

DADOS QUE IMPULSIONAM

DOMINE A ANÁLISE COM PYTHON DO ZERO



MAYARA CORASSA

O QUE VAMOS APRENDER NESSE EBOOK?

Você aprenderá a transformar dados brutos em insights valiosos. Começaremos pelos fundamentos do Python, passando por técnicas de manipulação de dados com Pandas e criação de visualizações com Matplotlib. Em seguida, exploraremos a análise estatística e aplicações práticas de machine learning. Ao final deste curso, você estará preparado para aplicar seus conhecimentos em projetos reais e tomar decisões mais assertivas com base em dados.

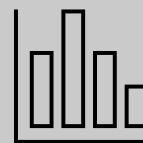


INTRODUÇÃO



O que é análise de dados?

A análise de dados é a arte de transformar números em histórias. Imagine ter um quebra-cabeça gigante com milhões de peças. Cada peça representa um dado, e a imagem completa revela insights valiosos sobre o seu negócio. Ao analisar esses dados, você pode descobrir padrões, tendências e oportunidades que podem impulsionar o seu crescimento.



Por que Python?

O Python é a linguagem de programação preferida para análise de dados por diversas razões. Sua sintaxe clara e concisa facilita o aprendizado, e sua vasta biblioteca de ferramentas, como Pandas e Matplotlib, torna a análise de dados eficiente e divertida. Além disso, a comunidade Python é extremamente ativa, oferecendo suporte e recursos para você se tornar um especialista.





CAPÍTULO 01

Instalando o Ambiente



Preparando seu ambiente

Para começar a analisar dados, precisamos de um ambiente de trabalho configurado corretamente. Imagine-o como um laboratório onde você realizará seus experimentos.

O que faremos:

Instalar o Python: Nossa ferramenta principal para "conversar" com os dados.

Criar um ambiente virtual: Um espaço isolado para cada projeto, evitando conflitos.

Instalar bibliotecas: Ferramentas prontas para análise de dados (NumPy, Pandas, Matplotlib, Seaborn).

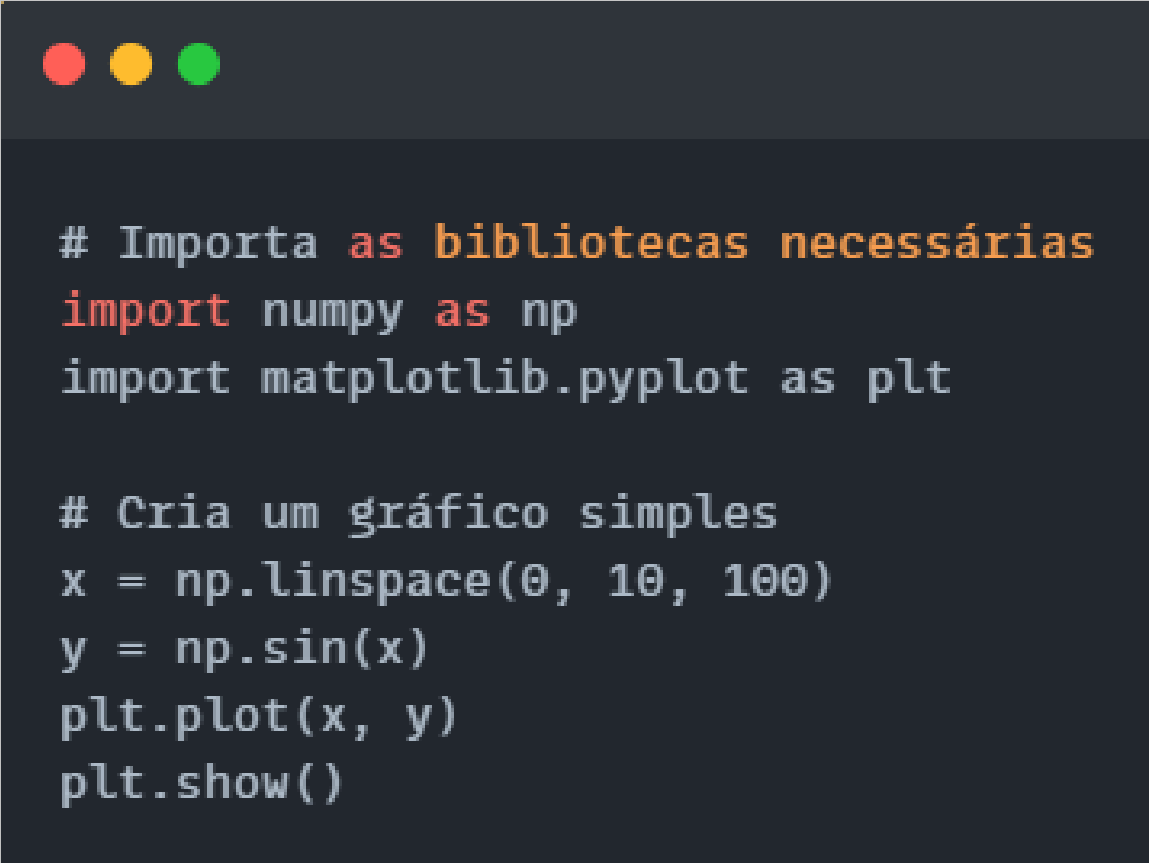
Seus primeiros passos com Python e Jupyter Notebook

Agora que seu ambiente está pronto, vamos começar a explorar suas ferramentas:

O Jupyter Notebook: é como seu caderno de anotações digital. Aqui, você escreve seu código Python, vê os resultados e cria gráficos. Com ele, você:

- ✓ Escreve código em células.
- ✓ Executa o código pressionando Shift+Enter.
- ✓ Visualiza os resultados abaixo da célula.
- ✓ Cria gráficos para entender melhor seus dados.

Exemplo:

A screenshot of a Jupyter Notebook code cell. It features a dark background with light gray text. At the top left, there are three colored circles (red, yellow, green) representing window controls. The code is written in a monospaced font and includes comments in Portuguese. The code imports numpy and matplotlib, creates an array x, calculates the sine of x, and plots it.

```
# Importa as bibliotecas necessárias
import numpy as np
import matplotlib.pyplot as plt

# Cria um gráfico simples
x = np.linspace(0, 10, 100)
y = np.sin(x)
plt.plot(x, y)
plt.show()
```




CAPÍTULO 02

Fundamentos de Python



Python é uma linguagem de programação poderosa e versátil, criada por Guido van Rossum em 1989. Desde então, evoluiu para se tornar uma das linguagens mais populares no mundo da tecnologia, graças à sua sintaxe simples e facilidade de aprendizado. Com Python, é possível desenvolver uma ampla gama de aplicações, desde scripts simples de automação até complexos modelos de aprendizado de máquina.

Variáveis e Tipos de Dados

Em Python, variáveis são usadas para armazenar informações que podem ser usadas e manipuladas ao longo do programa. Vamos explorar os principais tipos de dados:

- **Números Inteiros e Float:** Inteiros são números sem casas decimais, enquanto floats são números que incluem casas decimais.
- **Strings:** São usadas para representar texto e são delimitadas por aspas simples ou duplas.
- **Booleanos:** representam valores lógicos, sendo True para verdadeiro e False para falso.
- **Listas, Tuplas e Dicionários:** São coleções ordenadas de elementos, tuplas são semelhantes a listas, mas imutáveis, e dicionários são coleções de pares chave-valor.

Estruturas de Controle

As estruturas de controle permitem que você direcione o fluxo de execução do seu programa:

- **Condicionais (if, elif, else):** Estas declarações permitem que você execute diferentes blocos de código com base em certas condições.
- **Laços de repetição (for, while):** é usado para iterar sobre uma sequência de elementos.
- **While:** continua executando enquanto a condição especificada for verdadeira.

Funções

- **Definindo e chamando funções:** Você pode definir funções usando a palavra-chave `def`, seguida pelo nome da função e parâmetros entre parênteses.
- **Parâmetros e retorno:** Funções podem receber parâmetros de entrada e retornar valores de saída, permitindo modularizar e organizar seu código de forma mais eficiente
- **Try, except:** `Try` é usado para envolver o código que pode gerar exceções, `except` para definir como tratar essas exceções.



CAPÍTULO 03

Manipulando Dados com Pandas



O Pandas é uma biblioteca essencial para análise de dados em Python, proporcionando ferramentas de alto desempenho para manipulação de grandes volumes de dados.

Importação de dados, Filtros

Uma das funcionalidades mais úteis do Pandas é a capacidade de importar dados de diferentes formatos, como CSV, Excel, SQL, entre outros.

Além disso com Pandas, é fácil acessar e manipular dados específicos.

- **De um arquivo CSV:**


Python

 Copiar

```
df = pd.read_csv("arquivo.csv")  
print(df.head())
```

- **De um arquivo Excel:**


Python

 Copiar

```
df = pd.read_excel("arquivo.xlsx")  
print(df.head())
```

- **Selecionando colunas:**


Python

 Copiar

```
nomes = df["Nome"]  
info_pessoal = df[["Nome", "Idade"]]
```

- **Filtrando linhas:**

Python


 Copiar

```
adultos = df[df["Idade"] > 30]
```

Series e DataFrames

Series: É um array unidimensional que pode armazenar dados de qualquer tipo

Python


 Copiar

```
import pandas as pd

# Criando uma Series
dados = [1, 2, 3, 4, 5]
serie = pd.Series(dados)
print(serie)
```

DataFrame: é uma estrutura de dados bidimensional, similar a uma tabela em uma base de dados ou uma planilha do Excel

Python

 Copiar

```
# Criando um DataFrame
dados = {
    "Nome": ["Alice", "Bob", "Charlie"],
    "Idade": [25, 30, 35],
    "Cidade": ["São Paulo", "Rio de Janeiro", "Belo Horizonte"]
}
df = pd.DataFrame(dados)
print(df)
```




CAPÍTULO 04


Visualizando Dados com Matplotlib e Seaborn



Matplotlib é uma biblioteca fundamental para visualização de dados em Python. Vamos explorar alguns dos gráficos básicos que podemos criar com ela.

Criando gráficos, histogramas, Scatter plots


Python

 Copiar

```
import matplotlib.pyplot as plt


# Criando um gráfico de linha
x = [1, 2, 3, 4, 5]
y = [10, 20, 25, 30, 40]
plt.plot(x, y)
plt.xlabel('Eixo X')
plt.ylabel('Eixo Y')
plt.title('Gráfico de Linha Simples')
plt.show()
```

Python

 Copiar

```
# Criando um histograma
dados = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]
plt.hist(dados, bins=4)
plt.xlabel('Valores')
plt.ylabel('Frequência')
plt.title('Histograma')
plt.show()
```

Python

 Copiar


```
# Criando um scatter plot
x = [1, 2, 3, 4, 5]
y = [10, 20, 25, 30, 40]
plt.scatter(x, y)
plt.xlabel('Eixo X')
plt.ylabel('Eixo Y')
plt.title('Scatter Plot')
plt.show()
```


Personalizando Gráficos com Matplotlib

Personalizar gráficos é importante para torná-los mais informativos e visualmente agradáveis.

- **Adicionando títulos e rótulos:**


Python

 Copiar

```
plt.plot(x, y)
plt.xlabel('Eixo X')
plt.ylabel('Eixo Y')
plt.title('Gráfico de Linha Personalizado')
plt.show()
```

- **Customizando cores e estilos:**

Python

 Copiar

```
plt.plot(x, y, color='green', linestyle='--', marker='o')
plt.xlabel('Eixo X')
plt.ylabel('Eixo Y')
plt.title('Gráfico de Linha Customizado')
plt.show()
```



CAPÍTULO 05

Análise Exploratória de Dados




A Análise Exploratória de Dados (AED) é um processo de resumo das principais características dos dados, frequentemente com métodos visuais. Antes de mergulharmos em modelos preditivos, a AED nos ajuda a formular hipóteses e a guiar nossa análise.

Estatísticas Descritivas

- **Média, Mediana e Moda:**

Python

 Copiar

```
media = df['coluna'].mean()
mediana = df['coluna'].median()
moda = df['coluna'].mode()
```

- **Desvio Padrão e Variância:**

Python

 Copiar

```
desvio_padrao = df['coluna'].std()
variancia = df['coluna'].var()
```

- **Resumo Estatístico:**

Python

 Copiar


```
resumo = df.describe()
print(resumo)
```

Analizando a Correlação entre Variáveis

A correlação nos ajuda a entender a relação entre diferentes variáveis em um conjunto de dados.

- **Matriz de Correlação:**


Python

 Copiar

```
correlacao = df.corr()  
print(correlacao)
```

- **Heatmap de Correlação com Seaborn:**

Python

 Copiar


```
import seaborn as sns  
import matplotlib.pyplot as plt  
  
plt.figure(figsize=(10, 8))  
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')  
plt.title('Heatmap de Correlação')  
plt.show()
```


Visualização Exploratória de Dados

Visualizações ajudam a identificar padrões e outliers nos dados de maneira intuitiva.

- **Histogramas:**


Python

 Copiar

```
df['coluna'].hist(bins=10)
plt.xlabel('Valores')
plt.ylabel('Frequência')
plt.title('Histograma')
plt.show()
```

- **Boxplots:**


Python

 Copiar

```
sns.boxplot(x=df['coluna'])
plt.title('Boxplot')
plt.show()
```

- **Scatter Plots:**

Python

 Copiar

```
plt.scatter(df['variavel1'], df['variavel2'])
plt.xlabel('Variável 1')
plt.ylabel('Variável 2')
plt.title('Scatter Plot')
plt.show()
```



Conclusão



Parabéns por chegar ao final deste e-book sobre Análise de Dados com Python! Ao longo dos capítulos, exploramos diversos conceitos fundamentais que servirão como base sólida para suas futuras análises e projetos.

Próximos passos

Agora que você dominou os fundamentos, o próximo passo é continuar praticando e explorando novos recursos e ferramentas. Considere participar de comunidades online, como o Kaggle, e explorar cursos gratuitos para aprofundar seus conhecimentos. Ferramentas adicionais como NumPy, Scikit-learn e Statsmodels, bem como IDEs como VS Code e Pycharm, podem enriquecer ainda mais seu aprendizado e produtividade.

Lembre-se: a análise de dados é uma jornada contínua de aprendizado e descoberta. Continue praticando, explorando novos métodos e ferramentas, e aplicando seu conhecimento em projetos do mundo real. O mundo dos dados oferece inúmeras oportunidades, e com o que você aprendeu aqui, está bem equipado para aproveitar todas elas.

Boa sorte na sua jornada de análise de dados!