SEAZONE

Autor:
Leonardo Colussi

# Relatório do Desafio do Processo Seletivo

Florianópolis
2021

Leonardo Colussi

# Relatório do Desafio do Processo Seletivo

Relatório do desafio do Processo Seletivo da Seazone sobre scraping e análise de dados.

Florianópolis

2021

# SUMÁRIO

# 1 INTRODUÇÃO

Este relatório visa realizar de maneira objetiva o desenvolvimento e os resultados obtidos no projeto proposto no desafio do Processo Seletivo da empresa Seazone. O projeto consiste em um *scraper* para realização da coleta de dados de anúncios sobre terrenos na região de Florianópolis e ao seu redor. Esses dados serão armazenados em um arquivo e analisados em seguida.

# 2 DESENVOLVIMENTO

Para realizar esse projeto foi utilizada a biblioteca “Beautiful Soup”, “requests_html”, “lxml”, “numpy”, “csv”, “os” e “json”.

## 2.1 Análise do site

O primeiro passo para poder realizar o scraping do site da OLX foi analisar a estrutura do HTML e localizar todas as informações necessárias que precisavam ser extraídas. No caso desse projeto as informações extraídas foram:

- Tipo de categoria (terrenos, sítios ou fazendas);
- Título do anúncio;
- Descrição do anúncio;
- Preço do anúncio;
- Tamanho em metros quadrados;
- Localização do anúncio (CEP, município, bairro, logradouro);
- Nome do anunciante;

Todas essas informações eram contidas em tags com atributos específicos. No caso do projeto em questão, todas as informações foram retiradas através das classes.

## 2.2 Desenvolvimento do Projeto

Na primeira versão do scraper foi feito um código base sem muitas otimizações, com o objetivo de obter resultados primários. Para isso, foi criado a base do link a ser utilizado. O link principal precisa ter todas as informações de localização, categoria a ser requerido e página da lista de anúncios. Essas informações foram colocadas em constantes no começo do código:

```
PREFIXO = 'sc'
REGIAO = 'florianopolis-e-regiao'
CATEGORIA = 'imoveis'
SUBCATEGORIA = 'terrenos'
PAGINAS = 1
```

Constantes utilizadas no código.

Nesse caso, o link estava em um for loop para que sejam pegos todos os links de todos os anúncios até a página requerida pelo utilizador do mesmo.

O resultado obtido foi o seguinte:

```
for pagina in range(1, PAGINAS + 1):
    url = f'https://{PREFIXO}.olx.com.br/{REGIAO}/{CATEGORIA}/{SUBCATEGORIA}?o={pagina}'
    result = self.sessao.get(url, headers=HEADERS)
```

For loop para encontrar todas as listas de links.

Todos esses links são colocados em uma lista através de um for loop para fazer outra requisição em cada link e obter o html do anúncio individual.

```
div = doc.find(class_='classe')
itens = div.find_all('li')
for item in itens:
    try:
        a = item.find('a')
        link = a['href']
    except TypeError:
        pass
```

Código para encontrar o link de cada anúncio.

Para pegar cada informação do anúncio foi utilizado o método de busca através de uma classe definida. Um resultado exemplar foi o seguinte:

```
titulo = anuncio.find(class_='classe1')
return titulo.find(class_='classe2').string
```

Código modelo de como foram retiradas as informações.

Em cada informação foram feitas manutenções para poder obter melhor leitura das mesmas. Além disso, para catalogar somente os terrenos, o que não inclui fazendas e sítios, foi feito um filtro para obter resultados finais mais precisos. Isso pelo fato que a média entre vários valores baixos e um valor muito grande produz uma média distorcida que não se aplica à realidade.

No final todas as informações foram colocadas em um arquivo csv chamado “terrenos.csv” e informações como preço, local e tamanho do terreno foram utilizados para fazer análises como mínimo, máximo e média.

```
with open('terrenos.csv', 'a', encoding='utf-8', newline='') as file:
    writer = csv.writer(file)
    writer.writerow([
        tipo, titulo, preco,
        vendedor, tamanho,
        localizacao, descricao
    ])
```

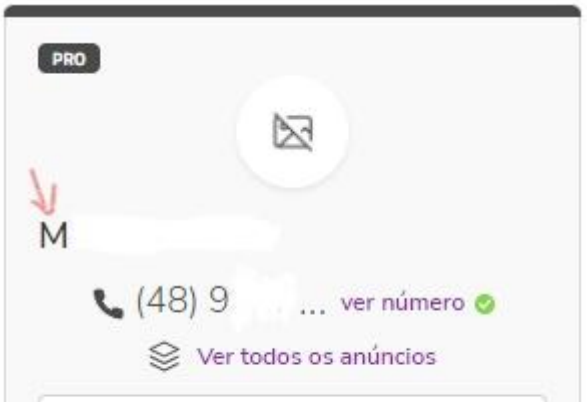
Código usado para a adição do anúncio no arquivo csv.

```
min_tam = np.min(self.tamanhos)
max_tam = np.max(self.tamanhos)
med_tam = np.mean(self.tamanhos)
```

Código modelo utilizado para a análise dos dados.

## 2.3 Problemas encontrados

O maior problema encontrado no desafio foi a obtenção do nome do anunciante, pois na página do anúncio, tal informação é contida em uma parte dinâmica do site web, feita em JS.




Parte dinâmica do anúncio da OLX.

Para resolver esse problema, foram encontradas duas soluções que foram aplicadas no projeto.

### 2.3.1 A renderização

Nessa solução foi implementado o método render para poder renderizar e forçar o aparecimento da parte javascript do código no request da página.

Graças a isso foi possível obter a informação, porém, o tempo de carregamento da página aumentou drasticamente de [0.1,0.5] segundos para [5.0, 8.0] segundos, dificultando ou impossibilitando o scraping do site em larga escala.

```
req_an = session.get(link, headers=HEADERS)
req_an.html.render(timeout=20)
anuncio = bs(req_an.html.html, "lxml")
```

Solução com renderização forçada da página.

### 2.3.2 O “xpath”

Na solução apresentada, que é a atual implementada no código, foi utilizado o xpath que direciona o código, através da biblioteca “lxml” para a informação desejada. Nesse caso existe uma tag dentro do html que, sem a renderização completa da página, tem todas as informações que seriam mostradas após o carregamento da parte dinâmica do código. Sendo assim, foi possível retirar o nome do vendedor com o tempo de scraping anterior, sendo ele de [0.1,0.5] segundos.

A solução é a seguinte apresentada:

```
root = html.fromstring(anuncio.content)
tag = root.xpath('/html/body/script[1]/@data-json')
resultado = tag[0].replace("'","/")
res_dict = json.loads(resultado)
return res_dict['ad']['user']['name']
```

Solução com xpath de redirecionamento à tag script.

## 2.4 Melhorias futuras

Todos os programas podem ser melhorados seja em questão de otimização de performance com código mais eficiente como também em questão de visualização de resultados. No caso do scraper de OLX teriam duas melhorias futuras que podem ser implementadas:

### 2.4.1 Multithread

Um scraper consegue ser melhor quando consegue obter as informações com maior velocidade, sendo assim, uma maneira de conseguir atingir tal objetivo é a implementação da lógica async/await nas funções de requisição de página para que possam ser feitas mais requisições em paralelo, aumentando o número de páginas analisadas em um intervalo de tempo.

### 2.4.2 Gráficos adicionais

Outra implementação interessante que pode ser feita no código é a inserção e utilização de mais gráficos usando os dados já coletados através do scraping. Tal mudança pode aumentar ainda mais as tomada de decisões e conseguir assim obter melhores resultados financeiros, entendendo sempre mais em uma visão macro e micro da área de atuação da empresa.

## 3 FEEDBACK

O desafio proposto foi muito interessante pois consegue instigar a proatividade e a capacitação individual do próprio candidato para conseguir executar a tarefa com todas as especificações citadas. Além disso, norteia o candidato sobre uma visão geral de quais serão os trabalhos realizados dentro da empresa Seazone com um Data Engineer Intern.

## 4 CONCLUSÃO

Durante a execução desse projeto foi possível entender a importância desse ramo da ciência e análise de dados para uma melhor tomada de decisões dentro de uma empresa. No caso da empresa Seazone, entender as tendências dos anúncios dos terrenos para poder realizar melhores negócios.