# Projeto de prática Integrada de ciência de dados, inteligência artificial e machine learning

# 

## 

## Relatório de avistamento de Objetos Voadores Não Identificados.

## Sprint 2 → Exploração com Gráficos e Mapas

Curso: Tecnologia em sistemas para internet

Estudantes:

Brenda Lopes Miranda Teixeira

Mateus Gomes da Silva Fonteles

Rickson Queiroz Marques de Souza

Samuel Araújo Lopes

Professores

Fábio Henrique

Diego Queiroz

Ana Régia

Brasília, agosto de 2021

## Sumário

[1. Objetivos](#_10388dqmfo5m) 3

[2. Descrição do problema](#_5ux8aubh88f4) 4

[3. Desenvolvimento](#_wkz1qes89g9u) 5

[3.1. Código implementado - Gráficos](#_q0vrp2dio7y5) 6

[3.1. Código implementado - Mapas](#_85367nbp8kvh) 12

[4. Considerações Finais](#_t4rgjanj1qvz) 14

[Referências](#_wvqxb1nnry4r) 15

## 1. Objetivos

Nesta etapa será realizada uma exploração dos dados com uso de gráficos, com algúns objetivos específicos, sendo eles:

* Indicar os Estados com maior frequência de relatos.
* Indicar as formas mais comuns dos objetos relatados.
* Utilizar gráficos de barras agrupadas e empilhadas
* Utilizar mapas
* Plotar um para dos Estados Unidos para representar a quantidade de relatos por Estado e por Cidade.
* Plorar um mapa para detalhar o Estado da Califórnia.

## 2. Descrição do problema

Continuando o trabalho iniciado na primeira *sprint*, vamos explorar o nosso *dataset* fazendo o uso de recursos gráficos disponíveis pelo Python para obter *insights* da informação contida dos dados coletados.

O uso de recursos gráficos na análise de dados permite ao analista visualizar uma grande quantidade de informações de maneira rápida e clara.

O uso de mapas é de especial importância para este tema, pois será assim possível avaliar as regiões que possam ser compostas por mais de um Estado ou Município onde existem zonas com maiores ou menores quantidades de avistamentos.

Os Gráficos de barras nos auxiliam a absorver a informação em um contexto macro, facilitando a compreensão de grandes bancos de dados, que dificilmente poderiam ser absorvidos de uma em uma instância.

## 3. Desenvolvimento

Este trabalho está sendo desenvolvido usando um Script Python por ser uma linguagem orientada a objetos é bastante maleável, o grupo está utilizando a plataforma Google Colaboratory, assim todos podem modificar e acrescentar o código quando necessário.

## 3.1. Código implementado - Gráficos

* Primeiramente são importadas as bibliotecas necessárias:
* Aqui podemos ressaltar a importância das bibliotecas matplotlib, para a criação de gráficos e para a folium, que permite o uso do mapa.

#importações para o código

!pip install -U pandasql

!pip install folium

import pandas as pd

import seaborn as sns

import pandasql

import numpy as np

import matplotlib.pyplot as plt

import folium

from folium.plugins import HeatMap

* Logo foi carregado o arquivo OVNIS.csv;

baseOvnis = pd.read\_csv("OVNIS.csv")

* Logo, serão apresentados os 4 Estados com maior número de relatos.
* Isso será realizado ordenando os **relatos por estado**, em ordem decrescente de **números de relatos** por Estado, logo apresentando somente as 4 primeiras linhas;

#Estados

estados = baseOvnis['State'].value\_counts().head(4)

#Criando um dataFrame

df\_relatos = pd.DataFrame(estados)

#Criando a coluna dos estado da indice

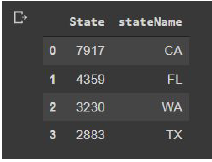
df\_relatos['stateName'] = df\_relatos.index

#retornando o indice

df\_relatos.reset\_index(drop=True, inplace=True)

df\_relatos

* O resultado desta operação é o seguinte:



* Logo, será realizado o processo correspondente para apresentar as formas OVNIs mais relatadas;

#Ovnis mais populares dos relatos

tipos = baseOvnis['Shape'].value\_counts().head(4)

#Criando um dataFrame

df\_ovnis = pd.DataFrame(tipos)

#Criando a coluna dos Ovnis com a indice

df\_ovnis['shapesName'] = df\_ovnis.index

#mudando o nome

df\_ovnis.columns = ['Qtd', 'shapesName']

#retornando o indice

df\_ovnis.reset\_index(drop=True, inplace=True)

df\_ovnis

* O resultado desta operação é o seguinte:



* Em seguida, se aplica uma pesquisa SQL para filtrar Estados por Tipos

q= """

SELECT State, Shape, COUNT(\*) AS VIEWS

FROM baseOvnis

WHERE State IN('CA','FL','WA','TX') AND Shape IN('Light','Circle','Triangle','Fireball')

GROUP BY State, Shape

ORDER BY VIEWS DESC

"""

* Logo, separamos o registro de Shapes mais populares (com mais de 1000 ocorrências):

n\_shape = df1['Shape'].value\_counts()

maisQueMil = n\_shape[n\_shape > 1000]

maisQueMil

* Logo agrupamos e filtramos os dados.

#Filtro de dados, agrupar depois filtrar

q = """

SELECT \*

FROM df1

WHERE Shape in ('Light', 'Circle', 'Triangle', 'Fireball', 'Sphere', 'Other', 'Oval', 'Disk', 'Formation', 'Changing', 'Cigar, 'Flash', 'Rectangle')

"""

Tabela1 = pandasql.sqldf(q, locals())

pd.DataFrame(Tabela1)

* Logo é iniciada a confecção do primeiro gráfico de barras, representando as formas avistadas por Estados.

# Gerando o primeiro Gráfico

Light = [1701, 826, 780, 579]

Circle = [881, 551, 330, 290]

Fireball = [703, 541, 296, 183]

Triangle = [640, 344, 301, 245]

barWidth = 0.15

plt.figure(figsize=(7,5))

r1 = np.arange(len(Light))

r2 = [x + barWidth for x in r1]

r3 = [x + barWidth for x in r2]

r4 = [x + barWidth for x in r3]

plt.bar(r1, Light, color='gold', width=barWidth, label='Light')

plt.bar(r2, Circle, color='grey', width=barWidth, label='Circle')

plt.bar(r3, Triangle , color='saddlebrown', width=barWidth, label='Fireball')

plt.bar(r4, Fireball, color='blue', width=barWidth, label='Triangle')

* Para dar maior claridade ao mapa, são incluídas as legendas no mesmo.

# inserindo legendas

plt.xlabel('State')

plt.xticks([r + barWidth for r in range(len(Light))], ['CA', 'WA', 'FL', 'TX'])

plt.ylabel('Views')

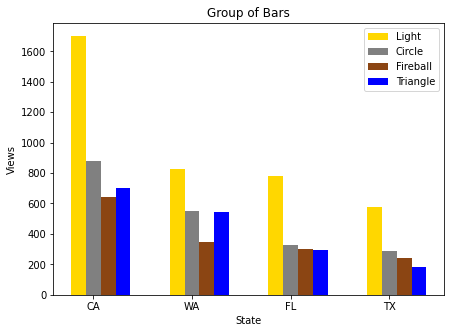
plt.title('Group of Bars')

# gerando a legenda e criando o grafico

plt.legend()

plt.show()

* A seguir, o gráfico gerado:



* Logo, é gerado um segundo gráfico de barras, de estilo distinto, apresentando as mesmas informações de outra forma:

# Gerando o Segundo Gráfico

Light = np.array((1701, 826, 780, 579))

Circle = np.array((881, 551, 330, 290))

Fireball = np.array((703, 541, 296, 183))

Triangle = np.array((640, 344, 301, 245))

shape = ['Light','Circle','Fireball', 'Triangle']

states = ['CA','WA','FL', 'TX']

plt.figure(figsize=(7,5))

plt.bar(states, Light, color = 'gold')

plt.bar(states, Circle, color = 'grey', bottom = Light)

plt.bar(states, Fireball, color = 'saddlebrown', bottom = Light + Circle)

plt.bar(states, Triangle, color = 'blue', bottom = Light + Circle + Fireball)

# inserindo legendas

plt.xlabel('State')

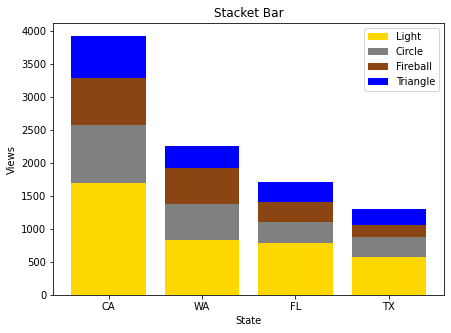
plt.ylabel('Views')

plt.title('Stacket Bar')

plt.legend(('Light', 'Circle', 'Fireball', 'Triangle'))

plt.show()

* A seguir, o gráfico gerado:



## 3.1. Código implementado - Mapas

* Gerando o mapa da Califórnia:

#Tipos diferentes de mapas

folium.Map(

location=[-19.916667,-43.933333],

tiles='Stamen Toner',

)

folium.Map(

location=[-19.916667,-43.933333],

tiles='Stamen Terrain',

)

#Gerando o Mapa CA

CAlat = 37.2502200

CAlon = -119.7512600

WAlat = 38.904

WAlon = -77.0171

FLlat = 28.4159

FLlon = -81.2988

TXlat = 29.3838500

TXlon = -94.9027000

#mapa = folium.Map(location=[CAlat, CAlon])

#mapa = folium.Map(location=[WAlat, WAlon])

#mapa = folium.Map(location=[FLlat, FLlon])

mapa = folium.Map(location=[TXlat, TXlon])

mapa

* Foi também instalada a biblioteca ZipCode, que permite obter informações de localização por meio do código postal:

#ZIPCODES ou cep

!pip install zipcodes

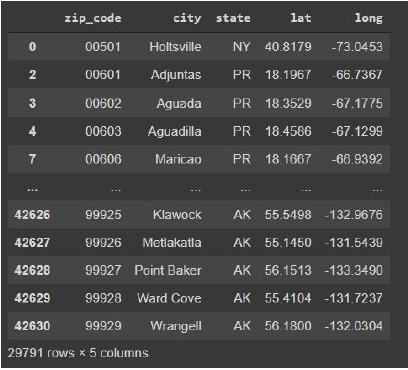
#importando a biblioteca zipcode

import zipcodes

zipcodes\_json = zipcodes.list\_all()

df\_zipcodes = pd.DataFrame(zipcodes\_json)

df\_zipcodes



## 4. Considerações Finais

A criação de gráficos e mapas faz parte da exploração dos dados coletados e é uma das mais poderosas ferramentas para auxiliar especialistas e leigos na interpretação das informações contidas em bancos de dados.

## Referências

* Silveira. Guilherme . Select count(\*), count(1) e count(nome): a batalha dos counts de SQL. 2017.
  + Disponível em: <https://www.alura.com.br/artigos/select-count-count1-e-countnome-a-batalha-dos-counts-de-sql >
* Teixeira. Douglas. Manipulação de Listas em Python.
  + Disponível em: <https://algoritmosempython.com.br/cursos/programacao-python/listas/>