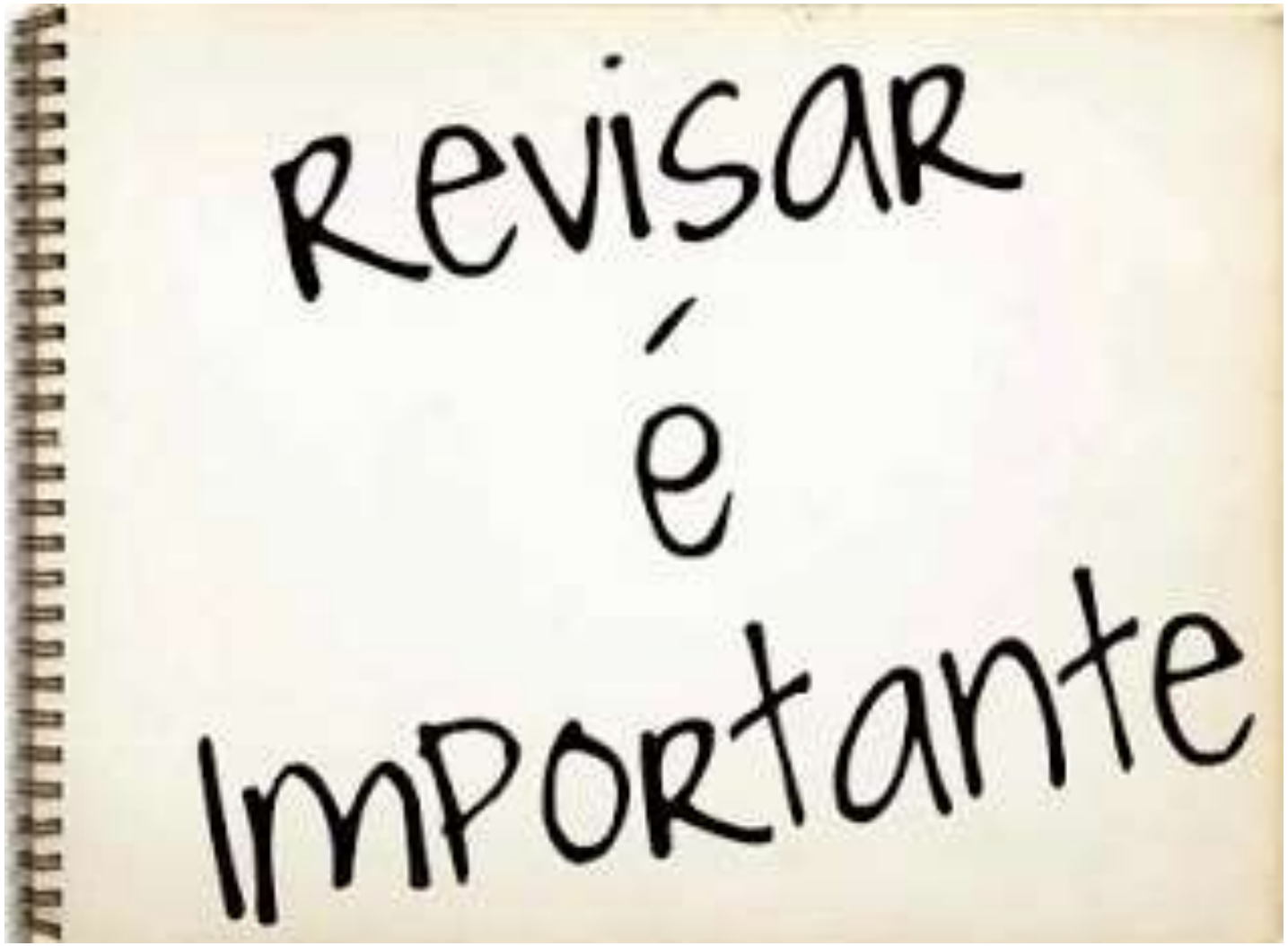
CENTRO UNIVERSITÁRIO
INSTITUTO DE EDUCAÇÃO SUPERIOR DE BRASÍLIA – IESB

Aprendizagem de Máquina

PABLO COELHO FERREIRA

Antes de começarmos...
.



pablo.coelho@gmail.com

SETS

- São coleções **não ordenadas** de **elementos únicos**.

```
setMeuConjunto = set()
setMeuConjunto.add('a')
setMeuConjunto.add('d')
setMeuConjunto.add('b')
setMeuConjunto.add(2)
print (setMeuConjunto) # imprime: {'d', 'b', 2, 'a’} em qualquer ordem.
```

Sets e listas

```
lisMinhaLista = [1,1,1,1,2,2,21,3,4,51,23,1,2,3,5,8]
print (lisMinhaLista)
print (set(lisMinhaLista))

#Lista ->  [1, 1, 1, 1, 2, 2, 21, 3, 4, 51, 23, 1, 2, 3, 5, 8]
#Set -> {1, 2, 3, 4, 5, 8, 51, 21, 23}
```

Arquivos – Criar e gravar

- **1º passo – criar um arquivo.**

Se tentar criar um arquivo que já existe vai dar pau!

```
f = open ("meuArquivo.txt","x")
```

- **2º Passo – abrir o arquivo para escrever algo nele (append)**

```
f = open ("meuArquivo.txt","a")
f.write ("Minha primeira linha.")
f.write ("Minha segunda linha.\nTerceira linha")
```

- **3º Passo – fechar o arquivo**

```
f.close()
```

Estrutura de decisões

```
if condição:
    código1 (executa todas as linhas iniciadas na mesma coluna – indentação)
elif:
    código2 (executa todas as linhas iniciadas na mesma coluna – indentação)
else:
    código3 (executa todas as linhas iniciadas na mesma coluna – indentação)

Código 4 (fora da condição if)
```

FOR

- O For no Python funciona apenas para listas.

```
lstMinhaLista = [1,2,3,4,5]

for qlqNome in lstMinhaLista: # O primeiro parâmetro é uma variável
                              # que pode ter qualquer nome.
                              # o segundo parâmetro (depois do IN)
                              # é o nome da lista que você quer percorrer.

    print (qlqNome)    # Vai imprimir ‘de 1 a 5, sendo um número embaixo do
                       # outro.
```

in

```
lstMinhaLista1 = [1,2,3]

print (1 in lstMinhaLista1)
```

max e min

```
lstMinhaLista1 = [1,2,3,4,5]

print (max(lstMinhaLista1))

print (min(lstMinhaLista1))
```

Pandas e MathPlotLib

Revisão

pablo.coelho@gmail.com

Pandas

- É uma biblioteca escrita para programação em Python.
- Serve para analisar, limpar, explorar e manipular dados (Big Data).
- Oferece estrutura e operações para manipular tabelas numéricas e séries temporais.
- É software livre rodando sob a licença Three-Clause BSD.

Instalação – Visual Code ou outras IDEs

Instalar gerenciador de pacotes PIP

Depois executar:

pip install pandas

Funcionalidades básicas

pablo.coelho@gmail.com

Série de dados

```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a)

print(myvar)

print(myvar[0])
```

Saída do código

```
0  1
1  7
2  2


1
```

Observe que foi criada uma coluna com o índice da lista. Se nada estiver especificado para ser este índice, sua posição será utilizada.

Etiquetando os dados (nomes para colunas)

```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a, index = ["x", "y", "z"])

print(myvar)

print(myvar[“y”])
```

Saída do código

x 1

y 7

z 2

7

Séries a partir de dicionários

```
import pandas as pd

calories = {"dia 1": 420, "dia 2": 380, "dia 3": 390}

myvar = pd.Series(calories)

print(myvar)
```

Saída do código

dia 1 420

dia 2 380

dia 3 390

Selecionando dados para a série

```
import pandas as pd

calories = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(calories, index = ["day1", "day2"])

print(myvar)
```

Saída do Código

day1 420

day2 380

DataFrame (tabela completa)

```
import pandas as pd

mydataset = {
  'cars': ["BMW", "Volvo", "Ford"],
  'passings': [3, 7, 2]
}

myvar = pd.DataFrame(mydataset)

print(myvar)
```

Saída do código

	cars	passings
0	BMW	3
1	Volvo	7
2	Ford	2

Estrutura com duas dimensões. Similar a um vetor (array) de duas dimensões ou a uma tabela de dados.

(Considerando o código anterior)

```
print (myvar.loc[0])
```

Saída:

```
cars        BMW
Passings    3
```

(Considerando o código anterior)

```
print (myvar.loc[[0,1]])
```

```
      cars  passings
0      BMW         3
1    Volvo         7
```

Carregando um arquivo para um DataFrame

- import pandas as pd
- df = pd.read_csv('data.csv')
- print(df)

Saída do código

	nome	dia nascimento	mes nascimento	uf	cidade
0	Pablo	26	1	df	brasilia
1	Enzo	6	5	df	brasilia
2	Tiago	2	3	df	florianopolis
3	maria	26	4	sp	sao paulo

Arquivo dados2.csv

- Contêm 169 registros referentes a Duração, Pulsação, Max e Calorias.

Duracao,Pulso,Max,Calorias
60,110,130,409.1
60,117,145,479.0
60,103,135,340.0
45,109,175,282.4
45,117,148,406.0
60,102,127,300.0
60,110,136,374.0
45,104,134,253.3
(...)

to_string

```
import pandas as pd

df = pd.read_csv('dados2.csv')

print(df.to_string())
```

Utilize o to_string para imprimir todos os dados de um DataFrame. Caso contrário vai imprimir apenas os cinco primeiros e os cinco últimos.

Exemplo – sem o to_string

	Duracao	Pulso	Max	Calorias
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0
..	...	...	...	...
164	60	105	140	290.8
165	60	110	145	300.0
166	60	115	145	310.2
167	75	120	150	320.4
168	75	125	150	330.4

Lendo todos os itens de uma coluna

```
import pandas as pd

df = pd.read_csv(<nome arquivo>)
vetDados = df[<nome coluna>]

for item in vetDados:
    <Código onde ”item” tem os dados da coluna>
```

Número máximo de registros retornados

```
import pandas as pd

print(pd.options.display.max_rows)
```

Vai mostrar o número máximos de linhas (registros) que um Dataframe vai retornar. Se existir mais registros do que o max_rows, o Pandas trás o cabeçalho e os cinco primeiros e últimos registros.

Esse valor pode ser alterado: **pd.options.display.max_rows = 9999**

Lendo JSON

- Arquivos JSON são arquivos texto formatados como objetos.

```
import pandas as pd

df = pd.read_json('data.json')

print(df.to_string())
```

Arquivo JSON


```
{
   "Duration":{
     "0":60,
     "1":60,
(...)
 },
   "Pulse":{
     "0":110,
     "1":117,
(...)
 },
   "Maxpulse":{
     "0":130,
     "1":145,
(...)
 }
 }
```

head e tail

```
import pandas as pd

df = pd.read_csv('dados2.csv')

print("cabecalho")
print(df.head())
print("cabecalho (dez primeiros)")
print(df.head(10))
print("cauda")
print(df.tail())
print("cauda (dez últimos)")
print(df.tail(10))
```

Saída do código

cabecalho

	Duracao	Pulso	Max	Calorias
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0

cabecalho (dez primeiros)

	Duracao	Pulso	Max	Calorias
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0
5	60	102	127	300.0
6	60	110	136	374.0
7	45	104	134	253.3
8	30	109	133	195.1
9	60	98	124	269.0

cauda

	Duracao	Pulso	Max	Calorias
164	60	105	140	290.8
165	60	110	145	300.0
166	60	115	145	310.2
167	75	120	150	320.4
168	75	125	150	330.4

cauda (dez últimos)

	Duracao	Pulso	Max	Calorias
159	30	80	120	240.9
160	30	85	120	250.4
161	45	90	130	260.4
162	45	95	130	270.0
163	45	100	140	280.9
164	60	105	140	290.8
165	60	110	145	300.0
166	60	115	145	310.2
167	75	120	150	320.4
168	75	125	150	330.4

pablo.coelho@gmail.com

Informações sobre os dados

```
print(df.info())
```

```
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Duracao   169 non-null    int64
 1   Pulso     169 non-null    int64
 2   Max       169 non-null    int64
 3   Calorias  164 non-null    float64 #(existem 5 registros sem o dado no
                                       arquivo.)
```

Limpando dados

Daqui para frente iremos realizar os exercícios....

pablo.coelho@gmail.com

Limpeza de dados

- Para uma correta aprendizagem a máquina deve ser “educada” com dados o mais limpo possível.

- Evitar:
  - Células vazias
  - Dados em formato errado
  - Dados errados
  - Dupicados

Retirando registros com célula(s) vazia(s)

import pandas as pd

df = pd.read_csv('dados3.csv’) #Arquivo de dados é o **dados3.csv**

print(df.info()) #32 registros

new_df = df.**dropna**() #Retorna um novo DataFrame, não altera o original

print(df.info()) #31 registros

pablo.coelho@gmail.com

Para alterar o DataFrame original

```
import pandas as pd

df = pd.read_csv(‘dados3.csv')

df.dropna(inplace = True)

print(df.to_string())
```

Inserindo um valor padrão onde está vazio

```
import pandas as pd

df = pd.read_csv('dados3.csv')

df.fillna(130, inplace = True) #Toda célula vazia receberá o valor 130.
```

Alterando vazio para um valor padrão em uma coluna

```
import pandas as pd

df = pd.read_csv('dados3.csv')

df["Calorias"].fillna(130, inplace = True)
```

Média, Mediana e Moda

- A **MÉDIA** de um conjunto de dados é encontrada somando-se todos os números do conjunto de dados e então dividindo o resultado pelo número de valores do conjunto.

- A **MEDIANA** é o número central de uma lista de dados organizados de forma crescente ou decrescente, sendo uma medida de tendência central ou, de centralidade. Indica o centro da distribuição da variável, ou seja, é o valor acima do qual estão 50% dos valores da variável e abaixo os restantes 50%

- A **MODA** é o número que aparece mais vezes em um conjunto de dados.

pablo.coelho@gmail.com

Substituindo pelo valor médio, mediana e moda.

```
import pandas as pd

df = pd.read_csv('dados3.csv')

x = df["Calorias"].mean()     #median ou mode()[0]
                              #mode retorna uma série, em cada posição,
                              #em ordem, volta o valor do maior para a
                              #menor quantidade presente na série

df["Calorias"].fillna(x, inplace = True)
```

Correção de tipo de valor

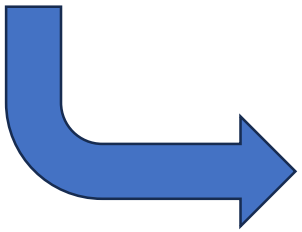
```
import pandas as pd

df = pd.read_csv('dados3.csv')

print(df.to_string())

df['data'] = pd.to_datetime(df['data'], format='mixed') # vai corrigir os
                                                        # registros 22 (NaT) e
                                                        # 26 (vai formatar a data)

print(df.to_string())
```

Limpar linhas que não tenham datas

- Mesmo código anterior, acrescentando a linha:

df.dropna(subset=['data'], inplace = True)

21	60	2020-12-21	108	131	364.2
22	45	NaT	100	119	282.0
23	60	2020-12-23	130	101	300.0



21	60	2020-12-21	108	131	364.2
23	60	2020-12-23	130	101	300.0

Ajustando dados errados

```
import pandas as pd

df = pd.read_csv('dados3.csv')

print(df.to_string())

df.loc[7, 'Duracao'] = 45 # ajusta o registro 7 para o valor desejado.

print(df.to_string())
```

Pode ser utilizado para vários registros

```
import pandas as pd

df = pd.read_csv('dados3.csv')

for x in df.index:
  if df.loc[x, "Duracao"] > 120:
    df.loc[x, "Duracao"] = 120 # Altera o valor da coluna Duracao de
                                  # uma determinada linha
OU
for x in df.index:
  if df.loc[x, "Duracao"] > 120:
    df.drop(x, inplace = True)   # Elimina a linha
```

Removendo duplicatas

```
import pandas as pd

df = pd.read_csv('dados3.csv')

print(df.duplicated()) # Mostra as linhas duplicadas

df.drop_duplicates(inplace = True) # Remove as linhas duplicadas

print(df.to_string())
```

Plotando os dados em um gráfico

```
import pandas as pd

import matplotlib.pyplot as plt

df = pd.read_csv('dados3.csv')

df.plot()

plt.show()
```

Gráfico de Pizza

```
import pandas as pd
from matplotlib import pyplot as plt

df = pd.read_csv('dados4.csv')
from matplotlib import pyplot as plt
import numpy as np

numA = 0
numB = 0
vetNumeros = df['survived']
```

```
for item in vetNumeros:
    if item == 0:  #verifica se o registro
                   #é 0. A coluna só tem
                   #registros 0 e 1.
        numA += 1
    else:
        numB += 1

dados = [numA,numB]
plt.pie(dados, labels = ['Números 0',' Números 1'])
plt.show()
```

Gráfico de dispersão

plt.scatter (x, y)

plt.title(<Título do Gráfico>)
plt.xlabel(<Label do eixo X>)
plt.ylabel(<Label do eixo X>)

“Causos” – um pouco de experiência...

- Um dos maiores clientes/compradores não exige que vc tenha marca, local físico de trabalho, vendedores...

- Vc, seu conhecimento, um notebook e um link de internet.

pablo.coelho@gmail.com

Dever de casa

Trabalhando com Sets:

Crie uma lista contendo os seguintes elementos: "maçã", "banana", "laranja", "uva", "maçã", "melão", "mamão" e "banana".

Crie um arquivo chamado minhas_frutas.txt e grave nele os nomes das frutas do conjunto, com uma quantidade aleatória entre 0 e 100 para cada registro.

.

Abra o arquivo, leia e exiba seu conteúdo no console através de um Data Frame com os nomes das colunas: Fruta e quantidade

Atente-se que há frutas repetidas no arquivo e as suas quantidades devem ser somadas.

pablo.coelho@gmail.com

Trabalho

Descrição do Trabalho

- O grupo deverá desenvolver um **analisador de dados** que carregue, processe e visualize informações de um conjunto de dados fornecido*.

- O sistema deve ser implementado em **Python**, utilizando as bibliotecas **Pandas** para manipulação de dados e **Matplotlib** para visualização gráfica. O trabalho deverá seguir o padrão de desenvolvimento **PEP 8 – Style Guide for Python Code** (referência) **(1 ponto).**

*https://www.kaggle.com/datasets/mahmoudelhemaly/students-grading-dataset

Trabalho – Apresentar na aula seguinte a prova.

Requisitos do Projeto – Itens:

1. O programa deve permitir o carregamento de um arquivo **CSV ou JSON** com dados de alunos (students-grading-dataset). (1 ponto)
    1. Perguntar ao usuário o caminho do arquivo
    2. O usuário deve poder visualizar um resumo estatístico dos dados carregados.
        1. Quantidade de dados carregados
        2. Quantidade de homens e mulheres
        3. Quantos registros sem dados sobre a educação dos pais.
2. Limpeza de dados (3 pontos)
    1. Remover os registros que tem a educação dos pais vazios.
    2. Alterar os dados de presença (Attendance) que estão null para a mediana.
    3. Apresentar o somatório de Attendance.
3. Consulta a dados (2 pontos)
    1. O usuário pode escolher uma das colunas e o sistema deve apresentar: média, mediana, moda, desvio padrão dos dados daquela coluna.
4. Gráficos (3 pontos)
    1. O sistema deve produzir gráfico de dispersão para “horas de sono” x “nota final”
    2. Gráfico de barras – idade x média das notas intermediárias (midterm_Score)
    3. Gráfico de pizza para as idades (Agrupadas: até 17; 18 a 21; 21 a 24; 25 ou mais).

pablo.coelho@gmail.com

Critérios

- O sistema que não importar os dados OU travar durante a execução do código, seja por bug ou por não validação de entrada de usuário – 0 PONTOS.
  - Validar, entre outros, se o caminho do arquivo existe, se a coluna para a qual se está pedindo as médias, modas... É uma coluna numérica.

- Pontos extras (o máximo de pontos do trabalho é 10! Se passar, fica com apenas com 10)
  - 1 ponto: Documentação utilizando docstrings.
    - O resultado deve ser apresentado utilizando o Sphinx, pydoc ou MkDocs.
  - 0,1 ponto para cada função que tiver controle de erro (máximo 1 ponto).
  - 1 ponto se o sistema tiver um log de ações do usuário (todas).
    - Nesse caso deve pedir o nome do usuário como entrada (nesse caso não precisa ter crítica, exceto se é um nome composto APENAS por caracteres, com ao menos 3. Os nomes podem ser duplicados).
  - 1 ponto se o git do projeto “contar” a história da evolução. Com commits de todos alunos, evoluções, controle de bugs, etc...

Regras

- Grupos de até quatro integrantes.

- No dia da apresentação todos devem estar presentes. Quem não estiver não recebe os pontos do projeto.
  - Em caso de atestado médico, o aluno poderá fazer a apresentação na aula seguinte e/ou participar por vídeo – responsabilidade do grupo conseguir o acesso.

- A nota é do grupo. Perguntas serão feitas para os membros do grupo durante a apresentação sobre o código apresentado.

- As apresentações podem ser iniciadas até (quantidade de grupos restante)*15minutos – horário de término da aula.
  - Se estourar esse prazo, o grupo da vez será sorteado.
  - Ao iniciar a apresentação não pode alterar mais nada (Arquivos, códigos, parâmetros, caminhos, permissões....)


Thats all Folks!