checking	**OK**	No filtering, deduplication, or reformatting was performed; responses used as generated.
5e	Methods	Missing and imbalanced data	**OK**	Post-hoc study with no missing data; all 90 responses scored by all clinical raters and by the o3-mini evaluator.
6a	Methods	LLM name, version, training cutoff	**OK**	GPT-4o (version 2024-11-20) and o3-mini (version 2025-01-31) accessed via Azure OpenAI; both models have a stated knowledge cutoff of October 1, 2023.
6b	Methods	LLM development details	**N/A**	No de novo LLM development; no fine-tuning, alignment, or architecture modification performed.
6c	Methods	Prompt engineering and inference settings	**OK**	GPT-4o used with temperature 0.6 and top-p 0.9; o3-mini temperature and top-p cannot be set by vendor design; random seed fixed at 42 for both models. Verbatim prompts provided in Additional File 1.
6d	Methods	Initial and post-processed output	**OK**	o3-mini returned structured JSON object with integer 0-4 score and scoring rationale for each of the 25 IMPACT items; total scores computed programmatically as the sum.
7d	Methods	Assessor qualifications	OK	Six clinical raters participated: two intensivists (21 and 3 years of ICU experience), one resident (3 years), and three ICU nurses (16, 10, and 2 years). Human evaluation used the E1 standard scoring prompt. All raters received an introduction to the scoring criteria and one scored example for

Item	Section	Topic	Status	Compliance Statement
				calibration before evaluation
7e	Methods	Benchmark comparison	**OK**	Performance compared against the mean of three clinician raters as the reference standard.
8a	Methods	Annotation guidelines and examples	**OK**	Full 25-item IMPACT rubric with 0-to-4 scoring anchors provided in Additional File 1.
8b	Methods	Number of annotators and overlap	**OK**	Each response independently assessed by three clinical raters, providing multiple expert perspectives and 100% overlap.
8c	Methods	Annotator background	**OK**	Rater backgrounds reported: two intensivists (21 and 3 years of ICU experience), one resident (3 years), and three ICU nurses (16, 10, and 2 years); all raters received calibration training before evaluation.
9a	Methods	Prompt design, curation, selection	**OK**	Three prompt strategies described conceptually and illustrated in Figure 1; verbatim E1, E2, and E3 prompts provided in Additional File 1.
9b	Methods	Data used to develop prompts	**N/A**	Prompts designed a priori based on the published IMPACT framework; no empirical prompt tuning on the 90 evaluation cases.
11	Methods	Instruction tuning or alignment	**N/A**	No instruction tuning, fine-tuning, or alignment training performed; both models used as deployed by Azure OpenAI.
12	Methods	Compute	**N/A**	Post-hoc analysis conducted entirely via the Azure OpenAI API; no local or custom compute required.
13	Methods	Ethics approval	**OK**	MIT IRB (No. 0403000206) and BIDMC IRB (2001-P-001699/14) approvals reported; CITI certifications obtained.
14a	Methods	Funding source and role	**OK**	NSTC 112-2314-B-002-274 and NTUH 112-FY0002 grants acknowledged; funders had no role in study design, analysis, interpretation, or decision to publish.
14b	Methods	Conflicts of interest	**OK**	Authors declared no potential conflicts of interest.

Item	Section	Topic	Status	Compliance Statement
14e	Methods	Data availability	**OK**	Prompt materials and supporting data are available from the corresponding author upon reasonable request, as stated in the Data Sharing Statement.
14f	Methods	Code availability	**OK**	Code is available in the Additional file 1.
17	Results	Performance	**OK**	IMPACT scores, ICC values with 95% CIs, Bland-Altman analysis, and repeated-measures ANOVA reported with post-hoc comparisons.
18	Results	LLM updating	**OK**	GPT-4o and o3-mini used at fixed Azure deployment versions throughout the evaluation period; no model updating or retraining occurred.
19a	Discussion	Interpretation and fairness	**OK**	Overall interpretation provided with reference to prior studies and fairness considerations including anchoring and positional bias.
19b	Discussion	Limitations	**OK**	Single-center data, scale-orientation effects, and single-model-version constraints acknowledged.
19g	Discussion	Future research	**OK**	Future research directions stated in the closing paragraph and conclusion, including scoring orientation studies and re-evaluation with newer model architectures.

Summary

Total items required for M design: 41. OK: 31. Partial: 4. N/A: 6.

Reference: Gallifant J, Afshar M, Ameen S, et al. The TRIPOD-LLM reporting guideline for studies using large language models. Nat Med. 2025;31:60-69. https://www.tripod-statement.org

2. Python script for generating large language model responses

2.1 P1

```
import os
import sys
import json
import requests
import time
from datetime import datetime
import pandas as pd
import logging
import re    # for post-processing and regex operations
import csv    # Added for CSV quoting

logging.basicConfig(level=logging.INFO)

##############################################################################
##########
# 1) DEFINE FEATURES AND HOW TO FORMAT THEM
##############################################################################
##########
categorical_features = [
        "congestive_heart_failure",
        "cerebrovascular_disease",
        "dementia",
        "chronic_pulmonary_disease",
        "rheumatic_disease",
        "mild_liver_disease",
        "renal_disease",
        "malignant_cancer",
        "severe_liver_disease",
        "metastatic_solid_tumor",
        "emerg_ad",
        "first_icustay",
        "dm",
        "intubated_0h"
]

integer_features = [
        "gcs_0h",
        "heart_rate_0h",
        "resp_rate_0h",
        "sbp_0h",
        "dbp_0h",
        "mbp_0h",
        "spo2_0h"
```

```
]

numeric_features = [
    "admission_age",
    "temperature_0h",
    "weight_kg"
]

# Combine all features (ordering is not critical as we split by SHAP sign later)
ordered_features = categorical_features + integer_features + numeric_features

##########################################################################
##########
# 2) LOAD CSV (SEMICOLON-DELIMITED)
##########################################################################
##########
def load_query_file(filepath):
    """
    Load a CSV that is delimited by semicolons.
    Ensure 'stay_id' is integer.
    """
    data = pd.read_csv(filepath, sep=';')
    data['stay_id'] = data['stay_id'].astype(int)
    return data

##########################################################################
##########
# 3) FEATURE FORMATTING
##########################################################################
##########
def format_feature_value(feature_name, value):
    if value is None:
        return "N/A"
    try:
        float_val = float(value)
    except ValueError:
        return str(value)
    # For categorical or integer features, show as integer
    if feature_name in categorical_features or feature_name in integer_features:
        return str(int(round(float_val)))
    # For numeric features, use 0 decimals for admission_age, 1 decimal for others.
    elif feature_name in numeric_features:
        if feature_name == "admission_age":
            return f"{float_val:.0f}"
        else:
            return f"{float_val:.1f}"
    else:
        return str(value)

def get_feature_value(row, feature_name):
```

```
    try:
        return row[feature_name]
    except KeyError:
        logging.error(f"Feature '{feature_name}' not found in the data.")
        return None

def get_feature_shap_value(row, feature_name):
    shap_col = f"{feature_name}_SHAP"
    try:
        return row[shap_col]
    except KeyError:
        logging.error(f"SHAP value for feature '{feature_name}' not found in the
data.")
        return None

##############################################################################
##########
# 4) CALL AZURE OPENAI
##############################################################################
##########
def call_azure_openai(api_endpoint, api_key, system_prompt, user_prompt,
temperature, top_p, max_tokens):
    """
    Return (llm_response_text, usage_info)
    usage_info may contain {"prompt_tokens":..., "completion_tokens":...,
"total_tokens":...}
    """
    headers = {
        "Content-Type": "application/json",
        "api-key": api_key
    }
    payload = {
        "messages": [
            {"role": "system", "content": system_prompt},
            {"role": "user", "content": user_prompt}
        ],
        "temperature": temperature,
        "top_p": top_p,
        "max_tokens": max_tokens
    }

    max_retries = 3
    backoff_factor = 2    # seconds exponentiation base

    for attempt in range(1, max_retries + 1):
        try:
            response = requests.post(api_endpoint, headers=headers,
json=payload, timeout=360)
            response.raise_for_status()
            data = response.json()
```

```
            llm_response_text = data["choices"][0]["message"]["content"]
            usage_info = data.get("usage", {})
            return llm_response_text, usage_info
        except requests.exceptions.RequestException as e:
            logging.error(f"Attempt {attempt}: Error calling Azure OpenAI: {e}")
            if e.response is not None:
                logging.error(f"Response text: {e.response.text}")
            if attempt == max_retries:
                logging.error("Max retries reached. Exiting.")
                sys.exit(1)
            else:
                wait_time = backoff_factor ** attempt
                logging.info(f"Retrying in {wait_time} seconds...")
                time.sleep(wait_time)

################################################################################
##########
# 5) POST-PROCESSING FUNCTION FOR LLM RESPONSE
################################################################################
##########
def postprocess_llm_response(response_text):
    """
    In this example, we assume the LLM will use the required section headings as
per the prompt.
    Additional postprocessing can be added if needed.
    """
    return response_text

################################################################################
##########
# 6) PREPARE PROMPT (SEPARATE POSITIVE AND NEGATIVE SHAP
FEATURES),
#       RETURN LLM RESPONSE, SORTED REPORT, USAGE
################################################################################
##########
def send_to_llm(row_data, api_endpoint, api_key, temperature, top_p, max_tokens):
    """
    - Separate features into positive and negative SHAP groups.
    - Sort positive features by descending SHAP value; negative features by
ascending.
    - Build the prompt and return (llm_response, sorted_feature_report, usage_info).
    """
    predicted_risk = row_data["predicted_risk"]

    # Separate features into positive and negative groups
    positive_features = []
    negative_features = []
    for feature in ordered_features:
        val = get_feature_value(row_data, feature)
        shap_val = get_feature_shap_value(row_data, feature)
```

```
            if shap_val is None:
                continue
            if shap_val >= 0:
                positive_features.append((feature, val, shap_val))
            else:
                negative_features.append((feature, val, shap_val))

        # Sort features
        positive_features_sorted = sorted(positive_features, key=lambda x: x[2],
reverse=True)
        negative_features_sorted = sorted(negative_features, key=lambda x: x[2])

        # Build reports
        positive_report_lines = [
            f"SHAP = {shap_val:.4f}, {feature} = {format_feature_value(feature,
val)}"
            for feature, val, shap_val in positive_features_sorted
        ]
        negative_report_lines = [
            f"SHAP = {shap_val:.4f}, {feature} = {format_feature_value(feature,
val)}"
            for feature, val, shap_val in negative_features_sorted
        ]

        section_positive = "2. Features with positive SHAP values\n" + "\n".join("      -"
+ line for line in positive_report_lines)
        section_negative = "3. Features with negative SHAP values\n" + "\n".join("      -"
+ line for line in negative_report_lines)
        sorted_feature_report = section_positive + "\n" + section_negative

        # Updated user prompt without the major risk factors section
        user_prompt = (
            f"1. Predicted ICU Mortality Risk: {predicted_risk:.1%}\n"
            f"{sorted_feature_report}\n\n"
            "Based on the above patient-specific information, please provide your
answer in plain text (do not use markdown formatting, e.g., no '**' for bold). Your
response must be tailored to the actual clinical condition of this patient and should not
include generalized recommendations. Only recommend interventions that are
supported by the patient's data; if an intervention is only applicable under certain
conditions, qualify it with phrases such as 'if indicated' or 'for high risk of impending
[event]'. "
            "Your response must include the following sections with exactly these
headings:\n"
            "- [Interpretation of Risk Factors]\n"
            "- [Recommended Examinations]\n"
            "- [Recommended Management]\n"
            "- [Follow-up Plan]\n"
            "- [Summary]\n\n"
            "For example, start your answer with:\n"
            "[Interpretation of Risk Factors]\n"
```

```
            "and then continue with your interpretation, followed by the other sections as instructed.\n"
        )

        system_prompt = (
            "You are an elite Intensivist. Evaluate the risk factors of ICU mortality and provide professional recommendations in plain text. Do not use contractions in your response."
        )

        llm_response, usage_info = call_azure_openai(
            api_endpoint=api_endpoint,
            api_key=api_key,
            system_prompt=system_prompt,
            user_prompt=user_prompt,
            temperature=temperature,
            top_p=top_p,
            max_tokens=max_tokens
        )

        llm_response = postprocess_llm_response(llm_response)
        return llm_response, sorted_feature_report, usage_info

################################################################################
# 6.1) HELPER: EXTRACT SECTION FROM LLM RESPONSE
################################################################################
def extract_section(analysis_text, section_title):
        """
        Extracts text from the LLM response that follows the section title.
        The section title can be recognized whether or not it is enclosed in square brackets.
        Returns the text until the next known section header or the end of text.
        """
        lower_text = analysis_text.lower()
        lower_title = section_title.lower()
        marker_with_brackets = f"[{section_title}]".lower()

        if marker_with_brackets in lower_text:
            start_idx = lower_text.find(marker_with_brackets) + len(marker_with_brackets)
            parts = analysis_text[start_idx:]
        elif lower_title in lower_text:
            start_idx = lower_text.find(lower_title) + len(section_title)
            parts = analysis_text[start_idx:]
        else:
            return ""

        # List known section headers that might follow.
```

```
    possible_headers = [
        "Interpretation of Risk Factors",
        "Recommended Examinations",
        "Recommended Management",
        "Follow-up Plan",
        "Summary"
    ]

    next_header_index = None
    lower_parts = parts.lower()
    for header in possible_headers:
        if header.lower() == lower_title:
            continue
        idx = lower_parts.find(header.lower())
        if idx != -1:
            if next_header_index is None or idx < next_header_index:
                next_header_index = idx
    if next_header_index is not None:
        section_text = parts[:next_header_index].strip()
    else:
        section_text = parts.strip()
    return section_text

################################################################################
##########
# 7) SAVE REPORTS TO TEXT FILE
################################################################################
##########
def save_llm_response_txt(response_text, stay_id, LLM_MODEL, PROMPT,
save_directory="C:\\workspace\\001_GAI\\LLM"):
    """
    Save the LLM response to a text file with a revised naming convention.
    The LLM_MODEL's trailing date is removed if present.
    """
    date_str = datetime.now().strftime("%Y%m%d")
    base_model = re.sub(r'\s+\d{4}-\d{2}-\d{2}$', '', LLM_MODEL)
    filename = f"R_{base_model}_{stay_id}_{date_str}_{PROMPT}.txt"
    filepath = os.path.join(save_directory, filename)
    with open(filepath, 'w', encoding='utf-8') as file:
        file.write(response_text)
    logging.info(f"Saved text file for stay_id {stay_id} at {filepath}")

################################################################################
##########
# 8) DETERMINE RISK GROUP
################################################################################
##########
def get_risk_group(predicted_risk):
    """
    Determine risk group based on predicted_risk (decimal form).
```

```
    Returns "l" for < 10%, "m" for 10%-19.99%, and "h" for >= 20%.
    """
    if predicted_risk < 0.1:
        return "l"
    elif predicted_risk < 0.2:
        return "m"
    else:
        return "h"

################################################################################
##########
# 9) MAIN WORKFLOW: PROCESS EACH RECORD AND COLLECT
SUMMARY ROWS
################################################################################
##########
def complete_workflow(filepath, azure_api_key, azure_endpoint, save_directory,
LLM_MODEL, PROMPT, RAG, ML_MODEL, llm_waiting_default, temperature,
top_p, max_tokens, csv_path):
    data = load_query_file(filepath)
    summary_rows = []

    # Process each admission one by one
    for idx, row in data.iterrows():
        record_start_time = time.time()
        stay_id = int(row["stay_id"])
        real_icu_mort = row.get("icu_mortality", "N/A")
        if real_icu_mort not in ["N/A", None]:
            try:
                real_icu_mort = str(int(round(float(real_icu_mort))))
            except Exception:
                pass

        if "predicted_risk" not in row:
            logging.error(f"No 'predicted_risk' for stay_id {stay_id}.")
            continue
        predicted_risk = row["predicted_risk"]
        risk_group = get_risk_group(predicted_risk)

        try:
            llm_time_start = time.time()
            # Pass explicit configuration values for LLM call
            llm_response, sorted_feature_report, usage_info = send_to_llm(
                row_data=row,
                api_endpoint=azure_endpoint,
                api_key=azure_api_key,
                temperature=temperature,
                top_p=top_p,
                max_tokens=max_tokens
            )
```

```
                # Reconstruct the base context (patient info) without modifying
send_to_llm.
                patient_info = f"1. Predicted ICU Mortality Risk:
{predicted_risk:.1%}\n{sorted_feature_report}\n\n"

                llm_time_secs = time.time() - llm_time_start
                total_record_time_secs = time.time() - record_start_time

                prompt_tokens = usage_info.get("prompt_tokens", 0)
                completion_tokens = usage_info.get("completion_tokens", 0)
                total_tokens = usage_info.get("total_tokens", 0)

                # Extract sections from the LLM response
                interpretation = extract_section(llm_response, "Interpretation of Risk
Factors")
                examination = extract_section(llm_response, "Recommended
Examinations")
                management = extract_section(llm_response, "Recommended
Management")
                follow_up = extract_section(llm_response, "Follow-up Plan")
                summary_sec = extract_section(llm_response, "Summary")

                full_response = (
                    f"Report Date: {datetime.now().strftime('%Y-%m-%d
%H:%M:%S')}\n\n"
                    "---Hidden from LLM---\n"
                    f"Stay_id: {stay_id}\n"
                    f"ML Model: {ML_MODEL}\n"
                    f"LLM model: {LLM_MODEL}\n"
                    f"Prompt: {PROMPT}\n"
                    f"RAG: {RAG}\n"
                    f"ICU Mortality: {real_icu_mort}\n\n"
                    "---Information for LLM---\n\n"
                    f"1. Predicted ICU Mortality Risk: {predicted_risk:.1%}\n" +
                    f"{sorted_feature_report}\n\n" +
                    "--- Full LLM Response ---\n\n"
                    f"{llm_response}\n"
                )

                # Save text file for this admission
                save_llm_response_txt(full_response, stay_id, LLM_MODEL,
PROMPT, save_directory)

                # Compute additional timing metrics
                llm_response_t = llm_time_secs                  # LLM response
time
                llm_waiting_t = llm_waiting_default             # Now use the
explicitly passed waiting time
                python_time = total_record_time_secs - llm_time_secs    # Time spent
in Python processing
```

```
                total_time = total_record_time_secs                # Total processing
time

                summary_row = {
                    "stay_id": stay_id,
                    "ICU_mortality": real_icu_mort,
                    "Predicted_risk(%)": f"{predicted_risk * 100:.1f}",
                    "risk_group": risk_group,
                    "ML_model": ML_MODEL,
                    "LLM_model": LLM_MODEL,
                    "Prompt": PROMPT,
                    "RAG": RAG,
                    "prompt_tokens": prompt_tokens,
                    "completion_tokens": completion_tokens,
                    "total_tokens": total_tokens,
                    "LLM_response_t(s)": int(llm_response_t),
                    "LLM_waiting_t(s)": int(llm_waiting_t),
                    "Python_time(s)": int(python_time),
                    "Total_time(s)": int(total_time),
                    "Patient": patient_info,
                    "Interpretation": interpretation,
                    "Examination": examination,
                    "Management": management,
                    "Follow_up": follow_up,
                    "Summary": summary_sec,
                }

                summary_rows.append(summary_row)

                # Task 2: Save CSV file after finishing one case
                # Replace newline characters with a space in specified multiline
columns
                multiline_columns = ["Patient", "Interpretation", "Examination",
"Management", "Follow_up", "Summary"]
                for col in multiline_columns:
                    if col in summary_row:
                        summary_row[col] = summary_row[col].replace("\n", " ")
                # Append the current row to the CSV file
                df_row = pd.DataFrame([summary_row])
                df_row.to_csv(csv_path, mode='a', index=False, header=False,
quoting=csv.QUOTE_ALL)
                logging.info(f"Appended CSV row for stay_id {stay_id} to
{csv_path}")

            except Exception as e:
                logging.error(f"Error processing stay_id {stay_id}: {str(e)}")

        # Return summary rows (CSV saving is now handled case-by-case)
        return summary_rows
```

```
################################################################################
##########
# 10) EXAMPLE USAGE
################################################################################
##########
if __name__ == "__main__":
    FILEPATH = "C:\\workspace\\001_GAI\\LLM\\adm\\v_llm_group.csv"
    AZURE_ENDPOINT = " YOUR AZURE URI"
    AZURE_API_KEY = "YOUR KEY"
    ML_MODEL = "XGBoost_20250207"
    LLM_MODEL = "GPT 4o 2024-11-20"    # Trailing date will be removed for file
naming
    PROMPT = "P1"
    RAG = "None"
    SAVE_DIRECTORY = "C:\\workspace\\001_GAI\\LLM"

    # Settings for the LLM call (easily changeable here)
    TEMPERATURE = 0.6
    TOP_P = 0.9
    MAX_TOKENS = 4000
    LLM_WAITING_DEFAULT = 0

    # Task 1: Construct CSV file name using model name and current date, and
check if it exists/is accessible.
    base_model = re.sub(r'\s+\d{4}-\d{2}-\d{2}$', '', LLM_MODEL)
    date_str = datetime.now().strftime("%Y%m%d")
    csv_name = f"LLM_Response_Summary_{base_model}_{date_str}.csv"
    summary_csv_path = os.path.join(SAVE_DIRECTORY, csv_name)

    if os.path.exists(summary_csv_path):
        try:
            with open(summary_csv_path, 'a') as f:
                pass
            logging.info("Notice: CSV file exists and is accessible. Consider
renaming the file if you want to preserve the existing data.")
        except Exception as e:
            logging.error("Warning: CSV file exists but cannot be opened. Please
save, close, or rename the file and then restart the process.")
            sys.exit(1)
    else:
        # If the file does not exist, create it with header.
        columns = ["stay_id", "ICU_mortality", "Predicted_risk(%)", "risk_group",
"ML_model", "LLM_model", "Prompt", "RAG", "prompt_tokens",
"completion_tokens", "total_tokens", "LLM_response_t(s)", "LLM_waiting_t(s)",
"Python_time(s)", "Total_time(s)", "Patient", "Interpretation", "Examination",
"Management", "Follow_up", "Summary"]
        summary_df = pd.DataFrame(columns=columns)
        summary_df.to_csv(summary_csv_path, index=False,
quoting=csv.QUOTE_ALL)
```

```
results = complete_workflow(
        filepath=FILEPATH,
        azure_api_key=AZURE_API_KEY,
        azure_endpoint=AZURE_ENDPOINT,
        save_directory=SAVE_DIRECTORY,
        LLM_MODEL=LLM_MODEL,
        PROMPT=PROMPT,
        RAG=RAG,
        ML_MODEL=ML_MODEL,
        llm_waiting_default=LLM_WAITING_DEFAULT,
        temperature=TEMPERATURE,
        top_p=TOP_P,
        max_tokens=MAX_TOKENS,
        csv_path=summary_csv_path
)

# print("\nAll results:\n", results)
```

2.2 P2

```
import os
import sys
import json
import requests
import time
from datetime import datetime
import pandas as pd
import logging
import csv
import re    # for post-processing and regex operations
import textwrap    # for wrapping text lines

import openai #from openai import AzureOpenAI    # using the latest client for
AZUREOPENAI

logging.basicConfig(level=logging.INFO)

#########################################################################
##########
# 1) DEFINE FEATURES AND HOW TO FORMAT THEM
#########################################################################
##########
categorical_features = [
        "congestive_heart_failure",
        "cerebrovascular_disease",
        "dementia",
        "chronic_pulmonary_disease",
        "rheumatic_disease",
        "mild_liver_disease",
        "renal_disease",
        "malignant_cancer",
        "severe_liver_disease",
        "metastatic_solid_tumor",
        "emerg_ad",
        "first_icustay",
        "dm",
        "intubated_0h"
]

integer_features = [
        "gcs_0h",
        "heart_rate_0h",
        "resp_rate_0h",
        "sbp_0h",
        "dbp_0h",
        "mbp_0h",
        "spo2_0h"
]
```

```
numeric_features = [
    "admission_age",
    "temperature_0h",
    "weight_kg"
]

# Combine all features (ordering is not critical as we split by SHAP sign later)
ordered_features = categorical_features + integer_features + numeric_features

################################################################################
##########
# 2) LOAD CSV (SEMICOLON-DELIMITED)
################################################################################
##########
def load_query_file(filepath):
    """
    Load a CSV that is delimited by semicolons.
    Ensure 'stay_id' is integer.
    """
    data = pd.read_csv(filepath, sep=';')
    data['stay_id'] = data['stay_id'].astype(int)
    return data

################################################################################
##########
# 3) FEATURE FORMATTING
################################################################################
##########
def format_feature_value(feature_name, value):
    if value is None:
        return "N/A"
    try:
        float_val = float(value)
    except ValueError:
        return str(value)
    # For categorical or integer features, show as integer
    if feature_name in categorical_features or feature_name in integer_features:
        return str(int(round(float_val)))
    # For numeric features, use 0 decimals for admission_age, 1 decimal for others.
    elif feature_name in numeric_features:
        if feature_name == "admission_age":
            return f"{float_val:.0f}"
        else:
            return f"{float_val:.1f}"
    else:
        return str(value)

def get_feature_value(row, feature_name):
    try:
        return row[feature_name]
```

```
        except KeyError:
            logging.error(f"Feature '{feature_name}' not found in the data.")
            return None

def get_feature_shap_value(row, feature_name):
        shap_col = f"{feature_name}_SHAP"
        try:
            return row[shap_col]
        except KeyError:
            logging.error(f"SHAP value for feature '{feature_name}' not found in the
data.")
            return None

##########################################################################
##########
# 4) CALL AZURE OPENAI
##########################################################################
##########
def call_azure_openai(api_endpoint, api_key, system_prompt, user_prompt,
                                temperature=0.6, top_p=0.9, max_tokens=4000):
        """
        Return (llm_response_text, usage_info)
        usage_info may contain {"prompt_tokens":..., "completion_tokens":...,
"total_tokens":..., "waiting_time":...}
        """
        headers = {
            "Content-Type": "application/json",
            "api-key": api_key
        }
        payload = {
            "messages": [
                {"role": "system", "content": system_prompt},
                {"role": "user", "content": user_prompt}
            ],
            "temperature": temperature,
            "top_p": top_p,
            "max_tokens": max_tokens
        }

        # Use the TIMEOUT setting from the global scope (default to 60 if not defined)
        timeout_val = globals().get("TIMEOUT", 480)
        total_wait_time = 0
        attempt = 0
        while attempt < MAX_RETRIES:
            try:
                response = requests.post(api_endpoint, headers=headers,
json=payload, timeout=timeout_val)
                if response.status_code == 429:
                    attempt += 1
                    logging.warning(f"Received 429 error, retrying
```

```
{attempt}/{MAX_RETRIES} after waiting {DEFAULT_WAIT_SECONDS}
seconds.")
                        time.sleep(DEFAULT_WAIT_SECONDS)
                        total_wait_time += DEFAULT_WAIT_SECONDS
                        continue
                response.raise_for_status()
                data = response.json()
                llm_response_text = data["choices"][0]["message"]["content"]
                usage_info = data.get("usage", {})
                usage_info["waiting_time"] = total_wait_time    # Include accumulated
waiting time
                return llm_response_text, usage_info
            except requests.exceptions.Timeout as e:
                attempt += 1
                logging.warning(f"Timeout encountered, retrying
{attempt}/{MAX_RETRIES} after waiting {DEFAULT_WAIT_SECONDS}
seconds.")
                time.sleep(DEFAULT_WAIT_SECONDS)
                total_wait_time += DEFAULT_WAIT_SECONDS
                continue
            except requests.exceptions.RequestException as e:
                if hasattr(e, 'response') and e.response is not None and
e.response.status_code == 429:
                    attempt += 1
                    logging.warning(f"Received 429 error in exception, retrying
{attempt}/{MAX_RETRIES} after waiting {DEFAULT_WAIT_SECONDS}
seconds.")
                    time.sleep(DEFAULT_WAIT_SECONDS)
                    total_wait_time += DEFAULT_WAIT_SECONDS
                    continue
                logging.error(f"Error calling Azure OpenAI: {e}")
                if hasattr(e, 'response') and e.response is not None:
                    logging.error(f"Response text: {e.response.text}")
                sys.exit(1)

    logging.error(f"Max retries exceeded ({MAX_RETRIES}) for
call_azure_openai.")
    sys.exit(1)

##########################################################################
##########
# 5) POST-PROCESSING FUNCTION FOR LLM RESPONSE
##########################################################################
##########
def postprocess_llm_response(response_text):
    """
    In this example, we assume the LLM will use the required section headings as
per the prompt.
    Additional postprocessing can be added if needed.
```

```
    This updated function removes all occurrences of "**" from the response.
    """
    # Remove markdown bold formatting by removing all occurrences of "**"
    response_text = response_text.replace("**", "")
    return response_text

################################################################################
##########
# NEW FUNCTION: WRAP TEXT TO 80 CHARACTERS (Task 3)
################################################################################
##########
def wrap_text_line(line, width=80):
    # Check for bullet or numbered list markers (e.g., "-", "*", "1.")
    pattern = r'^(\s*([-*]|\d+\.)\s+)'
    m = re.match(pattern, line)
    if m:
        prefix = m.group(1)
        rest = line[len(prefix):]
        wrapped_lines = textwrap.wrap(rest, width=width - len(prefix))
        return "\n".join(prefix + l for l in wrapped_lines)
    else:
        return "\n".join(textwrap.wrap(line, width=width))

def wrap_text(text, width=80):
    # Process text line by line to wrap each one individually.
    lines = text.splitlines()
    wrapped = [wrap_text_line(line, width) if len(line) > width else line for line in
lines]
    return "\n".join(wrapped)

################################################################################
##########
# 6) PREPARE PROMPT (SEPARATE POSITIVE AND NEGATIVE SHAP
FEATURES),
#       RETURN LLM RESPONSE, SORTED REPORT, USAGE
################################################################################
##########
def send_to_llm(row_data, api_endpoint, api_key):
    """
    - Separate features into positive and negative SHAP groups.
    - Sort positive features by descending SHAP value; negative features by
ascending.
    - Build the prompt and return (llm_response, sorted_feature_report, usage_info).
    """
    predicted_risk = row_data["predicted_risk"]

    # Separate features into positive and negative groups
    positive_features = []
    negative_features = []
    for feature in ordered_features:
```

```
            val = get_feature_value(row_data, feature)
            shap_val = get_feature_shap_value(row_data, feature)
            if shap_val is None:
                continue
            if shap_val >= 0:
                positive_features.append((feature, val, shap_val))
            else:
                negative_features.append((feature, val, shap_val))

    # Sort features
    positive_features_sorted = sorted(positive_features, key=lambda x: x[2],
reverse=True)
    negative_features_sorted = sorted(negative_features, key=lambda x: x[2])

    # Build reports
    positive_report_lines = [
        f"SHAP = {shap_val:.4f}, {feature} = {format_feature_value(feature,
val)}"
        for feature, val, shap_val in positive_features_sorted
    ]
    negative_report_lines = [
        f"SHAP = {shap_val:.4f}, {feature} = {format_feature_value(feature,
val)}"
        for feature, val, shap_val in negative_features_sorted
    ]

    section_positive = "2. Features with positive SHAP values\n" + "\n".join("  -"
+ line for line in positive_report_lines)
    section_negative = "3. Features with negative SHAP values\n" + "\n".join("  -"
+ line for line in negative_report_lines)
    sorted_feature_report = section_positive + "\n" + section_negative

    # Updated user prompt without the major risk factors section
    user_prompt = (
        f"1. Predicted ICU Mortality Risk: {predicted_risk:.1%}\n"
        f"{sorted_feature_report}\n\n"
"""
```

Task 1: Interpretation of Risk Factors

1. Instruction

Focus on the highest SHAP-valued features, grouping those that reflect the same condition (e.g., hemodynamic related indicators) to reduce redundancy.

Exclude features with minimal impact unless they yield notably high SHAP values, and avoid overstating mild abnormalities.

Highlight significant clinical findings, prioritizing variables strongly linked to mortality risk.

When evaluating hemodynamic parameters, provide context by noting baseline blood pressure values and prior hypertension, and compare SHAP values across related features to identify key contributors.

For categorical features, interpret “1” as present and “0” as absent; first_icustay distinguishes a repeat ICU admission from a first.

Recognize mbp as mean arterial pressure, flagging discrepancies if MAP exceeds systolic or dips below diastolic.
If mortality risk exceeds two percent, list three to six major factors; if below that threshold, advise vigilance for unanticipated events.
Check serious conditions only if relevant, and refrain from interpreting negative SHAP values.
Do not provide diagnostic or management advice in this task.
2. Format of Response (plain text)
[Interpretation of Risk Factors]
1. Major Risk Facotrs and Their Potential Causes
2. Interactions and Other Risk Factors
3. Quick conclusion

Task 2: Recommended Examinations
1. Instruction
Remain guided by the patient's current clinical presentation.
Focus first on immediate bedside assessments that address the patient's critical risk factors and potential life-threatening conditions.
Evaluate shock states, severe trauma, respiratory failure, or neurological crises by checking vitals, conducting targeted exams (e.g., point-of-care lactate, ECG), and applying bedside ultrasound if indicated.
In urgent assessments, limit blood tests (CBC, metabolic panel, infection markers, cardiac enzymes) and imaging (chest X-ray, ultrasound, CT) to those relevant to the patient's significant risks.
Cultures and urinalysis are only necessary if infection is strongly suspected.
For secondary and follow-up assessments, consider advanced imaging (CT, MRI), specialized biomarkers, or repeated labs based on evolving clinical status.
If the patient's condition necessitates close monitoring, evaluate fluid responsiveness, respiratory findings, or neurological changes, and consider relevant infectious disease testing (e.g., COVID-19).
Throughout, focus on identifying potential deterioration causes at ICU admission, matching recommendations to this patient's pressing issues.
Omit unnecessary steps and avoid treatment details, emphasizing diagnostic priorities only.
2. Format of Response (plain text)
[Recommended Examinations]
1. Immediate Bedside Assessment
2. Urgent Assessment
3. Secondary and Follow-up Assessment
4. Additional Monitoring
5. Quick Conclusion

Task 3: Recommended Management
1. Instruction
Tailor all recommendations to the patient's condition and significant risk factors.
In the first six hours (Immediate Resuscitation), stabilize life-threatening condition.
When clinically indicated, initiate fluid resuscitation by selecting the appropriate fluid type and volume—while accounting for the fluid administered before ICU admission—and concurrently begin vasopressor therapy (e.g., norepinephrine in mcg/kg/min) to achieve an initial target MAP of 65 mmHg, adjusting for

comorbidities as appropriately.
When indicated, follow guidelines (e.g., Surviving Sepsis) to justify albumin use, and provide antibiotic options covering resistant organisms or viral etiologies (COVID-19, influenza). Address arrhythmias with specified drugs and dosing, ensuring adequate sedation and airway protection if intubation is required.
In hours six to twenty-four (Acute Management), refine fluid balance, wean vasopressors, and manage organ support based on clinical indices.
Initiate RRT for fluid overload or severe electrolyte/acid-base derangements, and consider ECMO for refractory hypoxemia or cardiogenic shock. Use diuretics cautiously, avoiding ESRD patients.
Prevent ICU complications with measures like infection control, VAP prophylaxis, stress ulcer prevention, and delirium management (PADIS).
Link all recommendations to comorbidities, prior fluids, and targeted MAP, involving multidisciplinary teams if mortality risk is high.
2. Format of Response (plain text)
[Recommended Management]
1. Immediate Resuscitation and Management
2. Acute Management
3. ICU Supportive Care
4. Quick Conclusion

Task 4: Follow-up Plan
1. Instruction
Tailor all monitoring to the patient's condition and significant risk factors.
For Continuous Monitoring or Hourly Check, focus on vital signs, hemodynamics, and neurological status (e.g., GCS).
Adjust intervals if the patient improves.
In Acute Reassessment (Every 2–8 Hours), perform focused exams, obtain ABGs, reassess pain and delirium, and adjust therapies (e.g., vasopressors) based on clinical changes such as rising lactate.
For Stability Assessment (Day-by-Day), determine readiness to wean from supports and identify infection risks, escalating reassessments if deterioration occurs.
If new symptoms or warning signs appear, consider additional tests. Emphasize thorough handover from prior teams.
Manage pain, anxiety, and delirium (ICDSC or CAM-ICU) regularly.
2. Format of Response (plain text)
[Follow-up Plan]
1. Continuous Monitoring or Hourly Check
2. Acute Reassessment
3. Stability Follow-up

Task 5: Summary
1. Instruction
Provide a concise overview (100–150 words) summarizing the patient's risk factors, mortality risk, and the top three immediate actions.
Also include the reassessment schedule and any necessary consultations.
If mortality risk is over 20%, emphasize family discussion; if exceeding 60%, consider a time-limited trial.
2. Format of Response (plain text)
[Summary]

Starting paragraph.
Subheading with "Top 3 Priority Actions" in new line, then put every action in new line
Ending paragraph

"""
)

system_prompt = (
"""
You are an expert intensivist providing evidence-based and clinically relevant recommendations.
As you develop your response, internally perform a detailed, step-by-step chain-of-thought analysis.
Avoid over-escalation, excessive testing, or unnecessary interventions unless clearly justified.
Prioritize clinical severity, trends, and patient-specific context over generalized protocols.
Use these structured guidelines to assess ICU mortality risks and provide targeted recommendations.

Risk Factor Analysis
- Evaluate ICU mortality risk using key factors: age, SHAP values, GCS, vital signs, blood pressure, chronic diseases, and intubation status.
-Highlight key variables with positive SHAP values over 0.1 and justify their impact.
-Distinguish between confirmed clinical data (vital signs, labs) and inferences (diagnostic possibilities).
-Avoid overinterpreting mild abnormalities without context.

Recommendations
- Provide interventions consistent with vital signs and stability, following standard ICU guidelines.
- Avoid unnecessary imaging or lab tests unless they change management decisions
- Do not propose broad-spectrum antibiotics or vasopressors without strong evidence or suspicion.
- Justify each intervention and avoid unnecessary treatments.
- Cite relevant guidelines for lab or imaging studies.
- Base management decisions on severity and clinical trends.
- Distinguish between confirmed data and assumptions.
- Reference ARDSNet or Surviving Sepsis guidelines or other guidelines when appropriate.
- Focus on clinically significant findings rather than mild abnormalities.
- When uncertain, clarify assumptions or note additional data needed.

Response Requirements
- Use plain text only, no Markdown.
- Avoid contractions in all statements.
- Summarize succinctly without adding new details.
- Provide step-by-step reasoning internally; do not reveal it.

Your recommendations must be clinically sound, justified, and selective—avoiding reflexive protocols, unnecessary escalation, and excessive interventions.

"""
)

    # Corrected call to call_azure_openai as requested:
    llm_response, usage_info = call_azure_openai(
        api_endpoint=api_endpoint,
        api_key=api_key,
        system_prompt=system_prompt,
        user_prompt=user_prompt
)

    llm_response = postprocess_llm_response(llm_response)
    return llm_response, sorted_feature_report, usage_info

```
################################################################################
##########
# 6.1) HELPER: EXTRACT SECTION FROM LLM RESPONSE
################################################################################
##########
def extract_section(analysis_text, section_title):
    """
    Extracts text from the LLM response that follows the section title.
    The section title can be recognized whether or not it is enclosed in square
brackets.
    Returns the text until the next known section header or the end of text.
    """
    marker_with_brackets = f"[{section_title}]"
    if marker_with_brackets in analysis_text:
        parts = analysis_text.split(marker_with_brackets, 1)[1]
    elif section_title in analysis_text:
        parts = analysis_text.split(section_title, 1)[1]
    else:
        return ""

    # List known section headers that might follow.
    possible_headers = [
        "Interpretation of Risk Factors",
        "Recommended Examinations",
        "Recommended Management",
        "Follow-up Plan",
        "Summary"
    ]

    next_header_index = None
    for header in possible_headers:
        if header.lower() == section_title.lower():
            continue
```

```
            idx = parts.find(header)
            if idx != -1:
                if next_header_index is None or idx < next_header_index:
                    next_header_index = idx
        if next_header_index is not None:
            section_text = parts[:next_header_index].strip()
        else:
            section_text = parts.strip()
        return section_text

##########################################################################
##########
# 7) SAVE REPORTS TO TEXT FILE (Task 3: with text wrapping)
##########################################################################
##########
def save_llm_response_txt(response_text, stay_id, LLM_MODEL, PROMPT,
save_directory="C:\\workspace\\001_GAI\\LLM"):
        """
        Save the LLM response to a text file with a revised naming convention.
        The LLM_MODEL's trailing date is removed if present.
        The text is wrapped so that no line exceeds 80 characters.
        """
        # Wrap the response text
        response_text = wrap_text(response_text, width=80)
        date_str = datetime.now().strftime("%Y%m%d")
        base_model = re.sub(r'\s+\d{4}-\d{2}-\d{2}$', '', LLM_MODEL)
        filename = f"R_{base_model}_{stay_id}_{date_str}_{PROMPT}.txt"
        filepath = os.path.join(save_directory, filename)
        with open(filepath, 'w', encoding='utf-8') as file:
            file.write(response_text)
        print(f"Saved txt file for stay_id {stay_id}: {filepath}")

##########################################################################
##########
# NEW FUNCTION: CHECK CSV FILE ACCESS (Task 1)
##########################################################################
##########
def check_csv_file_access(save_directory, csv_filename):
        """
        Constructs the full CSV file path. If the file exists, tries opening in append mode.
        If unable to open (e.g., file is open in another program), prints a warning and
stops.
        If accessible, prints a notice suggesting renaming if preservation is desired.
        Returns the full CSV file path.
        """
        csv_path = os.path.join(save_directory, csv_filename)
        if os.path.exists(csv_path):
            try:
                with open(csv_path, 'a', encoding='utf-8'):
                    pass
```

```
                print(f"Notice: CSV file '{csv_path}' exists and is accessible. Consider
renaming the file if you want to preserve the existing data.")
            except Exception as e:
                print(f"Warning: Unable to open CSV file '{csv_path}' for appending.
Please save, close, or rename the file and then stop the process.")
                sys.exit(1)
    return csv_path

##########################################################################
##########
# NEW FUNCTION: APPEND SUMMARY ROW TO CSV (Task 2)
##########################################################################
##########
def append_summary_row_to_csv(row, csv_path):
    """
    Append a summary row (a dictionary) to the CSV file.
    For multiline columns, replace newline characters with spaces.
    Every cell is enclosed in quotes.
    """
    multiline_columns = ["Patient", "Interpretation", "Examination", "Management",
"Follow_up", "Summary"]
    for col in multiline_columns:
        if col in row and isinstance(row[col], str):
            row[col] = row[col].replace("\n", " ")
    file_exists = os.path.exists(csv_path)
    write_header = True
    if file_exists and os.path.getsize(csv_path) > 0:
        write_header = False
    with open(csv_path, 'a', newline='', encoding='utf-8') as csvfile:
        writer = csv.DictWriter(csvfile, fieldnames=row.keys(),
quoting=csv.QUOTE_ALL)
        if write_header:
            writer.writeheader()
        writer.writerow(row)
    print(f"Appended CSV row for stay_id {row.get('stay_id', 'unknown')} to file:
{csv_path}")

##########################################################################
##########
# 8) DETERMINE RISK GROUP
##########################################################################
##########
def get_risk_group(predicted_risk):
    """
    Determine risk group based on predicted_risk (decimal form).
    Returns "l" for < 10%, "m" for 10%-19.99%, and "h" for >= 20%.
    """
    if predicted_risk < 0.1:
        return "l"
    elif predicted_risk < 0.2:
```

```
            return "m"
        else:
            return "h"

##########################################################################
##########
# 9) MAIN WORKFLOW: PROCESS EACH RECORD AND COLLECT
SUMMARY ROWS
#               (Modified: Append CSV after every case - Task 2)
##########################################################################
##########
def complete_workflow(filepath, azure_api_key, api_endpoint, save_directory,
LLM_MODEL, PROMPT, RAG, ML_MODEL):
    data = load_query_file(filepath)
    summary_rows = []

    # Process each admission one by one
    for idx, row in data.iterrows():
        record_start_time = time.time()
        stay_id = int(row["stay_id"])
        real_icu_mort = row.get("icu_mortality", "N/A")
        if real_icu_mort not in ["N/A", None]:
            try:
                real_icu_mort = str(int(round(float(real_icu_mort))))
            except Exception:
                pass

        if "predicted_risk" not in row:
            logging.error(f"No 'predicted_risk' for stay_id {stay_id}.")
            continue
        predicted_risk = row["predicted_risk"]
        risk_group = get_risk_group(predicted_risk)

        try:
            llm_time_start = time.time()
            llm_response, sorted_feature_report, usage_info = send_to_llm(
                row_data=row,
                api_endpoint=api_endpoint,
                api_key=azure_api_key
            )
            # Reconstruct the base context (patient info)
            patient_info = f"1. Predicted ICU Mortality Risk:
{predicted_risk:.1%}\n{sorted_feature_report}\n\n"

            llm_time_secs = time.time() - llm_time_start
            total_record_time_secs = time.time() - record_start_time

            prompt_tokens = usage_info.get("prompt_tokens", 0)
            completion_tokens = usage_info.get("completion_tokens", 0)
            total_tokens = usage_info.get("total_tokens", 0)
```

```
                # Extract sections from the LLM response using the appropriate
markers
                interpretation = extract_section(llm_response, "Interpretation of Risk
Factors")
                examination = extract_section(llm_response, "Recommended
Examinations")
                management = extract_section(llm_response, "Recommended
Management")
                follow_up = extract_section(llm_response, "Follow-up Plan")
                summary_sec = extract_section(llm_response, "Summary")

                full_response = (
                    f"Report Date: {datetime.now().strftime('%Y-%m-%d
%H:%M:%S')}\n\n"
                    "---Hidden from LLM---\n"
                    f"Stay_id: {stay_id}\n"
                    f"ML Model: {ML_MODEL}\n"
                    f"LLM model: {LLM_MODEL}\n"
                    f"Prompt: {PROMPT}\n"
                    f"RAG: {RAG}\n"
                    f"ICU Mortality: {real_icu_mort}\n\n"
                    "---Information for LLM---\n\n"
                    f"1. Predicted ICU Mortality Risk: {predicted_risk:.1%}\n"
                    f"{sorted_feature_report}\n\n"
                    "--- Full LLM Response ---\n\n"
                    f"{llm_response}\n"
                )

                # Save text file for this admission (with wrapped text)
                save_llm_response_txt(full_response, stay_id, LLM_MODEL,
PROMPT, save_directory)

                # Compute additional timing metrics:
                llm_waiting_t = usage_info.get("waiting_time", 0)
                llm_response_t = llm_time_secs - llm_waiting_t
                python_time = total_record_time_secs - llm_time_secs
                total_time = total_record_time_secs

                summary_row = {
                    "stay_id": stay_id,
                    "ICU_mortality": real_icu_mort,
                    "Predicted_risk(%)": f"{predicted_risk * 100:.1f}",
                    "risk_group": risk_group,
                    "ML_model": ML_MODEL,
                    "LLM_model": LLM_MODEL,
                    "Prompt": PROMPT,
                    "RAG": RAG,
                    "prompt_tokens": prompt_tokens,
                    "completion_tokens": completion_tokens,
```

```
                    "total_tokens": total_tokens,
                    "LLM_response_t(s)": int(llm_response_t),
                    "LLM_waiting_t(s)": 0,
                    "Python_time(s)": int(python_time),
                    "Total_time(s)": int(total_time),
                    "Patient": patient_info,
                    "Interpretation": interpretation,
                    "Examination": examination,
                    "Management": management,
                    "Follow_up": follow_up,
                    "Summary": summary_sec,
                }
                summary_rows.append(summary_row)

                # Append summary row to CSV file immediately (Task 2)
                append_summary_row_to_csv(summary_row, OUTPUT_CSV_PATH)

            except Exception as e:
                logging.error(f"Error processing stay_id {stay_id}: {str(e)}")

    return summary_rows

################################################################################
##########
# 10) EXAMPLE USAGE
################################################################################
##########
if __name__ == "__main__":
    # Define global adjustable parameters for rate limiting and timeout
    MAX_RETRIES = 3              # Maximum number of retries on 429 errors
    DEFAULT_WAIT_SECONDS = 60    # Wait time for retrying
    TIMEOUT = 600 # Timeout for API calls (in seconds)

    FILEPATH = "C:\\workspace\\001_GAI\\LLM\\adm\\v_llm_group.csv"
    AZURE_ENDPOINT = " YOUR AZURE URI"
    AZURE_API_KEY = "YOUR KEY"
    ML_MODEL = "XGBoost_20250207"
    LLM_MODEL = "GPT 4o 2024-11-20"    # Trailing date will be removed for file
naming
    PROMPT = "P2"
    RAG = "None"
    SAVE_DIRECTORY = "C:\\workspace\\001_GAI\\LLM"

    # Task 1: Construct the CSV file name using model name, current date and
prompt, and check its access.
    date_str = datetime.now().strftime("%Y%m%d")
    base_model = re.sub(r'\s+\d{4}-\d{2}-\d{2}$', '', LLM_MODEL)
    csv_filename =
f"LLM_Response_Summary_{base_model}_{date_str}_{PROMPT}.csv"
    OUTPUT_CSV_PATH = check_csv_file_access(SAVE_DIRECTORY,
```

```
csv_filename)

    results = complete_workflow(
        filepath=FILEPATH,
        azure_api_key=AZURE_API_KEY,
        api_endpoint=AZURE_ENDPOINT,
        save_directory=SAVE_DIRECTORY,
        LLM_MODEL=LLM_MODEL,
        PROMPT=PROMPT,
        RAG=RAG,
        ML_MODEL=ML_MODEL
    )

    # IMPORTANT: Do not change the following code block that processes a
DataFrame.
    if results:
        summary_df = pd.DataFrame(results)

        # List of columns expected to contain multiline text.
        multiline_columns = ["Patient", "Interpretation", "Examination",
"Management", "Follow_up", "Summary"]

        # For each specified column, replace newline characters with a space.
        for col in multiline_columns:
            if col in summary_df.columns:
                summary_df[col] = summary_df[col].astype(str).str.replace("\n", "
", regex=False)

        # Save the DataFrame as a CSV file with every cell enclosed in quotes.
        # (This block remains unchanged as per the important caution.)
        summary_csv_path = os.path.join(SAVE_DIRECTORY, csv_filename)
        summary_df.to_csv(summary_csv_path, index=False,
quoting=csv.QUOTE_ALL)
        logging.info(f"Saved summary CSV to: {summary_csv_path}")

       # print("\nAll results:\n", results)
```

2.3 P3

```
import os
import sys
import json
import requests
import time
from datetime import datetime
import pandas as pd
import logging
import re
import csv
import textwrap

logging.basicConfig(level=logging.INFO)

################################################################################
##########
# 1) DEFINE FEATURES
################################################################################
##########
categorical_features = [
        "congestive_heart_failure",
        "cerebrovascular_disease",
        "dementia",
        "chronic_pulmonary_disease",
        "rheumatic_disease",
        "mild_liver_disease",
        "renal_disease",
        "malignant_cancer",
        "severe_liver_disease",
        "metastatic_solid_tumor",
        "emerg_ad",
        "first_icustay",
        "dm",
        "intubated_0h"
]

integer_features = [
        "gcs_0h",
        "heart_rate_0h",
        "resp_rate_0h",
        "sbp_0h",
        "dbp_0h",
        "mbp_0h",
        "spo2_0h"
]

numeric_features = [
        "admission_age",
        "temperature_0h",
```

```
    "weight_kg"
]

ordered_features = categorical_features + integer_features + numeric_features

################################################################################
##########
# 2) LOAD CSV
################################################################################
##########
def load_query_file(filepath):
    """Load semicolon-delimited CSV; ensure 'stay_id' is integer."""
    data = pd.read_csv(filepath, sep=';')
    data['stay_id'] = data['stay_id'].astype(int)
    return data

################################################################################
##########
# 3) FEATURE FORMATTING
################################################################################
##########
def format_feature_value(feature_name, value):
    if value is None:
        return "N/A"
    try:
        float_val = float(value)
    except ValueError:
        return str(value)
    if feature_name in categorical_features or feature_name in integer_features:
        return str(int(round(float_val)))
    elif feature_name in numeric_features:
        if feature_name == "admission_age":
            return f"{float_val:.0f}"
        else:
            return f"{float_val:.1f}"
    else:
        return str(value)

def get_feature_value(row, feature_name):
    try:
        return row[feature_name]
    except KeyError:
        logging.error(f"Feature '{feature_name}' not found.")
        return None

def get_feature_shap_value(row, feature_name):
    shap_col = f"{feature_name}_SHAP"
    try:
        return row[shap_col]
    except KeyError:
```

```
        logging.error(f"SHAP value for '{feature_name}' not found.")
        return None

################################################################################
##########
# 4) CALL AZURE OPENAI
################################################################################
##########
def call_azure_openai(api_endpoint, api_key, system_prompt, user_prompt,
timeout=360):
    headers = {
        "Content-Type": "application/json",
        "api-key": api_key
    }
    payload = {
        "messages": [
            {"role": "system", "content": system_prompt},
            {"role": "user", "content": user_prompt}
        ],
        "temperature": 0.6,
        "top_p": 0.9,
        "max_tokens": 4000
    }
    try:
        response = requests.post(api_endpoint, headers=headers, json=payload,
timeout=timeout)
        response.raise_for_status()
        data = response.json()
        llm_response_text = data["choices"][0]["message"]["content"]
        usage_info = data.get("usage", {})
        return llm_response_text, usage_info
    except requests.exceptions.RequestException as e:
        logging.error(f"Error calling Azure OpenAI: {e}")
        if e.response is not None:
            logging.error(f"Response text: {e.response.text}")
        sys.exit(1)

################################################################################
##########
# NEW HELPER FUNCTION: CALL AZURE OPENAI WITH MESSAGES LIST
################################################################################
##########
def call_azure_openai_messages(api_endpoint, api_key, messages, temperature,
top_p, max_tokens,
                               max_retries=3, default_wait=30, timeout=360):
    headers = {
        "Content-Type": "application/json",
        "api-key": api_key
    }
    payload = {
```

```
            "messages": messages,
            "temperature": temperature,
            "top_p": top_p,
            "max_tokens": max_tokens
        }
        retry_count = 0
        total_wait_time = 0
        while True:
            try:
                response = requests.post(api_endpoint, headers=headers,
json=payload, timeout=timeout)
                response.raise_for_status()
                data = response.json()
                llm_response_text = data["choices"][0]["message"]["content"]
                usage_info = data.get("usage", {})
                usage_info["waiting_time"] = total_wait_time
                return llm_response_text, usage_info
            except requests.exceptions.RequestException as e:
                if e.response is not None and e.response.status_code == 429:
                    try:
                        error_json = e.response.json()
                        error_message = error_json.get("error", {}).get("message",
"")
                        match = re.search(r"retry after (\d+) seconds",
error_message, re.IGNORECASE)
                        if match:
                            wait_seconds = int(match.group(1))
                        else:
                            wait_seconds = default_wait
                    except Exception:
                        wait_seconds = default_wait
                    logging.warning(f"Received 429. Retrying after {wait_seconds}
seconds.")
                    time.sleep(wait_seconds)
                    total_wait_time += wait_seconds
                    retry_count += 1
                    if retry_count > max_retries:
                        logging.error("Exceeded max retries.")
                        sys.exit(1)
                    continue
                else:
                    logging.error(f"Error in multi-turn: {e}")
                    if hasattr(e, "response") and e.response is not None:
                        logging.error(f"Response text: {e.response.text}")
                    sys.exit(1)

##########################################################################
##########
# 5) POST-PROCESSING
##########################################################################
```

```
##########
def postprocess_llm_response(response_text):
    return response_text.replace("**", "")

##########################################################################
##########
# 6) PREPARE PROMPT (SEPARATE POSITIVE/NEGATIVE SHAP), RETURN
LLM RESPONSE
##########################################################################
##########
def send_to_llm(row_data, api_endpoint, api_key, max_retries, default_wait,
                    temperature, top_p, max_tokens, timeout=360):
    predicted_risk = row_data["predicted_risk"]
    positive_features = []
    negative_features = []
    for feature in ordered_features:
        val = get_feature_value(row_data, feature)
        shap_val = get_feature_shap_value(row_data, feature)
        if shap_val is None:
            continue
        if shap_val >= 0:
            positive_features.append((feature, val, shap_val))
        else:
            negative_features.append((feature, val, shap_val))
    positive_features_sorted = sorted(positive_features, key=lambda x: x[2],
reverse=True)
    negative_features_sorted = sorted(negative_features, key=lambda x: x[2])

    positive_report_lines = [
        f"SHAP = {shap_val:.4f}, {feature} = {format_feature_value(feature,
val)}"
        for feature, val, shap_val in positive_features_sorted
    ]
    negative_report_lines = [
        f"SHAP = {shap_val:.4f}, {feature} = {format_feature_value(feature,
val)}"
        for feature, val, shap_val in negative_features_sorted
    ]

    section_positive = "2. Features with positive SHAP values\n" + "\n".join("  -"
+ line for line in positive_report_lines)
    section_negative = "3. Features with negative SHAP values\n" + "\n".join("  -"
+ line for line in negative_report_lines)
    sorted_feature_report = section_positive + "\n" + section_negative
    base_context = (f"1. Predicted ICU Mortality Risk: {predicted_risk:.1%}\n"
                    f"{sorted_feature_report}\n\n")

    conversation = []
    system_message = {
        "role": "system",
```

```
            "content":
"""
You are an expert intensivist providing evidence-based and clinically relevant
recommendations.
As you develop your response, internally perform a detailed, step-by-step chain-of-
thought analysis.
Avoid over-escalation, excessive testing, or unnecessary interventions unless clearly
justified.
Prioritize clinical severity, trends, and patient-specific context over generalized
protocols.
Use these structured guidelines to assess ICU mortality risks and provide targeted
recommendations.

Risk Factor Analysis
-Evaluate ICU mortality risk using key factors: age, SHAP values, GCS, vital signs,
blood pressure, chronic diseases, and intubation status.
-Highlight key variables with positive SHAP values over 0.1 and justify their impact.
-Distinguish between confirmed clinical data (vital signs, labs) and inferences
(diagnostic possibilities).
-Avoid overinterpreting mild abnormalities without context.

Recommendations
- Provide interventions consistent with vital signs and stability, following standard
ICU guidelines.
- Avoid unnecessary imaging or lab tests unless they change management decisions
- Do not propose broad-spectrum antibiotics or vasopressors without strong evidence
or suspicion.
- Justify each intervention and avoid unnecessary treatments.
- Cite relevant guidelines for lab or imaging studies.
- Base management decisions on severity and clinical trends.
- Distinguish between confirmed data and assumptions.
- Reference ARDSNet or Surviving Sepsis guidelines or other guidelines when
appropriate.
- Focus on clinically significant findings rather than mild abnormalities.
- When uncertain, clarify assumptions or note additional data needed.

Response Requirements
- Use plain text only, no Markdown.
- Avoid contractions in all statements.
- Summarize succinctly without adding new details.
- Provide step-by-step reasoning internally; do not reveal it.

Your recommendations must be clinically sound, justified, and selective—avoiding
reflexive protocols, unnecessary escalation, and excessive interventions.

"""
        }
        conversation.append(system_message)

        aggregated_usage = {"prompt_tokens": 0, "completion_tokens": 0,
```

"total_tokens": 0, "waiting_time": 0}

```
        # Turn 1
        user_message_turn1 = {
            "role": "user",
            "content": base_context + "[Interpretation of Risk Factors]\n" +
"""
[Instruction]

1. In Interpretating of Major Risk Factors
- Begin with variables displaying the highest SHAP values for strong clinical relevance, describe their impacts to predicted risk.
- Find the potential causes of instability or deterioration of these factors. Provide differential diagnoses rather than assuming a single cause.
- Focus on high-impact SHAP values (>0.1) with clear clinical relevance.
- Assess for any acute or urgent conditions.
- Consolidate features pointing to the same condition (e.g., hypotension-related parameters) to avoid redundancy.
- For hypoxemia (SpO2 < 92%), provide a differential (respiratory, cardiac, metabolic) rather than assigning a single cause without supporting data.

2. In Interaction and other risk factors
- Recognize interaction of these risk factors.
- Identify modifiable risk factors.
- Mentions clinical features with low SHAP value (<0.1) here only when they are clinically significant. Omit when clinically insignificantly.
- Features that are with normal clinical valuess or absent of chronic diseases with negative SHAP value can be treated as protectrive factors.
- Features that are with abnormal clinical values or presenece of chronic diseases with negative SHAP valuse should not be treated as protective factors.
- Can mention hemodynamic is stable, when sbp, mbp, dbp, and hr are all within normal range with negative SHAP values.
- Provide hemodynamic context by reminding to compare current blood pressure or heart rate to typical baseline values.
- Check any clues of critical ICU conditions only if relevant, including
- Sepsis/septic shock (high mortality, unstable vitals, organ dysfunction).
- Cardiogenic, hypovolemic, obstructive shock.
- Severe trauma, ARDS, respiratory failure, status asthmaticus/epilepticus.
- PE, ACS, arrhythmias, severe heart failure.
- AKI with electrolyte imbalances.
- Endocrine crises (DKA, HHS, thyroid storm, myxedema coma).
- GI bleeding, poisoning, anaphylaxis, MODS.

3. Additional Considerations
- first_icustay: 0 indicates a repeat ICU stay, 1 indicates the first ICU stay.
- For categorical features, 1 indicates presence, and 0 indicates absence.
- Treat mbp as MAP. If MAP is shown as higher than SBP or lower than DBP, remind the reader to reconcile this discrepancy.
- Do not comment on body weight if there is no reference to ideal body weight.
- If a risk factor is chronic and stable (e.g., cerebrovascular disease) but not actively
```

worsening, do not assume it adds acute risk. Just remind increased risk is ok.
- Avoid diagnosing severe conditions (e.g., ARDS) unless supported by more definitive measures (e.g., PaO_2/FiO_2 ratio). Can advice furtehr evaluation.
- Do not inflate minor variations (e.g., mild tachycardia) into major risks unless there is a proven worsening trend.
- Do not provide diagnostic, monitoring, or management advice at this section.

[Format of Responses]
1. Major Risk facotrs and Their Potential Causes
2. Interaction and Other Risk Factors
3. Quick conclusion

"""
}
conversation.append(user_message_turn1)
response_turn1, usage_info = call_azure_openai_messages(
api_endpoint, api_key, conversation, temperature, top_p, max_tokens,
max_retries=max_retries, default_wait=default_wait, timeout=timeout
)
conversation.append({"role": "assistant", "content": response_turn1})
for k in ["prompt_tokens", "completion_tokens", "total_tokens",
"waiting_time"]:
aggregated_usage[k] += usage_info.get(k, 0)
interpretation = response_turn1

# Turn 2
user_message_turn2 = {
"role": "user",
"content": "[Recommended Examinations]\n" +
"""
[Instruction]

1. Immediate Bedside Assessments
- Tailor to the patient's most significant risk factors or suspected causes of deterioration.
- Check for shock states (vital signs, point-of-care lactate, ECG), severe trauma/TBI (bleeding, FAST ultrasound), or respiratory failure (oxygen saturation, signs of distress).
- Evaluate for respiratory distress, status asthmaticus, or COPD exacerbation (airway patency, ABG if hypercapnia is likely).
- Assess neurological crises (focused neuro exam), cardiac emergencies (ECG, hemodynamic status), severe heart failure (JVD, hypotension), and other life-threatening conditions as needed.
2. Urgent Assessments
- Order blood tests (CBC, metabolic panel, lactate, infection markers, coagulation, cardiac enzymes) if significant risk factors suggest they are necessary.
- Order ABG if if there are any ventilatory concerns or if hypoxemia, hypercapnia, or metabolic derangements are suspected.
- Obtain imaging (chest X-ray, ultrasound, CT) only if strongly indicated by the clinical picture.

- Draw cultures if infection is highly suspected; check urinalysis if relevant.
3. Additional Monitoring
- Assess fluid responsiveness (IVC ultrasound, passive leg raises) if indicated by hemodynamic concerns.
- If fever plus respiratory/neurological symptoms develop, consider viral testing (e.g., COVID-19, influenza).
4. Advanced and Supplementary Assessments
- Perform additional or specialized tests (e.g., CT, MRI, specific biomarkers) only if driven by the patient's major risk factors or evolving ICU complications.
5. More Information
- Focus on diagnostic steps; avoid treatment recommendations in this stage.
- Match the urgency and frequency of testing to clinical severity.
- Cite relevant guidelines when applicable (e.g., Surviving Sepsis).
- Omit assessments not clearly warranted.
- CRP is preferred over procalcitonin in the acute stage.
- Recommend tests proportionate to presentation; avoid over-testing.
- Use ABG only if there is worsening hypoxemia or concern for CO_2 retention.
- Do not order brain imaging for chronic neurological conditions unless new deficits arise.
- Avoid reflex chest X-rays; order them only if a respiratory cause is likely.
- Limit monitoring frequency (e.g., hourly neuro checks) unless new or progressing symptoms appear.

[Format of Responses]
1. Immediate Bedside Assessment
2. Urgent Assessment
3. Additional Monitoring
4. Advanced and Supplemenatry Assessment
5. Quick Conclusion

"""
    }
    conversation.append(user_message_turn2)
    response_turn2, usage_info = call_azure_openai_messages(
        api_endpoint, api_key, conversation, temperature, top_p, max_tokens,
        max_retries=max_retries, default_wait=default_wait, timeout=timeout
)
    conversation.append({"role": "assistant", "content": response_turn2})
    for k in ["prompt_tokens", "completion_tokens", "total_tokens",
"waiting_time"]:
        aggregated_usage[k] += usage_info.get(k, 0)
    examination = response_turn2

    # Turn 3
    user_message_turn3 = {
        "role": "user",
        "content": "[Recommended Management]\n" +
"""
[Instruction]

1.Immediate Resuscitation and Management (First 6 hours)
- Stabilize any life-threatening conditions, guided by recognized ICU or sepsis protocols (e.g., Surviving Sepsis Campaign).
- Confirm hypotension before starting vasopressors: initiate norepinephrine (e.g., 0.05–0.1 mcg/kg/min) only if MAP < 65 mmHg or signs of hypoperfusion persist.
- Tailor fluid resuscitation to volume status. Avoid aggressive fluid boluses unless there is clear hypovolemia; consider albumin only if indicated (per Surviving Sepsis Guidelines).
- If inotropic support is needed for low cardiac output or persistent hypotension, specify agent (e.g., dobutamine at 2–5 mcg/kg/min) and clinical goals.
- Monitor for arrhythmias (e.g., atrial fibrillation); address rate or rhythm control with appropriate agents and dosing.
- Recommend empiric antibiotics only if strong suspicion of infection (e.g., fever, elevated WBC, clear source). Provide at least two sensible regimens tailored to likely pathogens (avoid broad-spectrum coverage unless multidrug-resistant organisms are a concern).
- If advanced airway management is required due to compromised oxygenation or airway protection, use techniques (e.g., video-assisted intubation) appropriate to patient risk.
- Target SpO_2 90–94%; do not intubate prematurely unless there is evidence of severe respiratory failure. Apply lung-protective ventilation (tidal volume ≈ 6 mL/kg IBW) if mechanically ventilated.

2. Acute Management (6–24 hours)
- Reassess volume status, wean vasopressors when MAP is stable ≥ 65 mmHg without organ hypoperfusion.
- Refine organ support (sedation, analgesia, nutrition) based on clinical trends and risk factor severity (e.g., significant SHAP values).
- Initiate renal replacement therapy (continuous or intermittent) only if specific triggers (e.g., refractory hyperkalemia, fluid overload) are met.
- Address ongoing arrhythmias or new-onset tachyarrhythmias with rate control or antiarrhythmic drugs, adjusting inotropes to optimize cardiac output.
- Escalate to advanced supports (e.g., ECMO) only for refractory hypoxemia or shock that fails conventional measures.
- Consider diuretic therapy (e.g., furosemide) if fluid overload is evident and the patient is not in end-stage renal disease.

3. ICU Supportive Cares
- Implement ICU-specific measures (infection control, VAP prevention, delirium protocols) tailored to the patient's condition and comorbidities.
- Use stress ulcer prophylaxis (e.g., PPIs) only for high-risk patients (prolonged mechanical ventilation, shock, coagulopathy).
- Apply light sedation strategies; avoid excessive benzodiazepine use.
- Provide early enteral nutrition (within 24–48 hours) but avoid overfeeding.
- Manage underlying chronic diseases carefully to prevent exacerbations.
- Maintain MAP targets and hemodynamic support in line with comorbid conditions (e.g., chronic hypertension).
- Reference PADIS guidelines for pain, agitation, and delirium management if relevant.

4. More information:
  - Avoid additional diagnostic test suggestions—strictly address management priorities and rationale.
  - Avoid long-term or post-ICU management here.
  - Please mention guideline's title if you are giving your recommendation according to it.
  - May mention the importance of multidisciplinary input when the patient's condition is not stable or the predicted risk of ICU mortality is high.

[Format of Responses]
1. Immediate Resuscitation and Management
2. Acute Management
3. ICU Supportive Care
4. Quick Conclusion

"""
        }
        conversation.append(user_message_turn3)
        response_turn3, usage_info = call_azure_openai_messages(
            api_endpoint, api_key, conversation, temperature, top_p, max_tokens,
            max_retries=max_retries, default_wait=default_wait, timeout=timeout
)
        conversation.append({"role": "assistant", "content": response_turn3})
        for k in ["prompt_tokens", "completion_tokens", "total_tokens",
"waiting_time"]:
            aggregated_usage[k] += usage_info.get(k, 0)
        management = response_turn3

        # Turn 4
        user_message_turn4 = {
            "role": "user",
            "content": "[Follow-up Plan]\n" +
"""
[Instruction]

1.In Continuous Monitoring or Hourly Check
- Monitor vital signs continuously if shock is present or imminent; include real-time arterial pressure if available.
- Check GCS hourly only if mental status is unstable or changing.
- Adjust monitoring intervals (e.g., q1H → q2H) if the patient shows significant improvement.

2. Acute Reassessment (Every 2–8 Hours)
- Perform focused physical exams, labs, or imaging updates guided by evolving clinical status (e.g., rising lactate, stable MAP ≥ 65 mmHg).
- Place arterial blood gases here, checking only if respiratory or metabolic issues change.
- Reassess sedation, analgesia, and delirium parameters (CAM-ICU or ICDSC).
- Taper vasopressors when clinical criteria (stable hemodynamics) are met.
- Modify frequency of reassessment based on improvement or deterioration.

3. Stability Assessment (Day-by-Day)
- Evaluate readiness for weaning (ventilator, vasopressors), organ support adjustments, and complication prevention.
- Increase assessment frequency if the patient's condition worsens.
- Seek multidisciplinary input (nutrition, rehab, specialty consults) as needed.

4. More Information
- Recommend new or additional tests only if the patient's condition or risk factors change (e.g., unexpected hypotension, rising lactate).
- Identify clear warning signs (e.g., altered mental status, sudden drop in BP, arrhythmias) that require earlier reassessment.
- Emphasize thorough handover from transferring or admitting teams to ensure continuity of care.
- Outline intervals for reassessment in low-risk scenarios (once per shift if stable).
- Avoid excessive ABGs or imaging unless clinical changes warrant them.
- Highlight key triggers for more frequent monitoring or additional tests (e.g., new fever, acute respiratory changes).

[Format of Responses]
1. Continuous Monitoring or Hourly Check
2. Acute Reassessment
3. Stability Follow-up

"""
```
        }
        conversation.append(user_message_turn4)
        response_turn4, usage_info = call_azure_openai_messages(
            api_endpoint, api_key, conversation, temperature, top_p, max_tokens,
            max_retries=max_retries, default_wait=default_wait, timeout=timeout
        )
        conversation.append({"role": "assistant", "content": response_turn4})
        for k in ["prompt_tokens", "completion_tokens", "total_tokens",
"waiting_time"]:
            aggregated_usage[k] += usage_info.get(k, 0)
        follow_up = response_turn4

        # Turn 5
        user_message_turn5 = {
            "role": "user",
            "content": "[Summary]\n" +
"""
```
[Instruction]
Provide a concise overview (100–150 words) summarizing the patient's risk factors, mortality risk, and the top three immediate actions.
Also include the reassessment schedule and any necessary consultations.
If mortality risk is over 20%, emphasize family discussion; if exceeding 60%, consider a time-limited trial.

[Format of Responses]

```
Startin paragraph.
Subheading with "Top 3 Priority Actions" in new line, then put every action in new
line.
Ending paragraph.

"""
        }
        conversation.append(user_message_turn5)
        response_turn5, usage_info = call_azure_openai_messages(
            api_endpoint, api_key, conversation, temperature, top_p, max_tokens,
            max_retries=max_retries, default_wait=default_wait, timeout=timeout
        )
        conversation.append({"role": "assistant", "content": response_turn5})
        for k in ["prompt_tokens", "completion_tokens", "total_tokens",
"waiting_time"]:
            aggregated_usage[k] += usage_info.get(k, 0)
        summary = response_turn5

        final_response = (
            "[Interpretation of Risk Factors]\n" + interpretation + "\n\n" +
            "[Recommended Examinations]\n" + examination + "\n\n" +
            "[Recommended Management]\n" + management + "\n\n" +
            "[Follow-up Plan]\n" + follow_up + "\n\n" +
            "[Summary]\n" + summary
        )
        final_response = postprocess_llm_response(final_response)
        return final_response, sorted_feature_report, aggregated_usage

##########################################################################
##########
# 6.1) EXTRACT SECTION
##########################################################################
##########
def extract_section(analysis_text, section_title):
        """
        Extract text from the LLM response following the given section title.
        The section title may be enclosed in square brackets or not.
        Returns the text until the next known section header or the end of text.
        """
        import re

        # Create a regex pattern to match the section title with or without brackets
        pattern = re.compile(
            r'(?:\[' + re.escape(section_title) + r'\]|' + re.escape(section_title) + r')\s*(.*)',
            re.IGNORECASE | re.DOTALL
        )
        match = pattern.search(analysis_text)
        if not match:
            return ""
        section_content = match.group(1)
```

```
    # List of known section headers that might follow
    possible_headers = [
        "Interpretation of Risk Factors",
        "Recommended Examinations",
        "Recommended Management",
        "Follow-up Plan",
        "Summary"
    ]

    # Identify the earliest occurrence of any other section header
    end_index = len(section_content)
    for header in possible_headers:
        if header.lower() == section_title.lower():
            continue
        header_pattern = re.compile(
            r'(?:\[' + re.escape(header) + r'\]|' + re.escape(header) + r')',
            re.IGNORECASE
        )
        header_match = header_pattern.search(section_content)
        if header_match:
            idx = header_match.start()
            if idx < end_index:
                end_index = idx
    return section_content[:end_index].strip()

################################################################################
##########
# 7) SAVE REPORTS TO TEXT FILE
################################################################################
##########
def save_llm_response_txt(response_text, stay_id, LLM_MODEL, PROMPT,
save_directory="C:\\workspace\\001_GAI\\LLM"):
    date_str = datetime.now().strftime("%Y%m%d")
    base_model = re.sub(r'\s+\d{4}-\d{2}-\d{2}$', '', LLM_MODEL)
    filename = f"R_{base_model}_{stay_id}_{date_str}_{PROMPT}.txt"
    filepath = os.path.join(save_directory, filename)

    # Wrap text so that no line exceeds 80 characters while preserving bullet or
numbered list markers
    wrapped_lines = []
    for line in response_text.splitlines():
        match = re.match(r'^(\s*([-*\d]+[\.\)]\s+))', line)
        if match:
            prefix = match.group(1)
            wrapped_line = textwrap.fill(line, width=80, initial_indent=prefix,
subsequent_indent=" " * len(prefix))
        else:
            wrapped_line = textwrap.fill(line, width=80)
        wrapped_lines.append(wrapped_line)
```

```
        wrapped_text = "\n".join(wrapped_lines)

        with open(filepath, 'w', encoding='utf-8') as file:
            file.write(wrapped_text)

##########################################################################
##########
# 8) DETERMINE RISK GROUP
##########################################################################
##########
def get_risk_group(predicted_risk):
    if predicted_risk < 0.1:
        return "l"
    elif predicted_risk < 0.2:
        return "m"
    else:
        return "h"

##########################################################################
##########
# 9) MAIN WORKFLOW
##########################################################################
##########
def complete_workflow(filepath, azure_api_key, azure_endpoint, save_directory,
                      LLM_MODEL, PROMPT, RAG, ML_MODEL,
max_retries, default_wait,
                      temperature, top_p, max_tokens, timeout, csv_file_path):
    data = load_query_file(filepath)
    summary_rows = []
    for idx, row in data.iterrows():
        record_start_time = time.time()
        stay_id = int(row["stay_id"])
        real_icu_mort = row.get("icu_mortality", "N/A")
        if real_icu_mort not in ["N/A", None]:
            try:
                real_icu_mort = str(int(round(float(real_icu_mort))))
            except Exception:
                pass

        if "predicted_risk" not in row:
            logging.error(f"No 'predicted_risk' for {stay_id}.")
            continue
        predicted_risk = row["predicted_risk"]
        risk_group = get_risk_group(predicted_risk)

        try:
            llm_time_start = time.time()
            llm_response, sorted_feature_report, usage_info = send_to_llm(
                row_data=row,
                api_endpoint=azure_endpoint,
```

```
                    api_key=azure_api_key,
                    max_retries=max_retries,
                    default_wait=default_wait,
                    temperature=temperature,
                    top_p=top_p,
                    max_tokens=max_tokens,
                    timeout=timeout
                )
                # Reconstruct the base context as patient information.
                patient_info = f"1. Predicted ICU Mortality Risk:
{predicted_risk:.1%}\n{sorted_feature_report}\n\n"

                llm_time_secs = time.time() - llm_time_start
                total_record_time_secs = time.time() - record_start_time
                llm_waiting_t = usage_info.get("waiting_time", 0)
                llm_response_t = llm_time_secs - llm_waiting_t
                python_time = total_record_time_secs - llm_time_secs
                total_time = llm_response_t + llm_waiting_t + python_time
                prompt_tokens = usage_info.get("prompt_tokens", 0)
                completion_tokens = usage_info.get("completion_tokens", 0)
                total_tokens = usage_info.get("total_tokens", 0)

                interpretation = extract_section(llm_response, "Interpretation of Risk
Factors")
                examination = extract_section(llm_response, "Recommended
Examinations")
                management = extract_section(llm_response, "Recommended
Management")
                follow_up = extract_section(llm_response, "Follow-up Plan")
                summary_sec = extract_section(llm_response, "Summary")

                full_response = (
                    f"Report Date: {datetime.now().strftime('%Y-%m-%d
%H:%M:%S')}\n\n"
                    "---Hidden from LLM---\n"
                    f"Stay_id: {stay_id}\n"
                    f"ML Model: {ML_MODEL}\n"
                    f"LLM model: {LLM_MODEL}\n"
                    f"Prompt: {PROMPT}\n"
                    f"RAG: {RAG}\n"
                    f"ICU Mortality: {real_icu_mort}\n\n"
                    "---Information for LLM---\n\n"
                    f"1. Predicted ICU Mortality Risk: {predicted_risk:.1%}\n" +
                    f"{sorted_feature_report}\n\n"
                    "--- Full LLM Response ---\n\n"
                    f"{llm_response}\n"
                )

                save_llm_response_txt(full_response, stay_id, LLM_MODEL,
PROMPT, save_directory)
```

```
                summary_rows.append({
                    "stay_id": stay_id,
                    "ICU_mortality": real_icu_mort,
                    "Predicted_risk(%)": f"{predicted_risk * 100:.1f}",
                    "risk_group": risk_group,
                    "ML_model": ML_MODEL,
                    "LLM_model": LLM_MODEL,
                    "Prompt": PROMPT,
                    "RAG": RAG,
                    "prompt_tokens": prompt_tokens,
                    "completion_tokens": completion_tokens,
                    "total_tokens": total_tokens,
                    "LLM_response_t(s)": int(llm_response_t),
                    "LLM_waiting_t(s)": int(llm_waiting_t),
                    "Python_time(s)": int(python_time),
                    "Total_time(s)": int(total_time),
                    "Patient": patient_info,                # <-- New column added here
                    "Interpretation": interpretation,
                    "Examination": examination,
                    "Management": management,
                    "Follow_up": follow_up,
                    "Summary": summary_sec,
                })

                # Incremental saving after processing each case (Task 2)
                try:
                    temp_df = pd.DataFrame(summary_rows)
                    multiline_columns = ["Patient", "Interpretation", "Examination",
"Management", "Follow_up", "Summary"]
                    for col in multiline_columns:
                        if col in temp_df.columns:
                            temp_df[col] = temp_df[col].str.replace("\n", " ",
regex=False)
                    temp_df.to_csv(csv_file_path, index=False,
quoting=csv.QUOTE_ALL)
                    logging.info(f"Incrementally saved summary CSV to:
{csv_file_path}")
                except Exception as e:
                    logging.error(f"Error saving incremental CSV for stay_id
{stay_id}: {str(e)}")

            except Exception as e:
                logging.error(f"Error for stay_id {stay_id}: {str(e)}")

        return summary_rows

##########################################################################
##########
# 10) EXAMPLE USAGE
```

```
##########################################################################
##########
if __name__ == "__main__":
    FILEPATH = "C:\\workspace\\001_GAI\\LLM\\adm\\v_llm_group.csv"
    AZURE_ENDPOINT = "YOUR AZURE URI"
    AZURE_API_KEY = "YOUR KEY"
    ML_MODEL = "XGBoost_20250207"
    LLM_MODEL = "GPT 4o 2024-11-20"
    PROMPT = "P3"
    RAG = "None"

    MAX_RETRIES = 3
    DEFAULT_WAIT_SECONDS = 30
    TEMPERATURE = 0.6
    TOP_P = 0.9
    MAX_TOKENS = 6000
    TIMEOUT = 360       # need to set other timeouts

    save_directory="C:\\workspace\\001_GAI\\LLM"

    # Task 1: Check if the output CSV file exists and is accessible.
    date_str = datetime.now().strftime("%Y%m%d")
    csv_name = f"R_{LLM_MODEL}_{date_str}_{PROMPT}.csv"
    summary_csv_path = os.path.join(save_directory, csv_name)
    if os.path.exists(summary_csv_path):
        try:
            with open(summary_csv_path, 'a'):
                pass
        except Exception as e:
            logging.error("CSV file exists and may be open in another program.
Please save, close, or rename the file.")
            sys.exit(1)
        logging.info("CSV file exists and is accessible. Consider renaming the file
if you want to preserve the existing data.")
    else:
        logging.info("CSV file does not exist. Proceeding with evaluation.")

    results = complete_workflow(
        filepath=FILEPATH,
        azure_api_key=AZURE_API_KEY,
        azure_endpoint=AZURE_ENDPOINT,
        save_directory=save_directory,
        LLM_MODEL=LLM_MODEL,
        PROMPT=PROMPT,
        RAG=RAG,
        ML_MODEL=ML_MODEL,
        max_retries=MAX_RETRIES,
        default_wait=DEFAULT_WAIT_SECONDS,
        temperature=TEMPERATURE,
        top_p=TOP_P,
```

```
            max_tokens=MAX_TOKENS,
            timeout=TIMEOUT,
            csv_file_path=summary_csv_path
        )

        # The following code for processing the DataFrame and saving the final CSV
remains unchanged.
        if results:
            summary_df = pd.DataFrame(results)
            # Replace newline characters with a space in specified multiline columns
            multiline_columns = ["Patient", "Interpretation", "Examination",
"Management", "Follow_up", "Summary"]
            for col in multiline_columns:
                if col in summary_df.columns:
                    summary_df[col] = summary_df[col].str.replace("\n", " ",
regex=False)
            # Save with all cells quoted to prevent misinterpretation of multiline cells.
            summary_df.to_csv(summary_csv_path, index=False,
quoting=csv.QUOTE_ALL)
            logging.info(f"Saved summary CSV to: {summary_csv_path}")

            # print("\nAll results:\n", results)
```

3. Example responses of P1, P2, and P3

3.1 Response of P1

---Information for LLM---
1. Predicted ICU Mortality Risk: 24.6%
2. Features with positive SHAP values
    -SHAP = 0.7932, resp_rate_0h = 31
    -SHAP = 0.4383, spo2_0h = 92
    -SHAP = 0.3296, malignant_cancer = 1
    -SHAP = 0.3044, heart_rate_0h = 107
    -SHAP = 0.1885, admission_age = 79
    -SHAP = 0.0929, gcs_0h = 15
    -SHAP = 0.0609, emerg_ad = 1
    -SHAP = 0.0566, mbp_0h = 102
    -SHAP = 0.0411, sbp_0h = 113
    -SHAP = 0.0289, dbp_0h = 99
    -SHAP = 0.0178, dementia = 0
3. Features with negative SHAP values
    -SHAP = -0.3999, intubated_0h = 0
    -SHAP = -0.0927, mild_liver_disease = 0
    -SHAP = -0.0652, weight_kg = 63.2
    -SHAP = -0.0515, renal_disease = 0
    -SHAP = -0.0496, temperature_0h = 36.6
    -SHAP = -0.0438, metastatic_solid_tumor = 0
    -SHAP = -0.0351, severe_liver_disease = 0
    -SHAP = -0.0271, congestive_heart_failure = 0
    -SHAP = -0.0209, chronic_pulmonary_disease = 0
    -SHAP = -0.0203, cerebrovascular_disease = 0
    -SHAP = -0.0172, first_icustay = 1
    -SHAP = -0.0043, dm = 0
    -SHAP = -0.0010, rheumatic_disease = 0

--- Full LLM Response ---
[Interpretation of Risk Factors]
The predicted ICU mortality risk for this patient is 24.6%, which indicates a moderate risk profile. The most significant positive contributors to this risk include an elevated respiratory rate (31 breaths per minute), decreased oxygen saturation (SpO2 of 92%), the presence of malignant cancer, tachycardia (heart rate of 107 beats per minute), advanced age (79 years), and an emergency admission. Other factors such as a Glasgow Coma Scale (GCS) score of 15 and blood pressure values (mean arterial pressure, systolic, and diastolic) contribute positively but less significantly. Negative SHAP values indicate protective or less concerning factors, including the absence of intubation, normal liver function, normal weight (63.2 kg), absence of renal disease, normal temperature (36.6°C), and the absence of significant comorbidities such as metastatic cancer, heart

failure, or pulmonary disease. The patient is not experiencing their first ICU stay, which has a minimal negative contribution to mortality risk.

[Recommended Examinations]
1. Perform arterial blood gas (ABG) analysis to confirm the degree of hypoxemia and assess for hypercapnia or acidosis, given the elevated respiratory rate and SpO2 of 92%.
2. Conduct a chest X-ray or CT scan of the chest if indicated, to identify potential causes of respiratory compromise such as infection, pulmonary edema, or malignancy-related complications.
3. Perform routine laboratory tests, including a complete blood count (CBC), basic metabolic panel, and lactate levels, to assess for infection, metabolic derangements, or tissue hypoperfusion.
4. Evaluate cardiac function with an electrocardiogram (ECG) and consider echocardiography if arrhythmias or cardiac dysfunction are suspected, given the tachycardia and advanced age.
5. Review oncologic history and perform cancer staging if not recently completed, as malignant cancer is a significant risk factor.

[Recommended Management]
1. Optimize oxygenation: Administer supplemental oxygen to maintain SpO2 levels above 94%, and consider non-invasive ventilation if hypoxemia persists despite oxygen therapy.
2. Monitor respiratory status closely for signs of worsening, and intubation should be considered if there is evidence of respiratory failure or fatigue.
3. Manage tachycardia: Ensure adequate fluid resuscitation if volume depletion is suspected, and assess for pain, anxiety, or sepsis as potential contributors. Beta-blockers or other rate-controlling medications should be used cautiously and only if indicated.
4. Address oncologic needs: Collaborate with oncology to ensure that the cancer is managed appropriately, including palliative care measures if applicable.
5. Maintain hemodynamic stability: Monitor blood pressure closely, and initiate vasoactive medications only if signs of hypotension or shock develop.
6. Implement infection prevention strategies, given the patient's advanced age and potential for immunosuppression due to malignancy.
7. Consider early palliative care consultation to address goals of care and quality of life, especially given the patient's age and cancer diagnosis.

[Follow-up Plan]
1. Reassess oxygenation, respiratory rate, and hemodynamic parameters every 1 to 2 hours initially, adjusting interventions as needed.
2. Repeat ABG and lactate measurements within 6 hours to evaluate the response to therapy and detect any worsening hypoxemia or metabolic derangements.
3. Monitor for complications related to malignancy or ICU stay, such as infections, thromboembolism, or delirium, and address promptly.
4. Engage the patient and family in discussions about prognosis and treatment

priorities, ensuring alignment of care with their preferences.

[Summary]
This patient has a moderate risk of ICU mortality, primarily driven by respiratory compromise (elevated respiratory rate and low oxygen saturation), malignant cancer, advanced age, and tachycardia. Immediate priorities include optimizing oxygenation, monitoring for respiratory failure, and ensuring hemodynamic stability. A comprehensive evaluation of the underlying causes of these abnormalities, particularly malignancy-related complications, is essential. Palliative care involvement should be considered to address quality-of-life concerns and align treatment with the patient's goals. Regular reassessment and close monitoring are critical to guide ongoing management.

3.2 Response of P2

---Information for LLM---
1. Predicted ICU Mortality Risk: 24.6%
2. Features with positive SHAP values
    -SHAP = 0.7932, resp_rate_0h = 31
    -SHAP = 0.4383, spo2_0h = 92
    -SHAP = 0.3296, malignant_cancer = 1
    -SHAP = 0.3044, heart_rate_0h = 107
    -SHAP = 0.1885, admission_age = 79
    -SHAP = 0.0929, gcs_0h = 15
    -SHAP = 0.0609, emerg_ad = 1
    -SHAP = 0.0566, mbp_0h = 102
    -SHAP = 0.0411, sbp_0h = 113
    -SHAP = 0.0289, dbp_0h = 99
    -SHAP = 0.0178, dementia = 0
3. Features with negative SHAP values
    -SHAP = -0.3999, intubated_0h = 0
    -SHAP = -0.0927, mild_liver_disease = 0
    -SHAP = -0.0652, weight_kg = 63.2
    -SHAP = -0.0515, renal_disease = 0
    -SHAP = -0.0496, temperature_0h = 36.6
    -SHAP = -0.0438, metastatic_solid_tumor = 0
    -SHAP = -0.0351, severe_liver_disease = 0
    -SHAP = -0.0271, congestive_heart_failure = 0
    -SHAP = -0.0209, chronic_pulmonary_disease = 0
    -SHAP = -0.0203, cerebrovascular_disease = 0
    -SHAP = -0.0172, first_icustay = 1
    -SHAP = -0.0043, dm = 0
    -SHAP = -0.0010, rheumatic_disease = 0

--- Full LLM Response ---
[Interpretation of Risk Factors]
1. Major Risk Factors and Their Potential Causes
   The major risk factors contributing to the patient's ICU mortality risk include:
   - Respiratory Rate (31, SHAP = 0.7932): Indicates potential respiratory distress or compensation for hypoxemia, which is concerning given SpO2 of 92%. This requires close monitoring for respiratory failure.
   - SpO2 (92, SHAP = 0.4383): Suggests mild hypoxemia, potentially linked to underlying respiratory or cardiovascular compromise.
   - Malignant Cancer (SHAP = 0.3296): A significant chronic condition that increases mortality risk due to potential immunosuppression, cachexia, or treatment-related complications.
   - Heart Rate (107, SHAP = 0.3044): Reflects tachycardia, possibly secondary to hypoxemia, pain, or an underlying condition such as sepsis.
   - Admission Age (79, SHAP = 0.1885): Advanced age is an independent risk factor for ICU mortality, often associated with reduced physiological reserve

and comorbidities.

2. Interactions and Other Risk Factors
The mean blood pressure (102 mmHg) is within an acceptable range, and GCS of 15 indicates preserved neurological status, which are reassuring findings. The emergency admission (SHAP = 0.0609) reflects a potentially acute condition but does not independently elevate risk significantly.

3. Quick Conclusion
The patient's ICU mortality risk is influenced by respiratory compromise, advanced age, malignancy, and tachycardia. These findings necessitate careful monitoring and targeted evaluation of respiratory and cardiovascular status.

[Recommended Examinations]
1. Immediate Bedside Assessment
- Perform a focused respiratory examination, including auscultation and assessment of work of breathing.
- Initiate continuous pulse oximetry and capnography to monitor oxygenation and ventilation.
- Conduct a point-of-care arterial blood gas (ABG) to assess PaO2, PaCO2, and acid-base status.

2. Urgent Assessment
- Obtain a chest X-ray to evaluate for potential causes of hypoxemia, such as pulmonary edema, infection, or malignancy-related complications.
- Perform a 12-lead ECG to assess for cardiac ischemia or arrhythmias contributing to tachycardia.
- Check laboratory tests: CBC (to assess for anemia or infection), basic metabolic panel (to evaluate for electrolyte imbalances), and lactate (to rule out tissue hypoperfusion).

3. Secondary and Follow-up Assessment
- If hypoxemia worsens or persists, consider CT chest for a detailed evaluation of pulmonary pathology.
- Assess D-dimer and consider CT pulmonary angiography if pulmonary embolism is suspected.

4. Additional Monitoring
- Monitor respiratory rate, heart rate, and blood pressure hourly.
- Reassess GCS periodically to detect any evolving neurological compromise.

5. Quick Conclusion
The focus should be on identifying and addressing the underlying cause of respiratory distress and hypoxemia. Initial bedside and urgent assessments are crucial for guiding management.

[Recommended Management]
1. Immediate Resuscitation and Management
   - Administer supplemental oxygen to maintain SpO2 ≥94%, using a nasal cannula or mask as appropriate. Escalate to high-flow oxygen or noninvasive ventilation if hypoxemia persists.
   - If ABG shows significant hypoxemia or hypercapnia, prepare for possible intubation following ARDSNet guidelines.
   - Initiate IV fluids cautiously if hypoperfusion is suspected but avoid fluid overload due to potential malignancy-related complications.

2. Acute Management
   - If infection is suspected, obtain blood and sputum cultures and initiate empiric antibiotics per local guidelines.
   - Monitor for signs of acute respiratory distress syndrome (ARDS) and manage accordingly, including lung-protective ventilation if intubated.

3. ICU Supportive Care
   - Implement VTE prophylaxis with low-molecular-weight heparin unless contraindicated.
   - Provide stress ulcer prophylaxis with a proton pump inhibitor if risk factors for GI bleeding are present.
   - Address potential delirium with non-pharmacologic measures and daily sedation interruptions if sedated.

4. Quick Conclusion
   Management should prioritize stabilization of respiratory status, identification of potential infection, and prevention of ICU complications.

[Follow-up Plan]
1. Continuous Monitoring or Hourly Check
   - Monitor SpO2, respiratory rate, heart rate, blood pressure, and GCS hourly.
   - Reassess ABG if oxygenation deteriorates or clinical condition changes.

2. Acute Reassessment
   - Perform focused respiratory and cardiac examinations every 2–4 hours.
   - Reassess imaging and lab results to guide further interventions.

3. Stability Follow-up
   - Evaluate readiness for weaning from oxygen support daily.
   - Monitor for complications such as VAP or thromboembolism.

[Summary]
This 79-year-old patient presents with a predicted ICU mortality risk of 24.6%, driven by respiratory distress (RR 31, SpO2 92%), malignancy, tachycardia, and advanced age. Initial stabilization and diagnostic evaluation should focus on respiratory and cardiovascular systems, with particular attention to hypoxemia and potential malignancy-related complications.

Top 3 Priority Actions:
- Administer supplemental oxygen to maintain SpO2 ≥94% and monitor respiratory status closely.
- Obtain ABG, chest X-ray, ECG, and relevant labs to evaluate hypoxemia and tachycardia.
- Initiate infection workup if clinically indicated and provide supportive care to prevent ICU complications.

Given the elevated mortality risk, frequent reassessment and multidisciplinary input are critical to optimize outcomes. Consider discussing prognosis with the family to align goals of care.

3.3 Response of P3

---Information for LLM---
1. Predicted ICU Mortality Risk: 24.6%
2. Features with positive SHAP values
    -SHAP = 0.7932, resp_rate_0h = 31
    -SHAP = 0.4383, spo2_0h = 92
    -SHAP = 0.3296, malignant_cancer = 1
    -SHAP = 0.3044, heart_rate_0h = 107
    -SHAP = 0.1885, admission_age = 79
    -SHAP = 0.0929, gcs_0h = 15
    -SHAP = 0.0609, emerg_ad = 1
    -SHAP = 0.0566, mbp_0h = 102
    -SHAP = 0.0411, sbp_0h = 113
    -SHAP = 0.0289, dbp_0h = 99
    -SHAP = 0.0178, dementia = 0
3. Features with negative SHAP values
    -SHAP = -0.3999, intubated_0h = 0
    -SHAP = -0.0927, mild_liver_disease = 0
    -SHAP = -0.0652, weight_kg = 63.2
    -SHAP = -0.0515, renal_disease = 0
    -SHAP = -0.0496, temperature_0h = 36.6
    -SHAP = -0.0438, metastatic_solid_tumor = 0
    -SHAP = -0.0351, severe_liver_disease = 0
    -SHAP = -0.0271, congestive_heart_failure = 0
    -SHAP = -0.0209, chronic_pulmonary_disease = 0
    -SHAP = -0.0203, cerebrovascular_disease = 0
    -SHAP = -0.0172, first_icustay = 1
    -SHAP = -0.0043, dm = 0
    -SHAP = -0.0010, rheumatic_disease = 0

--- Full LLM Response ---
[Interpretation of Risk Factors]
1. Major Risk Factors and Their Potential Causes
    - resp_rate_0h = 31 (SHAP = 0.7932): The elevated respiratory rate suggests potential respiratory distress or compensatory mechanisms. Causes could include hypoxemia, metabolic acidosis, pulmonary embolism, pneumonia, heart failure, or sepsis. Further assessment of arterial blood gases, chest imaging, and clinical context is needed to narrow the differential diagnosis.
    - spo2_0h = 92 (SHAP = 0.4383): This borderline hypoxemia could indicate early respiratory compromise. Potential causes include chronic lung disease (e.g., COPD), pneumonia, pulmonary edema, or cardiac dysfunction. An acute process like pulmonary embolism or ARDS should also be considered if clinical deterioration is suspected.
    - malignant_cancer = 1 (SHAP = 0.3296): The presence of malignant cancer increases overall mortality risk, likely due to associated complications such as immunosuppression, thromboembolic events, or metastatic disease.

Assess for acute complications such as infection, tumor lysis syndrome, or organ involvement.
- heart_rate_0h = 107 (SHAP = 0.3044): Mild tachycardia could be a response to pain, hypoxemia, fever, anemia, or early sepsis. Without hypotension or other signs of instability, it is less likely to represent a critical condition like shock.
- admission_age = 79 (SHAP = 0.1885): Advanced age is a significant risk factor for ICU mortality, often due to reduced physiological reserve and comorbidities. This factor alone does not indicate acute instability but requires careful monitoring for deterioration.
- gcs_0h = 15 (SHAP = 0.0929): A normal Glasgow Coma Scale (GCS) score is reassuring and suggests no acute neurological compromise.

2. Interactions and Other Factors
- The combination of elevated respiratory rate, borderline hypoxemia, and mild tachycardia suggests a potential respiratory or metabolic issue. This could be due to underlying cancer-related complications or an acute process like infection or pulmonary embolism.
- Hemodynamic parameters (sbp = 113, mbp = 102, dbp = 99) are stable and do not indicate shock or critical hypotension. The heart rate of 107, while slightly elevated, does not reach concerning thresholds without additional evidence of instability.
- Negative SHAP values for chronic diseases (e.g., dementia, renal disease, liver disease) are protective, indicating these conditions are not contributing significantly to mortality risk.
- The absence of intubation (SHAP = -0.3999) is protective, as it suggests the patient is not in acute respiratory failure requiring mechanical ventilation.

3. Quick Conclusion
The patient's primary risks stem from respiratory parameters (elevated respiratory rate and borderline hypoxemia), advanced age, and malignant cancer. These factors necessitate close monitoring for respiratory or metabolic decompensation. Hemodynamics are stable, and there are no immediate signs of critical ICU conditions such as sepsis or shock. Further evaluation should focus on identifying and addressing the cause of respiratory compromise while maintaining supportive care.

[Recommended Examinations]
1. Immediate Bedside Assessment
- Perform a focused respiratory assessment, including signs of respiratory distress (use of accessory muscles, cyanosis, audible wheezing, or crackles).
- Measure oxygen saturation continuously and ensure supplemental oxygen is available if needed to maintain SpO2 > 92%.
- Check for signs of shock (heart rate, blood pressure trends, capillary refill, and skin perfusion).
- Perform a focused cardiovascular exam for any signs of heart failure (jugular

venous distension, peripheral edema, pulmonary crackles).
- Perform a focused neurological exam to confirm GCS of 15 and assess for any subtle changes in mental status.
- Consider point-of-care ultrasound (POCUS) to assess for lung pathology (e.g., pleural effusion, consolidation) and IVC collapsibility if fluid status is a concern.

2. Urgent Assessment
- Order arterial blood gas (ABG) to assess for hypoxemia, hypercapnia, or metabolic derangements given the elevated respiratory rate and borderline SpO2.
- Obtain a chest X-ray to evaluate for potential respiratory causes such as pneumonia, pulmonary edema, or pleural effusion.
- Perform blood tests: CBC (to evaluate for anemia or infection), metabolic panel (electrolytes, renal function), and CRP (to assess for inflammation or infection).
- Consider measuring lactate to rule out occult tissue hypoperfusion, especially in the context of tachycardia and cancer-related complications.
- Draw blood cultures if infection is suspected based on clinical findings or elevated inflammatory markers.

3. Additional Monitoring
- Monitor respiratory status closely with continuous pulse oximetry and repeat ABG if respiratory parameters worsen.
- Monitor hemodynamics, including heart rate and blood pressure trends, to ensure stability.
- If there is any development of fever or worsening respiratory symptoms, consider viral testing (e.g., COVID-19, influenza).

4. Advanced and Supplementary Assessment
- Consider CT chest with contrast if pulmonary embolism or malignancy-related complications (e.g., lymphangitic carcinomatosis) are suspected based on clinical findings or chest X-ray results.
- Perform a POCUS-guided fluid responsiveness assessment if there is concern for hypovolemia or fluid overload.

5. Quick Conclusion
The immediate focus should be on assessing respiratory status (ABG, chest X-ray) and ruling out infection or malignancy-related complications (CBC, CRP, blood cultures). Hemodynamic monitoring and point-of-care ultrasound can assist in evaluating for shock or fluid status. Advanced imaging should be reserved for cases where initial assessments suggest a more specific diagnosis, such as pulmonary embolism. Ensure interventions are proportional to clinical findings.

[Recommended Management]
1. Immediate Resuscitation and Management

- Oxygen Support: Administer supplemental oxygen to maintain SpO_2 between 90–94%. Avoid intubation unless there is evidence of severe respiratory failure or worsening hypoxemia.
- Fluid Management: Assess volume status using dynamic measures (e.g., IVC ultrasound, passive leg raise). Administer a conservative fluid bolus (e.g., 250–500 mL crystalloid) only if hypovolemia is confirmed. Avoid excessive fluids given the absence of hypotension or signs of shock.
- Antibiotics: If there is clinical suspicion of infection (e.g., fever, elevated inflammatory markers, or suggestive imaging findings), initiate empiric antibiotics targeting likely pathogens. For example:
- Piperacillin-tazobactam (3.375 g IV q6h) for broad coverage of respiratory or abdominal infections.
- Ceftriaxone (2 g IV q24h) plus azithromycin (500 mg IV q24h) if a community-acquired pneumonia etiology is suspected.
- Hemodynamic Monitoring: Continue to monitor MAP, heart rate, and signs of organ perfusion. No vasopressors are indicated at this time given stable hemodynamics (MAP = 102 mmHg).

2. Acute Management
   - Reassess Oxygenation: Perform serial ABG or pulse oximetry to monitor for worsening respiratory status. Consider escalation to non-invasive ventilation (e.g., BiPAP) if hypoxemia persists despite supplemental oxygen.
   - Adjust Antibiotics: Refine antibiotic therapy based on culture results or clinical response. De-escalate to narrower agents if infection is confirmed and sensitivities are available.
   - Fluid Management: Reassess fluid balance and avoid further fluid administration unless new evidence of hypovolemia arises. Consider diuretics (e.g., furosemide 20–40 mg IV) if signs of fluid overload (e.g., pulmonary congestion) develop.
   - Cancer-Related Complications: Monitor for cancer-related emergencies such as tumor lysis syndrome or thromboembolism. If a pulmonary embolism is suspected and confirmed, initiate anticoagulation (e.g., enoxaparin 1 mg/kg subcutaneously q12h).

3. ICU Supportive Care
   - Infection Control: Implement strict infection control measures, including VAP prevention if intubation becomes necessary.
   - Delirium Prevention: Follow PADIS guidelines by using light sedation protocols and early mobilization to minimize delirium risk.
   - Stress Ulcer Prophylaxis: Avoid routine stress ulcer prophylaxis unless there are specific risk factors (e.g., coagulopathy, mechanical ventilation > 48 hours).
   - Nutrition: Initiate early enteral nutrition within 24–48 hours if the patient is stable and able to tolerate it. Avoid overfeeding.
   - Pain and Agitation: Use non-opioid analgesics where possible and avoid benzodiazepines unless absolutely necessary.

4. Quick Conclusion
The immediate focus should be on maintaining oxygenation, monitoring for respiratory or infectious deterioration, and addressing potential cancer-related complications. Hemodynamics are stable, so fluids and vasopressors are not currently required. Antibiotic therapy should be initiated only if infection is strongly suspected. ICU supportive care should prioritize infection prevention, delirium protocols, and early nutrition. Multidisciplinary input, including oncology, may be beneficial given the presence of malignant cancer.

[Follow-up Plan]
1. Continuous Monitoring or Hourly Check
- Monitor vital signs continuously, focusing on respiratory rate, SpO_2, heart rate, and blood pressure, given the elevated respiratory rate and borderline hypoxemia.
- Hourly checks of GCS are not required as the patient currently has a stable score of 15. Increase frequency only if there is a change in mental status.
- Ensure continuous pulse oximetry to detect any worsening hypoxemia.

2. Acute Reassessment
- Reassess respiratory status every 4–6 hours, including focused physical exams and repeat ABG if there is worsening oxygenation or signs of respiratory fatigue.
- Monitor inflammatory markers (e.g., CRP) and complete blood count every 6–8 hours to evaluate for infection trends if antibiotics are initiated.
- Reassess fluid status and hemodynamics every 4–6 hours using physical exam findings and dynamic measures (e.g., IVC ultrasound) to guide fluid management.
- If antibiotics are started, evaluate clinical response and consider de-escalation based on culture results and clinical improvement.
- Assess sedation, analgesia, and delirium parameters (e.g., CAM-ICU) every 8 hours or more frequently if new symptoms arise.

3. Stability Follow-up
- Perform daily evaluations for readiness to wean oxygen support if respiratory status improves. Adjust supplemental oxygen to maintain SpO_2 in the target range (90–94%).
- Monitor for potential cancer-related complications (e.g., thromboembolism, tumor lysis syndrome) daily and involve oncology for further input as needed.
- Reassess the need for ICU-level care daily. If the patient stabilizes (e.g., improved respiratory rate and SpO_2), consider transitioning to a lower level of care.
- Engage multidisciplinary teams (nutrition, physical therapy, oncology) to optimize long-term management and recovery.

Quick Conclusion
The follow-up plan should focus on continuous respiratory monitoring and

periodic reassessment of infection, hemodynamics, and organ function. Adjust monitoring intervals based on clinical trends, with more frequent assessments if deterioration occurs. Multidisciplinary input is recommended to address cancer-related complications and optimize recovery. Avoid unnecessary tests or imaging unless there is a clinical change.

[Summary]
This 79-year-old patient has a predicted ICU mortality risk of 24.6%, driven by advanced age, malignant cancer, elevated respiratory rate (31), borderline hypoxemia (SpO_2 92%), and mild tachycardia (HR 107). Hemodynamics are stable, and GCS is 15, with no immediate evidence of shock or critical organ failure. The primary focus is on addressing respiratory compromise and monitoring for infection or cancer-related complications. Family discussion regarding prognosis and goals of care is recommended given the elevated mortality risk.

Top 3 Priority Actions
- Administer supplemental oxygen to maintain SpO_2 between 90–94% and monitor for respiratory deterioration.
- Perform ABG and chest X-ray to evaluate respiratory compromise and identify potential causes such as pneumonia or pulmonary embolism.
- Initiate empiric antibiotics if infection is strongly suspected, tailoring therapy to likely pathogens.

Reassess respiratory, hemodynamic, and infection status every 4–6 hours. Engage oncology and consider multidisciplinary input for cancer-related complications. Focus on close monitoring and supportive care to guide further management.

4. Scoring criteria for clinical AI response evaluation

Part 1: Evaluation of the Interpretation of Risk Factors

1. Item 1.1 Integration – Clinical Relevance

Score	Description
0	Key risk factors are omitted and there is no integration of SHAP values with the clinical context.
1	Risk factors are mentioned in a bare manner without any contextual linkage or reference to SHAP values.
2	Basic identification of risk factors is provided, but without clear reference to SHAP values or their clinical impact.
3	A clear explanation is provided that references SHAP values (though without detailing the specific numerical impact), linking risk factors to the patient's condition.
4	A detailed integration is achieved by explicitly incorporating numerical SHAP values and thoroughly correlating them with the patient's clinical information.

2. Item 1.2 Integration – Clarity of Information

Score	Description
0	The interpretation is disorganized and unclear, lacking a coherent structure or logical flow.
1	The interpretation is partially clear with noticeable ambiguities that impede full understanding of the risk factors.
2	The interpretation is generally clear but only provides a basic structure without exploring potential causes of the risk factors.
3	The interpretation is clear and organized, with a logical flow and mention of at least one potential cause of the risk factors.
4	The interpretation is exceptionally clear and well-organized, seamlessly presenting multiple distinct potential causes of the risk factors.

3. Item 1.3 Mastery – Balance and Bias Assessment

Score	Description
0	The response considers only risk-enhancing factors without mentioning any mitigating or protective factors.
1	The response minimally addresses mitigating factors, with a predominant focus on risk factors and only superficial mention of protections.
2	The response offers an overall balanced view but does not adequately emphasize protective factors, resulting in a somewhat uneven interpretation.
3	The response is balanced, addressing both risk-enhancing and mitigating factors, though the emphasis on protective factors is only mild.
4	The response is well-balanced, providing a comprehensive evaluation that equally emphasizes both risk-enhancing and protective factors.

4. Item 1.4 Mastery – Uncertainty Acknowledgment

Score	Description
0	No acknowledgment of uncertainty, limitations, or assumptions is provided.
1	Minimal acknowledgment of uncertainty is present, but it lacks detail and does not discuss how risk factors might interact.
2	A basic acknowledgment of uncertainty is made with a brief mention of limitations or assumptions, yet it lacks depth.
3	A clear acknowledgment of uncertainty is provided, detailing potential interactions of risk factors and limitations in the interpretation.
4	A comprehensive acknowledgment of uncertainties is offered, with an in-depth discussion of assumptions, limitations, and possible further deterioration.

5. Item 1.5 Precision – Accuracy of Content

Score	Description
0	The interpretation is incorrect, with major misinterpretations of key risk factors and their impact.

Score	Description
1	The interpretation is mostly inaccurate, containing notable errors or misrepresentations of the clinical significance of the risk factors.
2	The interpretation is partially accurate, correctly identifying some risk factors but overemphasizing minor details or missing the overall clinical context.
3	The interpretation is mostly accurate, adequately reflecting the clinical severity of the risk factors with proper integration of the SHAP values.
4	The interpretation is fully accurate, thoroughly incorporating numerical SHAP values to clearly and precisely reflect the clinical severity and implications of each risk factor.

6. Item 1.6 Precision – Knowledge Depth and Currency

Score	Description
0	Superficial: the response lacks depth and fails to integrate current clinical evidence or detailed reasoning.
1	Limited: the response offers a basic, student-level interpretation with minimal integration of current clinical insights.
2	Adequate: the response demonstrates acceptable depth and understanding typical of a junior resident, though some nuances may be missing.
3	Good: the response shows in-depth reasoning with current clinical insight appropriate for a senior resident, clearly explaining key risk factors and their significance.
4	Expert-level: the response exhibits advanced, nuanced clinical understanding with comprehensive integration of the latest evidence, characteristic of an attending physician.

7. Item 1.7 Applicability – Patient-specific Considerations

Score	Description
0	Superficial: the response ignores key patient-specific factors and provides a generic, non-tailored analysis.
1	Limited: the response mentions patient-specific details in a superficial manner without adequate explanation of their significance.

Score	Description
2	Adequate: the response identifies most risk factors but lacks depth in explaining how these factors interact specifically in the patient's context.
3	Good: the response clearly explains the major risk factors and their clinical impact with appropriate patient-specific considerations, though without extensive nuance.
4	Expert-level: the response provides a comprehensive and nuanced analysis that thoroughly integrates each patient-specific factor and their interactions, offering detailed clinical rationale.

8. Item 1.8 Comprehensiveness – Complete Scope

Score	Description
0	Major omissions: the response omits many key risk factors, providing an incomplete picture.
1	Several missing: multiple important risk factors are not addressed, resulting in a noticeably incomplete analysis.
2	Mostly complete: most of the major risk factors are mentioned, but some relevant details or interactions are missing.
3	Comprehensive: nearly all critical risk factors and their interactions are addressed, though with minor omissions in detail.
4	Exhaustive: every relevant risk factor is clearly detailed, including nuanced interplay and critical conditions.

9. Item 1.9 Timeliness – Response Urgency

Score	Description
0	No urgency: the response does not mention any critical conditions or the need for prompt intervention.
1	Minimal urgency: the response briefly mentions risk factors without adequately addressing the immediate need for action.
2	Moderate urgency: the response acknowledges risk factors with some indication of the

Score	Description
	need for timely intervention, though it lacks strong actionable urgency.
3	High urgency: the response clearly highlights critical risk factors with an emphasis on prompt intervention, though not to the extreme degree.
4	Critical urgency: the response emphatically stresses immediate, life-saving actions with clear, direct recommendations for urgent intervention; alternatively, if the patient is stable, a response that explicitly confirms stability and integrates that into its management strategy may also be scored as 4.

Part 2: Evaluation of the Recommended Examinations

10. Item 2.1 Integration – Clear Aim

Score	Description
0	No clear diagnostic aim (e.g., suggests tests that do not address the primary risk factors for ICU mortality).
1	Vague or minimal diagnostic aim with limited focus (e.g., recommends tests without explicitly linking them to the confirmation of underlying risk factors).
2	Moderately integrated diagnostic aims (e.g., recommends tests that address major risk factors but lack explicit prioritization or complete integration of diagnostic objectives).
3	Clear diagnostic aims with substantial integration (e.g., recommends tests clearly aimed at confirming risk factors with good linkage, though some aspects may not be fully comprehensive).
4	Fully integrated, explicit diagnostic aims (e.g., precisely prioritizes bedside, urgent, and advanced tests that explicitly target the confirmation and differentiation of ICU risk factors).

11. Item 2.2 Mastery – Correct Clinical Reasoning

Score	Description
0	Entirely misdirected clinical reasoning with recommendations that do not address the patient's major risk factors.

Score	Description
1	Basic clinical reasoning with significant omissions, leading to incomplete or misaligned test recommendations.
2	Partial clinical reasoning that covers some key aspects but contains notable gaps or inconsistencies.
3	Solid clinical reasoning with appropriate test recommendations that address major risk factors, albeit with minor nuances missed.
4	Exemplary clinical reasoning fully aligned with the patient's risk factors, providing a balanced, integrated, and comprehensive set of examinations.

12. Item 2.3 Precision – Accuracy of Content

Score	Description
0	Incorrect recommendations with tests that are irrelevant or misleading for the patient's condition.
1	Several inaccuracies in the test recommendations, with significant errors in clinical appropriateness.
2	Partially accurate content with some tests appropriate while others are not fully aligned with clinical needs.
3	Mostly accurate content with recommended tests that are clinically appropriate, though with minor issues or less optimal precision.
4	Completely accurate content with all examinations precisely matching clinical needs and aligned with the patient's risk factors.

13. Item 2.4 Applicability – Patient-specific Considerations

Score	Description
0	No relevance to patient specifics; recommendations are entirely generic and ignore individual patient factors.
1	Minimal tailoring with superficial consideration of patient-specific factors, resulting in largely generic recommendations.

Score	Description
2	Moderately tailored recommendations that address some key patient-specific risks but lack full personalization.
3	Well-aligned with patient-specific risks, demonstrating thoughtful consideration of the patient's unique clinical profile.
4	Fully personalized and contextually comprehensive recommendations that integrate all relevant patient-specific data in a nuanced manner.

14. Item 2.5 Comprehensiveness – Complete Scope

Score	Description
0	Fails to include most evaluation components; only a single or minimal test is recommended, leading to an inadequate assessment.
1	Poor coverage with multiple critical evaluations missing, reflecting a significantly incomplete workup.
2	Moderate scope with partial evaluation coverage; includes several tests but lacks comprehensive detail or depth.
3	Comprehensive scope with minor omissions; covers all major evaluations but may miss some nuanced details.
4	Exemplary complete coverage of all evaluations, fully integrating bedside assessment, urgent labs, continuous monitoring, advanced imaging, and a clear conclusion aligned with the patient's risk profile.

15. Item 2.6 Timeliness – Priority

Score	Description
0	No recognition of urgency or misalignment in the recommended test sequence, leading to inappropriate prioritization.
1	Minimal attention to urgency with vague or poorly sequenced recommendations that do not match the patient's critical needs.
2	Moderately reflects response urgency with some timely recommendations but lacking a

Score	Description
	clear, structured sequence.
3	Accurately matches urgency needs with a logical sequence of immediate, urgent, and follow-up assessments, with only minor adjustments needed.
4	Optimal alignment with the required time sequence, precisely prioritizing bedside assessment, urgent investigations, and additional monitoring in strict accordance with the patient's clinical status.

Part 3: Evaluation of the Recommended Management

16. Item 3.1 Integration – Clear Aims

Score	Description
0	The management plan fails to state any clear treatment aims; interventions are listed without reference to major risk factors or any organization.
1	The plan mentions basic treatment steps (e.g., resuscitation or acute care) in isolation without linking them to patient-specific risk factors, resulting in no unified strategy.
2	The response groups interventions (e.g., separating immediate resuscitation from acute care) but overall integration is weak, with only partial linkage between phases and a limited summary.
3	The plan outlines clear and mostly targeted aims by dividing management into distinct phases (resuscitation, acute management, and supportive measures) and addresses major risk factors, though the integration may be brief or only partially cohesive.
4	The response provides a fully integrated, cohesive management plan that explicitly links immediate resuscitation, acute intervention, and ICU supportive care into a unified strategy, with a concise and clear conclusion tying all aspects together seamlessly.

17. Item 3.2 Mastery – Correct Clinical Reasoning

Score	Description
0	The plan shows no valid clinical reasoning; it recommends generic measures without addressing the patient's urgent resuscitation or ICU needs.

Score	Description
1	The plan demonstrates limited, inconsistent clinical reasoning, addressing some resuscitation elements while omitting crucial acute and supportive management steps.
2	The plan exhibits moderate clinical reasoning by incorporating key elements like immediate resuscitation and acute measures but has notable gaps in comprehensive ICU supportive care.
3	The plan displays strong clinical reasoning with nearly complete integration of immediate, acute, and supportive care components, though with minor omissions.
4	The plan demonstrates exemplary, fully integrated clinical reasoning by thoroughly addressing resuscitation, acute intervention, ICU support, and providing a clear, succinct conclusion.

18. Item 3.3 Precision – Accuracy of Content

Score	Description
0	Recommendations are clinically inaccurate or irrelevant, failing to address the patient's primary risk factors.
1	Recommendations contain several inaccuracies, such as suggesting inappropriate interventions while neglecting key immediate treatments.
2	Recommendations are partially accurate, addressing some essential interventions but omitting significant details in supportive care.
3	Recommendations are mostly accurate, providing precise guidance for immediate and targeted interventions with only minor omissions.
4	Recommendations are completely accurate, offering detailed and precise interventions that comprehensively address immediate, acute, and supportive care tailored to the patient's condition.

19. Item 3.4 Applicability – Patient-specific Considerations

Score	Description
0	No tailoring is evident; the recommendations follow a standard protocol without considering the patient's unique clinical profile.

Score	Description
1	The recommendations are generic, offering standard advice with minimal or no adjustments based on patient-specific factors.
2	Some adjustments are made based on patient data, but the recommendations lack depth in customization to the patient's unique profile.
3	The recommendations are well-tailored, incorporating key patient-specific factors such as age, comorbidities, and current clinical status to inform management decisions.
4	The recommendations are highly customized, integrating detailed patient-specific information to precisely modify interventions and optimize outcomes.

20. Item 3.5 Comprehensiveness – Complete Scope

Score	Description
0	The management plan omits several key interventions, failing to address immediate resuscitation, acute management, ICU supportive care, or a unifying conclusion.
1	The management plan includes several recommendations, but multiple critical interventions are missing, resulting in an incomplete scope.
2	The management plan is mostly complete, covering most key interventions, yet it lacks sufficient detail or integration in some areas.
3	The management plan is comprehensive, addressing immediate resuscitation, acute management, and ICU supportive care with a coherent structure, though minor details may be underdeveloped.
4	The management plan is exhaustive, including every essential intervention with thorough detail and seamless integration across immediate, acute, and supportive phases of care.

21. Item 3.6 Timeliness – Priority

Score	Description
0	The management plan fails to address timeliness, lacking any indication of urgency or time-sensitive intervention sequencing.
1	The plan mentions urgency but provides minimal focus on critical risk factors and lacks a

Score	Description
	clearly structured time sequence.
2	The plan moderately prioritizes major risk factors with some indication of time-sensitive steps, but the overall sequence and urgency remain too simplistic.
3	The plan mostly prioritizes critical interventions with a clear time-sensitive ordering, though the sequence may not be fully detailed.
4	The plan exemplarily prioritizes critical management steps with an explicit, well-structured time sequence that clearly distinguishes immediate from subsequent interventions.

Part 4: Evaluation of the Follow-up Plan

22. Item 4.1 Precision – Accuracy of Content

Score	Description
0	The follow-up plan is grossly inaccurate, missing essential monitoring elements, and lacks any actionable or time-sequenced structure.
1	The plan includes some monitoring elements but has major inaccuracies or omissions. It lacks a coherent time sequence and fails to address the patient's dynamic needs.
2	The plan is partially accurate and contains basic monitoring steps; however, it is overly simplistic and does not provide a clear, sequential timeline for continuous monitoring, acute reassessment, and stability follow-up.
3	The plan is accurate with mostly complete details. It outlines monitoring and reassessment with a reasonable time sequence across continuous, acute, and stability phases, though minor refinements in the sequential structure may be needed.
4	The plan is highly precise and comprehensive. It delivers detailed, sequential follow-up that includes continuous monitoring, clearly defined intervals for acute reassessment, and structured stability follow-up, fully addressing the patient's evolving clinical profile.

23. Item 4.2 Applicability – Patient-specific Considerations

Score	Description
0	The follow-up plan is completely generic with no incorporation of patient-specific details. It does not address unique factors such as advanced age, intubation, hypothermia, or other individualized risks.
1	The plan shows minimal patient-specific tailoring. It may mention standard monitoring elements but lacks clear reference to the patient's unique risk factors and does not integrate these with a sequential or time-based structure.
2	The plan includes basic patient-specific details, addressing some aspects of the patient's risk profile, but remains too simplistic. It fails to integrate a detailed, time-sequenced structure that fully tailors monitoring and interventions to the patient's dynamic needs.
3	The plan provides substantial customization to the patient's clinical profile. It includes tailored monitoring and reassessment intervals that acknowledge key risk factors (e.g., intubation, hypothermia, advanced age) with a reasonably clear time sequence, though some minor details may be omitted.
4	The plan is highly tailored and fully patient-specific. It delivers a comprehensive, sequential follow-up that incorporates all relevant patient factors and dynamically adjusts monitoring and interventions based on the patient's evolving clinical status.

24. Item 4.3 Timeliness – Timely Reassessment Intervals

Score	Description
1	Very limited, vague priority details (e.g., mentions monitoring but lacks clear timing or explicit reassessment and stability protocols).
2	Partial coverage of priority tiers (e.g., specifies continuous monitoring and acute reassessment but omits structured stability follow-up details).
3	Mostly comprehensive, priority-based follow-up (e.g., outlines continuous monitoring, timely acute reassessment, and basic stability follow-up with minor omissions).
4	Fully tailored, comprehensive priority follow-up (e.g., precisely details hourly checks, acute reassessment intervals, and structured stability follow-up, perfectly matching patient need).

Part 5: Evaluation of the Summary

25. Item 5.1 Comprehensiveness – Summary and Overall Quality

Score	Description
0	The summary omits major risk factors and lacks any clear structure. No priority actions or concluding recommendations are provided.
1	The summary includes only a few risk factors or provides vague or minimal details. It mentions one or less than two priority actions, or includes actions that are not directly related to the key risk factors. The structure is minimal and lacks a clear introduction or conclusion.
2	The summary partially integrates key risk factors with some mention of patient risks. It lists one to two priority actions but lacks a clear, organized introduction or a concise conclusion. Some important details or recommendations may be missing or underdeveloped.
3	The summary is mostly comprehensive and presents a clear structure. It opens with an introduction that outlines the major risk factors (even if not exhaustively detailed). It includes three priority actions, though they may not be perfectly enumerated, and provides a brief concluding statement. (Example: A response that outlines the risk factors and suggests addressing hypothermia, optimizing ventilatory/metabolic status, and monitoring chronic diseases would qualify for this score.)
4	The summary offers an exemplary synthesis with a highly structured overview. It succinctly restates all key risk factors, provides a clear and detailed introduction, explicitly lists three well-defined and actionable priority steps, and offers a concise, insightful conclusion. All elements (introduction, priority actions, conclusion) are effectively integrated and directly aligned with the major risk factors.

5. Python script for evaluation

5.1 E1

Evaluation for GPT o3-mini

```
import os
import json
import time
import sys
import logging
import csv
import pandas as pd
import datetime
from openai import AzureOpenAI

# ------------------------------------------------------------------------
# Scoring Criteria.
# ------------------------------------------------------------------------

SCORING_CRITERIA_PART1 = r"""
Part 1: Evaluation of the Interpretation of Risk Factors

Item 1.1 Integration – Clinical Relevance
0: No relevant risk factors mentioned. (Example: Omitting key factors such as intubation status or vital signs)
1: Bare mention of risk factors without context or proper interpretation. (Example: Simply listing "intubation" without linking it to the patient's risk outcome)
2: Basic identification of risk factors without explicit reference to SHAP values, or includes misinterpretation (e.g., normal variables are incorrectly identified as increasing risk). (Example: Listing intubation and SpO₂ as risk factors without showing their SHAP values or clarifying their true clinical impact)
3: Clear explanation that includes numerical SHAP values to identify risk factors. (Example: Explains that intubation increases risk by citing its SHAP value, but with limited integration of broader clinical context)
4: Detailed integration of numerical SHAP values with thorough clinical correlation and context, accurately relating both risk-enhancing and protective
```

factors. (Example: Thoroughly explains how the high SHAP value for intubation and the low or negative SHAP values for normal variables combine to define the patient's overall risk profile)

Item 1.2 Integration – Clarity of Information
0: The interpretation is disorganized and unclear, lacking a coherent structure or logical flow.
1: The interpretation is partially clear with noticeable ambiguities that impede full understanding of the risk factors.
2: The interpretation is generally clear but only provides a basic structure without exploring potential causes of the risk factors.
3: The interpretation is clear and organized, with a logical flow and mention of at least one potential cause of the risk factors.
4: The interpretation is exceptionally clear and well-organized, seamlessly presenting multiple distinct potential causes of the risk factors.

Item 1.3 Mastery – Balance and Bias Assessment
0: One-sided – Only risk-enhancing factors are discussed, with no mention of protective factors.
1: Minimal balance – Protective factors are mentioned but are mischaracterized or insufficiently addressed (e.g., normal variables are erroneously noted as contributing to increased risk).
2: Moderately balanced – Both risk-enhancing and protective factors are mentioned, but the protective factors receive limited or imprecise emphasis (e.g., protective factors are noted but with slight misinterpretation).
3: Balanced – Both risk-enhancing and protective factors are clearly discussed, with a mild emphasis on protective factors and overall fair balance.
4: Well-balanced – A comprehensive and thorough discussion of both risk-enhancing and protective factors is provided, with detailed and accurate emphasis on the protective factors.

Item 1.4 Mastery – Uncertainty Acknowledgment
0: No acknowledgment of uncertainty, limitations, or assumptions is provided when they are clearly relevant (e.g., in moderate-to-high risk scenarios, failing to mention any potential variability or interaction among risk factors).
1: Minimal acknowledgment of uncertainty is present but lacks detail and does not explore how risk factors might interact (e.g., a brief, cursory note on possible influencing factors without further explanation).

2: A basic acknowledgment of uncertainty is made with a brief mention of limitations or assumptions, or in cases where the predicted risk is extremely low (<2% ICU mortality) the absence of detailed uncertainty discussion is acceptable (e.g., a low-risk scenario where the response neutrally states the risk without additional qualifiers).
3: A clear acknowledgment of uncertainty is provided, detailing potential interactions among risk factors and limitations in the interpretation (e.g., a response that explicitly discusses assumptions and outlines areas where further assessment might be needed).
4: A comprehensive acknowledgment of uncertainties is offered, with an in-depth discussion of assumptions, limitations, and potential for further deterioration (e.g., a detailed exploration of risk factors, their interactions, and the boundaries of the prediction model).

Item 1.5 Precision – Accuracy of Content
0: The interpretation is incorrect, with major misinterpretations of key risk factors and their impact.
1: The interpretation is mostly inaccurate, containing notable errors or misrepresentations of the clinical significance of the risk factors.
2: The interpretation is partially accurate, correctly identifying some risk factors but overemphasizing minor details or missing the overall clinical context.
3: The interpretation is mostly accurate, adequately reflecting the clinical severity of the risk factors with proper integration of the SHAP values.
4: The interpretation is fully accurate, thoroughly incorporating numerical SHAP values to clearly and precisely reflect the clinical severity and implications of each risk factor.

Item 1.6 Precision – Knowledge Depth and Currency
0: Superficial – The response lacks depth and fails to integrate current clinical evidence or detailed reasoning.
1: Limited – The response offers a basic, student-level interpretation with minimal integration of current clinical insights.
2: Adequate – The response demonstrates acceptable depth and understanding typical of a junior resident, though some nuances may be missing.

3: Good – The response shows in-depth reasoning with current clinical insight appropriate for a senior resident, clearly explaining key risk factors and their significance.
4: Expert-level – The response exhibits advanced, nuanced clinical understanding with comprehensive integration of the latest evidence, characteristic of an attending physician.

Item 1.7 Applicability – Patient-specific considerations
0: Superficial – The response ignores key patient-specific factors and provides a generic, non-tailored analysis.
1: Limited – The response mentions patient-specific details in a superficial manner without adequate explanation of their significance.
2: Adequate – The response identifies most risk factors but lacks depth in explaining how these factors interact specifically in the patient's context.
3: Good – The response clearly explains the major risk factors and their clinical impact with appropriate patient-specific considerations, though without extensive nuance.
4: Expert-level – The response provides a comprehensive and nuanced analysis that thoroughly integrates each patient-specific factor and their interactions, offering detailed clinical rationale

Item 1.8 Comprehensiveness – Complete Scope
0: Major omissions – The response omits many key risk factors, providing an incomplete picture.
1: Several missing – Multiple important risk factors are not addressed, resulting in a noticeably incomplete analysis.
2: Mostly complete – Most of the major risk factors are mentioned, but some relevant details or interactions are missing.
3: Comprehensive – Nearly all critical risk factors and their interactions are addressed, though with minor omissions in detail.
4: Exhaustive – Every relevant risk factor is clearly detailed, including nuanced interplay and critical conditions.

Item 1.9 Timeliness – Response Urgency
0: No appropriate urgency – The response fails to match the risk profile, neither identifying critical/life-threatening factors in high-risk cases nor reassuring stability in low-risk scenarios.

1: Minimal urgency – The response mentions risk factors but underemphasizes life-threatening conditions in high-risk cases or overstates them in stable situations.
2: Moderate urgency – The response recognizes risk factors, identifying potential life-threatening conditions in high-risk cases or confirming stability in low-risk scenarios without extraneous commentary.
3: High urgency – The response distinguishes key risk factors by prioritizing recognition of critical/life-threatening conditions in high-risk cases or confidently confirming stability in low-risk situations.
4: Critical calibration – The response flawlessly aligns with the risk profile by precisely identifying critical/life-threatening conditions in high-risk cases or accurately assuring stability in low-risk scenarios.

"""

SCORING_CRITERIA_PART2 = r"""
Part 2: Evaluation of the Recommended Examinations

Item 2.1 Integration – Clear aim
0: No clear diagnostic aim (e.g., suggests tests that do not address the primary risk factors for ICU mortality)
1: Vague or minimal diagnostic aim with limited focus (e.g., recommends tests without explicitly linking them to the confirmation of underlying risk factors)
2: Moderately integrated diagnostic aims (e.g., recommends tests that address major risk factors but lack explicit prioritization or complete integration of diagnostic objectives)
3: Clear diagnostic aims with substantial integration (e.g., recommends tests clearly aimed at confirming risk factors with good linkage, though some aspects may not be fully comprehensive)
4: Fully integrated, explicit diagnostic aims (e.g., precisely prioritizes bedside, urgent, and advanced tests that explicitly target the confirmation and differentiation of ICU risk factors)

Item 2.2 Mastery – Correct clinical reasoning
0: Entirely misdirected clinical reasoning with recommendations that do not address the patient's major risk factors
1: Basic clinical reasoning with significant omissions, leading to incomplete or misaligned test recommendations

2: Partial clinical reasoning that covers some key aspects but contains notable gaps or inconsistencies
3: Solid clinical reasoning with appropriate test recommendations that address major risk factors, albeit with minor nuances missed
4: Exemplary clinical reasoning fully aligned with the patient's risk factors, providing a balanced, integrated, and comprehensive set of examinations

Item 2.3 Precision – Accuracy of Content
0: Incorrect recommendations with tests that are irrelevant or misleading for the patient's condition
1: Several inaccuracies in the test recommendations, with significant errors in clinical appropriateness
2: Partially accurate content with some tests appropriate while others are not fully aligned with clinical needs
3: Mostly accurate content with recommended tests that are clinically appropriate, though with minor issues or less optimal precision
4: Completely accurate content with all examinations precisely matching clinical needs and aligned with the patient's risk factors

Item 2.4 Applicability – Patient-specific Considerations
0: No relevance to patient specifics; recommendations are entirely generic and ignore individual patient factors
1: Minimal tailoring with superficial consideration of patient-specific factors, resulting in largely generic recommendations
2: Moderately tailored recommendations that address some key patient-specific risks but lack full personalization
3: Well-aligned with patient-specific risks, demonstrating thoughtful consideration of the patient's unique clinical profile
4: Fully personalized and contextually comprehensive recommendations that integrate all relevant patient-specific data in a nuanced manner

Item 2.5 Comprehensiveness – Complete Scope
0: Fails to include most evaluation components; only a single or minimal test is recommended, leading to an inadequate assessment
1: Poor coverage with multiple critical evaluations missing, reflecting a significantly incomplete workup
2: Moderate scope with partial evaluation coverage; includes several tests but lacks comprehensive detail or depth

3: Comprehensive scope with minor omissions; covers all major evaluations but may miss some nuanced details
4: Exemplary complete coverage of all evaluations, fully integrating bedside assessment, urgent labs, continuous monitoring, advanced imaging, and a clear conclusion aligned with the patient's risk profile

Item 2.6 Timeliness – Priority
0: No recognition of urgency or misalignment in the recommended test sequence, leading to inappropriate prioritization
1: Minimal attention to urgency with vague or poorly sequenced recommendations that do not match the patient's critical needs
2: Moderately reflects response urgency with some timely recommendations but lacking a clear, structured sequence
3: Accurately matches urgency needs with a logical sequence of immediate, urgent, and follow-up assessments, with only minor adjustments needed
4: Optimal alignment with the required time sequence, precisely prioritizing bedside assessment, urgent investigations, and additional monitoring in strict accordance with the patient's clinical status

"""

SCORING_CRITERIA_PART3 = r"""
Part 3: Evaluation of the Recommended Management

Item 3.1 Integration – Clear aims
0: The management plan fails to state any clear treatment aims; interventions are listed without reference to major risk factors or any organization.
1: The plan mentions basic treatment steps (e.g., resuscitation or acute care) in isolation without linking them to patient-specific risk factors, resulting in no unified strategy.
2: The response groups interventions (e.g., separating immediate resuscitation from acute care) but overall integration is weak, with only partial linkage between phases and a limited summary.
3: The plan outlines clear and mostly targeted aims by dividing management into distinct phases (resuscitation, acute management, and supportive measures) and addresses major risk factors, though the integration may be brief or only partially cohesive.

4: The response provides a fully integrated, cohesive management plan that explicitly links immediate resuscitation, acute intervention, and ICU supportive care into a unified strategy, with a concise and clear conclusion tying all aspects together seamlessly.

Item 3.2 Mastery – Correct clinical reasoning
0: The plan shows no valid clinical reasoning; it recommends generic measures without addressing the patient's urgent resuscitation or ICU needs.
1: The plan demonstrates limited, inconsistent clinical reasoning, addressing some resuscitation elements while omitting crucial acute and supportive management steps.
2: The plan exhibits moderate clinical reasoning by incorporating key elements like immediate resuscitation and acute measures but has notable gaps in comprehensive ICU supportive care.
3: The plan displays strong clinical reasoning with nearly complete integration of immediate, acute, and supportive care components, though with minor omissions.
4: The plan demonstrates exemplary, fully integrated clinical reasoning by thoroughly addressing resuscitation, acute intervention, ICU support, and providing a clear, succinct conclusion.

Item 3.3 Precision – Accuracy of Content
0: Recommendations are clinically inaccurate or irrelevant, failing to address the patient's primary risk factors.
1: Recommendations contain several inaccuracies, such as suggesting inappropriate interventions while neglecting key immediate treatments.
2: Recommendations are partially accurate, addressing some essential interventions but omitting significant details in supportive care.
3: Recommendations are mostly accurate, providing precise guidance for immediate and targeted interventions with only minor omissions.
4: Recommendations are completely accurate, offering detailed and precise interventions that comprehensively address immediate, acute, and supportive care tailored to the patient's condition.

Item 3.4 Applicability – Patient-specific Considerations
0: No tailoring is evident; the recommendations follow a standard protocol without considering the patient's unique clinical profile.

1: The recommendations are generic, offering standard advice with minimal or no adjustments based on patient-specific factors.
2: Some adjustments are made based on patient data, but the recommendations lack depth in customization to the patient's unique profile.
3: The recommendations are well-tailored, incorporating key patient-specific factors such as age, comorbidities, and current clinical status to inform management decisions.
4: The recommendations are highly customized, integrating detailed patient-specific information to precisely modify interventions and optimize outcomes.

Item 3.5 Comprehensiveness – Complete Scope
0: The management plan omits several key interventions, failing to address immediate resuscitation, acute management, ICU supportive care, or a unifying conclusion.
1: The management plan includes several recommendations, but multiple critical interventions are missing, resulting in an incomplete scope.
2: The management plan is mostly complete, covering most key interventions, yet it lacks sufficient detail or integration in some areas.
3: The management plan is comprehensive, addressing immediate resuscitation, acute management, and ICU supportive care with a coherent structure, though minor details may be underdeveloped.
4: The management plan is exhaustive, including every essential intervention with thorough detail and seamless integration across immediate, acute, and supportive phases of care.

Item 3.6 Timeliness – Priority
0: The management plan fails to address timeliness, lacking any indication of urgency or time-sensitive intervention sequencing.
1: The plan mentions urgency but provides minimal focus on critical risk factors and lacks a clearly structured time sequence.
2: The plan moderately prioritizes major risk factors with some indication of time-sensitive steps, but the overall sequence and urgency remain too simplistic.
3: The plan mostly prioritizes critical interventions with a clear time-sensitive ordering, though the sequence may not be fully detailed.
4: The plan exemplarily prioritizes critical management steps with an explicit, well-structured time sequence that clearly distinguishes immediate from subsequent interventions

"""

SCORING_CRITERIA_PART4_5 = r"""
Part 4: Evaluation of the Follow-up Plan

Item 4.1 Precision – Accuracy of Content
0: The follow-up plan is grossly inaccurate, missing essential monitoring elements, and lacks any actionable or time-sequenced structure.
1: The plan includes some monitoring elements but has major inaccuracies or omissions. It lacks a coherent time sequence and fails to address the patient's dynamic needs.
2: The plan is partially accurate and contains basic monitoring steps; however, it is overly simplistic and does not provide a clear, sequential timeline for continuous monitoring, acute reassessment, and stability follow-up.
3: The plan is accurate with mostly complete details. It outlines monitoring and reassessment with a reasonable time sequence across continuous, acute, and stability phases, though minor refinements in the sequential structure may be needed.
4: The plan is highly precise and comprehensive. It delivers detailed, sequential follow-up that includes continuous monitoring, clearly defined intervals for acute reassessment, and structured stability follow-up, fully addressing the patient's evolving clinical profile.

Item 4.2 Applicability – Patient-specific Considerations
0: The follow-up plan is completely generic with no incorporation of patient-specific details. It does not address unique factors such as advanced age, intubation, hypothermia, or other individualized risks.
1: The plan shows minimal patient-specific tailoring. It may mention standard monitoring elements but lacks clear reference to the patient's unique risk factors and does not integrate these with a sequential or time-based structure.
2: The plan includes basic patient-specific details—addressing some aspects of the patient's risk profile—but remains too simplistic. It fails to integrate a detailed, time-sequenced structure that fully tailors monitoring and interventions to the patient's dynamic needs.
3: The plan provides substantial customization to the patient's clinical profile. It includes tailored monitoring and reassessment intervals that acknowledge key risk factors (e.g., intubation, hypothermia, advanced age) with a reasonably clear time sequence, though some minor details may be omitted.

4: The plan is highly tailored and fully patient-specific. It delivers a comprehensive, sequential follow-up that incorporates all relevant patient factors and dynamically adjusts monitoring and interventions based on the patient's evolving clinical status.

Item 4.3 Timeliness – Timely Reassessment Intervals
0: The management plan fails to address timeliness, lacking any indication of urgency or time-sensitive intervention sequencing.
1: Very limited, vague priority details (e.g., Mentions monitoring but lacks clear timing or explicit reassessment and stability protocols)
2: Partial coverage of priority tiers (e.g., Specifies continuous monitoring and acute reassessment but omits structured stability follow-up details)
3: Mostly comprehensive, priority-based follow-up (e.g., Outlines continuous monitoring, timely acute reassessment, and basic stability follow-up with minor omissions)
4: Fully tailored, comprehensive priority follow-up (e.g., Precisely details hourly checks, acute reassessment intervals, and structured stability follow-up, perfectly matching patient need)

Part 5: Evaluation of the Summary

Item 5.1 Comprehensiveness – Summary and Overall Quality
0: The summary lacks structure, omits major risk factors, and provides no priority actions or concluding recommendations.
1: The summary includes minimal risk factors with vague details, mentions fewer than two priority actions (often unrelated to key risks), and has minimal structure without clear introduction or conclusion.
2: The summary partially integrates key risk factors with some patient risks, lists one to two priority actions, but lacks clear organization in introduction or conclusion with some important details missing.
3: The summary presents a mostly comprehensive, clearly structured overview with an introduction outlining major risk factors, includes three priority actions (though not perfectly enumerated), and provides a brief concluding statement.
4: The summary delivers an exemplary, highly structured synthesis that succinctly restates all key risk factors, provides a detailed introduction, explicitly lists three well-defined priority steps, and offers an insightful conclusion with all elements effectively integrated.

```
"""

# ------------------------------------------------------------------------
# New method to call AzureOpenAI API using single conversation
# ------------------------------------------------------------------------
def call_openai(system_prompt, user_prompt, max_tokens, api_key,
llm_model, azure_api_version, azure_endpoint, timeout=60):
    """
    Return (llm_response_text, usage_info)
    """
    # Instantiate the AzureOpenAI client using Azure-specific parameters.
    client = AzureOpenAI(
        api_version=azure_api_version,
        azure_endpoint=azure_endpoint,
        api_key=api_key,
    )
    try:
        response = client.chat.completions.create(
            model=llm_model,
            messages=[
                {"role": "system", "content": system_prompt},
                {"role": "user", "content": user_prompt}
            ],
            max_completion_tokens=max_tokens,    # Use
max_completion_tokens instead of max_tokens
            timeout=timeout    # Set the timeout to the provided value
        )
        llm_response_text = response.choices[0].message.content
        usage_info = getattr(response, "usage", {})
        return llm_response_text, usage_info
    except Exception as e:
        logging.error(f"Error calling AzureOpenAI: {e}")
        if hasattr(e, 'response') and e.response is not None:
            logging.error(f"Response text: {e.response.text}")
        sys.exit(1)

# ------------------------------------------------------------------------
```

```
# Helper function for retry logic for each conversation
# ------------------------------------------------------------------------
def call_with_retry(system_prompt, user_prompt, adjustable_params):
    max_retries = adjustable_params.get("MAX_RETRIES", 3)
    attempt = 0
    evaluation = None
    usage_info = None
    start_time = time.time()
    while attempt < max_retries:
        attempt += 1
        try:
            response_content, usage_info = call_openai(
                system_prompt,
                user_prompt,
                adjustable_params["MAX_TOKENS"],
                adjustable_params["OPENAI_API_KEY"],
                adjustable_params["MODEL_NAME"],
                adjustable_params["AZURE_API_VERSION"],     # New
parameter for Azure API version
                adjustable_params["AZURE_ENDPOINT"],            # New
parameter for Azure endpoint
                timeout=adjustable_params.get("TIMEOUT", 60)
            )
            evaluation = json.loads(response_content)
            if evaluation:
                break
            else:
                logging.error(f"Attempt {attempt}: Incomplete evaluation
returned, retrying...")
        except Exception as e:
            logging.error(f"Attempt {attempt}: Error parsing LLM response.
Error: {e}")
        time.sleep(1)
    if evaluation is None:
        evaluation = {}
    end_time = time.time()
    usage_info = dict(usage_info) if usage_info is not None else {}
    usage_info["evaluation_time"] = int(end_time - start_time)
```

```
        return evaluation, usage_info

# ----------------------------------------------------------------------------
# Evaluate each row with the AzureOpenAI API using the revised prompt with
multiple conversations
# ----------------------------------------------------------------------------
def evaluate_with_api(row, adjustable_params):
        overall_total = 0
        detailed_evaluation_parts = []
        conversation_results = {}
        total_prompt_tokens = 0
        total_completion_tokens = 0
        total_tokens = 0
        total_evaluation_time = 0

        # ---- Conversation 1: Part 1 (Interpretation) ----
        part1_system_prompt = (
            "You are an expert clinical evaluator with extensive clinical and
evidence-based practice experience.\n"
            "Your task is to thoroughly analyze the LLM response provided for
the Interpretation section and provide a detailed numerical scoring
breakdown."
        )
        part1_user_prompt = f"""
Patient Information:
{row.get('Patient', 'N/A')}

LLM Response for Interpretation:
{row.get('Interpretation', 'N/A')}

Instructions:
Evaluate the Interpretation section using the following scoring criteria:
{SCORING_CRITERIA_PART1}

Provide a JSON output with the key "Interpretation" containing:
   - "subtotal": <subtotal score for Interpretation>,
   - "items": [{{ "item": "<item name>", "score": <score>, "comment":
"<explanation>" }}, ...]
```

```
Ensure the JSON is valid.
"""
        result1, usage_info1 = call_with_retry(part1_system_prompt,
part1_user_prompt, adjustable_params)
        conversation_results["Interpretation"] = result1.get("Interpretation",
{"subtotal": 0, "items": []})
        overall_total += conversation_results["Interpretation"].get("subtotal", 0)
        detailed_evaluation_parts.append(result1.get("Detailed_Evaluation",
"Interpretation evaluation completed."))
        total_prompt_tokens += usage_info1.get("prompt_tokens", 0)
        total_completion_tokens += usage_info1.get("completion_tokens", 0)
        total_tokens += usage_info1.get("total_tokens", 0)
        total_evaluation_time += usage_info1.get("evaluation_time", 0)

        # ---- Conversation 2: Part 2 (Examination) ----
        part2_system_prompt = (
            "You are an expert clinical evaluator with extensive clinical and
evidence-based practice experience.\n"
            "Your task is to thoroughly analyze the LLM responses provided for
the Interpretation and Examination sections and provide a detailed numerical
scoring breakdown for the Examination evaluation."
        )
        part2_user_prompt = f"""
Patient Information:
{row.get('Patient', 'N/A')}

LLM Response for Interpretation:
{row.get('Interpretation', 'N/A')}

LLM Response for Examination:
{row.get('Examination', 'N/A')}

Instructions:
Evaluate the Examination section using the following scoring criteria:
{SCORING_CRITERIA_PART2}

Provide a JSON output with the key "Examination" containing:
   - "subtotal": <subtotal score for Examination>,
```

```
   - "items": [{{ "item": "<item name>", "score": <score>, "comment":
"<explanation>" }}, ...]
Ensure the JSON is valid.
"""
        result2, usage_info2 = call_with_retry(part2_system_prompt,
part2_user_prompt, adjustable_params)
        conversation_results["Examination"] = result2.get("Examination",
{"subtotal": 0, "items": []})
        overall_total += conversation_results["Examination"].get("subtotal", 0)
        detailed_evaluation_parts.append(result2.get("Detailed_Evaluation",
"Examination evaluation completed."))
        total_prompt_tokens += usage_info2.get("prompt_tokens", 0)
        total_completion_tokens += usage_info2.get("completion_tokens", 0)
        total_tokens += usage_info2.get("total_tokens", 0)
        total_evaluation_time += usage_info2.get("evaluation_time", 0)

        # ---- Conversation 3: Part 3 (Management) ----
        part3_system_prompt = (
            "You are an expert clinical evaluator with extensive clinical and
evidence-based practice experience.\n"
            "Your task is to thoroughly analyze the LLM responses provided for
the Interpretation, Examination, and Management sections and provide a
detailed numerical scoring breakdown for the Management evaluation."
        )
        part3_user_prompt = f"""
Patient Information:
{row.get('Patient', 'N/A')}

LLM Response for Interpretation:
{row.get('Interpretation', 'N/A')}

LLM Response for Examination:
{row.get('Examination', 'N/A')}

LLM Response for Management:
{row.get('Management', 'N/A')}

Instructions:
```

```
Evaluate the Management section using the following scoring criteria:
{SCORING_CRITERIA_PART3}

Provide a JSON output with the key "Management" containing:
   - "subtotal": <subtotal score for Management>,
   - "items": [{{ "item": "<item name>", "score": <score>, "comment":
"<explanation>" }}, ...]
Ensure the JSON is valid.
"""
        result3, usage_info3 = call_with_retry(part3_system_prompt,
part3_user_prompt, adjustable_params)
        conversation_results["Management"] = result3.get("Management",
{"subtotal": 0, "items": []})
        overall_total += conversation_results["Management"].get("subtotal", 0)
        detailed_evaluation_parts.append(result3.get("Detailed_Evaluation",
"Management evaluation completed."))
        total_prompt_tokens += usage_info3.get("prompt_tokens", 0)
        total_completion_tokens += usage_info3.get("completion_tokens", 0)
        total_tokens += usage_info3.get("total_tokens", 0)
        total_evaluation_time += usage_info3.get("evaluation_time", 0)

        # ---- Conversation 4: Part 4+5 (Follow_up and Summary) ----
        part45_system_prompt = (
            "You are an expert clinical evaluator with extensive clinical and
evidence-based practice experience.\n"
            "Your task is to thoroughly analyze the LLM responses provided for
the Interpretation, Examination, Management, Follow_up, and Summary
sections and provide a detailed numerical scoring breakdown for the
Follow_up and Summary evaluations."
        )
        part45_user_prompt = f"""
Patient Information:
{row.get('Patient', 'N/A')}

LLM Response for Interpretation:
{row.get('Interpretation', 'N/A')}

LLM Response for Examination:
```

```
{row.get('Examination', 'N/A')}

LLM Response for Management:
{row.get('Management', 'N/A')}

LLM Response for Follow_up:
{row.get('Follow_up', 'N/A')}

LLM Response for Summary:
{row.get('Summary', 'N/A')}

Instructions:
Evaluate the Follow_up and Summary sections using the following scoring criteria:
{SCORING_CRITERIA_PART4_5}

Provide a JSON output with the keys:
  "Follow_up": {{
      "subtotal": <subtotal score for Follow_up>,
      "items": [{{ "item": "<item name>", "score": <score>, "comment": "<explanation>" }}, ...]
  }},
  "Summary": {{
      "subtotal": <subtotal score for Summary>,
      "items": [{{ "item": "<item name>", "score": <score>, "comment": "<explanation>" }}]
  }}
Ensure the JSON is valid.
"""
    result4, usage_info4 = call_with_retry(part45_system_prompt, part45_user_prompt, adjustable_params)
    conversation_results["Follow_up"] = result4.get("Follow_up", {"subtotal": 0, "items": []})
    conversation_results["Summary"] = result4.get("Summary", {"subtotal": 0, "items": []})
    overall_total += conversation_results["Follow_up"].get("subtotal", 0) + conversation_results["Summary"].get("subtotal", 0)
```

```
        detailed_evaluation_parts.append(result4.get("Detailed_Evaluation",
"Follow_up and Summary evaluation completed."))
        total_prompt_tokens += usage_info4.get("prompt_tokens", 0)
        total_completion_tokens += usage_info4.get("completion_tokens", 0)
        total_tokens += usage_info4.get("total_tokens", 0)
        total_evaluation_time += usage_info4.get("evaluation_time", 0)

        combined_detailed_evaluation = "\n".join(detailed_evaluation_parts)

        final_evaluation = {
            "Interpretation": conversation_results.get("Interpretation"),
            "Examination": conversation_results.get("Examination"),
            "Management": conversation_results.get("Management"),
            "Follow_up": conversation_results.get("Follow_up"),
            "Summary": conversation_results.get("Summary"),
            "Total": overall_total,
            "Detailed_Evaluation": combined_detailed_evaluation,
            "prompt_tokens": total_prompt_tokens,
            "completion_tokens": total_completion_tokens,
            "total_tokens": total_tokens,
            "eva_prompt_token": total_prompt_tokens,
            "eva_completion_token": total_completion_tokens,
            "eva_total_token": total_tokens,
            "LLM_evaluation_time(s)": total_evaluation_time,
            "response_id": row.get("response_id", ""),
            "stay_id": row.get("stay_id", "")
        }

        return final_evaluation

# ---------------------------------------------------------------------------
# Formatting Function for TXT Output using dynamic JSON evaluation result
# ---------------------------------------------------------------------------
def format_evaluation(eval_item):
        formatted = ""
        sections = ["Interpretation", "Examination", "Management", "Follow_up",
"Summary"]
        for section in sections:
```

```
            section_data = eval_item.get(section) or {}
            subtotal = section_data.get("subtotal", 0)
            formatted += f"Part {section} ({subtotal})\n"
            items = section_data.get("items", [])
            for obj in items:
                item_name = obj.get("item", "N/A")
                score = obj.get("score", 0)
                comment = obj.get("comment", "")
                formatted += f"• {item_name}: Score {score}. {comment}\n"
            formatted += f"Subtotal for {section} = {subtotal}\n\n"
        total = eval_item.get("Total", 0)
        detailed_summary = eval_item.get("Detailed_Evaluation", "")
        formatted += f"Final Overall Score = {total}\n\n"
        formatted += f"Detailed Evaluation Summary:\n{detailed_summary}\n"
        return formatted

# ------------------------------------------------------------------------
# Main processing function with immediate saving for each evaluation
# ------------------------------------------------------------------------
def process_evaluations(adjustable_params):
        input_file = adjustable_params["INPUT_FILE_PATH"]
        base_dir = os.path.dirname(os.path.abspath(input_file))
        today_date = datetime.datetime.now().strftime("%Y%m%d")
        csv_output_filename =
f"evaluation_{adjustable_params['MODEL_NAME']}_{today_date}.csv"
        csv_output_path = os.path.join(base_dir, csv_output_filename)

        # Define CSV header columns
        summary_columns = [
            "response_id",
            "stay_id",
            "ICU_mortality",
            "Predicted_risk(%)",
            "risk_group",
            "ML_model",
            "LLM_model",
            "Prompt",
            "RAG",
```

```
            "prompt_tokens",
            "completion_tokens",
            "total_tokens",
            "eva_prompt_token",
            "eva_completion_token",
            "eva_total_token",
            "LLM_response_t(s)",
            "LLM_waiting_t(s)",
            "Python_time(s)",
            "Total_time(s)",
            "LLM_evaluation_time(s)",
            "Interpretation",
            "Examination",
            "Management",
            "Follow_up",
            "Summary",
            "Total",
            "Interpretation_JSON",
            "Examination_JSON",
            "Management_JSON",
            "Follow_up_JSON",
            "Summary_JSON"
        ]

        # Task 1: Check if the CSV file exists and if it is open
        if os.path.exists(csv_output_path):
            try:
                with open(csv_output_path, 'a', newline='', encoding='utf-8') as
csvfile:
                    pass
                print(f"Notice: The file {csv_output_filename} already exists and
is accessible. Consider renaming it if you want to preserve the existing data.")
            except Exception as e:
                print(f"Warning: The file {csv_output_filename} appears to be
open or locked by another program. Please save, close, or rename the file
and then stop the process.")
                return    # Stop the process if the file cannot be opened.
        else:
```

```
            # File does not exist, create it and write header immediately.
            with open(csv_output_path, 'w', newline='', encoding='utf-8') as
csvfile:
                writer = csv.DictWriter(csvfile, fieldnames=summary_columns)
                writer.writeheader()

        try:
            df = pd.read_csv(
                input_file,
                sep=",",
                encoding="cp1252",
                engine="python",
                quoting=csv.QUOTE_MINIMAL,
                skip_blank_lines=True
            )
            df['response_id'] = df['response_id'].astype(str)
            df['stay_id'] = df['stay_id'].astype(str)
        except Exception as e:
            print(f"Error reading CSV file: {e}")
            return

        total_cases = len(df)

        for idx, row in df.iterrows():
            print(f"Evaluating case {idx+1}/{total_cases}
(response_id={row.get('response_id', 'N/A')}, stay_id={row.get('stay_id',
'N/A')})...")
            eval_item = evaluate_with_api(row, adjustable_params)
            if eval_item is None:
                continue

            # Save individual text file for this evaluation
            header_info = (
                f"response_id: {row.get('response_id', '')}\n"
                f"stay_id: {row.get('stay_id', '')}\n"
                f"ICU_mortality: {row.get('ICU_mortality', '')}\n"
                f"Predicted_risk(%): {row.get('Predicted_risk(%)', '')}\n"
                f"risk_group: {row.get('risk_group', '')}\n"
```

```
                f"ML_model: {row.get('ML_model', '')}\n"
                f"LLM_model: {row.get('LLM_model', '')}\n"
                f"Prompt: {row.get('Prompt', '')}\n\n"
            )
            formatted_evaluation = format_evaluation(eval_item)
            prompt_str = row.get("Prompt", "").replace(" ", "_")
            txt_filename = f"{row.get('response_id',
'N/A')}_{prompt_str}_{adjustable_params['MODEL_NAME']}_{today_date}.txt"
            txt_filepath = os.path.join(base_dir, txt_filename)
            try:
                with open(txt_filepath, 'w', encoding='utf-8') as f:
                    f.write(header_info + formatted_evaluation)
            except Exception as e:
                print(f"Error writing text file for response_id
{row.get('response_id', '')}: {e}")

            # Build the combined record for CSV
            combined_record = {
                "response_id": row.get("response_id", ""),
                "stay_id": row.get("stay_id", ""),
                "ICU_mortality": row.get("ICU_mortality", ""),
                "Predicted_risk(%)": row.get("Predicted_risk(%)", ""),
                "risk_group": row.get("risk_group", ""),
                "ML_model": row.get("ML_model", ""),
                "LLM_model": row.get("LLM_model", ""),
                "Prompt": row.get("Prompt", ""),
                "RAG": row.get("RAG", "None"),
                "prompt_tokens": row.get("prompt_tokens", ""),
                "completion_tokens": row.get("completion_tokens", ""),
                "total_tokens": row.get("total_tokens", ""),
                "eva_prompt_token": eval_item.get("eva_prompt_token", ""),
                "eva_completion_token": eval_item.get("eva_completion_token",
""),
                "eva_total_token": eval_item.get("eva_total_token", ""),
                "LLM_response_t(s)": row.get("LLM_response_t(s)", ""),
                "LLM_waiting_t(s)": row.get("LLM_waiting_t(s)", ""),
                "Python_time(s)": row.get("Python_time(s)", ""),
                "Total_time(s)": row.get("Total_time(s)", ""),
```

```
                "LLM_evaluation_time(s)":
eval_item.get("LLM_evaluation_time(s)", ""),
                "Interpretation": eval_item.get("Interpretation", {}).get("subtotal",
0),
                "Examination": eval_item.get("Examination", {}).get("subtotal",
0),
                "Management": eval_item.get("Management", {}).get("subtotal",
0),
                "Follow_up": eval_item.get("Follow_up", {}).get("subtotal", 0),
                "Summary": eval_item.get("Summary", {}).get("subtotal", 0),
                "Total": eval_item.get("Total", ""),
                "Interpretation_JSON":
json.dumps(eval_item.get("Interpretation", {})),
                "Examination_JSON": json.dumps(eval_item.get("Examination",
{})),
                "Management_JSON":
json.dumps(eval_item.get("Management", {})),
                "Follow_up_JSON": json.dumps(eval_item.get("Follow_up", {})),
                "Summary_JSON": json.dumps(eval_item.get("Summary", {}))
            }

            # Append the record to the CSV file immediately
            try:
                with open(csv_output_path, 'a', newline='', encoding='utf-8') as
csvfile:
                    writer = csv.DictWriter(csvfile,
fieldnames=summary_columns)
                    writer.writerow(combined_record)
            except Exception as e:
                print(f"Error appending CSV for response_id
{row.get('response_id', '')}: {e}")

            time.sleep(adjustable_params["API_SLEEP_SECONDS"])

    print("Evaluation completed.")
    print(f" - Individual text files saved in: {base_dir}")
    print(f" - Consolidated summary CSV saved to: {csv_output_path}")
```

```
# ------------------------------------------------------------------------
# Example Usage & Adjustable Parameters
# ------------------------------------------------------------------------
if __name__ == "__main__":
    adjustable_params = {
        "INPUT_FILE_PATH":
r"C:\workspace\001_GAI\Evaluation\LLM_Responses.csv",    # Path to the
input CSV
        "CSV_OUTPUT_FILENAME": r"Evaluation_Summary.csv",    # Will
be overridden by new naming
        "MODEL_NAME": "o3-mini-101",    # Set to your Azure deployment
name
        "MAX_TOKENS": 4000,
        "OPENAI_API_KEY": " YOUR KEY ",    # Replace with your Azure
subscription key
        "AZURE_API_VERSION": "2024-12-01-preview",     # New
adjustable parameter for Azure API version
        "AZURE_ENDPOINT": " YOUR AZURE URI ",    # New adjustable
parameter for Azure endpoint
        "API_SLEEP_SECONDS": 1.0,
        "REASONING_EFFORT": "medium",
        "MAX_RETRIES": 3,
        "TIMEOUT": 300    # Timeout set to 300 seconds
    }

    process_evaluations(adjustable_params)
```

5.2 Examples of system prompt of E1, E2, E3

System Prompt for Scoring for Interpretation	
E1 **Standard**	You are an expert clinical evaluator with extensive clinical and evidence-based practice experience. Your task is to thoroughly analyze the LLM response provided for the Interpretation section and provide a detailed numerical scoring breakdown.
E2 **Top-Down**	You are an expert clinical evaluator with extensive clinical and evidence-based practice experience. Your task is to thoroughly analyze the LLM response provided for the Interpretation section and provide a detailed numerical scoring breakdown. Per scoring item, evaluate **top-down**: if level 4 is met, score 4; if not, check level 3 for a 3; then level 2 for a 2; then level 1 for a 1; else, score 0.
E3 **Bottom-up**	You are an expert clinical evaluator with extensive clinical and evidence-based practice experience. Your task is to thoroughly analyze the LLM response provided for the Interpretation section and provide a detailed numerical scoring breakdown. Per scoring item, evaluate **bottom-up**: first check if level 0 is met; then sequentially check for levels 1, 2, 3, and 4, assigning the highest level satisfied.

6. Python script for calculating intraclass correlation coefficient

```
import pandas as pd
import pingouin as pg
import numpy as np
from tqdm.auto import trange

# ————————————————————————————————————————————————————————————————————————

# 1. Load and reshape the data
# ————————————————————————————————————————————————————————————————————————

file_path = r"C:\workspace\001_GAI\ICC\ICC_test.xlsx"
data_wide = pd.read_excel(file_path, engine="openpyxl")

# Long (tidy) format ⇒ one row per rating
data_long = (
    data_wide
    .melt(id_vars=["Patient"], var_name="Rater", value_name="Score")
    .dropna(subset=["Score"])                    # safety in case of blanks
)

# ————————————————————————————————————————————————————————————————————————

# 2. Point estimates for every ICC type
# ————————————————————————————————————————————————————————————————————————

keep_types = ["ICC1", "ICC2", "ICC3", "ICC1k", "ICC2k", "ICC3k"]

icc_point = (
    pg.intraclass_corr(
        data=data_long,
        targets="Patient",
        raters="Rater",
        ratings="Score",
```

```
            nan_policy="omit",
        )
        .query("Type in @keep_types")
        .reset_index(drop=True)
)

# ──────────────────────────────────────────────────────────────────────────────
# 3. Bootstrap 95 % confidence intervals (1 000 resamples)
# ──────────────────────────────────────────────────────────────────────────────
def bootstrap_ci(df_long, n_boot=1000, seed=42):
    """Return dicts {type: low_ci}, {type: high_ci} via non-parametric bootstrap."""
    rng        = np.random.default_rng(seed)
    patients   = df_long["Patient"].unique()
    boot_vals  = {t: [] for t in keep_types}

    for _ in trange(n_boot, desc="Bootstrapping", leave=False):
        # (a) sample patients WITH replacement
        sampled = rng.choice(patients, size=len(patients), replace=True)

        # (b) gather all ratings for those patients
        boot_df = df_long[df_long["Patient"].isin(sampled)].copy()

        # (c) re-index patients consecutively so Pingouin treats duplicates correctly
        boot_df["Patient"] = pd.factorize(boot_df["Patient"])[0]

        # (d) compute ICCs for this bootstrap sample
        icc_boot = pg.intraclass_corr(
            data=boot_df,
            targets="Patient",
            raters="Rater",
            ratings="Score",
            nan_policy="omit",
        )
```

```
            for t in keep_types:
                boot_vals[t].append(icc_boot.loc[icc_boot["Type"] == t,
"ICC"].iloc[0])

    # percentiles → CI limits
    ci_low  = {t: np.percentile(boot_vals[t], 2.5)  for t in keep_types}
    ci_high = {t: np.percentile(boot_vals[t], 97.5) for t in keep_types}
    return ci_low, ci_high

ci_low, ci_high = bootstrap_ci(data_long, n_boot=1000, seed=42)

# ──────────────────────────────────────────────────────────────────────
# 4. Assemble final table
# ──────────────────────────────────────────────────────────────────────
icc_point["CI_low"]  = icc_point["Type"].map(ci_low)
icc_point["CI_high"] = icc_point["Type"].map(ci_high)

# sanity-check: every bound must bracket the point estimate
for _, r in icc_point.iterrows():
    assert r["CI_low"] <= r["ICC"] <= r["CI_high"], f"CI problem in {r.Type}"

# rounding / formatting per user request
icc_point["ICC"]       = icc_point["ICC"].round(3)
icc_point["CI_low"]    = icc_point["CI_low"].round(4)
icc_point["CI_high"] = icc_point["CI_high"].round(4)
icc_point["CI95%"]     = icc_point.apply(
    lambda r: f"[{r.CI_low:.4f}, {r.CI_high:.4f}]", axis=1
)
icc_point["p value"] = icc_point["pval"].apply(
    lambda p: "P < 0.0001" if p < 0.0001 else f"P = {p:.4f}"
)

# ──────────────────────────────────────────────────────────────────────
# 5. Print the tidy summary
```

```
# ──────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────────────

print(
    icc_point.loc[:, ["Type", "ICC", "CI95%", "F", "p value"]]
                .to_string(index=False)
)
```