

O CAMINHO DOS DADOS

O Pandas como Ferramenta de Mestre



Caio Arruda

Sumário

Sabedoria e Maestria com Pandas	5
O Pandas como Ferramenta de Mestre	5
O que é Pandas?	7
Instalação e Primeiros Passos	7
A Importância do Pandas na Ciência de Dados	8
Series e DataFrames	10
Diferenças entre Series, DataFrames e Arrays NumPy	11
Criação e Manipulação de DataFrames	12
Criando DataFrames:	12
Manipulando DataFrames:	12
Importando Dados de CSV, Excel, JSON, SQL, etc.	15
Importando CSV:	15
Importando Excel:	15
Importando JSON:	16
Importando SQL:	16
Salvando DataFrames em Diferentes Formatos	17
Salvando em CSV:	17
Salvando em Excel:	17
Salvando em JSON:	18
Salvando em SQL:	18
Configurações Avançadas de Importação	18
Importando com opções avançadas:	19
Indexação Básica e Avançada	21
Indexação Básica:	21
Indexação Avançada:	22
Seleção por Labels, Condições e Índices	22
Seleção por Labels:	23
Seleção por Condições:	23
Seleção por Índices:	23
Trabalhando com o Método loc[] e iloc[]	24
Usando loc[]:	24
Usando iloc[]:	24

Adicionando, Removendo e Renomeando Colunas	27
Adicionando Colunas:	27
Removendo Colunas:	27
Renomeando Colunas:	28
Modificação de Dados em Massa	28
Modificando Dados:	28
Substituindo Valores:	29
Aplicação de Funções: apply(), map(), applymap()	29
Usando apply():	29
Usando map():	30
Usando applymap():	30
Identificação e Tratamento de Valores Nulos	32
Identificando Valores Nulos:	32
Métodos de Preenchimento de Dados Faltantes	32
Preenchendo com um Valor Específico:	33
Preenchendo com a Média, Mediana ou Moda:	33
Remoção de Dados Incompletos	33
Removendo Linhas com Dados Faltantes:	34
Removendo Colunas com Dados Faltantes:	34
Introdução ao groupby	36
Exemplo Básico de groupby:	36
Agregação, Transformação e Filtragem de Grupos	37
Agregação:	37
Transformação:	37
Filtragem:	38
Análise de Dados Agrupados	38
Estatísticas Descritivas:	38
Visualização dos Dados Agrupados:	39
Combinação de DataFrames com merge() e join()	41
Usando merge():	41
Usando join():	41
Concatenação de DataFrames	42
Usando concat():	42

Casos Práticos de Junção de Dados	42
Caso Prático 1: Mesclagem de Vendas e Produtos	43
Caso Prático 2: Junção de Dados Financeiros	43
Manipulando Dados de Data com Pandas	46
Convertendo Strings para Datas:	46
Extraindo Componentes de Datas:	46
Indexação e Resampling de Séries Temporais	47
Indexando com Datas:	47
Resampling de Séries Temporais:	47
Operações Comuns com Datas e Períodos	48
Adicionando e Subtraindo Datas:	48
Calculando Diferenças entre Datas:	48
Trabalhando com Períodos:	49
Ordenando Colunas e Índices	51
Ordenando Colunas:	51
Ordenando Índices:	51
Ordenação Baseada em Múltiplos Critérios	52
Classificação e Rankeamento de Dados	52
Introdução à Visualização com Pandas e Matplotlib	55
Importando as Bibliotecas:	55
Criando Gráficos Simples e Avançados	55
Gráfico de Barras Simples:	56
Gráfico de Linha:	56
Gráfico Avançado (Subplots):	56
Integração de Pandas com Outras Bibliotecas de Visualização	57
Usando Seaborn:	57
Usando Plotly:	57
Operações Vetorizadas e Desempenho	60
Exemplo de Operações Vetorizadas:	60
Trabalhando com Grandes Datasets	60
Usando tipos de dados otimizados:	61
Leitura eficiente de arquivos grandes:	61
Lidando com Dados Complexos e Pipelines de Dados	61

Transformações com Pipelines:	62
Manipulação Avançada de Dados:	62
Obrigado!	64

Sabedoria e Maestria com Pandas

O Pandas como Ferramenta de Mestre

No universo da programação e ciência de dados, o Python se destaca como uma das linguagens mais poderosas e versáteis. Entre as diversas bibliotecas que tornam essa linguagem ainda mais robusta, uma se destaca no campo da análise de dados: Pandas. Criada para facilitar a manipulação, limpeza e análise de dados, a Pandas transformou a maneira como cientistas de dados, analistas e engenheiros lidam com dados complexos e volumosos.

Neste ebook, vamos explorar as funcionalidades fundamentais da Pandas, desde as operações básicas de leitura e escrita de dados até técnicas avançadas de manipulação e análise. Seja você um iniciante que está dando os primeiros passos no mundo da ciência de dados, ou um profissional experiente procurando aprimorar suas habilidades, este ebook oferece um guia abrangente e prático para dominar o uso da Pandas.





Introdução ao Pandas

E aí, jovem mestre dos dados! Pronto para aprender o estilo "Kung Fu Panda" de lidar com informação?

Imagine-se como Po, o panda fofinho e desajeitado, que no início parecia um desastre, mas com treinamento e dedicação, virou o Dragão Guerreiro. Vamos tornar você um mestre dos dados com o Pandas!

O que é Pandas?

Pandas é uma biblioteca poderosa do Python para manipulação e análise de dados. Pense nele como seu próprio treinamento com o mestre Shifu – essencial para se tornar um guerreiro dos dados! Ele ajuda a organizar, analisar e explorar seus dados de forma mais eficaz.

Instalação e Primeiros Passos

Como Po aprendendo Kung Fu, você precisa começar com os básicos. Primeiro, instalamos o Pandas:



Instalação

```
pip install pandas
```

Agora que tem o Pandas instalado, é hora de carregar sua primeira DataFrame – pense nisso como seu "Palácio de Jade" cheio de dados.



Dataframe

```
import pandas as pd
# Criando um DataFrame básico
data = {'nome': ['Po', 'Shifu', 'Tigresa'], 'poder': [100, 95, 90]}
df = pd.DataFrame(data)
print(df)
```

A Importância do Pandas na Ciência de Dados

Assim como o pergaminho do Dragão ensinou a Po que ele era especial do jeito dele, o Pandas revela o poder escondido nos seus dados. Ele é a arma secreta dos cientistas de dados, ajudando a transformar dados crus em insights valiosos com facilidade.

De limpeza de dados a análises complexas, com Pandas você vira o verdadeiro Dragão Guerreiro dos Dados!

Prepare-se para uma jornada épica. E lembre-se: você não precisa ser perfeito – apenas persistente, assim como nosso herói, Po!



Estruturas de Dados no Pandas

Pronto para dominar as técnicas secretas do Kung Fu dos Dados? Vamos mergulhar nas Series e DataFrames, os movimentos fundamentais do Pandas!

Series e DataFrames

Pense nas Series como um único golpe de punho. Elas são como listas especiais com rótulos – simples e diretas.

```
● ● ● Series  
  
import pandas as pd  
# Criando uma Series  
po_powers = pd.Series([100, 95, 90], index=['Po', 'Shifu', 'Tigresa'])  
print(po_powers)
```

Agora, os DataFrames são como uma combinação épica de golpes, juntando várias Series para formar uma tabela poderosa – uma verdadeira coreografia dos dados!

```
● ● ● DF  
  
# Criando um DataFrame  
data = {'nome': ['Po', 'Shifu', 'Tigresa'], 'poder': [100, 95, 90]}  
df = pd.DataFrame(data)  
print(df)
```

Diferenças entre Series, DataFrames e Arrays NumPy

Imagine as Series como Po treinando sozinho. Um vetor unidimensional com rótulos. Já os DataFrames são como todos os mestres treinando juntos – várias Series combinadas em duas dimensões.

```
Series vs DF

# Exemplo de Series
print(type(po_powers)) # pandas.core.series.Series

# Exemplo de DataFrame
print(type(df)) # pandas.core.frame.DataFrame
```

Os Arrays NumPy são como Po no início de seu treinamento – sem rótulos, apenas força bruta e números. Enquanto Pandas adiciona contexto e flexibilidade, o NumPy foca na eficiência e cálculo.

```
Array

import numpy as np
# Criando um Array NumPy
np_array = np.array([100, 95, 90])
print(np_array)
print(type(np_array)) # numpy.ndarray
```

Criação e Manipulação de DataFrames

Vamos aprimorar nosso Kung Fu criando e manipulando DataFrames. Pense nisso como Po aprendendo novas técnicas e se tornando cada vez mais forte.

Criando DataFrames:

```
● ● ● DF  
# Criando um DataFrame  
data = {'nome': ['Po', 'Shifu', 'Tigresa'], 'poder': [100, 95, 90]}  
df = pd.DataFrame(data)  
print(df)
```

Manipulando DataFrames:

```
● ● ● Manipulação DF  
# Adicionando uma nova coluna  
df['habilidade'] = ['Kung Fu', 'Sabedoria', 'Força']  
print(df)  
  
# Selecionando uma coluna específica  
print(df['poder'])  
  
# Filtrando linhas com base em uma condição  
fortes = df[df['poder'] > 90]  
print
```

Com essas técnicas, você está um passo mais perto de se tornar um verdadeiro mestre dos dados. Continue praticando, jovem guerreiro, e logo você dominará o Kung Fu dos Dados!



Leitura e Escrita de Arquivos

Bem-vindo de volta, jovem guerreiro dos dados! Agora vamos aprender a arte de importar e exportar dados – essencial para qualquer mestre Pandas.

Importando Dados de CSV, Excel, JSON, SQL, etc.

Pandas é como um mestre que conhece várias artes marciais – ele pode lidar com diversos formatos de dados sem suar. Vamos conferir alguns exemplos:

Importando CSV:

```
... Import CSV  
  
import pandas as pd  
# Lendo um arquivo CSV  
df_csv = pd.read_csv('dados.csv')  
print(df_csv)
```

Importando Excel:

```
... Import Excel  
  
# Lendo um arquivo Excel  
df_excel = pd.read_excel('dados.xlsx')  
print(df_excel)
```

Importando JSON:

```
● ● ● Import JSON  
  
# Lendo um arquivo JSON  
df_json = pd.read_json('dados.json')  
print(df_json)
```

Importando SQL:

```
● ● ● Import SQL  
  
import sqlite3  
# Conectando ao banco de dados  
conn = sqlite3.connect('banco_de_dados.db')  
# Lendo dados de uma tabela SQL  
df_sql = pd.read_sql_query("SELECT * FROM tabela", conn)  
print(df_sql)  
conn.close()
```

Salvando DataFrames em Diferentes Formatos

Assim como Po e seus amigos protegem o Vale da Paz, precisamos salvar nossos dados para garantir que nada seja perdido.

Salvando em CSV:



Export CSV

```
# Salvando um DataFrame em CSV  
df_csv.to_csv('dados_salvos.csv', index=False)
```

Salvando em Excel:



Export Excel

```
# Salvando um DataFrame em Excel  
df_excel.to_excel('dados_salvos.xlsx', index=False)
```

Salvando em JSON:

```
● ● ● Export JSON  
# Salvando um DataFrame em JSON  
df_json.to_json('dados_salvos.json')
```

Salvando em SQL:

```
● ● ● Export SQL  
# Salvando um DataFrame em uma tabela SQL  
conn = sqlite3.connect('banco_de_dados.db')  
df_sql.to_sql('tabela_salva', conn, if_exists='replace', index=False)  
conn.close()
```

Configurações Avançadas de Importação

À medida que você avança em seu treinamento, aprenderá que pequenas configurações podem fazer uma grande diferença no Kung Fu dos Dados.

Importando com opções avançadas:

```
● ● ● Import advance

# Lendo um CSV com separador personalizado e sem cabeçalho
df_csv_custom = pd.read_csv('dados.csv', sep=';', header=None)
print(df_csv_custom)

# Lendo um Excel com múltiplas planilhas
excel_file = pd.ExcelFile('dados_multiplos.xlsx')
df_planilha1 = pd.read_excel(excel_file, 'Planilha1')
df_planilha2 = pd.read_excel(excel_file, 'Planilha2')
print(df_planilha1)
print(df_planilha2)

# Importando apenas algumas colunas do JSON
df_json_custom = pd.read_json('dados.json', usecols=['coluna1', 'coluna2'])
print(df_json_custom)

# Lendo SQL com condições específicas
df_sql_filtered = pd.read_sql_query("SELECT * FROM tabela WHERE coluna > 10", conn)
print(df_sql_filtered)
```

Com essas técnicas, você está se tornando um verdadeiro Dragão Guerreiro dos Dados. Continue praticando, pois o domínio dessas habilidades garantirá que você esteja preparado para qualquer desafio que encontrar no mundo dos dados!



Seleção e Filtragem de Dados

Vamos continuar nosso treinamento no estilo Kung Fu Panda para dominar a arte de Seleção e Filtragem de Dados no Pandas! Prepare-se para se tornar um verdadeiro mestre.

Indexação Básica e Avançada

A indexação no Pandas é como escolher seus movimentos com precisão em uma luta. Começamos com o básico e depois avançamos para técnicas mais complexas.

Indexação Básica:

```
Basic index

import pandas as pd
# Criando um DataFrame básico
data = {'nome': ['Po', 'Shifu', 'Tigresa'], 'poder': [100, 95, 90]}
df = pd.DataFrame(data)
print(df)
# Selecionando uma coluna
print(df['nome'])
# Selecionando múltiplas colunas
print(df[['nome', 'poder']])
```

Indexação Avançada:

```
● ● ● Advance index

# Usando um índice personalizado
df.set_index('nome', inplace=True)
print(df)
# Selecionando dados por índice
print(df.loc['Po'])
```

Seleção por Labels, Condições e Índices

Agora vamos afiar nossa seleção, seja por labels, condições ou índices. Pense nisso como aperfeiçoar seus movimentos em uma luta.

Seleção por Labels:

```
● ● ● Seleção por label  
# Selecionando dados por rótulos  
print(df.loc['Shifu'])
```

Seleção por Condições:

```
● ● ● Seleção por condição  
# Filtrando dados com base em uma condição  
fortes = df[df['poder'] > 90]  
print(fortes)
```

Seleção por Índices:

```
● ● ● Seleção por índices  
# Selecionando dados por posição de índice  
print(df.iloc[0])
```

Trabalhando com o Método loc[] e iloc[]

Os métodos loc[] e iloc[] são como os diferentes estilos de Kung Fu – cada um com suas próprias vantagens. O loc[] é usado para seleção com base em labels, enquanto o iloc[] é usado para seleção com base em índices posicionais.

Usando loc[]:

```
● ● ● Loc  
# Selecionando uma linha específica por label  
print(df.loc['Tigresa'])  
# Selecionando múltiplas linhas e colunas por label  
print(df.loc[['Po', 'Shifu'], ['poder']])
```

Usando iloc[]:

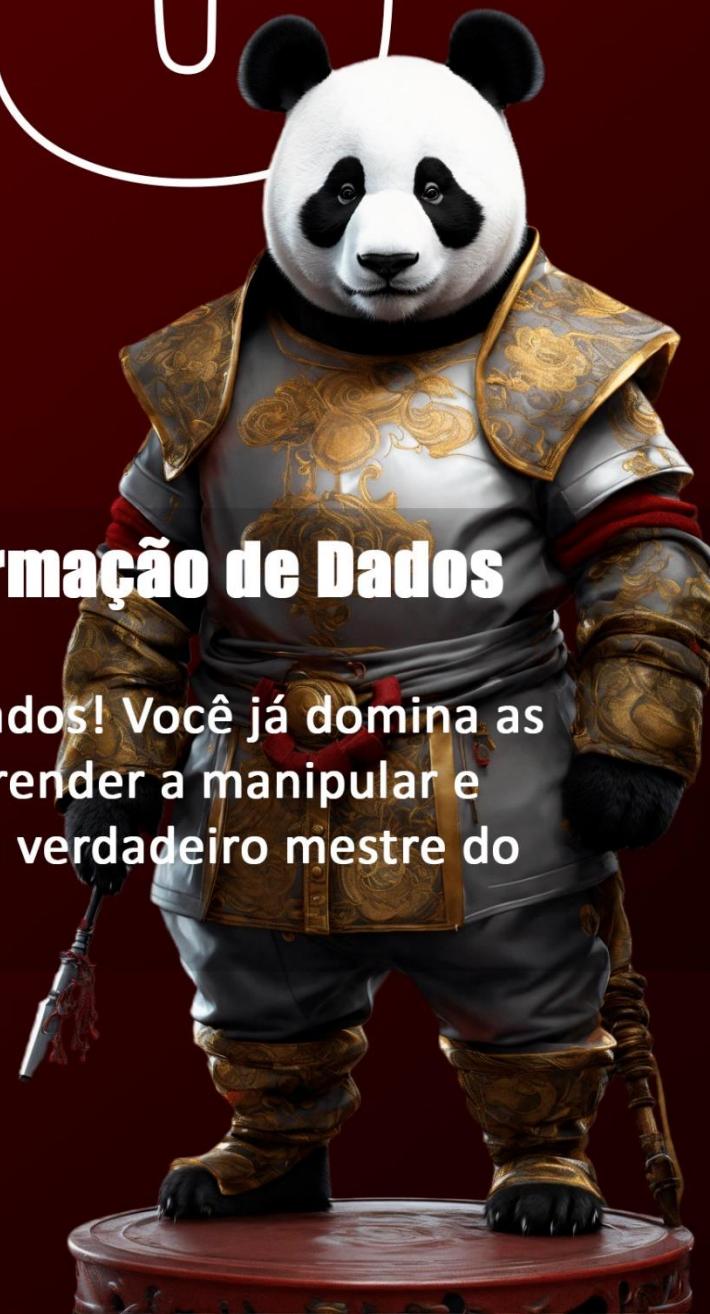
```
● ● ● iLoc  
# Selecionando uma linha específica por posição  
print(df.iloc[1])  
# Selecionando múltiplas linhas e colunas por posição  
print(df.iloc[0:2, 0:2])
```

Com essas técnicas, você estará pronto para enfrentar qualquer desafio de dados que encontrar. Continue treinando e logo será um verdadeiro Dragão Guerreiro dos Dados!

05

Manipulação e Transformação de Dados

Parabéns, jovem guerreiro dos dados! Você já domina as técnicas básicas, agora vamos aprender a manipular e transformar seus dados como um verdadeiro mestre do Kung Fu dos Dados.



Adicionando, Removendo e Renomeando Colunas

Imagine adicionar, remover e renomear colunas como ajustar sua postura e movimentos durante uma batalha. É fundamental para obter o máximo de eficiência!

Adicionando Colunas:

```
... Add Colunas ...  
  
import pandas as pd  
# Criando um DataFrame  
data = {'nome': ['Po', 'Shifu', 'Tigresa'], 'poder': [100, 95, 90]}  
df = pd.DataFrame(data)  
# Adicionando uma nova coluna  
df['habilidade'] = ['Kung Fu', 'Sabedoria', 'Força']  
print(df)
```

Removendo Colunas:

```
... Remove Colunas ...  
  
# Removendo uma coluna  
df.drop('habilidade', axis=1, inplace=True)  
print(df)
```

Renomeando Colunas:

```
● ● ● Rename Colunas  
# Renomeando colunas  
df.rename(columns={'poder': 'força'}, inplace=True)  
print
```

Modificação de Dados em Massa

Modificar dados em massa é como executar um combo de golpes rápidos e precisos. Aqui estão algumas técnicas para fazer isso.

Modificando Dados:

```
● ● ● Modificar dados  
# Modificando todos os valores de uma coluna  
df['força'] = df['força'] + 10  
print
```

Substituindo Valores:



Substituir dados

```
# Substituindo valores específicos
df['nome'] = df['nome'].replace('Po', 'Dragon Warrior')
print(df)
```

Aplicação de Funções: apply(), map(), applymap()

Aplicar funções aos dados é como utilizar técnicas secretas de Kung Fu – aumenta seu poder e eficácia.

Usando apply():



Usando apply

```
# Criando uma função personalizada
def aumentar_poder(x):
    return x + 5

# Aplicando a função a uma coluna inteira
df['força'] = df['força'].apply(aumentar_poder)
print(df)
```

Usando map():

```
● ● ● Usando map  
# Mapeando valores com uma função lambda  
df['nome'] = df['nome'].map(lambda x: x.upper())  
print(df)
```

Usando applymap():

```
● ● ● Usando applymap  
# Aplicando uma função a todos os elementos do DataFrame  
df['força'] = df[['força']].applymap(lambda x: x * 2)  
print(df)
```

Com essas técnicas, você estará pronto para enfrentar qualquer desafio que surgir no caminho dos dados. Continue praticando, e logo será um verdadeiro Dragão Guerreiro dos Dados!



Trabalhando com Dados Faltantes

Olá novamente, jovem mestre dos dados! Agora vamos enfrentar um dos maiores vilões do Kung Fu dos Dados: os valores faltantes. Vamos transformá-los em aliados no nosso treinamento!

Identificação e Tratamento de Valores Nulos

Antes de podermos combater os dados faltantes, precisamos identificá-los. Pense nisso como Po descobrindo os pontos fracos de seus inimigos.

Identificando Valores Nulos:

```
● ● ● Identificar nulos

import pandas as pd
# Criando um DataFrame com valores nulos
data = {'nome': ['Po', 'Shifu', 'Tigresa', None], 'poder': [100, 95, 90, None]}
df = pd.DataFrame(data)
# Identificando valores nulos
print(df.isnull())
# Contando valores nulos
print(df.isnull().sum())
```

Métodos de Preenchimento de Dados Faltantes

Agora que sabemos onde estão os dados faltantes, vamos preencher esses vazios com técnicas poderosas.

Preenchendo com um Valor Específico:

```
● ● ● Preencher nulos

# Preenchendo valores nulos com um valor específico
df.fillna({'nome': 'Desconhecido', 'poder': 0}, inplace=True)
print(df)
```

Preenchendo com a Média, Mediana ou Moda:

```
● ● ● Preencher nulos medi_
```

```
# Criando um novo DataFrame para exemplificar
data = {'nome': ['Po', 'Shifu', 'Tigresa', None], 'poder': [100, 95, 90, None]}
df = pd.DataFrame(data)
# Preenchendo valores nulos com a média
df['poder'].fillna(df['poder'].mean(), inplace=True)
print(df)
```

Remoção de Dados Incompletos

Às vezes, a melhor forma de lidar com dados faltantes é removê-los. É como Po eliminando distrações para se concentrar em seu treinamento.

Removendo Linhas com Dados Faltantes:

```
● ● ● Remover linhas nulas

# Criando um DataFrame com valores nulos
data = {'nome': ['Po', 'Shifu', 'Tigresa', None], 'poder': [100, 95, 90, None]}
df = pd.DataFrame(data)
# Removendo linhas com valores nulos
df.dropna(inplace=True)
print(df)
```

Removendo Colunas com Dados Faltantes:

```
● ● ● Remover colunas nulas

# Criando um DataFrame com valores nulos
data = {'nome': ['Po', 'Shifu', 'Tigresa', None], 'poder': [100, 95, 90, None]}
df = pd.DataFrame(data)
# Removendo colunas com valores nulos
df.dropna(axis=1, inplace=True)
print(df)
```

Com essas técnicas, você está bem equipado para lidar com os dados faltantes como um verdadeiro Dragão Guerreiro. Continue praticando e, em breve, será um mestre dos dados completo!



Agrupamento e Agregação de Dados

Bem-vindo de volta, jovem guerreiro dos dados! Chegou a hora de aprender a arte do agrupamento e agregação de dados no estilo Kung Fu Panda. Esses movimentos vão transformar sua habilidade de analisar grandes volumes de dados como um mestre!

Introdução ao groupby

O método groupby do Pandas é como a técnica do mestre Shifu para organizar seus discípulos. Com ele, você pode agrupar seus dados com base em uma ou mais colunas, tornando a análise muito mais eficiente.

Exemplo Básico de groupby:

```
● ● ● pd groupby

import pandas as pd

# Criando um DataFrame exemplo
data = {
    'nome': ['Po', 'Shifu', 'Tigresa', 'Víbora', 'Garça'],
    'categoria': ['Herói', 'Mestre', 'Herói', 'Herói', 'Mestre'],
    'poder': [100, 95, 90, 85, 80]
}
df = pd.DataFrame(data)

# Agrupando por categoria
grupo = df.groupby('categoria')
print(grupo.mean())
```

Agregação, Transformação e Filtragem de Grupos

O próximo passo é agregar, transformar e filtrar esses grupos. Pense nisso como Po aprendendo a usar suas habilidades de maneiras inovadoras.

Agregação:

```
● ● ● Agregação  
# Agregando dados  
agrupado = df.groupby('categoria').agg({'poder': ['mean', 'sum']})  
print(agrupado)
```

Transformação:

```
● ● ● Transformação  
# Transformando dados do grupo  
transformado = df.groupby('categoria')['poder'].transform(lambda x: x - x.mean())  
df['poder_ajustado'] = transformado  
print
```

Filtragem:

```
● ● ● Filtragem  
# Filtrando grupos com base em uma condição  
filtrado = df.groupby('categoria').filter(lambda x: x['poder'].mean() > 85)  
print(filtrado)
```

Análise de Dados Agrupados

Agora que você sabe agrupar e manipular seus dados, vamos analisar esses grupos e extrair insights poderosos – como Po se tornando o Dragão Guerreiro.

Estatísticas Descritivas:

```
● ● ● Estatística Descritiva  
# Obtendo estatísticas descritivas dos grupos  
estatisticas = df.groupby('categoria').describe()  
print(estatisticas)
```

Visualização dos Dados Agrupados:



Visualização de Dados

```
import matplotlib.pyplot as plt

# Plotando a média de poder por categoria
media_poder = df.groupby('categoria')['poder'].mean()
media_poder.plot(kind='bar')
plt.xlabel('Categoria')
plt.ylabel('Média de Poder')
plt.title('Média de Poder por Categoria')
plt.show()
```

Com essas técnicas, você está pronto para enfrentar qualquer desafio de análise de dados que surgir. Continue praticando e logo será um verdadeiro Dragão Guerreiro dos Dados!



Mesclagem e Junção de DataFrames

Bem-vindo de volta, jovem guerreiro dos dados! Vamos aprender a combinar e juntar DataFrames como um mestre de Kung Fu combinando seus movimentos. Isso permitirá que você reúna dados de diferentes fontes com precisão e agilidade.

Combinação de DataFrames com merge() e join()

A função **merge()** é como uma fusão poderosa, unindo DataFrames com base em colunas comuns. Já o **join()** é como um movimento de colaboração suave, juntando DataFrames com base em índices.

Usando **merge()**:

```
●●● Merge  
import pandas as pd  
  
# Criando DataFrames de exemplo  
df1 = pd.DataFrame({'nome': ['Po', 'Shifu', 'Tigresa'], 'poder': [100, 95, 98]})  
df2 = pd.DataFrame({'nome': ['Po', 'Tigresa', 'Garça'], 'habilidade': ['Kung Fu', 'Força',  
'Vôo']})  
  
# Mesclando DataFrames com base na coluna 'nome'  
df_merged = pd.merge(df1, df2, on='nome', how='inner')  
print(df_merged)
```

Usando **join()**:

```
●●● Join  
  
# Ajustando índices para usar o join()  
df1.set_index('nome', inplace=True)  
df2.set_index('nome', inplace=True)  
  
# Juntando DataFrames com base no índice  
df_joined = df1.join(df2, how='inner')  
print(df_joined)
```

Concatenação de DataFrames

Concatenar DataFrames é como alinhar vários movimentos em uma sequência harmoniosa. O método `concat()` ajuda a juntar DataFrames ao longo de um eixo específico.

Usando `concat()`:

```
# Criando DataFrames de exemplo
df3 = pd.DataFrame({'nome': ['Víbora', 'Macaco'], 'poder': [85, 80]})

# Concatenando DataFrames ao longo do eixo 0 (linhas)
df_concatenado = pd.concat([df1.reset_index(), df3], axis=0, ignore_index=True)
print(df_concatenado)

# Concatenando DataFrames ao longo do eixo 1 (colunas)
df_concatenado_colunas = pd.concat([df1.reset_index(), df2.reset_index()], axis=1)
print(df_concatenado_colunas)
```

Casos Práticos de Junção de Dados

Agora que dominamos as técnicas básicas, vamos ver alguns casos práticos para aplicar essas técnicas.

Caso Prático 1: Mesclagem de Vendas e Produtos

```
● ● ● mesclagem venda

# DataFrame de vendas
vendas = pd.DataFrame({
    'produto_id': [1, 2, 3],
    'quantidade': [10, 20, 15]
})

# DataFrame de produtos
produtos = pd.DataFrame({
    'produto_id': [1, 2, 3],
    'produto_nome': ['Produto A', 'Produto B', 'Produto C']
})

# Mesclando vendas com produtos para obter informações detalhadas
vendas_com_produtos = pd.merge(vendas, produtos, on='produto_id', how='left')
print(vendas_com_produtos)
```

Caso Prático 2: Junção de Dados Financeiros

```
● ● ● junção venda

# DataFrame de receitas
receitas = pd.DataFrame({
    'mês': ['Jan', 'Feb', 'Mar'],
    'receita': [5000, 7000, 8000]
})

# DataFrame de despesas
despesas = pd.DataFrame({
    'mês': ['Jan', 'Feb', 'Mar'],
    'despesa': [3000, 4000, 2000]
})

# Juntando receitas e despesas para análise financeira
df_financeiro = pd.merge(receitas, despesas, on='mês')
df_financeiro['lucro'] = df_financeiro['receita'] - df_financeiro['despesa']
print(df_financeiro)
```

Com essas técnicas de mesclagem e junção, você está bem equipado para enfrentar qualquer desafio de dados que surgir. Continue praticando e logo será um verdadeiro Dragão Guerreiro dos Dados!



Operações de Tempo e Data

Bem-vindo novamente, jovem guerreiro dos dados!
Vamos aprender a lidar com datas e tempos no Pandas,
como um mestre Kung Fu manipula o tempo e o espaço.

Manipulando Dados de Datas com Pandas

Manipular dados de datas no Pandas é como controlar o fluxo do tempo – poderoso e essencial.

Convertendo Strings para Datas:

```
••• String to date

import pandas as pd

# Criando um DataFrame com datas em formato string
data = {'data': ['2023-01-01', '2023-02-01', '2023-03-01'], 'valor': [100, 200, 300]}
df = pd.DataFrame(data)

# Convertendo strings para datetime
df['data'] = pd.to_datetime(df['data'])
print(df)
```

Extraindo Componentes de Datas:

```
••• Extrair data

# Extraindo o ano, mês e dia
df['ano'] = df['data'].dt.year
df['mês'] = df['data'].dt.month
df['dia'] = df['data'].dt.day
print(df)
```

Indexação e Resampling de Séries Temporais

Indexar e resampling de séries temporais é como ajustar seus movimentos de Kung Fu para se adaptar ao ritmo da luta.

Indexando com Datas:

```
● ● ● Index de data  
# Configurando a coluna de data como índice  
df.set_index('data', inplace=True)  
print(df)
```

Resampling de Séries Temporais:

```
● ● ● Serie temporal  
# Criando uma série temporal exemplo  
data = pd.date_range(start='2023-01-01', periods=6, freq='D')  
df = pd.DataFrame({'valor': [10, 20, 30, 40, 50, 60]}, index=data)  
  
# Resampling para agregação mensal  
df_resampled = df.resample('M').sum()  
print(df_resampled)
```

Operações Comuns com Datas e Períodos

Executar operações comuns com datas e períodos é como usar técnicas avançadas de Kung Fu para vencer seus oponentes.

Adicionando e Subtraindo Datas:

```
● ● ● Add e subtraindo datas  
# Criando uma série temporal  
df['data'] = pd.date_range(start='2023-01-01', periods=3, freq='M')  
  
# Adicionando 10 dias a cada data  
df['data_mais_10dias'] = df['data'] + pd.Timedelta(days=10)  
print
```

Calculando Diferenças entre Datas:

```
● ● ● Diferença de datas  
# Calculando a diferença entre duas datas  
df['diferenca'] = df['data_mais_10dias'] - df['data']  
print(df)
```

Trabalhando com Períodos:

```
● ● ● Períodos  
  
# Criando uma série de períodos mensais  
periodos = pd.period_range(start='2023-01', periods=6, freq='M')  
df_periodos = pd.DataFrame({'valor': range(6)}, index=periodos)  
print(df_periodos)  
  
# Convertendo períodos para timestamps  
df_periodos.index = df_periodos.index.to_timestamp()  
print(df_periodos)
```

Com essas técnicas, você está pronto para controlar o tempo e os dados como um verdadeiro Dragão Guerreiro dos Dados. Continue praticando e, em breve, será um mestre completo!



Ordenação e Classificação de Dados

Vamos aprender Ordenação e Classificação de Dados de forma similar ao que Po vivenciou ao aprender Kung Fu. No início, Po se sentia perdido e desajeitado, assim como pode acontecer ao lidarmos com grandes volumes de dados pela primeira vez. No entanto, com a prática dos fundamentos, assim como Po dominou as técnicas básicas de luta, você também dominará as ferramentas de organização de dados, transformando o caos em ordem com eficiência e precisão.

Ordenando Colunas e Índices

Imagine que você está organizando o Grande Torneio de Kung Fu no Vale da Paz. Vamos organizar nossos lutadores (dados) com precisão!

Ordenando Colunas:

```
● ● ● Ordenar coluna

import pandas as pd

# Criando um DataFrame de exemplo
data = {'nome': ['Po', 'Shifu', 'Tigresa'], 'poder': [100, 95, 90]}
df = pd.DataFrame(data)

# Ordenando por poder (decrescente)
df_ordenado = df.sort_values(by='poder', ascending=False)
print(df_ordenado)
```

Ordenando Índices:

```
● ● ● Ordenar índice

# Ajustando índice para o nome
df.set_index('nome', inplace=True)
# Ordenando pelo índice (nome)
df_ordenado_indice = df.sort_index()
print(df_ordenado_indice)
```

Ordenação Baseada em Múltiplos Critérios

Agora imagine que, além do poder, você também quer considerar a idade dos lutadores. Vamos ordenar com base em múltiplos critérios, como o mestre Shifu organizando um combate perfeito.

```
● ● ● Ordenar multi

# Criando um DataFrame de exemplo
data = {'nome': ['Po', 'Shifu', 'Tigresa', 'Víbora'],
        'poder': [100, 95, 95, 85],
        'idade': [20, 60, 30, 25]}
df = pd.DataFrame(data)

# Ordenando por poder e depois por idade
df_multi_ordenado = df.sort_values(by=['poder', 'idade'], ascending=[False, True])
print(df_multi_ordenado)
```

Classificação e Rankeamento de Dados

Rankear nossos heróis é como decidir quem merece o título de Dragão Guerreiro! Vamos ver quem está no topo.

```
● ● ● Rankeamento

# Rankeando os lutadores por poder
df['rank'] = df['poder'].rank(ascending=False)
print(df)
```

Com essas técnicas, você está pronto para ordenar e classificar seus dados com a precisão de um mestre Kung Fu. Continue praticando e em breve será um verdadeiro Dragão Guerreiro dos Dados!



Visualização de Dados com Pandas

Vamos começar nossa jornada na Visualização de Dados com Pandas, seguindo uma abordagem clara e eficiente, assim como Po aprimorou suas habilidades ao longo do tempo. Através de técnicas práticas e aplicadas, você desenvolverá a capacidade de criar visualizações eficazes e informativas, facilitando a interpretação de grandes volumes de dados com precisão.

Introdução à Visualização com Pandas e Matplotlib

Visualizar dados é como dominar os movimentos do Dragão Guerreiro - você precisa das ferramentas certas. Pandas e Matplotlib são a dupla dinâmica que vão te ajudar a transformar dados em insights visuais.

Importando as Bibliotecas:

```
● ● ● Importando libs

import pandas as pd
import matplotlib.pyplot as plt

# Criando um DataFrame de exemplo
data = {'nome': ['Po', 'Shifu', 'Tigresa'], 'poder': [100, 95, 90]}
df = pd.DataFrame(data)
```

Criando Gráficos Simples e Avançados

Vamos começar com movimentos básicos e depois partir para combinações avançadas, como se você estivesse aprendendo os passos de Kung Fu.

Gráfico de Barras Simples:

```
● ● ● Gráfico simples  
# Gráfico de barras do poder dos personagens  
df.plot(kind='bar', x='nome', y='poder', title='Poder dos Personagens')  
plt.show()
```

Gráfico de Linha:

```
● ● ● Gráfico linha  
# Gráfico de linha para mostrar a progressão de poder  
df.plot(kind='line', x='nome', y='poder', title='Progressão de Poder')  
plt.show()
```

Gráfico Avançado (Subplots):

```
● ● ● Gráfico avançado  
# Criando subplots para múltiplos gráficos  
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 5))  
df.plot(kind='bar', x='nome', y='poder', ax=axes[0], title='Poder dos Personagens')  
df.plot(kind='line', x='nome', y='poder', ax=axes[1], title='Progressão de Poder')  
plt.show()
```

Integração de Pandas com Outras Bibliotecas de Visualização

Assim como Po aprendeu técnicas de diferentes mestres, você pode integrar Pandas com outras bibliotecas de visualização para ampliar seu arsenal.

Usando Seaborn:

```
Seaborn
import seaborn as sns

# Gráfico de barras com Seaborn
sns.barplot(x='nome', y='poder', data=df).set(title='Poder dos Personagens com Seaborn')
plt.show()
```

Usando Plotly:

```
Plotly
import plotly.express as px

# Gráfico interativo com Plotly
fig = px.bar(df, x='nome', y='poder', title='Poder dos Personagens com Plotly')
fig.show()
```

Com essas técnicas, você está pronto para criar visualizações incríveis que vão impressionar qualquer mestre do Kung Fu dos Dados. Continue praticando, e logo você será um verdadeiro Dragão Guerreiro das Visualizações!

12

Técnicas Avançadas de Pandas

Vamos explorar as Técnicas Avançadas de Pandas com a mesma determinação do Po. Esteja preparado para elevar seu nível e lidar com dados de forma eficiente e poderosa.

Operações Vetorizadas e Desempenho

Operações vetorizadas são como movimentos rápidos e precisos que economizam tempo e esforço. Elas permitem que você manipule grandes volumes de dados de forma eficiente.

Exemplo de Operações Vetorizadas:

```
Operação vetorizada

import pandas as pd
import numpy as np

# Criando um DataFrame grande para exemplo
data = {'A': np.random.rand(1000000), 'B': np.random.rand(1000000)}
df = pd.DataFrame(data)

# Operação vetorizada para somar colunas
df['C'] = df['A'] + df['B']
print(df.head())
```

Trabalhando com Grandes Datasets

Lidar com grandes datasets é como enfrentar um exército sozinho, mas com as ferramentas certas, você será invencível. Otimizações no Pandas ajudam a manter a performance mesmo com muitos dados.

Usando tipos de dados otimizados:

Otimização dados

```
# Otimizando tipos de dados para economizar memória
df['A'] = df['A'].astype('float32')
df['B'] = df['B'].astype('float32')
print(df.info())
```

Leitura eficiente de arquivos grandes:

Otimização de leitura

```
# Leitura de um arquivo CSV grande em pedaços (chunks)
chunk_size = 100000
for chunk in pd.read_csv('arquivo_grande.csv', chunksize=chunk_size):
    process_chunk(chunk) # Função fictícia para processar cada pedaço
```

Lidando com Dados Complexos e Pipelines de Dados

Trabalhar com dados complexos e pipelines é como seguir uma coreografia bem ensaiada – cada movimento é calculado para alcançar o resultado desejado.

Transformações com Pipelines:

```
● ● ● Trnsformação pipeline

from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

# Criando um pipeline de exemplo para transformar dados
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('model', SomeModel()) # Substitua por um modelo real
])

# Ajustando o pipeline aos dados
pipeline.fit(df[['A', 'B']], target)
```

Manipulação Avançada de Dados:

```
● ● ● Manipulação avançada

# Aplicando múltiplas transformações a um DataFrame
def transformar_dados(df):
    df['log_A'] = np.log(df['A'] + 1)
    df['B_squared'] = df['B'] ** 2
    return df

# Usando o método pipe para criar um pipeline de transformações
df_transformado = df.pipe(transformar_dados)
print(df_transformado.head())
```

Com essas técnicas avançadas, você está pronto para enfrentar qualquer desafio com confiança e habilidade. Continue aprimorando suas habilidades e em breve será um verdadeiro mestre dos dados.

Obrigado!

A jornada de explorar os mistérios dos dados com Pandas só foi possível graças a você, leitor dedicado. Espero que este ebook tenha ajudado você a avançar no domínio dessa poderosa ferramenta.

Gostaria de expressar minha gratidão a todos que contribuíram para este projeto e a você, que dedicou seu tempo para aprender e crescer.

Conecte-se comigo nas redes profissionais:

devcaiaida



Sobre o autor

Caio Arruda é um entusiasta dos dados e um profissional apaixonado por transformar números em insights significativos.