**Coleta e Construção da Base de Dados**

**link do projeto GitHub:** <https://github.com/DaniloRamalhoSilva/HP_Cartucho_Pirata_Detector>

**Integrantes do grupo:**

Danilo Ramalho Silva | RM: 555183

Israel Dalcin Alves Diniz | RM: 554668

João Vitor Pires da Silva | RM: 556213

Matheus Hungaro | RM: 555677

Pablo Menezes Barreto | RM: 556389

Tiago Toshio Kumagai Gibo | 556984

**1. Visão geral**

Este entregável descreve um pipeline automatizado para identificar anúncios de cartuchos HP suspeitos no Mercado Livre. O fluxo compreende:

* Coleta de URLs de anúncios;
* Extração de dados de produto e comentários;
* Armazenamento em SQLite;
* Rotulagem heurística (original vs. suspeito) combinando regex e LLM.
* Exportação para .CSV

**2. Coleta de Dados**

**2.1 Ferramentas e decisões**

* **Selenium WebDriver (Chrome)**: necessário para renderizar JavaScript e interagir com botões dinâmicos.
* **Delays e WebDriverWait**: time.sleep (2–4s) e espera explícita garantem carregamento e mitigam bloqueios do ML.
* **Seletores CSS**: identificados inspecionando o DOM do Mercado Livre; mantidos genéricos para resistir a pequenas mudanças de layout.

**2.2 Pipeline de scraping**

|  |  |
| --- | --- |
| **Módulo** | **Responsabilidade** |
| scrap\_list | Navega na listagem, captura e extrai links e páginas. |
| scrap\_product | Abre anúncio, coleta: título, preço, avaliações, descrição e vendedor; salva em products\_data. |
| scrap\_comments | Acessa iframe de comentários, expande com JS, itera até o limite gravando cada review em products\_review. |
| export\_dataset | Gera um arquivo CSV com as principais features para facilitar a análise exploratória e treinamento de modelos de machine learning. |

**Decisões no processo**

* **Carregamento de 3s** antes de extrair detalhes do produto para garantir scripts e recursos externos.
* **Execução de scripts JS** (scrollIntoView, click com setTimeout) para acionar carregamentos adicionais de conteúdo.
* **Tratamento de exceções** em cada etapa para continuar fazendo scraping mesmo com falhas pontuais.

**3. Estrutura do Banco de Dados**

Utilizamos SQLite com três tabelas normalizadas:

* **products\_url** (id, produto, url, scraped, data\_cadastro)
* **products\_data** (id, products\_url\_id, url, title, price, review\_rating, review\_amount, seller, description, positive\_occurrences, negative\_occurrences, label, data\_cadastro)
* **products\_review** (id, products\_data\_id, rating, review, review\_date, data\_cadastro)

**Decisões de modelagem**

* **Tabelas separadas**: isola reviews para análise granular.
* **Colunas de evidência**: pré-calcula contagens de keywords para não sobrecarregar o LLM.

**4. Rotulagem Heurística (Classificação Binária)**

**4.1 Contagem com regex (update\_comment\_counts)**

* **Objetivo**: resumir rapidamente os comentários sem enviar todo o texto ao LLM.
* **Método**: busca padrões em products\_review.review:
* **Negativas**: falso, pirata, não é original, genérico, defeito.
* **Positivas**: produto original, ótima qualidade, recomendo.
* **Decisão**: usar regex case-insensitive para cobertura ampla.
* **Resultado**: popula positive\_occurrences e negative\_occurrences em products\_data.

**4.2 Classificação com LLM (classify\_and\_update)**

* **Prompt few-shot** com exemplos manuais que definem claramente cada classe.
* **Componentes do prompt**: title, price, seller, description, positive\_occurrences, negative\_occurrences.
* **Configurações**: temperature=0 (saída estável), modelo gpt-4o-mini.
* **Fallback**: qualquer resposta fora de original/suspeito é tratada como suspeito.
* **Racional**: combina dados quantitativos (contagens) e qualitativos (texto) para decisão mais robusta, sem necessidade de treinar modelo próprio.