Mundo Tech

PREPARAÇÃO E ANÁLISE DE DADOS COM A BIBLIOTECA PANDAS

SENAI <LAB365>

SUMÁRIO

MAS, ANTES DE COMEÇAR...

Para resolver esse desafio, você recebeu a planilha em csv com o Dataset do Enem. Para conferir esses arquivos, você deve retornar ao Moodle e baixar o arquivo: **Dataset_Enem_2019** e iniciar suas análises.

Aqui você irá trabalhar com formas básicas de buscas de informações em *Data Frames*, a alteração de elementos, tais como: renomear uma coluna, excluir colunas e linhas, filtrar e ordenar os dados por diferentes colunas.

Além disso, outro recurso muito importante para os analistas de dados são as possibilidades do uso de funções da estatística descritiva para a realização da análise exploratória. Assim, neste material, você conhecerá esses recursos do Pandas e posteriormente irá praticá-los.

Vamos explorar essas possibilidades.

SELECIONANDO ELEMENTOS

Após carregar um *DataSet*, é muito comum selecionarmos elementos específicos para nossas análises. Para isso, a biblioteca Pandas fornece diferentes elementos para essa tarefa.

A primeira função para a busca é **.loc**, que é utilizada quando precisamos selecionar alguma linha específica em nosso *Data Frame*. No exemplo, a seguir, estamos solicitando a localização do registro número 20. Desta forma, ele nos apresenta todos os dados desta linha específica.

```
df.loc[20]
```

```
NOTA_CN                370.8
NOTA_CH                430.2
NOTA_LC                477.7
NOTA_MT                534.7
NOTA_REDACAO           380.0
NOTA_MEDIA            453.35
TP_ESCOLA            Pública
IDADE                     18
SEXO                       F
ESTADO_CIVIL               1
COR_RACA                   3
TP_ENSINO                1.0
TREINEIRO                  0
UF_ESC                    PA
ESTUDO_PAI                 E
ESTUDO_MAE                 D
PESSOAS_RESIDENCIA         3
RENDA_MENSAL               C
Name: 20, dtype: object
```

Também é possível, além selecionar uma linha e uma coluna específica, escolher somente o elemento que desejamos. No exemplo, a seguir, realizamos a busca do valor da média somente do registro 50.

```
df.loc[50,'NOTA_MEDIA']
```

```
523.825
```

É possível também passar um conjunto de linhas. Por exemplo, a seguir, selecionamos a média somente das posições 10 até 15.

```
df.loc[10:15,'NOTA_MEDIA']
```

```
10    602.875
11    603.725
12    576.125
13    450.400
14    464.675
15    590.875
Name: NOTA_MEDIA, dtype: float64
```

ALTERANDO ELEMENTOS

Se for necessário, podemos alterar os valores de elementos do nosso *Data Frame*. Para isso, utilizamos o comando de busca **.loc** e atribuímos um novo valor. Imagine que o sexo do aluno do registro 15 foi cadastrado errado e precisaremos realizar a correção. A seguir, apresentamos o processo realizado para esta alteração.

`df.loc[15,'SEXO']`
`'M'`
Valor Original do Sexo = M
	`df.loc[15,'SEXO'] = "F"`
	Realizada alteração para F
		`df.loc[15,'SEXO']`
		`'F'`
		Valor do Sexo após correção = F

Renomeando Colunas

Muitas vezes, o nome de uma coluna pode não estar muito claro e precisamos renomeá-lo. Neste caso, utilizamos a função **.rename().** Ao utilizar essa função, devemos informar o nome da coluna atual e o novo nome, conforme demonstrado no exemplo, a seguir.

Nomes Originais

```
df[['IDADE','UF_ESC']]
```

	IDADE	UF_ESC
0	18	PA
1	26	PA

Realizada alteração nomes das colunas

```
df = df.rename(columns={'IDADE': 'IDADE_ALUNO', 'UF_ESC': 'UF_ESCOLA'})
```

Novos Nomes

```
df[['IDADE_ALUNO','UF_ESCOLA']]
```

	IDADE_ALUNO	UF_ESCOLA
0	18	PA
1	26	PA

Excluindo Colunas

Existem diferentes formas de excluir uma coluna do nosso *Data Frame.* Uma das mais utilizadas é a função .drop(). Acompanhe, a seguir, um exemplo em que o comando exclui do *Data Frame* a coluna Treineiro. Note que agora temos somente 17 colunas no *Data Frame*.

```
df = df.drop(columns=['TREINEIRO'])
```

909194 rows × 17 columns

É importante você saber que o ***drop*** não é definitivo, isto é, no arquivo CSV, a coluna permanece. Desta forma, basta você carregar novamente o DataSet através do comando read_csv() que a coluna retornará.

Excluindo Linhas

Da mesma forma, para exclusão da linha, utilizaremos o mesmo comando .drop(), porém agora referenciamos o número da linha a ser excluída. No caso do exemplo, realizamos a exclusão da linha 2.

```
df = df.drop(2)
```

	NOTA_CN	NOTA_CH
0	574.2	538.7
1	369.7	398.2
3	412.5	417.2

REALIZANDO FILTROS

É muito comum realizar as buscas utilizando filtros. Eles são úteis para respondermos questões específicas, como: Quais alunos tiveram uma nova média maior que 500? Quais são os alunos abaixo de 18 anos? Ou, então, desejo selecionar apenas os alunos das escolas públicas. Para isso, continuamos utilizando a função **.loc**, incluindo o parâmetro de comparação desejado, conforme demonstração a seguir.

Busca Original	`df` 909194 rows × 18 columns
Busca Registros Média maior 500	`df.loc[df['NOTA_MEDIA'] > 500]` 445370 rows × 18 columns

Busca Registros Idade menor 18	`df.loc[df['IDADE_ALUNO'] < 18]` 447578 rows × 18 columns
Busca Registros Tipo Escola Pública	`df.loc[df['TP_ESCOLA'] == "Pública"]` 447578 rows × 18 columns

Mas, se precisamos selecionar todos os alunos das escolas de Santa Catarina OU do Paraná? Ou, se desejamos saber quantos os alunos tiraram nota maior que 500 na redação e se eles são de escola pública?

Nestes casos, precisaremos utilizar mais de um filtro simultaneamente. Para isso, temos o operador OU, representado por '|'; e o operador E, representado por '&'. Vamos ver como resolver isso?

Busca Original	`df` 909194 rows × 18 columns
Busca alunos das escolas de Santa Catarina OU Paraná	`df[(df['UF_ESCOLA'] == "SC") | (df['UF_ESCOLA'] == "PR")]` 68960 rows × 18 columns
Busca alunos que tiraram nota maior que 500 na redação e se são de escola pública	`df[(df['NOTA_REDACAO'] > 500) & (df['TP_ESCOLA'] == "Pública")]` 551609 rows × 18 columns

Dica

Caso você queira filtrar elementos que sejam diferentes de algum valor, deve usar '!='.

ORDENANDO OS DADOS

Podemos também organizar o nosso *Data Frame* ordenando-o por diferentes colunas. Neste caso, utilizamos a função .sort_values(). Por exemplo, caso queira ordenar os dados da menor média para a maior média.

Ordena da menor para maior	`df.sort_values(by='NOTA_MEDIA')`
Ordena da maior para menor	`df.sort_values(by='NOTA_MEDIA', ascending=False)`

REALIZANDO ANÁLISE EXPLORATÓRIA COM PANDAS

Chegamos a um momento muito importante: a realização dos cálculos estatísticos. Para isso, a biblioteca Pandas nos oferece algumas funções que tornam nossa jornada rápida e simples.

As principais funções da estatística descritiva que utilizamos para a Análise Exploratória são:

df.mean()	Retorna a média de todas as colunas.
df.corr()	Retorna a correlação entre as colunas de um *Data Frame.*
df.count()	Retorna o número de valores não nulos em cada coluna do *Data Frame*.
df.max()	Retorna o maior valor em cada coluna.
df.min()	Retorna o menor valor em cada coluna.
df.median()	Retorna a mediana de cada coluna.
df.std()	Retorna o desvio padrão de cada coluna.

Outra função muito interessante é o .describe(), que apresenta um resumo de todos os dados numéricos de nosso *Data Frame*. Em resumo, as informações são:

Medidas de posição: quantidade de registros (count), valor médio (mean), menor valor (min), maior valor (max).

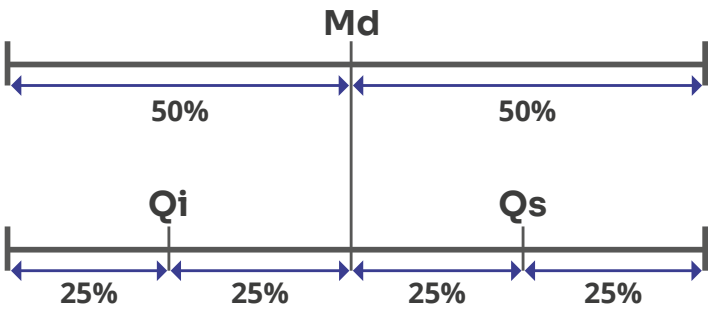
Medidas de dispersão: desvio padrão (std), quartil inferior (25%), quartil superior (75%), quartil do meio ou mediana (50%).

```
df.describe()
```

	NOTA_CN	NOTA_CH	NOTA_LC	NOTA_MT	NOTA_REDACAO	NOTA_MEDIA	IDADE_ALUNO	ESTADO_CIVIL	COR_RACA	TP_ENSINO	PESSOAS_RESIDENCIA
count	909193.000000	909193.000000	909193.000000	909193.000000	909193.000000	909193.000000	909193.000000	909193.000000	909193.000000	909193.000000	909193.000000
mean	475.507856	507.890279	520.570855	527.234399	589.151698	507.800847	17.739127	0.993503	2.084586	1.002577	4.075015
std	75.580747	80.171542	63.489450	108.960370	188.870976	71.051400	1.512293	0.236926	1.027948	0.050699	1.347421
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	2.000000	0.000000	0.000000	1.000000	1.000000
25%	415.200000	449.200000	483.300000	439.700000	500.000000	452.600000	17.000000	1.000000	1.000000	1.000000	3.000000
50%	467.700000	511.500000	526.500000	505.900000	600.000000	498.025000	18.000000	1.000000	2.000000	1.000000	4.000000
75%	531.600000	566.300000	565.500000	601.800000	700.000000	556.125000	18.000000	1.000000	3.000000	1.000000	5.000000
max	853.500000	835.100000	801.700000	985.500000	1000.000000	818.525000	70.000000	4.000000	5.000000	2.000000	20.000000

A mediana divide o conjunto de dados em duas partes. Já os quartis dividem o conjunto de dados em quatro partes iguais.

O primeiro quartil, ou quartil inferior (Qi), é o valor do conjunto que delimita os 25% menores valores: 25% dos valores são menores do que Qi e 75% são maiores do que Qi. O segundo quartil, ou quartil do meio, é a própria mediana (Md), que separa os 50% menores dos 50% maiores valores. O terceiro quartil, ou quartil superior (Qs), é o valor que delimita os 25% maiores valores: 75% dos valores são menores do que Qs e 25% são maiores do que Qs.



Neste material, foi possível conhecer mais algumas funções do Pandas, funcionalidades para buscas dos dados, alteração de elementos, ordenação e também a análise exploratória. Agora é com você. Pratique sempre!

REFERÊNCIAS

WES MCKINNEY AND THE PANDAS DEVELOPMENT TEAM. **Pandas**: powerful Python data analysis toolkit Release 1.4.4. Disponível em: https://pandas.pydata.org/pandas-docs/stable/pandas.pdf. Acesso em: 08 de ago. 2022.

SENAI <LAB365>