Relatório - 4 - Prática: Principais Bibliotecas e Ferramentas Python para Aprendizado de Máquina (I)

Paulo Victor Sousa de Almeida

Parte II – Pandas

01- Pandas

O pandas e uma biblioteca de Python utilizada para análise de dados, ela fornece funções e métodos muito importantes e amplamente utilizadas para realizar tarefas de análise de dados.

Importando biblioteca Pandas
Por convenção utilizaremos a referência “pd” na importação da biblioteca Pandas

```
In [10]: import numpy as np
         import pandas as pd
```

As principais funções do Pandas utilizadas no curso foram:

Series:

E uma matriz unidimensional capas de armazenar dados de tipos variados e exibi-los em colunas em uma tabela.

Dados

```
In [2]: labels = ['a', 'b', 'c' ]
```

```
In [3]: minha_lista = [10, 20, 30]
        arr = np.array([10, 20, 30])
        d = {'a':10, 'b':20, 'c':30}
```

Função Series(). Exibi colunas referente aos dados “minha_lista” e “labels” a cima.

```
In [4]: pd.Series(data=minha_lista, index = labels)

Out[4]: a    10
        b    20
        c    30
        dtype: int64
```

DataFrame:

E uma estrutura de dados bidimensional em forma de tabela, e uma das mais utilizadas da biblioteca pandas.

O Dataframe foi criado de dados gerados da função Random do Numpy, “index” se refere as linhas de (A, B, C, D e E), “columns” se refere as colunas (W, X, Y e Z).

```
In [24]: df = pd.DataFrame(np.random.randn(5, 4), index = 'A B C D E'.split(), columns='W X Y Z'.split())

In [25]: df
```

Out[25]:

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

Apagando uma coluna.
Para apagarmos uma coluna primeira criamos uma coluna “new” para utilizar como teste, a coluna “new” e a soma da “W” + “X”.

```
In [47]: df['new'] = df['W'] + df['X']

In [48]: df
```

Out[48]:

	W	X	Y	Z	new
A	2.706850	0.628133	0.907969	0.503826	3.334983
B	0.651118	-0.319318	-0.848077	0.605965	0.331800
C	-2.018168	0.740122	0.528813	-0.589001	-1.278046
D	0.188695	-0.758872	-0.933237	0.955057	-0.570177
E	0.190794	1.978757	2.605967	0.683509	2.169552

Para apagar a coluna do “DF” se utiliza a função drop() junto com os parâmetros axis = 1 e inplace = True para alterar o “DF” imediatamente. O “DF” já e exibido sem a coluna “new”.

```
In [53]: df.drop('new', axis=1, inplace=True)

In [54]: df
```

Out[54]:

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

Dropna()

E uma função do pandas que e usada para remover valores nulos “NaN” de um DF ou Series, retornando apenas as linhas ou colunas que não contem valores NaN.

```
In [12]: df.dropna() #por padrao axis = 0
```

Out[12]:

	A	B	C
0	1.0	5.0	1

```
In [13]: df.dropna(axis=1)
```

Out[13]:

	C
0	1
1	2
2	3

df.dropna(thresh = 2)
E usado para remover linhas que contem ao menos 2 valores NaN.

```
In [23]: df.dropna(thresh = 2)
```

Out[23]:

	A	B	C
0	1.0	5.0	1
1	2.0	NaN	2

Inplace = True
Quando se define o parâmetro inplace como True ao usar o método pandas as alterações são feitas diretamente no DF original em vez de se criar um novo DF com as alterações e retornar. Isso significa que não e preciso atribuir o resultado a uma nova variável para exibi-la.

DF.fillna()
E um método que preenche os valores NaN com um valor especifico.

```
In [28]: df.fillna(value='x')
```

Out[28]:

	A	B	C
0	1.0	5.0	1
1	2.0	x	2
2	x	x	3

Também podendo ser usado o parâmetro method para utilizar métodos específicos de preenchimento do valor NaN.

```
In [31]: df.fillna(method='ffill')
```

Out[31]:

	A	B	C
0	1.0	5.0	1
1	2.0	5.0	2
2	2.0	5.0	3

GroupBy:

E uma função da biblioteca pandas que permite agrupar dados com valores de uma ou mais colunas para realizar operações em cada grupo separadamente.

Criando um DataFrame.

```
In [8]: import pandas as pd
        #Criar um DataFrame
        data = {'Empresa': ['GOOG','GOOG','MSFT','MSFT','FB','FB'],
                'Nome': ['Sam','Charlie','Amy','Vanessa','Carl','Sarah'],
                'Venda': [200,120,340,124,243,350]}
```

```
In [9]: df = pd.DataFrame(data)
```

```
In [10]: df
```

Out[10]:

	Empresa	Nome	Venda
0	GOOG	Sam	200
1	GOOG	Charlie	120
2	MSFT	Amy	340
3	MSFT	Vanessa	124
4	FB	Carl	243
5	FB	Sarah	350

Agrupando coluna Empresa

```
In [13]: group = df.groupby('Empresa')
```

Usando a função sum() para somar valores vendas das empresas

```
In [31]: group.sum()
```

```
C:\Users\victo\
eGroupBy.sum is
columns which s
  group.sum()
```

Out[31]:

Empresa	Venda
FB	593
GOOG	320
MSFT	464

O método describe() fornece estatísticas descritivas reunidas em um DF

```
In [18]: group.describe()
```

Out[18]:

	Venda							
	count	mean	std	min	25%	50%	75%	max
Empresa								
FB	2.0	296.5	75.660426	243.0	269.75	296.5	323.25	350.0
GOOG	2.0	160.0	56.568542	120.0	140.00	160.0	180.00	200.0
MSFT	2.0	232.0	152.735065	124.0	178.00	232.0	286.00	340.0

Conectar DataFrame:

Para conectar DFs se usa a função concat()

```
In [21]: df1
```

Out[21]:

	A	B
0	A0	B0
1	A1	B1

```
In [22]: df2
```

Out[22]:

	A	B
2	A2	B2
3	A3	B3

```
In [23]: df3
```

Out[23]:

	A	B
4	A4	B4
5	A5	B5

```
In [24]: pd.concat([df1,df2,df3])
```

Out[24]:

	A	B
0	A0	B0
1	A1	B1
2	A2	B2
3	A3	B3
4	A4	B4
5	A5	B5

Para mesclar DFs se usa a função merge()

```
In [30]: esquerda
```

Out[30]:

	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3

```
In [31]: direita
```

Out[31]:

	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

```
In [33]: pd.merge(esquerda,direita, how='inner',on='key')
```

Out[33]:

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K2	A2	B2	C2	D2
3	K3	A3	B3	C3	D3

Para Juntar DFs se usa a função Join()

```
In [35]: esquerda
```

Out[35]:

	A	B
K0	A0	B0
K1	A1	B1
K2	A2	B2

```
In [36]: direita
```

Out[36]:

	C	D
K0	C0	D0
K2	C2	D2
K3	C3	D3

```
In [37]: esquerda.join(direita)
```

Out[37]:

	A	B	C	D
K0	A0	B0	C0	D0
K1	A1	B1	NaN	NaN
K2	A2	B2	C2	D2

```
In [39]: esquerda.join(direita, how='outer')
```

Out[39]:

	A	B	C	D
K0	A0	B0	C0	D0
K1	A1	B1	NaN	NaN
K2	A2	B2	C2	D2
K3	NaN	NaN	C3	D3

Entrada e Saida de Dados:

Read_csv()

E usado para ler dados de um arquivo csv e criar um DF a partir desses dados.

```
In [2]: df = pd.read_csv('exemplo.csv', sep=',')

In [3]: df
```

Out[3]:

	Unnamed: 0	a	b	c	d
0	0	0	1	2	3
1	1	4	5	6	7
2	2	8	9	10	11
3	3	12	13	14	15

Read_excel()

E usado para ler dados em arquivos xlsx e criar um DF a partir desses dados.

```
In [7]: df.to_csv('exemplo.csv', sep=',', decimal=',')

In [14]: df = pd.read_excel('Exemplo_Excel.xlsx')

In [15]: df
```

Out[15]:

	Unnamed: 0	a	b	c	d
0	0	0	1	2	3
1	1	4	5	6	7
2	2	8	9	10	11
3	3	12	13	14	15

Read_html()

E usado para ler dados em arquivos html e criar um DF a partir desses dados.

```
In [19]: df = pd.read_html('https://www.fdic.gov/resources/resolutions/bank-failures/failed-bank-list/')

In [20]: type(df)
Out[20]: list

In [21]: len(df)
Out[21]: 1

In [22]: df[0]
```

Out[22]:

	Bank NameBank	CityCity	StateSt	CertCert	Acquiring InstitutionAI	Closing DateClosing	FundFund
0	Heartland Tri-State Bank	Elkhart	KS	25851	Dream First Bank, N.A.	July 28, 2023	10544
1	First Republic Bank	San Francisco	CA	59017	JPMorgan Chase Bank, N.A.	May 1, 2023	10543
2	Signature Bank	New York	NY	57053	Flagstar Bank, N.A.	March 12, 2023	10540
3	Silicon Valley Bank	Santa Clara	CA	24735	First–Citizens Bank & Trust Company	March 10, 2023	10539
4	Almena State Bank	Almena	KS	15426	Equity Bank	October 23, 2020	10538
...	...	...	...	...	...	...	...
562	Superior Bank, FSB	Hinsdale	IL	32646	Superior Federal, FSB	July 27, 2001	6004
563	Malta National Bank	Malta	OH	6629	North Valley Bank	May 3, 2001	4648
564	First Alliance Bank & Trust Co.	Manchester	NH	34264	Southern New Hampshire Bank & Trust	February 2, 2001	4647
565	National State Bank of Metropolis	Metropolis	IL	3815	Banterra Bank of Marion	December 14, 2000	4646
566	Bank of Honolulu	Honolulu	HI	21029	Bank of the Orient	October 13, 2000	4645

567 rows × 7 columns