UFCD 5417
Laboratório 8 (pandas)
v1.2

Nelson Santos
nelson.santos.0001376@edu.atec.pt

ATEC — 28 de novembro de 2024

Conteúdo

1 Introdução

O pandas é uma biblioteca indispensável em Python para a análise e manipulação de dados, amplamente reconhecida pela sua eficiência no processamento de grandes volumes de informação. Com estruturas de dados poderosas, como o Dataframe e o Series, o pandas permite a manipulação de milhões de registos com facilidade, tornando-se ideal para tarefas complexas em *data science* e automação de processos.

Empresas de renome, como a Google, Netflix e Airbnb, utilizam o pandas para análises de dados, visualizações avançadas e apoio à tomada de decisões. Este laboratório tem como objetivo explorar as principais funcionalidades do pandas, ilustrando com exemplos práticos como utilizar esta ferramenta para transformar, analisar e visualizar dados de forma eficiente.

2 Estruturas de dados do pandas

2.1 Series

Uma Series é uma estrutura de dados uni-dimensional que pode conter qualquer tipo de dados, semelhante a uma lista, mas com a capacidade de utilizar rótulos para indexação.

Exemplo de criação de uma Series:

```
import pandas as pd

# Criar uma Series com rotulos personalizados
dados = pd.Series([10, 20, 30], index=['a', 'b', 'c'])
print(dados)
```

No exemplo acima, é criada uma Series contendo os valores 10, 20 e 30, com rótulos associados aos índices.

2.2 Dataframe

Um Dataframe é uma estrutura de dados bi-dimensional, semelhante a uma tabela de uma base de dados, com linhas e colunas, onde cada coluna pode ter um tipo de dado diferente.

Exemplo de Criação de um DataFrame:

```
# Criar um DataFrame a partir de um dicionario
dados = {'Produto': ['Produto A', 'Produto B', 'Produto C'],
         'Preco': [10.5, 22.3, 19.99],
         'Quantidade': [5, 3, 10]}
df = pd.DataFrame(dados)
print(df)
```

Neste exemplo, o Dataframe é criado a partir de um dicionário, onde as chaves correspondem aos nomes das colunas, e os valores são listas com os dados de cada coluna.

3 Manipulação de dados

3.1 Seleção de dados

Os dados num DataFrame podem ser selecionados utilizando o rótulo das colunas ou o índice das linhas. Existem diversas formas de realizar a seleção, tais como loc e iloc.

Exemplo de seleção de uma coluna com loc:

```
# Selecionar a coluna "Produto"
produtos = df['Produto']
print(produtos)

# Selecionar uma linha especifica utilizando o rotulo
linha_b = df.loc[1]
print(linha_b)
```

loc permite selecionar dados com base nos rótulos, enquanto iloc utiliza índices inteiros para seleção.

3.2 Filtros de dados

Os filtros permitem restringir os dados de acordo com uma condição. É possível filtrar um `DataFrame` com base nos valores das suas colunas.

Exemplo da execução de um filtro simples:

```
# Filtrar os produtos cujo preco superior a 20
filtro = df[df['Preco'] > 20]
print(filtro)
```

O resultado do filtro acima será um novo `DataFrame` contendo apenas as linhas onde o preço é superior a 20.

3.3 Manipulação de dados em falta

Os dados em falta são comuns em *datasets* reais, e o `pandas` fornece métodos para lidar com esses valores de forma eficiente. O método `isnull()` permite verificar a presença de dados em falta, enquanto `fillna()` pode ser utilizado para substituir esses valores:

Exemplo de verificação e substituição de dados em falta:

```
# Criar um DataFrame com dados em falta
dados = {'Produto': ['Produto A', 'Produto B', 'Produto C'],
         'Preco': [10.5, None, 19.99],
         'Quantidade': [5, 3, None]}
df = pd.DataFrame(dados)

# Verificar dados em falta
print(df.isnull())

# Substituir dados em falta
df_filled = df.fillna(0)
print(df_filled)
```

O método `fillna()` permite substituir valores em falta por um valor específico, neste caso, por zero.

4 Operações avançadas

4.1 Agrupamento de dados

O `pandas` permite agrupar os dados de acordo com os valores de uma coluna, de modo a aplicar funções agregadoras.

Exemplo de agrupamento e soma:

```
# Agrupar dados por "Produto" e somar as quantidades
grupo = df.groupby('Produto').sum()
print(grupo)
```

O método `groupby()` é utilizado para agrupar os dados, seguido de uma função agregadora, como `sum()`.

4.2 Junção de `Dataframe`

A junção de `Dataframe` permite combinar dados de diferentes fontes, semelhante a uma junção de tabelas em SQL.

Exemplo de junção de `Dataframe`:

```
# Criar dois DataFrames
df1 = pd.DataFrame({'Produto': ['Produto A', 'Produto B'],
                    'Preco': [10.5, 22.3]})
df2 = pd.DataFrame({'Produto': ['Produto A', 'Produto B'],
                    'Quantidade': [5, 3]})

# Juntar os DataFrames com base na coluna "Produto"
df_merged = pd.merge(df1, df2, on='Produto')
print(df_merged)
```

O método `merge()` é utilizado para realizar junções de `DataFrames`, unindo-os com base numa coluna em comum, onde no exemplo a junção ocorreu com base na coluna `Produto`.

4.3 Operações matemáticas em colunas

No pandas, é possível realizar operações matemáticas entre colunas de um Dataframe, como somar, subtrair, multiplicar ou dividir valores entre colunas, ou até adicionar novas colunas baseadas em cálculos.

Exemplo de adição de uma coluna com soma e multiplicação:

```
# Preencher valores NaN em 'Preco' e 'Quantidade' com 0 para evitar NaN no resultado
df['Valor_Total'] = df['Preco'].fillna(0) * df['Quantidade'].fillna(0)

# Adicionar uma nova coluna "Soma_Preco_Quantidade" com a soma de 'Preco' e 'Quantidade',
    tratando NaNs
df['Soma_Preco_Quantidade'] = df['Preco'].fillna(0) + df['Quantidade'].fillna(0)

print(df)
```

Neste exemplo, foram adicionadas duas novas colunas: Valor_Total, que é o resultado da multiplicação entre o preço e a quantidade, e Soma_Preco_Quantidade, que é a soma dessas mesmas colunas.

4.4 Exportação para Excel

O pandas permite exportar dados para diversos formatos, incluindo ficheiros Excel. Isto é útil quando se deseja partilhar ou guardar os dados analisados. Poderá ser necessário instalar a biblioteca openpyxl.

Exemplo de exportação para Excel:

```
# Exportar o DataFrame para um ficheiro Excel
df.to_excel('dados_produtos.xlsx', index=False)
```

O método to_excel() exporta o DataFrame para um ficheiro Excel. O argumento index=False remove o índice na exportação, ou seja, as linhas não terão numeração automática no ficheiro.

4.5 Criação de Gráficos

O pandas está integrado com a biblioteca matplotlib, o que facilita a criação de gráficos a partir dos dados de um DataFrame. Poderá ser necessário instalar a biblioteca matplotlib.

Exemplo de gráfico de barras:

```
import matplotlib.pyplot as plt

# Criar um grqfico de barras para a quantidade de produtos
df.plot(kind='bar', x='Produto', y='Quantidade', title='Quantidade de Produtos')

# Mostrar o grafico
plt.show()
```

Neste exemplo, é criado um gráfico de barras que mostra a quantidade de cada produto. O método plot() permite criar vários tipos de gráficos, e o argumento kind='bar' especifica que o gráfico deve ser de barras.

4.6 Ordenação de dados

O pandas permite ordenar os dados de um Dataframe com base nos valores de uma ou mais colunas.

Exemplo de ordenação por preço:

```
# Ordenar o DataFrame pelo valor da coluna 'Preco'
df_sorted = df.sort_values(by='Preco', ascending=False)
print(df_sorted)
```

O método sort_values() ordena os dados com base na coluna Preco, neste caso em ordem decrescente, por conta do argumento ascending=False.

4.7 Remoção de duplicados

É possível remover entradas duplicadas de um DataFrame utilizando o método drop_duplicates().

Exemplo de Remoção de duplicados:

```
# Remover linhas duplicadas no DataFrame
df_unique = df.drop_duplicates()
print(df_unique)
```

Este método remove todas as linhas que contêm valores duplicados, mantendo apenas a primeira ocorrência.

4.8 Aplicação de Funções Personalizadas

A função `apply()` permite aplicar funções personalizadas a colunas ou linhas de um `DataFrame`.

Exemplo de aplicação de função personalizada:

```
# Aplicar uma funcao que aumenta o preco em 10%
df['Preco_Ajustado'] = df['Preco'].apply(lambda x: x * 1.10)
print(df)
```

Neste exemplo:

- **Lambda**: O `lambda` é uma forma de criar funções anónimas (sem nome ou designação) em `Python`, úteis para operações rápidas e simples. A função `lambda x: x * 1.10` recebe um valor `x` e devolve o valor aumentado em 10%, ou seja, `x` multiplicado por 1.10
- **apply()**: O método `apply()` do Pandas aplica uma função a cada elemento de uma coluna ou linha de um `Dataframe`, permitindo realizar transformações personalizadas de forma flexível e eficiente

A operação realiza os seguintes passos:

1. A coluna `Preco` do `DataFrame` é passada como entrada para o método `apply()`
2. Cada valor na coluna é processado pela função `lambda`, que calcula o preço ajustado
3. O resultado é guardado numa nova coluna chamada `Preco_Ajustado`, que contém os valores transformados

Esta abordagem é particularmente útil para manipulação de dados, pois combina simplicidade e eficiência.

5 Desafios Práticos

Com base nos dados disponíveis no ficheiro `vendas.csv`, os seguintes desafios devem ser resolvidos, aplicando as técnicas aprendidas ao longo do laboratório.

5.1 Desafio 1: Agrupamento e Soma

Agrupar as vendas por produto e somar as quantidades vendidas de cada produto.

5.2 Desafio 2: Criação de Nova Coluna com Cálculos

Criar uma nova coluna chamada `Lucro`, onde o lucro seja calculado com base numa margem de 20% sobre o valor total de cada venda.

5.3 Desafio 3: Exportação para Excel

Exportar os resultados dos cálculos realizados no Desafio 1 e Desafio 2 para um ficheiro Excel.

5.4 Desafio 4: Visualização de Dados

Criar um gráfico de barras que mostre o total de vendas por produto.

5.5 Desafio 5: Ordenação de Dados

Ordenar o ficheiro de dados por valor de `Total_Venda` de forma decrescente.

5.6 Desafio 6: Remoção de Duplicados

Verificar se existem vendas duplicadas e removê-las, se necessário.

5.7 Desafio 7: Aplicação de Função Personalizada

Criar uma função que aplique um desconto de 10% em todas as vendas e adicionar uma nova coluna com os preços ajustados.

5.8 Desafio 8: Análise de séries Temporais

Explorar como o `pandas` pode ser utilizado para a análise de séries temporais.

- Utilizando o conjunto de dados, realize uma análise de séries temporais com base na coluna `Data`. Examine as vendas ao longo do tempo e responda às seguintes questões:
  - Qual é a tendência geral das vendas ao longo do tempo?
  - Existem padrões sazonais ou flutuações nos dados?

Bom trabalho!