INSTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Brasília
Campus Brasília

TECNOLOGIA EM SISTEMAS PARA INTERNET

Daniel Evangelista Pereira
Leonardo Campos Muniz

RELATÓRIO DE PRÁTICA INTEGRADA
DE
CIÊNCIA DE DADOS E APRENDIZADO DE MÁQUINA

Brasília - DF

13/03/2021

Sumário

1. Objetivos

Este é o primeiro relatório de atividades da PRÁTICA INTEGRADA DE CIÊNCIA DE DADOS E APRENDIZADO DE MÁQUINA. O desafio envolve um time misto de discentes das disciplinas optativas de Introdução à Ciência de Dados - ministrada no quinto semestre - e de Aprendizado de Máquina - ministrada no quarto semestre - no âmbito do curso de Tecnologia em Sistemas para Internet do campus Brasília do Instituto Federal. Os docentes são, respectivamente, o Professor Doutor Fábio Henrique de Oliveira e o Professor Doutor Diego Queiroz.

A equipe é composta pelos alunos Wendell Rodrigues Feliciano, Leonardo Campos Muniz e Daniel Evangelista Pereira, com a mentoria de Thiago Marinho e utilizará a metodologia de desenvolvimento ágil SCRUM, com um total de 4 sprints de uma semana.

Este documento inaugural, portanto, trata do desafio proposto para a primeira semana: **Exploração com Gráficos e Mapas**. Aproveitando o desafio da semana passada onde envolvia realizar a raspagem de dados em um *site* que tem o foco no tema de “Extraterrestre”, foi realizada a estruturação desses dados por meio de gráficos e mapas.

2. Descrição do problema

De forma clara o problema a ser resolvido é o seguinte, mostrar em forma de gráficos quais são os quatro estados onde os relatos de OVNIs são mais populares e organizar os gráficos nos seguintes parâmetros: o primeiro seria em barras agrupadas, o segundo em barras empilhadas e por formato do OVNI.

Seguindo pra segunda parte do desafio é conseguir obter a latitude e longitude do pontos onde as ocorrências aconteceram, criar um mapa do país onde aconteceu (no caso o EUA) e plotar nele as ocorrências de todas as cidades. O próximo passo é criar um mapa apenas do estado da Califórnia para poder fazer a análise e descobrir se os casos ocorrem de forma homogênea dentro do estado.

3. Desenvolvimento

Para facilitar o desenvolvimento do código e da criação dos gráficos, foram utilizadas as seguintes bibliotecas:

- Pandas: Biblioteca utilizada para fazer a carga de dados e a transformação da mesma em forma de tabela;
- Matplotlib: Biblioteca utilizada para poder fazer a plotagem de informações no mapa;
- Numpy: Biblioteca utilizada para realização e elaboração de operações matemáticas complexas e manipulação de array;
- Zipcodes: Biblioteca utilizada com o intuito de unir informações e dados de dois códigos em um único;
- Folium: Biblioteca utilizada neste projeto com o intuito de criar o mapa dos EUA neste projeto;

3.1 Código implementado

```
# -*- coding: utf-8 -*-
import pandas as pd
import matplotlib.pyplot as plt
import zipcodes
import numpy as np
import folium
from folium.plugins import HeatMap



# peaga retorna uma lista conforme o agrupamento
def  filter_ocorrencias_by_column_dataframe(dataframe,  column,  count=4,
ascending=True):
    filter = dataframe.groupby([column]).size().reset_index(name='counts')
                        filter      =      filter.sort_values(by='counts',
ascending=ascending).head(count)
    filter = filter[column].values
    return filter



# metodo que cria e salva os graficos
```

```
def create_grafico_barras(dataframe, set_ylabel="", file="", stacked=False):
    ax = dataframe.plot.bar(rot=0, stacked=stacked)
    ax.set_ylabel(set_ylabel)

    if file:
        fig = ax.get_figure()
        fig.savefig(file)
    return ax


def plotar_grafico(ovnis):
    # pesquisando os quatro estados com maior incidencia de apariçoes
        states = filter_ocorrencias_by_column_dataframe(dataframe=ovnis, column='State', ascending=False)

    # filtrando os estados com maiores aparicoes
    filtred = ovnis[ovnis.State.isin(states)]

     # pesquisando os quatro shapes com maior incidencia de apariçoes nos estados
       shapes = filter_ocorrencias_by_column_dataframe(dataframe=filtred, column='Shape', ascending=False)

    # filtrando shapes com maior numero de aparicoes por estado
    filtred = filtred[filtred.Shape.isin(shapes)]

    # agrupando os shanpes por estado
                            grouped = filtred.groupby(['State', 'Shape']).size().reset_index(name='counts')

    # invertendo as colunas para mostrar shapes por estado
    grouped = grouped.pivot_table('counts', ['State'], 'Shape')

    print(grouped)

        grafico1 = create_grafico_barras(grouped, set_ylabel='Views', file='ocorrencias_barras_agrupadas.jpg')
        grafico2 = create_grafico_barras(grouped, set_ylabel='Views', file='ocorrencias_barras_empilhadas.jpg',
                                         stacked=True)

    plt.show()
```

```
def buscar_latitude_longitude(dataframe):
    dataframe.City = dataframe.City.str.strip()
    dataframe.State = dataframe.State.str.strip()

    lat_long = {
        "lat": [],
        "long": []
    }

    for i, row in dataframe.iterrows():
        city = row.City
        state = row.State

        cities = zipcodes.filter_by(city=city, state=state)

        if cities:
            lat_long["lat"].append(cities[0]['lat'])
            lat_long["long"].append(cities[0]['long'])
            continue

        lat_long["lat"].append(np.nan)
        lat_long["long"].append(np.nan)

    df = pd.DataFrame(lat_long)
    df.to_csv('lat_long.csv', index=False)

    return df


def visualizar_todas_ocorrencias_no_mapa(dataframe):
    extra_terrestre = dataframe
     extra_terrestre.columns = [col.strip().replace(' ', '').replace(r'/',
'') for col in extra_terrestre.columns]

    extra_terrestre.City = extra_terrestre.City.str.strip()
    extra_terrestre.State = extra_terrestre.State.str.strip()

    data_frame = extra_terrestre.groupby(['City', 'State', 'lat', 'long'])

    lat_long = data_frame[['lat', 'long']].fillna(0)
```

```
                BBox    =    [[lat_long.lat.max(),    lat_long.long.max()],
[lat_long.lat.min(), lat_long.long.min()]]
    np_array = lat_long.to_numpy().tolist()

    mapa = folium.Map(location=[0, 0], zoom_start=13)
    mapa.fit_bounds(BBox)
    HeatMap(np_array, radius=10).add_to(mapa)
    mapa.save("todas_as_ocorrencias.html")


def buscar_e_salvar_latitude_longitude_das_ocorrencias(ovnis):
    ovnis.City = ovnis.City.str.strip()
    ovnis.State = ovnis.State.str.strip()

    lat_long = buscar_latitude_longitude(ovnis)

    ovnis[['lat', 'long']] = lat_long

    print(ovnis)

    ovnis.to_csv('ovnis_with_lat_long.csv', index=False)


def visualizar_ocorrencias_no_mapa_por_estado(dataframe, estado):
    estado_dataframe = dataframe[dataframe['State'] == estado]

    lat_long = estado_dataframe[['lat', 'long']].fillna(0)

                BBox    =    [[lat_long.lat.max(),    lat_long.long.max()],
[lat_long.lat.min(), lat_long.long.min()]]
    np_array = lat_long.to_numpy().tolist()

    mapa = folium.Map(location=[0, 0], zoom_start=13)
    mapa.fit_bounds(BBox)
    HeatMap(np_array, radius=10).add_to(mapa)
    mapa.save("ocorrencias_por_estado.html")


def main():
    ovnis = pd.read_csv('OVNIS.csv')
     ovnis.columns  =  [col.strip().replace('  ', '').replace(r'/',  '')  for
col in ovnis.columns]
```

```
    plotar_grafico(ovnis)

    buscar_e_salvar_latitude_longitude_das_ocorrencias(ovnis)

    extra_terrestre = pd.read_csv('ovnis_with_lat_long2.csv')

    visualizar_todas_ocorrencias_no_mapa(extra_terrestre)

    extra_terrestre.City = extra_terrestre.City.str.strip()
    extra_terrestre.State = extra_terrestre.State.str.strip()

        data_frame  =  extra_terrestre.groupby(['City',  'State',  'lat',
'long']).size().reset_index(name='counts')

    visualizar_ocorrencias_no_mapa_por_estado(data_frame, 'CA')


if __name__ == '__main__':
    main()
```

4. Considerações Finais

Bom para começar a lógica para criar os dois primeiros gráficos em barras e em barras agrupadas, teríamos que filtrar os dados com base no módulo anterior, precisaríamos agrupar as ocorrências por estado e contá-las, por estado e por shape. A primeira dúvida que surgiu foi em relação à filtragem, se agrupamos as ocorrências por estado e os 4 estados com mais ocorrências, e depois agrupamos todas as ocorrências por shape, e filtramos primeiro os dados por esses dois grupos. Porém optamos por filtrar os dados pelos estados com mais ocorrências e os shapes que mais aparecem nesses estados. Feito esse processo utilizamos a própria função do pandas que já possui integração com o matplotlib e criamos os gráficos e salvamos no formato png.

A segunda parte do trabalho acreditamos que foi a parte mais complicada, mas primeiro passo foi remover os espaços duplos e os e os espaços no início e no final de cada Cidade, após isso pegamos os dados com a biblioteca zipcodes filtrando por estado e cidade, para pegar a latitude e longitude, a pesquisa devolve uma lista e pegamos a primeira ocorrência, as cidades não encontradas preenchemos com NaN da biblioteca numpy, criando um dataframe com a mesma quantidade de linhas do arquivo original e unimos esse dataframe ao um novo e salvamos em um arquivo csv.

Após a obtenção da latitude e longitude utilizamos a biblioteca folium para criar um mapa de calor das ocorrências, primeiro precisávamos construir a caixa de zoom com os Estados unidos em foco, pegando a latitude e longitude máxima e a latitude e longitude mínima, após isso criamos um dataframe apenas com colunas de latitude e longitude e passamos para o plugin HeatMap da biblioteca folium que criou o mapa com a lista. Para não haver qualquer erro as colunas com NaN tiveram seu valor alterado para 0 um valor numérico inteiro.

Bom não temos a certeza se fizemos a contagem da forma mais correta possível, porém o uso das bibliotecas facilitou a criação dos mapas.

Referências

Aqui vocês podem colocar quaisquer referências externas que tenham utilizado (Sugiro colocar na ABNT, porque a falta de padrão dificulta a leitura).