INSTITUTO FEDERAL
Brasília
Campus Brasília

TECNOLOGIA EM SISTEMAS PARA INTERNET

Kálita Rodrigues de Souza
Lucas Nascimento Verdam de Araújo
Nikolle de Lacerda Nascimento
Pedro Henrique Fernandes de Santana

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

Brasília - DF

12/01/2022

Sumário

1. Objetivos

Os objetivos dessa sprint são a introdução a temática, a coleta, análise, preparação e exploração dos dados obtidos pelo sistema de uma pulseira que usa coordenadas para identificar padrões de movimentos e fazer predições relacionadas a acidentes domésticos a fim de auxiliar pessoas idosas a serem socorridas.

2. Descrição do problema

A problemática possui as seguintes principais etapas:

- Baixar o arquivo compactado que contém os dados;
- Fazer a coleta dos dados desejados;
- Fazer a exploração dos dados a fim de obter o máximo de informações úteis;
- Fazer a preparação dos dados para otimizar as etapas de exploração e análise;

3. Desenvolvimento

As principais tecnologias utilizadas no desenvolvimento desta tarefa foram:

Python

Python é uma linguagem de programação de alto nível, do termo em inglês, *high level language*. Estruturas desse tipo são, geralmente, classificadas como orientadas a objetos. (COUTINHO, 2019). De acordo com Roveda (2021), ao trabalhar com ciência de dados, desenvolver em Python pode ser uma alternativa, pois o crescente número de bibliotecas disponíveis em Python voltadas à análise de dados oferece funções e métodos de otimização para praticamente quaisquer objetivos.

Google Colaboratory

O Google Colaboratory, ou Colab, é um serviço de nuvem gratuito hospedado pelo Google para incentivar a pesquisa de Aprendizado de Máquina e Inteligência Artificial (SANTOS, 2020). A ferramenta funciona com uma estruturação em células. Os códigos rodados dentro dessas células podem ser rodados separadamente ou em conjunto, permitindo uma programação mais dinâmica (NOLETO, 2020). A plataforma roda linguagens como Python e Jupyter.

Github

É uma plataforma online de trabalho colaborativo. A primeira parte do nome, “Git”, é por causa da utilização do sistema de controle de versão e a segunda parte, “Hub”, tem a ver com a conexão entre profissionais de programação de qualquer lugar do mundo. (SOUZA, 2020).

Criamos um notebook para o projeto na plataforma Google Colab, pois ela possibilita o upload dos arquivos de dados bem como o desenvolvimento compartilhado dos códigos necessários para se realizar as tarefas solicitadas. Neste notebook, para a codificação dos scripts necessários para coleta, preparação e exploração dos dados utilizou-se a linguagem Python por ser uma das mais apropriadas e usadas para tarefas relacionadas à ciência de dados. Além é claro, dos módulos e bibliotecas necessários para tal, sendo eles: a biblioteca Pandas e o módulo ‘os’ do python para a coleta e preparação; a biblioteca Matplotlib e PandasSQL para a exploração.

3.1 Código implementado

1.1 - Introdução e coleta de dados

Depois de extrair os arquivos, a coleta será iniciada de fato, inserindo o conteúdo de cada arquivo em um único *Data Frame*.

Importar o *pandas* e colocar o nome dele como "pd" e módulo "os"

```
import pandas as pd
import os
```

Fazer um *array* com os nomes das pastas de cada movimento:

```
nome_movimentos = ['Brush_teeth','Climb_stairs','Comb_hair','Descend_stairs','Drink_glass','Eat_meat','Eat_soup','Getup_bed','Liedown_bed','Pour_water','Sitdown_chair','Standup_chair','Use_telephone','Walk',]
```

Criando um *dataframe* vazio para adicionar os dados dos arquivos:

```
df_all = pd.DataFrame(columns=["X","Y","Z","Ano","Mes","Dia","Horas","Minutos","Segundos","Tipo_movimento","Genero","Num_voluntario"])
```

Loop para iterar pelas pastas:

```
for y in range(len(nome_movimentos)):
  directory = os.fsencode(f'dados/{nome_movimentos[y]}')
```

Função para iterar por cada arquivo de uma pasta e extrair o nome dos arquivos:

```
def itera(directory):

    x=0
    array=[]
    for file in os.listdir(directory):
        filename = os.fsdecode(file)

        x += 1

        array.append(filename)

    return array
```

Retorna um *array* com o nome dos arquivos de uma pasta:

```
    nome_arquivos = itera(directory)
```

Loop que itera sobre os arquivos:

```
    for i in range(len(nome_arquivos)):
        split_nome_arquivos = nome_arquivos[i].split('-')
        split_nome_arquivos
```

Criação do *dataframe* com os dados de um arquivo e com todas as colunas necessárias:

```
    df = pd.read_csv(f'dados/{nome_movimentos[y]}/{nome_arquivos[i]}',
header=None,
names=["X","Y","Z","Ano","Mes","Dia","Horas","Minutos","Segundos","Tipo_
movimento","Genero","Num_voluntario"],sep=" ")
        df['Ano'] = split_nome_arquivos[1]
        df['Mes'] = split_nome_arquivos[2]
        df['Dia'] = split_nome_arquivos[3]
        df['Horas'] = split_nome_arquivos[4]
        df['Minutos'] = split_nome_arquivos[5]
        df['Segundos'] = split_nome_arquivos[6]
        df['Tipo_movimento'] = split_nome_arquivos[7]
        df['Genero'] = split_nome_arquivos[8][0]
        df['Num_voluntario'] = split_nome_arquivos[8][1]
```

Junção do *dataframe* ‘`df_all`’ com o *dataframe* `‘df’`:

```
    df_all = pd.merge(df_all, df, how = 'outer')
    del nome_arquivos
```

[4]:

	X	Y	Z	Ano	Mes	Dia	Horas	Minutos	Segundos	Tipo_movimento	Genero	Num_voluntario
0	26	51	36	2011	04	11	13	29	54	brush_teeth	f	1
1	26	51	36	2011	04	11	13	29	54	brush_teeth	f	1
2	26	51	37	2011	04	11	13	29	54	brush_teeth	f	1
3	26	51	37	2011	04	11	13	29	54	brush_teeth	f	1
4	26	51	37	2011	04	11	13	29	54	brush_teeth	f	1
...	...	...	...	...	...	...	...	...	...	...	...	...
446524	10	36	40	2012	06	06	08	59	00	walk	m	5
446525	11	36	40	2012	06	06	08	59	00	walk	m	5
446526	13	40	39	2012	06	06	08	59	00	walk	m	5
446527	16	41	38	2012	06	06	08	59	00	walk	m	5
446528	12	39	35	2012	06	06	08	59	00	walk	m	5

446529 rows × 12 columns

Salvar o *dataframe* em um arquivo `csv` com o nome ‘`dados_coletados.csv`’:

```
df_all.to_csv('dados_coletados.csv', encoding='utf-8', index=False)
```

1.2 - Exploração

Importar o *pandas*, *matplotlib* e baixar o *pandasql* e o importar:

```
import pandas as pd
import matplotlib.pyplot as plt
pip install -U pandasql
import pandasql
```

Conhecer a estrutura do data frame criado

```
df = pd.read_csv('dados_coletados.csv')
```

Linhas e colunas do data frame:

```
df.shape
```

Obtendo várias informações sobre o dataframe, tipo dos dados, quantidades de entradas, se há valores nulos, tamanho da memória e etc:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 446529 entries, 0 to 446528
Data columns (total 12 columns):
 #   Column           Non-Null Count   Dtype 
---  ------           --------------   ----- 
 0   X                446529 non-null  int64 
 1   Y                446529 non-null  int64 
 2   Z                446529 non-null  int64 
 3   Ano              446529 non-null  int64 
 4   Mes              446529 non-null  int64 
 5   Dia              446529 non-null  int64 
 6   Horas            446529 non-null  int64 
 7   Minutos          446529 non-null  int64 
 8   Segundos         446529 non-null  int64 
 9   Tipo_movimento   446529 non-null  object
 10  Genero           446529 non-null  object
 11  Num_voluntario   446529 non-null  int64 
dtypes: int64(10), object(2)
memory usage: 40.9+ MB
```

	X	Y	Z	Ano	Mes	Dia	Horas	Minutos	Segundos	Tipo_movimento	Genero	Num_voluntario
0	39	57	42	2011	5	30	21	55	4	brush_teeth	m	2
1	37	33	32	2011	5	30	21	55	4	brush_teeth	m	2
2	40	59	39	2011	5	30	21	55	4	brush_teeth	m	2
3	40	59	39	2011	5	30	21	55	4	brush_teeth	m	2
4	40	57	35	2011	5	30	21	55	4	brush_teeth	m	2
...	...	...	...	...	...	...	...	...	...	...	...	...
446524	17	32	37	2012	6	6	9	10	2	walk	m	5
446525	16	33	32	2012	6	6	9	10	2	walk	m	5
446526	12	34	36	2012	6	6	9	10	2	walk	m	5
446527	12	35	37	2012	6	6	9	10	2	walk	m	5
446528	15	36	37	2012	6	6	9	10	2	walk	m	5

446529 rows × 12 columns

Obter diversas métricas sobre as variáveis, sendo elas: Média (mean), mediana (50%) e desvio padrão (std):

```
df.describe()
```

	X	Y	Z	Ano	Mes	Dia	Horas	Minutos	Segundos	Num_voluntario
count	446529.000000	446529.000000	446529.000000	446529.000000	446529.000000	446529.000000	446529.000000	446529.000000	446529.000000	446529.000000
mean	24.671555	38.219491	41.839842	2011.393708	5.254532	17.401645	13.836747	32.101232	30.131481	2.191325
std	12.157593	7.690688	8.303436	0.488572	1.694558	11.124160	4.504576	17.139545	17.353014	1.816291
min	0.000000	0.000000	0.000000	2011.000000	3.000000	1.000000	3.000000	0.000000	0.000000	1.000000
25%	13.000000	35.000000	35.000000	2011.000000	5.000000	6.000000	10.000000	17.000000	15.000000	1.000000
50%	25.000000	38.000000	42.000000	2011.000000	5.000000	20.000000	14.000000	33.000000	31.000000	1.000000
75%	34.000000	42.000000	50.000000	2012.000000	6.000000	30.000000	17.000000	48.000000	44.000000	3.000000
max	63.000000	63.000000	63.000000	2012.000000	12.000000	31.000000	22.000000	59.000000	59.000000	9.000000

Observações por gênero usando o panda SQL:

```
def qtd_rel_genero(df):
  q = """
```

```
          SELECT Genero, COUNT(*) as "Qtd_relatos"
          FROM df
          GROUP BY Genero
          ORDER BY count(*) DESC

          """

  consulta1 = pandasql.sqldf(q.lower(), locals())
  return consulta1

qtd_rel_genero(df)
```

:

	Genero	qtd_relatos
0	f	283189
1	m	163340

Observações por movimento usando o pandasSQL:

```
def qtd_rel_move(df):
  q = """
          SELECT Tipo_movimento, COUNT(*) as "Qtd_relatos"
          FROM df
          GROUP BY Tipo_movimento
          ORDER BY count(*) DESC

          """

  consulta2 = pandasql.sqldf(q.lower(), locals())
  return consulta2

qtd_rel_move(df)
```

	Tipo_movimento	qtd_relatos
0	walk	92254
1	getup_bed	45801
2	drink_glass	42792
3	pour_water	41673
4	climb_stairs	40258
5	eat_meat	31236
6	brush_teeth	29829
7	standup_chair	25417
8	sitdown_chair	25036
9	comb_hair	23504
10	descend_stairs	15375
11	use_telephone	15225
12	liedown_bed	11446
13	eat_soup	6683

Procurar algumas correlação entre as coordenadas (valores negativos indicam correlação inversa, isto é, quando um cresce o outro diminui (inversamente proporcional) ; positivos indicam crescimento uniforme, ou seja, ambas crescem ou ambas diminuem (diretamente proporcional) ; o valores vão de -1 a +1 e quanto mais próximo o valor absoluto 1, mais forte a ligação entre as duas variáveis). Observe que a relação de uma variável com ela mesma é igual a 1:

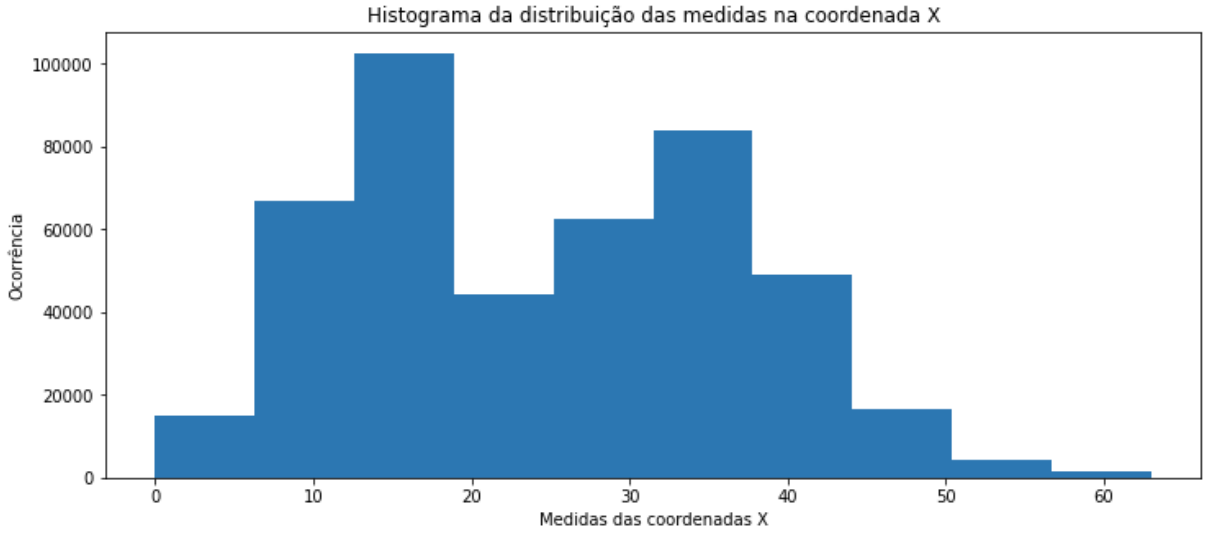
```
x = pd.Series(df['X'])

y = pd.Series(df['Y'])

z = pd.Series(df['Z'])

print('A correlação entre X e Y é muito fraca:', x.corr(y))
print('A correlação entre X e Z é moderada:', x.corr(z))
print('A correlação entre Y e Z é muito fraca:', y.corr(z))
```

```
A correlação entre X e Y é muito fraca: -0.036538437948152085
A correlação entre X e Z é moderada: 0.6027531936837502
A correlação entre Y e Z é muito fraca: -0.16681230480283113
```

Histograma revelando a distribuição das medidas obtidas nas coordenadas X:

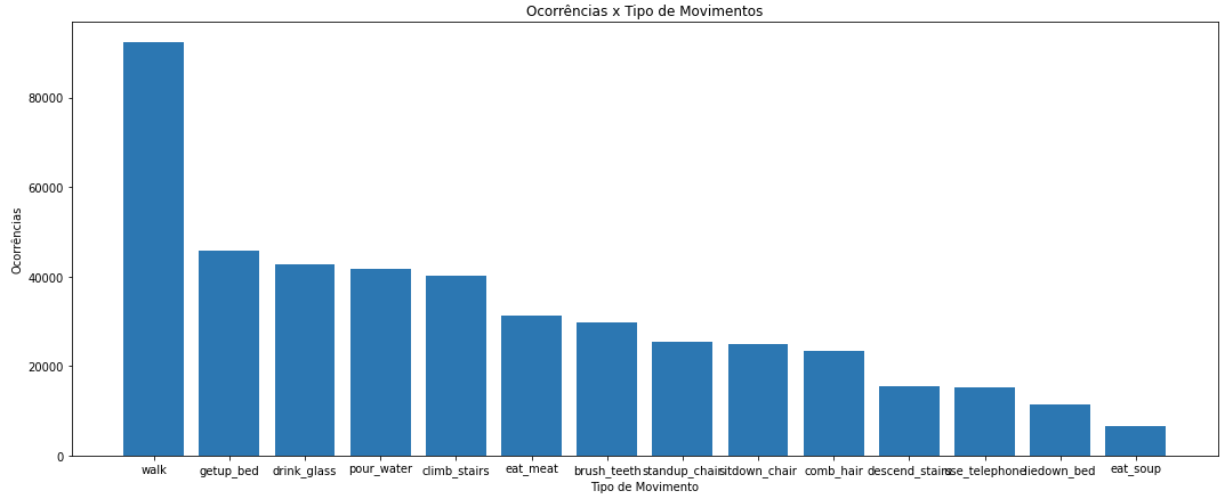
```
plt.figure(figsize=(12, 5))
plt.hist(df.X)
plt.title('Histograma da distribuição das medidas na coordenada X')
plt.xlabel('Medidas das coordenadas X')
```

```
plt.ylabel('Ocorrência')
plt.show()
```



Gerar um gráfico com ocorrências por tipo de movimento, subset = none, normalize = false, sort = true:

```
xis = qtd_rel_move(df)

plt.figure(figsize=(18, 7))
plt.bar(xis.Tipo_movimento, xis.qtd_relatos)
plt.title('Ocorrências x Tipo de Movimentos')
plt.xlabel('Tipo de Movimento')
plt.ylabel('Ocorrências')

plt.show()
```



1.3 - Preparação:

Importando Pandas, numpy e o dataframe:

```
import pandas as pd
import numpy as np
df = pd.read_csv('dados_coletados.csv')
```

Eliminar a variável com nome do arquivo:

```
del nome_arquivos
```

Acrescentar nova variável representando a média das coordenadas, ou seja, obter, para cada instante, a média das três coordenadas (X, Y, Z) e ao final você deverá ter um vetor de médias ao longo do tempo.

Criar uma variável ‘data’ para delimitar os instantes:

```
df['data']=df['Dia'].astype(str)+"/"+df['Mes'].astype(str)+"/"+df['Ano'].a
stype(str)+"
"+df['Horas'].astype(str)+":"+df['Minutos'].astype(str)+":"+df['Segundos']
.astype(str)
```

Criação das colunas ‘MediaX’, ‘MediaY’ e ‘MediaZ’:

```
df['MediaX'] = ''
df['MediaY'] = ''
df['MediaZ'] = ''
```

Atribuindo os instantes na variável ‘instantes’:

```
instantes = df.data.unique()
```

Loop para fazer a média das coordenadas X, Y e Z e atribuir nas variáveis respectivas:

```
for x in range(len(instantes)):
  copydf = df[df.data == instantes[x]].copy()
  copydf['MediaX'] = round(copydf['X'].mean())
  copydf['MediaY'] = round(copydf['Y'].mean())
  copydf['MediaZ'] = round(copydf['Z'].mean())
  df[df.data == instantes[x]] = copydf
df
```

	X	Y	Z	Ano	Mes	Dia	Horas	Minutos	Segundos	Tipo_movimento	Genero	Num_voluntario	data	MediaX	MediaY	MediaZ
0	39	57	42	2011	5	30	21	55	4	brush_teeth	m	2	30/5/2011 21:55:4	30	51	39
1	37	33	32	2011	5	30	21	55	4	brush_teeth	m	2	30/5/2011 21:55:4	30	51	39
2	40	59	39	2011	5	30	21	55	4	brush_teeth	m	2	30/5/2011 21:55:4	30	51	39
3	40	59	39	2011	5	30	21	55	4	brush_teeth	m	2	30/5/2011 21:55:4	30	51	39
4	40	57	35	2011	5	30	21	55	4	brush_teeth	m	2	30/5/2011 21:55:4	30	51	39
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
446524	17	32	37	2012	6	6	9	10	2	walk	m	5	6/6/2012 9:10:2	12	37	34
446525	16	33	32	2012	6	6	9	10	2	walk	m	5	6/6/2012 9:10:2	12	37	34
446526	12	34	36	2012	6	6	9	10	2	walk	m	5	6/6/2012 9:10:2	12	37	34
446527	12	35	37	2012	6	6	9	10	2	walk	m	5	6/6/2012 9:10:2	12	37	34
446528	15	36	37	2012	6	6	9	10	2	walk	m	5	6/6/2012 9:10:2	12	37	34

446529 rows × 16 columns

Veja o grau de correlação da nova variável que criamos em relação as demais coordenada:

```
x = pd.Series(df['MediaX'])
```

```
y = pd.Series(df['MediaY'])

z = pd.Series(df['MediaZ'])

print('A correlação entre MediaX e MediaY é muito fraca:',
x.astype('float64').corr(y.astype('float64')))
print('A correlação entre MediaX e MediaZ é forte:',
x.astype('float64').corr(z.astype('float64')))
print('A correlação entre MediaY e MediaZ é fraca:',
y.astype('float64').corr(z.astype('float64')))
```

```
A correlação entre MediaX e MediaY é muito fraca: -0.03804779820593199
A correlação entre MediaX e MediaZ é forte: 0.7653569961648434
A correlação entre MediaY e MediaZ é fraca: -0.31265651490990276
```

Salvar o dataframe em um arquivo csv com o nome ‘dados preparados.csv’:

```
df.to_csv('dados_preparados.csv', encoding='utf-8', index=False)
```

4. Considerações finais

A Sprint foi realizada com desempenho satisfatório. O grupo conseguiu realizar as tarefas sem grandes problemas. Porém, houve muita dificuldade de entender alguns itens da sprint, mais especificamente, os itens: 1.1/Coleta, 1.3.2 e 1.3.3. Os referidos comandos não foram considerados claros pelo o grupo, o que demandou muito tempo ao tentar interpretar a finalidade concreta dessas tarefas. Possíveis melhorias seriam a melhor organização do código e dos horários das reuniões em grupo.

Referências

Pandas- https://pandas.pydata.org/docs/

PandaSql- https://pypi.org/project/pandasql/

Python- https://docs.python.org/3/

Matplotlib- https://matplotlib.org/

Sobre correlação- https://escoladedados.org/tutoriais/correlacao-nao-e-causalidade-mas-o-que-e-entao/

COUTINHO, Thiago. O que é Python e pra que serve?. **Voitto**, 2019. Disponível em: <https://www.voitto.com.br/blog/artigo/python>. Acesso em: 01 de aug. de 2021.

NOLETO, Cairo. Google Colab: saiba o que é essa ferramenta e como usar. **Trybe**, 2020. Disponível em: <https://blog.betrybe.com/carreira/google-colab/>. Acesso em: 29 de jul. de 2021.

ROVEDA, Ugo. O que é Python, para que serve e por que aprender?. **Kenzie**, 2021. Disponível em: <https://kenzie.com.br/blog/o-que-e-python/>. Acesso em: 01 de aug. de 2021.

SANTOS, Thiago G. Google Colab: O que é, Tutorial de Como Usar e Criar Códigos. **Alura**, 2020. Disponível em: <https://www.alura.com.br/artigos/google-colab-o-que-e-e-como-usar>. Acesso em: 29 de jul. de 2021.

SOUZA, Ivan de. Entenda de uma vez o que é Github e a importância dele num negócio. **Rock Content**, 2020. Disponível em: <https://rockcontent.com/br/blog/o-que-e-github/>. Acesso em: 01 de aug. de 2021.