Pandas

Ferramentas para Data Science

Carolina Ribeiro Xavier
CCOMP - UFSJ

Índice

O que é Pandas?

- Pandas é uma biblioteca open-source para análise e manipulação de dados.
- Facilita o trabalho com dados estruturados como tabelas.
- Criada para manipular grandes volumes de dados.

Por que usar Pandas?

- Pandas é amplamente usada em ciência de dados.
- Possui uma interface intuitiva para trabalhar com dados tabulares.
- É altamente integrada com bibliotecas como NumPy e Matplotlib.

Principais funcionalidades

- Manipulação de dados tabulares (DataFrames e Series).
- Importação/exportação de dados (CSV, Excel, json, ..).
- Transformações, filtragens e agregações de dados.
- Tratamento de dados ausentes e duplicados.
- entre outras ...

Instalação e importação

- pip3 install pandas
- import pandas as pd

Estruturas de dados principais

- **Series**: Um array unidimensional com rótulos.
- **DataFrame**: Uma tabela bidimensional com rótulos nas linhas e colunas.

Exemplo de Series

Sintaxe:

```
pd.Series(data, index=index)
```

Exemplo:

```
import pandas as pd
s = pd.Series([1,2,3,4],index=['a','b','c','d'])
print(s)
```

```
a    1
b    2
c    3
d    4
dtype: int64
```

Exemplo de DataFrame

Sintaxe:

```
pd.DataFrame(data, columns=colunas)
```

Exemplo:

```
data = {'Nome': ['Ana', 'Bruno', 'Carla'],
        'Idade': [23, 35, 28]}
df = pd.DataFrame(data)
print(df)
```

```
    Nome  Idade
0    Ana     23
1  Bruno     35
2  Carla     28
```

Selecionando colunas

Sintaxe:

```
df['coluna']
```

Exemplo:

```
df['Nome']
```

será um objeto do tipo Series

Filtrar linhas

Sintaxe:

```
df[condicao]
```

Exemplo:

```
df[df['Idade'] > 30]

df[['col1', 'col2']]  # Seleciona mais colunas
df.loc[linha]  # Seleciona uma linha
```

Retorna o dataframe filtrado, se quiser manter somente as linhas filtradas você deve atribuir a uma nova variável.

Filtrar linhas

operadores lógicos e relacionais podem compor os filtros.

```
df[(df['A'] > 2) & (df['B'] < 9)]
resultado = df[df['B'].isin(['x', 'y'])]
resultado = df[df['A'].str.contains('ba')]
```

Adicionar nova coluna

Sintaxe:

```
df['nova_coluna'] = valores
```

Exemplo:

```
df['Salario'] = [2000, 3000, 4000]
```

Os valores serão atribuídos a cada linha existente do dataframe, logo devemos ter um array com o mesmo número de valores do que as linhas do dataframe.

Remover coluna

Sintaxe:

```
df.drop('coluna', axis=1)
```

Reorganização de Colunas

- O Pandas permite reorganizar colunas facilmente.
- Reorganização pode ser útil para análise e apresentação de dados.

Método `.reindex()`

Sintaxe:

```
df = df.reindex(columns=['col3', 'col1', 'col2'])
```

- `.reindex()` permite reorganizar colunas na ordem desejada.

loc e iloc

Outra forma para reordenar colunas em um DataFrame do pandas é usar os métodos loc e iloc, que são tradicionalmente usados para indexação. Esses métodos também podem reordenar o DataFrame especificando a ordem das colunas ao selecionar um subconjunto de colunas. Aqui está um exemplo:

```
## Usando loc
df = df.loc[:, ['B', 'A', 'C']]

## Usando iloc
df = df.iloc[:, [1, 0, 2]]

print(df)
```

No primeiro caso, loc é usado com uma lista de nomes de colunas para reordenar o DataFrame. No segundo caso, iloc usa indexação baseada em números inteiros para especificar a nova ordem das colunas.

Remover valores nulos

Sintaxe:

```
df.dropna()
```

Substituir valores nulos

Sintaxe:

```
df.fillna(valor)
```

Exemplo:

```
df.fillna(0)
```

Remover duplicatas

Sintaxe:

```
df.drop_duplicates()
```

Leitura de arquivos CSV

Sintaxe:

```
pd.read_csv('caminho_do_arquivo', sep=',')
```

Exemplo:

```
df = pd.read_csv('dados.csv')
```

Diferentes separadores de CSV

Separador vírgula (padrão):

```
df = pd.read_csv('dados.csv', sep=',')
```

Separador ponto e vírgula:

```
df = pd.read_csv('dados.csv', sep=';')
```

Separador tabulação:

```
df = pd.read_csv('dados.csv', sep='\t')
```

Escrever dados em um arquivo CSV

Sintaxe:

```
df.to_csv('caminho_do_arquivo.csv')
```

Exemplo:

```
df.to_csv('saida.csv')
```

Proposta Prática

- Trabalhar com um dataset real (Titanic).
- Realizar manipulações e análises de dados.
- Praticar leitura, filtragem, agregação e exportação de dados.

Passo 1: Carregar o dataset

```
df = pd.read_csv('titanic.csv')
```

Passo 2: Visualizar dados

Sintaxe:

```
df.head()
```

Passo 3: Análise Exploratória

Exemplo:

- Quantos passageiros a bordo?
- Quantos sobreviveram?
- Quantas mulheres estavam na primeira classe?

Passo 5: Completando Dados vazios

Sintaxe:

```
df['coluna'].fillna(valor, inplace=True)
```

Exemplo:

```
df['Age'].fillna(df['Age'].mean(), inplace=True)
```

Passo 6: Agrupamento

Sintaxe:

```
df.groupby('coluna')['coluna_agregada'].funcao()
```

Exemplo:

```
df.groupby('Pclass')['Age'].mean()
```

Passo 7: Exportar Dados

Sintaxe:

```
df.to_csv('caminho_do_arquivo.csv', index=False)
```

Exemplo:

```
df[df['Pclass'] == 1].to_csv('primeira_classe.csv',
```

Exercício

Utilizando o dataset do Titanic, siga as etapas abaixo para realizar uma análise detalhada sobre a sobrevivência de diferentes grupos de passageiros. Sua tarefa é identificar padrões de sobrevivência baseados em diversas características dos passageiros.

- Distribuição de Idade por Gênero e Classe Social
- Taxa de Sobrevivência por Faixa Etária
- Relação entre Tarifa e Sobrevivência
- Tente fazer alguns gráficos

Dúvidas?