Introdução ao Pandas e Manipulação de DataFrames

Aula 14 - UC2

Ciência de Dados

Objetivos da Aula

Compreender o Pandas

Entender o conceito de DataFrame e Series como estruturas fundamentais do Pandas, e sua importância para a manipulação de dados tabulares em Python.

Importar e Inspecionar Dados

Aprender a carregar dados de diferentes fontes (CSV, Excel) e utilizar comandos essenciais para inspeção rápida dos dados.

Manipular DataFrames

Dominar operações básicas de manipulação, incluindo seleção de colunas, criação de novas colunas e transformação de dados.

Tratar Dados Ausentes

Identificar e aplicar estratégias adequadas para lidar com valores ausentes (nulos) e dados duplicados em DataFrames.

Filtrar e Consultar Dados

Aplicar filtros simples e compostos usando operadores lógicos para extrair informações específicas dos dados.

Documentar Código

Desenvolver boas práticas de documentação e organização de código, essenciais para projetos colaborativos e sustentáveis.

O Pandas é o Excel da Ciência de Dados

> *"O Pandas é o Excel da Ciência de Dados"*

O que é o Pandas?

O **Pandas** é a biblioteca essencial em Python para transformar dados brutos em inteligência. Ela permite organizar dados complexos de forma tabular, facilitando a análise e manipulação de grandes volumes de informação.

Assim como o Excel revolucionou a forma como trabalhamos com planilhas, o Pandas revolucionou a manipulação de dados em programação, oferecendo recursos poderosos e flexíveis para cientistas de dados e analistas.

Por que usar o Pandas?

- **Estrutura tabular intuitiva:** Organiza dados em linhas e colunas, similar a planilhas, mas com muito mais poder computacional.
- **Manipulação eficiente:** Permite filtrar, transformar e agregar milhões de registros em segundos.
- **Integração com outras ferramentas:** Conecta-se facilmente com bancos de dados, arquivos CSV, Excel e APIs.
- **Análise exploratória simplificada:** Oferece funções prontas para estatísticas descritivas e visualização de dados.

DataFrame e Series: Estruturas Fundamentais

DataFrame

O **DataFrame** é a estrutura central do Pandas. Trata-se de uma tabela bidimensional com rótulos de linha (índices) e rótulos de coluna, similar a uma planilha do Excel.

Exemplo de DataFrame:

Índice	Nome	Idade	Cidade
0	Ana Silva	28	São Paulo
1	João Santos	35	Rio de Janeiro
2	Maria Costa	42	Belo Horizonte
3	Pedro Lima	31	Brasília

- **Linhas (índices):** Cada linha representa um registro ou observação.
- **Colunas:** Cada coluna representa uma variável ou característica dos dados.

Series

Uma **Series** é uma estrutura unidimensional que representa uma única coluna de um DataFrame. Cada coluna dentro de um DataFrame é, na verdade, uma Series.

Exemplo de Series (coluna "Idade"):

```
0 28
1 35
2 42
3 31
Name: Idade, dtype: int64
```

Características da Series:

- Possui um índice associado a cada valor para acesso rápido aos dados.
- Todos os valores têm o mesmo tipo de dado (int, float, string, etc.).
- Pode ser extraída de um DataFrame: `df['Nome']`

Relação entre DataFrame e Series:

Um DataFrame é uma coleção de Series que compartilham o mesmo índice. Cada coluna é uma Series independente.

Funcionalidades Básicas do Pandas

1 Importação de Dados

Carregue dados de diferentes formatos (CSV, Excel, JSON) para um DataFrame usando funções específicas do Pandas.

```
import pandas as pd

# Carregar arquivo CSV
df = pd.read_csv('dados.csv')

# Carregar arquivo Excel
df = pd.read_excel('dados.xlsx')
```

2 df.head() - Primeiras Linhas

Visualize as primeiras linhas do DataFrame para ter uma visão rápida dos dados. Por padrão, mostra as 5 primeiras linhas.

```
# Mostrar as 5 primeiras linhas
df.head()

# Mostrar as 10 primeiras linhas
df.head(10)
```

3 df.info() - Informações Gerais

Obtenha informações sobre o DataFrame, incluindo tipos de dados de cada coluna, quantidade de valores não-nulos e uso de memória.

```
df.info()

# Retorna:
# - Número de linhas e colunas
# - Nome das colunas
# - Tipo de dados (int, float, object)
# - Contagem de valores não-nulos
```

4 df.describe() - Estatísticas

Gere estatísticas descritivas para colunas numéricas, incluindo média, desvio padrão, valores mínimo e máximo, e quartis.

```
df.describe()

# Retorna para cada coluna numérica:
# - count, mean, std
# - min, 25%, 50%, 75%, max
```

Manipulação Básica de DataFrames

Seleção de Colunas

Para selecionar colunas em um DataFrame, use colchetes com o nome da coluna.

1. Selecionar uma única coluna

```
# Retorna uma Series
df['Nome']
```

O resultado é uma **Series** (estrutura unidimensional).

2. Selecionar múltiplas colunas

```
# Retorna um DataFrame
df[['Nome', 'Idade', 'Cidade']]
```

Use uma lista de nomes entre colchetes duplos.

Exemplo Prático

```
# Selecionar informações de contato
contatos = df[['Nome', 'Email']]
```

Criação de Novas Colunas

Crie novas colunas atribuindo valores ou através de cálculos.

1. Valor constante

```
df['Pais'] = 'Brasil'
```

2. Baseada em cálculo

```
df['Ano_Nascimento'] = 2024 - df['Idade']
```

3. Operações complexas

```
# Desconto de 10%
df['Preco_Desc'] = df['Preco'] * 0.9

# Concatenar strings
df['Nome_Completo'] = df['Nome'] + ' ' + df['Sobrenome']
```

Exemplo: Antes e Depois

Nome	Idade	Ano_Nascimento
Ana	28	1996
Bruno	35	1989
Carlos	42	1982

Atividade Prática 1: Setup e Primeiros Comandos

Objetivo da Atividade

Realizar o setup inicial do ambiente Python com Pandas e executar os primeiros comandos para carregar e inspecionar um dataset no **Google Colab** ou **Jupyter Notebook**.

Passo a Passo Detalhado

1 Importar a biblioteca Pandas

Comece importando o Pandas com o alias padrão "pd":

```
import pandas as pd
```

2 Carregar um dataset

Carregue um arquivo CSV usando a função read_csv():

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

3 Visualizar as primeiras linhas

Use o comando head() para ver as primeiras 5 linhas:

```
df.head()
```

```
# Saída: Tabela com as 5 primeiras linhas
```

4 Obter informações sobre o DataFrame

Use info() para ver tipos de dados e valores nulos:

```
df.info()
```

```
# Saída: Informações sobre colunas e tipos
```

5 Obter estatísticas descritivas

Use describe() para ver estatísticas das colunas numéricas:

```
df.describe()
```

```
# Saída: Média, desvio padrão, min, max
```

6 Selecionar colunas específicas

Selecione uma coluna ou múltiplas colunas:

```
# Uma coluna df['Nome'] # Múltiplas colunas
df[['Nome', 'Idade', 'Cidade']]
```

> **Dica Importante**
>
> Execute cada comando separadamente e observe os resultados. Experimente modificar os parâmetros e explorar o dataset!

Tratamento de Dados Ausentes e Duplicados

Atividade Prática 2: Consulta e Filtragem de Dados

Esta atividade foca no **raciocínio lógico aplicado à consulta de dados**. Você aprenderá a filtrar DataFrames usando condições simples e compostas.

Boas Práticas e Documentação

Organização de Scripts

Divida o código em blocos lógicos (células no Colab) e use nomes de variáveis claros e descritivos. Um código bem organizado é mais fácil de entender, manter e depurar.

```
# Bloco 1: Importação de bibliotecas
import pandas as pd

# Bloco 2: Carregamento de dados
df_vendas = pd.read_csv('vendas.csv')

# Bloco 3: Limpeza de dados
df_vendas.dropna(inplace=True)
```

Comentários Eficientes

O código deve ser lido como um livro. Comentários explicam o **porquê** das decisões complexas, não o **o quê** o código faz (isso deve ser óbvio pelo próprio código).

```
# Removemos outliers acima de 3 desvios padrão
# para evitar distorções na análise de tendências
limite = df['Valor'].mean() + 3 * df['Valor'].std()
df_limpo = df[df['Valor'] <= limite]
```

Princípios de Código Limpo

1. **Nomes descritivos:** Use nomes que revelam a intenção (ex: df_clientes_ativos ao invés de df1).
2. **Funções pequenas:** Cada função deve fazer apenas uma coisa e fazê-la bem.
3. **Evite repetição:** Não repita código. Se você copiou e colou, considere criar uma função.
4. **Formatação consistente:** Mantenha indentação e espaçamento uniformes em todo o código.
5. **Documente decisões:** Explique escolhas técnicas importantes que não são óbvias.

Responsabilidade Ética

A atenção aos detalhes no tratamento de dados é um ato de responsabilidade profissional. Dados sujos levam a conclusões equivocadas e decisões de negócio incorretas. Documentar seu processo garante transparência e reprodutibilidade.

Atividade Prática 3: Documentação Colaborativa do Código

Objetivo da Atividade

Trabalhar em **duplas ou trios** para documentar adequadamente o código desenvolvido na Atividade Prática 2, inserindo comentários explicativos e estruturando o script de forma profissional.

Passo a Passo Detalhado

1 Inserir Comentários no Código

Adicione comentários explicativos em linhas com lógica complexa:

```
# Aplicando filtro composto para encontrar
# clientes com idade > 30 E que moram em SP
clientes_sp = df[(df['Idade'] > 30) &
(df['Cidade'] == 'São Paulo')]
```

2 Criar Célula de Markdown Inicial

No início do script, crie uma célula de Markdown com:
• Objetivo do projeto • Origem dos dados
• Autores • Data de criação

3 Estruturar Logicamente o Script

Organize o código em blocos lógicos com títulos descritivos:

```
## 1. Importação de Bibliotecas
import pandas as pd

## 2. Carregamento dos Dados
df = pd.read_csv('vendas.csv')

## 3. Limpeza e Tratamento
df.dropna(inplace=True)

## 4. Análise e Filtragem
resultado = df[df['Valor'] > 1000]
```

4 Compartilhar e Revisar

Compartilhe o script documentado com outra dupla para revisão. O revisor deve verificar se o código é fácil de entender, confirmar se os comentários são claros e sugerir melhorias.

Colaboração é Fundamental

Um código bem documentado é um código que qualquer membro da equipe pode entender e manter. Pense sempre em quem lerá seu código no futuro!

Encerramento e Próximos Passos

Síntese dos Comandos Aprendidos

- `pd.read_csv()` - Importar dados
- `df.head()` - Visualizar primeiras linhas
- `df.info()` - Informações gerais
- `df.isnull()` - Identificar nulos
- `df.fillna()` - Preencher valores ausentes
- `df.drop_duplicates()` - Remover duplicatas
- `& e |` - Operadores lógicos para filtros

Reflexão: A proficiência em Pandas e a disciplina na documentação são os pilares para construir projetos de Ciência de Dados robustos e sustentáveis.

Próximos Passos

1. Pratique os comandos aprendidos com diferentes datasets para consolidar o conhecimento.
2. Explore funções avançadas do Pandas como merge, join e pivot_table para análises mais complexas.
3. Aplique as técnicas de filtragem e tratamento de dados em projetos reais do seu dia a dia.
4. Continue documentando seu código seguindo as boas práticas apresentadas nesta aula.
5. Integre o Pandas com bibliotecas de visualização como Matplotlib e Seaborn para criar gráficos informativos.

🎓 Parabéns por concluir esta aula!
Continue praticando e explorando o mundo dos dados!