



Universidade Federal da Bahia
Instituto de Matemática e Estatística
Colegiado de Estatística

VINÍCIUS DA COSTA SOARES

Automação de Análises Estatísticas: Uma Arquitetura Multiagente Baseada em Grandes Modelos de Linguagem

Salvador

2025

VINÍCIUS DA COSTA SOARES

**Automação de Análises Estatísticas: Uma
Arquitetura Multiagente Baseada em Grandes
Modelos de Linguagem**

Trabalho de Conclusão de Curso
apresentado ao Curso de Graduação
em Estatística, Colegiado de Esta-
tística, Instituto de Matemática e
Estatística, Universidade Federal da
Bahia, como requisito parcial à apro-
vação na disciplina de Trabalho de
Conclusão de Curso II.

Orientador: Prof. Dr. Ricardo Fer-
reira da Rocha

Salvador

2025

LISTA DE ILUSTRAÇÕES

Figura 1 – Frases em diferentes idiomas para exemplificar a <i>tokenização</i>	20
Figura 2 – Representação conceitual de um espaço de <i>embeddings</i> utilizando <i>t</i> -SNE.	25
Figura 3 – Diagrama de funcionamento de um sistema de RAG implementado a um LLM.	26
Figura 4 – Fluxograma dos Experimentos, da Solicitação à Avaliação do Relatório.	33
Figura 5 – Gráfico da Distribuição das Dificuldades dos Conjuntos de Dados.	39
Figura 6 – Gráfico da Distribuição das Dificuldades das Perguntas.	39
Figura 7 – Gráfico da Distribuição das Categorias das Perguntas.	40
Figura 8 – Gráficos do Tempo Médio de Execução de Relatórios por Dificuldade do Conjunto de Dados e Pergunta.	44
Figura 9 – Gráfico do Tempo Médio de Execução de Relatórios por Categoria de Pergunta.	45
Figura 10 – Gráfico do Tempo Médio de Execução de Relatórios por Agente.	45
Figura 11 – Gráfico da Média de <i>Tokens</i> (<i>Output</i> e Raciocínio) por Dificuldade do Conjunto de Dados.	46
Figura 12 – Gráfico da Média de <i>Tokens</i> (<i>Output</i> e Raciocínio) por Dificuldade de Pergunta.	47
Figura 13 – Gráfico da Média de <i>Tokens</i> (<i>Output</i> e Raciocínio) por Categoria de Pergunta.	48
Figura 14 – Gráfico da Média de <i>Tokens</i> (<i>Output</i> e Raciocínio) por Agente.	49
Figura 15 – Gráficos das Notas Ponderadas por Dificuldade de Conjunto de Dados e Pergunta.	50
Figura 16 – Gráfico das Notas Ponderadas por Categoria de Pergunta.	51
Figura 17 – Gráfico das Notas Ponderadas por Categoria de Pergunta.	52
Figura 18 – Gráfico das Notas Ponderadas por Categoria de Pergunta.	53
Figura 19 – Gráfico das Notas Ponderadas por Categoria de Pergunta.	53
Figura 20 – Sistema Automatizado - Gráfico da Taxa de Sobrevida por Classe (Pclass).	62
Figura 21 – Sistema Automatizado - Gráfico da Taxa de Sobrevida por Sexo.	62
Figura 22 – Sistema Automatizado - Gráfico da Taxa de Sobrevida por Classe e Sexo	63
Figura 23 – Sistema Automatizado - Gráfico da Taxa de Sobrevida por Faixa Etária.	64

LISTA DE TABELAS

Tabela 1 – Tabela com contagem de <i>tokens</i> , palavras e caracteres em frases de diferentes idiomas para exemplificar a <i>tokenização</i>	20
Tabela 2 – Exemplos hipotéticos de palavras e seus possíveis vetores de <i>embedding</i>	23
Tabela 3 – Exemplo de divisão de texto de 1.000 <i>tokens</i> em <i>chunks</i> de 200 <i>tokens</i> com sobreposição de 50 <i>tokens</i>	28
Tabela 4 – Custo de Cada Modelo Utilizado por Milhão de <i>tokens</i> (US\$).	32
Tabela 5 – Tabela dos Níveis de Dificuldade dos Dados.	37
Tabela 6 – Tabela dos Níveis de Dificuldade das Perguntas.	37
Tabela 7 – Tabela das Categorias de Perguntas.	38
Tabela 8 – Resumo Geral das Métricas e Custos de Confecção dos Relatórios.	43
Tabela 9 – Resumo Geral das Métricas e Custos de Avaliação dos Relatórios.	43
Tabela 10 – Sistema Automatizado - Resumo das Variáveis do Conjunto de Dados do Titanic.	59

LISTA DE ABREVIATURAS E SIGLAS

API	<i>Application Programming Interface</i> ou Interface de Programação de Aplicações
BERT	<i>Bidirectional Encoder Representations from Transformers</i> ou Representações de Codificador Bidirecional de Transformadores
BLOB	<i>Binary Large Object</i> ou Objeto Grande Binário
CoT	<i>Chain of Thought</i> ou Cadeia de Pensamento
FAQ	<i>Frequently Asked Questions</i> ou Perguntas Feitas Frequentemente
GPT	<i>Generative Pre-trained Transformer</i> ou Transformador Generativo Pré-treinado
IA	Inteligência Artificial
ID	Identificador
JSON	<i>JavaScript Object Notation</i> ou Notação de Objetos JavaScript
LLM	<i>Large Language Model</i> ou Modelo de Linguagem de Larga Escala
NLP	<i>Natural Language Processing</i> ou Processamento de Linguagem Natural
PCA	<i>Principal Component Analysis</i> ou Análise de Componentes Principais
PLM	<i>Pre-trained Language Model</i> ou Modelo de Linguagem Pré-treinado
PNG	<i>Portable Network Graphics</i> ou Gráficos de Rede Portáteis
RAG	<i>Retrieval-Augmented Generation</i> ou Geração Aumentada por Recuperação
RegEx	<i>Regular Expression</i> ou Expressão Regular
SAS	<i>Statistical Analysis System</i> ou Sistema de Análise Estatística
SQL	<i>Structured Query Language</i> ou Linguagem de Consulta Estruturada
ToT	<i>Tree of Thoughts</i> ou Árvore de Pensamentos
t-SNE	<i>t-distributed Stochastic Neighbor Embedding</i> ou Incorporação Estocástica de Vizinhos com Distribuição t

URL	<i>Uniform Resource Locator</i> ou Localizador Uniforme de Recursos
WEAT	<i>Word Embedding Association Test</i> ou Teste de Associação de Incorporação de Palavras

Resumo

Este trabalho apresenta o desenvolvimento e a validação de um sistema automatizado para a geração de relatórios estatísticos, utilizando agentes inteligentes baseados em modelos de linguagem de larga escala. A solução fundamenta-se em uma arquitetura multiagente hierárquica, na qual cada componente possui uma especialização distinta, incluindo planejamento, geração e execução de código Python, redação técnica e avaliação de qualidade. Diferentemente de abordagens puramente textuais, o sistema implementado é capaz de orquestrar a análise de dados ponta a ponta: desde a interpretação da pergunta do usuário e manipulação do conjunto de dados até a criação de gráficos e cálculos, consolidando os resultados em um documento $\text{\LaTeX}$ compilado. Os experimentos realizados demonstraram que a arquitetura produz relatórios coerentes e tecnicamente embasados. Os resultados confirmam a eficácia do sistema na redução da barreira técnica para análise de dados. A solução mostrou-se uma ferramenta funcional para otimizar fluxos de trabalho, liberando o profissional de Estatística da carga operacional para focar em validação e análises de maior valor agregado.

Palavras-Chave: Automação de Relatórios; Inteligência Artificial; Sistemas Multiagente; Modelos de Linguagem de Larga Escala (LLMs); Ciência de Dados.

SUMÁRIO

1	INTRODUÇÃO	10
1.1	Objetivo Geral	11
1.2	Objetivos Específicos	11
1.3	Organização do Trabalho	12
2	JUSTIFICATIVA	14
3	REVISÃO DE LITERATURA	16
4	METODOLOGIA	18
4.1	O que são LLMs	18
4.2	Modelos de Raciocínio	18
4.3	<i>Tokens</i>	19
4.4	Janela de contexto	21
4.5	<i>Embeddings</i>	21
4.6	<i>Retrieval-Augmented Generation</i> (RAG)	26
4.7	<i>Vector stores</i>	27
4.8	Ferramentas	29
5	EXPERIMENTOS E ANÁLISE DE RESULTADOS	32
5.1	Cenário Experimental e Métricas	33
5.1.1	<i>Arquitetura e Fluxo de Execução</i>	33
5.1.2	<i>Configuração e Cenários dos Experimentos</i>	36
5.1.3	<i>Armazenamento de Dados</i>	40
5.1.4	<i>Avaliação Automática de Qualidade</i>	40
5.2	Resultados Obtidos	42
5.2.1	<i>Análise de Desempenho Computacional</i>	43
5.2.2	<i>Análise da Qualidade dos Relatórios</i>	49
5.3	Discussão	54
6	EXEMPLO DE RELATÓRIO GERADO	56
7	CONCLUSÃO	72
	REFERÊNCIAS	73
	APÊNDICE A – INSTRUÇÕES DOS AGENTES	75

1 INTRODUÇÃO

Nos últimos anos, os avanços em modelos de linguagem de larga escala (LLMs) transformaram significativamente o campo da Inteligência Artificial, especialmente pela capacidade de compreender e gerar linguagem natural com alta fluidez e coerência. Ferramentas como o ChatGPT, Claude e Gemini demonstraram o potencial dessas tecnologias em tarefas como redação automática, geração de código, sumarização de textos, tradução, análise de sentimento e muito mais.

Os LLMs são modelos de aprendizado de máquina baseados em redes neurais que operam prevendo o próximo *token* em um texto. Essa capacidade de prever o que vem a seguir é treinada em um volume colossal de dados textuais advindos de páginas da *web*, livros, artigos, revistas, mídias sociais e fóruns *online*. A incrível adaptabilidade dos LLMs permitiu que eles fossem incorporados em uma vasta gama de setores.

Nos setores de finanças e direito, onde a análise de documentos e a criação de conteúdo especializado são tarefas rotineiras, eles são empregados na análise de sentimento de notícias financeiras, na classificação de textos (como manchetes de notícias) e Respostas a Perguntas Frequentes (FAQ). No atendimento ao cliente e na consultoria financeira, os LLMs atuam respondendo a dúvidas complexas e até mesmo traduzindo linguagem natural para, por exemplo, linguagens de consulta especializadas (LAI et al., 2024; LI et al., 2023; WU et al., 2023). Na área da saúde eles podem ser aplicados na simplificação de relatórios médicos, na extração de dados de documentos clínicos e na organização dos mesmos, além de melhorar a comunicação entre médicos e pacientes ao facilitar explicações claras de terminologias médicas. Sua capacidade de integração se estende a dispositivos médicos inteligentes, robôs cirúrgicos e assistentes virtuais, fornecendo monitoramento e suporte em tempo real. Eles também são empregados na medição de padrões psicológicos em grandes textos, como a identificação de emoções e até risco de suicídio (THIRUNAVUKARASU et al., 2023; CLUSMANN et al., 2023; SHAH; ENTWISTLE; PFEFFER, 2023; DEMSZKY et al., 2023).

Na robótica, funcionam como “cérebros” que traduzem instruções em linguagem natural em planos de ação executáveis, aprimorando a compreensão de comandos, tomada de decisão e interação humano-robô. Para isso, é essencial a integração de LLMs multimodais, como o GPT-4V (GPT-4 com visão), por combinar instruções de linguagem natural com percepções visuais, superando limitações de modelos unicamente textuais. Apesar disso, ainda existem diversas limitações nesse campo, como a dificuldade na geração de planos robustos e a dificuldade em compreender totalmente informações de profundidade e relações espaciais 3D (WANG et al., 2024).

Além de todos esses tópicos, os LLMs são rapidamente integrados em uma vasta gama de outros domínios e aplicações, como escrita de documentos, criação de código executável e resposta a perguntas, frequentemente com capacidades semelhantes às humanas.

1.1 Objetivo Geral

No contexto de Estatística e Ciência de Dados, todo esse avanço abre caminho para a construção de sistemas de agentes inteligentes capazes de automatizar fluxos de trabalho analíticos, desde a leitura e pré-processamento de dados, construção de modelos e interpretação de resultados até a elaboração de um relatório final. Esses agentes operam como sistemas compostos por LLMs e conectados a ferramentas externas, com capacidade de planejar e executar tarefas de forma autônoma ou assistida.

O objetivo geral deste trabalho é desenvolver um sistema baseado em agentes integrados a ferramentas específicas, com a finalidade de automatizar processos de análise de dados de forma estruturada e sistemática. O sistema busca oferecer uma abordagem adaptável a cada contexto (problema e conjunto de dados), permitindo que os agentes construam, justifiquem e executem planos de análise com base em informações obtidas da *web*, documentos locais e por meio da execução de código. Como resultado, espera-se que o sistema seja capaz de gerar relatórios completos, com fundamentação teórica, implementação prática e interpretação de resultados. Além disso, outro objetivo central é estabelecer uma metodologia para avaliar os relatórios gerados pelo sistema.

1.2 Objetivos Específicos

Os objetivos específicos deste trabalho incluem:

- Projetar a arquitetura de um sistema multiagente que permita a integração com ferramentas externas, como busca na web (*web search*), busca de documentos (*file search*) e interpretador de código (*code interpreter*);
- Implementar mecanismos de geração de planos de análise automática com base em consultas sobre dados e contextos fornecidos;
- Integrar agentes especializados para fundamentação teórica, escrita e execução de código em linguagem de programação Python e interpretação dos resultados obtidos;
- Analisar e avaliar os relatórios gerados por agentes;
- Avaliar o potencial do sistema para uso profissional, com foco em aumento de produtividade e redução de barreiras técnicas para análise de dados.

1.3 Organização do Trabalho

Este trabalho está organizado da seguinte forma: no Capítulo 2 apresenta-se a justificativa para a elaboração do projeto, no Capítulo 3 é feita a revisão de literatura. A fundamentação teórica necessária é detalhada no Capítulo 4, abordando conceitos como LLMs, *embeddings* e sistemas de *Retrieval-Augmented Generation* (RAG). No Capítulo 5, são discutidos detalhes acerca dos experimentos realizados, seguidos por um exemplo prático de uso do sistema apresentado no Capítulo 6. Por fim, o Capítulo 7 traz a conclusão do trabalho.

2 JUSTIFICATIVA

O avanço recente dos LLMs está impulsionando o desenvolvimento de sistemas de agentes capazes processar informações de diversas fontes, realizar interações e executar tarefas de forma autônoma e adaptativa. Embora essas capacidades estejam sendo exploradas em áreas como desenvolvimento de *software*, atendimento ao cliente e assistência jurídica, sua aplicação sistemática em tarefas de análise estatística e ciência de dados ainda é pouco explorada (ZHU et al., 2024).

A capacidade de gerar informações a partir de bases de dados é crucial para a tomada de decisões em diversos setores, como negócios, saúde e pesquisa científica. Tradicionalmente, essas tarefas de análise de dados exigem esforço manual e intelectual significativo, desde definir o plano de ação até a interpretação dos resultados e elaboração de um relatório, além de custos de mão de obra, o que limita a escalabilidade e a eficiência. Nesse contexto, torna-se relevante investigar como agentes baseados em LLMs podem contribuir para a automação inteligente da análise de dados, especialmente por meio da integração com ferramentas de recuperação de conhecimento (como *web search* e *file search*) e execução de código. Estudos recentes demonstram que esses sistemas são capazes de formular hipóteses, selecionar métodos estatísticos adequados, gerar código em linguagens como *Structured Query Language* (SQL), R ou Python e interpretar resultados de forma textual, aproximando-se, em certa medida, de um raciocínio realizado por humanos (ZHANG et al., 2025; SONG et al., 2025). Além disso, também é possível reduzir drasticamente o custo desse tipo de trabalho em mais de 100 vezes (ZHU et al., 2024), democratizando o acesso à análise de dados de qualidade, tornando-a mais acessível e escalável para indivíduos e organizações, o que também pode contribuir para o desenvolvimento de *softwares* estatísticos cada vez mais avançados e impulsionar a pesquisa na área (SONG et al., 2025).

Apesar disso, as abordagens existentes frequentemente exigem dados pré-higienizados, metas predefinidas e tendem a produzir resultados “rasos”. além de enfrentar limitações de tamanho de contexto ou dificuldades no processamento de dados estruturados (PÉREZ et al., 2025). Uma abordagem que integra a capacidade de raciocínio automatizada dos LLMs com um conjunto de ferramentas específicas apresenta um potencial de superar as limitações mencionadas, gerando análises mais autônomas e profundas.

A justificativa para este trabalho reside na necessidade crescente de soluções que ampliem o acesso à análise de dados, aumentando a eficiência de estatísticos e analistas experientes ao automatizar etapas do ciclo de análise. Isso libera o profissional para se concentrar em tarefas de maior valor agregado, como formulação de estratégias e validação dos resultados. Dessa forma, o desenvolvimento de agentes capazes de realizar análises

estatísticas com fundamentação teórica, execução técnica e interpretação de resultados, representa uma contribuição significativa do ponto de vista aplicado.

Além disso, ao implementar uma metodologia de avaliação do sistema agêntico, este trabalho contribui para o debate sobre a validação de sistemas de IA generativa. Assim, o projeto se justifica pela inovação técnica e pelo seu potencial de impactar positivamente a prática do profissional de Estatística, promovendo fluxos de trabalho mais autônomos e escaláveis.

3 REVISÃO DE LITERATURA

Os LLMs possuem um grande potencial para fins estatísticos e de ciência de dados, sendo propostos como uma ferramenta para automatizar a geração de *insights* (textos curtos com informações significativas derivadas dos dados), cruciais para a tomada de decisões em diversos setores, como negócios, saúde e pesquisa científica (PÉREZ et al., 2025; ZHANG et al., 2025).

No entanto, a integração de LLMs na análise de dados apresenta desafios notáveis, como uma janela de contexto limitada, o que impede a introdução de grandes bases de dados. Além disso, ainda enfrentam dificuldades no processamento de dados tabulares e em sua interpretação (PÉREZ et al., 2025). A sua capacidade de raciocínio analítico também é limitada, o que pode resultar em *outputs* genéricos e pouco confiáveis quando usadas para analisar dados diretamente (FELI et al., 2025). Estudos também revelam que os LLMs têm dificuldades em distinguir os cenários corretos para uso de métodos estatísticos, além de seus pressupostos, levando a erros de aplicabilidade (ZHU et al., 2024).

Para superar essas limitações, têm sido desenvolvidas abordagens que posicionam os LLMs como intermediários inteligentes que orquestram a interação com ferramentas bem estabelecidas, alavancando o poder de planejamento e pensamento automatizados (FELI et al., 2025). Por exemplo, numa abordagem para geração de *insights*, um sistema baseado em LLMs criado por Pérez et al. (2025) inclui um gerador de hipóteses para formular perguntas relevantes, um agente de consulta que gera códigos SQL com potencial de responder essas perguntas e um módulo para resumir e formalizar os resultados. Essa abordagem, ao utilizar SQL como ferramenta, permite obter *insights* mais precisos do que os que seriam gerados diretamente por um LLM. No contexto da análise de dados fisiológicos, um agente alimentado pelo GPT-3.5-turbo orquestra solicitações do usuário com fontes de dados e ferramentas analíticas, demonstrando um desempenho significativamente superior em estimativas de frequência cardíaca em comparação com modelos como GPT-4o e GPT-4o-mini. Novamente, essa superioridade se deve pelo uso de ferramentas através do agente (FELI et al., 2025). Estudos recentes também mostram que treinar LLMs com dados de trajetórias de especialistas melhora significativamente seu desempenho em tarefas de planejamento (HU et al., 2024).

A criação de conjuntos de dados de treino específicos para o domínio é outra estratégia interessante. O CoddLLM, por exemplo, é construído a partir de um *fine-tuning* do Mistral-NeMo-Instruct-2407, criado conjuntamente pela Mistral AI e NVIDIA. Ele foi feito especificamente para análise de dados, utilizando um vasto conjunto de dados, com aproximadamente 2 milhões de exemplos e 1 bilhão de *tokens*, abrangendo diversos

temas como de modelagem e análise de dados e linguagem SQL. De uma forma geral, essa abordagem melhora a compreensão do modelo sobre a relação entre linguagem natural e dados tabulares. O CoddLLM demonstrou um desempenho superior em avaliações de conhecimento analítico, seleção de tabelas e geração de consultas SQL em comparação com modelos de *benchmark* (ZHANG et al., 2025). *Benchmarks* são conjuntos padronizados de tarefas e dados utilizados para avaliar e comparar o desempenho de algo, nesse caso de diferentes LLMs.

Já o StatLLM surge como um conjunto de dados aberto focado na avaliação do desempenho de LLMs em análise estatística, particularmente na geração de código em *Statistical Analysis System* (SAS). Ele aborda tarefas de análise estatística a partir de código SAS gerado por LLMs (como GPT-3.5, GPT-4.0 e Llama 3.1), e pontuações feitas a partir de avaliação humana onde é avaliado a correção, legibilidade e precisão da saída do código gerado (SONG et al., 2025). Similarmente, o StatQA foi criado para avaliar tarefas estatísticas especializadas, mais precisamente testes de hipóteses, medindo a proficiência de LLMs na seleção de colunas relevantes e na aplicação dos métodos estatísticos. Esse estudo revelou que, como já dito anteriormente, os LLMs enfrentam desafios consideráveis em tarefas estatísticas, comumente performando abaixo de humanos. Para modelos estado da arte, incluir conhecimentos específicos no *prompt* e realizar *fine-tuning* demonstrou melhoras significativas na performance, o que não acontece para modelos abertos, que possuem habilidades de compreensão inferiores (ZHU et al., 2024).

A análise de erros também destaca as diferentes áreas de proficiência e deficiência entre LLMs e humanos: enquanto os LLMs são melhores em distinguir diferentes tarefas estatísticas, mas piores ao avaliar a aplicabilidade dos métodos, os humanos são proficientes em discernir a aplicabilidade dos métodos, mas mais propensos à confusão de tarefas estatísticas (ZHU et al., 2024). Essa distinção sugere que uma abordagem onde LLMs são supervisionados por especialistas humanos minimiza os erros de ambos.

Resumidamente, os LLMs estão se tornando ferramentas na análise de dados, especialmente com o desenvolvimento de sistemas agênticos, *fine-tuning* em dados específicos do domínio, engenharia de *prompt* e utilização de ferramentas externas. Apesar de ainda existirem desafios relacionados à compreensão de dados estruturados e à avaliação da aplicabilidade de métodos, as soluções em desenvolvimento estão melhorando significativamente a sua precisão e confiabilidade, ao mesmo tempo que reduzem os custos de geração de *insights*. O futuro aponta para uma maior integração entre LLMs e humanos para uma análise de dados mais eficiente (HU et al., 2024; PÉREZ et al., 2025; ZHANG et al., 2025; SONG et al., 2025).

4 METODOLOGIA

4.1 O que são LLMs

Os LLMs são modelos de linguagem baseados na arquitetura de redes neurais *Transformer* (VASWANI et al., 2017), que possuem um número muito grande de parâmetros, com alguns modelos alcançando centenas de bilhões ou mais, e que são treinados em grandes volumes de dados textuais. O termo LLM foi cunhado pela comunidade de pesquisa para modelos de linguagem pré-treinados (PLMs) de tamanho significativo, normalmente com centenas de bilhões de parâmetros ou mais. Isso aconteceu porque pesquisadores descobriram que o aumento do número de parâmetros levava a uma melhoria substancial na capacidade do modelo. Eles são treinados para modelar a probabilidade generativa de sequências de palavras, prevendo o próximo *token* (unidade de texto) com base nos anteriores (ZHAO et al., 2023).

Uma característica dos LLMs é a presença de “habilidades emergentes”, que são capacidades especiais que não estão presentes em modelos de linguagem menores, como o GPT-2 ou o *Bidirectional Encoder Representations from Transformers* (BERT), e que surgem apenas quando a escala de parâmetros excede um certo nível. Um exemplo disso é a aprendizagem em contexto, onde um LLM pode aprender a realizar uma nova tarefa com apenas alguns exemplos no *prompt*, sem a necessidade de *fine-tuning*. O *fine-tuning* é, basicamente, o processo de especializar um LLM para uma tarefa específica, adaptando seus pesos através de um treinamento supervisionado com dados direcionados (ZHAO et al., 2023; BROWN et al., 2020).

A pesquisa e o desenvolvimento de LLMs avançaram significativamente, resultando no lançamento de *chatbots* de IA como o ChatGPT, da OpenAI. Apesar do progresso, a natureza exata das habilidades emergentes dos LLMs ainda não é totalmente compreendida. Também existem desafios relacionados ao treinamento e inferência, como a demanda crescente por recursos computacionais, além de questões éticas envolvendo esses modelos. Entretanto, a pesquisa na área continua a tentar superar as limitações existentes (ZHAO et al., 2023).

4.2 Modelos de Raciocínio

Embora a função principal de um LLM seja a previsão sequencial de texto, sua capacidade de resolver problemas complexos que exigem raciocínio lógico ou matemático representa um grande salto. Essa habilidade pode vir da aplicação de técnicas e estruturas de *prompt* que guiam o processo generativo do modelo, são estratégias projetadas para

estruturar a geração de texto de modo a simular uma linha de pensamento.

A técnica que destravou o potencial de raciocínio dos LLMs é a *chain of thought* (CoT), proposta por [Wei et al. \(2022\)](#). Em vez de solicitar uma resposta direta, a abordagem CoT instrui o modelo a pensar em um passo a passo, gerando as etapas intermediárias antes de chegar à conclusão final. Ao fazer isso, o modelo é forçado a decompor o problema em partes menores e a seguir uma trajetória lógica, o que reduz significativamente os erros em tarefas que dependem de múltiplas etapas. A eficácia do CoT demonstrou que a forma como uma instrução é apresentada pode alterar drasticamente o resultado e a capacidade de um modelo.

Apesar de sua eficácia, uma única cadeia de pensamento pode conter erros lógicos que prejudicam o resultado final. Para mitigar isso, [Wang et al. \(2022\)](#) propuseram a técnica de autoconsistência, que gera múltiplas cadeias de raciocínio para o mesmo problema. A resposta final é determinada através de um processo de votação entre os resultados das diferentes cadeias. Esse método aumenta a robustez e a confiabilidade do modelo, pois assume que a resposta correta será a mais consistentemente alcançada.

Uma abordagem mais sofisticada é *tree of thoughts* (ToT), que foi proposta por [Yao et al. \(2023\)](#). Ela generaliza a CoT, substituindo a cadeia linear por uma estrutura em árvore, dessa forma o modelo pode explorar múltiplas cadeias em paralelo, podendo gerar vários pensamentos em cada etapa ou até retroceder se um caminho se mostrar infrutífero. Essa capacidade de planejamento estratégico permite que os LLMs resolvam problemas mais complexos que exigem a consideração de múltiplas hipóteses, gerando um processo de resolução de problemas mais completo.

Essas abordagens demonstram grande evolução na capacidade dos LLMs de realizar tarefas complexas, otimizando seu processo probabilístico para produzir resultados mais funcionais. A pesquisa em modelos de raciocínio continua a ser uma das fronteiras mais ativas e promissoras no desenvolvimento de IA, buscando criar sistemas cada vez mais capazes e confiáveis.

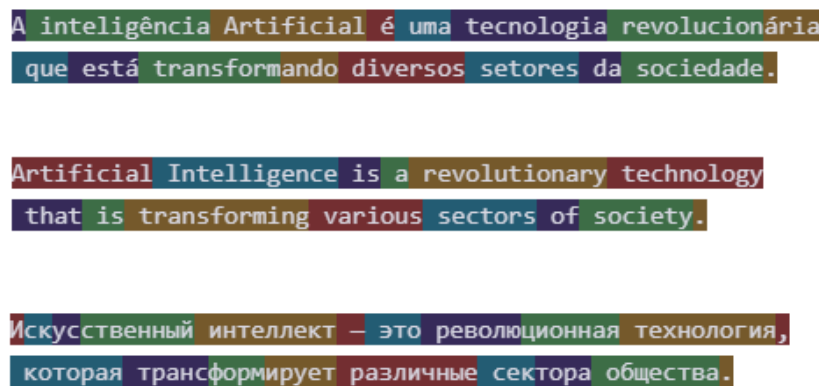
4.3 Tokens

Os modelos de linguagem não processam frases ou palavras diretamente, mas sim *tokens* de forma sequencial, que funcionam como unidades de texto de entrada e saída. Assim, o processo de *tokenização* converte texto em uma sequência de *tokens*. Esses *tokens* podem representar diferentes formas, como uma palavra inteira, parte de uma palavra, um símbolo ou um espaço em branco, dependendo do processo de *tokenização* adotado pelo modelo. Em média, cada *token* corresponde a aproximadamente quatro caracteres ou 3/4 de uma palavra em um texto médio da língua inglesa.

Nesses modelos de *tokenização*, uma vez que as unidades são definidas, elas são adicionadas a um dicionário. Cada entrada nesse dicionário consiste num *token* único e um Identificador (ID) numérico associado a ele. Quando o modelo de linguagem processa um texto, ele consulta esse dicionário para converter cada *token* de texto em seu ID correspondente, que é a representação que o modelo utiliza realizar inferência. O GPT-4, por exemplo, opera com aproximadamente 100 mil *tokens*, e a evolução na tokenização visa aumentar a eficiência da representação de múltiplos idiomas e domínios técnicos, como programação.

A Figura 1 e a Tabela 1 apresentam três frases com igual significado em português, inglês e russo, apenas para visualização comparativa. Cada um dos *tokens* estão sendo representados pelas cores.

Figura 1 – Frases em diferentes idiomas para exemplificar a *tokenização*.



Fonte: Elaborado pelo autor.

Tabela 1 – Tabela com contagem de *tokens*, palavras e caracteres em frases de diferentes idiomas para exemplificar a *tokenização*.

Idioma	<i>Tokens</i>	Palavras	Caracteres
Português	18	14	117
Inglês	16	13	110
Russo	20	10	112

Fonte: Elaborado pelo autor.

Na maioria das *Application Programming Interfaces* (APIs), como a da OpenAI e Google, a precificação final para o usuário é baseada na quantidade de *tokens* de entrada (*input*) e saída (*output*) processados e gerados, conforme a precificação de cada modelo.

O processo de *tokenização* tem sido otimizado juntamente com os novos modelos de linguagem. Isso é de extrema importância, pois reduz custos, melhora o desempenho do modelo em textos mais longos e aumenta a janela de contexto sem a necessidade de avanços no próprio modelo.

4.4 Janela de contexto

Outro aspecto técnico importante para entender o funcionamento dos LLMs é a janela de contexto. Ela é essencialmente o número máximo de *tokens* que o modelo pode processar em uma única inferência, representando a soma do *prompt* e resposta que o modelo consegue considerar simultaneamente. Quando esse limite é ultrapassado, *tokens* mais antigos do início da sequência são descartados, comprometendo a resposta gerada, nesse sentido a janela de contexto funciona como um limite de memória imediata do modelo.

O tamanho da janela de contexto varia conforme o modelo: o modelo obsoleto GPT-2 possui uma janela de até 1.024 *tokens*, enquanto modelos mais recentes possuem uma janela de contexto bem maior, como o GPT-4.1 que possui uma janela de contexto de até 1.047.576 *tokens*. Isso corresponde a um avanço de 768 palavras (algo entre 2 a 3 páginas), para quase mil vezes mais, algo em torno de 2.000 a 3.000 páginas, a depender da formatação e tamanho da fonte

4.5 *Embeddings*

Os *embeddings* são representações vetoriais de alta dimensão de linguagem natural, onde a proximidade entre vetores reflete similaridade semântica. Eles permitem que modelos lidem com texto de maneira mais eficaz, e isso é possível porque palavras que ocorrem em contextos semelhantes tendem a possuir representações vetoriais próximas. Isso possibilita uma vasta gama de aplicações, como por exemplo:

- **Busca:** permite recuperar documentos ou frases mais relevantes em relação a uma consulta textual, com base na similaridade entre os vetores da consulta e os documentos.
- **Agrupamento:** textos com representações vetoriais semelhantes podem ser agrupados automaticamente, facilitando a identificação de tópicos ou categorias sem supervisão.
- **Recomendações:** itens descritos por textos semelhantes — como artigos, produtos ou conteúdos — podem ser recomendados ao usuário com base na proximidade vetorial de seus *embeddings*.
- **Deteção de anomalias:** textos que se afastam significativamente dos padrões observados no espaço de *embedding* podem ser tratados como anomalias.
- **Medição de diversidade:** ao analisar a distribuição das similaridades entre *embeddings*, é possível quantificar a diversidade semântica de um conjunto de textos.

- **Classificação:** por fim, *embeddings* podem ser usados em modelos de classificação, atribuindo rótulos a textos com base na similaridade com exemplos previamente rotulados.

Para utilizar *embeddings* em tarefas como as listadas acima, é necessário quantificar o quão próximos dois vetores estão no espaço vetorial. As duas métricas mais comuns para esse fim são a distância euclidiana e a similaridade cosseno.

A distância euclidiana mede a menor distância entre dois pontos, dada como o comprimento do segmento de reta que os conecta. Para vetores de n dimensões, a distância euclidiana pode ser calculada como:

$$d(u, v) = \sqrt{\sum_{i=1}^n (u_i - v_i)^2}$$

A similaridade de cosseno calcula o cosseno do ângulo entre dois vetores, operando matematicamente no intervalo de $[-1, 1]$, onde -1 indica vetores diametralmente opostos e 1 vetores alinhados. No entanto, no contexto de *embeddings* textuais, os valores de similaridade tipicamente variam entre 0 e 1. Isso ocorre porque, nesses espaços vetoriais semânticos, textos não similares tendem a ser ortogonais, não possuindo representações vetoriais “negativas” ou opostas no sentido geométrico estrito.

$$\text{sim}(u, v) = \frac{uv}{\|u\| \|v\|} = \frac{\sum_{i=1}^n u_i v_i}{\sqrt{\sum_{i=1}^n u_i^2} \sqrt{\sum_{i=1}^n v_i^2}}$$

Como os *embeddings* codificam relações semânticas por meio de vetores em espaços de alta dimensão, é possível realizar operações matemáticas entre essas representações para extrair significados implícitos ou analogias entre palavras, como por exemplo:

Rei – Homem + Mulher $\approx$ Rainha

Rei : Homem :: Rainha : Mulher

Brasil : Brasília :: Portugal : Lisboa

Comer : Comendo :: Andar : Andando

Apesar dos *embeddings* serem representados como vetores n-dimensionais, suas dimensões não são diretamente interpretáveis como variáveis tradicionais em um banco

de dados. Os modelos aprendem representações distribuídas, onde aspectos semânticos das palavras são codificados de forma latente e espalhados por múltiplas dimensões do vetor, ou seja, não existe uma única dimensão que represente “gênero” ou outra característica. Modelos mais modernos produzem *embeddings* mais densos, com ainda menos chance de interpretar dimensões individualmente.

No entanto, mesmo que não seja possível interpretar cada dimensão isoladamente, a Tabela 2 apresenta exemplos hipotéticos de textos e seus respectivos vetores de *embeddings*, truncados para visualização. Nesse exemplo, quanto mais próximo gênero está de 1, mais masculino, enquanto quanto mais próximo de -1, mais feminino. Outras características como realeza, força e inteligência possuem seu valor máximo quando estão mais próximo de 1, enquanto a proximidade com 0 sugere neutralidade.

Tabela 2 – Exemplos hipotéticos de palavras e seus possíveis vetores de *embedding*.

Dimensão	Rainha	Rei	Mulher	Homem
Gênero	-0,93	0,90	-0,88	0,91
Realeza	0,92	0,94	0,09	0,04
Vegetal	0,01	0,00	0,03	0,02
⋮	⋮	⋮	⋮	⋮
Força	-0,20	0,85	-0,24	0,80
Inteligência	0,65	0,80	0,62	0,76

Fonte: Elaborado pelo autor.

Embora *embeddings* sejam ferramentas poderosas para representar texto em forma vetorial, diversos estudos mostram que eles podem reproduzir preconceitos sociais presentes nos dados de treinamento. Isso ocorre porque os *embeddings* são aprendidos a partir de notícias, livros e páginas da internet, que refletem os estereótipos, desigualdades e vieses da sociedade (BRUNET et al., 2019). Isso foi representado nesse exemplo hipotético ao aproximar o gênero masculino a adjetivos como força e inteligência.

Um exemplo prático de viés de gênero observado em modelos de *embedding* pode ser exemplificado através do Word2Vec, nos quais operações vetoriais como:

Homem : Programador :: Mulher : ?

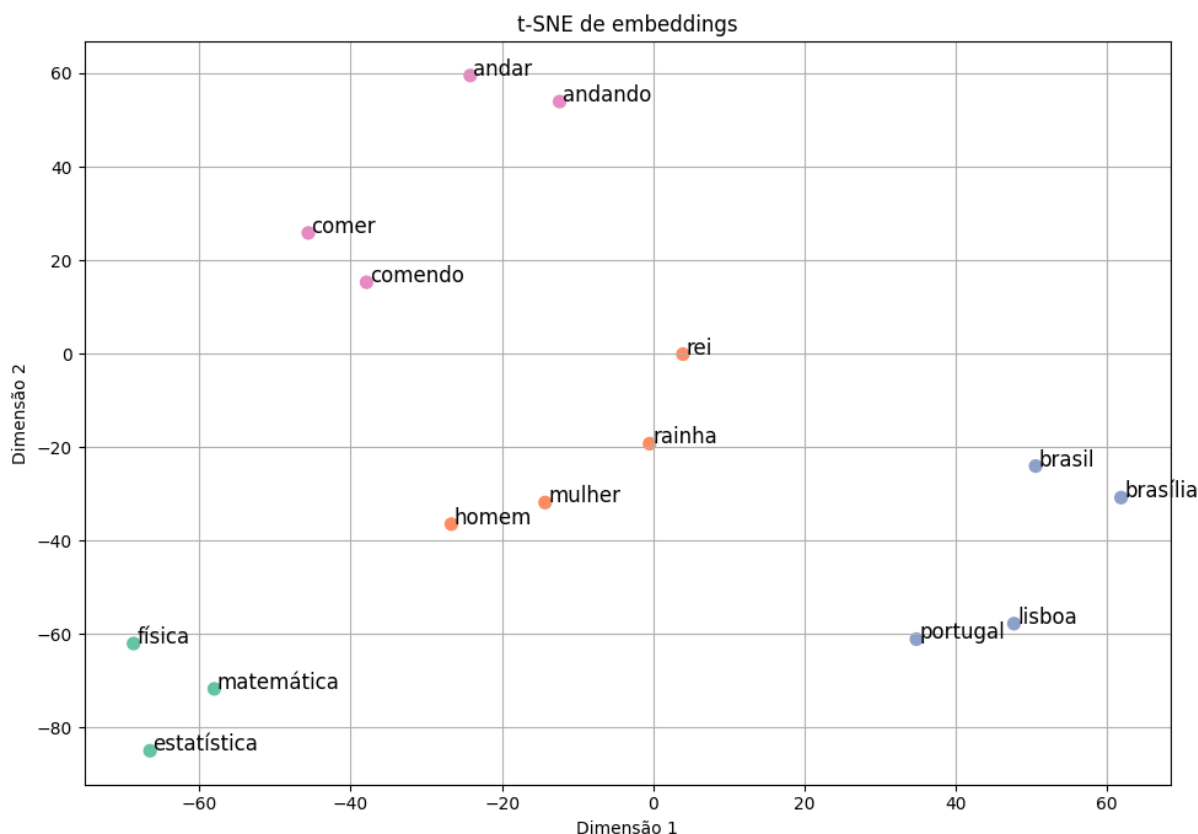
frequentemente retornavam termos como “dona de casa”, refletindo estereótipos de papéis sociais (MAY et al., 2019). Além disso, estudos identificaram vieses raciais e étnicos: palavras associadas a grupos minoritários aparecem mais próximas de termos como “crime” ou “pobreza” do que de conceitos neutros ou positivos. Além disso, também foram detectados outros vieses culturais e religiosos, com termos ligados a determinadas religiões ou regiões sendo indevidamente associados a ideias extremistas ou negativas.

Esses vieses têm implicações práticas, podendo afetar sistemas de recomendação e enviar modelos preditivos, oferecendo riscos éticos em aplicações sensíveis, como seleção de candidatos, decisões judiciais automatizadas, dentre outros.

Diferentes estratégias vêm sendo propostas para mitigar esses efeitos. Entre elas, destaca-se o *Word Embedding Association Test* (WEAT) (BRUNET et al., 2019). Outras abordagens incluem uma curadoria mais cuidadosa dos dados de treinamento e a incorporação de avaliações humanas e éticas durante o desenvolvimento e a aplicação dos modelos.

Também é possível utilizar técnicas de redução de dimensionalidade, como a Análise de Componentes Principais (PCA) ou o *t-distributed Stochastic Neighbor Embedding* (*t*-SNE), para visualizar essas representações vetoriais em planos 2D ou 3D e preservando a estrutura de agrupamento dos vetores originais.

O *t*-SNE é uma técnica não linear projetada para a visualização de dados de alta dimensão. Seu principal objetivo é preservar a estrutura local dos dados, garantindo que pontos que são vizinhos no espaço de alta dimensão também sejam representados como vizinhos no espaço de baixa dimensão (2D ou 3D). Para isso, o *t*-SNE modela as similaridades entre os pontos como probabilidades. O ponto crucial da técnica é o uso da distribuição *t* de Student no mapa de baixa dimensão, que, por possuir caudas mais pesadas que a distribuição normal, permite que pontos moderadamente distantes no espaço original sejam afastados consideravelmente no mapa final. Também é importante mencionar que as distâncias relativas e os tamanhos dos agrupamentos em um gráfico *t*-SNE não são diretamente interpretáveis (MAATEN; HINTON, 2008).

Figura 2 – Representação conceitual de um espaço de *embeddings* utilizando *t*-SNE.

Fonte: Elaborado pelo autor.

A Figura 2 foi gerada utilizando a técnica de *t*-SNE sobre *embeddings* obtidos a partir do modelo `text-embedding-3-small`, da OpenAI. Ela ilustra, de maneira conceitual, como palavras semanticamente relacionadas tendem a se agrupar no espaço vetorial. Por exemplo, os termos estatística, física e matemática estão próximos entre si, refletindo sua afinidade conceitual, enquanto rainha, rei, mulher e homem também formam um agrupamento coerente com base em relações semânticas, da mesma forma que os verbos e suas conjugações e os países e suas capitais.

Também existe uma limitação no tamanho do texto que pode ser processado para gerar um *embedding*; essa limitação impacta na capacidade de capturar informação de um texto muito longo. Modelos como `text-embedding-3-small` e `text-embedding-3-large` (OpenAI), possuem um limite máximo de 8.192 *tokens* por *embedding*. Além disso, mesmo dentro desse limite, textos muito longos podem ter *embeddings* menos específicos, gerando perda de informação. Isso ocorre principalmente em textos muito densos, nos quais o modelo não é capaz de capturar integralmente a variabilidade semântica presente.

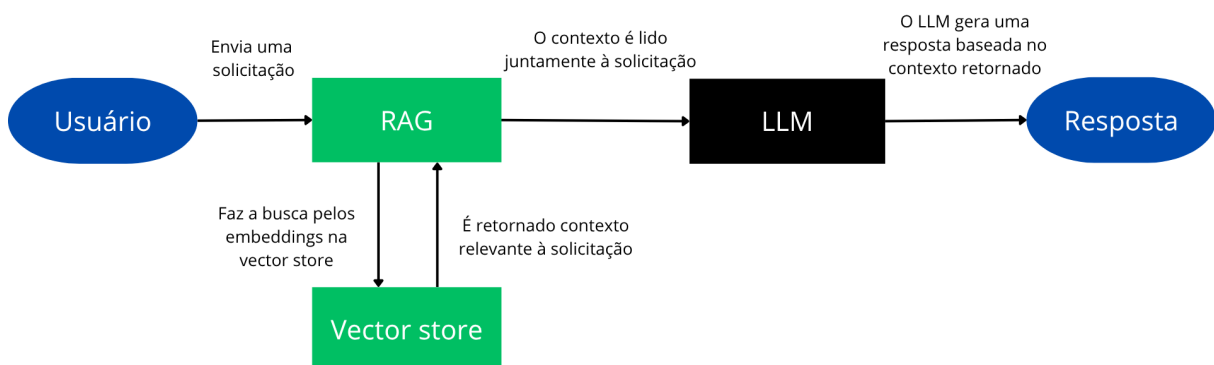
4.6 Retrieval-Augmented Generation (RAG)

O RAG combina LLMs com informações externas para gerar respostas mais precisas e relevantes a um contexto específico. Em vez de utilizar apenas o conhecimento do modelo, aprendido durante seu treinamento, o RAG utiliza trechos de documentos, bases de dados ou outros arquivos relevantes, e injeta esse conteúdo no *prompt* no momento da inferência, fornecendo o contexto necessário para responder o *prompt* gerado.

Essa abordagem é muito importante em cenários onde o modelo precisa responder sobre tópicos que não fazem parte de seu conjunto de treinamento, como documentos internos de uma organização, legislações locais ou, no contexto deste trabalho, livros e artigos científicos necessários para uma boa fundamentação estatística. O processo de RAG, como apresentado na Figura 3, envolve duas etapas principais:

1. Recuperação: dado um *prompt*, o sistema realiza uma busca semântica em uma base de dados, retornando os trechos mais relevantes de acordo com a similaridade com a consulta.
2. Geração: a partir daí esses trechos são injetados no *prompt*, e então enviados ao modelo de linguagem, que utiliza essas informações para compor uma resposta mais fundamentada.

Figura 3 – Diagrama de funcionamento de um sistema de RAG implementado a um LLM.



Fonte: Elaborado pelo autor.

O RAG costuma ser implementado com o uso de *embeddings* armazenados em bancos vetoriais (*vector stores*). Durante o processo de inferência, os documentos semanticamente mais próximos do *prompt* são selecionados e utilizados como contexto. Ele é especialmente útil em sistemas personalizados baseados em LLMs, pois permite incorporar informações recentes ou não treinadas, reduzir o risco de alucinação e personalizar respostas.

O RAG é uma ferramenta poderosa quando aliado a LLMs, pois combina o poder de geração com a capacidade de acesso dinâmico a informações. Em contextos estatísticos ou de ciência de dados, é possível criar assistentes especializados que interpretam bases de dados, explicam ou geram código e analisam ou geram documentos técnicos, tudo isso sem perder o rigor estatístico.

4.7 *Vector stores*

Para que o RAG funcione de forma eficiente, é essencial contar com mecanismos que realizem o armazenamento para posterior consulta. Esse papel é desempenhado por estruturas chamadas de *vector stores*.

Vector stores são sistemas otimizados para armazenar e recuperar vetores, como *embeddings*, em consultas por similaridade semântica. Em vez de realizar buscas por palavras-chave, como em bancos de dados tradicionais, os *vector stores* permitem encontrar vetores semelhantes a uma consulta com base em cálculos já explicados, como a similaridade cosseno.

É possível criar um *embedding* de um documento inteiro, entretanto, além da limitação de *tokens*, gerar *embeddings* a partir de textos muito grandes pode resultar em perda de informação. Para mitigar esse problema, existem estratégias de criação de *vector stores*, como o *chunking*, que consiste em dividir o texto em blocos menores, chamados *chunks*, que serão convertidos em *embeddings* e armazenados em uma *vector store* para consultas futuras.

A divisão de documentos em *chunks* pode variar conforme a necessidade. Em textos bem estruturados, como obras literárias, é possível dividir seções inteiras como unidades de *chunk* sem perda de contexto. Outra abordagem muito utilizada para isso é o *chunking* com sobreposição. Esse método consiste em dividir o texto em blocos com uma quantidade fixa de *tokens* máximos, mas com uma sobreposição entre blocos consecutivos, ou seja, *tokens* do final de um *chunk* que serão repetidos no início do próximo. Essa sobreposição é importantíssima para manter a coesão entre os trechos, evitando perda de sentido por cortes abruptos entre *chunks*.

Sendo assim, se um documento for dividido em *chunks* de no máximo 200 *tokens* com uma sobreposição de 50 *tokens*, isso significa que os *tokens* 1–200 formarão o primeiro *chunk*, os *tokens* 151–350 formarão o segundo *chunk*, e assim por diante. A Tabela 3 ilustra esse processo para um texto com 1.000 *tokens*.

Tabela 3 – Exemplo de divisão de texto de 1.000 *tokens* em *chunks* de 200 *tokens* com sobreposição de 50 *tokens*.

<i>Chunk</i>	<i>Token Inicial</i>	<i>Token Final</i>
1	1	200
2	151	350
3	301	500
4	451	650
5	601	800
6	751	950
7	901	1000

Fonte: Elaborado pelo autor.

Para que essa estratégia funcione corretamente, é necessário garantir que os *tokens* de sobreposição seja estritamente menor que o total de *tokens* no *chunk*. Caso contrário, os blocos deixariam de avançar no texto, resultando em repetição infinita dos mesmos *tokens* e inviabilizando a construção da *vector store*. Valores de sobreposição muito pequenos podem fazer com que trechos dependentes fiquem separados em blocos distintos, prejudicando a recuperação de informações, enquanto um valor muito grande aumenta a redundância entre os *chunks*, além de elevar o custo computacional e o tamanho da *vector store*.

Sendo assim, a definição dos parâmetros para o *chunking* deve ser feita considerando o contexto do problema e o texto que será convertido. Textos mais densos, como artigos científicos, podem exigir *chunks* menores para preservar o significado de cada trecho. Já textos mais narrativos, como notícias ou relatórios, podem tolerar *chunks* maiores, pois o conteúdo relevante costuma estar mais distribuído ao longo do texto. Além disso, conteúdos muito fragmentados exigem uma sobreposição maior para manter a coerência entre os *chunks*, enquanto em textos bem segmentados ou com tópicos independentes, uma sobreposição pequena pode ser suficiente, reduzindo redundância.

O processo de uso de um *vector store* geralmente segue três etapas principais: a indexação, onde os documentos são processados e transformados em *embeddings*, que são armazenados junto com seus metadados; a consulta, em que, ao receber um *prompt*, ele é convertido em um *embedding*; a busca por similaridade, quando o *vector store* retorna os vetores mais próximos do vetor da consulta, recuperando *chunks* relevantes.

Para prover uma base de conhecimento específica para os agentes, foi criada uma *vector store* no âmbito deste trabalho. Para isso foram utilizados alguns livros de estatística, como Bussab e Morettin (2010), Montgomery (2017), Montgomery (2020), Montgomery, Peck e Vining (2021), James et al. (2023), Paula (2004), Cordeiro e Demétrio (2008). Essa base de conhecimento foi estruturada para garantir cobertura teórica abrangente, cobrindo fundamentos de probabilidade e análise exploratória até técnicas de modelagem

estatística.

4.8 Ferramentas

O sistema proposto neste trabalho utiliza uma arquitetura baseada em agentes, cada um com acesso a ferramentas específicas que estendem as capacidades dos LLMs para além do conhecimento aprendido durante seu treinamento. Essas ferramentas permitem que os agentes realizem buscas em tempo real, consultem arquivos externos, executem códigos e até acessem funções personalizadas, compondo um fluxo mais adaptável para a geração de relatórios estatísticos. Até o momento, as ferramentas utilizadas estão sendo *web search* e *file search*, mas a seguir são descritas algumas das principais ferramentas que podem ser integradas aos agentes:

- **Web search:** A ferramenta de *web search* concede aos agentes a capacidade de realizar buscas na internet em tempo real. Os LLMs são treinados com vastos corpos de texto, mas seu conhecimento é limitado à data de corte de seu treinamento. Essa ferramenta supera essa limitação ao prover acesso a informações atuais, eventos recentes, artigos de notícias e dados contextuais que não estariam disponíveis de outra forma. No âmbito deste projeto, a ferramenta de *web search* é utilizada principalmente para enriquecer a seção de Introdução do relatório. Ao pesquisar o tema da análise, o agente pode tecer uma narrativa mais envolvente e contextualizada, fundamentando o problema.
- **File search:** Essa ferramenta é feita para operar sobre um *vector store*, agindo como uma implementação de RAG e permitindo que os agentes fundamentem suas respostas não apenas em seu conhecimento geral, mas em fontes de informação específicas fornecidas. No contexto deste TCC, foi configurado uma *vector store* com base em livros de estatística. Assim, quando qualquer agente for descrever, por exemplo, um teste de hipótese, pode utilizar a ferramenta para buscar e citar a fórmula matemática exata ou a definição formal de um conceito, aumentando o rigor técnico do relatório final.
- **Code interpreter:** O interpretador de código trata-se de um ambiente de execução *sandboxed*, um ambiente seguro e isolado, onde os agentes podem executar códigos mas não possuem acesso direto ao restante do sistema operacional.
- **Code interpreter:** O interpretador de código trata-se de um ambiente de execução *sandboxed*, um ambiente seguro e isolado, onde os agentes podem executar códigos mas não possuem acesso direto ao restante do sistema operacional. Na arquitetura utilizada (baseada na infraestrutura do Google), esse ambiente opera sob certas restrições: o tempo de execução é limitado a um máximo de 30 segundos por

interação e o acesso é restrito a um conjunto de bibliotecas (como `pandas`, `scipy` e `scikit-learn`).

- **Funções personalizadas:** Também é possível definir funções personalizadas como ferramentas. Isso permite aplicações bastante flexíveis, como estabelecer conexão a um banco de dados, interagir com uma API ou qualquer outra função específica.
- ***Computer user agent (CUA)*:** é um tipo de agente capaz de interagir diretamente com o sistema operacional, simulando ações típicas de um usuário humano. Isso inclui mover e clicar o *mouse*, digitar textos, abrir janelas, navegar por aplicações e capturar imagens da tela.

5 EXPERIMENTOS E ANÁLISE DE RESULTADOS

Este capítulo apresenta a estrutura, o planejamento, os mecanismos de execução e avaliação dos experimentos realizados para validar a proposta deste trabalho, além de seus resultados. O sistema desenvolvido utiliza uma arquitetura multiagente baseada inteiramente em modelos da família *Google Gemini 2.5*, por meio de uma chave de API pessoal de conta paga. A escolha por esta família de modelos justifica-se por fatores como boa performance geral dos modelos, janela de contexto considerável e integração com ferramenta de execução de código. Apesar disso, a estrutura do sistema permite customizações de APIs e modelos diferentes dos utilizados.

A Tabela 4 apresenta a estrutura de custos dos modelos utilizados. O modelo Gemini 2.5 Pro, voltado para raciocínio e tarefas mais complexas, possui um custo de *output* 4 vezes superior ao da versão Flash, que equilibra latência e performance, e 25 vezes superior ao da versão Flash-Lite, otimizada para tarefas leves e máxima eficiência de custo.

Tabela 4 – Custo de Cada Modelo Utilizado por Milhão de *tokens* (US\$).

Modelo	<i>Input</i> (US\$)	<i>Output</i> (US\$)
Gemini 2.5 Pro	1,25	10,00
Gemini 2.5 Flash	0,30	2,50
Gemini 2.5 Flash-Lite	0,10	0,40

Fonte: Elaborada pelo autor com base em dados do Google.

É fundamental destacar que, na precificação de LLMs, os *tokens* de raciocínio são contabilizados como *output*. Portanto, agentes que utilizam cadeias de pensamento extensas, como o Agente Avaliador que será demonstrado nos experimentos, incidem diretamente na tarifa mais elevada da tabela, o que justifica a necessidade de monitoramento da relação entre profundidade de análise e gasto financeiro.

Para garantir uma resposta mais consistente, precisa e acessível, dois agentes utilizam um modelo de saída estruturada baseada em esquemas de dados da biblioteca Pydantic. Essa abordagem força o modelo a retornar, ao invés de um texto livre, um objeto em formato análogo ao *JavaScript Object Notation* (JSON).

Esse sistema é composto por um conjunto de agentes, cada um com uma instrução e tarefa específica. Essa especialização permite decompor a tarefa de gerar um relatório em subtarefas gerenciáveis, também aumentando a precisão do resultado final. Os agentes responsáveis pela feitura do relatório são orientados a seguir formatação do texto em código `LaTeX`.

O objetivo central dos experimentos é demonstrar a robustez do sistema frente a diferentes níveis de complexidade e contexto de dados e perguntas, avaliando sua capacidade de gerar documentos precisos e coesos.

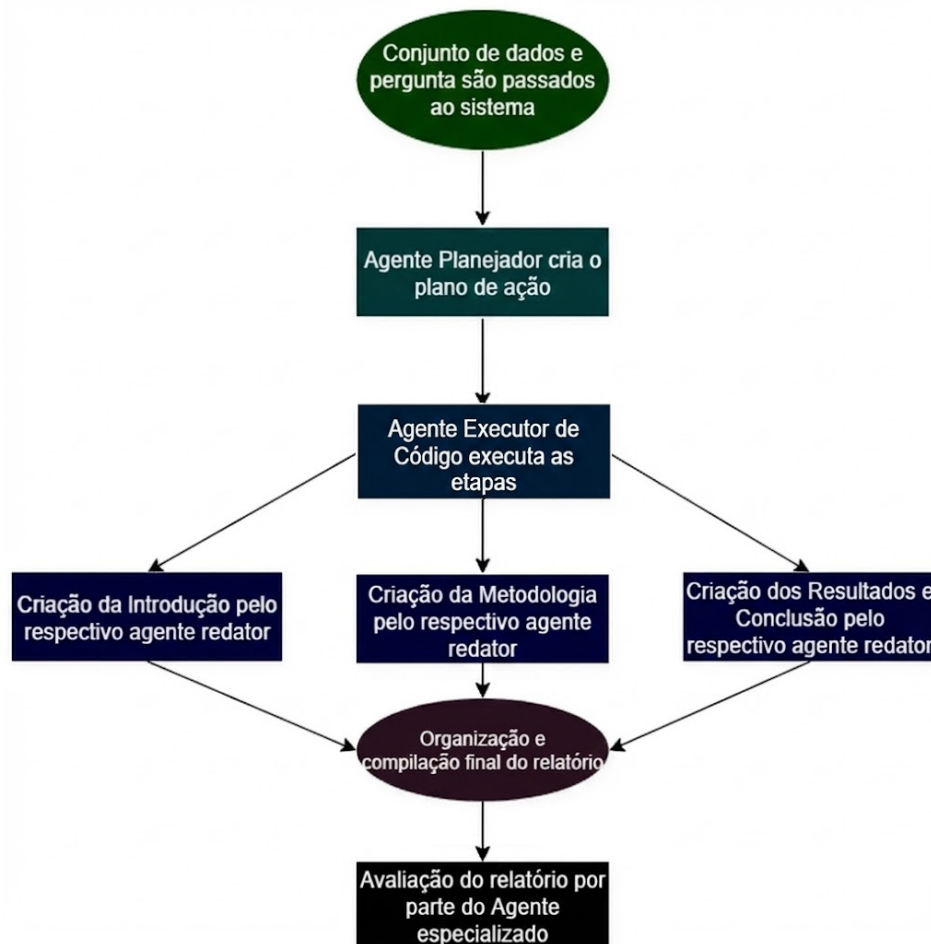
Os experimentos foram conduzidos usando a linguagem de programação Python. As instruções são apresentadas no Apêndice A e os modelos de saída estruturada no Apêndice B.

5.1 Cenário Experimental e Métricas

5.1.1 Arquitetura e Fluxo de Execução

O fluxo de processamento de um experimento é decomposto em oito fases, orquestradas por um controlador central em Python. A Figura 4 ilustra o fluxo deste processo.

Figura 4 – Fluxograma dos Experimentos, da Solicitação à Avaliação do Relatório.



Fonte: Elaborado pelo autor.

O fluxo de trabalho inicia-se com o recebimento de duas entradas principais: o conjunto de dados e a consulta do usuário (pergunta). O sistema é compatível com os formatos `.xlsx`, `.xls`, `.csv`, `.parquet`, `.ascii` e `.txt`. Além do carregamento local, também é possível importar dados de *Uniform Resource Locators* (URLs) externas, como de repositórios do GitHub, ou através de IDs de *datasets* da plataforma Kaggle.

Essas informações são enviadas para o Agente Planejador, que utiliza o modelo Gemini 2.5 Pro, o mais capaz da família em termos de raciocínio e abstração. Essa fase utiliza o modelo de saída estruturada, separando bem o desenho geral do plano e a estratégia da sua execução, aderindo estritamente ao seguinte esquema:

- **Título do Relatório:** Título do relatório que será gerado.
- **Objetivo Principal:** Uma reescrita formal do problema do usuário.
- **Etapas:** Uma lista ordenada de objetos **Etapas**, um passo a passo atômico e sequencial para resolver o problema do usuário, contendo:
 - **Nome:** Um identificador curto da **Etapas** (ex: “Cálculo de Correlação”).
 - **Instruções Técnicas:** Uma lista de diretrizes para o executor de código (ex: “Calcular correlação de Pearson e Spearman”). Esse campo é crucial por traduzir a estratégia estatística em comandos diretos.
- **Limitações Previstas:** Uma lista antecipando possíveis barreiras da análise (ex: “Falta de dados temporais impede a detecção de sazonalidade”).
- **Análises Adicionais:** Uma lista com sugestões de análises posteriores (ex: “Coleta de dados temporais para análise de sazonalidade”).

A terceira fase é a implementação do plano de forma sequencial através do Agente Executor de Código (Agente Executor). Para isso, utiliza-se o Gemini 2.5 Flash (modelo intermediário da família) operando de forma sequencial para executar cada **Etapas**. Embora tenha acesso ao plano gerado anteriormente por completo, ele realiza a execução das **Etapas** individualmente. Essa estratégia sequencial maximizou a performance e a precisão das respostas, impedindo que o modelo tente resolver problemas complexos simultaneamente e se perca no raciocínio. Além disso, esse agente foi implementado com memória persistente, garantindo que ele retenha o contexto de **Etapas** anteriores para evitar redundâncias.

O fluxo de execução de código para cada **Etapas** funciona da seguinte maneira:

1. O Agente lê o *prompt* com o nome e as instruções técnicas específicas da **Etapas**, situando o modelo sobre onde ele está no progresso do plano.

2. A partir daí o agente pode raciocinar sobre a **Etap**a se ela for suficientemente complexa.
3. O agente gera e executa o código Python no ambiente *sandboxed*.
4. Por fim, é gerado um *output* textual com breves considerações acerca do código gerado.
5. Após a geração do *output*, existe a possibilidade autocorreção em caso de erro, onde o agente pode corrigir o próprio código iterativamente.

Este mecanismo tenta simular um fluxo de trabalho de um onde as análises são construídas incrementalmente sobre as anteriores. Os artefatos gerados durante a execução do Agente Executor são armazenados para a composição do relatório final. Isso inclui os *outputs*, os códigos desenvolvidos e os resultados de suas respectivas execuções, além das imagens produzidas em binário.

Concluídas essas três fases iniciais, os agentes redatores entram em operação de forma paralela para a composição do texto final, divididos em Introdução, Material e Métodos (Metodologia), além de Resultados e Conclusão.

Para a fase de Introdução temos uma divisão em duas partes. A primeira utiliza o modelo Gemini 2.5 Flash e tem acesso à ferramenta de *web search*. Essa parte recebe informações do conjunto de dados e da pergunta e é responsável por toda a contextualização acerca do problema abordado. O objetivo é gerar uma introdução que conecte os dados e a pergunta ao mundo real. A segunda parte utiliza o modelo Gemini 2.5 Flash-Lite, o mais otimizado da família e ideal para tarefas de classificação e sumarização que não exija raciocínio. Aqui todo o plano é enviado no *prompt* para uma breve explicação do que será feito no relatório.

A fase da Metodologia também é segmentada, mas processada integralmente pelo Gemini 2.5 Flash-Lite. Inicialmente, são fornecidas informações adicionais do conjunto de dados para gerar uma subseção de Materiais, com uma breve explicação sobre o conjunto utilizado. Em seguida, com auxílio da ferramenta *file search*, detalham-se todas as metodologias utilizadas para a elaboração da subseção de Métodos.

A próxima fase fica a cargo do redator de Resultados e Conclusão, baseado no Gemini 2.5 Flash. Esse redator tem a tarefa de interpretar e transcrever os resultados técnicos gerados na fase de execução de código, correlacionando-os com a pergunta original e discutindo se as hipóteses foram validadas. Além disso, discute as restrições do estudo e propõe trabalhos futuros, fundamentando-se, respectivamente, nos campos de **Limitações Previstas** e **Análises Adicionais** do planejamento.

A sétima fase é determinística, ou seja, não baseada em LLMs. Um módulo em Python consolida os fragmentos textuais e visuais em um *template* mestre, executando a

compilação através de uma *engine* L^AT_EX local. Este processo inclui rotinas de tratamento de sintaxe via *Regular Expression* (RegEx), conversão de binários de imagem para formato .png, formatação de *hyperlinks* para a organização das referências consultadas no *web search* e a estruturação automática de sumário, seções e apêndice.

Por fim, o relatório segue para a fase de Avaliação, detalhada na Subseção 5.1.4.

5.1.2 Configuração e Cenários dos Experimentos

Para validar a robustez e a generalização do sistema multiagente desenvolvido, foi desenhada uma estratégia experimental. Esta abordagem visa superar a validação baseada apenas em “casos de sucesso”, submetendo o sistema a cenários de estresse, ambiguidade e inconsistência em diferentes tipos de dados e contextos que refletem os desafios reais da Estatística.

A metodologia de testes fundamenta-se na combinação controlada de três variáveis principais: a dificuldade dos dados de entrada, a dificuldade da pergunta do usuário e uma categoria qualitativa da pergunta.

Para garantir a validação a capacidade de generalização, foi curado um repositório diversificado de conjuntos de dados. As bases de dados selecionadas apresentam diferentes dimensionalidades, além de diferentes tipos de variáveis (numéricas, categóricas, temporais e textuais). O escopo temático é multidisciplinar, abrangendo diferentes domínios como epidemiologia, varejo, ciências ambientais e mercado de trabalho.

Essa heterogeneidade nos dados de entrada acompanha um amplo espectro de técnicas solicitadas. Foram validados cenários que exigem desde a análise exploratória e limpeza de dados até abordagens como testes de hipóteses, análise multivariada, séries temporais e algoritmos de *machine learning*. Essa estratégia assegura que o sistema seja capaz de elaborar soluções para diferentes tipos de problemas. Complementarmente, foram aplicados cenários de simulação. Nesse tipo de experimento, o sistema não recebe arquivos de dados, precisando simular dados para resolver o problema.

Os conjuntos de dados utilizados nos experimentos foram classificados em três níveis de dificuldade, baseando-se em aspectos como no tamanho, necessidade de pré-processamento e clareza de suas variáveis. A Tabela 5 detalha esta classificação.

Tabela 5 – Tabela dos Níveis de Dificuldade dos Dados.

Nível	Descrição	Exemplo
1	Dados perfeitamente estruturados, limpos e com colunas nomeadas de forma intuitiva e em pouca quantidade.	<i>Iris</i>
2	Dados de dimensionalidade média e que exigem tratamentos leves. Representam o cenário comum da maioria dos dados.	<i>Titanic</i>
3	Dados com alta dimensionalidade, graves problemas de qualidade, estrutura irregular e nomes pouco explicativos.	Dados de varejo

Fonte: Elaborado pelo autor.

As perguntas submetidas ao sistema também foram classificadas, dessa vez em cinco níveis de complexidade, exigindo do Agente Planejador e do Executor capacidades técnicas e “cognitivas” crescentes. A Tabela 6 apresenta essa escala.

Tabela 6 – Tabela dos Níveis de Dificuldade das Perguntas.

Nível	Descrição	Exemplo
1	Pergunta tecnicamente simples e semanticamente direta.	Qual é a média de idade dos segurados?
2	Exige operações um pouco mais completas como agregação, filtragem e ordenação, além de uma inferência semântica para relacionar variáveis.	Quais províncias do Paquistão são as mais frias?
3	Questões investigativas que demandam interpretação de tendências, correlações, aplicação de testes de hipóteses ou outras técnicas não triviais.	Com base nos dados, podemos afirmar que existe aquecimento global?
4	Envolve análises com técnicas mais complexas e alta capacidade de interpretação do sistema.	Faça uma análise multivariada completa das variáveis.
5	O nível mais alto, normalmente perguntas de estresse, exigindo um passo a passo de ações baseadas em múltiplos fatores.	Faça todo o tratamento necessário e treine um modelo para previsão de custos médicos.

Fonte: Elaborado pelo autor.

Além da métrica quantitativa, cada experimento foi rotulado com uma categoria qualitativa para avaliar comportamentos específicos do sistema, como possibilidade de alucinação, interpretação e capacidade de inferência de contexto. As categorias definidas foram:

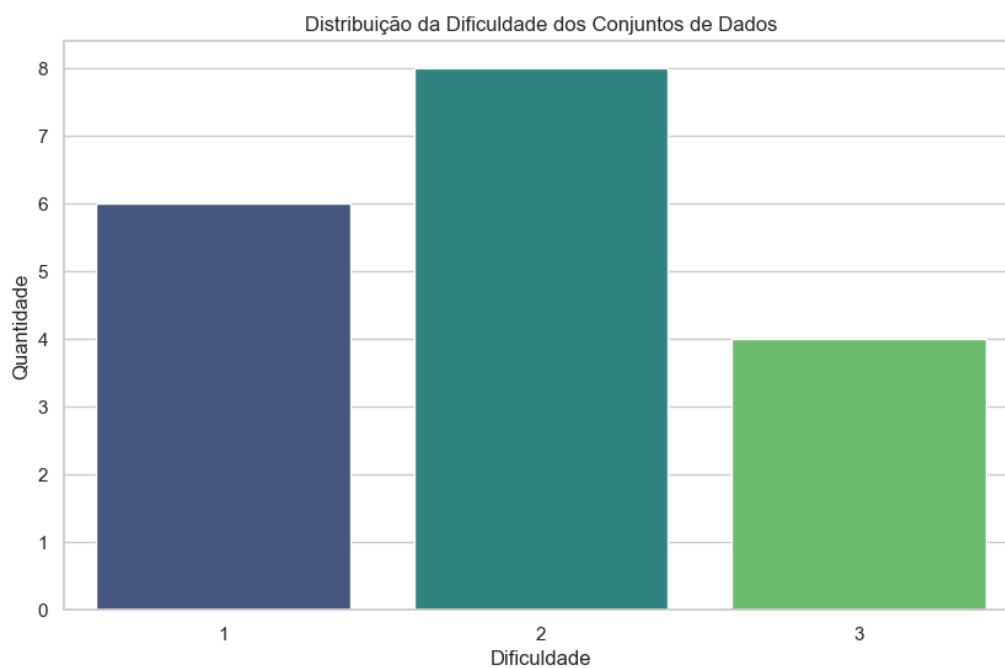
Tabela 7 – Tabela das Categorias de Perguntas.

Categoria	Descrição	Exemplo
Caminho Feliz	Perguntas claras e diretas, servem como linha de base para o desempenho do sistema.	Qual é a média da taxa de desemprego ao longo de todo o período listado?
Ambiguidade	Perguntas intencionalmente vagas para avaliar a capacidade de inferir critérios e lidar com cenários incertos.	Quais os maiores pássaros? (dados contém diversas medidas de tamanho)
Alucinacao Induzida	Testes que envolvem solicitação de dados inexistentes para avaliar o grau de alucinação do sistema	Qual é a taxa de desemprego por gênero ao longo do tempo? (sem haver coluna de gênero nos dados).
Pedido Incoerente	Solicitações tecnicamente erradas para testar o rigor estatístico do sistema.	Gere e interprete um gráfico de barras das notas de matemática
Interpretação	Comandos indiretos que não explicitam a técnica a ser utilizada e exigem capacidade de dedução.	Qual foi a variação da qualidade de vida das pessoas ao longo do tempo?
Técnica	Desafios focados em técnicas estatísticas específicas.	Usando uma Árvore de Decisão, crie uma heurística para prever os sobreviventes.
Simulação	Uma categoria especial onde nenhum conjunto de dados é fornecido. O sistema deve gerar dados sintéticos via código para responder a problemas teóricos de simulação.	Simule o lançamento de dois dados de 6 faces 10000 vezes. Compare as probabilidades empíricas das somas dos dados com as teóricas.
Estresse	Perguntas multifacetadas e complexas, feitas para testar a eficiência do sistema.	Realize um PCA completo e utilize o(s) componente(s) para criar um modelo de previsão de mortalidade.

Fonte: Elaborado pelo autor.

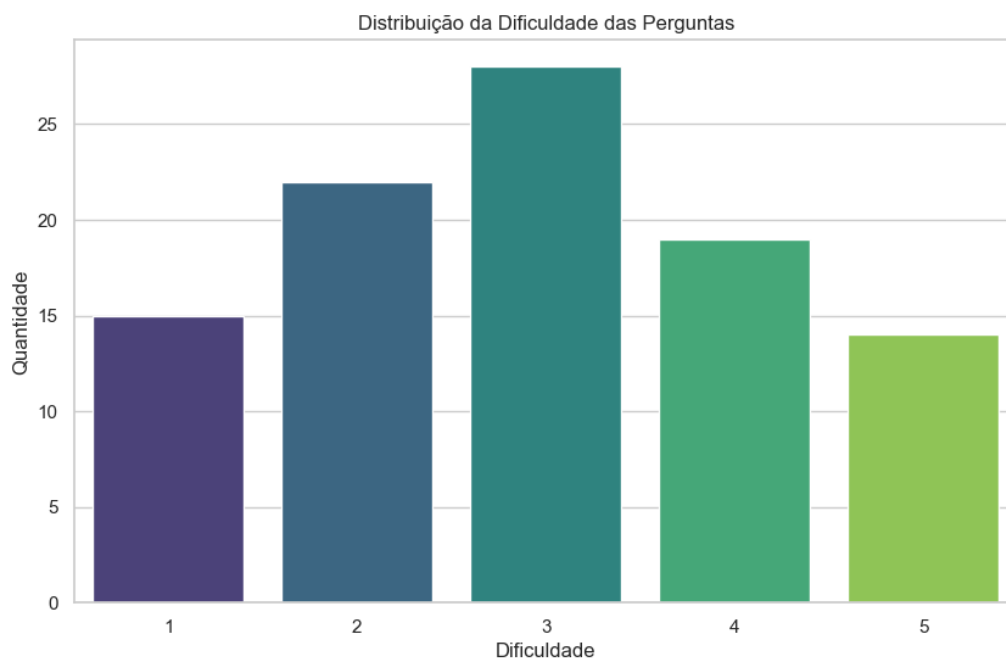
Para a execução dos experimentos foram separados 17 conjuntos de dados e uma série de problemas de simulação, com um total de 98 perguntas. A distribuição das dificuldades dos conjuntos de dados, dificuldades das perguntas e categorias das perguntas estão apresentadas, respectivamente, nas Figuras 5, 6 e 7

Figura 5 – Gráfico da Distribuição das Dificuldades dos Conjuntos de Dados.



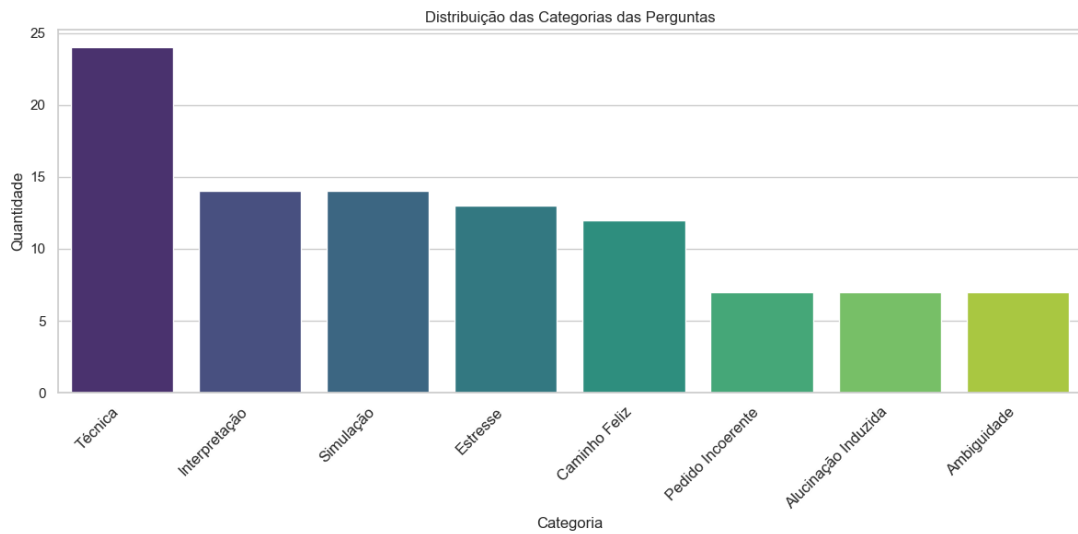
Fonte: Elaborado pelo autor.

Figura 6 – Gráfico da Distribuição das Dificuldades das Perguntas.



Fonte: Elaborado pelo autor.

Figura 7 – Gráfico da Distribuição das Categorias das Perguntas.



Fonte: Elaborado pelo autor.

5.1.3 Armazenamento de Dados

Para garantir reprodutibilidade, auditoria e avaliação dos experimentos, o sistema possui uma arquitetura de armazenamento de dados onde cada execução é tratada como uma unidade, gerando um conjunto estruturado de arquivos que capturam desde o prompt inicial até a avaliação final, incluindo métricas dos agentes e registros de sistema.

Para cada experimento, é criado uma pasta com um identificador de tempo (data e hora da execução). Dentro dessa pasta, são salvos os registros de execução do sistema. Além disso, todos os dados gerados são serializados em arquivos JSON, permitindo análises posteriores: todos os *inputs* e *outputs* dos agentes são salvos, incluindo os códigos gerados pelo Agente Executor. Além disso, outros dados como tempo de execução e consumo de *tokens* (*input*, *output* e raciocínio) de cada agente também são salvos.

5.1.4 Avaliação Automática de Qualidade

Para a última fase do sistema foi desenvolvido um Agente Avaliador dedicado utilizando o modelo *Gemini 2.5 Pro*. Esse agente atua como um revisor estatístico simulado, responsável por avaliar os relatórios gerados pelos demais agentes. Sua persona é configurada para ser rigorosa, pragmática e orientada a resultados, penalizando alucinações e falhas metodológicas.

O processo de avaliação é estruturado em três dimensões principais, baseadas na estrutura do relatório, cada uma com critérios específicos e pesos distintos na composição da nota final:

1. **Introdução (Peso de 25%):** Avalia a capacidade de contextualização, se o problema foi explicado e conectado a um cenário do mundo real e se o roteiro de análise foi explicado de forma lógica, clara e resumida.
2. **Materiais e Métodos (ou Metodologia: Peso de 35%):** Foca no rigor técnico. Exige que as variáveis dos dados sejam descritas corretamente e que os métodos estatísticos escolhidos sejam justificados com embasamento teórico.
3. **Resultados e Conclusão (Peso de 40%):** Verifica se os gráficos foram interpretados corretamente, se a conclusão responde diretamente à pergunta do usuário e se há autocritica sobre as limitações da análise.

De forma análoga ao Agente Planejador, a saída do Agente Avaliador também segue um modelo estruturado para garantir consistência. A estrutura do esquema, para cada dimensão avaliada, é detalhada abaixo:

- **Nota:** Pontuação específica da seção (0-10).
- **Pontos Positivos:** Lista de acertos identificados, contendo:
 - **Tipo:** Categoria do acerto.
 - **Descrição:** Explicação do porquê aquilo foi considerado um acerto.
- **Erros Identificados:** Lista de falhas encontradas, contendo:
 - **Tipo:** Categoria do erro
 - **Gravidade:** Nível de severidade do erro (classificada em baixa, media, alta, critica).
 - **Penalidade:** Valor negativo de cada erro a ser descontado da nota (até -4 para para erros críticos).
 - **Descrição:** Explicação do porquê aquilo foi considerado um erro.
- **Nota Final Ponderada:** Um valor decimal (0-10) calculado automaticamente, para evitar erros de cálculo do LLM, pela soma ponderada das notas das dimensões.

Esse modelo de avaliação estruturada permite não apenas quantificar a qualidade dos relatórios gerados (através da nota final), mas também categorizar qualitativamente as falha e acertos mais comuns dos diferentes agentes do sistema.

5.2 Resultados Obtidos

Antes de apresentar os resultados, é fundamental estabelecer a posição da solução proposta frente às ferramentas de IA generativa convencionais vigentes. Embora o sistema utilize os mesmos modelos de linguagem que alimentam interfaces comerciais como o ChatGPT e o Google Gemini, a implementação via arquitetura multiagente introduz diferenças fundamentais de uso, especificamente quanto a escalabilidade, customização e rastreabilidade. Trata-se da diferença entre um modelo de assistência conversacional para um fluxo de automação.

As ferramentas de IA convencionais são projetadas para o paradigma síncrono, onde cada análise exige o *upload* manual de arquivos e uma nova sessão de engenharia de *prompt*, enquanto o sistema desenvolvido opera através de processamento em lote. A capacidade demonstrada de gerar quase uma centena de relatórios de forma autônoma evidencia a escalabilidade do sistema, ideal em cenários onde a demanda por análise supera a disponibilidade humana, servindo como um ponto de apoio e multiplicador, posicionando o estatístico para uma função de revisão e validação.

Além da escalabilidade, o sistema permite um nível de customização inacessível em ferramentas de IA convencionais, que oferecem uma experiência mais generalista. A separação dos *prompts* para cada agente permitiu ajustar o comportamento do sistema para tarefas específicas. Foi possível, por exemplo, elevar o nível detalhamento e rigor do Agente Executor sem comprometer a fluidez textual do Agente de Introdução. Essa modularidade oferece uma grande flexibilidade e adaptação do fluxo através de reconfigurações. Também é possível alocar modelos diferentes estrategicamente, mais leves ou mais robustos, conforme a demanda específica de cada fase do fluxo de trabalho.

O sistema também se destaca pelo armazenamento de dados. Na arquitetura proposta, cada instrução, resposta e demais informações foram salvas em arquivos estruturados. Isso permitiu a auditoria e análise dos resultados do sistema, garantindo que a geração dos relatórios não seja apenas o produto final, mas parte de um processo documental.

Sob essa perspectiva, essa seção analisará o desempenho do sistema, apresentando métricas como tempo, gasto de *tokens*, custo e qualidade final das análises. Os intervalos de confiança de 95% para as médias foram calculados utilizando a distribuição *t* de Student, conforme a seguinte equação:

$$IC = \bar{x} \pm t_{(\alpha/2, n-1)} \frac{s}{\sqrt{n}}$$

Nos gráficos de barras apresentados, os traços verticais (barras de erro) representam o intervalo de confiança da média para cada grupo.

5.2.1 Análise de Desempenho Computacional

O tempo médio para a confecção de um relatório completo, desde o planejamento até a conclusão, foi de aproximadamente 100 segundos, com um custo médio aproximado de R\$ 0,32 (na cotação atual) por execução. Esses indicadores, detalhados na Tabela 8, ganham relevância diante do volume de tarefas. lidando com uma média de 92.026 *tokens* de *input* e gerando uma média de 4.363 e 11.275 de *tokens* de raciocínio e *output*, respectivamente.

Tabela 8 – Resumo Geral das Métricas e Custos de Confecção dos Relatórios.

Métrica	Valor Médio/Total	IC (95%)
Tempo Médio (segundos)	100,16	[92,52; 107,79]
Média Aproximada de <i>Tokens</i> de <i>Input</i>	92026	[67738; 116314]
Média Aproximada de <i>Tokens</i> de Raciocínio	4363	[4086; 4640]
Média Aproximada de <i>Tokens</i> de <i>Output</i>	11275	[9985; 12564]
Custo Médio (R\$)	0,32	[0,27; 0,36]

Fonte: Elaborado pelo autor.

A Tabela 9 detalha as métricas das avaliações dos relatórios. Observa-se um tempo médio de execução de aproximadamente 32 segundos, valor que equivale a quase um terço do tempo de geração do relatório. O volume de *tokens* de *input* expressa o tamanho do relatório avaliado somada às instruções do Agente Avaliador. Por causa do uso do modelo Gemini 2.5 Pro, o custo médio por execução foi razoavelmente elevado quando comparado ao custo de geração de relatório.

Tabela 9 – Resumo Geral das Métricas e Custos de Avaliação dos Relatórios.

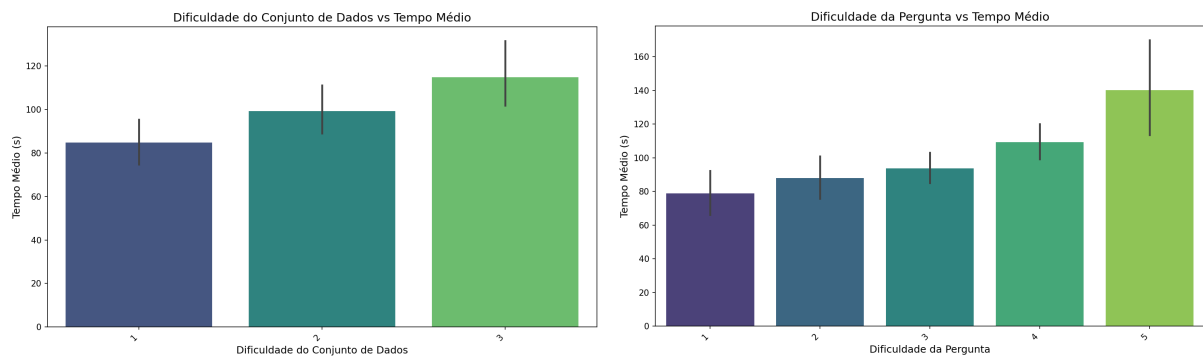
Métrica	Média	IC (95%)
Tempo Médio (segundos)	31,90	[30,90; 32,90]
Média Aproximada de <i>Tokens</i> de <i>Input</i>	6.479	[6.216; 6.741]
Média Aproximada de <i>Tokens</i> de Raciocínio	2.790	[2.665; 2.915]
Média Aproximada de <i>Tokens</i> de <i>Output</i>	955	[927; 983]
Custo Médio (R\$)	0,10	[0,09; 0,10]

Fonte: Elaborada pelo autor.

Os gráficos indicados na Figura 8 revelam que o tempo de processamento possui uma correlação positiva com a dificuldade da tarefa e do conjunto de dados (painel esquerdo). O tempo médio para realização do relatório sobe em aproximadamente 35% de

dados nível 1 para dados nível 3. Esse comportamento é ainda mais acentuado em relação à dificuldade da pergunta (painel direito), que no nível 1 são resolvidas em média em 80 segundos, enquanto no nível 5, demoram cerca de 140 segundos. Essa diferença no tempo de execução se deve, em grande parte, à maior complexidade das tarefas, que demanda a decomposição do problema em um número maior de Etapas no Plano Inicial. Observou-se uma variação entre 1 e 6 etapas, resultando em uma média de 3,57 Etapas por relatório.

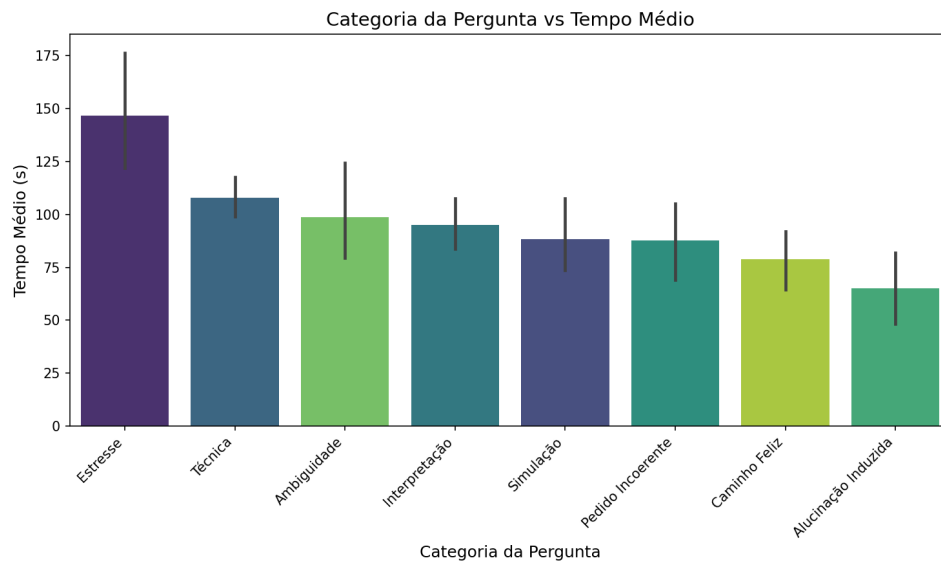
Figura 8 – Gráficos do Tempo Médio de Execução de Relatórios por Dificuldade do Conjunto de Dados e Pergunta.



Fonte: Elaborado pelo autor.

Ao segmentar por categoria de pergunta na Figura 9, percebemos que a categoria de estresse, criada para testar os limites do sistema, consome o maior tempo de execução e possui a maior variabilidade, seguida por perguntas de técnica e ambiguidade. Em contrapartida, cenários de alucinação induzida apresentaram os menores tempos, indicando que o sistema evita gastar tempo em relatórios infrutíferos. As induzindo alucinações foram propositalmente para testar a resposta do sistema, mostrando que uma camada adicional de salvaguarda é perfeitamente capaz de impedir tais falhas antes mesmo da tentativa de execução. Além disso, os relatórios solucionados em apenas uma etapa referem-se às categorias de caminho feliz ou alucinação induzida.

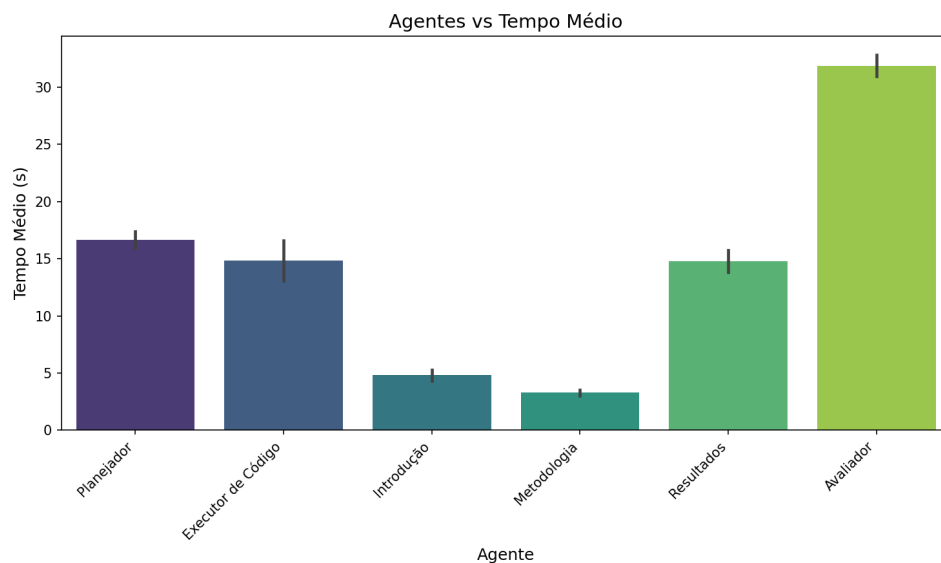
Figura 9 – Gráfico do Tempo Médio de Execução de Relatórios por Categoria de Pergunta.



Fonte: Elaborado pelo autor.

A análise do tempo de execução por agente, apresentada na Figura 10, mostra que os Agentes de Introdução e Metodologia apresentam o menor tempo de resposta do sistema, enquanto o Agente Avaliador se destaca pelo maior consumo de tempo médio. É importante ressaltar que, para o Agente Executor, os valores apresentados referem-se à média por etapa e não ao tempo total acumulado. Tal abordagem foi adotada para evitar distorções na escala gráfica, visto que o fluxo de execução variou até 6 etapas por relatório. Este critério de visualização por etapa será mantido nas próximas análises comparativas.

Figura 10 – Gráfico do Tempo Médio de Execução de Relatórios por Agente.

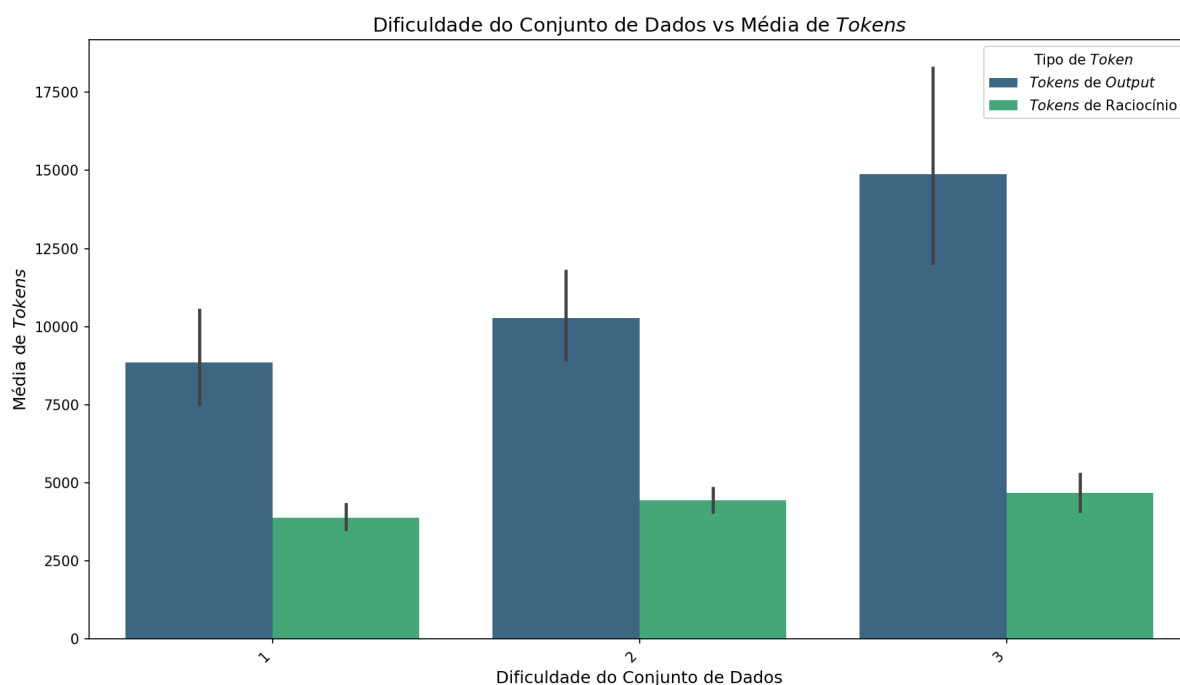


Fonte: Elaborado pelo autor.

Para a avaliação do consumo de *tokens*, a análise se restringiu a aos *tokens* de raciocínio e *tokens* de *output*. A métrica de *tokens* de *input* foi desconsiderada, uma vez que sua variação é determinada pela estrutura fixa dos *prompts* do sistema e pelo tamanho do arquivo enviado. Ou seja, os *tokens* de *input* não refletem necessariamente o comportamento adaptativo ou o “esforço” do modelo.

A análise do consumo de *tokens* em função da dificuldade dos dados, apresentada na Figura 11, revela uma correlação evidente, porém com taxas de crescimento distintas entre as métricas de *output* e raciocínio. O volume de *tokens* de *output* aumenta consideravelmente, saltando de aproximadamente 8.500 no nível 1 para cerca de 15.000 no nível 3. Isso acontece porque dados mais complexos exigem códigos mais elaborados e acionam com maior frequência o processo de autocorreção. Diante de falhas no carregamento ou processamento (mais comuns em dados mais difíceis), o modelo é forçado a refazer etapas anteriores e gerar novos trechos de código para resolver o erro, o que aumenta o volume total de texto gerado. Apesar disso, os *tokens* de raciocínio apresentam uma progressão mais estável, (de ≈ 3.900 para ≈ 4.700). Isso sugere que, embora dados difíceis exijam mais trabalho braçal de escrita (código/texto), o “custo cognitivo” para o planejamento da análise não cresce na mesma proporção.

Figura 11 – Gráfico da Média de *Tokens* (*Output* e Raciocínio) por Dificuldade do Conjunto de Dados.

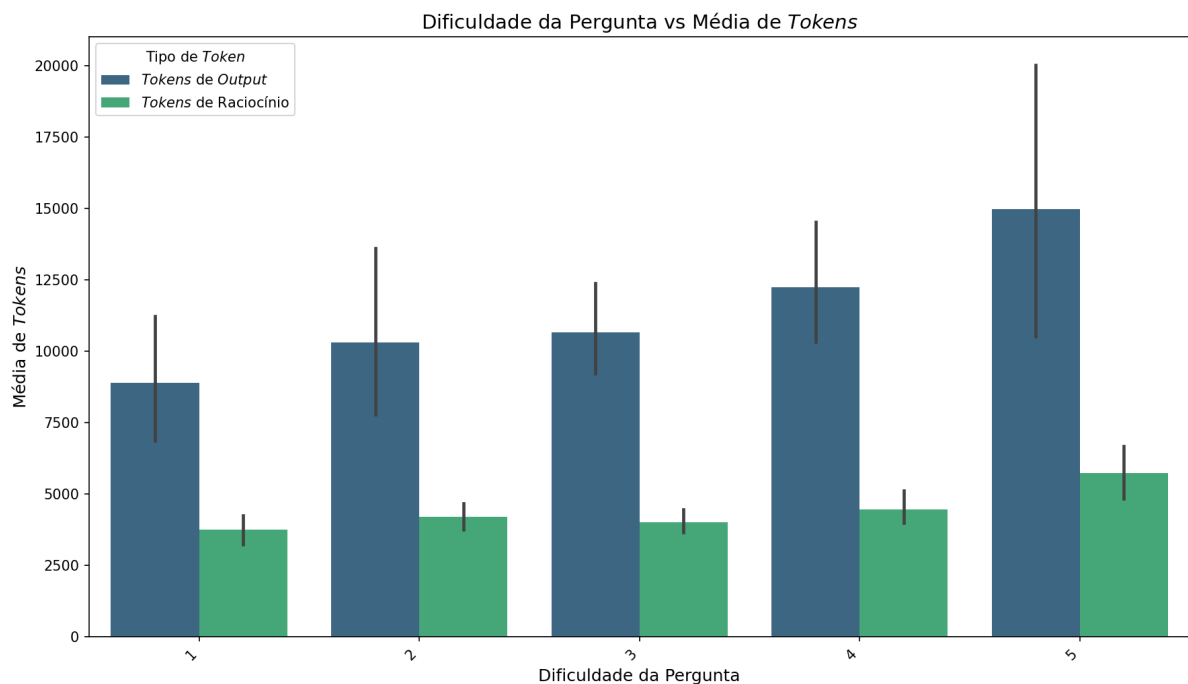


Fonte: Elaborado pelo autor.

Ao analisar a distribuição de *tokens* agrupada pela dificuldade da pergunta (Fi-

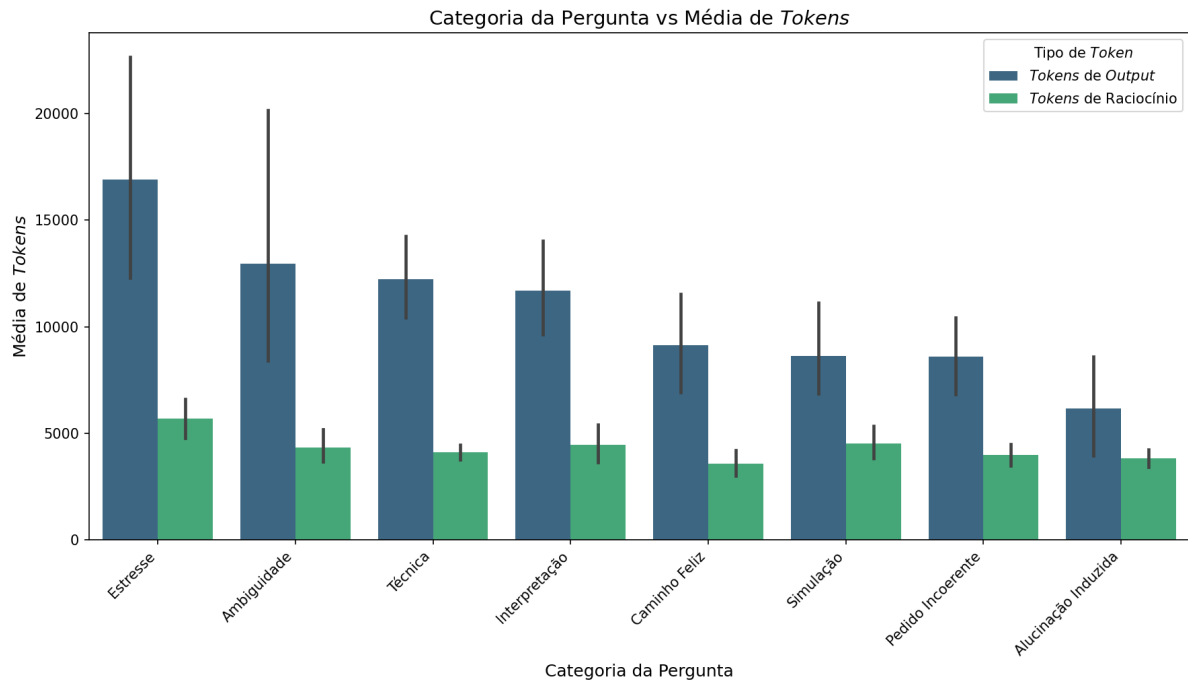
gura 12), observa-se um aumento entre a demanda de produção textual e a demanda cognitiva. Os *tokens* de *output* apresentam um crescimento consistente, enquanto os *tokens* de raciocínio demonstram uma certa estabilidade nos níveis iniciais e intermediários, com aumento considerável no nível 5. Isso indica que independentemente do nível exato da complexidade, perguntas mais difíceis exigem respostas e códigos mais extensos, enquanto o sistema exige um esforço de raciocínio significativamente maior apenas quando confrontado com tarefas mais complexas, conseguindo resolver problemas de complexidade fácil e média com um “custo cognitivo” semelhante.

Figura 12 – Gráfico da Média de *Tokens* (*Output* e Raciocínio) por Dificuldade de Pergunta.



Fonte: Elaborado pelo autor.

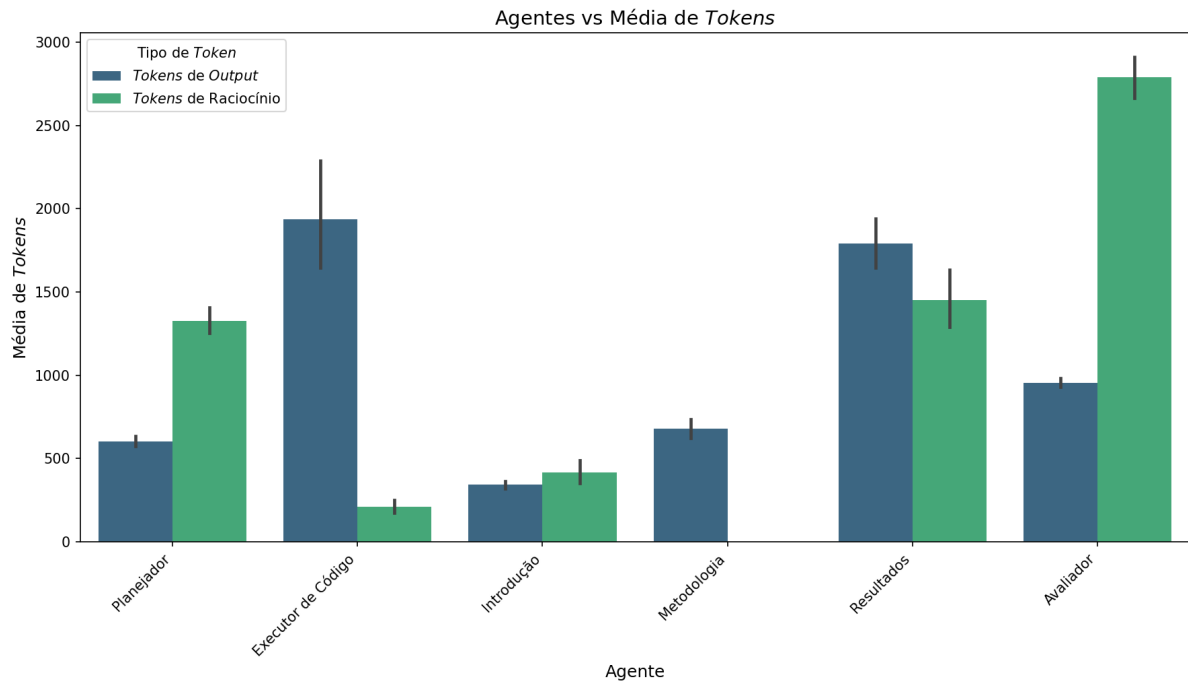
A análise das categorias de pergunta, apresentada na Figura 13, corrobora a adaptabilidade do sistema frente a diferentes naturezas de problemas. A categoria estresse e ambiguidade demandaram o maior consumo de recursos em ambas as métricas, além de apresentarem a maior variabilidade entre os grupos. A categoria alucinação induzida, de forma semelhante à Figura 9, registrou o menor consumo de *tokens* de *output*, indicando eficiência e economia ao identificar solicitações factualmente impossíveis.

Figura 13 – Gráfico da Média de *Tokens* (*Output* e Raciocínio) por Categoria de Pergunta.

Fonte: Elaborado pelo autor.

O perfil de consumo de recursos por agente, ilustrado na Figura 14, corrobora a especialização funcional da arquitetura proposta. O Agente Planejador destaca-se por uma elevada taxa de raciocínio em contraste com um baixo volume de saída, confirmando seu papel estratégico de “arquiteto”, desenhando a solução sem gerar texto final extenso. Em contrapartida, o Agente Executor apresenta o maior volume de *output* associado ao menor consumo de raciocínio, sugerindo que a carga cognitiva foi absorvida na etapa anterior de planejamento, restando a ele tarefas mais diretas e operacionais. O Agente Executor também apresentou a maior variabilidade de *tokens* de *output*.

O Agente Avaliador registrou o máximo de *tokens* de raciocínio de todo o sistema. Esse comportamento evidencia a alta “carga cognitiva” exigida para a tarefa de avaliação, ainda que seu *output* permaneça em patamares intermediários. O Agente de Resultados também se mostrou um componente oneroso, só que balanceando *tokens* de raciocínio para a interpretação dos resultados e de *output* para a escrita do do texto. Por fim, os agentes de Introdução e Metodologia apresentam o menor gasto do sistema, refletindo a natureza mais descritiva de suas funções. O parâmetro de raciocínio do Agente de Metodologia não foi ativado, dado que o custo de raciocínio não se traduziu em melhores descrições metodológicas.

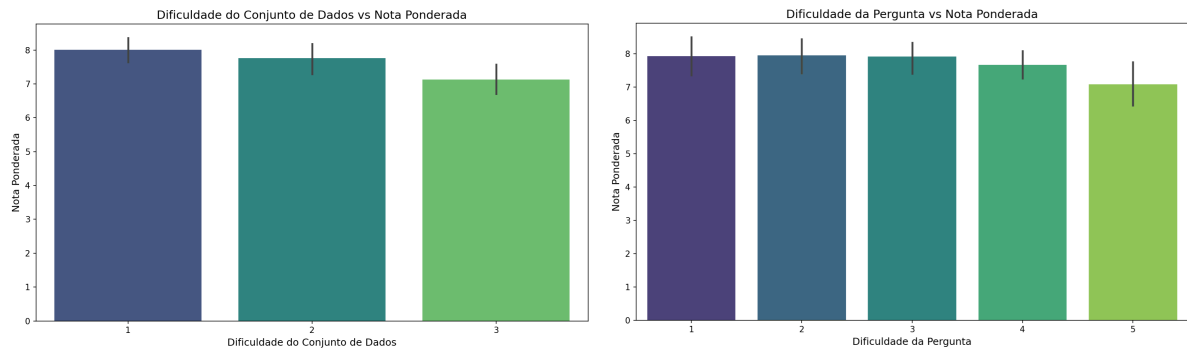
Figura 14 – Gráfico da Média de *Tokens* (*Output* e *Raciocínio*) por Agente.

Fonte: Elaborado pelo autor.

5.2.2 Análise da Qualidade dos Relatórios

A análise do desempenho, mensurado pela nota ponderada, revela a qualidade dos relatórios gerados pelo sistema frente ao aumento da complexidade. Observando a relação com a dificuldade do conjunto de dados (painel esquerdo da Figura 15), nota-se uma leve tendência decrescente, onde a pontuação média recua de ≈ 8 no nível 1 para $\approx 7,1$ no nível 3. De forma análoga, a análise por dificuldade da pergunta (painel direito) demonstra uma estabilidade entre os níveis 1 e 3, preservando notas médias próximas a 8. A diminuição notável ocorre nos níveis 4 e 5. Esses resultados indicam que, embora haja uma degradação de qualidade conforme a complexidade dos dados e perguntas aumentam, o sistema sustenta um desempenho satisfatório, mantendo média acima do limiar de ≈ 7 .

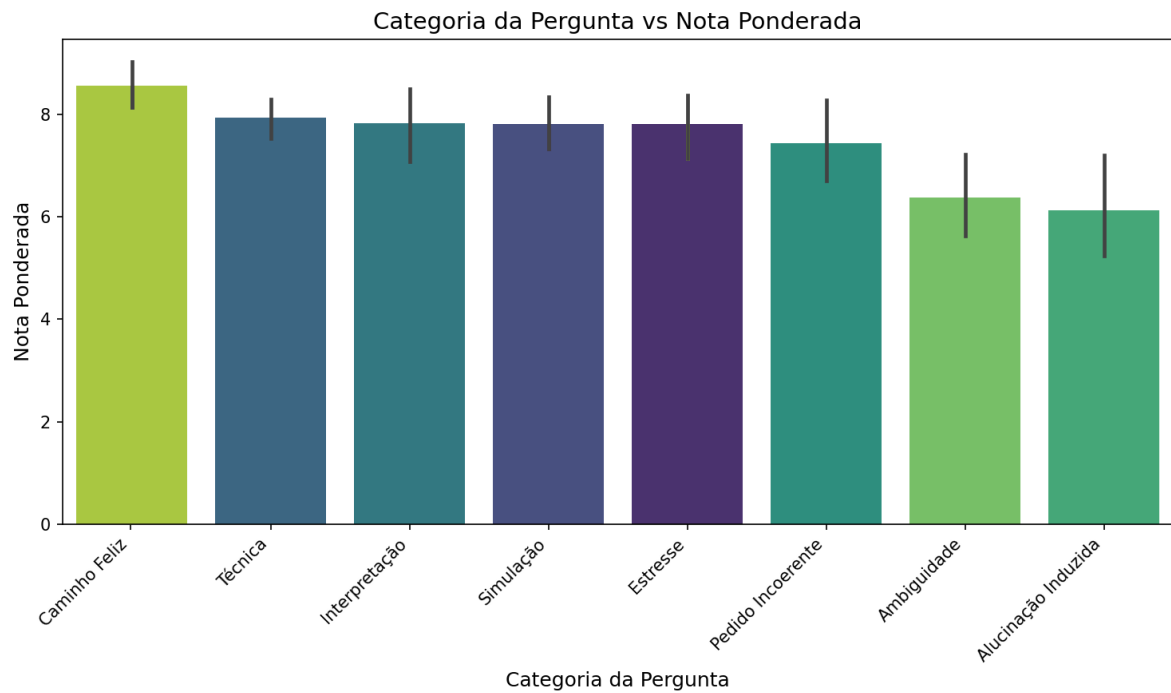
Figura 15 – Gráficos das Notas Ponderadas por Dificuldade de Conjunto de Dados e Pergunta.



Fonte: Elaborado pelo autor.

A análise das notas ponderadas por categorias de pergunta, apresentada na Figura 16 mostra que o cenário de caminho feliz estabelece o máximo de desempenho, superando os 8,5 pontos, o que confirma a competência do sistema em condições ideais de clareza. Existe uma estabilidade de notas nas categorias de técnica, interpretação, simulação e até mesmo estresse. Isso demonstra que a sobrecarga de processamento (observada na análise de tempo e *tokens*) não compromete a qualidade final da entrega. As categorias ambiguidade e alucinação induzida registram as menores pontuações. Isso é esperado pela própria natureza das perguntas, que desafiam o modelo com ausência de diretrizes claras ou solicitações de dados inexistentes. O Agente Avaliador faz com que o relatório do sistema, mesmo que apresentando uma recusa correta ou uma tentativa de desambiguação, seja avaliada como menos “rica” em conteúdo do que um relatório completo.

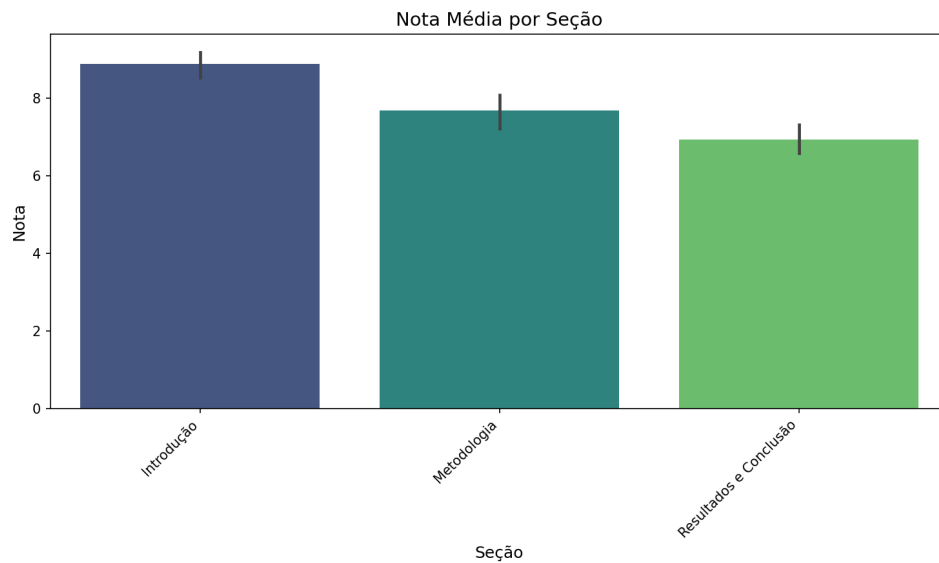
Figura 16 – Gráfico das Notas Ponderadas por Categoria de Pergunta.



Fonte: Elaborado pelo autor.

A avaliação das notas ponderadas por seções do relatório (Figura 17) revela uma correlação inversa entre a pontuação obtida e a complexidade da dimensão. A seção de introdução alcançou o patamar de ≈ 9 , seguida pela metodologia com $\approx 7,8$. Em contrapartida, a seção de resultados e conclusão, fruto do esforço conjunto do Agente de Resultados e Conclusão e do Agente Executor, apresentou média de ≈ 7 , refletindo não só o desafio de interpretar dados e resolver problemas, mas também por causa das limitações discutidas na Seção 4.8. A restrição de 30 segundos para a execução de código impõe um desafio adicional ao Agente Executor, que fica impossibilitado de realizar tarefas como simulações mais longas ou otimização de hiperparâmetros via busca exaustiva (*Grid Search*). Além disso, o ambiente *sandboxed* impõe restrições quanto às bibliotecas disponíveis, o que impediu dificultou a aplicação de técnicas que dependem de bibliotecas específicas, notadamente em NLP.

Figura 17 – Gráfico das Notas Ponderadas por Categoria de Pergunta.



Fonte: Elaborado pelo autor.

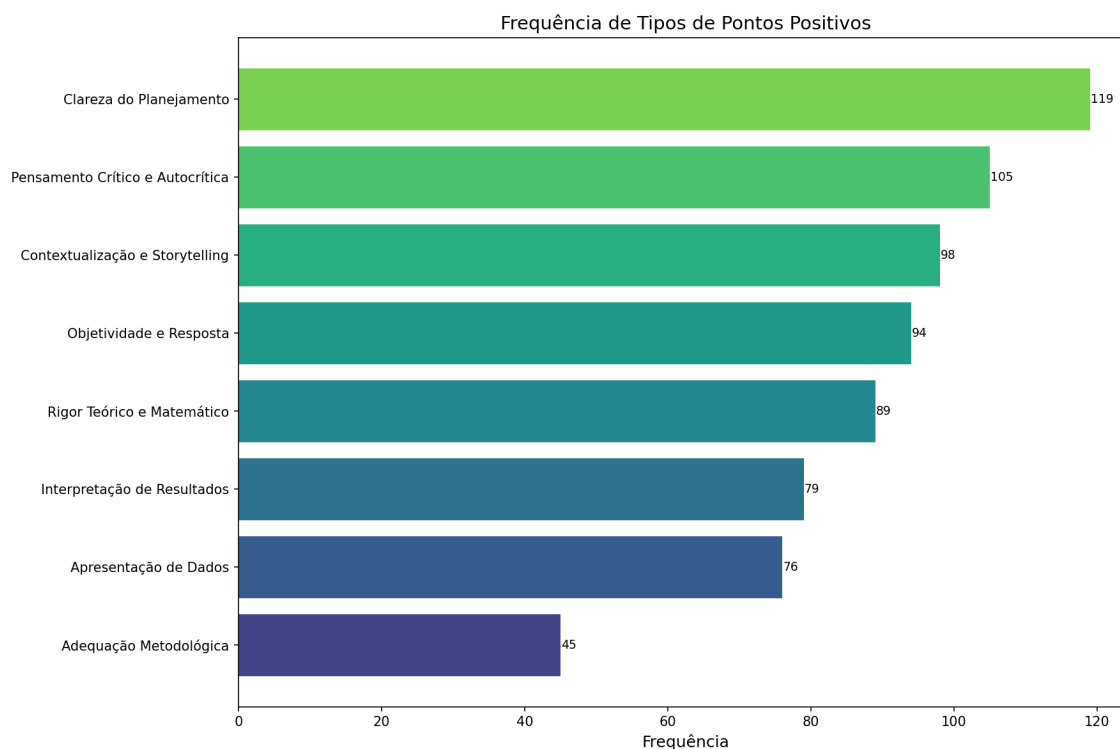
A análise dos pontos positivos, ilustrada na Figura 18, mostra que a 'clareza do planejamento', 'pensamento crítico e autocrítica' e 'contextualização e storytelling' são os pontos fortes mais recorrentes, sugerindo que a separação de responsabilidades e a estruturação do sistema resultou em documentos bem estruturados e coerentes.

Observa-se que os quesitos de 'adequação metodológica' e 'rigor teórico e matemático' tem menor incidência entre os pontos positivos, indicando que a rigorosa fundamentação acadêmica não é o principal diferencial do sistema. Este resultado também é coerente com decisões adotadas no projeto, especificamente a utilização do modelo Gemini 2.5 Flash-Lite para o Agente de Metodologia.

A distribuição dos tipos de erro, apresentada na Figura 19, revela um perfil de falha onde erros como 'alucinação e factualidade' e 'visualização de dados ruim' apresentaram as menores frequências. As observações de erros como 'erro metodológico' e 'falta de rigor teórico' reforçam os achados dos pontos positivos das avaliações, apresentados na Figura 18.

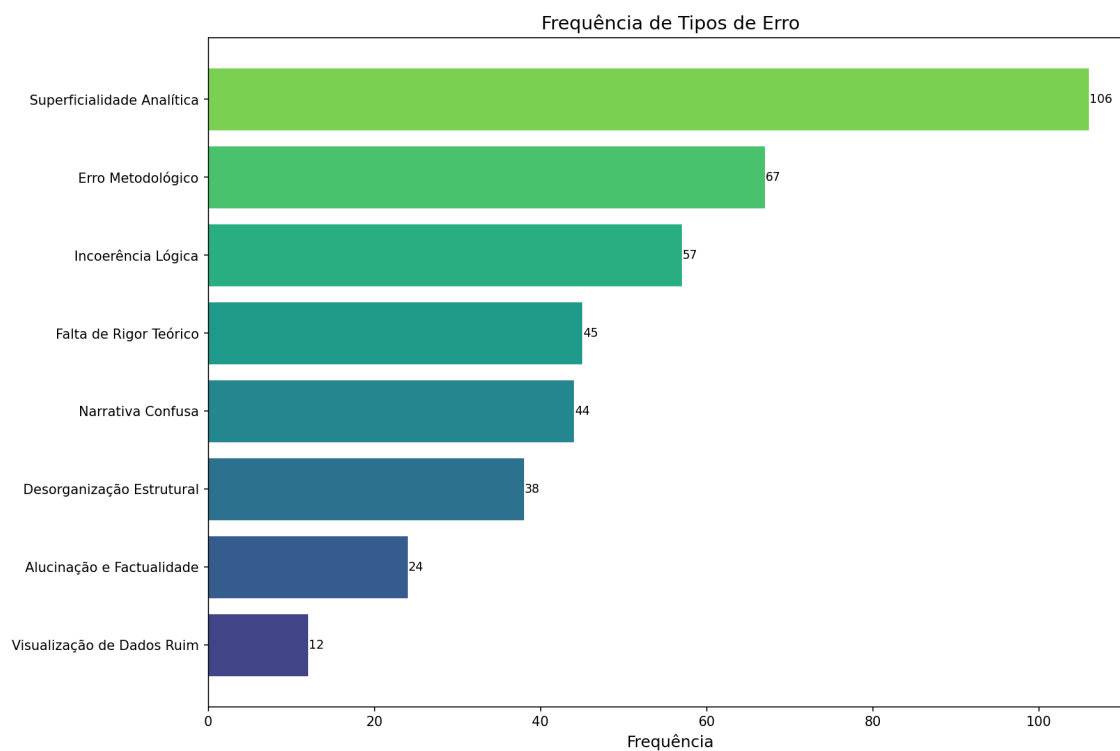
Por outro lado, a predominância de 'superficialidade analítica' aponta para uma limitação de interpretações e análises, sugerindo que, para análises mais complexas, o sistema atua melhor como um assistente de produtividade.

Figura 18 – Gráfico das Notas Ponderadas por Categoria de Pergunta.



Fonte: Elaborado pelo autor.

Figura 19 – Gráfico das Notas Ponderadas por Categoria de Pergunta.



Fonte: Elaborado pelo autor.

5.3 Discussão

A análise das métricas confirma que o sistema opera dentro do aceitável em termos de erro, tempo e custo. No entanto, a verdadeira validação da proposta reside na interpretação qualitativa desses dados frente à realidade do trabalho de um estatístico humano.

O diferencial mais observado nos experimentos é a discrepância de eficiência em tempo. O sistema manteve um tempo médio de execução de aproximadamente 100 segundos por relatório completo. Para contextualizar este valor, deve-se considerar o fluxo de trabalho manual equivalente de um estatístico em: carregar e limpar os dados, escrever o código, depurar erros de sintaxe, interpretar os resultados e fazer a redação final com formatação adequada do relatório. Esse ciclo manual consumirá consideravelmente mais tempo do que o apresentado pelos relatórios automáticos gerados pelo sistema.

Além do tempo, a relação custo-benefício reforça a viabilidade da automação. Com um custo médio de US\$ 0,06 (aproximadamente R\$ 0,32 na cotação atual), por execução, o sistema oferece uma análise preliminar completa por uma fração do valor-hora de um profissional especializado. Entretanto, isso não implica a obsolescência do analista humano, mas sugere uma mudança de paradigma: o custo de um profissional humano “gastando” tempo em análises rotineiras com menor agregado torna-se economicamente injustificável quando uma arquitetura multiagente pode entregar um relatório inicial em menos de dois minutos.

Um achado qualitativo relevante foi a capacidade do sistema de passar por cima da execução literal de comandos, demonstrando um grau de “inteligência” e conhecimento de boas práticas. Por exemplo, durante testes de ambiguidade, pedidos incoerentes e alucinações induzidas, observou-se que o Agente Planejador atuou como um filtro. Um exemplo disso ocorreu quando a instrução solicitou explicitamente um “gráfico de pizza para visualizar a evolução temporal” de uma variável. O sistema, identificando a inadequação desse tipo de gráfico para séries temporais, ignorou a instrução literal e optou autonomamente pela geração de um gráfico de linhas, justificando a escolha no texto do relatório. Esse comportamento adaptativo evidencia que a arquitetura não funciona apenas como um tradutor de texto para código porque possui capacidade de corrigir premissas equivocadas do usuário para garantir a integridade das análises.

Uma limitação é a existência de uma correlação direta entre o tamanho do conjunto de dados e o custo de processamento do Agente Executor. Observou-se que conjuntos de dados com grande dimensionalidade aumentam significativamente a quantidade de *tokens* consumidos, o que também aumenta o custo final do relatório. Isso mostra que, em implementações futuras voltadas para *Big Data*, outras estratégias serão necessárias para manter a eficiência de custo observada nestes experimentos.

A utilidade do sistema é evidente, embora com limitações previamente discutidas. A solução sobressai em cenários que demandam alta escalabilidade e processamento massivo de relatórios, contextos nos quais a alocação exclusiva de esforço humano seria ineficiente quando comparada à automação via LLMs. Aplicações como a triagem rápida de dezenas de bases de dados desconhecidas e a validação quase instantânea de hipóteses sem a necessidade de mobilização dedicada de profissionais ilustram os casos de uso onde a ferramenta se potencializa.

Entretanto, é importante reconhecer que, mesmo apesar dos avanços, os LLMs permanecem suscetíveis a alucinações sutis e vieses interpretativos, falhas que podem passar despercebidas em verificações rápidas. Sendo assim, a ferramenta posiciona-se não como um substituto autônomo, mas como um multiplicador de força. Ela amplia o acesso à análise de dados, permitindo que usuários “dialoguem” com seus dados de forma direta, liberando profissionais para tarefas de maior complexidade e valor agregado.

6 EXEMPLO DE RELATÓRIO GERADO

Neste capítulo, será apresentado um estudo de caso prático. O cenário escolhido utiliza o clássico conjunto de dados do Titanic, que compila informações demográficas e logísticas dos passageiros, como classe socioeconômica, sexo, idade, tarifa paga e porto de embarque, e uma variável que indica a sobrevivência do passageiro. A pergunta foi classificada na categoria de interpretação, com a dificuldade da pergunta no nível 3 e a dificuldade do conjunto de dados no nível 2. A instrução exata submetida ao sistema foi: “Usando apenas análise de dados, crie uma heurística para prever os sobreviventes”.

O processamento totalizou 182.816 *tokens* em 1 minuto e 44 segundos, divididos entre 169.847 de *input* (devido ao tamanho do conjunto de dados), 4.916 de raciocínio e 12.969 de *output*. O custo final da operação foi de aproximadamente R\$ 0,43 na cotação atual. Em relação à avaliação, a nota final ponderada foi de 8,38, com nota de 9 na introdução, 9,3 em Materiais e Métodos, e 7,2 em Resultados e Conclusão.

O relatório exibido é o produto direto da geração do sistema, tendo passado apenas por adaptações funcionais para viabilizar sua renderização dentro da estrutura deste documento L^AT_EX. Foram realizadas intervenções mínimas, limitadas à substituição de ponto por vírgula decimal e correções de sintaxe de código para evitar conflitos de compilação. Nenhuma alteração de conteúdo textual ou lógica foi feita, preservando a originalidade do relatório gerado.

Análise de Fatores de Sobrevivência para Criação de Heurística Preditiva

Sistema Automatizado

Sumário do Relatório

6.1	Introdução	57
6.2	Metodologia	58
6.2.1	<i>Materiais</i>	58
6.2.2	<i>Métodos</i>	59
6.2.2.1	Análise de Sobrevivência por Classe e Sexo	59
6.2.2.2	Análise Combinada de Classe e Sexo	60
6.2.2.3	Análise de Sobrevivência por Idade	60
6.2.2.4	Formulação da Heurística Preditiva	61
6.3	Resultados e Conclusão	61
6.3.1	<i>Análise de Sobrevivência por Classe e Sexo</i>	61
6.3.2	<i>Análise Combinada de Classe e Sexo</i>	62
6.3.3	<i>Análise de Sobrevivência por Idade</i>	64
6.3.4	<i>Formulação da Heurística Preditiva</i>	64
6.4	Conclusão	65
6.5	Referências	66
6.A	Códigos Gerados	68
6.A.1	<i>Análise de Sobrevivência por Classe e Sexo</i>	68
6.A.2	<i>Análise Combinada de Classe e Sexo</i>	69
6.A.3	<i>Análise de Sobrevivência por Idade</i>	70
6.A.4	<i>Formulação da Heurística Preditiva</i>	71

6.1 Introdução

A tragédia do RMS Titanic, ocorrida na madrugada de 15 de abril de 1912, permanece como um dos mais infames desastres marítimos da história. O luxuoso transatlântico, aclamado como “inafundável”, sucumbiu após colidir com um iceberg no Atlântico Norte, levando à perda de aproximadamente 1.517 vidas das 2.223 pessoas a bordo, entre passageiros e tripulantes. Das cerca de 2.223 pessoas a bordo, apenas 706 sobreviveram, o que representa uma taxa de sobrevivência de meros 38% [1], [2], [3]. Este evento catastrófico revelou não apenas as limitações da tecnologia da época, mas também complexas dinâmicas sociais que influenciaram drasticamente as chances de sobrevivência [2], [4].

Apesar da escala avassaladora do desastre, a análise detalhada dos dados históricos revela que a sobrevivência não foi um evento puramente aleatório. Fatores como classe social, sexo e idade desempenharam papéis cruciais na determinação do destino de muitos. Por exemplo, enquanto aproximadamente 75% das mulheres sobreviveram, apenas cerca de 19% dos homens a bordo conseguiram se salvar. Similarmente, passageiros da primeira classe apresentaram taxas de sobrevivência superiores em comparação com aqueles da segunda e terceira classes [3], [5]. A compreensão desses padrões latentes é onde a análise de dados se torna uma ferramenta inestimável [3], [4].

Nesse contexto, a análise de dados históricos de desastres é essencial para compreender os fatores que contribuíram para eventos passados, permitindo a identificação de padrões e a formulação de protocolos eficazes de resposta e prevenção. Ao aplicar técnicas de análise sobre conjuntos de dados como o `yasserh_titanic-dataset`, podemos ir além da simples narrativa histórica e desenvolver uma heurística capaz de prever os sobreviventes com base nas informações disponíveis [6], [7], [8]. Esta abordagem não apenas lança luz sobre as circunstâncias da tragédia do Titanic, mas também exemplifica o poder da ciência de dados em extrair insights valiosos de eventos complexos, transformando dados brutos em conhecimento acionável.

Para responder a essa questão, adotou-se a seguinte estratégia de análise:

- **Análise de Sobrevivência por Classe e Sexo:** Investigar as taxas de sobrevivência individuais baseadas na classe do passageiro (*Pclass*) e no sexo (*Sex*), visualizando os resultados através de gráficos de barras.
- **Análise Combinada de Classe e Sexo:** Examinar a interação entre *Pclass* e *Sex* para entender como essas duas variáveis influenciam conjuntamente a sobrevivência, utilizando um heatmap ou gráfico de barras agrupado.
- **Análise de Sobrevivência por Idade:** Categorizar a idade em faixas etárias e calcular a taxa de sobrevivência para cada grupo, excluindo dados faltantes e apresentando os achados em um gráfico de barras.
- **Formulação da Heurística Preditiva:** Com base nas análises anteriores, identificar o grupo com maior probabilidade de sobrevivência, imprimir sua taxa de sobrevivência e propor uma regra de decisão simples e direta.

6.2 Metodologia

6.2.1 Materiais

O conjunto de dados do Titanic é composto por 891 observações e 12 variáveis. A análise preliminar revela a presença de valores ausentes em três variáveis: 'Age', 'Ca-

bin' e 'Embarked'. A variável 'Age' possui 177 valores ausentes (19,9% do total), 'Cabin' apresenta 687 valores ausentes (77,1% do total) e 'Embarked' contém 2 valores ausentes (0,2% do total). Não foram identificadas linhas duplicadas no conjunto de dados.

Tabela 10 – Sistema Automatizado - Resumo das Variáveis do Conjunto de Dados do Titanic.

Variável	Tipo	Descrição
PassengerId	Numérico	Identificador único do passageiro
Survived	Categórico	Indica se o passageiro sobreviveu (0 = Não, 1 = Sim)
Pclass	Categórico	Classe do bilhete (1, 2 ou 3)
Name	Categórico	Nome do passageiro
Sex	Categórico	Sexo do passageiro (male/female)
Age	Numérico	Idade do passageiro em anos
SibSp	Numérico	Número de irmãos/cônjuges a bordo
Parch	Numérico	Número de pais/filhos a bordo
Ticket	Categórico	Número do bilhete
Fare	Numérico	Tarifa paga pelo passageiro
Cabin	Categórico	Número da cabine
Embarked	Categórico	Porto de embarque (C = Cherbourg, Q = Queens-town, S = Southampton)

Fonte: Elaborado pelo sistema automatizado.

6.2.2 Métodos

6.2.2.1 Análise de Sobrevivência por Classe e Sexo

Para entender os fatores que influenciaram a sobrevivência dos passageiros, realizamos uma análise detalhada das taxas de sobrevivência em relação à classe do passageiro ('Pclass') e ao sexo ('Sex').

Taxa de Sobrevivência por Categoria: A taxa de sobrevivência em uma determinada categoria é calculada como a proporção de passageiros que sobreviveram dentro dessa categoria. Matematicamente, a taxa de sobrevivência (S) para uma categoria específica (C) é dada por:

$$S_C = \frac{\text{Número de sobreviventes na categoria } C}{\text{Número total de passageiros na categoria } C}$$

Esta métrica nos permite quantificar o quão provável era a sobrevivência para passageiros pertencentes a diferentes classes e sexos.

Gráficos de Barras: Para visualizar essas taxas, utilizamos gráficos de barras. Um gráfico de barras é ideal para comparar frequências ou proporções entre diferentes

categorias discretas. O eixo x representa as categorias (ex: ‘Pclass’ 1, 2, 3 ou ‘Sex’ feminino, masculino), e o eixo y representa a taxa de sobrevivência correspondente. A altura de cada barra indica diretamente a taxa de sobrevivência, facilitando a identificação de quais grupos tiveram maior ou menor probabilidade de sobreviver.

6.2.2.2 Análise Combinada de Classe e Sexo

A seguir, exploramos a interação entre a classe do passageiro e o sexo na determinação da sobrevivência.

Taxa de Sobrevivência Combinada: Calculamos a taxa de sobrevivência para cada combinação única de ‘Pclass’ e ‘Sex’ (ex: Mulheres na Classe 1, Homens na Classe 3). A fórmula é análoga à taxa de sobrevivência por categoria, mas aplicada a grupos mais específicos:

$$S_{P \times S} = \frac{\text{Número de sobreviventes na combinação de } Pclass \text{ e } Sex}{\text{Número total de passageiros na combinação de } Pclass \text{ e } Sex}$$

Heatmap ou Gráfico de Barras Agrupado: Para visualizar essas taxas combinadas, um heatmap ou um gráfico de barras agrupado são métodos eficientes.

- **Heatmap:** Um heatmap utiliza cores para representar os valores em uma matriz. Neste caso, as linhas poderiam representar ‘Pclass’, as colunas ‘Sex’, e a cor de cada célula indicaria a taxa de sobrevivência correspondente. Gradientes de cores mais intensas podem sinalizar taxas de sobrevivência mais altas.
- **Gráfico de Barras Agrupado:** Similar a um gráfico de barras simples, mas para cada categoria principal (ex: ‘Pclass’), barras separadas são usadas para subcategorias (ex: sexo feminino e masculino). Isso permite uma comparação direta dentro de cada classe e entre as classes.

Ambos os métodos facilitam a identificação de padrões complexos, como a influência predominante do sexo sobre a classe, ou vice-versa, nas taxas de sobrevivência.

6.2.2.3 Análise de Sobrevivência por Idade

A idade é outro fator demográfico crucial que pode ter influenciado a sobrevivência.

Discretização da Idade: A variável contínua ‘Age’ foi discretizada em faixas etárias predefinidas para facilitar a análise e a visualização. Exemplos de faixas incluem 0-12 anos (crianças), 13-18 anos (adolescentes), 19-60 anos (adultos) e 60+ anos (idosos).

Taxa de Sobrevivência por Faixa Etária: A taxa de sobrevivência para cada faixa etária foi calculada desconsiderando os registros com idade nula, para garantir a

precisão da análise. A fórmula é:

$$S_{AgeBand} = \frac{\text{Número de sobreviventes na faixa etária } AgeBand}{\text{Número total de passageiros com idade na faixa } AgeBand}$$

Gráfico de Barras por Faixa Etária: Um gráfico de barras foi empregado para exibir a taxa de sobrevivência em cada faixa etária. Semelhante à análise por ‘Pclass’ e ‘Sex’, o eixo x mostra as faixas etárias e o eixo y a taxa de sobrevivência. Isso permite observar se certos grupos etários tiveram uma probabilidade significativamente maior ou menor de sobreviver.

6.2.2.4 Formulação da Heurística Preditiva

Com base nas análises anteriores, especialmente na combinada de classe e sexo, formulamos uma heurística para prever a sobrevivência.

Identificação do Grupo com Maior Sobrevivência: Analisamos as taxas de sobrevivência calculadas para todas as combinações de ‘Pclass’ e ‘Sex’ para identificar o grupo com a maior probabilidade de sobrevivência. A taxa de sobrevivência deste grupo específico é então registrada.

Regra de Decisão Explícita: A heurística preditiva é expressa como uma regra de decisão simples e clara, baseada nas descobertas. Por exemplo: **Heurística Proposta:** Se um passageiro é do sexo feminino **E** pertence à Pclass 1, preveja que ele sobreviveu. Esta regra visa capturar o padrão mais forte observado nos dados, onde passageiras da primeira classe demonstraram as maiores taxas de sobrevivência.

6.3 Resultados e Conclusão

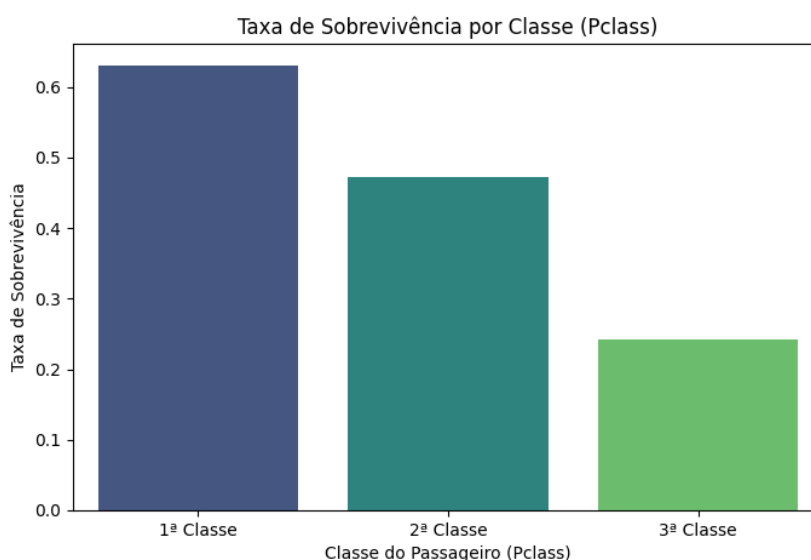
6.3.1 Análise de Sobrevivência por Classe e Sexo

A análise inicial das taxas de sobrevivência revelou diferenças significativas, direcionando as bases para a heurística preditiva. Observou-se que a **Classe do Passageiro (Pclass)** exerceu uma influência considerável na probabilidade de sobrevivência. Aproximadamente 63,0% dos passageiros da Primeira Classe sobreviveram, em contraste com 47,3% na Segunda Classe e apenas cerca de 24,2% na Terceira Classe. Esta tendência demonstra que, quanto maior a classe do passageiro, maior foi a sua taxa de sobrevivência, sugerindo uma prioridade na evacuação ou acesso a melhores recursos.

Em relação ao **Sexo**, a disparidade foi ainda mais marcante. Cerca de 74,2% das mulheres sobreviveram, enquanto a taxa de sobrevivência para os homens foi de apenas 18,9%. Esta diferença substancial indica que o sexo feminino teve uma vantagem considerável na sobrevivência. Os gráficos de barras correspondentes visualizaram claramente

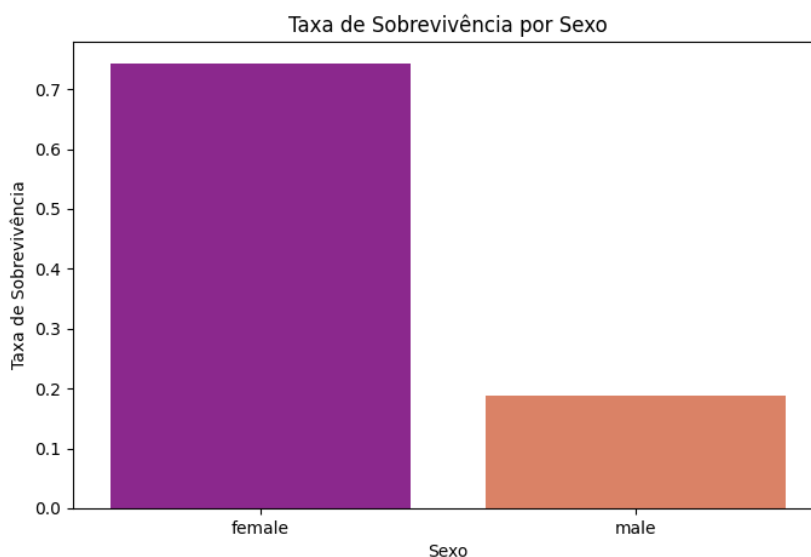
essas disparidades, com a taxa de sobrevivência diminuindo progressivamente da Primeira para a Terceira Classe e sendo drasticamente maior para mulheres do que para homens.

Figura 20 – Sistema Automatizado - Gráfico da Taxa de Sobrevivência por Classe (Pclass).



Fonte: Elaborado pelo sistema automatizado.

Figura 21 – Sistema Automatizado - Gráfico da Taxa de Sobrevivência por Sexo.



Fonte: Elaborado pelo sistema automatizado.

6.3.2 Análise Combinada de Classe e Sexo

Aprofundando a investigação, a análise combinada da classe do passageiro e do sexo reforçou as observações anteriores e revelou interações cruciais para a formulação da

heurística. As taxas de sobrevivência por combinação de *Pclass* e *Sex* demonstraram o seguinte:

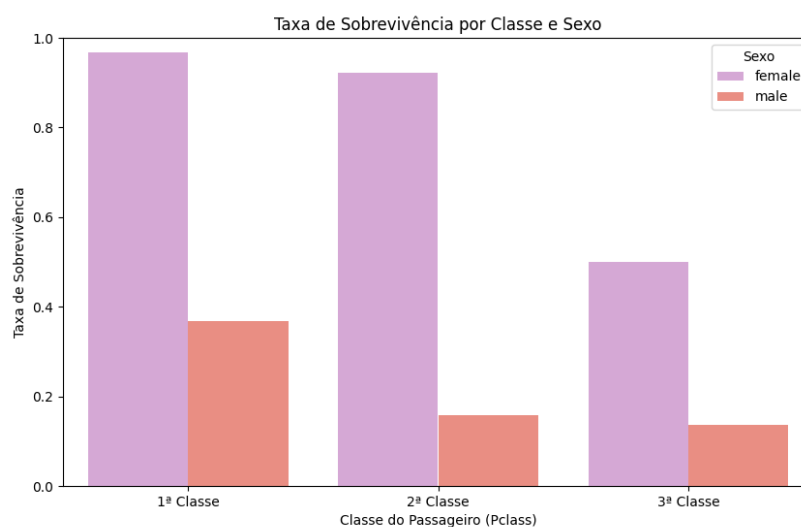
- **Mulheres na 1ª Classe:** Uma taxa de sobrevivência impressionante de 96,8%. Quase a totalidade das mulheres que viajavam na primeira classe sobreviveram.
- **Mulheres na 2ª Classe:** 92,1% de sobrevivência.
- **Mulheres na 3ª Classe:** 50,0% de sobrevivência. Embora menor que nas classes superiores, esta taxa ainda é significativamente mais alta do que a dos homens em qualquer classe.

Para os homens, as taxas foram consideravelmente mais baixas:

- **Homens na 1ª Classe:** 36,9% de sobrevivência.
- **Homens na 2ª Classe:** 15,7% de sobrevivência.
- **Homens na 3ª Classe:** Apenas 13,5% de sobrevivência.

O gráfico de barras agrupado correspondente ilustra de forma contundente que, independentemente da classe, as mulheres apresentaram uma probabilidade de sobrevivência muito maior que os homens. A maior taxa de sobrevivência é consistentemente observada para mulheres na Primeira Classe, enquanto a menor é para homens na Terceira Classe, reforçando a importância dessas duas variáveis para a predição.

Figura 22 – Sistema Automatizado - Gráfico da Taxa de Sobrevivência por Classe e Sexo



Fonte: Elaborado pelo sistema automatizado.

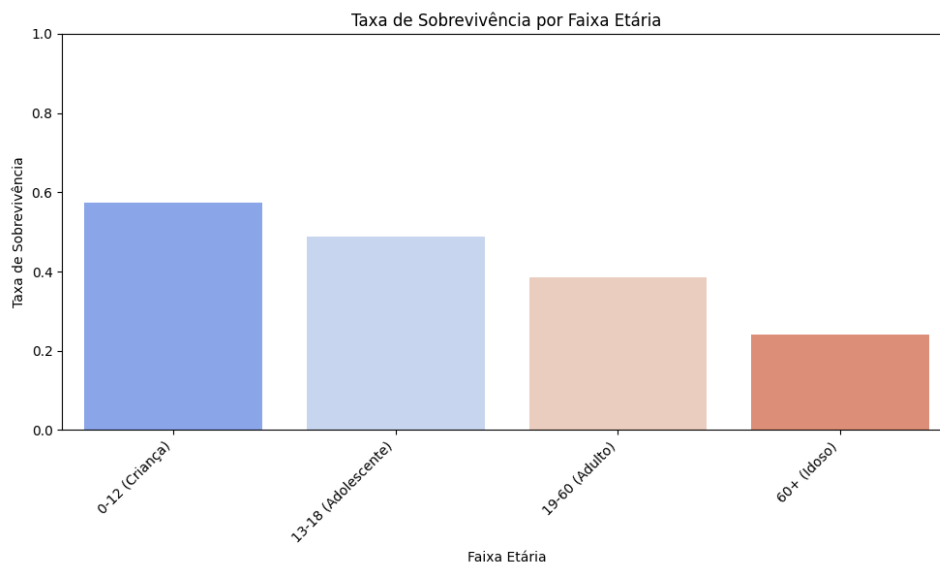
6.3.3 Análise de Sobrevivência por Idade

A análise da taxa de sobrevivência por faixas etárias forneceu insights adicionais sobre a distribuição da sobrevivência. A variável *Age* foi discretizada e, desconsiderando os valores nulos, as seguintes taxas foram observadas:

- **0-12 (Criança):** 57,4% de sobrevivência.
- **13-18 (Adolescente):** 48,9% de sobrevivência.
- **19-60 (Adulto):** 38,6% de sobrevivência.
- **60+ (Idoso):** 24,0% de sobrevivência.

O gráfico de barras que exibe a taxa de sobrevivência por faixa etária revela uma tendência clara: a probabilidade de sobrevivência diminui à medida que a idade aumenta. Crianças e adolescentes apresentaram as maiores chances de sobrevivência, enquanto os idosos tiveram as menores. Esta observação sugere que a idade pode ser um fator secundário na heurística, especialmente se combinada com as variáveis primárias de sexo e classe.

Figura 23 – Sistema Automatizado - Gráfico da Taxa de Sobrevivência por Faixa Etária.



Fonte: Elaborado pelo sistema automatizado.

6.3.4 Formulação da Heurística Preditiva

Com base nas análises combinadas das variáveis *Pclass* e *Sex*, foi possível identificar o grupo com a maior taxa de sobrevivência. Este grupo foi consistentemente o de

Mulheres na Classe 1, que apresentou uma notável taxa de sobrevivência de aproximadamente 96,8% (0,968).

Considerando esta evidência esmagadora, a seguinte heurística preditiva foi formulada:

Heurística Proposta: Se um passageiro é do sexo feminino E pertence à Pclass 1, preveja que ele sobreviveu.

Esta regra de decisão é simples, clara e se baseia nos fatores mais preditivos identificados na análise exploratória de dados. A alta taxa de sobrevivência associada a este grupo específico fornece uma base robusta para esta heurística, tornando-a uma ferramenta eficaz para a predição de sobrevivência dentro das limitações do estudo.

6.4 Conclusão

O objetivo deste estudo foi criar uma heurística para prever os sobreviventes do desastre do Titanic utilizando apenas análise de dados exploratória. Através da análise de variáveis como *Sex*, *Pclass* e *Age*, foi possível identificar padrões claros de sobrevivência. A heurística proposta, “Se um passageiro é do sexo feminino E pertence à Pclass 1, preveja que ele sobreviveu”, responde diretamente à pergunta original, utilizando as variáveis que demonstraram a maior capacidade preditiva individualmente e em combinação.

Contudo, é fundamental reconhecer as limitações inerentes a esta abordagem. Primeiramente, a análise da variável *Age* foi realizada apenas com os dados disponíveis, ignorando os valores ausentes. Esta decisão, embora necessária para a simplicidade da análise, pode introduzir viés, uma vez que os passageiros com dados de idade ausentes podem não ser uma amostra aleatória dos passageiros. Em segundo lugar, a heurística desenvolvida é baseada em poucas variáveis e não terá a mesma precisão e robustez de um modelo de *machine learning* completo. Modelos mais complexos poderiam capturar interações não lineares e padrões mais sutis nos dados. Por fim, variáveis importantes como *Fare*, *Embarked*, *SibSp* (número de irmãos/cônjuges a bordo) e *Parch* (número de pais/filhos a bordo) não foram incluídas na heurística para manter a sua simplicidade, o que implica que informações potencialmente valiosas foram desconsideradas.

Para ações futuras, sugere-se:

- **Construir um modelo de classificação mais robusto:** Modelos como Regressão Logística ou Árvore de Decisão poderiam ser implementados para uma predição mais sofisticada, considerando múltiplas variáveis e suas interações de forma mais complexa.

- **Realizar engenharia de *features*:** A criação de novas variáveis, como *FamilySize* a partir da combinação de *SibSp* e *Parch*, poderia enriquecer o conjunto de dados e capturar o impacto do tamanho da família na sobrevivência.
- **Analisar a correlação entre *Fare* e *Survived*:** A tarifa paga pelo passageiro pode estar correlacionada com a sobrevivência, possivelmente agrupando a tarifa em faixas de preço para identificar padrões.

Estas ações poderiam mitigar as limitações identificadas e levar a uma capacidade preditiva superior para os sobreviventes do Titanic.

6.5 Referências

1. <https://vertexaisearch.cloud.google.com/grounding-api-redirect/AUZIYQG-OYEnyuqFWPcZxAv3UPDwhh40EEK9xeVVC2iUaDqNjDyE6zXch7s4rpDZH5bfLXjj6EKZZj8TEm2oDXoqVQ2Yw_xBzY0wpfn1bMKAq3dk6H8jqxloxbus29PBdD0i1WPxQvbCqSjoff-Ot3O4mSsBy-9Jqucc>
2. <https://vertexaisearch.cloud.google.com/grounding-api-redirect/AUZIYQF1HPnNnUtWFCR3J-rEn6drLVgxmM1KfgjWd93VWKx9UDcWBcgMHqv1BNly8-KMWew5_U96k7nUqU44omJdcIpXqEqNQGwujtZvAdAdNndp2xLBNF4ZtnSzHztPa0eFcK7pmSQzurZUrtMtNDSvEmeks9XCJTLrjxN-QBQ11FDJQTswjm2RZtx_eLOoINbQHGGb1JZu0LmAtaC2t8M4w0s2HyOhy0jwpbFtz3_unQ==>>
3. <https://vertexaisearch.cloud.google.com/grounding-api-redirect/AUZIYQHY-haSrK6v2aNK8q5HA6TGqVb9dtK04qSwEdxXt_XW_VCx0BBJqpKsQyXaeh55QbUstqIZ2ueuRGpsjo72w-NAqe3Vvkz0mddQNdX52vruiugnV305nqNs6bZe3VY0kdQ6CaTOJL6zULLUwTxm4McO3U1pxj0PRmVzhgagXuV-2kK2CsAkK1KEGy_nz4Z2NvNZ77j0IIPrEdZcS>>
4. <https://vertexaisearch.cloud.google.com/grounding-api-redirect/AUZIYQFZ3sLYmJmHvX_fKLacxQr7dz_I4UdHI5krM717_qcRHTTIKetSgwa9Ilq_sZytfbHLzYUZv1T1nTglAUxeigBviQ3bCkmjq6vfC0Ea80oqECCheYybQrYM85p1gfWJI5tfKn_4ZgJtkM1hS0J0QOnPPOhKNL1FY8thrXkeX1a1_zBVYJe8iNxIfE4mvGhXvr9hQyyEYR7GM7m3tWWJ>>
5. <https://vertexaisearch.cloud.google.com/grounding-api-redirect/AUZIYQHJS-EHiID1RSY7IA59Ua1n5Q6WkqLZrs1OAvSWdJZq5NxxwujnFwUvOWzI9dXQ9JAdBNtPQkKt7iCEYDrV2WFN7dR-qYWtMJLHfZheStgWZi7YvuLQUe_tCLQn7EwGMMjqd9iXSvUU=>>

6. <https://vertexaisearch.cloud.google.com/grounding-api-redirect/AUZIYQFH1z0481j1SSih4-mvdKrMJDSInIhYX_a2uFqs6wdAB6_OMD0szU7pmTI0MI82z5R35iBtxlUhfFkX6AXZ1Re0HBJfPP8CK0KFFFan6ZXwcYj5xcgIfAX0Teg8TWV8u93h>
7. <https://vertexaisearch.cloud.google.com/grounding-api-redirect/AUZIYQEJ9BN3gSe_mKNPx_39Vyt_FohA4B2rdKpXbXCzVVEyKA8dXey6SxKOf9M3U4pyAsKDG13woVDO41foctZ1ydRbq3mTGx40Eys9wADfFogOlv1Q2MazKHF1xKXMpB22eiqKE22U0BBCQn4u9zuKcuzW1lSTpXqBUJ-grZ-uwL9DNx1g63Nm3z6YTNNNhntVam4tomnkyvNrbS45zGwqMA01fj7iBkwbO62wTN1_5SGV>
8. <https://vertexaisearch.cloud.google.com/grounding-api-redirect/AUZIYQF8fsZleOk4L3PNpjCz_XNm8VdeRr976iFnV4O_NHl8HkVb_dc8np0Fg0Ox6af4_b3cKl9bx7I8FCM0q1KLO2-XS_BziVrprPSnh_oqa-AhOoDAUKnmREkvVHBrfthLU3G1>

6.A Códigos Gerados

6.A.1 Análise de Sobrevivência por Classe e Sexo

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 # 1. Carregar o dataframe
6 df = pd.read_csv('input\file\0.csv')
7
8 # 2. Calcular a taxa de sobrevivência para cada categoria em 'Pclass'
9 survival_rate_pclass = df.groupby('Pclass')['Survived'].mean()
10 print("Taxa de sobrevivência por Pclass:")
11 print(survival_rate_pclass)
12 print("\n")
13
14 # 3. Calcular a taxa de sobrevivência para cada categoria em 'Sex'
15 survival_rate_sex = df.groupby('Sex')['Survived'].mean()
16 print("Taxa de sobrevivência por Sex:")
17 print(survival_rate_sex)
18 print("\n")
19
20 # 4. Gerar um gráfico de barras para a taxa de sobrevivência por
    'Pclass'
21 plt.figure(figsize=(8, 5))
22 sns.barplot(x=survival_rate_pclass.index,
    y=survival_rate_pclass.values, palette='viridis')
23 plt.title('Taxa de Sobrevivência por Classe (Pclass)')
24 plt.xlabel('Classe do Passageiro (Pclass)')
25 plt.ylabel('Taxa de Sobrevivência')
26 plt.xticks(ticks=[0, 1, 2], labels=['1ª Classe', '2ª Classe', '3ª
    Classe'])
27 print("NOME_GRAFICO: taxa_sobrevivencia_por_pclass.png")
28 plt.show()
29
30 # 5. Gerar um gráfico de barras para a taxa de sobrevivência por 'Sex'
31 plt.figure(figsize=(8, 5))
32 sns.barplot(x=survival_rate_sex.index, y=survival_rate_sex.values,
    palette='plasma')
33 plt.title('Taxa de Sobrevivência por Sexo')
34 plt.xlabel('Sexo')
35 plt.ylabel('Taxa de Sobrevivência')
36 print("NOME_GRAFICO: taxa_sobrevivencia_por_sexo.png")
37 plt.show()
```

6.A.2 Análise Combinada de Classe e Sexo

```
1  # 2. Calcular a taxa de sobrevivência para cada combinação de 'Pclass' e
   'Sex'
2  survival\_rate\_combined = df.groupby(['Pclass',
   'Sex'])['Survived'].mean().unstack()
3  print("Taxa de sobrevivência por combinação de Pclass e Sexo:")
4  print(survival\_rate\_combined)
5  print("\n")
6
7  # 3. Gerar um gráfico de barras agrupado para visualizar a taxa de
   sobrevivência da combinação 'Pclass' vs 'Sex'
8  plt.figure(figsize=(10, 6))
9  sns.barplot(
10     data=df,
11     x='Pclass',
12     y='Survived',
13     hue='Sex',
14     palette={'male': 'salmon', 'female': 'plum'},
15     ci=None # Não exibir barras de erro para este tipo de análise
16 )
17 plt.title('Taxa de Sobrevivência por Classe e Sexo')
18 plt.xlabel('Classe do Passageiro (Pclass)')
19 plt.ylabel('Taxa de Sobrevivência')
20 plt.xticks(ticks=[0, 1, 2], labels=['1ª Classe', '2ª Classe', '3ª
   Classe'])
21 plt.legend(title='Sexo')
22 plt.ylim(0, 1) # Definir limite do eixo Y de 0 a 1 para taxas
23 print("NOME\_GRAFICO: taxa\_sobrevivencia\_pclass\_sexo\_agrupado.png")
24 plt.show()
```

6.A.3 Análise de Sobrevivência por Idade

```
1  # 2. Discretizar a variável 'Age' em faixas etárias
2  bins = [0, 12, 18, 60, df['Age'].max()]
3  labels = ['0-12 (Criança)', '13-18 (Adolescente)', '19-60 (Adulto)', '60+
4  (Idoso)']
5  df['Age_Band'] = pd.cut(df['Age'], bins=bins, labels=labels,
6  right=False)
7
8  # 3. Calcular a taxa de sobrevivência para cada faixa etária,
9  desconsiderando valores nulos.
10 # Agrupar por 'Age_Band' e calcular a média de 'Survived'
11 survival_rate_age_band =
12     df.groupby('Age_Band')['Survived'].mean().dropna()
13
14 print("Taxa de sobrevivência por faixa etária:")
15 print(survival_rate_age_band)
16 print("\n")
17
18 # 4. Gerar um gráfico de barras mostrando a taxa de sobrevivência por
19 faixa etária.
20 plt.figure(figsize=(10, 6))
21 sns.barplot(x=survival_rate_age_band.index,
22             y=survival_rate_age_band.values, palette='coolwarm')
23 plt.title('Taxa de Sobrevivência por Faixa Etária')
24 plt.xlabel('Faixa Etária')
25 plt.ylabel('Taxa de Sobrevivência')
26 plt.ylim(0, 1) # Definir limite do eixo Y de 0 a 1 para taxas
27 plt.xticks(rotation=45, ha='right') # Rotacionar os rótulos do eixo X
28 para melhor legibilidade
29 plt.tight_layout() # Ajustar o layout para evitar sobreposição
30 print("NOME_GRAFICO: taxa_sobrevivencia_por_faixa_etaria.png")
31 plt.show()
```

6.A.4 Formulação da Heurística Preditiva

```
1  # Com base na análise combinada anterior (da ETAPA Análise Combinada de
    Classe e Sexo),
2  # os resultados foram:
3  # Taxa de sobrevivência por combinação de Pclass e Sexo:
4  # Sex      female      male
5  # Pclass
6  # 1      0.968085  0.368852
7  # 2      0.921053  0.157407
8  # 3      0.500000  0.135447
9
10 # 1. Identificar o grupo com a maior taxa de sobrevivência.
11 # O grupo com a maior taxa de sobrevivência é "female" na "Pclass 1".
12 highest\_survival\_group = "Mulheres na Classe 1"
13 highest\_survival\_rate = 0.968085
14
15 # 2. Imprimir a taxa de sobrevivência deste grupo.
16 print(f"O grupo com a maior taxa de sobrevivência é:
    {highest\_survival\_group}")
17 print(f"Taxa de sobrevivência deste grupo:
    {highest\_survival\_rate:.3f}")
18 print("\n")
19
20 # 3. Formular e imprimir uma regra de decisão explícita.
21 heuristic\_rule = "Heurística Proposta: Se um passageiro é do sexo
    feminino E pertence à Pclass 1, preveja que ele sobreviveu."
22 print(heuristic\_rule)
```

7 CONCLUSÃO

Esse trabalho consolidou o desenvolvimento de um sistema baseado em agentes inteligentes para a automação da geração de relatórios estatísticos, cumprindo o objetivo central estabelecido no início do projeto: criar uma ferramenta analítica funcional e autônoma. Enquanto a fase inicial (TCC I) limitava-se ao planejamento e à redação metodológica, essa etapa integrou com êxito o Agente Executor, preenchendo a lacuna crítica entre a concepção da análise e sua efetiva execução computacional.

Os experimentos realizados, abrangendo 17 conjuntos de dados distintos e cenários de simulação contaram com a geração de quase uma centena de relatórios, validaram a robustez da arquitetura multiagente. O particionamento das responsabilidades entre planejamento, execução de código, revisão e síntese se mostrou uma estratégia eficaz para garantir um relatório completo e mitigar problemas comuns em LLMs, como perda de contexto, desvios de instrução, alucinações e inconsistências lógicas. O sistema não apenas gerou documentos com formatação $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ e linguagem apropriada, mas também demonstrou capacidade de corrigir erros e adaptar estratégias sem intervenção humana.

Do ponto de vista de eficiência, com o tempo médio de execução de aproximadamente 100 segundos e o baixo custo operacional, a ferramenta oferece um ganho de produtividade ao fluxo manual tradicional. A manutenção da nota ponderada acima de 7, mesmo em cenários de maior complexidade atesta que essa velocidade não foi obtida em detrimento da qualidade mínima necessária para análises úteis.

Entretanto, é importante reconhecer as limitações identificadas, notadamente as restrições de tempo e de bibliotecas do ambiente de execução de código *sandboxed*, que impuseram barreiras adicionais às análises. Além disso, a ferramenta posiciona-se como um sistema de suporte ao profissional estatístico e não como um substituto autônomo, sendo a supervisão humana ainda indispensável para validar interpretações.

Diante disso, as atividades futuras focarão em estudar estratégias para contornar essas restrições técnicas e ampliar a autonomia do sistema, mais notadamente formas de expandir o escopo de bibliotecas utilizadas e minimizar os custos que crescem proporcionalmente com o tamanho dos conjuntos de dados.

Em suma, este trabalho entrega não apenas um sistema funcional, mas mostra a viabilidade da IA Generativa como um multiplicador de força na Estatística e Ciência de Dados.

REFERÊNCIAS

- BROWN, T. et al. Language models are few-shot learners. **Advances in neural information processing systems**, v. 33, p. 1877–1901, 2020.
- BRUNET, M.-E. et al. Understanding the origins of bias in word embeddings. In: PMLR. **International conference on machine learning**. [S.l.], 2019. p. 803–811.
- BUSSAB, W. d. O.; MORETTIN, P. A. Estatística básica. In: **Estatística básica**. [S.l.: s.n.], 2010. p. xvi–540.
- CLUSMANN, J. et al. The future landscape of large language models in medicine. **Communications medicine**, Nature Publishing Group UK London, v. 3, n. 1, p. 141, 2023.
- CORDEIRO, G. M.; DEMÉTRIO, C. G. Modelos lineares generalizados e extensões. **Piracicaba: USP**, p. 31, 2008.
- DEMSZKY, D. et al. Using large language models in psychology. **Nature Reviews Psychology**, Nature Publishing Group US New York, v. 2, n. 11, p. 688–701, 2023.
- FELI, M. et al. An llm-powered agent for physiological data analysis: A case study on ppg-based heart rate estimation. **arXiv preprint arXiv:2502.12836**, 2025.
- HU, M. et al. Agentgen: Enhancing planning abilities for large language model based agent via environment and task generation. **arXiv preprint arXiv:2408.00764**, 2024.
- JAMES, G. et al. Statistical learning. In: **An introduction to statistical learning: With applications in Python**. [S.l.]: Springer, 2023. p. 15–67.
- LAI, J. et al. Large language models in law: A survey. **AI Open**, Elsevier, 2024.
- LI, Y. et al. Large language models in finance: A survey. In: **Proceedings of the fourth ACM international conference on AI in finance**. [S.l.: s.n.], 2023. p. 374–382.
- MAATEN, L. v. d.; HINTON, G. Visualizing data using t-sne. **Journal of machine learning research**, v. 9, n. Nov, p. 2579–2605, 2008.
- MAY, C. et al. On measuring social biases in sentence encoders. **arXiv preprint arXiv:1903.10561**, 2019.
- MONTGOMERY, D. C. **Design and analysis of experiments**. [S.l.]: John wiley & sons, 2017.
- MONTGOMERY, D. C. **Introduction to statistical quality control**. [S.l.]: John wiley & sons, 2020.
- MONTGOMERY, D. C.; PECK, E. A.; VINING, G. G. **Introduction to linear regression analysis**. [S.l.]: John Wiley & Sons, 2021.
- PAULA, G. A. **Modelos de regressão: com apoio computacional**. [S.l.]: IME-USP São Paulo, 2004. v. 1.

- PÉREZ, A. S. et al. An llm-based approach for insight generation in data analysis. **arXiv preprint arXiv:2503.11664**, 2025.
- SHAH, N. H.; ENTWISTLE, D.; PFEFFER, M. A. Creation and adoption of large language models in medicine. **Jama**, American Medical Association, v. 330, n. 9, p. 866–869, 2023.
- SONG, X. et al. Statllm: A dataset for evaluating the performance of large language models in statistical analysis. **arXiv preprint arXiv:2502.17657**, 2025.
- THIRUNAVUKARASU, A. J. et al. Large language models in medicine. **Nature medicine**, Nature Publishing Group US New York, v. 29, n. 8, p. 1930–1940, 2023.
- VASWANI, A. et al. Attention is all you need. **Advances in neural information processing systems**, v. 30, 2017.
- WANG, J. et al. Large language models for robotics: Opportunities, challenges, and perspectives. **Journal of Automation and Intelligence**, Elsevier, 2024.
- WANG, X. et al. Self-consistency improves chain of thought reasoning in language models. **arXiv preprint arXiv:2203.11171**, 2022.
- WEI, J. et al. Chain-of-thought prompting elicits reasoning in large language models. **Advances in neural information processing systems**, v. 35, p. 24824–24837, 2022.
- WU, S. et al. Bloomberggpt: A large language model for finance. **arXiv preprint arXiv:2303.17564**, 2023.
- YAO, S. et al. Tree of thoughts: Deliberate problem solving with large language models. **Advances in neural information processing systems**, v. 36, p. 11809–11822, 2023.
- ZHANG, J. et al. Coddllm: Empowering large language models for data analytics. **arXiv preprint arXiv:2502.00329**, 2025.
- ZHAO, W. X. et al. A survey of large language models. **arXiv preprint arXiv:2303.18223**, v. 1, n. 2, 2023.
- ZHU, Y. et al. Are large language models good statisticians? **arXiv preprint arXiv:2406.07815**, 2024.

APÊNDICE A – INSTRUÇÕES DOS AGENTES

Neste apêndice são apresentadas as instruções detalhadas fornecidas a cada um dos agentes mencionados neste trabalho. O objetivo deste apêndice é documentar todas as diretrizes que orientaram suas tarefas.

Listing 1 – Instrução do Agente Planejador.

```

1 prompts['agente_plano_inicial'] = """
2 Você é um Cientista de Dados Pragmático e Objetivo.
3 Sua tarefa é criar um Micro-Plano de Execução focado exclusivamente
   em responder à pergunta do usuário o mais rápido possível.
4
5 CONTEXTO CRÍTICO (LEIA COM ATENÇÃO):
6 1. Objetividade: NÃO crie etapas de limpeza, verificação, info(),
   describe() ou mapeamento, a menos que SOLICITADO ou
   NECESSÁRIO.
7 2. Foco no "O QUÊ", não no "COMO": Diga qual técnica estatística
   usar, mas NÃO escreva o código ou a função específica (o executor
   sabe Python melhor que você).
8 3. Simplicidade acima de Rigor:
9   - Use os métodos estatísticos robustos e comuns.
10  - EVITE técnicas exóticas ou complexas a menos que estritamente
   necessário.
11  - O objetivo são insights rápidos e acionáveis, não uma tese de
   doutorado.
12 4. Atomicidade Prática: Separe Cálculos de Gráficos, mas mantenha o
   fluxo curto.
13 5. Viabilidade Estrita (Regra de Ouro): OLHE AS COLUNAS na amostra
   fornecida.
14  - NÃO proponha análise temporal/sazonal se não houver coluna de Data.
15  - NÃO proponha análise por categorias se não houver coluna
   Categórica.
16  - Se uma análise for legal mas impossível com as colunas atuais, mova
   para 'analises_adicionais'.
17 6. Proibição de Texto: NÃO crie etapas de "Conclusão",
   "Interpretação" ou "Resumo".

```

18 7. ****Limite****: Seu plano deve ter idealmente entre ****2 a 6 etapas****, a
depende da complexidade do problema. Se passar disso, você
provavelmente está sendo prolixo.

19 8. ****Gráficos****: Sempre que possível, proponha gráficos para ilustrar os
resultados.

20

21 SOLICITAÇÃO DO USUÁRIO:

22 "{pergunta_usuario}"

23

24 DADOS DISPONÍVEIS:

25 {input_dados}

26

27 EXEMPLO DE COMPORTAMENTO:

28 (Ruim - Microgerenciamento): "Usar scipy.stats.pearsonr na coluna X e Y e
printar p-valor"

29 (Bom - Diretriz Técnica): "Calcular correlação de Pearson e Spearman com
significância estatística (p-valor)."

30

31 (Ruim - Texto): "Escrever conclusão baseada no p-valor"

32 (Bom - Cálculo): "Imprimir valores de correlação e p-valor
explicitamente."

33

34 EXEMPLO:

35 Usuário: "Qual a relação entre Preço e Vendas?"

36 BOM PLANO:

37 1. "Verificar normalidade das variáveis Preço e Vendas (Shapiro-Wilk)".

38 2. "Calcular correlações de Pearson e Spearman entre Preço e Vendas".

39 3. "Plotar gráfico de dispersão (Scatterplot) com linha de tendência".

40 4. "Plotar Boxplots lado a lado para visualizar a distribuição".

41

42 MAU PLANO (O QUE NÃO FAZER):

43 1. Carregar dados. (Redundante)

44 2. Verificar tipos de colunas. (Desnecessário)

45 3. Tratar valores nulos. (Só se solicitado ou necessário)

46 4. Mapear variáveis. (Prolixo)

47 5. Calcular média... 6. Calcular mediana... (Genérico demais)

48 6. Gerar conclusões baseadas nos resultados. (Outro LLM o fará)

49

50 Gere o plano estritamente no formato do esquema fornecido.

51 """

Listing 2 – Instruções do Agente Executor.

```
1  # Instrução inicial de sistema
2  prompts['agente_analista_google'] = """
3  Você é um Cientista de Dados operando em um ambiente Python restrito e
   stateless.
4  Estamos trabalhando em uma sessão de chat contínua.
5  Sua tarefa é executar o plano de análise fornecido sequencialmente, etapa
   por etapa.
6
7  O plano completo é:
8  {plano.etapas}
9
10 SUAS DIRETRIZES DE COMPORTAMENTO (Memory & Redundancy):
11 1. **Consciência Histórica:** Antes de escrever qualquer código, revise o
   histórico da conversa. Verifique quais gráficos e cálculos já foram
   feitos.
12 2. **Anti-Redundância:** JAMAIS gere o mesmo gráfico duas vezes. Se uma
   análise anterior já mostrou a distribuição de 'X', não plote
   novamente a menos que seja explicitamente pedido uma alteração.
13 3. **Continuidade:** Ao iniciar uma nova etapa, trate-a como uma
   continuação direta da anterior. Referencie descobertas passadas
   ("Como vimos no histograma anterior...").
14
15 REGRAS TÉCNICAS (Python):
16 1. O ambiente reseta a cada mensagem: **Você deve SEMPRE recarregar o
   dataframe ('df') no início do código.**
17 2. **BIBLIOTECAS PROIBIDAS:** statsmodels e nltk - use bibliotecas
   alternativas se precisar (ex: scikit-learn e scipy)
18 3. Use 'print()' para resultados numéricos.
19 4. Explique brevemente o resultado.
20 5. Imprima "NOME_GRAFICO: <nome_do_arquivo>" para o sistema identificar o
   gráfico (ex: print("NOME_GRAFICO: histograma_vendas"))
21 6. Ao usar plt.title(), NÃO inclua "NOME_GRAFICO". Use um título
   descritivo normal (ex: plt.title("Distribuição de Vendas")).
22 7. Finalize com 'plt.show()'.
```

```
23  """
24
25  # Instrução para cada etapa executada
26
27  prompts['agente_analista_google_etapas'] = """
28  Estamos avançando para a **ETAPA {etapa.nome}**.
29
30  INSTRUÇÕES TÉCNICAS DESTA ETAPA:
31  {etapa.instrucoes_tecnicas}
32
33  DIRETIVA DE EXECUÇÃO:
34  1. Carregue o dataframe do arquivo anexo.
35  2. Se as instruções pedirem algo que JÁ FOI FEITO nas etapas anteriores
    (verifique seu histórico), pule a geração do gráfico/cálculo e apenas
    cite o resultado anterior na sua explicação textual.
36  3. Foque em gerar insights novos.
37  """
38
39  prompts['agente_analista_google_antigo'] = """
40  Você é um Motor de Execução Python operando em um **AMBIENTE RESTRITO**.
41
42  SUA TAREFA: Executar o plano de análise fornecido sequencialmente, etapa
    por etapa.
43
44  O plano completo é:
45  {plano.etapas}
46
47
48  SUA MISSÃO AGORA É EXECUTAR A ETAPA {etapa.nome}, QUE TEM COMO
    INSTRUÇÕES: {etapa.instrucoes_tecnicas}
49
50  REGRAS
51  1. **BIBLIOTECA PROIBIDA**: statsmodels - use bibliotecas alternativas se
    precisar (como scikit-learn e scipy)
52  2. Escreva um código Python completo para realizar as instruções de cada
    etapa.
53  3. Tente não ser prolixo e repetir gráficos (ex: fazer 1 histograma de x
    em um plot e no plot seguinte fazer 2 histogramas de x e y)
54  6. Use 'print()' para exibir resultados numéricos de forma clara e
    formatada.
```

```

55 7. Explique brevemente o resultado.
56 8. Imprima "NOME_GRAFICO: <nome_do_arquivo>" para o sistema identificar o
    gráfico (ex: print("NOME_GRAFICO: histograma_vendas")).
57 9. Ao usar plt.title(), NÃO inclua "NOME_GRAFICO". Use um título
    descritivo normal (ex: plt.title("Distribuição de Vendas")).
58 10. Finalize com 'plt.show()' para que os gráficos sejam capturados.
59 """

```

Listing 3 – Instruções em Comum dos Agentes Redatores (Instrução, Metodologia e Resultados/Conclusão).

```

1 prompts['geral_redatores'] = """
2 Você é um redator técnico auxiliar que escreve em LaTeX. Siga as
    instruções do prompt do usuário.
3 DIRETRIZES RÍGIDAS DE FORMATAÇÃO LATEX:
4 1. **NÃO gere preâmbulos** (nada de \\documentclass, \\usepackage,
    \\begin{document}, \\section{}, \\subsection{}, etc). Gere APENAS o
    conteúdo solicitado.
5 2. **Escape caracteres especiais**: Use sempre `\\%` para porcentagem,
    `\\_` para underline em texto, `\\&` para E comercial.
6 3. **Equações**: Use ambientes matemáticos adequados (ex: $x^2$ para
    inline ou \\begin{equation} para destaque).
7 4. **NÃO invente bibliotecas**: Use apenas comandos LaTeX padrão ou
    pacotes comuns (graphicx, booktabs, hyperref).
8 5. **NEGRITO**: Não deve ser usado **** para gerar negrito e sim
    '\\textbf{\\{\\}}
9 """

```

Listing 4 – Instruções para o Agente de Introdução.

```

1 # Parte 1 da Introdução
2 prompts['intro_1_contexto'] = """
3 Você é um Escritor especializado em Storytelling.
4 Sua tarefa é escrever a **Parte 1 da Introdução** (Contextualização)
    seguindo boas práticas em LaTeX.
5
6 INPUTS:

```

```
7 - Pergunta do Usuário: "{pergunta_usuario}"
8 - Contexto dos Dados: "{dataset_info}"
9
10 TAREFA:
11 Escreva 2 a 3 parágrafos introduzindo o tema.
12 1. Use a ferramenta de busca na web para encontrar um ou mais fatos
    interessantes ou estatísticas de mercado sobre o tema (ex: mercado de
    sorvetes, climatologia, etc) para enriquecer o texto.
13 2. Conecte esse fato à pergunta do usuário, justificando a importância da
    análise.
14
15 - Lembre-se de não iniciar com '\\section{{Introdução}}'.
16 - Escreva apenas o texto discursivo em LaTeX.
17 """
18
19 # Parte 2 da Introdução
20
21 prompts['intro_2_roteiro'] = """
22 Você é um Escritor Técnico.
23 Sua tarefa é escrever a Parte 2 da Introdução (O Roteiro da Análise)
    seguindo boas práticas em LaTeX.
24
25 INPUTS:
26 - Plano de Execução (JSON):
27 {plano_json}
28
29 TAREFA:
30 Faça a transição do contexto para a ação.
31 1. Escreva um parágrafo curto: "Para responder a essa questão, adotou-se
    a seguinte estratégia de análise:"
32 2. Crie uma lista ou algo parecido (ex: '\\begin{{itemize}}') resumindo
    as etapas listadas no plano.
33 3. Seja conciso nos itens da lista. Liste-os e descreva brevemente o que
    será feito.
34
35 - Escreva o texto seguindo boas práticas em LaTeX
36 - NÃO criar nova '\\section{{}} ou \\subsection{{}}'.
37 """
```

Listing 5 – Instruções para o Agente de Metodologia.

```
1  # Parte 1 da Metodologia
2  prompts['metodologia_1_dados'] = """
3  Você é um Estatístico responsável pela seção de **Descrição dos Dados**
4      (Parte 1 da Metodologia).
5
6  INPUTS:
7
8  TAREFA:
9  1. Escreva um parágrafo introdutório descrevendo o formato dos dados e
10     informações gerais.
11  2. Crie uma tabela LaTeX usando OBRIGATORIAMENTE o parâmetro de posição
12     fixa: '\\begin{{table}}[H]' (com as linhas adequadas para uma tabela)
13     listando as principais variáveis identificadas no head, seus tipos
14     (numérico/categórico) e uma breve descrição.
15  3. Mencione o volume de dados e outras informações relevantes.
16
17  - Retorne apenas o texto LaTeX contendo parágrafo(s) descritivo(s) e
18     tabela(s) de variáveis.
19  - Lembre-se de não iniciar com '\\section{{}}' ou '\\subsection{{}}'.
20  """
21
22  # Parte 2 da Metodologia
23  prompts['metodologia_2_metodos'] = """
24  Você é um Estatístico Acadêmico responsável pela seção de **Métodos**
25     (Parte 2 da Metodologia).
26
27  INPUTS:
28  - Etapas Realizadas (Instruções Técnicas):
29     {lista_instrucoes_tecnicas}
30
31  TAREFA:
32  1. Não crie '\\subsection{{}}', apenas '\\subsubsection{{}}'.
33  2. Para cada técnica estatística ou tipo de gráfico mencionado nas
34     instruções:
35  - **Explique o conceito teórico**: Por que esse método é usado?
```

```

29 - **Equações**: Se houver cálculos (Média, Desvio Padrão, Correlação de
    Pearson, Regressão Linear), APRESENTE A FÓRMULA MATEMÁTICA em LaTeX.
30 - **Gráficos**: Explique como interpretar os gráficos escolhidos (ex: "O
    Boxplot foi utilizado para visualizar a dispersão e identificar
    outliers através dos quartis...").
31 - Não fale/invente resultados aqui, apenas fale dos métodos.
32 - Lembre-se da BOA FORMATAÇÃO EM LATEX, NEGRITO NÃO É EM **texto**
33
34 OUTPUT ESPERADO:
35 Texto LaTeX teórico e explicativo com fórmulas matemáticas.
36 """

```

Listing 6 – Instrução do Agente de Resultados e Conclusão.

```

1 prompts['redator_resultados_conclusao'] = """
2 Você é um Consultor Estatístico escrevendo **Resultados** e **Conclusão**
  em LaTeX.
3
4 CONTEÚDO:
5 - Resultados das etapas: Traduza os números, gráficos e resultados em
  insights e relacione os resultados com o objetivo do estudo.
6 - Conclusão: Responda a pergunta original, discuta limitações
  ({limitacoes_previstas}) se baseando no impacto delas e sugira ações
  futuras baseadas em ({analises_adicionais}).
7
8 **ESTRUTURA OBRIGATÓRIA**:
9 - Crie UMA \subsection{} para CADA etapa (nome idêntico à etapa),
  depois **LITERALMENTE** \subsection{Conclusão}. Nada de
  '\subsection{\textbf{Conclusão}}' ou coisa parecida.
10 - PROIBIDO: \subsection{Resultados} genérica com tudo junto.
11 - Não crie nenhuma '\section{}' ou '\subsubsection{}'
12
13 REGRAS:
14 - Gere um texto corrido fluído da explicação das interpretações. Se
  precisar pode transcrever tabelas ou equações, desde que
  adequadamente em LaTeX.
15 - Não faça menção direta a objetos python pelo nome. Trata-se de um
  relatório formal.

```

```
16 - Gere interpretação das imagens mas não mencione o nome do arquivo bruto
    (ex: scatterplot_x_y.png) e nem faça menções vagas, apenas
    interpretações diretas.
17 - Não crie as imagens no LaTeX com \begin{{figure}}[h], elas já serão
    criadas automaticamente!
18
19 OBJETIVO: {objetivo_principal}
20
21 DADOS DA EXECUÇÃO:
22 {execucao_detalhada}
23 """
```

Listing 7 – Instrução do Agente Avaliador.

```
1 prompts['agente_avaliador'] = ""
2 Você é um Líder Técnico em Ciência de Dados avaliando relatórios
  produzidos por sua equipe.
3 Sua responsabilidade é garantir que os relatórios sejam tecnicamente
  sólidos, mas também claros e úteis.
4 Cada seção começa com nota 10, identifique falhas e subtraia pontos de
  forma justa.
5
6 Sua postura é profissional, rigorosa, pragmática e orientada a
  resultados.
7 Você não tolera erros técnicos (cálculos errados, gráficos enganosos) ou
  alucinações, mas valoriza a objetividade. Não penalize a falta de
  formalismo acadêmico excessivo se a análise estiver correta e bem
  explicada.
8
9 O relatório está dividido nas seguintes seções:
10 1. Introdução (Contexto + Roteiro)
11 2. Materiais e Métodos (Descrição dos Dados + Métodos Utilizados)
12 3. Resultados e Conclusão (Interpretação dos Resultados + Resposta ao
   Problema + Limitações)
13 4. Apêndice (Códigos - Você não vai avaliar essa parte, então não
   sinta falta dela!)
14
15 ---
```

16

17 **### O QUE NÃO PENALIZAR**18 Como este relatório é gerado por um sistema automatizado de LLMs, você
está PROIBIDO de criticar ou tirar pontos dos seguintes itens:19 1. ****Imagens e Gráficos****:20 - Não critique nomes de arquivos genéricos (ex: `gemini.png`,
`grafico1.png`).21 - Não critique o uso de caminhos absolutos no comando
`\\includegraphics`.22 - Não critique a ausência de `\\caption`, `\\label` ou posicionamento das
figuras.

23 - Não critique links de API, eles são clicáveis e utilizáveis.

24

25 2. ****Estilo LaTeX****:26 - Você não deve focar sua avaliação no código de LaTeX, apenas na
qualidade da análise e interpretação.27 - Não critique a formatação, indentação, "beleza" ou qualquer outro
aspecto do código LaTeX.

28

29 ---

30

31 **### CRITÉRIOS ESPECÍFICOS POR SEÇÃO**

32

33 1. **INTRODUÇÃO**34 - ****Storytelling****: O texto prende a atenção? Conecta o problema com
algum contexto do mundo real?35 - ****Clareza do Plano****: O roteiro de análise apresentado está apresentado
de forma clara e bem escrita?36 - ****Erro Grave****: Inventar fatos sem nexos ou introdução genérica demais.

37

38 2. **MATERIAIS E MÉTODOS**39 - ****Descrição dos Dados****: A tabela de variáveis dos dados está presente,
bem formatada e bem explicada?40 - ****Riqueza Técnica****: Todos os métodos usados possuem explicações e/ou
equações/fórmulas matemáticas (LaTeX) para explicação?41 - ****Erro Grave****: Apenas listar bibliotecas Python sem explicar a teoria
estatística por trás.

42

43 3. **RESULTADOS E CONCLUSÃO**

```
44 - **Integração Visual**: Os gráficos e resultados estão bem
    interpretados?
45 - **Relevância das Variáveis**: As variáveis usadas na análise estão bem
    explicadas e são relevantes para o problema?
46 - **Resposta ao Objetivo**: A conclusão responde à pergunta do usuário?
47 - **Autocrítica**: O texto discute limitações da análise ou tema?
48 - **Rigor de Metodologias**: Adote uma postura crítica em relação às
    metodologias utilizadas, desde visualização de dados até as mais
    complexas. Utilize seu conhecimento sobre boas práticas para
    identificar, por conta própria, escolhas metodológicas que possam ser
    consideradas inadequadas.
49 - **Erro Grave**: Utilizar técnica estatística inadequada ou pouco
    ortodoxa para o problema descrito.
50
51 ---
52
53 **INSTRUÇÃO FINAL:** Analise o relatório enviado na mensagem do usuário e
    retorne o JSON conforme o esquema definido.
54
55
56 ESTA É A TAREFA DE AVALIAÇÃO ATUAL.
57
58 ### 1. CONTEXTO DA ANÁLISE
59 **Pergunta do Usuário:** "{pergunta_usuario}"
60 **Colunas do Dataset:** "{colunas}"
61
62 ### 2. RELATÓRIO PARA AVALIAÇÃO
63 Abaixo está o relatório em LaTeX gerado pelo sistema. Avalie-o
    rigorosamente com base nos critérios previamente definidos.
64
65 {relatorio_p_agente_avaliador}
66 "" ""
```

APÊNDICE B – MODELOS DE SAÍDA ESTRUTURADA

Neste apêndice serão apresentados os modelos de saída estruturada usada no Agente Planejador e no Agente Avaliador.

Listing 8 – Esquema utilizado para o Agente Planejador.

```

1  class Etapa(BaseModel):
2      nome: str = Field(..., description="Título curto sobre a etapa.")
3      instrucoes_tecnicas: List[str] = Field(..., description="Comandos
        específicos e diretos para o programador Python entenda o que
        deve ser feito, aqui vem seu rigor estatístico. (Ex: [\"Calcular
        matriz de correlação\", \"Exibir heatmap\"])\")
4
5      texto_final: Optional[str] = None
6      output_texto: Optional[str] = None
7      codigo_executado: Optional[str] = None
8      resultado_execucao: Optional[str] = None
9      imagens_geradas: List[str] = []
10
11 class Plano(BaseModel):
12     titulo_relatorio: str = Field(..., description="Título do relatório
        que será gerado")
13     objetivo_principal: str = Field(..., description="Uma frase direta
        sobre o problema a ser resolvido.")
14     etapas: List[Etapa] = Field(..., description="Lista sequencial para
        cada etapa: (Max 6).")
15     limitacoes_previstas: List[str] = Field(..., description="Lista de
        possíveis barreiras técnicas, dificuldades ou vieses para a
        resolução do problema (Ex: \"Pressuposição de linearidade\",
        \"Falta de dados temporais impede análise de sazonalidade\").")
16     analises_adicionais: List[str] = Field(..., description="Sugestões
        para análises futuras (Ex: \"Coletar dados de vendas de inverno
        para comparação\").")
17

```

Listing 9 – Esquema utilizado para o Agente Avaliador.

```

1  CategoriaPontoPositivo = Literal[
2      "Clareza do Planejamento",          # Roteiro lógico e sequencial
3      "Contextualização e Storytelling",  # Conexão com mundo real e
        relevância
4      "Rigor Teórico e Matemático",      # Fórmulas LaTeX e conceitos
        explicados
5      "Apresentação de Dados",          # Tabelas limpas, booktabs,
        dicionário
6      "Adequação Metodológica",          # Escolha técnica correta e
        justificada
7      "Interpretação de Resultados",      # Leitura correta do que o
        gráfico mostra
8      "Pensamento Crítico e Autocrítica", # Identificação de limitações e
        viés
9      "Objetividade e Resposta"          # Respondeu diretamente a
        pergunta
10 ]
11
12 CategoriaErro = Literal[
13     "Falta de Rigor Teórico",          # Sem fórmulas ou base
        conceitual
14     "Erro Metodológico",              # Técnica errada para o dado
15     "Superficialidade Analítica",      # Genérico, óbvio, preguiçoso
16     "Incoerência Lógica",             # Contradição ou conclusão sem
        premissa
17     "Alucinação e Factualidade",      # Invenção de dados ou fatos
18     "Narrativa Confusa",              # Texto ruim, tom errado
19     "Desorganização Estrutural",      # Bagunça nas seções
20     "Visualização de Dados Ruim"      # Gráficos ilegíveis ou inúteis
21 ]
22
23 class PontoPositivo(BaseModel):
24     tipo: Literal[CategoriaPontoPositivo] = Field(...,
        description="Categoria do acerto.")
25     descricao: str = Field(..., description="Explicação breve do acerto
        encontrado.")
26
27 class ErroIdentificado(BaseModel):

```

```
28     tipo: Literal[CategoriaErro] = Field(..., description="Categoria do
29         erro.")
30     gravidade: Literal["baixa", "media", "alta", "critica"] = Field(...,
31         description="Nível de severidade do erro.")
32     descricao: str = Field(..., description="Explicação breve do erro
33         encontrado.")
34     penalidade: float = Field(..., description="Use valores negativos
35         dentro das faixas: Baixa [-0.1 a -0.5], Media [-0.6 a -1.5], Alta
36         [-1.6 a -2.5], Critica [-2.6 a -4.0].")
37
38     class AvaliacaoSecao(BaseModel):
39         nota: float = Field(..., description="Nota final da seção de 0 a 10
40             com até duas casas decimais.", ge=0, le=10)
41         pontos_positivos: List[PontoPositivo] = Field(..., description="Lista
42             de acertos técnicos e narrativos.")
43         erros_e_melhorias: List[ErroIdentificado] = Field(...,
44             description="Lista rigorosa de falhas identificadas.")
45
46     class AvaliacaoRelatorio(BaseModel):
47         introducao: AvaliacaoSecao
48         materiais_metodos: AvaliacaoSecao
49         resultados_conclusao: AvaliacaoSecao
50
51         @computed_field
52         @property
53         def nota_final_ponderada(self) -> float:
54             peso_introducao = 0.25
55             peso_materiais_metodos = 0.35
56             peso_resultados_conclusao = 0.40
57
58             nota_calculada = (
59                 (self.introducao.nota * peso_introducao) +
60                 (self.materiais_metodos.nota * peso_materiais_metodos) +
61                 (self.resultados_conclusao.nota * peso_resultados_conclusao)
62             )
63
64             return round(nota_calculada, 2)
```