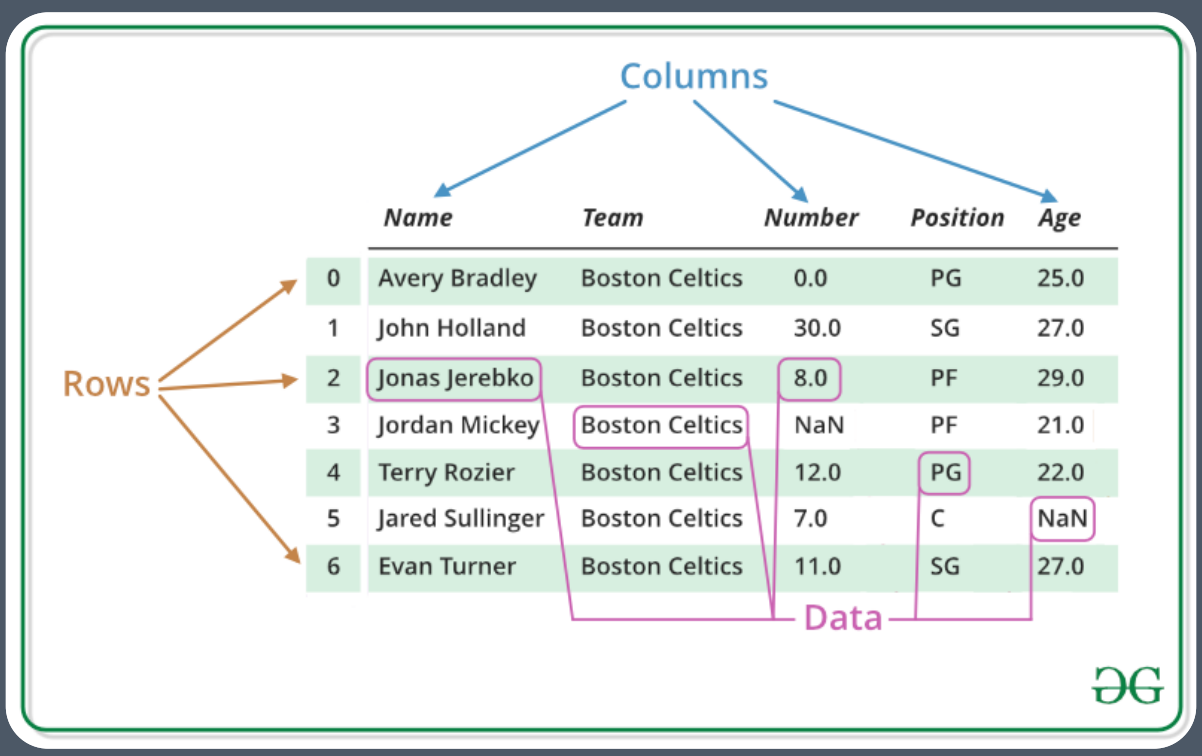
DATA Viking

Código rápido

Trabalhando com *class* (objeto) no Pandas

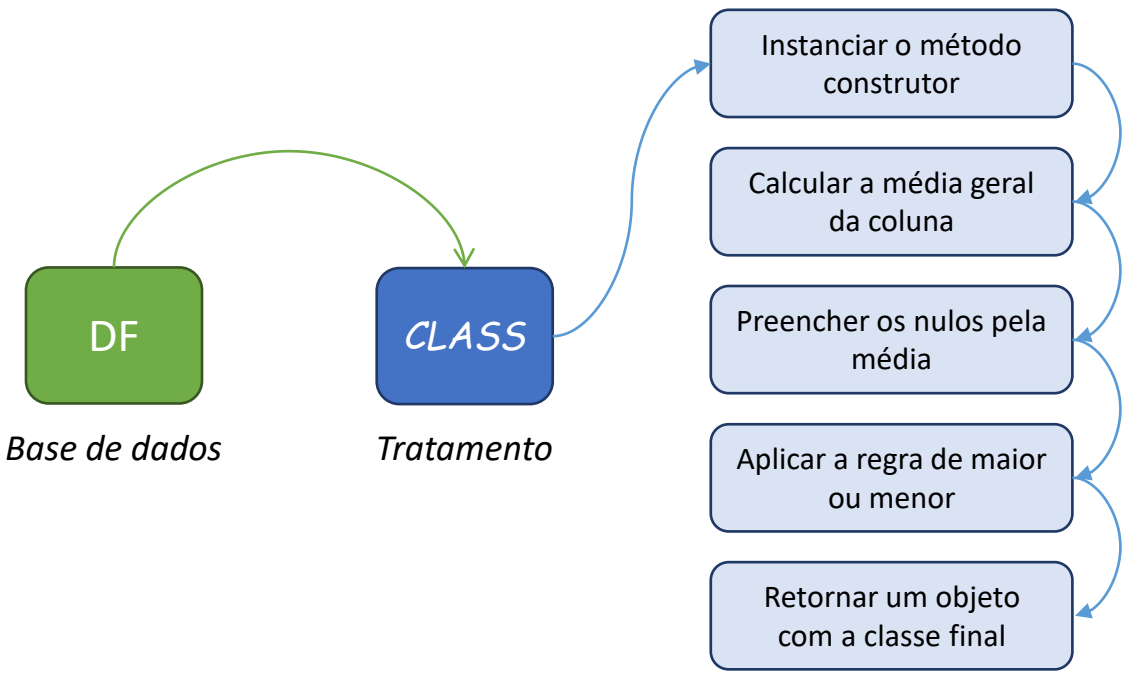


Odemir Depieri Jr
Ronisson Lucas C. da Conceição
Yan Vianna Sym
Wesley Felix

Fluxo do tratamento

Definimos uma classe chamada *"Calculo",* que tem a finalidade de calcular a média de uma coluna de um dataframe e comparar cada valor dessa coluna com a média.

O método construtor da classe recebe um dataframe e o nome da coluna que será analisada. Ele então preenche os valores nulos da coluna com a média dos valores não nulos, armazena a média em um atributo "media" e retorna o dataframe com a coluna preenchida.



Funções dentro do objeto

A função **__init__** é o construtor da classe e é responsável por df, nome_coluna, media, e por preencher os valores nulos **inicializar os atributos** om a média.

A função **calculo_media()** é responsável por **calcular a média da coluna**.

A função **preencher_nulos_com_media()** é responsável por **preencher os valores nulos** da coluna com a média.

A função **comparar_media()** é responsável por **comparar a média** com cada registro da coluna.

E a função **resultado()** é responsável por **retornar o resultado** da comparação entre a média e cada registro da coluna.

Gerar dados

Iremos **criar dados aleatórios** dentro uma classe.
Nessa classe temos função para gerar **tabela A** e **tabela B**.

```
import pandas as pd
import numpy as np
import random
import datetime

# Gerador de dados aleatórios
class GeradorDados:

    def __init__(self, quantidade):
        self.quantidade = quantidade

    def gerar_dados_a(self):

        def gerar_data():
            '''
              Geradpr de datas aleatórias considerando meses bissexto e demais dias
            '''
            ano = random.randint(2020, datetime.date.today().year-1)
            mes = random.randint(1, 12)

            # fevereiro
            if mes == 2:

                # ano bissexto
                if (ano % 4 == 0 and ano % 100 != 0) or ano % 400 == 0:
                    dia = random.randint(1, 29)
                else:
                    dia = random.randint(1, 28)

            # meses com 30 dias
            elif mes in [4, 6, 9, 11]:
                dia = random.randint(1, 30)

            # meses com 31 dias
            else:
                dia = random.randint(1, 31)

            return datetime.date(ano, mes, dia)

        dicionario = {
            #'Id_Venda' : [ loop for loop in range( self.quantidade) ],
            'Id_Venda' : random.sample( range(0, (self.quantidade*2) ), self.quantidade) ,
            'Valor_Imovel': [ random.randint(150000, 350000) for loop in range(self.quantidade) ]
,
            'Comissao_Venda': [ random.randint(1, 6) / 100 for loop in range(self.quantidade) ],
            'Data_Venda': [ gerar_data() for loop in range(self.quantidade) ],
            'Tipo_Produto': [ np.random.choice(['Casa Residencial', 'Apartamento'], p=[0.25, 0.75]
) for loop in range(self.quantidade) ],
        }

        Base = pd.DataFrame( dicionario )
        return Base

    def gerar_dados_b(self):

        dicionario = {
            'Id_Venda' : random.sample( range(0, (self.quantidade*2) ), self.quantidade) ,
            'Vendedor': [ np.random.choice(['Odemir Depieri Jr', 'Ronisson Lucas Calmon']) for lo
op in range(self.quantidade) ]
        }

        Base = pd.DataFrame( dicionario )
        return Base
```

Criando a base de dados

Vamos importar nossa função de geração de dados

```
Base_A = GeradorDados(500).gerar_dados_a()
Base_B = GeradorDados(500).gerar_dados_b()
Base_Dados = pd.merge( Base_A, Base_B, how='inner', on= 'Id_Venda' )
Base_Dados.shape
```

```
(250, 6)
```

```
Base_Dados.head()
```

	Id_Venda	Valor_Imovel	Comissao_Venda	Data_Venda	Tipo_Produto	Vendedor
0	875	299515	0.04	2020-02-10	Apartamento	Ronisson Lucas Calmon
1	13	163881	0.04	2021-12-24	Casa Residencial	Ronisson Lucas Calmon
2	530	210067	0.04	2020-10-18	Apartamento	Ronisson Lucas Calmon
3	508	286790	0.01	2021-10-23	Casa Residencial	Odemir Depieri Jr
4	61	262132	0.06	2022-08-10	Casa Residencial	Odemir Depieri Jr

Preencher com nulos

Vamos incluir alguns valores nulos na coluna ‘valor_imóvel’ para ser preenchimento na nossa classe.

```
# Vamos incluir 2 registros nulos na base de dados
Base_Dados.iloc[0:2, 1] = np.NAN

# Analisando
Base_Dados.head()
```

	Id_Venda	Valor_Imovel	Comissao_Venda	Data_Venda	Tipo_Produto	Vendedor	Classe
0	875	NaN	0.04	2020-02-10	Apartamento	Ronisson Lucas Calmon	acima da média
1	13	NaN	0.04	2021-12-24	Casa Residencial	Ronisson Lucas Calmon	acima da média
2	530	210067.0	0.04	2020-10-18	Apartamento	Ronisson Lucas Calmon	abaixo da média
3	508	286790.0	0.01	2021-10-23	Casa Residencial	Odemir Depieri Jr	acima da média
4	61	262132.0	0.06	2022-08-10	Casa Residencial	Odemir Depieri Jr	acima da média

Valores nulos foram aplicados pela função ‘*np.NAN*’ do numpy.

Criando a Classe para tratamento

Criando os parâmetros da classe.

```
# Classe para calcular a média de uma coluna e compara co
m ela mesma
class Calculo:

    # Metodo construtor
    def __init__(self, df, nome_coluna):
        self.df = df
        self.nome_coluna = str(nome_coluna)
        self.df[ self.nome_coluna]  = self.preencher_nulos_co
m_media()
        self.media = self.calculo_media()

    # Função para calcular a média da coluna
    def calculo_media(self):
        return self.df[self.nome_coluna].mean()

    # Preecnher com a média
    def preencher_nulos_com_media(self):
      media = self.df[self.nome_coluna].mean()
      return self.df[self.nome_coluna].fillna(media)

    # Função para comparar a média com os registros
    def comparar_media(self, valor):
        if valor > self.media:
            return 'acima da média'
        elif valor < self.media:
            return 'abaixo da média'
        else:
            return 'igual à média'

    # Função para retornar o df
    def resultado(self):
        return self.df[self.nome_coluna].apply( self.comparar
_media )
```

O final da classe será um objeto contendo uma coluna com a informação se o valor está acima ou abaixo da média.

Mas antes será feito o tratamento de dados dentro da classe.

Verificando o resultado

Vamos chamar nossa classe e acompanhar o resultado final.

```
Base_Dados['Classe'] = Calculo( Base_Dados, 'Valor_Imovel' ).resultado()
Base_Dados.head()
```

	Id_Venda	Valor_Imovel	Comissao_Venda	Data_Venda	Tipo_Produto	Vendedor	Classe
0	875	248862.306452	0.04	2020-02-10	Apartamento	Ronisson Lucas Calmon	acima da média
1	13	248862.306452	0.04	2021-12-24	Casa Residencial	Ronisson Lucas Calmon	acima da média
2	530	210067.000000	0.04	2020-10-18	Apartamento	Ronisson Lucas Calmon	abaixo da média
3	508	286790.000000	0.01	2021-10-23	Casa Residencial	Odemir Depieri Jr	acima da média
4	61	262132.000000	0.06	2022-08-10	Casa Residencial	Odemir Depieri Jr	acima da média

Valores foram preenchido pela média

Coluna criada com a informação