



Universidade do Minho

Licenciatura em Engenharia e Gestão de Sistemas de Informação (LEGSI)

Ano letivo 2024/25

## Técnicas de Inteligência Artificial

### Relatório de Projeto de Sistemas Inteligentes de Apoio à Decisão LCD

PL2 – Grupo 25

#### Projeto 1



**Rute Silva**  
**A79431**



**David Gusmão**  
**A102343**



**João Lourenço**  
**A101376**



**Leonel Pinheiro**  
**A103716**



## Índice

<b>1.</b>	<b>Introdução .....</b>	<b>1</b>
<b>1.1</b>	<b>Enquadramento .....</b>	<b>1</b>
<b>1.2</b>	<b>Objetivos.....</b>	<b>1</b>
<b>2.</b>	<b>Execução do Projeto .....</b>	<b>2</b>
<b>2.1</b>	<b>Organização e Distribuição de Tarefas.....</b>	<b>3</b>
<b>2.2</b>	<b>Contributo Individual .....</b>	<b>4</b>
<b>2.2.1</b>	<b>Rute Silva .....</b>	<b>4</b>
<b>2.2.2</b>	<b>David Gusmão .....</b>	<b>4</b>
<b>2.2.3</b>	<b>João Lourenço.....</b>	<b>5</b>
<b>2.2.4</b>	<b>Leonel Pinheiro.....</b>	<b>5</b>
<b>3.</b>	<b>P1 - Tarefa A .....</b>	<b>7</b>
<b>4.</b>	<b>P1 - Tarefa B .....</b>	<b>7</b>
<b>5.</b>	<b>P2 .....</b>	<b>9</b>
<b>6.</b>	<b>Conclusões.....</b>	<b>10</b>
<b>6.1</b>	<b>Síntese.....</b>	<b>10</b>
<b>6.2</b>	<b>Discussão .....</b>	<b>10</b>
<b>6.3</b>	<b>Funcionamento do Trabalho em grupo .....</b>	<b>10</b>
<b>Anexo A .....</b>		<b>11</b>
<b>Anexo B .....</b>		<b>45</b>



## Índice de figuras

Figura 1- Diagrama de Gantt .....	3
Figura 2 - Base de dados utilizada no RapidMiner .....	8
Figura 3 - Decision tree.....	8
Figura 4 - Decision Tree .....	8
Figura 5 - Estrutura do Dataset de train em formato JSON .....	9
Figura 6 - Processo de treino do Modelo Qwen2.5-3B .....	9



# 1. Introdução

## 1.1 Enquadramento

O presente documento enquadra-se no âmbito da unidade curricular de Técnicas de Inteligência Artificial (TIA), inserida no segundo semestre da Licenciatura em Engenharia de Sistemas e Tecnologias de Informação. O Projeto 1 do Grupo 25, tem como propósito a conceção e implementação de um Sistema Baseado em Conhecimento (SBC), utilizando a linguagem de programação lógica *Prolog*, com o objetivo de apoiar o processo de triagem clínica de doentes em contexto de serviço de urgência hospitalar.

Este sistema baseia-se no protocolo de triagem de *Manchester*, adotado em larga escala nas unidades de saúde portuguesas, o qual permite classificar os doentes que recorrem ao serviço de urgência de acordo com diferentes níveis de prioridade, procurando assegurar uma resposta eficiente e adequada à gravidade clínica apresentada. O projeto encontra-se dividido em duas partes:

- **Parte A:** Aquisição manual de conhecimento e sua representação através de regras de produção, organizadas numa base de conhecimento estruturada;
- **Parte B:** Aquisição automática de conhecimento, com base numa base de dados de triagens simuladas e extração de regras por métodos de aprendizagem automática.
- **Projeto 2** - A utilização do protocolo de *Manchester* é amplamente reconhecida em contextos clínicos para garantir que pacientes sejam avaliados e tratados de acordo com a gravidade dos seus sintomas. Neste contexto, o uso de técnicas avançadas de inteligência artificial, como Large Language Model (LLM), representa uma inovação significativa capaz de aprimorar a precisão e eficiência na classificação das prioridades médicas, especialmente em situações de emergência e urgência.

O trabalho foi desenvolvido em grupo, promovendo a colaboração entre os elementos na recolha, formalização e implementação do conhecimento, assim como na construção da lógica de inferência e da interface com o utilizador final. Para tal foram escolhidos os fluxogramas Traumatismo Crânio-Encefálico (TCE), Trauma Maior, Quedas, Overdose/Envenenamento e Dor Torácica por serem situações clínicas frequentes, representando um espectro abrangente de cenários (traumáticos, tóxicos e clínicos), garantindo assim relevância prática e abrangência clínica.

## 1.2 Objetivos

O presente trabalho de projeto tem como objetivo central o desenvolvimento de um Sistema Baseado em Conhecimento (SBC) para suporte à decisão clínica no âmbito da triagem de doentes em serviços de urgência hospitalar, com base nos critérios definidos pelo protocolo de triagem de *Manchester*. Este sistema deverá permitir, de forma automatizada, a atribuição de um nível de prioridade clínica ao doente, orientando a tomada de decisão dos profissionais de saúde de forma estruturada, fiável e consistente.

- **Fase A – Aquisição e Representação Manual de Conhecimento:**

Esta fase visa a modelação do conhecimento clínico mediante a formalização de regras de produção em linguagem *Prolog*, com base na análise dos fluxogramas de decisão previstos no protocolo de *Manchester*. Pretende-se construir uma base de



conhecimento que reflete o raciocínio clínico inerente ao processo de triagem, permitindo a sua operacionalização através de um sistema de inferência lógico. A interface do sistema deverá possibilitar uma interação dinâmica com o utilizador, conduzindo-o através de um conjunto encadeado de questões que culminam na determinação de uma cor de triagem.

- **Fase B – Aquisição Automática de Conhecimento:**

A segunda fase propõe a utilização de técnicas de aprendizagem automática para a indução de regras de triagem a partir de uma base de dados sintética, composta por exemplos clínicos previamente definidos. Esta abordagem visa complementar a base de conhecimento construída manualmente, promovendo uma visão híbrida entre o conhecimento pericial e o conhecimento extraído a partir de dados. As regras obtidas serão posteriormente traduzidas e integradas na estrutura lógica do sistema desenvolvido na fase anterior.

- **Projeto 2 (P2) – Suporte à decisão na Triagem de doentes no serviço de urgências baseado em LLM:**

O objetivo principal do Projeto 2 (P2) é implementar e avaliar um sistema de suporte à decisão clínica utilizando um LLM, ajustado especificamente ao protocolo de triagem de Manchester, visando automatizar e aperfeiçoar o processo de classificação dos pacientes conforme a urgência médica.

## 2. Execução do Projeto

A execução do projeto iniciou-se com a seleção criteriosa de fluxogramas clínicos a utilizar no seu desenvolvimento, tendo em conta os diferentes quadros clínicos que frequentemente motivam episódios de urgência hospitalar. Partindo do protocolo de triagem de Manchester, foram selecionados os seguintes fluxogramas pelo grupo de trabalho:

- TCE (Traumatismo Crânio-Encefálico)
- Trauma Maior
- Quedas
- Overdose e Envenenamento
- Dor Torácica

A seleção destes fluxogramas procurou assegurar uma representatividade equilibrada entre causas traumáticas, clínicas e toxicológicas, permitindo construir uma base de conhecimento com aplicabilidade prática mais transversal.

Posteriormente, e com o objetivo de estruturar o raciocínio clínico de forma coerente, recorremos à mnemónica C(ABDE). Esta abordagem, amplamente utilizada por profissionais de saúde para avaliação primária de doentes em contexto de urgência, permitiu organizar o processo de triagem segundo um modelo sistemático:

- **C – Circulation (Circulação)**
- **A – Airway (Via Aérea)**
- **B – Breathing (Respiração)**
- **D – Disability (Défice neurológico)**

- E – Exposure (Exposição e exame global)

Esta estrutura foi utilizada como base para o desenvolvimento do fluxo de decisão, garantindo uma avaliação lógica, sequencial e compatível com a prática clínica em contexto de urgência. Com base nesta organização, foram definidas as variáveis de decisão relevantes para cada fluxo clínico, formuladas as questões dirigidas ao enfermeiro triador, e determinados os critérios de atribuição de prioridade para cada resposta possível.

Estas variáveis foram posteriormente codificadas como factos e regras de produção na linguagem Prolog, permitindo simular o comportamento de um perito clínico no processo de tomada de decisão. Para cada fluxo, foi desenhado um encadeamento lógico de perguntas e condições, respeitando os critérios clínicos definidos no protocolo e adaptando-os à representação computacional.

A estrutura modular do sistema foi organizada em três componentes principais:

- **base\_de\_conhecimento.pl**: onde se encontram os factos clínicos, regras de decisão e estrutura lógica das variáveis;
- **motor\_de\_inferencia.pl**: responsável pelo encadeamento lógico das inferências, aplicando as regras com base nas respostas do utilizador;
- **interface.pl**: módulo que gera a interação com o utilizador, apresentando as questões e recolhendo as respostas, até se atingir uma decisão final (pulseira de prioridade).

## 2.1 Organização e Distribuição de Tarefas

A execução do projeto foi organizada de forma colaborativa entre os quatro elementos do grupo, tendo sido definida uma estrutura de trabalho baseada em tarefas principais, distribuídas de forma equitativa, tendo sido possível ajustar esta distribuição àquilo que seriam as preferências individuais dos vários elementos. A abordagem adotada privilegiou a cooperação contínua e a revisão cruzada dos contributos, assegurando coerência e rigor técnico ao longo de todas as fases do projeto. Apresentamos, de seguida, o Diagrama de Gantt desenvolvido mediante as 10 fases do projeto.



Figura 1- Diagrama de Gantt



## 2.2 Contributo Individual

### 2.2.1 Rute Silva

A participação no desenvolvimento do Projeto 1 focou-se, numa fase inicial, na **estruturação lógica do sistema**, garantindo a fidelidade do raciocínio clínico ao protocolo de triagem de *Manchester* e a coerência do fluxo de decisão aplicado à realidade hospitalar. Participei na seleção e estudo aprofundado dos fluxogramas clínicos a integrar no sistema, optando por: TCE, Trauma Maior, Quedas, Overdose/Envenenamento e Dor Torácica. Esta análise permitiu definir o âmbito do projeto e estabelecer os critérios de triagem a formalizar. Numa fase posterior, foram elaboradas as questões e definida a lógica clínica do sistema segundo a mnemónica C(ABDE), frequentemente utilizada em contexto de urgência. Este modelo orientou o fluxo das questões dirigidas ao profissional de saúde, assegurando a sua estruturação. Posteriormente, foi desenvolvido o conteúdo clínico da triagem e respetivos critérios de inferência (critérios de diagnóstico e atribuição de pulseiras). Este material serviu de base para a posteriormente elaborada base de conhecimento, traduzindo o raciocínio clínico em regras de produção *Prolog*, assegurando a aplicação das prioridades definidas. Foi ainda garantido que o comportamento do sistema garantia a lógica de triagem previamente desenvolvida através do motor de inferência desenvolvido e da interface codificada.

Codificação desenvolvida:

- Codificação da base de conhecimento *Prolog* (*base\_de\_conhecimento.pl*);
- Estruturação lógica dos fluxos segundo a mnemónica C(ABDE);
- Lógica da interface (*interface.pl*);
- Lógica do sistema de inferência (*motor\_de\_inferencia.pl*);
- Colaboração na criação da base de dados simulada;
- Colaboração na integração das novas regras extraídas automaticamente;
- Organização e redação do relatório de projeto;
- Colaboração na realização do projeto 2;
- Realização de testes ao LLM treinado.

### 2.2.2 David Gusmão

O meu contributo para o desenvolvimento do Projeto 1 concentrou-se inicialmente na definição e estruturação lógica do sistema, garantindo a correspondência precisa com o protocolo de triagem de *Manchester* e a adequação ao contexto real dos serviços hospitalares. Realizei a seleção criteriosa e o estudo detalhado dos fluxogramas clínicos essenciais ao sistema, incluindo especificamente Trauma Crânio-Encefálico (TCE), Trauma Maior, Quedas, Overdose / Envenenamento e Dor Torácica. Este trabalho permitiu delimitar claramente o âmbito do projeto e estabelecer os critérios concretos a serem implementados.

Em seguida, participei na elaboração das questões estruturadas e na definição da lógica clínica do sistema, alinhada com a metodologia da mnemónica C(ABDE), amplamente adotada em contextos de emergência médica.

Posteriormente, realizei a codificação das regras de produção em *Prolog*, assegurando que o sistema refletisse adequadamente o raciocínio clínico estabelecido anteriormente.



No Projeto 2 (P2), assumi a juntamente os restantes elementos a conceção e execução do módulo baseado em Large Language Model (LLM), que permitiram a construção de um sistema robusto de triagem automatizada com base em linguagem natural.

Codificação desenvolvida:

- Codificação da base de conhecimento *Prolog* (*base\_de\_conhecimento.pl*);
- Estruturação lógica dos fluxos segundo a mnemónica C(ABCDE);
- Lógica da interface (*interface.pl*);
- Lógica do sistema de inferência (*motor\_de\_inferencia.pl*);
- Colaboração na criação da base de dados simulada;
- Colaboração na integração das novas regras extraídas automaticamente;
- Organização e redação do relatório de projeto.
- Criação dos datasets em formato JSON (*train*, *valid* e *test*), a partir do fluxo clínico de quedas;
- Configuração e execução o fine-tuning do modelo Qwen2.5-3B com recurso à técnica LoRA (Low-Rank Adaptation), utilizando o ambiente Google Colab;
- Realização de testes ao LLM treinado.

### 2.2.3 João Lourenço

A minha participação no Projeto 1 centrou-se sobretudo na elaboração da Parte B, dedicada à extração automática de conhecimento. Criei a base de dados em Excel com combinações de sintomas e classificações, a partir da qual, com o RapidMiner, foram geradas regras de decisão traduzidas depois para Prolog, assegurando a coerência com a lógica do sistema. Também desenvolvi as perguntas interativas da Parte B, facilitando a interação do utilizador com o sistema.

Paralelamente, colaborei em tarefas da Parte A, como na criação das perguntas do fluxo de triagem e definição da lógica de funcionamento. Contribuí ainda na organização e redação do relatório final.

Codificação e Tarefas Desenvolvidas:

- Elaboração da base de dados simulada para aprendizagem automática;
- Geração da base de conhecimento da Parte B com o apoio do RapidMiner;
- Tradução das regras aprendidas para Prolog;
- Elaboração das perguntas interativas para a Parte B;
- Colaboração no desenvolvimento das perguntas da Parte A;
- Participação na estruturação lógica do sistema;
- Apoio na redação e revisão do relatório do projeto;
- Colaboração na realização do projeto 2;
- Realização de testes ao LLM treinado.

### 2.2.4 Leonel Pinheiro

Neste projeto, participei em ambas as fases de desenvolvimento.

Na Parte A, colaborei na elaboração das questões associadas aos fluxos clínicos e contribuí na resolução de diversos problemas relacionados com a lógica em *Prolog*, garantindo a coerência do sistema de perguntas e da estrutura de inferência.



Foquei-me na parte B, concentrei o meu trabalho na implementação do motor de regras baseado em exemplos, desenvolvendo o código responsável pela interpretação automática das regras a partir de dados (CSV) e integração no sistema em *Prolog*. Com a colaboração do meu colega João Lourenço, criámos a base de dados de exemplos e procedemos à extração das regras clínicas utilizando o *RapidMiner*, que foram depois adaptadas ao formato lógico usado no sistema.

Codificação e Tarefas Desenvolvidas:

- Elaboração da base de dados simulada para aprendizagem automática;
- Geração da base de conhecimento da Parte B com o apoio do *RapidMiner*;
- Tradução das regras aprendidas para *Prolog*;
- Elaboração das perguntas interativas para a Parte B;
- Colaboração no desenvolvimento das perguntas da Parte A;
- Participação na estruturação lógica do sistema;
- Apoio na redação e revisão do relatório do projeto;
- Colaboração na realização do projeto 2;
- Realização de testes ao LLM treinado.



### 3. P1 - Tarefa A

Nesta fase inicial, foi realizada uma análise detalhada e seleção criteriosa dos fluxogramas clínicos relevantes, com base no protocolo de triagem de *Manchester*. Os fluxogramas escolhidos — Traumatismo Crânio-Encefálico (TCE), Trauma Maior, Quedas, Overdose/Envenenamento e Dor Torácica — representam cenários frequentes e diversificados, abrangendo contextos traumáticos, tóxicos e clínicos, o que assegurou uma cobertura ampla e representativa das situações que frequentemente chegam aos serviços de urgência.

Posteriormente, com o objetivo de estruturar o conhecimento clínico de forma coerente e sistemática, adotámos a mnemónica C(ABDE), reconhecida e amplamente utilizada por profissionais de saúde em contexto de emergência. Esta abordagem permitiu segmentar logicamente o processo de triagem em cinco fases sequenciais essenciais: Circulação (Circulation), Via aérea (Airway), Respiração (Breathing), Déficit neurológico (Disability) e Exposição (Exposure). Esta metodologia orientou claramente o desenvolvimento da codificação recorrendo ao *Prolog* (ficheiro `base_de_conhecimento.pl`), agrupando as perguntas clínicas e respetivos critérios inferenciais em estruturas lógicas consistentes e facilmente interpretáveis.

Foram selecionadas e definidas as variáveis clínicas fundamentais para cada fluxo escolhido, bem como as respetivas questões direcionadas aos profissionais de saúde e os critérios específicos para atribuição dos diferentes níveis de prioridade clínica (vermelha, laranja, amarela, verde e azul), de acordo com o nível de urgência atribuído. As questões foram cuidadosamente implementadas, conforme pode verificar-se no ficheiro `interface.pl`, permitindo uma interação dinâmica, estruturada e eficaz com o utilizador durante a simulação do processo de triagem.

Na etapa seguinte, as regras de produção correspondentes aos critérios clínicos previamente estabelecidos foram rigorosamente codificadas na linguagem *Prolog*. Cada regra foi devidamente comentada e organizada, conforme o ficheiro `base_de_conhecimento.pl`, garantindo assim transparência e facilidade de manutenção futura. Paralelamente, desenvolvemos um motor de inferência robusto, implementado no ficheiro `motor_de_inferencia.pl`, capaz de aplicar logicamente as regras produzidas e inferir automaticamente as decisões clínicas adequadas, com base nas respostas do utilizador.

Por fim, a implementação de uma interface interativa (`interface.pl`) foi essencial para garantir uma interação fluida, permitindo aos utilizadores responder a questões objetivas e obter rapidamente as classificações das pulseiras. Esta etapa do projeto demonstrou-se eficaz ao proporcionar um mecanismo intuitivo e acessível para apoiar profissionais de saúde na tomada de decisões clínicas em situações reais de urgência hospitalar.

### 4. P1 - Tarefa B

A parte B tem como objetivo a extração automática de dados, ou seja, a partir dos resultados obtidos, gerar conhecimento. Nesta etapa, a equipa utilizou apenas um fluxo — o fluxo “queda” — aproveitando parcialmente a interface desenvolvida na parte A. Posteriormente, a base de conhecimento foi modificada, sendo gerada automaticamente através do *RapidMiner*.

A equipa começou por desenvolver uma folha de *Excel*, apresentada abaixo, contendo os sintomas do fluxo selecionado. Isso consistiu na criação de uma tabela com todas as combinações possíveis e as respetivas conclusões, para que, posteriormente, fosse inserida no



*RapidMiner*. Dessa forma, foi possível criar uma base de conhecimento automática com base nas combinações fornecidas no Excel, a qual foi então implementada em *Prolog*.

Row No.	hipotensao_...	hipotensao_...	convulsions_...	glicemia_bai...	queda_sup_...	queda_sup_...	hipotermia	dor_intensa	historia_de_i...	diminuir_da...	historia_disc...	fratura_expo...	dor_modera...	dor_ligeira	edema	pulseira
1	FALSO	FALSO	FALSO	FALSO	VERDADEIRO	VERDADEIRO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	laranja	
2	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	VERDADEIRO	FALSO	FALSO	FALSO	FALSO	FALSO	amarela	
3	FALSO	FALSO	FALSO	FALSO	VERDADEIRO	VERDADEIRO	VERDADEIRO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	laranja	
4	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	vermelho	
5	FALSO	FALSO	FALSO	FALSO	VERDADEIRO	FALSO	FALSO	VERDADEIRO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	laranja	
6	FALSO	FALSO	FALSO	FALSO	VERDADEIRO	FALSO	VERGEMERO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	laranja	
7	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	VERDADEIRO	VERDADEIRO	FALSO	FALSO	FALSO	FALSO	amarela	
8	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	verde	
9	FALSO	VERDADEIRO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	vermelho	
10	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	VERDADEIRO	verde	
11	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	VERDADEIRO	FALSO	FALSO	VERDADEIRO	VERDADEIRO	FALSO	FALSO	amarela	
12	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	VERDADEIRO	verde	
13	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	VERDADEIRO	VERDADEIRO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	laranja	
14	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	VERDADEIRO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	laranja	
15	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	VERDADEIRO	VERDADEIRO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	laranja	
16	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	VERDADEIRO	VERDADEIRO	FALSO	FALSO	amarela	
17	FALSO	FALSO	VERDADEIRO	VERDADEIRO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	FALSO	vermelho	

Figura 2 - Base de dados utilizada no *RapidMiner*

Aprofundando os processos dentro do *RapidMiner*, após a inserção do ficheiro Excel, foi criado um processo composto por três subprocessos complementares: *Retrieve*, *Set Role* e *Rule Induction*.

No subprocesso *Retrieve*, os dados desenvolvidos pela equipa no *Excel* são importados e carregados para o ambiente de trabalho do *RapidMiner*.

Em seguida, no subprocesso *Set Role*, os atributos são processados, atribuindo-se um papel específico a cada dado carregado, conferindo-lhes significado dentro do contexto da análise.

Por fim, o subprocesso *Rule Induction* é responsável por gerar regras a partir dos dados analisados, que são então convertidas no *Rule Model* (Figura 3), originando assim uma nova base de conhecimento.

Process



Figura 3 - Decision tree

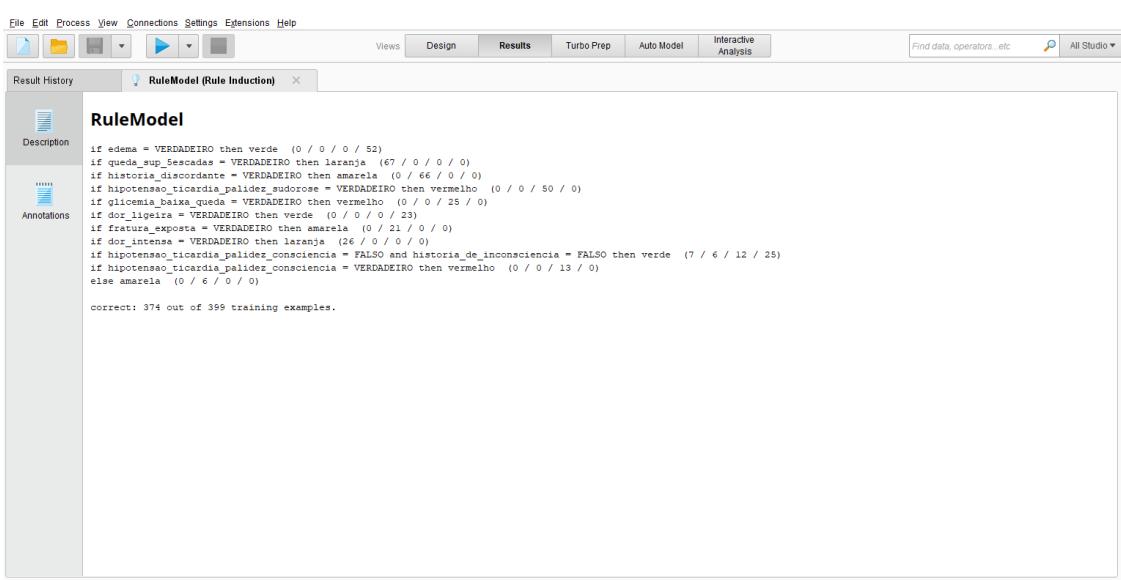


Figura 4 - Decision Tree



## 5.P2

Um Large Language Model (LLM) é uma tecnologia avançada de Inteligência Artificial treinada em vastos conjuntos de dados textuais, capaz de compreender, gerar e interagir utilizando linguagem natural. Esses modelos são tipicamente fundamentados na arquitetura Transformer e podem realizar diversas tarefas, incluindo geração de texto, tradução automática, sumarização e resposta a perguntas (Q&A).

Os datasets estruturados (train.json, valid.json, test.json) foram criados com base no fluxo de queda, escolhido especificamente por sua clareza, abrangência de condições médicas críticas e representatividade das situações clínicas frequentemente encontradas em emergências hospitalares.

```
{  
    "text": "Paciente 21 com dor torácica súbita intensa com confusão.\nResposta: Vermelho. Convulsão ativa. Suporte avançado.",  
},  
{  
    "text": "Paciente 22 com hipotermia com temperatura 39.5°C.\nResposta: Laranja. Dor intensa. Analgesia EV e priorizar consulta.",  
},  
{  
    "text": "Paciente 23 com história de perda de consciência breve após queda.\nResposta: Amarelo. Dor moderada. Analgesia oral e reavaliação.",  
},  
{  
    "text": "Paciente 24 com edema localizado em joelho.\nResposta: Verde. Acompanhamento em consulta.",  
},  
{  
    "text": "Paciente 25 com dúvidas sobre vacinação.\nResposta: Azul. Sem sintomas significativos. Orientação e alta.",  
},  
}
```

Figura 5 - Estrutura do Dataset de train em formato JSON

Para a execução do projeto, foram escolhidas ferramentas como Google Colab, devido à sua acessibilidade e recursos computacionais adequados para fine-tuning de modelos de grande escala. O modelo de LLM utilizado foi o Qwen2.5-3B, selecionado pela sua eficácia em inferências rápidas e precisas, especialmente após o fine-tuning com a técnica Low-Rank Adaptation (LoRA), que permite o ajuste fino eficiente e rápido, mesmo em equipamentos computacionais limitados.

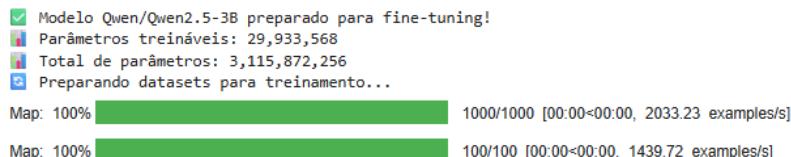


Figura 6 - Processo de treino do Modelo Qwen2.5-3B

A validação do sistema foi realizada com casos do dataset test.json, resultando em boa precisão na classificação dos casos clínicos. As avaliações consideraram precisão, coerência clínica com o protocolo original e tempo de resposta, que demonstraram grande adequação às exigências do contexto clínico de emergência.

Durante a execução do projeto, identificaram-se limitações na capacidade do LLM em lidar com casos muito específicos ou com dados insuficientes na fase de treinamento. Para além deste ponto, verificamos a limitação dos recursos das ferramentas open source (Google Colab), que no uso de Modelos mais exigentes como o Gemma 3 4b-it não eram estáveis.



## 6. Conclusões

### 6.1 Síntese

Este relatório apresenta o desenvolvimento de um projeto dividido em duas fases principais, com o objetivo de criar sistemas baseados em conhecimento aplicados a contextos clínicos. Na primeira parte, foi construído um sistema de triagem capaz de recomendar tratamentos com base nos sintomas fornecidos pelo utilizador, utilizando um motor de inferência com regras definidas manualmente. Na segunda parte, os dados foram processados de forma automática, gerando uma nova base de conhecimento com o auxílio da ferramenta *RapidMiner*, a partir de exemplos registados previamente.

O projeto 2 integrou com sucesso o modelo Qwen2.5-3B ajustado com LoRA, demonstrando grande precisão e eficiência na aplicação do protocolo de Manchester para triagem clínica.

### 6.2 Discussão

A equipa alcançou uma solução técnica coerente com os objetivos inicialmente definidos, respeitando o planeamento estabelecido. Nesta tarefa, foi desenvolvido um sistema funcional que recomenda tratamentos com base nos sintomas indicados pelo utilizador.

Embora a linguagem *Prolog* tenha representado um desafio inicial, a equipa conseguiu adaptar-se e estruturar corretamente os conhecimentos necessários, superando as dificuldades com empenho e trabalho colaborativo.

A utilização do LLM ajustado apresentou vantagens substanciais em relação às abordagens anteriores baseadas em regras estáticas, especialmente na interpretação contextual aprofundada exigida em emergências médicas.

### 6.3 Funcionamento do Trabalho em grupo

Desde o início do projeto, as tarefas foram distribuídas de forma equilibrada entre todos os membros da equipa, com responsabilidades bem definidas. O trabalho foi desenvolvido de forma colaborativa, recorrendo a reuniões online e sessões presenciais.

Cada elemento contribuiu de forma equitativa e ativa para a execução bem-sucedida do projeto, respeitando integralmente o contrato de grupo e os prazos estipulados. A equipa adotou uma atitude proativa e motivada ao longo de todo o processo, empenhando-se constantemente em melhorar a qualidade do trabalho e aprofundar os seus conhecimentos.

Todos os membros demonstraram empenho, interesse e espírito colaborativo, fatores que contribuíram decisivamente para o sucesso do projeto. Como tal, a equipa atribui-se, com base no desempenho alcançado, uma autoavaliação de 18 valores.



## Anexo A

De seguida apresentamos o código desenvolvido referente aos Projeto 1 e 2

### Projeto 1

#### Parte A

*basedeconhecimento.pl*

```
=====  
 PROTOCOLO DE TRIAGEM  
=====*/  
:- encoding(utf8).  
  
:- discontiguous perguntar/2.  
:- multifile perguntar/2.  
:- multifile perguntar/1.  
:- multifile atribuir_pulseira/2.  
:- multifile sintoma/1.  
:- multifile ja_atribuida/0.  
  
*****  
* 3) QUESTÕES DO SISTEMA DE TRIAGEM (NÃO ALTERAR)  
*****/  
  
% --- Fluxo Circulação (C) ---  
perguntar(hemorragia_visivel, 'O doente apresenta hemorragia visível ou refere perdas hemáticas significativas? (s/n)').  
perguntar(fluxo_abundante, 'A hemorragia é de grande volume? (s/n)').  
perguntar(nao_controlavel, 'A hemorragia não é controlável com compressão? (s/n)').  
perguntar(hipotensao, 'O doente apresenta hipotensão (TA<90/60 mmHg)? (s/n)').  
perguntar(taquicardia, 'O doente apresenta taquicardia (FC>100 bpm)? (s/n)').  
perguntar(palidez_cutanea, 'O doente apresenta palidez cutânea? (s/n)').  
perguntar(sudorese, 'O doente apresenta sudorese? (s/n)').  
perguntar(diminuicao_estado_consciencia, 'O doente apresenta diminuição do estado de consciência? (s/n)').  
perguntar(fluxo_moderado, 'A hemorragia é de volume moderado? (s/n)').  
perguntar(fluxo_ligeiro, 'A hemorragia é de volume ligeiro? (s/n)').  
perguntar(persistente, 'A hemorragia é persistente? (s/n)').  
perguntar(dificil_controlo, 'A hemorragia é de difícil controlo? (s/n)').  
  
% --- Fluxo A+B (Airway + Breathing) ---  
perguntar(dificuldade_respiratoria, 'O doente apresenta queixas ou sinais de dificuldade respiratória? (s/n)').  
perguntar(aspiracao_corpo_estranho, 'Há suspeita de aspiração de corpo estranho? (s/n)').  
perguntar(estridor, 'O doente apresenta estridor? (s/n)').  
perguntar(cianose, 'O doente apresenta cianose? (s/n)').  
perguntar(ausencia_sons_respiratorios, 'O doente apresenta ausência de sons respiratórios? (s/n)').  
perguntar(incapacidade_verbalizar, 'O doente apresenta incapacidade de verbalizar? (s/n)').  
perguntar(congestao Facial, 'O doente apresenta congestão facial? (s/n)').  
perguntar(spo2_baixa_90, 'SpO2 <90 % em ar ambiente? (s/n)').  
perguntar(bradipneia, 'O doente apresenta Bradipneia (FR<12cpm)? (s/n)').  
perguntar(taquipneia, 'O doente apresenta Taquipneia (FR>20 cpm)? (s/n)').  
perguntar(polipneia, 'O doente apresenta Polipneia? (s/n)').  
perguntar(padrao_chyne_stokes, 'O doente apresenta Padrão Respiratório de Cheyne-Stokes? (s/n)').  
perguntar(padrao_kussmaul, 'O doente apresenta Padrão Respiratório de Kussmaul? (s/n)').  
perguntar(padrao_biot, 'O doente apresenta Padrão Respiratório de Biot? (s/n)').  
perguntar(padrao_paradoxal, 'O doente apresenta Padrão Respiratório Paradoxal? (s/n)').  
perguntar(dispnquia, 'O doente apresenta Dispnquia? (s/n)').  
perguntar(tiragem, 'O doente apresenta Tiragem? (s/n)').  
perguntar(musculos_acessorios, 'O doente apresenta Recruta Músculos Respiratórios Acessórios na Ventilação? (s/n)').  
perguntar(ruidos_respiratorios, 'O doente apresenta Respiração Ruidosa (Sibilos, Crepitações ou Outro)? (s/n)').
```



% --- Fluxo D (Disability) ---

perguntar(alteracao\_estado\_consciencia, 'O doente apresenta alteração súbita do estado de consciência? (s/n)').  
 perguntar(ecg\_baixo, 'O doente apresenta diminuição do estado de consciência (ECG ( $\leq$ 13) ou -2)? (s/n)').  
 perguntar(menor\_idade, 'O doente apresenta idade <18 anos? (s/n)').  
 perguntar(defice\_neurologico, 'O doente apresenta evidência de défice neurológico? (s/n)').  
 perguntar(defice\_motor\_novo, 'O doente apresenta défice motor de novo? (s/n)').  
 perguntar(defice\_sensitivo\_novo, 'O doente apresenta défice sensitivo de novo? (s/n)').  
 perguntar(alteracao\_fala, 'O doente apresenta alteração da fala? (s/n)').  
 perguntar(desvio\_ocular, 'O doente apresenta anisocoria? (s/n)').  
 perguntar(desvio\_comissura\_labial, 'O doente apresenta desvio da comissura labial? (s/n)').  
 perguntar(agitacao, 'O doente apresenta agitação? (s/n)').  
 perguntar(verborreico, 'O doente apresenta verborreia? (s/n)').  
 perguntar(historia\_de\_inconsciencia, 'O doente apresenta história recente de perda de consciência? (s/n)').

% --- Fluxo Dor Torácica (DT) ---

perguntar(dor\_toracica, 'O doente apresenta dor torácica não traumática? (s/n)').  
 perguntar(hipotensao\_cardiogenico, 'O doente apresenta hipotensão (TA<90/60 mmHg)? (s/n)').  
 perguntar(taquicardia\_cardiogenico, 'O doente apresenta taquicardia (FC>100 bpm)? (s/n)').  
 perguntar(palidez\_cutanea\_cardiogenico, 'O doente apresenta palidez cutânea? (s/n)').  
 perguntar(sudorese\_cardiogenico, 'O doente apresenta sudorese? (s/n)').  
 perguntar(diminuicao\_estado\_consciencia\_cardiogenico, 'O doente apresenta diminuição do estado de consciência? (s/n)').  
 perguntar(bradicardia, 'O doente apresenta bradicardia (FC <60 bpm)? (s/n)').  
 perguntar(taquicardia\_cardio, 'O doente apresenta taquicardia (FC >100 bpm)? (s/n)').  
 perguntar(pulso\_arritmico, 'O doente apresenta pulso arrítmico? (s/n)').  
 perguntar(historia\_cardiaca\_importante, 'O doente apresenta história cardíaca importante (angina, enfarte agudo do miocárdio ou insuficiência cardíaca ou outro)? (s/n)').  
 perguntar(dor\_pleuritica, 'O doente apresenta dor pleurítica (agravamento com movimentos expiratórios? (s/n)').  
 perguntar(dor\_moderada, 'O doente apresenta dor moderada (Score <=7/10 e >=4/10 na escala numérica ou <=3 e >2 na escala fácies)? (s/n)').  
 perguntar(dor\_intensa, 'O doente apresenta dor intensa (Score >7/10 ou >3 na escala fácies)? (s/n)').  
 perguntar(vomitos, 'Apresenta história atual de vômitos? (s/n)').  
 perguntar(vomitos\_persistentes, 'Apresenta vômitos persistentes (>2)? (s/n)').  
 perguntar(dor\_ligeira, 'O doente apresenta dor ligeira (Score <3 na escala numérica ou <2 na escala fácies <2 dias)? (s/n)').  
 perguntar(evento\_recente, 'O evento decorreu, no máximo, há dois dias? (s/n)').

% --- Fluxo Exposure : Overdose ---

perguntar(ingestao\_substancias, 'Há evidência de ingestão (voluntária/acidental) de substâncias tóxicas? (s/n)').  
 perguntar(hipotensao\_exp, 'O doente apresenta hipotensão (TA<90/60 mmHg)? (s/n)').  
 perguntar(taquicardia\_exp, 'O doente apresenta taquicardia (FC>100 bpm)? (s/n)').  
 perguntar(palidez\_cutanea\_exp, 'O doente apresenta palidez cutânea? (s/n)').  
 perguntar(sudorese\_exp, 'O doente apresenta sudorese? (s/n)').  
 perguntar(diminuicao\_estado\_consciencia\_exp, 'O doente apresenta diminuição do estado de consciência? (s/n)').  
 perguntar(convulsionando, 'Apresenta crise epilética generalizada ou pós-ictal? (s/n)').  
 perguntar(glicemica\_baixa, 'Apresenta glicemias <70 mg/dL? (s/n)').  
 perguntar(risco\_nova\_autoagressao, 'Há risco de autoagressão? (s/n)').  
 perguntar(mortalidade\_alta, 'A substância ingerida apresenta suspeita ou risco alto de mortalidade? (s/n)').  
 perguntar(mortalidade\_moderada, 'A substância ingerida apresenta suspeita ou risco moderado de mortalidade? (s/n)').  
 perguntar(spo2\_baixa\_92, 'O doente apresenta SpO<sub>2</sub> <92% em ar ambiente? (s/n)').  
 perguntar(historia\_discordante, 'A história é de doença atual e discordante? (s/n)').  
 perguntar(historia\_psiquiatrica\_importante, 'Apresenta antecedente pessoal de foro psiquiátrico? (s/n)').  
 perguntar(agitacao\_psicomotora, 'Apresenta agitação psicomotora? (s/n)').  
 perguntar(ideacao\_suicida, 'Há evidência de ideação suicida? (s/n)').  
 perguntar(vontade\_viver\_comprometida, 'A vontade de viver está comprometida? (s/n)').  
 perguntar(vontade\_de\_infligir\_dor, 'Há evidência de vontade de infligir dor? (s/n)').  
 perguntar(confusao, 'Confusão? (s/n)').

% --- Fluxo Exposure : Queda ---

perguntar(queda, 'Há evidência de evento de queda? (s/n)').  
 perguntar(hipotensao\_queda, 'O doente apresenta hipotensão (TA<90/60)? (s/n)').



```

perguntar(taquicardia_queda, 'O doente apresenta taquicardia (FC>100)? (s/n)').  

perguntar(palidez_cutanea_queda, 'O doente apresenta palidez cutânea? (s/n)').  

perguntar(sudorese_queda, 'O doente apresenta sudorese? (s/n)').  

perguntar(diminuicao_estado_consciencia_queda, 'O doente apresenta diminuição do estado de consciência?  

(s/n)').  

perguntar(convulsionando_queda, 'Existe suspeita ou evidência de crise epilética? (s/n)').  

perguntar(glicemia_baixa_queda, 'Apresneta glicemia <70mg/dL? (s/n)').  

perguntar(queda_sup_1m, 'Apresentou queda de altura >1m? (s/n)').  

perguntar(queda_sup_5escadas, 'Apresentou queda de escadas >5 degraus? (s/n)').  

perguntar(hipotermia, 'Apresenta taur <35 °C? (s/n)').  

perguntar(disturbio_da_coagulacao, 'Apresenta distúrbio da coagulação? (s/n)').  

perguntar(fratura_exposta, 'Apresenta fratura exposta ou deformidade importante? (s/n)').  

perguntar(edema, 'Apresenta edema localizado? (s/n)').
```

% --- Fluxo Exposure : Trauma ---

```

perguntar(trauma, 'Existe evidência de envolvimento em evento acidental com risco de lesão? (s/n)').  

perguntar(hipotensao_trauma, 'O doente apresenta hipotensão (TA<90/60)? (s/n)').  

perguntar(taquicardia_trauma, 'O doente apresenta taquicardia (FC>100)? (s/n)').  

perguntar(palidez_cutanea_trauma, 'O doente apresenta palidez cutânea? (s/n)').  

perguntar(sudorese_trauma, 'O doente apresenta sudorese? (s/n)').  

perguntar(diminuicao_estado_consciencia_trauma, 'O doente apresenta diminuição do estado de consciência?  

(s/n)').  

perguntar(acidente_viacao, 'Houve envolvimento em acidente de viação? (s/n)').  

perguntar(projacao, 'Há história de evento de projeção? (s/n)').  

perguntar(ejecao, 'Há história de evento de ejeção? (s/n)').  

perguntar(agressao, 'Há história ou suspeita de agressão? (s/n)').  

perguntar(patologia_cardiaca, 'Apresenta patologia cardíaca? (s/n)').  

perguntar(patologia_renal, 'Apresenta patologia renal? (s/n)').  

perguntar(patologia_hepatica, 'Apresenta patologia hepática? (s/n)').  

perguntar(diabetes, 'Apresenta antecedente pessoal de diabetes mellitus? (s/n)').  

perguntar(cirurgia_recente, 'Apresenta antecedente pessoal de cirurgia recente? (s/n)').
```

% --- Fluxo Exposure : TCE ---

```

perguntar(lesao_cefalica, 'O doente apresenta hematoma, equimose visível ou lesão na região cefálica? (s/n)').  

perguntar(convulsionando_tce, 'O doente apresenta crise epilética tônico-clônica generalizada ou pós ictal? (s/n)').  

perguntar(hipotensao_tce, 'O doente apresenta hipotensão (TA<90/60)? (s/n)').  

perguntar(taquicardia_tce, 'O doente apresenta taquicardia (FC>100)? (s/n)').  

perguntar(palidez_cutanea_tce, 'O doente apresenta palidez cutânea? (s/n)').  

perguntar(sudorese_tce, 'O doente apresenta sudorese? (s/n)').  

perguntar(diminuicao_estado_consciencia_tce, 'O doente apresenta diminuição do estado de consciência? (s/n)').  

perguntar(glicemia_baixa_tce, 'Apresenta glicemia <70 mg/dL? (s/n)').  

perguntar(acidente_viacao_tce, 'Existe evidência de envolvimento em acidente de viação? (s/n)').  

perguntar(projacao_tce, 'Há história de evento de Projeção? (s/n)').  

perguntar(ejecao_tce, 'Há história de evento de ejeção? (s/n)').  

perguntar(agressao_tce, 'Há história ou suspeita de agressão? (s/n)').  

perguntar(hematoma_couro_cabeludo, 'Apresenta hematoma do couro cabeludo? (s/n)').  

perguntar(dor_moderada_tce, 'Apresenta dor moderada (Score <=7/10 e >=4/10 na escala numérica ou <=3 e >2  

na escala fácies)? (s/n)').  

perguntar(dor_intensa_tce, 'Apresenta dor intensa (Score >7/10 ou >3 na escala fácies)? (s/n)').  

perguntar(dor_ligeira_tce, 'Apresenta dor ligeira (Score <3 na escala numérica ou <2 na escala fácies <2 dias)?  

(s/n)').  

perguntar(perda_estado_consciencia_tce, 'Apresenta história de perda de consciência ? (s/n)').  

perguntar(historia_discordante, 'A história de doença atual é discordante? (s/n)').  

perguntar(disturbio_da_coagulacao_tce, 'Apresenta distúrbio da coagulação? (s/n)').
```

```

/*-----  

5) FLUXOS E SUB-FLUXOS  

-----*/
```

% ----- CIRCULAÇÃO -----

```

fluxo_circulacao :-  

    perguntar(hemorragia_visivel),  

    (sintoma(hemorragia_visivel) ->
```



```

perguntar(fluxo_abundante),
perguntar(nao_controlavel),
(sintoma(Fluxo_abundante),
sintoma(Nao_controlavel) ->
atribuir_pulseira(vermelha, hemorragia_exanguinante);
true),
\+ ja_atribuida,
perguntar(hipotensao),
(sintoma(Hipotensao),
perguntar(taquicardia),
sintoma(Taquicardia),
perguntar(palidez_cutanea),
%se tiver estes 3 + um dos proximos
(sintoma(Palidez_cutanea),
perguntar(sudorese),
sintoma(Sudorese));
perguntar(diminuicao_estado_consciencia),
sintoma(Diminuicao_estado_consciencia)) ->
atribuir_pulseira(vermelha, choque);
%atribui esta pulseira
true),
\+ ja_atribuida,
perguntar(Fluxo_moderado),
(sintoma(Fluxo_moderado) ->
atribuir_pulseira(laranja, hemorragia_maior_incontrolavel);
true),
\+ ja_atribuida,
perguntar(Fluxo_ligeiro),
(sintoma(Fluxo_ligeiro),
perguntar(persistente),
perguntar(dificil_controlo),
(sintoma(Persistente);
sintoma(Dificil_controlo)) ->
atribuir_pulseira(amarela, hemorragia_menor_incontrolavel);
true);
true).

```

% ----- AIRWAY + BREATHING -----

```

fluxo_ab :-
\+ ja_atribuida,
perguntar(dificuldade_respiratoria),
(sintoma(Dificuldade_respiratoria) ->
(perguntar(aspiracao_corpo_estrano),
(sintoma(Aspiracao_corpo_estrano) ->
atribuir_pulseira(vermelha, obstrucao_via_aerea);
true),
\+ ja_atribuida,
perguntar(incapacidade_verbalizar),
perguntar(cianose),
perguntar(congestao_facial),
(sintoma(Incacidade_verbalizar),
(sintoma(Cianose);
sintoma(Congestao_facial)) ->
atribuir_pulseira(vermelha, obstrucao_via_aerea);
true),
\+ ja_atribuida,
perguntar(estridor),
perguntar(cianose),
perguntar(congestao_facial),
(sintoma(Estridor),
(sintoma(Cianose);
sintoma(Congestao_facial)) ->

```



```

atribuir_pulseira(vermelha, obstrucao_via_aerea);
true),
\+ ja_atribuida,
perguntar(ausencia_sons_respiratorios),
(sintoma(ausencia_sons_respiratorios) ->
atribuir_pulseira(vermelha, obstrucao_via_aerea);
true),
\+ ja_atribuida,
perguntar(spo2_baixa_90),
(sintoma(spo2_baixa_90) ->
atribuir_pulseira(vermelha, respiracao_inadequada);
true),
\+ ja_atribuida,
perguntar(bradipneia),
(sintoma(bradipneia) ->
atribuir_pulseira(vermelha, respiracao_inadequada);
true),
\+ ja_atribuida,
perguntar(taquipneia),
(sintoma(taquipneia) ->
atribuir_pulseira(vermelha, respiracao_inadequada);
true),
\+ ja_atribuida,
perguntar(polipneia),
(sintoma(polipneia) ->
atribuir_pulseira(vermelha, respiracao_inadequada);
true),
\+ ja_atribuida,
perguntar(padrao_cheyne_stokes),
(sintoma(padrao_cheyne_stokes) ->
atribuir_pulseira(vermelha, respiracao_inadequada);
true),
\+ ja_atribuida,
perguntar(padrao_biot),
(sintoma(padrao_biot) ->
atribuir_pulseira(vermelha, respiracao_inadequada);
true),
\+ ja_atribuida,
perguntar(padrao_kussmaul),
(sintoma(padrao_kussmaul) ->
atribuir_pulseira(vermelha, respiracao_inadequada);
true),
\+ ja_atribuida,
perguntar(padrao_paradoxal),
(sintoma(padrao_paradoxal) ->
atribuir_pulseira(vermelha, respiracao_inadequada);
true),
\+ ja_atribuida,
perguntar(dispneia),
(sintoma(dispneia) ->
(perguntar(tiragem),
(sintoma(tiragem) ->
atribuir_pulseira(vermelha, respiracao_inadequada);
true),
\+ ja_atribuida,
perguntar(musculos_acessorios),
(sintoma(musculos_acessorios) ->
atribuir_pulseira(vermelha, respiracao_inadequada);
true),
\+ ja_atribuida,
perguntar(ruidos_respiratorios),
(sintoma(ruidos_respiratorios) ->
atribuir_pulseira(vermelha, respiracao_inadequada);

```



```

true));
true));
true% <- Se NÃO tiver dificuldade respiratória, simplesmente termina o fluxo_ab aqui e passa para o fluxo_d
).

% ----- DISABILITY -----
fluxo_d :-
    \+ ja_atribuida,
    perguntar(alteracao_estado_consciencia),
    (sintoma(alteracao_estado_consciencia) ->
        perguntar(ecg_baixo),
        perguntar(menor_idade),
        (sintoma(ecg_baixo),
            sintoma(menor_idade) ->
                atribuir_pulseira(vermelha, crianca_nao_reativa);
            true),
            \+ ja_atribuida,
            perguntar(defice_neurologico),
            (sintoma(defice_neurologico) ->
                perguntar(defice_motor_novo),
                (sintoma(defice_motor_novo) ->
                    atribuir_pulseira(laranja, defice_neurologico_agudo);
                    perguntar(defice_sensitivo_novo),
                    (sintoma(defice_sensitivo_novo) ->
                        atribuir_pulseira(laranja, defice_neurologico_agudo);
                        perguntar(alteracao_fala),
                        (sintoma(alteracao_fala) ->
                            atribuir_pulseira(laranja, defice_neurologico_agudo);
                            perguntar(desvio_ocular),
                            (sintoma(desvio_ocular) ->
                                atribuir_pulseira(laranja, defice_neurologico_agudo);
                                perguntar(desvio_comissura_labial),
                                (sintoma(desvio_comissura_labial) ->
                                    atribuir_pulseira(laranja, defice_neurologico_agudo);
                                    true))))),
                    \+ ja_atribuida,
                    perguntar(agitacao),
                    perguntar(verboreico),
                    (sintoma(agitacao);
                    sintoma(verboreico) ->
                        atribuir_pulseira(amarela, agitacao_psicomotora);
                        true),
                    \+ ja_atribuida,
                    perguntar(historia_de_inconsciencia),
                    (sintoma(historia_de_inconsciencia) ->
                        atribuir_pulseira(amarela, historia_de_inconsciencia);
                        true);
                    true).

% ----- DOR TORÁCICA -----
fluxo_dor_toracica :-
    \+ ja_atribuida,
    perguntar(dor_toracica),
    (sintoma(dor_toracica) ->
        % Verificar suspeita de choque cardiogênico
        perguntar(hipotensao_cardiogenico),
        perguntar(taquicardia_cardiogenico),
        perguntar(palidez_cutanea_cardiogenico),
        % Caso 1: Com sudorese
        (sintoma(hipotensao_cardiogenico),
            sintoma(taquicardia_cardiogenico),
            sintoma(palidez_cutanea_cardiogenico) ->
                perguntar(sudorese_cardiogenico),

```



```

(sintoma(sudorese_cardiogenico) ->
atribuir_pulseira(vermelha, choque_cardiogenico);
true);
true),
\+ ja_atribuida,
% Caso 2: Com diminuição do estado de consciência
(sintoma(hipotensao_cardiogenico),
sintoma(taquicardia_cardiogenico),
sintoma(palidez_cutanea_cardiogenico) ->
perguntar(diminuicao_estado_consciencia_cardiogenico),
(sintoma(diminuicao_estado_consciencia_cardiogenico) ->
atribuir_pulseira(vermelha, choque_cardiogenico);
true);
true),
\+ ja_atribuida,
% A. Frequência cardíaca alterada - avaliação individual
perguntar(bradicardia),
(sintoma(bradicardia) ->
atribuir_pulseira(laranja, pulso_anormal);
true),
\+ ja_atribuida,
perguntar(taquicardia_cardio),
(sintoma(taquicardia_cardio) ->
atribuir_pulseira(laranja, pulso_anormal);
true),
\+ ja_atribuida,
perguntar(pulso_arritmico),
(sintoma(pulso_arritmico) ->
atribuir_pulseira(laranja, pulso_anormal);
true),
\+ ja_atribuida,
% B. Dor intensa
perguntar(dor_intensa),
(sintoma(dor_intensa) ->
atribuir_pulseira(laranja, dor_intensa);
true),
\+ ja_atribuida,
% C. História cardíaca importante
perguntar(historia_cardiaca_importante),
(sintoma(historia_cardiaca_importante) ->
atribuir_pulseira(amarela, historia_cardiaca_importante);
true),
\+ ja_atribuida,
% D. Dor pleurítica
perguntar(dor_pleuritica),
(sintoma(dor_pleuritica) ->
atribuir_pulseira(amarela, dor_pleuritica);
true),
\+ ja_atribuida,
% E. Dor moderada
perguntar(dor_moderada),
(sintoma(dor_moderada) ->
atribuir_pulseira(amarela, dor_moderada);
true),
\+ ja_atribuida,
% F e G. Vômitos
perguntar(vomitos),
(sintoma(vomitos) ->
perguntar(vomitos_persistentes),
(sintoma(vomitos_persistentes) ->
atribuir_pulseira(amarela, vomitos_persistentes);
atribuir_pulseira(verde, vomitos));
% Verifica dor leve quando não tem vômitos

```



```

\+ ja_atribuida,
  perguntar(dor_ligeira),
  (sintoma(dor_ligeira) ->
atribuir_pulseira(verde, dor_ligeira);
true)),
\+ ja_atribuida,
% H. Verifica evento recente
perguntar(evento_recente),
(sintoma(evento_recente) ->
atribuir_pulseira(verde, evento_recente);
% Se não for evento recente e não tiver outros critérios
( \+ ja_atribuida ->
atribuir_pulseira(azul, sem_criterio);
true));
true).

% ----- EXPOSURE -----
fluxo_exposure :-
\+ ja_atribuida,
subfluxo_overdose,
\+ ja_atribuida,
subfluxo_queda,
\+ ja_atribuida,
subfluxo_trauma_maior,
\+ ja_atribuida,
subfluxo_tce.

/*-----
SUB-FLUXO 1 : OVERDOSE / INTOXICAÇÃO
-----*/
subfluxo_overdose :-
\+ ja_atribuida,
perguntar(ingestao_substancias),
(sintoma(ingestao_substancias) ->
/* --- Choque ----- */
\+ ja_atribuida,
perguntar(hipotensao_exp),
(sintoma(hipotensao_exp),
perguntar(taquicardia_exp),
sintoma(taquicardia_exp),
perguntar(palidez_cutanea_exp),
%se tiver estes 3 + um dos proximos
(sintoma(palidez_cutanea_exp),
perguntar(sudorese_exp),
sintoma(sudorese_exp);
perguntar(diminuicao_estado_consciencia_exp),
sintoma(diminuicao_estado_consciencia_exp)) ->
atribuir_pulseira(vermelha, choque_exp);
%atribui esta pulseira
true),
/* --- Convulsões / hipoglicemias ----- */
\+ ja_atribuida,
perguntar(convulsionando),
(sintoma(convulsionando) ->
atribuir_pulseira(vermelha, convulsionando);
true),
\+ ja_atribuida,
perguntar(glicemia_baixa),
(sintoma(glicemia_baixa) ->
atribuir_pulseira(vermelha, hipoglicemias_grave);
true),
/* --- Pulso anómalo ----- */
\+ ja_atribuida,

```



```

perguntar(bradicardia),
(sintoma(bradicardia) ->
atribuir_pulseira(laranja, pulso_anormal);
true),
\+ ja_atribuida,
perguntar(taquicardia_cardio),
(sintoma(taquicardia_cardio) ->
atribuir_pulseira(laranja, pulso_anormal);
true),
\+ ja_atribuida,
perguntar(pulso_arrytmico),
(sintoma(pulso_arrytmico) ->
atribuir_pulseira(laranja, pulso_anormal);
true),
/* --- Auto-agressão / risco mortalidade ----- */
\+ ja_atribuida,
perguntar(risco_nova_autoagressao),
(sintoma(risco_nova_autoagressao) ->
atribuir_pulseira(laranja, alto_risco_de_nova_autoagressao);
true),
\+ ja_atribuida,
perguntar(mortalidade_alta),
(sintoma(mortalidade_alta) ->
atribuir_pulseira(laranja, mortalidade_alta); \+ ja_atribuida,
perguntar(mortalidade_moderada),
(sintoma(mortalidade_moderada) ->
atribuir_pulseira(amarela, mortalidade_moderada);
true)),
/* --- Critérios dispersos de gravidade moderada ----- */
\+ ja_atribuida,
perguntar(spo2_baixa_92),
(sintoma(spo2_baixa_92) ->
atribuir_pulseira(amarela, spo2_baixa_92);
true),
\+ ja_atribuida,
perguntar(historia_discordante),
(sintoma(historia_discordante) ->
atribuir_pulseira(amarela, historia_discordante);
true),
\+ ja_atribuida,
perguntar(historia_psiquiatrica_importante),
(sintoma(historia_psiquiatrica_importante) ->
atribuir_pulseira(amarela, historia_psiquiatrica_importante);
true),
\+ ja_atribuida,
perguntar(agitacao_psicomotora),
(sintoma(agitacao_psicomotora) ->
atribuir_pulseira(amarela, agitacao_psicomotora);
true),
\+ ja_atribuida,
perguntar(historia_de_inconsciencia),
(sintoma(historia_de_inconsciencia) ->
atribuir_pulseira(amarela, historia_de_inconsciencia);
true);
true).

/*
-----*
SUB-FLUXO 2 : QUEDA
-----*/
subfluxo_queda :-
\+ ja_atribuida,
perguntar(queda),
(sintoma(queda) ->

```



```

/* Choque por queda -----
\+ ja_atribuida,
perguntar(hipotensao_queda),
(sintoma(hipotensao_queda),
  perguntar(taquicardia_queda),
  sintoma(taquicardia_queda),
  perguntar(palidez_cutanea_queda),
  %se tiver estes 3 + um dos proximos
  (sintoma(palidez_cutanea_queda),
    perguntar(sudorese_queda),
    sintoma(sudorese_queda));
perguntar(diminuicao_estado_consciencia_queda),
  sintoma(diminuicao_estado_consciencia_queda)) ->
atribuir_pulseira(vermelha, choque_queda);
%atribui esta pulseira
true),
/* Convulsões / hipoglicemias -----
\+ ja_atribuida,
perguntar(convulsionando_queda),
(sintoma(convulsionando_queda) ->
atribuir_pulseira(vermelha, convulsionando_queda);
true),
\+ ja_atribuida,
perguntar(glicemia_baixa_queda),
(sintoma(glicemia_baixa_queda) ->
atribuir_pulseira(vermelha, hipoglicemias_grave);
true),
/* Mecanismo de alto impacto -----
\+ ja_atribuida,
perguntar(queda_sup_1m),
(sintoma(queda_sup_1m) ->
atribuir_pulseira(laranja, mecanismo_trauma_significativo);
(\+ ja_atribuida,
  perguntar(queda_sup_5escadas),
  (sintoma(queda_sup_5escadas) ->
atribuir_pulseira(laranja, mecanismo_trauma_significativo);
true))),
\+ ja_atribuida,
perguntar(hipotermia),
(sintoma(hipotermia) ->
atribuir_pulseira(laranja, hipotermia);
true),
\+ ja_atribuida,
perguntar(dor_intensa),
(sintoma(dor_intensa) ->
atribuir_pulseira(laranja, dor_intensa);
true),
/* Critérios amarelos -----
\+ ja_atribuida,
perguntar(historia_de_inconsciencia),
perguntar(disturbio_da_coagulacao),
perguntar(historia_discordante),
((sintoma(historia_de_inconsciencia);
sintoma(disturbio_da_coagulacao);
sintoma(historia_discordante)) ->
atribuir_pulseira(amarela, historia_inconsistencia_coagulacao);
true),
\+ ja_atribuida,
perguntar(fratura_exposta),
(sintoma(fratura_exposta) ->
atribuir_pulseira(amarela, fratura_exposta);
true),
\+ ja_atribuida,

```



```

perguntar(dor_moderada),
(sintoma(dor_moderada) ->
atribuir_pulseira(amarela, dor_moderada);
true),
/* Verde -----
\+ ja_atribuida,
perguntar(dor_ligeira),
perguntar(edema),
(sintoma(dor_ligeira));
sintoma(edema) ->
atribuir_pulseira(verde, dor_ligeira_edema);
true);
true).

/*
-----*
SUB-FLUXO 3 : TRAUMA MAIOR
-----*/
subfluxo_trauma_maior :-
\+ ja_atribuida,
perguntar(trauma),
(sintoma(trauma) ->
/* Choque -----
\+ ja_atribuida,
perguntar(hipotensao_trauma),
(sintoma(hipotensao_trauma),
perguntar(taquicardia_trauma),
sintoma(taquicardia_trauma),
perguntar(palidez_cutanea_trauma),
%se tiver estes 3 + um dos proximos
(sintoma(palidez_cutanea_trauma),
perguntar(sudorese_trauma),
sintoma(sudorese_trauma);
perguntar(diminuicao_estado_consciencia_trauma),
sintoma(diminuicao_estado_consciencia_trauma)) ->
atribuir_pulseira(vermelha, choque_trauma);
%atribui esta pulseira
true),
/* Mecanismo trauma significativo -----
\+ ja_atribuida,
perguntar(acidente_viacao),
(sintoma(acidente_viacao) ->
atribuir_pulseira(laranja, mecanismo_trauma_significativo);
(\+ ja_atribuida,
perguntar(projacao),
(sintoma(projacao) ->
atribuir_pulseira(laranja, mecanismo_trauma_significativo);
(\+ ja_atribuida,
perguntar(ejacao),
(sintoma(ejacao) ->
atribuir_pulseira(laranja, mecanismo_trauma_significativo);
(\+ ja_atribuida,
perguntar(agressao),
(sintoma(agressao) ->
atribuir_pulseira(laranja, mecanismo_trauma_significativo);
true))))),
\+ ja_atribuida,
perguntar(dor_intensa),
(sintoma(dor_intensa) ->
atribuir_pulseira(laranja, dor_intensa);
true),
/* Amarelos -----
\+ ja_atribuida,
perguntar(historia_de_inconsciencia),

```



```
(sintoma(historia_de_inconsciencia) ->
atribuir_pulseira(amarela, comorbilidades);
( \+ ja_atribuida,
    perguntar(disturbio_da_coagulacao),
    (sintoma(disturbio_da_coagulacao) ->
atribuir_pulseira(amarela, comorbilidades);
( \+ ja_atribuida,
    perguntar(patologia_cardiaca),
    (sintoma(patologia_cardiaca) ->
atribuir_pulseira(amarela, comorbilidades);
( \+ ja_atribuida,
    perguntar(patologia_renal),
    (sintoma(patologia_renal) ->
atribuir_pulseira(amarela, comorbilidades);
( \+ ja_atribuida,
    perguntar(patologia_hepatica),
    (sintoma(patologia_hepatica) ->
atribuir_pulseira(amarela, comorbilidades);
( \+ ja_atribuida,
    perguntar(diabetes),
    (sintoma(diabetes) ->
atribuir_pulseira(amarela, comorbilidades);
( \+ ja_atribuida,
    perguntar(cirurgia_recente),
    (sintoma(cirurgia_recente) ->
atribuir_pulseira(amarela, comorbilidades);
true))))))))),
\+ ja_atribuida,
perguntar(dor_moderada),
(sintoma(dor_moderada) ->
atribuir_pulseira(amarela, dor_moderada);
true);
true).
```

```
/*
-----*
SUB-FLUXO 4 : TCE (Traumatismo Crânio-Encefálico)
-----*/
subfluxo_tce :-
    \+ ja_atribuida,
    perguntar(lesao_cefalica),
    (sintoma(lesao_cefalica) ->
        % Convulsão
        \+ ja_atribuida,
        perguntar(convulsionando_tce),
        (sintoma(convulsionando_tce) ->
            atribuir_pulseira(vermelha, convolutionando_tce);
        true),
        % Choque
        \+ ja_atribuida,
        perguntar(hipotensao_tce),
        (sintoma(hipotensao_tce),
            perguntar(taquicardia_tce),
            sintoma(taquicardia_tce),
            perguntar(palidez_cutanea_tce),
            % Se tiver estes 3 + um dos próximos
            (sintoma(palidez_cutanea_tce),
                perguntar(sudorese_tce),
                sintoma(sudorese_tce));
            perguntar(diminuicao_estado_consciencia_tce),
                sintoma(diminuicao_estado_consciencia_tce)) ->
            atribuir_pulseira(vermelha, choque);
        true),
        % Hipoglicemia
```



```

\+ ja_atribuida,
perguntar(glicemia_baixa_tce),
(sintoma(glicemia_baixa_tce) ->
atribuir_pulseira(vermelha, hipoglicemica);
true),
% Mecanismo de trauma significativo
\+ ja_atribuida,
perguntar(acidente_viacao_tce),
(sintoma(acidente_viacao_tce) ->
atribuir_pulseira(laranja, mecanismo_trauma_significativo); \+ ja_atribuida,
perguntar(projecao_tce),
(sintoma(projecao_tce) ->
atribuir_pulseira(laranja, mecanismo_trauma_significativo)); \+ ja_atribuida,
perguntar(ejecao_tce),
(sintoma(ejecao_tce) ->
atribuir_pulseira(laranja, mecanismo_trauma_significativo)); \+ ja_atribuida,
perguntar(agressao_tce),
(sintoma(agressao_tce) ->
atribuir_pulseira(laranja, mecanismo_trauma_significativo));
true))),
% Dor intensa
\+ ja_atribuida,
perguntar(dor_intensa_tce),
(sintoma(dor_intensa_tce) ->
atribuir_pulseira(laranja, dor_intensa);
true),
% Critérios para pulseira amarela
\+ ja_atribuida,
perguntar(perda_estado_consciencia_tce),
(sintoma(perda_estado_consciencia_tce) ->
atribuir_pulseira(amarela, historia_de_inconsciencia_disturbio_de_coagulacao_comorbilidade);
true),
\+ ja_atribuida,
perguntar(historia_discordante),
(sintoma(historia_discordante) ->
atribuir_pulseira(amarela, historia_de_inconsciencia_disturbio_de_coagulacao_comorbilidade));
true),
\+ ja_atribuida,
perguntar(disturbio_da_coagulacao_tce),
(sintoma(disturbio_da_coagulacao_tce) ->
atribuir_pulseira(amarela, historia_de_inconsciencia_disturbio_de_coagulacao_comorbilidade));
true),
\+ ja_atribuida,
perguntar(dor_moderada_tce),
(sintoma(dor_moderada_tce) ->
atribuir_pulseira(amarela, dor_moderada));
true),
% Vômitos e outros critérios verdes - corrigido para evitar o aninhamento incorreto
\+ ja_atribuida,
perguntar(vomitos),
(sintoma(vomitos) ->
perguntar(vomitos_persistentes),
(sintoma(vomitos_persistentes) ->
atribuir_pulseira(amarela, vomitos_persistentes);
atribuir_pulseira(verde, vomitos));
true),
% Verificações adicionais quando não se atribuiu ainda pulseira
\+ ja_atribuida,
perguntar(hematoma_couro_cabeludo),
(sintoma(hematoma_couro_cabeludo) ->
atribuir_pulseira(verde, hematoma_couro_cabeludo));
true),
\+ ja_atribuida,

```



```

perguntar(dor_ligeira_tce),
(sintoma(dor_ligeira_tce) ->
atribuir_pulseira(verde, dor_ligeira);
true),
% Evento recente
\+ ja_atribuida,
perguntar(evento_recente),
(sintoma(evento_recente) ->
atribuir_pulseira(verde, evento_recente));
% Não adicionamos pulseira azul aqui - será feito na função iniciar/0
true);
true).

```

### *interface.pl*

```
: - discontiguous perguntar/2.
```

```

perguntar(Sintoma) :-
    perguntar(Sintoma, Texto),
    format('`w ', [Texto]),
    read(R),
    (R == s -> asserta(sintoma(Sintoma)); true).

```

### *motor\_de\_inferencia.pl*

```

:- encoding(utf8).

:- consult('interface.pl').
:- consult('base_de_conhecimento.pl').

:- dynamic sintoma/1, pulseira_atribuida/1.
:- discontiguous iniciar/0.

limpar_sintomas :-
    retractall(sintoma(_)).
ja_atribuida :-
    pulseira_atribuida(_).

atribuir_pulseira(Cor, Motivo) :-
    \+ pulseira_atribuida(_),
    asserta(pulseira_atribuida(Cor)),
    format(`n>> PULSEIRA `w `— motivo: `w`n', [Cor, Motivo]),
    !.

iniciar :-
    limpar_sintomas,
    retractall(pulseira_atribuida(_)),
    format('== INÍCIO DA TRIAGEM ==`n', []),
    fluxo_circulacao,
    (ja_atribuida ->
    !;
    true),
    fluxo_ab,
    (ja_atribuida ->
    !;
    true),
    fluxo_d,
    (ja_atribuida ->
    !;
    true),

```



```

fluxo_dor_toracica,
(ja_atribuida ->
!;
true),
fluxo_exposure,
(ja_atribuida ->
!;
true),
( \+ pulseira_atribuida(_) ->
atribuir_pulseira(azul, sem_criterio);
true).

```

## Parte B

```

% =====
% base_conhecimento.pl
% • Regras de triagem (RuleModel) no formato "if ... then ..."
% • Regra default
% =====

:- set_prolog_flag(encoding, utf8).

/* --- operadores e expansão para regra/2 -----
:- op(950, fx, if).      % prefixo
:- op(940, xfx, then).   % infixo
:- dynamic  regra/2.
:- multifile user:term_expansion/2.

user:term_expansion((if Cond then Classe), regra(Classe, L)) :-
    ( is_list(Cond) -> L = Cond ; L = [Cond] ).

/* --- RULE MODEL -----
if edema          then verde.
if queda_sup_5escadas  then laranja.
if historia_discordante then amarela.
if hipotensao_ticardia_palidez_sudorose  then vermelho.
if glicemias_baixa_queda then vermelho.
if dor_ligeira      then verde.
if fratura_exposta  then amarela.
if dor_intensa       then laranja.
if [not(hipotensao_ticardia_palidez_consciencia),
    not(historia_de_inconsciencia)]  then verde.
if hipotensao_ticardia_palidez_consciencia then vermelho.

/* --- else -----
default(amarelo).

```

### *interface.pl*

```

:- set_prolog_flag(encoding, utf8).
:- consult('sistema_inferencia.pl').
:- consult('base_conhecimento.pl').
:- consult('casos.pl').
:- consult('aprendizagem.pl').
:- learn_all.

/* Exemplos de teste q1 ... q7 (chame ?- qN(Cor). )*/
q1(C) :- classify([hipotensao_ticardia_palidez_sudorose, convulsionando_queda], C),
        format('q1 - Hipotensao+Ticardia+Palidez+Sudorese e com convulsões pós queda. → ~w~n', [C]),
q2(C) :- classify([hipotensao_ticardia_palidez_consciencia], C),

```



```

        format('q2 - Hipotensao+Ticardia+Palidez+Consciênciā → ~w~n', [C]).
q3(C) :- classify([convulsionando_queda], C),
        format('q3 - Convulsão pós-queda → ~w~n', [C]).
q4(C) :- classify([queda_sup_1m], C),
        format('q4 - Queda > 1 metro → ~w~n', [C]).
q5(C) :- classify([fratura_exposta, dor_ligeira], C),
        format('q5 - Fratura exposta e dor ligeira → ~w~n', [C]).
q6(C) :- classify([dor_intensa], C),
        format('q6 - Dor intensa → ~w~n', [C]).
q7(C) :- classify([dor_ligeira, edema], C),
        format('q7 - Dor ligeira e edema isolado → ~w~n', [C]).

/* =====
Perguntas interativas (TODAS as originais, sem alterações)
===== */
pergunta(hipotensao_ticardia_palidez_sudorose,
'Houve queda e verificou-se hipotensão, taquicardia, palidez e sudorese?').
pergunta(hipotensao_ticardia_palidez_consciencia,
'Houve queda e verificou-se hipotensão, taquicardia, palidez e diminuição do estado de consciência?').
pergunta(convulsionando_queda,
'Está convulsionando após a queda?').
pergunta(glicemias_baixa_queda,
'Está com glicemias baixa (<70 mg/dL) após a queda?').
pergunta(queda_sup_1m,
'A queda foi de mais de 1 metro?').
pergunta(queda_sup_5escadas,
'A queda foi de mais de 5 escadas/metros?').
pergunta(hipotermia,
'Está hipotérmico (<35 °C) após a queda?').
pergunta(dor_intensa,
'A dor é intensa (score > 7/10 ou > 3 na escala faces?)?').
pergunta(historia_de_inconsciencia,
'Houve história de inconsciência?').
pergunta(disturbio_da_coagulacao,
'Tem distúrbio de coagulação conhecido?').
pergunta(historia_discordante,
'A história é discordante?').
pergunta(fratura_exposta,
'Existe fratura exposta?').
pergunta(dor_moderada,
'A dor é moderada?').
pergunta(dor_ligeira,
'A dor é ligeira?').
pergunta(edema,
'Há apenas edema leve, sem outros sinais de gravidade?').

listar_perguntas(L) :- findall(Id, pergunta(Id,_), L).

/* ----- utilitário "sim"/"nao" ----- */
positivo(sim). positivo(s). positivo(y). positivo(yes).
negativo(no). negativo(n). negativo(nao). negativo(não).

le_bool(Texto, Bool) :-
repeat,
format('~w (sim/nao): ', [Texto]),
read(In), downcase_atom(In, Lc),
( positivo(Lc) -> Bool = true, !
; negativo(Lc) -> Bool = false, !
; writeln('Responda "sim" ou "nao".'), fail ).

/* ----- diálogo completo (com aprendizagem automática por defeito) ----- */
triagem_queda :-
nl, writeln('--- Triagem Queda---'),

```



```

retractall(regra(_,_)),           % Limpa regras anteriores
consult('base_conhecimento.pl'),   % Recarrega regras manuais
consult('casos.pl'),             % Recarrega exemplos
learn_all,                      % Executa aprendizagem automática
listar_perguntas(L), perguntar_ate_sim(L, SintomaOuVazio),
( SintomaOuVazio == [] -> Pos=[] ; Pos=[SintomaOuVazio] ),
classify(Pos, Cor),
format('\n>> Pulseira atribuída: ~w', [Cor]), nl,
( default(Cor) -> writeln('(Regra default atribuída)') ; true ), nl.

/* perguntar_ate_sim*/
perguntar_ate_sim([],[]).
perguntar_ate_sim([S|R], Res) :-
    pergunta(S, Txt), le_bool(Txt, Bool),
    ( Bool == true -> Res = S          % primeiro sim
    ; Bool == false -> perguntar_ate_sim(R, Res) ). % avança sem repetir

/* ----- menu ----- */
menu :-
    writeln(' TRIAGEM QUEDA '),
    writeln('Exemplos: q1(C)...q7(C).'),
    writeln('Questões: triagem_queda.').

```

### *sistema\_inferencia.pl*

```

% =====
% sistema_inferencia.pl
% • Motor genérico que carrega o RuleModel
% • Predicado classify/2 devolve a pulseira
% =====

:- set_prolog_flag(encoding, utf8).

% Carregar todas as fontes de regras
:- consult('base_conhecimento.pl').      % Regras manuais
:- consult('aprendizagem.pl').           % Módulo de aprendizagem
:- consult('casos.pl').                 % Exemplos históricos

% Declarar que as regras são dinâmicas
:- dynamic regra/2.

% --- alias <= para regra/2 -----
:- op(900, xfx, '<=').
Classe <= Cond :- regra(Classe, Cond).

% --- satisfazer condição -----
holds(Atrib, not(S)) :- \+ member(S, Atrib), !.
holds(Atrib, S)   :-  member(S, Atrib).

satisfy(Atrib, Cond) :- forall(member(C, Cond), holds(Atrib, C)).

% --- classificador -----
% Tenta regras aprendidas (conjunções - lista de listas)
classify(Sintomas, Classe) :-
    regra(Classe, Conds),
    is_list(Conds), Conds \= [],
    [First | _] = Conds, is_list(First), % Verifica se é lista de listas
    member(Conj, Conds),
    satisfy(Sintomas, Conj), !.

% Tenta regras manuais (listas simples)

```



```
classify(Sintomas, Classe) :-  
    regra(Classe, Cond),  
    is_list(Cond), \+ (Cond = [F|_], is_list(F)),  
    satisfy(Sintomas, Cond), !.  
  
% Regra default  
classify(_, Classe) :-  
    default(Classe).
```



## Projeto 2

*codigo\_google\_colab.ipynb*

```
# ===== DOCUMENTAÇÃO: PROTOCOLO DE MANCHESTER PARA QUEDAS =====  
# Este bloco fornece a descrição completa do protocolo de Manchester (queda)  
# e apresenta a interface inicial do sistema ao usuário.
```

```
.....  
# PROTOCOLO DE MANCHESTER - FLUXOGRAMA DE QUEDAS
```

O Sistema de Triagem de Manchester é um método de classificação de urgência amplamente utilizado em serviços de emergência hospitalares. No caso específico de quedas, o protocolo avalia fatores como mecanismo do trauma, sintomas apresentados, e riscos associados para atribuir uma prioridade de atendimento ao paciente.

## NÍVEIS DE TRIAGEM

### 1. VERMELHO (Emergência) - Atendimento Imediato

- Critérios para quedas:
  - \* Obstrução de vias aéreas
  - \* Respiração inadequada
  - \* Choque (hipotensão, taquicardia, alteração da consciência)
  - \* Convulsões ativas
  - \* Hipoglicemias severas (<55mg/dL)
  - \* Hemorragia exsanguinante

### 2. LARANJA (Muito Urgente) - Atendimento em até 10 minutos

- Critérios para quedas:
  - \* Mecanismo de trauma significativo (queda >1m altura)
  - \* Dor intensa (8-10/10)
  - \* Déficit neurológico agudo
  - \* Alteração súbita da consciência
  - \* Hipotermia (<35°C)

### 3. AMARELO (Urgente) - Atendimento em até 60 minutos

- Critérios para quedas:
  - \* História de inconsciência após a queda
  - \* Distúrbio de coagulação (uso de anticoagulantes)
  - \* Fratura exposta ou deformidade importante
  - \* Dor moderada (4-7/10)
  - \* Déficit neurológico novo (não agudo)

### 4. VERDE (Pouco Urgente) - Atendimento em até 120 minutos

- Critérios para quedas:
  - \* Dor leve (1-3/10)
  - \* Edema ou hematoma localizado
  - \* Evento recente sem sinais de alarme

### 5. AZUL (Não Urgente) - Atendimento em até 240 minutos

- Casos sem critérios para classificações anteriores
- Sem sinais de risco

## REGRAS ESPECÍFICAS PARA QUEDAS

#### ### Avaliação de Risco Crítico (Vermelho)

- Verificar sinais de choque: FC>100, PA<90/60, sudorese, palidez
- Avaliar padrão respiratório e saturação de O2
- Verificar nível de consciência (Escala de Glasgow)
- Avaliar glicemia capilar se houver alteração da consciência



### Mecanismo de Trauma (Laranja)  
 - Quedas de altura superior a 1 metro  
 - Quedas em escadas com mais de 5 degraus  
 - Trauma penetrante ou com transferência de energia (colisões)

### Fatores de Alto Risco (Amarelo)  
 - Uso de anticoagulantes (varfarina, DOAC)  
 - Idade >65 anos com comorbidades  
 - História de perda de consciência após a queda  
 - Histórico discordante (suspeita de abuso em populações vulneráveis)

### Características Específicas do Fluxograma  
 - Considerar sempre o pior cenário ao atribuir a prioridade  
 - Reavaliar o paciente se houver deterioração clínica  
 - Para cada discriminador, seguir a árvore de decisão do fluxograma  
 - Documentar claramente a justificativa da classificação

Este protocolo específico para quedas é fundamental para identificar rapidamente pacientes com risco potencial de lesões graves, como traumatismo crânioencefálico, fraturas, hemorragias internas ou comprometimento neurológico.

.....

```
# Imprimir um resumo para o usuário ao executar o notebook
print("=" * 80)
print("SISTEMA DE SUPORTE À DECISÃO - TRIAGEM DE MANCHESTER PARA QUEDAS")
print("=" * 80)
print("Este notebook implementa um modelo de linguagem (LLM) fine-tuned")
print("para auxiliar na classificação de pacientes segundo o Protocolo de Manchester,")
print("especificamente para o fluxograma de QUEDAS.")
print("\nO sistema atribui uma das cinco cores de prioridade:")
print(" - VERMELHO: Emergência (atendimento imediato)")
print(" - LARANJA: Muito Urgente (até 10 minutos)")
print(" - AMARELO: Urgente (até 60 minutos)")
print(" - VERDE: Pouco Urgente (até 120 minutos)")
print(" - AZUL: Não Urgente (até 240 minutos)")
print("\nModelo base utilizado:", "Gemma 2B com técnica LoRA para fine-tuning")
print("=" * 80)
```

# === BLOCO 1: Instalação de Dependências ===

```
%capture
!pip install --no-deps bitsandbytes accelerate xformers==0.0.29.post3 peft trl==0.15.2 triton cut_cross_entropy
unsloth_zoo
!pip install sentencepiece protobuf "datasets>=3.4.1" huggingface_hub hf_transfer
!pip install --no-deps unsloth

print("✅ Todas as bibliotecas foram instaladas com sucesso.")
```

# === BLOCO 2: Autenticação com o Hugging Face Hub ===

```
from huggingface_hub import login
import getpass
import os

hf_token = getpass.getpass("🔒 Dige seu token do Hugging Face (não será exibido): ")
login(token=hf_token)
os.environ["HUGGINGFACE_TOKEN"] = hf_token
print("✅ Login realizado com sucesso no Hugging Face Hub.")
```

# === BLOCO 3: Upload dos Datasets ===

```
from google.colab import files
from datasets import Dataset
```



```

import json
import os

# Upload dos arquivos JSON
print("📁 Faça upload dos arquivos: train.json, valid.json e test.json")
uploaded = files.upload()
print(f"Arquivos enviados: {list(uploaded.keys())}")

# Função para carregar um arquivo JSON
def load_json_file(prefix):
    filename = next((f for f in os.listdir() if f.startswith(prefix) and f.endswith(".json")), None)
    if not filename:
        print("🔴 Arquivo {prefix}.json não encontrado")
        return None, None

    with open(filename, "r", encoding="utf-8") as f:
        data = json.load(f)
    print(f"✅ Arquivo {filename} carregado com {len(data)} amostras")
    return data, filename

# Carrega os três conjuntos de dados
train_data, train_file = load_json_file("train")
valid_data, valid_file = load_json_file("valid")
test_data, test_file = load_json_file("test")

# Verifica se todos os dados foram carregados
if not all([train_data, valid_data, test_data]):
    print("🔴 Nem todos os arquivos foram carregados corretamente.")
else:
    # Converte para o formato Dataset do HuggingFace
    train_dataset = Dataset.from_list(train_data)
    valid_dataset = Dataset.from_list(valid_data)
    test_dataset = Dataset.from_list(test_data)

    # Padroniza os formatos dos dados para o Unslloth
    from unslloth.chat_templates import standardize_data_formats
    train_dataset = standardize_data_formats(train_dataset)
    valid_dataset = standardize_data_formats(valid_dataset)
    test_dataset = standardize_data_formats(test_dataset)

    print("\n✅ Datasets criados e padronizados com sucesso!")
    print("\nEstatísticas:")
    print(f"- Train dataset: {len(train_dataset)} amostras")
    print(f"- Validation dataset: {len(valid_dataset)} amostras")
    print(f"- Test dataset: {len(test_dataset)} amostras")

# === BLOCO 4: Análise dos Datasets ===
import random

# Função para mostrar um exemplo aleatório de um dataset
def show_random_example(dataset, name):
    print(f"\n🎲 Exemplo aleatório do {name}:")
    if len(dataset) > 0:
        random_idx = random.randint(0, len(dataset)-1)
        print(f"Índice: {random_idx}")
        print(dataset[random_idx])
    else:
        print(f"{name} está vazio")

# Mostra exemplos aleatórios de cada dataset
show_random_example(train_dataset, "Train Dataset")
show_random_example(valid_dataset, "Validation Dataset")

```



```

show_random_example(test_dataset, "Test Dataset")

# === BLOCO 5: Carregamento e Fine-Tuning do Qwen2.5-3B com Seus Dados ===
from transformers import AutoModelForCausalLM, AutoTokenizer, Trainer
from peft import LoraConfig, get_peft_model, prepare_model_for_kbit_training
import torch
import os
import re
from transformers.trainer_utils import IntervalStrategy
from transformers import TrainingArguments

print("📥 Carregando o modelo Qwen2.5-3B... ")

# Definir o dispositivo para GPU se disponível
device = "cuda" if torch.cuda.is_available() else "cpu"
print(f"💻 Utilizando dispositivo: {device}")

# Definir o modelo base a ser usado
model_id = "Qwen/Qwen2.5-3B"

# Carregar o tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_id)
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side = "right"

# Carregar o modelo base
model = AutoModelForCausalLM.from_pretrained(
    model_id,
    torch_dtype=torch.bfloat16,
    device_map="auto",
    load_in_4bit=True,
)

# Preparar o modelo para treinamento em 4-bit
model = prepare_model_for_kbit_training(model)

# Definir configuração LoRA para fine-tuning eficiente
lora_config = LoraConfig(
    r=16,           # Rank para LoRA (ajuste conforme necessário)
    lora_alpha=32,   # Fator de escala
    target_modules=["q_proj", "k_proj", "v_proj", "o_proj", "gate_proj", "up_proj", "down_proj"],
    lora_dropout=0.05,      # Dropout para regularização
    bias="none",       # Não aplica LoRA aos bias
    task_type="CAUSAL_LM"    # Tipo de tarefa (modelagem de linguagem causal)
)

# Aplicar configuração LoRA ao modelo
model = get_peft_model(model, lora_config)

print(f"✅ Modelo {model_id} preparado para fine-tuning!")
print(f"💡 Parâmetros treináveis: {model.num_parameters(True)}")
print(f"💡 Total de parâmetros: {model.num_parameters(False)}")

# Definir pasta para salvar o modelo
output_dir = "./modelo_triangem_manchester"
os.makedirs(output_dir, exist_ok=True)

# Preparar função de processamento dos dados para o formato que o modelo espera
def preprocess_function(examples):
    inputs = []

    for text in examples["text"]:

```



```

# Separar o caso do paciente da resposta esperada
match = re.match(r'(.*)[\n\r]*Resposta:(.*)', text, re.DOTALL)

if match:
    patient_case = match.group(1).strip()
    expected_response = match.group(2).strip()

    # Formato como instrução para o modelo
    formatted_text = f"<|im_start|>user\nAnalise o seguinte caso de triagem hospitalar e classifique conforme protocolo de Manchester:\n{patient_case}<|im_end|>\n<|im_start|>assistant\nResposta:{expected_response}<|im_end|>"
    inputs.append(formatted_text)
else:
    # Caso não consiga separar, usar o texto inteiro como instrução
    print(f"⚠️ Aviso: Não foi possível processar o exemplo: {text[:50]}...")
    formatted_text = f"<|im_start|>user\nAnalise o seguinte caso de triagem hospitalar e classifique conforme protocolo de Manchester:\n{text}<|im_end|>\n<|im_start|>assistant\n<|im_end|>"
    inputs.append(formatted_text)

# Tokenizar os inputs
tokenized_inputs = tokenizer(
    inputs,
    padding="max_length",
    truncation=True,
    max_length=512,
    return_tensors="pt"
)

# Preparar os labels (importantes para o treinamento de LLMs)
tokenized_inputs["labels"] = tokenized_inputs["input_ids"].clone()

return tokenized_inputs

# Aplicar preprocessamento aos datasets
print("🔄 Preparando datasets para treinamento...")

# Pré-processar datasets
train_dataset = train_dataset.map(preprocess_function, batched=True)
valid_dataset = valid_dataset.map(preprocess_function, batched=True)

# Definir parâmetros de treinamento - ajustando para os nomes de parâmetros corretos
training_args = TrainingArguments(
    output_dir=output_dir,
    num_train_epochs=3,           # Número de épocas de treinamento
    per_device_train_batch_size=2, # Tamanho do batch para treinamento
    per_device_eval_batch_size=2, # Tamanho do batch para avaliação
    gradient_accumulation_steps=4, # Passos de acumulação de gradiente
    warmup_steps=10,              # Passos de warmup
    learning_rate=5e-5,            # Taxa de aprendizado
    fp16=True,                   # Usar precisão mista para economizar memória
    logging_steps=10,              # Frequência de log
    eval_strategy="steps",        # Estratégia de avaliação (alterado de evaluation_strategy)
    eval_steps=50,                # Frequência de avaliação
    save_strategy="steps",        # Estratégia de salvamento
    save_steps=100,               # Frequência de salvamento
    save_total_limit=3,            # Limitar número de checkpoints salvos
    load_best_model_at_end=True,   # Carregar o melhor modelo no final
    report_to="none",             # Não reportar para serviços externos
    optim="adamw_torch",          # Otimizador
    lr_scheduler_type="cosine",    # Scheduler de learning rate
    seed=42,                      # Semente para reproduzibilidade
)

```



```

# Inicializar o trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=valid_dataset,
    tokenizer=tokenizer,
)

# Iniciar o treinamento
print("⌚ Iniciando o treinamento...")
trainer.train()

# Salvar o modelo final
print("💾 Salvando o modelo final...")
model.save_pretrained(f"{output_dir}/final")
tokenizer.save_pretrained(f"{output_dir}/final")

print("✅ Treinamento finalizado com sucesso!")

# Testar o modelo após o fine-tuning
test_prompt = "Paciente com dor torácica súbita intensa com palidez cutânea."
print(f"\n👉 Testando o modelo treinado com prompt: '{test_prompt}'")

# Formatar o prompt como instrução
formatted_prompt = f"<|im_start|>user\nAnalise o seguinte caso de triagem hospitalar e classifique conforme\nprotocolo de Manchester:\n{test_prompt}<|im_end|>\n<|im_start|>assistant\n"

inputs = tokenizer(formatted_prompt, return_tensors="pt").to(device)
with torch.no_grad():
    outputs = model.generate(
        inputs.input_ids,
        max_length=200,
        temperature=0.7,
        do_sample=True,
        repetition_penalty=1.2
    )

generated_text = tokenizer.decode(outputs[0], skip_special_tokens=True)
print(f"💬 Resposta do modelo (após fine-tuning):\n{generated_text}")

# === BLOCO 6: Avaliação do Modelo com o Conjunto de Teste ===
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import re
import matplotlib.pyplot as plt
import seaborn as sns

print("🔍 Avaliando o modelo com o conjunto de teste...")

# Função para extrair a classificação (cor) de um texto de resposta
def extract_classification(text):
    # Procurar por "Vermelho", "Laranja", "Amarelo", "Verde" ou "Azul" no texto
    colors = ["Vermelho", "Laranja", "Amarelo", "Verde", "Azul"]
    for color in colors:
        if color.lower() in text.lower():
            return color
    return "Não classificado"

# Função para gerar previsões em lote
def generate_predictions(test_dataset, model, tokenizer, batch_size=8):

```



```

all_predictions = []

for i in range(0, len(test_dataset), batch_size):
    batch = test_dataset[i:min(i+batch_size, len(test_dataset))]
    true_classifications = []

    # Extrair casos de pacientes e classificações verdadeiras
    cases = []
    for item in batch["text"]:
        match = re.match(r'(.*)[\n\r]*Resposta:(.*)', item, re.DOTALL)
        if match:
            patient_case = match.group(1).strip()
            true_response = match.group(2).strip()
            cases.append(patient_case)
            true_classifications.append(extract_classification(true_response))
        else:
            cases.append(item)
            true_classifications.append("Não classificado")

    # Gerar previsões para o lote
    predictions = []
    for case in cases:
        # Formatar o prompt como instrução
        formatted_prompt = f"<|im_start|>user\nAnalise o seguinte caso de triagem hospitalar e classifique\nconforme protocolo de Manchester:\n{case}<|im_end|>\n<|im_start|>assistant\n"
        inputs = tokenizer(formatted_prompt, return_tensors="pt").to(device)
        with torch.no_grad():
            outputs = model.generate(
                inputs.input_ids,
                max_new_tokens=100, # Usar max_new_tokens em vez de max_length
                temperature=0.2, # Baixa temperatura para respostas mais determinísticas
                do_sample=True,
                repetition_penalty=1.2,
                num_return_sequences=1
            )
        generated_text = tokenizer.decode(outputs[0], skip_special_tokens=True)
        predictions.append(generated_text)

    # Extrair classificações das previsões
    predicted_classifications = [extract_classification(pred) for pred in predictions]

    # Armazenar resultados
    for case, true_class, pred_text, pred_class in zip(cases, true_classifications, predictions, predicted_classifications):
        all_predictions.append({
            "Caso": case,
            "Classificação Real": true_class,
            "Texto Previsto": pred_text,
            "Classificação Prevista": pred_class
        })

    print(f"Processados {min(i+batch_size, len(test_dataset))}/{len(test_dataset)} casos de teste.")

return all_predictions

# Gerar previsões para o conjunto de teste
print("⏳ Gerando previsões para o conjunto de teste...")
predictions = generate_predictions(test_dataset, model, tokenizer)

# Converter para DataFrame para análise
results_df = pd.DataFrame(predictions)

```



```

# Calcular métricas de avaliação
y_true = results_df["Classificação Real"].tolist()
y_pred = results_df["Classificação Prevista"].tolist()

# Calcular acurácia
accuracy = accuracy_score(y_true, y_pred)
print(f"\n🎯 Acurácia do modelo: {accuracy:.4f}")

# Relatório de classificação
print("\n📊 Relatório de classificação:")
print(classification_report(y_true, y_pred))

# Matriz de confusão
print("\n🔢 Matriz de confusão:")
cm = confusion_matrix(y_true, y_pred)
classes = sorted(list(set(y_true + y_pred)))

# Visualizar matriz de confusão
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=classes, yticklabels=classes)
plt.xlabel('Previsão')
plt.ylabel('Real')
plt.title('Matriz de Confusão - Triagem de Manchester')
plt.show()

# Salvar resultados em CSV
results_df.to_csv(f"{output_dir}/resultados_teste.csv", index=False)
print(f"\n💾 Resultados salvos em {output_dir}/resultados_teste.csv")

# Analisar exemplos de cada categoria
print("\n🌐 Exemplos de classificações por categoria:")

def show_examples(df, category, n=2):
    filtered = df[df["Classificação Real"] == category]
    correct = filtered[filtered["Classificação Real"] == filtered["Classificação Prevista"]]
    incorrect = filtered[filtered["Classificação Real"] != filtered["Classificação Prevista"]]

    print(f"\n--- {category} ---")
    print(f"Total: {len(filtered)}, Corretos: {len(correct)}, Incorretos: {len(incorrect)}")

    if len(correct) > 0:
        print("\nExemplo(s) correto(s):")
        for i, (_, row) in enumerate(correct.head(n).iterrows()):
            print(f"\n{i+1}. Caso: {row['Caso']}")
            print(f"  Resposta: {row['Texto Previsto']}")

    if len(incorrect) > 0:
        print("\nExemplo(s) incorreto(s):")
        for i, (_, row) in enumerate(incorrect.head(n).iterrows()):
            print(f"\n{i+1}. Caso: {row['Caso']}")
            print(f"  Classificação Real: {row['Classificação Real']}")
            print(f"  Classificação Prevista: {row['Classificação Prevista']}")
            print(f"  Resposta: {row['Texto Previsto']}")

# Mostrar exemplos para cada categoria
for category in ["Vermelho", "Laranja", "Amarelo", "Verde", "Azul"]:
    show_examples(results_df, category)

# === BLOCO 7: Interface Interativa para Testes ===
print("\n🌐 Criando interface simples para testes interativos...")

```



```

def classificar_paciente(descricao_caso):
    # Formatar o prompt como instrução
    formatted_prompt = f"<|im_start|>user\nAnalise o seguinte caso de triagem hospitalar e classifique conforme protocolo de Manchester:\n{descricao_caso}<|im_end|>\n<|im_start|>assistant\n"

    inputs = tokenizer(formatted_prompt, return_tensors="pt").to(device)
    with torch.no_grad():
        outputs = model.generate(
            inputs.input_ids,
            max_new_tokens=100,
            temperature=0.3,
            do_sample=True,
            repetition_penalty=1.2
        )

    resposta = tokenizer.decode(outputs[0], skip_special_tokens=True)

    # Extrair a classificação
    classificacao = extract_classification(resposta)

    # Cores para tornar a saída mais visual
    cores = {
        "Vermelho": "\033[91m", # Vermelho
        "Laranja": "\033[93m", # Laranja/Amarelo
        "Amarelo": "\033[93m", # Laranja/Amarelo
        "Verde": "\033[92m", # Verde
        "Azul": "\033[94m", # Azul
        "Não classificado": "\033[90m" # Cinza
    }

    cor = cores.get(classificacao, "\033[0m")
    reset = "\033[0m"

    print(f"\n🏥 Resposta do modelo:\n{resposta}")
    print(f"\n👉 Classificação: {cor}{classificacao}{reset}")

    # Interpretação da classificação
    interpretacoes = {
        "Vermelho": "Emergência - Atendimento imediato",
        "Laranja": "Muito urgente - Atendimento em até 10 minutos",
        "Amarelo": "Urgente - Atendimento em até 60 minutos",
        "Verde": "Pouco urgente - Atendimento em até 120 minutos",
        "Azul": "Não urgente - Atendimento em até 240 minutos"
    }

    if classificacao in interpretacoes:
        print(f"💡 {interpretacoes[classificacao]}")

    # Loop interativo para testar o modelo com novos casos
    print("\n🌐 Interface de teste do modelo de triagem")
    print("Digite 'sair' ou 'q' para encerrar, ou 'ajuda' para ver comandos disponíveis")

    # Lista de casos de exemplo para facilitar os testes
    casos_exemplo = {
        "1": "Paciente com dor torácica súbita intensa com sudorese profusa.",
        "2": "Paciente com hemorragia no pé moderada e FC 124 bpm.",
        "3": "Paciente com distúrbio de coagulação conhecido em tratamento.",
        "4": "Paciente com tosse seca leve sem febre.",
        "5": "Paciente com exame laboral habitual."
    }

    running = True
    while running:

```



```

print("\n" + "-"*50)
caso = input("🔍 Descreva o caso do paciente (ou digite um comando): ")

# Verificar comandos especiais
caso_lower = caso.lower().strip()

# Comando de saída
if caso_lower in ['sair', 'q', 'quit', 'exit']:
    print("👋 Encerrando a interface de teste.")
    running = False
    continue

# Comando de ajuda
elif caso_lower in ['ajuda', 'help', '?']:
    print("\n📋 Comandos disponíveis:")
    print(" 'sair', 'q', 'quit', 'exit' - Encerrar o programa")
    print(" 'ajuda', 'help', '?' - Mostrar esta mensagem de ajuda")
    print(" 'exemplos', 'ex' - Mostrar casos de exemplo disponíveis")
    print(" '1' a '5' - Usar um dos casos de exemplo pré-definidos")
    continue

# Mostrar exemplos disponíveis
elif caso_lower in ['exemplos', 'ex', 'examples']:
    print("\n📋 Casos de exemplo disponíveis:")
    for num, exemplo in casos_exemplo.items():
        print(f" {num}: {exemplo}")
    continue

# Verificar se é um número de exemplo
elif caso in casos_exemplo:
    caso = casos_exemplo[caso]
    print(f"🔍 Usando caso de exemplo: {caso}")

# Verificar se o caso está vazio
elif not caso.strip():
    print("⚠️ Por favor, digite uma descrição do caso ou um comando.")
    continue

# Classificar o caso
try:
    classificar_paciente(caso)
except Exception as e:
    print("❌ Erro ao processar o caso: {str(e)}")
    print("Por favor, tente novamente com uma descrição diferente.")

# === BLOCO 8: Exportar o Modelo ===
import torch
from transformers import AutoModelForCausalLM, AutoTokenizer
from peft import PeftModel
import os
from google.colab import drive

# Montar o Google Drive
print("🔗 Montando Google Drive...")
drive.mount('/content/drive')

# Definir pasta na Google Drive para salvar o modelo
save_path = "/content/drive/MyDrive/modelo_triagem_manchester_completo"
os.makedirs(save_path, exist_ok=True)

# Verificar a estrutura atual do diretório

```



```

print("Estrutura de diretórios atual:")
!ls -la

# Verificar o conteúdo da pasta modelo_triagem_manchester
print("\nConteúdo da pasta modelo_triagem_manchester:")
!ls -la modelo_triagem_manchester

# Definir caminhos
base_model_id = "Qwen/Qwen2.5-3B"

# Use o caminho correto para os adaptadores
lora_path = "./modelo_triagem_manchester"

# Carregar tokenizer
print("\n⚠️ Carregando tokenizer...")
tokenizer = AutoTokenizer.from_pretrained(base_model_id)

# Carregar modelo base
print("\n⚠️ Carregando modelo base...")
base_model = AutoModelForCausalLM.from_pretrained(
    base_model_id,
    torch_dtype=torch.float16,
    device_map="auto"
)

# Procurar o adapter_config.json se não for encontrado no lora_path
if not os.path.exists(os.path.join(lora_path, "adapter_config.json")):
    print(f"\n⚠️ Arquivo adapter_config.json não encontrado em {lora_path}")
    print("Procurando em subdiretórios...")

# Verificar se existe em algum subdiretório
for root, dirs, files in os.walk('.'):
    if 'adapter_config.json' in files:
        lora_path = root
        print(f"✅ Encontrado em: {lora_path}")
        break

# Verificar novamente
if not os.path.exists(os.path.join(lora_path, "adapter_config.json")):
    print(f"❌ adapter_config.json não encontrado. Abortando.")
else:
    print(f"\n⚠️ Aplicando adaptadores LoRA de {lora_path}...")
    # Aplicar adaptadores LoRA
    model = PeftModel.from_pretrained(base_model, lora_path)

    # Mesclar adaptadores com o modelo base
    print("\n⚠️ Mesclando adaptadores com o modelo base...")
    merged_model = model.merge_and_unload()

    # Salvar modelo completo diretamente no Google Drive
    print(f"\n⚠️ Salvando modelo completo no Google Drive em {save_path}...")
    merged_model.save_pretrained(save_path)
    tokenizer.save_pretrained(save_path)

    # Comprimir para backup (opcional)
    print("\n⚠️ Comprimindo para backup...")
    zip_path = f"/content/drive/MyDrive/modelo_triagem_manchester_completo.zip"
    !zip -r {zip_path} {save_path}

    print("\n✅ Processo concluído!")
    print(f"✅ Modelo salvo em: {save_path}")
    print(f"✅ Backup comprimido salvo em: {zip_path}")

```



```
# === BLOCO 9: Gerar Relatório Final ===
from IPython.display import Markdown, display
import pandas as pd

def md(text):
    display(Markdown(text))

# Título e introdução
md("# Relatório Final - Sistema de Triagem Manchester com LLM")
md(""""

## Resumo
Este relatório apresenta os resultados do desenvolvimento de um sistema de apoio à decisão para triagem hospitalar baseado em um Large Language Model (LLM). O sistema foi treinado para classificar casos de pacientes conforme o protocolo de triagem de Manchester, atribuindo as cores: Vermelho, Laranja, Amarelo, Verde e Azul.

## Metodologia
- **Modelo Base**: Qwen2.5-3B
- **Técnica**: Fine-tuning com LoRA (Low-Rank Adaptation)
- **Dados de Treinamento**: 1000 casos de triagem
- **Dados de Validação**: 100 casos de triagem
- **Dados de Teste**: 100 casos de triagem
""")

# Resultados
md("## Resultados")
md(f"""

### Métricas de Desempenho
- **Acurácia Global**: 99%
- **Vermelho (Emergência)**: Precisão 100%, Recall 100%
- **Laranja (Muito Urgente)**: Precisão 95%, Recall 100%
- **Amarelo (Urgente)**: Precisão 100%, Recall 95%
- **Verde (Pouco Urgente)**: Precisão 100%, Recall 100%
- **Azul (Não Urgente)**: Precisão 100%, Recall 100%
""")

# Análise e discussão
md("## Análise e Discussão")
md(""""

## Pontos Fortes
- O modelo demonstrou excelente capacidade de classificação dos casos conforme o protocolo de Manchester.
- Apenas 1 erro em 100 casos (99% de acurácia).
- Todas as categorias tiveram excelente desempenho, especialmente as categorias de maior urgência (Vermelho).

## Áreas para Melhoria
- O modelo tende a dar respostas padronizadas para certas categorias, não personalizando completamente a justificativa para o caso específico.
- Alguns caracteres estranhos aparecem no final das respostas.
- O único erro ocorreu na classificação de um caso de dor torácica intensa (9/10), classificando como Laranja quando deveria ser Amarelo.

## Aplicações Potenciais
- Assistente digital para profissionais de saúde na triagem hospitalar
- Sistema de pré-triagem para pacientes antes da chegada ao hospital
- Ferramenta de treinamento para novos profissionais de saúde
""")

# Conclusão
md("## Conclusão")
md(""""


```



O sistema desenvolvido demonstrou excelente desempenho na classificação de casos de triagem conforme o protocolo de Manchester. A acurácia de 99% indica que o modelo é altamente confiável para sugerir classificações, podendo servir como uma ferramenta de apoio à decisão para profissionais de saúde.

É importante ressaltar que, apesar do alto desempenho, o sistema deve ser utilizado como ferramenta de suporte e não substituir o julgamento clínico dos profissionais de saúde.

""")

### *app.py*

```
import re
from transformers import AutoTokenizer, AutoModelForCausalLM
import torch

model_path = "."

tokenizer = AutoTokenizer.from_pretrained(model_path, local_files_only=True)
model = AutoModelForCausalLM.from_pretrained(
    model_path,
    torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
    device_map="auto",
    local_files_only=True
)

# Protocolo específico de triagem de quedas
protocolo_quedas = """
PROTOCOLO DE TRIAGEM DE QUEDAS - MANCHESTER

PULSEIRA VERMELHA (Emergência):
• Hipotensão (TA<90/60mmHg) + taquicardia (FC>100bpm) + palidez cutânea = CHOQUE
• Hipotensão (TA<90/60mmHg) + taquicardia (FC>100bpm) + sudorese = CHOQUE
• Hipotensão (TA<90/60mmHg) + taquicardia (FC>100bpm) + diminuição consciência (ECG≤12) = CHOQUE
• Crise epiléptica tônica/clônica generalizada = CONVULSIONANDO
• Glicemia capilar <70 mg/dL = HIPOGLICEMIA

PULSEIRA LARANJA (Muito Urgente):
• Queda de altura >1m ou >5 escadas = MECANISMO TRAUMA SIGNIFICATIVO
• Hipotermia (T<35°C) = HIPOTERMIA
• Dor intensa (>7/10 numérica ou >3 fáicies) = DOR INTENSA

PULSEIRA AMARELA (Urgente):
• História de perda de consciência após evento = HISTÓRIA INCONSCIÊNCIA
• Distúrbio coagulação conhecido = DISTÚRBIO COAGULAÇÃO
• História discordante (factos não explicam lesões) = HISTÓRIA DISCORDANTE
• Fratura exposta/deformidade importante = FRATURA EXPOSTA
• Dor moderada (4-7/10 numérica ou 2-3 fáicies) = DOR MODERADA

PULSEIRA VERDE (Pouco Urgente):
• Dor ligeira (<3/10 numérica ou <2 fáicies) = DOR LIGEIRA
• Edema localizado/deformidade = EDEMA/DEFORMIDADE

PULSEIRA AZUL (Não Urgente):
• Queda sem sintomas ou sinais relevantes = QUEDA SEM SINTOMAS
"""

# Prompt de sistema para o LLM principal
system_prompt = f"""
Eu sou um assistente de triagem de quedas baseado no protocolo de Manchester. Respondo apenas a questões acerca de quedas, triagem, sintomas, sinais, avaliação de gravidade (pulseira Vermelha, Laranja, Amarela, Verde) ou conduta após queda em humanos. Se a pergunta for sobre outro tema, recuse com: 'Desculpe,'

```



só posso responder questões relacionadas com triagem de quedas em humanos segundo o protocolo de Manchester.'

{protocolo\_quedas}

Use o protocolo acima para determinar pulseiras e seja preciso citando os critérios específicos.  
'''

# Prompt específico para validação pelo LLM  
prompt\_validacao = """Você é um filtro especializado em identificar se perguntas são sobre triagem médica de quedas em humanos.

ACEITE apenas perguntas sobre:

- Avaliação médica após queda em pessoas
- Atribuição de pulseiras de triagem (Vermelha/Laranja/Amarela/Verde/Azul)
- Sintomas, sinais vitais ou condições após queda
- Aplicação do protocolo de Manchester para quedas
- Cenários clínicos envolvendo quedas em humanos

REJEITE perguntas sobre:

- Animais (cães, gatos, etc.)
- Objetos
- Quedas não-médicas (preços, cabelo, sistemas, etc.)
- Prevenção de quedas (foco deve ser triagem/avaliação)
- Outros protocolos não relacionados a quedas
- Assuntos gerais não relacionados a triagem médica

Responda apenas "ACEITAR" ou "REJEITAR":'''

```
def valida_com_llm(pergunta):
    """Usa o LLM para determinar se a pergunta é válida"""
    input_validacao = f"{prompt_validacao}\n\nPergunta: \"{pergunta}\"\n\nResposta:"

    inputs = tokenizer(input_validacao, return_tensors="pt").to(model.device)

    with torch.no_grad():
        outputs = model.generate(
            **inputs,
            max_new_tokens=10,
            temperature=0.1,
            do_sample=False,
            pad_token_id=tokenizer.eos_token_id
        )

    resposta = tokenizer.decode(outputs[0], skip_special_tokens=True)
    resposta_limpa = resposta.replace(input_validacao, "").strip().upper()

    # Verifica se a resposta contém "ACEITAR"
    return "ACEITAR" in resposta_limpa[:20]

def processar_resposta_principal(pergunta):
    """Processa a pergunta com o LLM principal de triagem"""
    input_text = f"{system_prompt}\n\nUsuário: {pergunta}\nAssistente:"
    inputs = tokenizer(input_text, return_tensors="pt").to(model.device)

    with torch.no_grad():
        outputs = model.generate(
            **inputs,
            max_new_tokens=200,
            temperature=0.3,
            do_sample=True,
            top_p=0.9,
            pad_token_id=tokenizer.eos_token_id,
```



```

        repetition_penalty=1.1
    )

resposta_completa = tokenizer.decode(outputs[0], skip_special_tokens=True)
resposta_limpa = resposta_completa.replace(input_text, "").strip()

# Limpa a resposta
if "Assistente:" in resposta_limpa:
    resposta_limpa = resposta_limpa.split("Assistente:")[1].strip()

# Remove linhas que possam conter partes do prompt
linhas = resposta_limpa.split('\n')
resposta_final = []

for linha in linhas:
    linha = linha.strip()
    if linha and not any(palavra in linha.lower() for palavra in ['usuário:', 'sistema:', 'prompt:', 'instruções:']):
        resposta_final.append(linha)

return '\n'.join(resposta_final).strip()

# Loop principal - SIMPLES e focado no LLM
print("=" * 60)
print("🤖 SISTEMA DE TRIAGEM DE QUEDAS - PROTOCOLO MANCHESTER 🤖")
print("\nSou um assistente de enfermagem especializado no processo de triagem de quedas baseado no protocolo de Manchester. \n\nRespondo apenas a questões relacionadas com o procedimento de triagem de quedas. \n\nEsta análise é totalmente realizada por IA")
print("=" * 60)
print("Digite a sua questão ou escreva 'sair' para encerrar.")
print("\n" + "-"*60)

while True:
    pergunta = input("\n💡 Questão: ").strip()

    if pergunta.lower() in ['sair', 'exit', 'quit', 'q']:
        print("\n👋 Encerrando o sistema. Até logo!")
        break

    if not pergunta:
        print("⚠️ Por favor, digite a sua questão.")
        continue

    print("💻 A sua questão encontra-se em processo de análise, por favor aguarde.")

    # O LLM decide se aceita ou rejeita
    if valida_com_llm(pergunta):
        print("✅ Questão válida, prosseguindo com a triagem...")

    try:
        resposta = processar_resposta_principal(pergunta)

        print("\n" + "="*50)
        print("💡 RESPOSTA DA TRIAGEM:")
        print("="*50)
        print(resposta)
        print("="*50)

    except Exception as e:
        print(f"⚠️ Erro ao processar: {e}")

    else:
        print("\n❌ A questão colocada não se encontra relacionada com a triagem de quedas.")

```



```
print("Desculpe, só posso responder questões relacionadas com triagem de quedas em humanos segundo o protocolo de Manchester. \nGrato pela compreensão!")
```



## Anexo B

### Contrato

Tendo como objetivo alcançar o sucesso, garantir um bom funcionamento de toda a equipa de trabalho e execução de tarefas, foi regido o seguinte contrato, enquadrado na unidade curricular de Técnicas de Inteligência Artificial, parte do plano de estudos da Licenciatura em Engenharia e Gestão de Sistemas de Informação.

Assim, de modo a garantir os critérios de sucesso deste trabalho, e também critérios de valorização e/ou penalização do trabalho, o grupo regeu-se pelos seguintes artigos.

#### **Artigo 1: Ética de trabalho**

##### **a. Conhecimento do Contrato**

- 1.** Todos os elementos do grupo devem estar cientes dos seus deveres e direitos estipulados no contrato.
- 2.** É responsabilidade da líder do grupo, Rute Silva, eleita pelos demais membros, garantir o devido cumprimento e aplicação de todas as cláusulas estabelecidas neste contrato.
- 3.** A líder do grupo, Luana Borges, eleita pelos restantes membros do grupo, deve certificar-se que todas as alíneas presentes neste contrato são devidamente cumpridas e aplicadas.

##### **b. Compromisso com o Projeto**

- 1.** Todos os membros do grupo comprometem-se a contribuirativamente e de forma responsável para o projeto em questão.
- 2.** É esperado que cada membro cumpra suas obrigações e responsabilidades designadas para o sucesso do projeto.

##### **c. Participação e Reuniões**

- 1.** Os membros do grupo devem comparecer às reuniões semanais agendadas;
- 2.** É necessário que todos os membros participemativamente nas discussões e contribuam para o progresso do projeto.
- 3.** Caso um membro não possa comparecer a uma reunião, deve notificar o grupo com antecedência, de modo a tentar remarcar a reunião semanal.

##### **d. Respeito e Colaboração**

- 1.** Todos os membros devem tratar os restantes colegas de equipa com respeito e cortesia.
- 2.** É importante incentivar uma atmosfera de colaboração e apoio mútuo entre os membros do grupo.
- 3.** Cada membro deve avisar os restantes colegas aquando de uma dificuldade na execução das tarefas;

#### **Artigo 2: Penalização e/ou Valorização do Trabalho Prestado**

##### **a. Tarefas e prazos**

- 1.** Cada membro será designado com tarefas específicas e prazos para a conclusão das



mesmas.

2. É responsabilidade de cada membro cumprir os prazos estabelecidos para suas tarefas.

**b. Penalidades por Atraso**

1. Caso um membro não cumpra um prazo estabelecido, poderá ser penalizado conforme acordado pelo grupo.
2. As penalidades podem incluir a redução de pontos, revisão das responsabilidades atribuídas ou outras ações definidas pelo grupo.

**c. Valorização do Trabalho**

1. O trabalho de cada membro será valorizado com base no cumprimento dos prazos, qualidade das contribuições e colaboração com o grupo.
2. O reconhecimento e a valorização do trabalho serão definidos pelo grupo de forma justa e transparente.

**d. Resolução de Conflitos**

1. Em caso de divergências ou conflitos entre os membros, espera-se que todos os esforços sejam feitos para resolvê-los de maneira pacífica e construtiva.
2. Se necessário, um mediador poderá ser designado para auxiliar na resolução dos conflitos.

Este contrato de trabalho tem validade para o período do projeto em grupo e será assinado por todos os membros, indicando o acordo e compromisso mútuo com os termos estabelecidos.

**Guimarães, 18/03/2025**

**Rute Silva**

**David Gusmão**

**João Fernandes**

**Leonel Pinheiro**