INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Brasília
Campus Brasília

TECNOLOGIA EM SISTEMAS PARA INTERNET

Matheus Nícollas de Souza Mota
Thiago Marinho da Silva Campos

RELATÓRIO DE PRÁTICA INTEGRADA
DE
CIÊNCIA DE DADOS E INTELIGÊNCIA ARTIFICIAL

Brasília - DF
18/09/2020

Sumário

1. Objetivos

O objetivo é usar através de consultas as informações provenientes das outras etapas do projeto para criar gráficos analíticos que mostram os quatro estados que possuem maior frequência de relatos, assim como os tipos de óvnis mais populares. Também é objetivo enriquecer os dados e construir mapas com dados de latitude e longitude, com o país inteiro (EUA e plotar as ocorrências para todas as cidades, de forma a construir uma apresentação visual da quantidade de ocorrências por cidade e estado. Também será gerado um mapa somente do estado da Califórnia para verificar como se distribuem as visualizações dentro do estado.

2. Descrição do problema

O projeto consiste em reunir fatos interessantes relacionados a OVINIs, a partir de relatos realizados dentro de um período de vinte anos usando o site Nuforc. O desafio consiste em fazer a extração de dados de forma tabular, afinal, para que os dados possam ser analisados eles acabam se tornando tabelas.

O WebScraping consiste em extrair os dados formatados com tag's da linguagem HTML. Como iremos extrair vinte anos de dados, consultaremos 240 páginas web, uma por cada mês, por vinte anos, entre setembro 1997 e agosto de 2017.

Para extrair conhecimento destes dados é necessário uma exploração dos mesmos, e nesta etapa consiste em usar através de consultas as informações provenientes das outras etapas do projeto para criar gráficos analíticos que mostram os quatro estados que possuem maior frequência de relatos, assim como os tipos de óvnis mais populares, construir mapas com dados de latitude e longitude e plotar as ocorrências de forma a construir uma apresentação visual da quantidade de ocorrências por cidade e estado para verificar como se distribuem as visualizações.

3. Desenvolvimento

O desenvolvimento do algoritmo foi feito na plataforma Google Collab, esta plataforma foi escolhida pois ao iniciar um notebook na mesma, as bibliotecas e dependências do Python são todas da nuvem.

O desenvolvimento foi feito por partes, na primeira saída o resultado é um gráfico de barras com os quatro estados com mais visualizações de óvnis, para desenvolver este gráfico, foi feito um “count” e foi usada a função sort para organizar a exibição por ordem crescente.

A segunda parte do código vai plotar um gráfico com o formato de óvnis mais avistados, foi criado varáveis que vão receber os valores apenas dos dataframes correspondentes ao seu tipo de formato nos relatos, então são declarados vetores vazios que receberão uma sequencia de relatos de cada estado, é criado um vetor com os 4 estados com mais relatos em ordem decrescente, então é criada uma estrutura de repetição que incrementa nos vetores criados anteriormente o número de relatos referente ao estado e formato, depois é declarado 4 variáveis que serão usadas para riar o eixo x com uma separação de 0.20 entre as barras, então usamos o plt.bar para passar as variáveis e plotar as barras. Foi usado o plt.xticks para plotar os nomes de cada estado no eixo x. Também usamos o plt.legend() para colocar a legenda e os labels no gráfico.

No terceiro estágio do projeto, usamos a informações da segunda parte, ou seja foi necessário somente implementar a parte gráfica, a parte de variáveis onde os dados estão continuou sendo usada. Criamos o eixo x ele receberá somente 1 variável como parâmetro e com a mesma separação entre as barras, declaramos vetores que servirão para distanciar a margem inferior das barras, usando a propriedade bottom, para isso funcionar foi feito um for que vai de 0 a 4, para construir o gráfico. Plotamos o nome de cada estado no eixo x e inserimos a legenda e os labels no gráfico.

A quarta etapa é formar os mapas, precisamos encontrar um arquivo csv com os dados de latitude o longitude dos estados e cidades dos EUA pois no site que buscamos os dados iniciais não existem esses dados. Não usamos a biblioteca indicada, pois tivemos erros ao carregar os dados. Solucionamos

utilizando um dataframe e fazendo as funções. Criamos 3 arrays para ler o arquico csv com todas as latitudes e longitudes dos estados, iniciamos os arrays e adicionamos aos arrays todas as cidades, latitudes e longitudes com os mesmos index. Usamos o vetor que foi carregado anteriormente contendo as siglas dos estados dos EUA, filtramos os estados e criamos as funções que retornam latitude e longitude de acordo com o nome da cidade. Iniciamos o array que receberá coordenadas de todas as cidades com relatos. Percebemos que há uma lentidão nesta fase, pois são muitas cidades contidas que precisarão carregar os dados.Depois que se sabe a latitude e longitude usamos a função folium.map para plotar o mapa e o plugin hitmap para plotar o mapa de calor de acordo com os relatos.

No quinto passo, criamos um mapa apenas do estado da Califórnia para analisar se as visualizações se distribuem de forma homogênea dentro do estado. Para isso, utilizamos a mesma lógica passando os parâmetros somente do estado.

4 Código implementado

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import folium
from folium import plugins

df_ovnis = pd.read_csv("ovnis.csv", index_col=[0])

df_relatos = df_ovnis.groupby('State').count()
df_relatos = df_relatos.sort_values(ascending=False, by="Posted")['
Posted']

print("Os quatro Estados com mais relatos")
print(df_relatos.head(4))

#Dataframes que recebem apenas relatos referente aos seu tipo de fo
rmato
light = df_ovnis[df_ovnis['Shape'] == 'Light']
circle = df_ovnis[df_ovnis['Shape'] == 'Circle']
fireball = df_ovnis[df_ovnis['Shape'] == 'Fireball']
```

```
sphere = df_ovnis[df_ovnis['Shape'] == 'Sphere']

#Declara vetores vazios que receberao uma sequencia de relatos de c
ada Estado
estados_light = []
estados_circle = []
estados_fireball = []
estados_sphere = []

#Vetor com os 4 estados com mais relatos em ordem decrescente
states = ['CA', 'FL', 'WA', 'TX']

#incrementa nos vetores criados anteriormente o numero de relatos r
eferente ao Estado e formato
for state in states:
  estados_light.append(light[light['State'] == state].count()['Post
ed'])
  estados_circle.append(circle[circle['State'] == state].count()['P
osted'])
  estados_fireball.append(fireball[fireball['State'] == state].coun
t()['Posted'])
  estados_sphere.append(sphere[sphere['State'] == state].count()['P
osted'])

# Cria eixo x com uma separação de 0.20 entre as barras
x1 =  np.arange(len(estados_light))
x2 = [x + 0.20 for x in x1]
x3 = [x + 0.20 for x in x2]
x4 = [x + 0.20 for x in x3]

#Plota as barras
plt.bar(x1, estados_light, width=0.20, label = 'Light')
plt.bar(x2, estados_circle, width=0.20, label = 'Circle')
plt.bar(x3, estados_fireball, width=0.20, label = 'Fireball')
plt.bar(x4, estados_sphere, width=0.20, label = 'Sphere')

#Plota o nome de cada Estado no eixo X
plt.xticks([x + 0.30 for x in range(len(estados_light))], states)

# insere uma legenda no gráfico
plt.legend()

plt.title("Grupo de Barras")
plt.xlabel('State')
plt.ylabel('Views')
plt.show()

# Cria eixo x com a mesma separação entre as barras
x =  np.arange(len(estados_light))
```

```
#Declara vetores que sevirao para distanciar a margem inferior das
barras
fireball_bottom = []
sphere_bottom = []
for i in range(0,4):
  fireball_soma = int(estados_light[i]) + int(estados_circle[i]) #s
oma a primeira barra com a barra de baixo
  fireball_bottom.append(fireball_soma) #incrementa soma no vetor

  sphere_soma = int(estados_light[i]) + int(estados_circle[i]) + in
t(estados_fireball[i]) #soma as duas primeiras barras com a de baix
o
  sphere_bottom.append(sphere_soma) #incrementa soma no vetor

#Plota as barras
plt.bar(x, estados_light, width=0.50, label = 'Light')
plt.bar(x, estados_circle, width=0.50, label = 'Circle', bottom=est
ados_light) #Plota a barra com a margem inferior referente a ultima
 barra
plt.bar(x, estados_fireball, width=0.50, label = 'Fireball', bottom
=fireball_bottom)
plt.bar(x, estados_sphere, width=0.50, label = 'Sphere', bottom=sph
ere_bottom)

#Plota o nome de cada Estado no eixo X
plt.xticks([x for x in range(len(estados_light))], states)

# insere uma legenda no gráfico
plt.legend()

plt.title("Grupo de Barras Empilhadas")
plt.xlabel('State')
plt.ylabel('Views')
plt.show()

#lê o  arquivo csv com todas latitudes e longitudes dos estados
df_longlat = pd.read_csv("uscities.csv")

#inicias os arrays
cidades = []
lats = []
lngs = []

#adicionas aos arrays todas cidades, latitudes e longitudes com os
mesmos index
for i, row in df_longlat.iterrows():
  cidades.append(row["city"])
  lats.append(row["lat"])
  lngs.append(row["lng"])
```

```
print("Carregou todas as cidades, longitudes e latitudes")

#Array contendo as siglas de  todos os estados dos EUA
states = ["AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DC", "DE", "FL
", "GA",
          "HI", "ID", "IL", "IN", "IA", "KS", "KY", "LA", "ME", "MD
",
          "MA", "MI", "MN", "MS", "MO", "MT", "NE", "NV", "NH", "NJ
",
          "NM", "NY", "NC", "ND", "OH", "OK", "OR", "PA", "RI", "SC
",
          "SD", "TN", "TX", "UT", "VT", "VA", "WA", "WV", "WI", "WY
"]

selection = df_ovnis['State'].isin(states)
#Dataframe ja filtrado com estados so dos EUA
df_eua = df_ovnis[selection]

#funcao que retorna a latitude de acordo com o nome da cidade
def getLatitude(city):
  latitude = 0
  for i in range(0, len(cidades)):
    if(city == cidades[i]):
      latitude = lats[i]
  return latitude

#funcao que retorna a longitude de acordo com o nome da cidade
def getLongitude(city):
  longitude = 0
  for i in range(0, len(cidades)):
    if(city == cidades[i]):
      longitude = lngs[i]
  return longitude

#inicia array que recebera coordenas de todas as cidades com relato
s
coordenadas = []
for i in range(0, len(df_eua)):
  coordenadas.append([getLatitude(df_eua.iloc[i].City), getLongitud
e(df_eua.iloc[i].City)])

print("Dataframe Carregado!")
#ESSE PROCESSO PODE DEMORAR MUITO, POIS SÃO MUITAS CIDADES

# Plotando mapa dos EUA:
mapa = folium.Map(
    location=[37.7550585, -106.2454855], #coordenas dos EUA
    tiles='Stamen Terrain',
    zoom_start = 6
)
```

```
#adiciona o plugin de calor pra plotar o mapa com relatos
mapa.add_child(plugins.HeatMap(coordenadas))
mapa

print("5. A próxima etapa é criar um mapa apenas do estado da Calif
órnia, para analisar se essas visualizações se distribuem de forma
homogênea dentro do estado.")

#inicias os arrays
cidades_ca = []
lats_ca = []
lngs_ca = []

for i, row in df_longlat.iterrows():
  if(row["state_id"] == "CA"):
    cidades_ca.append(row["city"])
    lats_ca.append(row["lat"])
    lngs_ca.append(row["lng"])

#funcao que retorna a latitude de acordo com o nome da cidade
def getLatitudeCA(city):
  latitude = 0
  for i in range(0, len(cidades_ca)):
    if(city == cidades_ca[i]):
      latitude = lats_ca[i]
  return latitude

#funcao que retorna a longitude de acordo com o nome da cidade
def getLongitudeCA(city):
  longitude = 0
  for i in range(0, len(cidades_ca)):
    if(city == cidades_ca[i]):
      longitude = lngs_ca[i]
  return longitude

#plotando mapa da california
mapa_california = folium.Map(
    location=[34.0194, -118.411], #coordenas de Los Angeles
    tiles='Stamen Terrain',
    zoom_start = 6
)

df_california = df_eua[df_eua['State']=='CA']

coordenadas_california = []
for i in range(0, len(df_california)):
  coordenadas_california.append([getLatitudeCA(df_california.iloc[i
].City), getLongitudeCA(df_california.iloc[i].City)])
```

```
mapa_california.add_child(plugins.HeatMap(coordenadas_california))
mapa_california
```

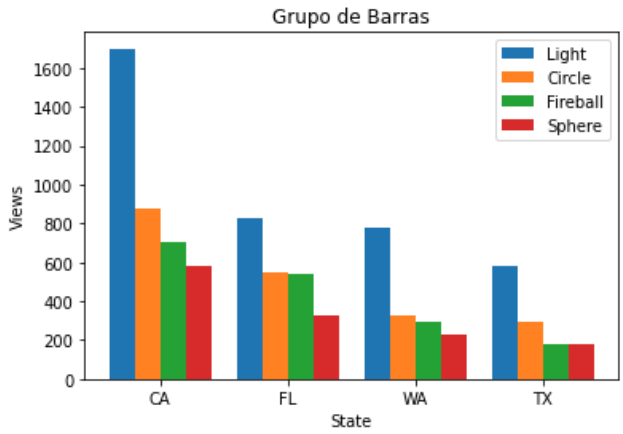
```
print("6. Onde na Califórnia está localizada a maior quantidade de visualizações de objetos voadores não identificados? E qual será a razão?")
print("As áreas da California que mais têm relatos é Los Angeles e São Francisco, muito provavelmente porque é um Estado liberal onde existe fácil acesso a entorpecentes.")
```

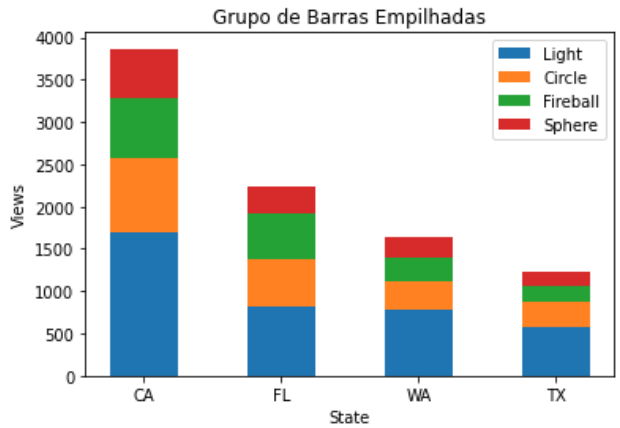
Github: https://github.com/Prof-Fabio-Henrique/pratica-integrada-icd-e-iia-2020-1-g13-mmt

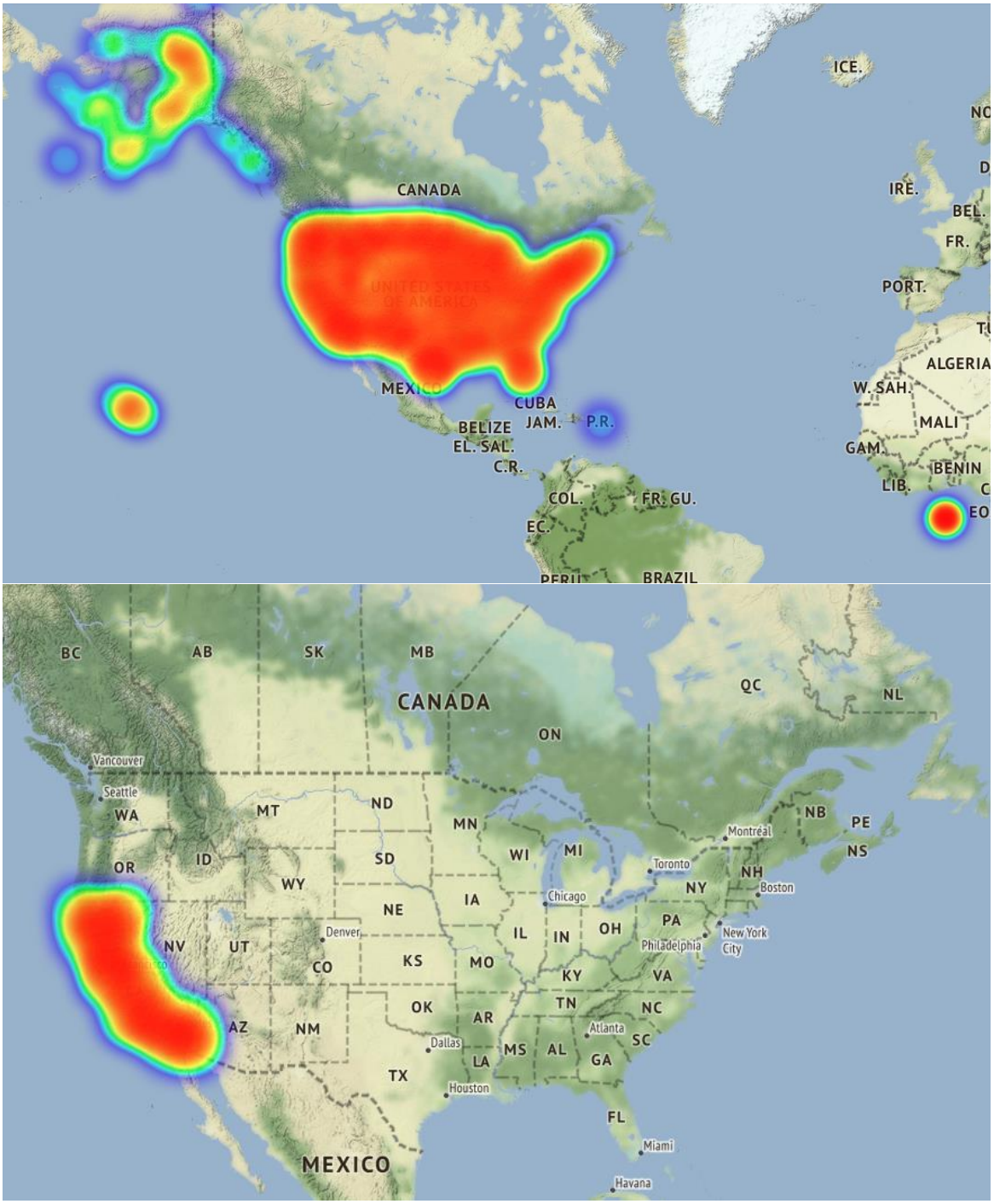
5. Considerações Finais

Com este projeto foi possível responder a uma série de questões com embasamento no webscrapping feito no projeto anterior, criando modelos de visualização de dados em forma gráfica e geoespacial, evidenciando fatos e tendências obtidas através de análise de dados, abaixo seguem as imagens dos resultados e respostas obtidas.

```
Os quatro Estados com mais relatos
State
CA    7911
FL    4352
WA    3225
TX    2882
Name: Posted, dtype: int64
```






ICE.
CANADA
IRE.
BEL.
FR.
PORT.
ALGERIA
W. SAH.
MALI
GAM.
BENIN
LIB.
MEXICO
CUBA
JAM.
P.R.
BELIZE
EL. SAL.
C.R.
COL.
FR. GU.
EC.
BRAZIL
BC
AB
SK
MB
CANADA
QC
NL
ON
Vancouver
Seattle
WA
MT
ND
MN
NB
PE
NS
Montréal
OR
ID
SD
WI
MI
Toronto
NH
WY
NY
Boston
Chicago
IA
NE
Denver
IL
IN
OH
PA
New York City
Philadelphia
NV
UT
CO
KS
MO
KY
VA
TN
NC
OK
AR
Atlanta
SC
AZ
NM
Dallas
MS
AL
GA
LA
TX
Houston
FL
Miami
MEXICO
Havana

6. Referências

PANDAS. **Https://pandas.pydata.org/docs/**. Disponível em: https://pandas.pydata.org/docs/. Acesso em: 19 set. 2020.

REQUESTS. **Requests: HTTP for Humans™**. Disponível em: https://requests.readthedocs.io/en/master/. Acesso em: 19 set. 2020.

SOUP, Beautiful. **Beautiful Soup Documentation**. Disponível em: https://www.crummy.com/software/BeautifulSoup/bs4/doc/. Acesso em: 19 set. 2020.

SILVA, Réulison. **Como fazer um Web Scraping com Python**. Disponível em: https://goomore.com/blog/web-scraping-python/. Acesso em: 19 set. 2020.