Python's Wizardry

Unleashing Data Spells

by Luiza Boiko

PYTHON PARA ANÁLISE DE DADOS

Python é uma linguagem poderosa e versátil, amplamente utilizada em análise de dados. Sua sintaxe simples e uma vasta coleção de bibliotecas a tornam ideal para manipular, analisar e visualizar dados. Neste ebook, vamos explorar os principais comandos e bibliotecas de Python para análise de dados, sempre com exemplos práticos.

01

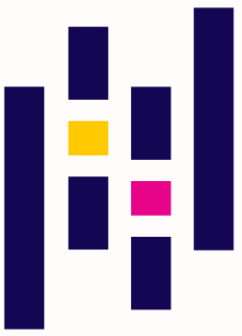
Instalando Bibliotecas Essenciais

COMO INSTALAR BIBLIOTECAS

Para começar, precisamos instalar algumas bibliotecas essenciais. As principais bibliotecas para análise de dados em Python são pandas, numpy e matplotlib.

Instalando

```
# Instale as bibliotecas com o comando pip
!pip install pandas numpy matplotlib
```



02

Carregando Dados com Pandas

LEITURA DE ARQUIVOS CSV

Uma das tarefas mais comuns é carregar dados de arquivos CSV. Usamos a função read_csv do pandas para isso.

Leitura de arquivos CSV

```python
import pandas as pd

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

# Exibir as primeiras linhas do dataframe
print(df.head())
```

03

Análise Exploratória de Dados

DESCRIÇÃO DO DATAFRAME

Uma visão geral do seu dataframe é crucial para entender os dados que você está trabalhando. O método *describe* fornece estatísticas resumidas.

Descrição do dataframe

```
# Obter estatísticas descritivas
print(df.describe())
```

SELEÇÃO DE COLUNAS

Selecionar colunas específicas pode ser feito de maneira muito simples.

Seleção de colunas

```python
# Selecionar a coluna 'idade'
idades = df['idade']
print(idades.head())
```

FILTRAGEM DE DADOS

Filtrar dados baseados em condições específicas é uma tarefa comum.

Filtragem de dados

```python
# Filtrar dados onde a idade é maior que 30
maiores_que_30 = df[df['idade'] > 30]
print(maiores_que_30.head())
```

SELEÇÃO DE COLUNAS

Selecionar colunas específicas pode ser feito de maneira muito simples.

Seleção de colunas

```python
# Selecionar a coluna 'idade'
idades = df['idade']
print(idades.head())
```

04

Manipulando Dados com NumPy

OPERAÇÕES MATEMÁTICAS

NumPy é a biblioteca padrão para operações matemáticas em Python. Vamos ver como criar arrays e realizar operações básicas.

Operações matemáticas

```python
import numpy as np

# Criar um array numpy
arr = np.array([1, 2, 3, 4, 5])

# Calcular a média
media = np.mean(arr)
print("Média:", media)
```

05

Visualização de Dados com Matplotlib

GRÁFICOS SIMPLES

A visualização de dados é fundamental para entender os padrões e tendências. Vamos criar um gráfico simples usando Matplotlib.

Gráficos simples

```python
import matplotlib.pyplot as plt

# Dados para o gráfico
x = [1, 2, 3, 4, 5]
y = [10, 20, 25, 30, 40]

# Criar um gráfico de linha
plt.plot(x, y)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Gráfico Simples')
plt.show()
```

06

Análise de Dados com Pandas

AGRUPAMENTO DE DADOS

Agrupar dados para obter insights é uma tarefa comum. Usamos o método groupby do pandas para isso.

Agrupamento de dados

```
# Agrupar por uma coluna e calcular a média
grupo = df.groupby('categoria').mean()
print(grupo)
```

07

Lidando com Dados Faltantes

IDENTIFICAÇÃO DE DADOS FALTANTES

Dados faltantes podem distorcer os resultados das análises. É importante identificar e tratar esses dados.

Identificando dados faltantes

```
# Identificar dados faltantes
print(df.isnull().sum())
```

PREENCHIMENTO DE DADOS FALTANTES

Preencher dados faltantes com valores adequados pode ser uma solução.

Preencher dados faltantes

```
# Preencher dados faltantes com a média da coluna
df['idade'].fillna(df['idade'].mean(), inplace=True)
print(df.head())
```

08

Combinação de DataFrames

CONCATENANDO DATAFRAMES

Concatenar Dataframes permite juntar diferentes conjuntos de dados.

Concatenando Dataframes

```python
# Criar dois DataFrames
df1 = pd.DataFrame({'A': [1, 2], 'B': [3, 4]})
df2 = pd.DataFrame({'A': [5, 6], 'B': [7, 8]})

# Concatenar os DataFrames
df_concat = pd.concat([df1, df2])
print(df_concat)
```

MESCLANDO DATAFRAMES

Mesclar Dataframes é útil quando se tem informações complementares em diferentes tabelas.

Mescalando dataframes

```python
# Criar dois DataFrames
df1 = pd.DataFrame({'id': [1, 2], 'nome': ['Alice', 'Bob']})
df2 = pd.DataFrame({'id': [1, 2], 'idade': [24, 27]})

# Mesclar os DataFrames
df_merged = pd.merge(df1, df2, on='id')
print(df_merged)
```

09

Operações Avançadas com NumPy

OPERAÇÕES DE ÁLGEBRA LINEAR

NumPy oferece funções avançadas para operações de álgebra linear, como multiplicação de matrizes.

Operações de Álgebra Linear

```
# Criar duas matrizes
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])

# Multiplicar as matrizes
C = np.dot(A, B)
print(C)
```

TRANSFORMAÇÕES DE ARRAYS

Transformar arrays é uma operação comum em análises de dados.

Transformando Arrays

```
# Criar um array e transformá-lo
arr = np.array([1, 2, 3, 4, 5])
arr_reshaped = arr.reshape((1, 5))
print(arr_reshaped)
```

10

Visualizações Avançadas com Matplotlib

GRÁFICOS DE BARRAS

Gráficos de barras são úteis para comparar categorias.

Criando um gráfico de barras

```
# Dados para o gráfico
categorias = ['A', 'B', 'C']
valores = [10, 20, 15]

# Criar um gráfico de barras
plt.bar(categorias, valores)
plt.xlabel('Categorias')
plt.ylabel('Valores')
plt.title('Gráfico de Barras')
plt.show()
```

HISTOGRAMAS

Histogramas ajudam a visualizar a distribuição de dados.

Criando um histograma

```
# Dados para o histograma
dados = np.random.randn(1000)

# Criar um histograma
plt.hist(dados, bins=30)
plt.xlabel('Valores')
plt.ylabel('Frequência')
plt.title('Histograma')
plt.show()
```

Conclusões

OBRIGADA POR LER ATÉ AQUI!

Esse ebook foi gerado IA, e diagramado por humano.

Esse conteúdo foi gerado com fins didáticos de construção, não foi realizado uma validação cuidadosa humana no conteúdo e pode conter erros gerados por uma IA.