



## Vamos conversar sobre as LLM's possíveis para o projeto.

O arquivo que vamos ter que usar, tem 78MB

O arquivo indicado é o famoso dataset de fraude em cartão de crédito do Kaggle.

Características conhecidas (versão original):

Linhas (registros de transações): 284807

Colunas: 31

(Time, V1 ... V28 (componentes PCA), Amount, Class)

C:\Users\A.Gil\Downloads\i2a2\Kaggle - Credit Card Fraud\creditcard.csv

Tamanho, mais de 124 MB

Com essas características, qual LLM vc me aconselha usar, falando de API. Uma LLM gratuita, como Grok consegue atender, já que é um projeto de curso?

Para um projeto de curso com um dataset de fraude em cartão de crédito do Kaggle com cerca de 78MB (e o arquivo original tem em torno de 124MB e 284.807 registros), o uso de uma LLM para análise direta no dataset completo pode ser desafiador dependendo do contexto da chamada API, pois o modelo recebe texto/token e não arquivos brutos.

### Sobre LLMs Gratuitas como Grok

- Grok 3, uma LLM gratuita bastante conhecida, possui limites de uso no plano gratuito, como número restrito de queries em determinados períodos e limites no tamanho de tokens por requisição (em torno de 128 mil tokens por chamada). Para um dataset desse tamanho, você precisaria fazer a prévia separação, resumir ou extrair características específicas para enviar ao modelo, pois não é possível carregar o dataset bruto integralmente numa única chamada. [\[1\]](#) [\[2\]](#) [\[3\]](#) [\[4\]](#)

### Recomendações para Uso em Projeto

- Para esse tipo de análise, normalmente, a LLM seria usada para:
  - Gerar insights baseados em perguntas específicas sobre o dataset.
  - Auxiliar na criação de código para manipulação e análise dos dados.
  - Suporte para explicações, geração de relatórios e automação de análises baseadas em resultados agregados ou amostras do dataset.
- Modelos como Grok funcionam bem para esses casos desde que seja feita a preparação dos dados externamente (usando Python, Pandas, etc.), pois o dataset inteiro não é

adequado para ingestão direta em uma LLM via API.

## Alternativas e Abordagens

- Avaliar uma abordagem de RAG (Retrieval Augmented Generation), onde a LLM consulta um banco vetorial com representações do dataset, permitindo assim trabalhar com grandes volumes de dados localmente e usar o LLM para contexto e geração.
- Em projetos educativos, Grok ou outras LLM gratuitas podem ser suficientes para fornecer explicações, gerar código e modelar interações sobre a fraude, desde que o tratamento do dado bruto permaneça no lado do código externo, utilizando o dataset para alimentar a análise e interpretar outputs da LLM.

Em resumo, Grok pode atender para o projeto de curso comunicando com a LLM via API para tarefas interpretativas, código e perguntas específicas, mas não para ingestão do dataset inteiro diretamente. É essencial integrar processamento local com consulta inteligente para maximizar eficiência na análise do crédito. [\[5\]](#) [\[2\]](#) [\[3\]](#) [\[1\]](#)

Se desejar, posso ajudar a montar exemplos de código Python para pré-processar o dataset e estruturar chamadas para a LLM.

```
<div style="text-align: center">✖✖</div>
```

## **Analise o que foi avaliado pelo Agente do Copilot com relação ao dataset que estamos falando. Digo arquivo csv de falde de cartões de crédito**

Answer skipped.

```
<div style="text-align: center">✖✖</div>
```

## **Analise o que foi avaliado pelo Agente do Copilot com relação ao dataset que estamos falando. Digo arquivo csv de falde de cartões de crédito**

O que o Agente do Copilot avaliou sobre o dataset de fraude em cartão de crédito (creditcard.csv) foi um processo bastante estruturado voltado para uso na arquitetura RAG multiagente do projeto.

Aqui está o resumo da análise feita pelo agente:

## Visão Geral do Dataset

- Dataset público do Kaggle com cerca de 284.807 registros e 31 colunas (incluindo as componentes PCA V1 a V28, Time, Amount e Class).
- Altamente desbalanceado: só cerca de 492 casos de fraude (0,172% do total).
- Dados numéricos com features transformadas via PCA, anonimizados e sem significado direto nas componentes.
- Métricas recomendadas para avaliação: AUC-ROC, Precision-Recall, F1 para classe positiva, MCC. Evitar usar apenas acurácia.
- Recomendação de técnicas de reamostragem (SMOTE, undersampling) ou ajuste de threshold.

## Estratégia Proposta para Uso da RAG e Embeddings

- Gerar resumos estatísticos compactos para cada feature e agrupamentos.
- Construir uma camada de contexto para perguntas analíticas em linguagem natural, por exemplo: "Quais variáveis apresentam mais correlação com fraudes?"
- Suporte a um agente de profiling para sintetizar padrões e anomalias.
- Criar embeddings textuais a partir de descrições derivadas, não diretamente dos vetores PCA numéricos.
- Proposta de divisão em chunks textuais (resumos de clusters, perfis, importâncias, estatísticas), com tamanho médio de cerca de 120 tokens por chunk.
- Gerar e indexar esses chunks com embeddings (dimensão 1536), armazenados em banco vetorial (exemplo: Postgres pgvector) para consultas semânticas eficientes.

## Estimativas e Eficiência

- Embedding do dataset linha a linha seria custoso e ineficiente (~42 milhões de tokens).
- A estratégia reduz tokens drasticamente para algumas dezenas de chunks textuais.
- Uso de técnicas de clustering (ex: KMeans, HDBSCAN) para agrupar transações e gerar descrições agregadas.
- Uso de testes estatísticos KS para definir importância das features.

## Pipeline e Arquitetura Técnica

- Pipeline inclui ingestão segura do CSV, geração de perfis, clustering, geração textual, embeddings, armazenamento e indexação.
- Dados armazenados em tabelas relacionando chunks, embeddings, metadados e índices para busca.
- Módulos Python planejados para cada etapa (profiling, clustering, geração de descrições, indexação, etc).

- Preocupação em evitar reverter as componentes PCA anonimizadas para preservar ética e privacidade.

## Riscos e Mitigações

- Overfitting em clusters muito específicos que podem gerar ruído no contexto.
- Custo elevado se granularidade explodir, mitigado limitando número inicial de chunks e expandindo sob demanda.
- Garantia de dimensionamento correto do vetor de embedding.
- Restringir respostas para evitar exposição de dados individuais (só agregados e análises).

## Próximos Passos

- Implementar módulos de profiling e geração de descrições para alimentar a camada semântica.
- Testar embedding e indexação incremental.
- Criar agentes especializados para ingestão, profiling, indexação e orquestração de perguntas ao modelo.

Em suma, o agente do Copilot avaliou o dataset e elaborou um plano para transformar o grande dataset numérico de fraude em um conhecimento resumido, acessível via perguntas em linguagem natural, utilizando técnicas de clustering, geração textual e embeddings para viabilizar consulta semântica eficiente e escalável na arquitetura RAG do projeto.

Posso ajudar a detalhar qualquer uma das etapas desse pipeline ou montar exemplos de código para o trabalho com o dataset.

```
<div style="text-align: center">✖✖</div>
```

**Guarde essa sugestão que vc fez, vamos usala adiante. Vou perguntar, liste as suas sugestões deste tópico e preciso que vc liste essa e as próximas para podermos definir as linhas de ação. De forma resumida, foi valiosa para o projeto essa analise?**

A sugestão que foi feita referente à análise do dataset de fraude em cartão de crédito está muito valiosa para o projeto, pois oferece uma estratégia clara, eficiente e escalável para transformar um arquivo grande e numérico em um conhecimento textual e consultável por meio de uma arquitetura RAG com LLMs.

## **Sugestão Guardada para Uso Futuro**

- Transformar o dataset em resumos estatísticos.
- Realizar clustering para agregação e gerar descrições textuais.
- Criar chunks textuais e vector embeddings para consulta eficiente.
- Utilizar métricas específicas para lidar com desequilíbrio das classes.
- Implementar pipeline modular para profiling, clustering, indexação e orquestração.

## **Avaliação da Validação para o Projeto**

- Altamente valiosa para definir linhas de ação práticas.
- Permite trabalhar com dataset grande e complexo via LLM, sem sobrecarregar a API.
- Facilita a elaboração de perguntas em linguagem natural sobre o conjunto de dados.
- Propicia abordagem ética e eficiente na análise do dataset transformado.

Quando quiser, posso listar essa e as próximas sugestões/contextos levantados para definir as linhas de ação do projeto de forma resumida e organizada. Essa organização vai facilitar o planejamento e execução do projeto. A sugestão feita, com foco em transformar o dataset numérico e grande em resumos estatísticos, clusters e descrições textuais para geração de embeddings e consulta eficiente via RAG, é muito valiosa para o projeto. Ela traz uma linha de ação clara para trabalhar com o dataset de fraude usando LLMs sem sobrecarregar a API, além de lidar com o desbalanceamento e preservar privacidade.

Posso listar essa e as próximas sugestões para definirmos as linhas de ação de forma organizada e resumida, facilitando o planejamento e execução do projeto.

## **Essas sugestões de técnicas, vão de encontro e garantem eficiencia e precisão nas respostas as questões levantadas no documento anexo no contexto deste espaço referente ao desafio extra da I2A2? Seja breve na resposta, máximo 2 paragrafos curtos**

Sim, as técnicas sugeridas vão exatamente ao encontro das questões do desafio extra I2A2, garantindo eficiência e precisão nas respostas. A abordagem de transformar o dataset em resumos estatísticos, aplicar clustering para detectar padrões e gerar descrições textuais para alimentar um sistema RAG com embeddings permite responder perguntas analíticas complexas, produzir visualizações e identificar anomalias de forma escalável e contextualizada.

Além disso, o uso dessas técnicas atende à necessidade do agente de funcionar de forma genérica para qualquer arquivo CSV, com respostas automatizadas, criação de gráficos e geração de conclusões baseadas em memória de análise, conforme especificado no documento

do desafio. Assim, garante não só o desempenho técnico, como também a usabilidade e flexibilidade exigidas para o projeto. [\[23\]](#)

```
<div style="text-align: center">✖✖</div>
```

## **Veja o arquivo de estimativa de custo da API, valide se está coerente e de acordo com a realidade**

O arquivo de estimativa de custo da API está coerente e alinhado com a realidade para um projeto acadêmico utilizando LLMs na abordagem RAG com dataset de fraude em cartão de crédito. A estimativa considera parâmetros realistas de tokens por query, contexto e resposta, com cerca de 40 chunks embeddados e uso de top-k seleções para reduzir custo. O cenário mais intenso de uso ainda mantém o custo mensal abaixo de US\$ 3 para o modelo GPT-4o mini, que oferece bom custo-benefício.

A análise também apresenta recomendações práticas como uso de cache de prompt, limitação do número de chunks lidos por query e monitoramento do consumo de tokens, além de considerar planos alternativos (ex: Sonar/Perplexity) sob um custo fixo mais alto. As fórmulas e valores tarifários estão de acordo com preços públicos atuais típicos da OpenAI e práticas comuns de uso de APIs LLM, tornando a estimativa confiável para planejamento do projeto. Posso ajudar a detalhar ou ajustar cenários específicos se necessário.

```
<div style="text-align: center">✖✖</div>
```

## **É possível limitar esse custo, na OPEN AI, exemplo, travar a despesa em 5 dolares para evitar surpresa e cobrança além do esperado?**

Sim, na OpenAI é possível limitar os custos configurando um gasto máximo ("hard limit") para seu projeto ou conta, que bloqueia automaticamente as chamadas API quando esse teto for atingido, evitando cobranças maiores do que o esperado. Além disso, pode-se configurar alertas de uso ("soft limit") para receber notificações antes de chegar ao limite definido, o que ajuda no controle dos gastos.

Essas configurações são feitas no painel da OpenAI em Settings > Billing > Usage Limits, onde pode-se estabelecer limites mensais para projetos e restringir quais modelos podem ser usados para evitar uso inadvertido de opções mais caras. Essa abordagem garante maior segurança no orçamento e evita surpresas na cobrança. [\[25\]](#) [\[26\]](#) [\[27\]](#)

```
<div style="text-align: center">✖✖</div>
```

1. <https://www.byteplus.com/en/topic/499378>
2. <https://www.byteplus.com/en/topic/568375>
3. <https://www.cometapi.com/does-grok-3-have-a-limit/>
4. [https://www.reddit.com/r/grok/comments/1jpjmy6/free\\_tier\\_limits\\_as\\_of\\_2nd\\_april\\_2025/](https://www.reddit.com/r/grok/comments/1jpjmy6/free_tier_limits_as_of_2nd_april_2025/)
5. <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>
6. <https://arxiv.org/html/2506.11635v1>
7. <https://www.kaggle.com/datasets/ealtman2019/credit-card-transactions>
8. <https://github.com/NotSooShariff/credit-fraud-detection-api>
9. <https://www.kaggle.com/datasets/sriharshaeedala/financial-fraud-detection-dataset>
10. <https://github.com/amazon-science/fraud-dataset-benchmark>
11. <https://www.kaggle.com/datasets/valakhorasani/bank-transaction-dataset-for-fraud-detection>
12. <https://arxiv.org/html/2501.15290v1>
13. <https://www.kaggle.com/datasets/subhajournal/credit-card-fraud-dataset>
14. <https://docs.x.ai/docs/models>
15. <https://www.openml.org/search?type=data&id=43627>
16. <https://www.kaggle.com/datasets/rupakroy/online-payments-fraud-detection-dataset>
17. <https://www.kaggle.com/datasets/kartik2112/fraud-detection>
18. <https://www.kaggle.com/datasets/helgiryewithana/credit-card-fraud-detection-dataset-2023>
19. <https://www.kaggle.com/datasets/ealaxi/paysim1>
20. [https://ludwig.ai/latest/examples/gbm\\_fraud/](https://ludwig.ai/latest/examples/gbm_fraud/)
21. Descricao-da-Atividade-Obrigatoria-2025-09-15-1.pdf
22. [dataset-analyse.md](#)
23. Descricao-da-Atividade-Obrigatoria-2025-09-15-1.pdf
24. [estimativa-custo-api.md](#)
25. <https://community.openai.com/t/how-to-set-billing-limits-and-restrict-model-usage-for-a-project-via-openai-api/1087771>
26. <https://docs.usagepanda.com/usecases/limit-openai-api-requests-by-cost.html>
27. <https://mehmetbaykar.com/posts/setting-openai-api-key-limits-by-project/>
28. <https://www.toriihq.com/articles/how-to-monitor-spending-openai>
29. [https://www.reddit.com/r/AZURE/comments/1bzq1sk/azure\\_openai\\_is\\_it\\_easy\\_to\\_cap\\_api\\_usage\\_by/](https://www.reddit.com/r/AZURE/comments/1bzq1sk/azure_openai_is_it_easy_to_cap_api_usage_by/)
30. <https://platform.openai.com/account/limits>
31. <https://platform.openai.com/settings/organization/billing/overview>
32. <https://support.cmts.jhu.edu/hc/en-us/articles/38383798293133-Guide-to-Managing-API-Keys-and-Usage-Limits-on-platform-openai-com>
33. <https://community.openai.com/t/api-charged-1000-above-spending-hard-limit/722967>
34. <https://platform.openai.com/account/billing/overview>
35. <https://platform.openai.com/docs/guides/production-best-practices/managing-costs>
36. <https://community.openai.com/t/how-to-set-a-price-limit/13086>
37. <https://help.openai.com/en/collections/3943089-account-login-and-billing>

38. <https://www.cloudzero.com/blog/openai-cost-optimization/>
39. <https://community.openai.com/t/how-to-control-the-expenditure-of-a-budget/579500>
40. <https://community.openai.com/t/spending-limits-per-user-or-per-api-key/126955>
41. <https://community.openai.com/t/monthly-budget-limit-silently-removed/1193635>
42. <https://milvus.io/ai-quick-reference/how-can-i-check-my-api-usage-and-limits-with-openai>
43. <https://metronome.com/blog/ai-pricing-and-billing-playbook-the-openai-case-study>
44. <https://platform.openai.com/docs/guides/rate-limits>