



escola
britânica de
artes criativas
& tecnologia

Profissão: Analista de dados



PROJETO: ANÁLISE EXPLORATÓRIA DE DADOS EM PYTHON



GUIA DA AULA 4



Transformação e limpeza de dados

- Dados
- Enriquecimento
- Qualidade



Acompanhe aqui
os temas que
serão tratados
na videoaula



1. Dados

O dado bruto é um arquivo do tipo `JSON` com uma lista de instâncias de entregas. Cada instância representa um conjunto de **entregas** que devem ser realizadas pelos **veículos** do **hub** regional. Exemplo:

```

{
  "name": "cvrp-0-df-0",
  "region": "df-0",
  "origin": {
    "lng": -47.802664728268745,
    "lat": -15.657013854445248
  },
  "vehicle_capacity": 180,
  "deliveries": [
    {
      "id": "ed0993f8cc70d998342f38ee827176dc",
      "point": {
        "lng": -47.7496622016347,
        "lat": -15.65879313293694
      },
      "size": 10
    },
    {
      "id": "c7220154adc7a3def8f0b2b8a42677a9",
      "point": {
        "lng": -47.75887552060412,
        "lat": -15.651440380492554
      },
      "size": 10
    },
    ...
  ]
}
  
```



Processamos o dado bruto e construímos o DataFrame Pandas `deliveries_df` através de operações como achatamento (`flatten`) e explosão (`explode`) de colunas:

```
In [ ]: !wget -q << EOF
        https://raw.githubusercontent.com/andre-marcos-perez/ebac-course-
        utils/main/dataset/deliveries.json
        EOF \
        -O deliveries.json
```

```
In [ ]: import json

        import pandas as pd

        # dado bruto em um dict
```



```

with open('deliveries.json', mode='r', encoding='utf8') as file:
    data = json.load(file)
    # dado bruto no pandas

deliveries_df = pd.DataFrame(data)

# coluna origin

hub_origin_df = pd.json_normalize(deliveries_df["origin"])

deliveries_df = pd.merge(left=deliveries_df, right=hub_origin_df,
                          how='inner', left_index=True, right_index=True)
deliveries_df = deliveries_df.drop("origin", axis=1)
deliveries_df = deliveries_df[
    ["name", "region", "lng", "lat", "vehicle_capacity", "deliveries"]
]
deliveries_df.rename(columns={"lng": "hub_lng", "lat": "hub_lat"},
                    inplace=True)
  
```



```

# coluna deliveries
deliveries_exploded_df = deliveries_df[["deliveries"]].explode("deliveries")

deliveries_normalized_df = pd.concat([

    pd.DataFrame(deliveries_exploded_df["deliveries"].apply(
        lambda record: record["size"]
    )),
    pd.DataFrame(deliveries_exploded_df["deliveries"].apply(
        lambda record: record["point"]["lng"]
    )),
    pd.DataFrame(deliveries_exploded_df["deliveries"].apply(
        lambda record: record["point"]["lat"]
    )),
], axis=1)

deliveries_df = deliveries_df.drop("deliveries", axis=1)
deliveries_df = pd.merge(left=deliveries_df, right=deliveries_normalized_df,
                        how='right', left_index=True, right_index=True)
deliveries_df.reset_index(inplace=True, drop=True)

```

```

In [ ]: deliveries_df.head()

```



2. Enriquecimento

- **Geocodificação reversa do hub**

A geocodificação é o processo que transforma uma localização descrita por um texto (endereço, nome do local etc.) em sua respectiva coordenada geográfica (latitude e longitude). A geocodificação reversa faz o oposto, transforma uma coordenada geográfica de um local em suas respectivas descrições textuais.




```

In [ ]:
hub_df = deliveries_df[["region", "hub_lng", "hub_lat"]]
hub_df = hub_df.drop_duplicates().sort_values(by="region").reset_index (
    drop=True
)
hub_df.head()
  
```

Empresas como Google, Bing e Yahoo! fornecem geocodificação como serviço (e cobram por isso). Existe uma projeto *open source* chamado de [OpenStreetMap](#) que mantem um serviço gratuito de geocodificação chamado [Nominatim](#), serviço este que apresenta como limitação a quantia de [uma única consuta por segundo](#). Vamos utilizá-lo através do pacote Python `geopy` para fazer a operação reversa e enriquecer o nosso DataFrame principal.



In []:

```

import json

import geopy
from geopy.geocoders import Nominatim

geolocator = Nominatim(user_agent="ebac_geocoder")
location = geolocator.reverse("-15.657013854445248, -47.802664728268745" )

print(json.dumps(location.raw, indent=2, ensure_ascii=False))

```

Vamos, então, aplicar a geocodificação nas coordenadas das três regiões e extrair informações de **cidade** e **bairro**.

In []:

```

from geopy.extra.rate_limiter import RateLimiter

geocoder = RateLimiter(geolocator.reverse, min_delay_seconds=1)

```



```

In [ ]:
hub_df["coordinates"] = hub_df["hub_lat"].astype(str) + ", "
hub_df["coordinates"] = hub_df["coordinates"] + hub_df["hub_lng"].astype(str)
hub_df["geodata"] = hub_df["coordinates"].apply(geocoder)
hub_df.head()

```

```

In [ ]:
hub_geodata_df = pd.json_normalize(hub_df["geodata"].apply(
    lambda data: data.raw
))
hub_geodata_df.head()

```

```

In [ ]:
import numpy as np

hub_geodata_df = hub_geodata_df[
    ["address.town", "address.suburb", "address.city"]
]

```



```

hub_geodata_df.rename(
    columns={
        "address.town": "hub_town",
        "address.suburb": "hub_suburb",
        "address.city": "hub_city"
    }, inplace=True)
hub_geodata_df["hub_city"] = np.where(
    hub_geodata_df["hub_city"].notna(),
    hub_geodata_df["hub_city"],
    hub_geodata_df["hub_town"]
)
hub_geodata_df["hub_suburb"] = np.where(
    hub_geodata_df["hub_suburb"].notna(),
    hub_geodata_df["hub_suburb"],
    hub_geodata_df["hub_city"])
hub_geodata_df = hub_geodata_df.drop("hub_town", axis=1)
hub_geodata_df.head()

```



O DataFrame `hub_geodata_df` com as informações de **cidade** e **bairro** é, então, combinado ao DataFrame principal `deliveries_df` enriquecendo assim o dado.

```

In [ ]:
hub_df = pd.merge(
    left=hub_df, right=hub_geodata_df, left_index=True, right_index=True
)
hub_df = hub_df[["region", "hub_suburb", "hub_city"]]
hub_df.head()
  
```



```

In [ ]: deliveries_df = pd.merge(
        left=deliveries_df, right=hub_df, how="inner", on="region"
    )
    deliveries_df = deliveries_df[[
        "name",
        "region",
        "hub_lng",
        "hub_lat",
        "hub_city",
        "hub_suburb",
        "vehicle_capacity",
        "delivery_size",
        "delivery_lng",
        "delivery_lat"
    ]]
    deliveries_df.head()
  
```



- **Geocodificação reversa da entrega**

Enquanto o **hub** contem apenas **3** geolocalizações distintas, as **entregas** somam o total de **636.149**, o que levaria em torno de 7 dias para serem consultadas no servidor do Nominatim, dada a restrição de uma consulta por segundo. Contudo, para cargas pesadas como esta, o *software* oferece uma instalação local (na sua própria máquina) que pode ser utilizada sem restrição.

Confira o *link* de instalação local:

<https://nominatim.org/release-docs/latest/admin/Installation/>



Como a instalação do servidor local envolve tecnologias que estão fora do escopo deste curso, como [Docker](#), vamos providenciar estes dados através do *link*:

<https://raw.githubusercontent.com/andre-marcos-perez/ebac-course-utils/main/dataset/deliveries-geodata.csv>

In []:

```

!wget -q << EOF
https://raw.githubusercontent.com/andre-marcos-perez/ebac-
course-utils/main/dataset/deliveries-geodata.csv
EOF \
-O deliveries-geodata.csv
  
```




```

In [ ]: deliveries_geodata_df = pd.read_csv("deliveries-geodata.csv")
        deliveries_geodata_df.head()
  
```

```

In [ ]: deliveries_df = pd.merge(
        left=deliveries_df,
        right=deliveries_geodata_df[["delivery_city",
        "delivery_suburb"]], how="inner",
        left_index=True,
        right_index=True
    )
        deliveries_df.head()
  
```



3. Qualidade

Qualidade do dados está relacionado a consistência do seu `schema` valores faltantes etc.

```
In [ ]: deliveries_df.info()
```

```
In [ ]: deliveries_df.isna().any()
```



• Geocodificação reversa

```
In [ ]: 100 * (
         deliveries_df["delivery_city"].isna().sum() / len(deliveries_df)
       )
```

```
In [ ]: 100 * (
         deliveries_df["delivery_suburb"].isna().sum() / len(deliveries_df)
       )
```

```
In [ ]: prop_df = deliveries_df[["delivery_city"]].value_counts (
         ) / len(deliveries_df)
         prop_df.sort_values (ascending=False).head(10)
```

```
In [ ]: prop_df = deliveries_df[["delivery_suburb"]].value_counts (
         ) / len(deliveries_df)
         prop_df.sort_values (ascending=False).head(10)
```

