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

Igor Carvalho Marchi

Descrição da atividade

Neste card foi apresentado duas bibliotecas, sendo elas a NumPy e Pandas, onde ambas contribuem para ciência de dados e análise numérica, durante as aulas foi apresentado que o Numpy fornece o suporte para arrays multidimensionais e funções matemáticas aplicadas nos arrays. Porém já a outra biblioteca serve para manipulação e análise de dados estruturados como tabelas, por exemplo análise de uma tabela do Excel como foi apresentado durante o card.

NumPy

Array:

```
np.array(minha_lista)
```

```
array([1, 2, 3])
```

```
minha_matriz = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

```
np.array(minha_matriz)
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

Como apresentado na imagem acima o comando array tem como função gerar arrays(listas) simples onde guarda os números inseridos.

Arange:

```
np.arange (0, 10)
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
np.arange (0, 10 , 2)
```

```
array([0, 2, 4, 6, 8])
```

A imagem acima mostra o comando arange onde ele tem a função de gerar um array baseado como se fosse um for, por exemplo a primeira linha fala para ele ir da sequência numérica de 0 a 10, porém como ele não tem número do passo ele entende que o passo é 1, por isso dentro do array tem do 0 ao 9, já no outro comando mostra o número 2 no passo, ou seja ele vai andando em 2 e até completar a sequência.

Zeros:

```
:  np.zeros(3)

:  array([0., 0., 0.])
```

O comando de zeros é responsável por criar uma lista de zeros, onde é definida a quantidade de zeros.

Ones:

```
[18]: np.ones((3, 3))

[18]: array([[1., 1., 1.],
             [1., 1., 1.],
             [1., 1., 1.]])
```

O comando Ones é responsável por criar uma lista de apenas números 1, na imagem acima é possível perceber que ele criou 3 listas que guardam 3 números 1.

Eye:

```
[20]: np.eye(4)

[20]: array([[1., 0., 0., 0.],
             [0., 1., 0., 0.],
             [0., 0., 1., 0.],
             [0., 0., 0., 1.]])
```

O comando eye tem como finalidade criar uma matriz de identidade, ou seja, ele cria uma matriz onde a diagonal principal tem números 1, porém o resto da matriz tem os números 0.

Linspace:

```
[21]: np.linspace(0, 10, 2)

[21]: array([ 0., 10.])

[22]: np.linspace(0, 10, 3)

[22]: array([ 0.,  5., 10.])
```

No comando linspace o usuário deve determinar uma sequência numérica e a quantidade dos espaçamentos iguais dentro dela, na imagem acima tem dois

exemplos com a mesma sequência numérica, porém com a quantidade de espaçamentos diferentes.

Random:

```
[29]: np.random.rand(4)

[29]: array([0.73270853, 0.610071  , 0.2596268 , 0.79361699])

[30]: np.random.randn(4)

[30]: array([-0.53431149, -0.27624699,  2.91712632,  0.34349988])
```

O comando random tem como finalidade gerar um array com números aleatórios, porém quando acrescentamos outros comandos neles podemos determinar um ponto para esse número aleatório deve estar, por exemplo o random.rand são números aleatórios que estão entre 0 e 1, já o random.randn são números aleatórios entre -1 e 0, existe bem mais comando para combinar como o random.

Reshape:

```
array

array([0.41203844, 0.50879582, 0.80141425, 0.55927561, 0.7016656 ,
       0.06413814, 0.54023816, 0.48383437, 0.82183691, 0.08859371,
       0.58882535, 0.07315406, 0.02243744, 0.87470578, 0.31339313,
       0.61725542, 0.34170806, 0.322724  , 0.61260991, 0.49450597])

array.reshape(5,4)

array([[0.41203844, 0.50879582, 0.80141425, 0.55927561],
       [0.7016656 , 0.06413814, 0.54023816, 0.48383437],
       [0.82183691, 0.08859371, 0.58882535, 0.07315406],
       [0.02243744, 0.87470578, 0.31339313, 0.61725542],
       [0.34170806, 0.322724  , 0.61260991, 0.49450597]])

array.reshape(2,10)

array([[0.41203844, 0.50879582, 0.80141425, 0.55927561, 0.7016656 ,
        0.06413814, 0.54023816, 0.48383437, 0.82183691, 0.08859371],
       [0.58882535, 0.07315406, 0.02243744, 0.87470578, 0.31339313,
        0.61725542, 0.34170806, 0.322724  , 0.61260991, 0.49450597]])
```

O comando reshape serve para reformatar a lista de arrays onde o primeiro número representa a quantidade de linhas e a segunda é a coluna.

Max e min:

```
array([[0.39265544, 0.1263744 , 0.90788995, 0.69550912, 0.56025822],
       [0.0973678 , 0.91486565, 0.69601876, 0.21176454, 0.26538792],
       [0.22394016, 0.42090031, 0.5791962 , 0.33418668, 0.02450032],
       [0.42048044, 0.8257862 , 0.45775391, 0.08449992, 0.32722693],
       [0.47327297, 0.56286675, 0.19138398, 0.9519195 , 0.04493932]])
```

```
arr.max()
```

```
0.9519195040512641
```

```
arr.min()
```

```
0.024500324129057005
```

O max tem como finalidade achar o maior número do array, porém já o min tem como finalidade achar o menor número do array.

argMax e argMin:

```
array([[0.39265544, 0.1263744 , 0.90788995, 0.69550912, 0.56025822],
       [0.0973678 , 0.91486565, 0.69601876, 0.21176454, 0.26538792],
       [0.22394016, 0.42090031, 0.5791962 , 0.33418668, 0.02450032],
       [0.42048044, 0.8257862 , 0.45775391, 0.08449992, 0.32722693],
       [0.47327297, 0.56286675, 0.19138398, 0.9519195 , 0.04493932]])
```

```
arr.argmax()
```

```
23
```

```
arr.argmin()
```

```
14
```

O argMax é responsável por retornar o índice do maior numero, já o argMin é responsável por retornar o índice do menor número.

Além das operações básicas, a biblioteca NumPy oferece suporte a cálculos matemáticos avançados utilizando arrays, como raiz quadrada, funções exponenciais, desvio padrão e funções trigonométricas. Essas operações são aplicadas de forma vetorizada e otimizada, permitindo cálculos eficientes entre múltiplos arrays. Esse conjunto de funcionalidades torna o NumPy um recurso essencial para aplicações em estatística, simulações numéricas e machine learning.

Pandas

GroupBy:

[3]:

	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

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

[6]: group

[6]: <pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000018862E90BC0>

[16]: group.sum()
```

[16]:

Empresa	Nome	Venda
FB	CarlSarah	593
GOOG	SamCharlie	320
MSFT	AmyVanessa	464

O groupBy é uma função usada para agrupar elementos de uma coleção (como uma lista ou array) com base em alguma chave de agrupamento. Na imagem acima ela agrupa as linhas do DataFrame de acordo com os valores únicos da coluna 'Empresa'.

Dentro do groupBy pode ser usados:

- Sum() calcula a soma total dos valores numéricos para cada grupo. Por exemplo, se um grupo tem duas vendas de 200 e 300, o resultado será 500.
- Mean() calcula a média (valor médio) dos valores numéricos em cada grupo. Se as vendas forem 200 e 300, a média será 250.
- Count() conta quantos registros existem em cada grupo. Ela indica, por exemplo, quantas linhas ou quantas vendas foram feitas por grupo. Importante: ela não soma valores, só conta quantos itens existem.
- Max() retorna o maior valor dentro de cada grupo. Se as vendas de uma empresa forem 200 e 350, o max() será 350.

Dropna:

```
d = {'A':[1,2,np.nan],'B':[5,np.nan,np.nan],'C':[1,2,3]}
```

```
df  = pd.DataFrame(d)
```

```
df
```

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

```
df.dropna()
```

	A	B	C
0	1.0	5.0	1

O comando dropna removeu as linhas que tinham os valores NaN(vazios).

Ffill e Bfill:

```
df
```

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

```
df.bfill()
```

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

```
df
```

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

```
df.ffill()
```

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

Eles são os novos comandos do parâmetro fillna() que não está mais funcionando, o Bfill preenche NaN com o valor seguinte, já o Ffill Preenche NaN com o valor anterior.

O Pandas é uma biblioteca essencial para ciência de dados, pois permite importar dados de vários formatos (CSV, Excel, JSON, APIs) e facilita a manipulação e análise eficiente desses dados para projetos de machine learning e análise.

Conclusões

Neste relatório, foi possível compreender a importância das bibliotecas NumPy e Pandas no contexto da ciência de dados e aprendizado de máquina. A NumPy se destacou por fornecer estruturas eficientes para manipulação de arrays multidimensionais e operações matemáticas otimizadas, essenciais para cálculos numéricos complexos. Já o Pandas mostrou-se indispensável para a manipulação e análise de dados estruturados, facilitando a organização, limpeza e agrupamento de informações, além da integração com diversas fontes de dados.

Referencias

Introdução ao NumPy

Disponível em: http://www.opl.ufc.br/post/numpy/Tutorial_numpy.pdf

Acesso em: 15 jul. 2025.

Introdução ao Pandas

Disponível em: https://medium.com/tech-grupozap/introdu%C3%A7%C3%A3o-a-biblioteca-pandas-89fa8ed4fa38

Acesso em: 15 jul. 2025.