Introdução ao Pandas para Análise de Dados

Tempo estimado: 10 Minutos

Objetivo:

1. Aprender o que são as Séries do Pandas e como criá-las.
2. Entender como acessar e manipular dados dentro de uma Série.
3. Descobrir o básico sobre como criar e trabalhar com DataFrames do Pandas.
4. Aprender como acessar, modificar e analisar dados em DataFrames.
5. Obter insights sobre atributos e métodos comuns de DataFrames.

O que é Pandas?

Pandas é uma biblioteca popular de manipulação e análise de dados de código aberto para a linguagem de programação Python. Ela fornece um conjunto poderoso e flexível de ferramentas para trabalhar com dados estruturados, tornando-se uma ferramenta fundamental para cientistas de dados, analistas e engenheiros. Pandas é projetado para lidar com dados em vários formatos, como dados tabulares, séries temporais e mais, tornando-se uma parte essencial do fluxo de trabalho de processamento de dados em muitas indústrias.

Aqui estão algumas **principais características e funcionalidades do Pandas**:

Estruturas de Dados: Pandas oferece duas estruturas de dados principais - DataFrame e Series.

1. Um DataFrame é uma estrutura de dados tabular bidimensional, de tamanho mutável e potencialmente heterogênea, com eixos rotulados (linhas e colunas).
2. Uma Series é um array unidimensional rotulado, essencialmente uma única coluna ou linha de dados.

Importação e Exportação de Dados: Pandas facilita a leitura de dados de várias fontes, incluindo arquivos CSV, planilhas do Excel, bancos de dados SQL e mais. Ele também pode exportar dados para esses formatos, permitindo uma troca de dados sem costura.

Mesclagem e Junção de Dados: Você pode combinar múltiplos DataFrames usando métodos como merge e join, semelhantes às operações SQL, para criar conjuntos de dados mais complexos a partir de diferentes fontes.

Indexação Eficiente: Pandas fornece métodos de indexação e seleção eficientes, permitindo que você acesse rapidamente linhas e colunas específicas de dados.

Estruturas de Dados Personalizadas: Você pode criar estruturas de dados personalizadas e manipular dados de maneiras que atendam às suas necessidades específicas, ampliando as capacidades do Pandas.

Importando Pandas:

Importe o Pandas usando o comando de importação, seguido pelo nome da biblioteca. Comumente, o Pandas é importado como pd para encurtar o código.

```
import pandas as pd
```

Carregamento de Dados:

- O Pandas pode ser usado para carregar dados de várias fontes, como arquivos CSV e Excel.
- A função read_csv é usada para carregar dados de um arquivo CSV em um DataFrame do Pandas.

Para ler um arquivo CSV (Comma-Separated Values) em Python usando a biblioteca Pandas, você pode usar a função pd.read_csv(). Aqui está a sintaxe para ler um arquivo CSV:

```
import pandas as pd
# Read the CSV file into a DataFrame
df = pd.read_csv('your_file.csv')
```

Substitua 'your_file.csv' pelo caminho real do seu arquivo CSV. Certifique-se de que o arquivo esteja localizado no mesmo diretório que seu script Python, ou forneça o caminho correto do arquivo.

O que é uma Série?

Uma Série é um array unidimensional rotulado em Pandas. Pode ser pensado como uma única coluna de dados com rótulos ou índices para cada elemento. Você pode criar uma Série a partir de várias fontes de dados, como listas, arrays NumPy ou dicionários.
Aqui está um exemplo básico de criação de uma Série em Pandas:

```
import pandas as pd
# Create a Series from a list
data = [10, 20, 30, 40, 50]
s = pd.Series(data)
print(s)
```

Neste exemplo, criamos uma Série chamada **s** com dados numéricos. Note que o Pandas atribuiu automaticamente índices numéricos (0, 1, 2, 3, 4) a cada elemento, mas você também pode especificar rótulos personalizados, se necessário.

Acessando Elementos em uma Série

Você pode acessar elementos em uma Série usando os rótulos de índice ou posições inteiras. Aqui estão alguns métodos comuns para acessar dados de Série:

Acessando por rótulo

```
print(s[2])     # Access the element with label 2 (value 30)
```

Acessando por posição

```
print(s.iloc[3]) # Access the element at position 3 (value 40)
```

Acessando múltiplos elementos

```
print(s[1:4])   # Access a range of elements by label
```

Atributos e Métodos da Série

As Séries do Pandas vêm com vários atributos e métodos para ajudar você a manipular e analisar dados de forma eficaz. Aqui estão alguns essenciais:

- **values**: Retorna os dados da Série como um array NumPy.
- **index**: Retorna o índice (rótulos) da Série.
- **shape**: Retorna uma tupla representando as dimensões da Série.
- **size**: Retorna o número de elementos na Série.
- **mean(), sum(), min(), max()**: Calcula estatísticas resumidas dos dados.
- **unique(), nunique()**: Obtém valores únicos ou o número de valores únicos.
- **sort_values(), sort_index()**: Ordena a Série por valores ou rótulos de índice.
- **isnull(), notnull()**: Verifica a presença de valores ausentes (NaN) ou não ausentes.
- **apply()**: Aplica uma função personalizada a cada elemento da Série.

O que é um DataFrame?

Um DataFrame é uma estrutura de dados bidimensional rotulada, com colunas de diferentes tipos de dados potenciais. Pense nisso como uma tabela onde cada coluna representa uma variável e cada linha representa uma observação ou ponto de dados. DataFrames são adequados para uma ampla gama de dados, incluindo dados estruturados de arquivos CSV, planilhas do Excel, bancos de dados SQL e muito mais.

Criando DataFrames a partir de Dicionários:

DataFrames podem ser criados a partir de dicionários, com chaves como rótulos de coluna e valores como listas representando linhas.

```
import pandas as pd
# Creating a DataFrame from a dictionary
data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],
        'Age': [25, 30, 35, 28],
        'City': ['New York', 'San Francisco', 'Los Angeles', 'Chicago']}
df = pd.DataFrame(data)
print(df)
```

Seleção de Colunas:

Você pode selecionar uma única coluna de um DataFrame especificando o nome da coluna entre colchetes duplos. Várias colunas podem ser selecionadas de maneira semelhante, criando um novo DataFrame.

```
print(df['Name'])  # Access the 'Name' column
```

Acessando Linhas:

Você pode acessar linhas pelo seu índice usando .iloc[] ou pelo rótulo usando .loc[].

```
print(df.iloc[2])   # Access the third row by position
print(df.loc[1])    # Access the second row by label
```

Fatiamento:

Você pode fatiar DataFrames para selecionar linhas e colunas específicas.

```
print(df[['Name', 'Age']])  # Select specific columns
print(df[1:3])              # Select specific rows
```

Encontrando Elementos Únicos:

Use o método unique para determinar os elementos únicos em uma coluna de um DataFrame.

```
unique_dates = df['Age'].unique()
```

Filtragem Condicional:

Você pode filtrar dados em um DataFrame com base em condições usando operadores de desigualdade. Por exemplo, você pode filtrar álbuns lançados após um determinado ano.

```
high_above_102 = df[df['Age'] > 25]
```

Salvando DataFrames:

Para salvar um DataFrame em um arquivo CSV, use o método to_csv e especifique o nome do arquivo com a extensão “.csv”. O Pandas oferece outras funções para salvar DataFrames em diferentes formatos.

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

Atributos e Métodos do DataFrame

DataFrames oferecem numerosos atributos e métodos para manipulação e análise de dados, incluindo:

- **shape**: Retorna as dimensões (número de linhas e colunas) do DataFrame.
- **info()**: Fornece um resumo do DataFrame, incluindo tipos de dados e contagens de valores não nulos.
- **describe()**: Gera estatísticas resumidas para colunas numéricas.
- **head(), tail()**: Exibe as primeiras ou últimas n linhas do DataFrame.
- **mean(), sum(), min(), max()**: Calcula estatísticas resumidas para colunas.
- **sort_values()**: Ordena o DataFrame por uma ou mais colunas.
- **groupby()**: Agrupa dados com base em colunas específicas para agregação.
- **fillna(), drop(), rename()**: Lida com valores ausentes, remove colunas ou renomeia colunas.
- **apply()**: Aplica uma função a cada elemento, linha ou coluna do DataFrame.

  O Pandas oferece uma ampla gama de métodos além desses exemplos. Para informações mais detalhadas, consulte a documentação oficial disponível no site oficial do Pandas.

Conclusão

Em conclusão, dominar o uso de Pandas Series e DataFrames é essencial para uma manipulação e análise de dados eficaz em Python. As Series fornecem uma base para lidar com dados unidimensionais com rótulos, enquanto os DataFrames oferecem uma estrutura versátil, semelhante a uma tabela, para trabalhar com dados bidimensionais. Seja limpando, explorando, transformando ou analisando dados, essas estruturas de dados do Pandas, juntamente com seus atributos e métodos, permitem que você manipule dados de forma eficiente e flexível para obter insights valiosos. Ao incorporar Series e DataFrames em seu conjunto de ferramentas de ciência de dados, você estará bem preparado para enfrentar uma ampla gama de tarefas relacionadas a dados e aprimorar suas capacidades de análise de dados.
Para aprimorar suas habilidades em análise de dados com Pandas, considere os seguintes próximos passos:

Prática:

Trabalhe com conjuntos de dados reais para aplicar o que você aprendeu e ganhar experiência prática.

Explorar Documentação:

Visite o site oficial do Pandas para explorar a documentação extensa e descobrir mais funções e métodos.

Author

Akansha Yadav

Changelog

Data	Versão	Alterado por	Descrição da Mudança
2023-10-02	1.0	Akansha Yadav	Criado Leitura