Python数据分析
入门实战指南

——从 Pandas 到数据可视化的完整路径

资料类型	学习教程 / 实战笔记
适用人群	计算机专业本科生、数据分析入门学习者
难度等级	初级 → 中级
预计学习时长	15–20 小时
更新日期	2026年10月

资料简介

本教程面向零基础或初级水平的数据分析学习者，系统讲解使用 Python 进行数据分析的核心技能。内容涵盖 NumPy 数值计算、Pandas 数据处理、Matplotlib 与 Seaborn 数据可视化三大模块，并配套一个完整的电商销售数据分析综合项目，帮助读者将理论知识落地为实战能力。教程特点为：每个知识点均配以可运行的代码示例与输出说明，重点章节附常见坑点提示；综合项目从数据读取、清洗、探索性分析到可视化报告形成完整闭环，学完即可独立完成中小型数据分析任务。建议学习者具备基础 Python 语法知识，按照章节顺序循序渐进学习，每完成一章动手复现代码。本资料适用于计算机专业本科生课程复习、自学数据分析入门、转行数据分析岗位的前期准备。

适用人群

1. 计算机、软件相关专业本科生 / 研究生，需要补充数据分析技能；

2. 有 Python 基础但未系统接触过数据分析的自学者；

3. 准备转行数据分析、数据运营岗位的在校生或职场新人；

4. 需要处理实验数据、问卷数据的理工科研究人员。

使用方法

1. 按章节顺序学习，每章先阅读概念讲解，再动手运行代码示例；

2. 代码示例基于 Python 3.10+、Pandas 2.0、Matplotlib 3.8、Seaborn 0.13；

3. 综合项目建议使用 Jupyter Notebook 环境，边运行边查看结果；

4. 附录速查表可作为日常开发中的快速参考。

目录

第一章	环境搭建与工具准备	3
第二章	NumPy 数值计算基础	5
第三章	Pandas 核心数据结构	8
第四章	数据清洗与预处理	12
第五章	数据可视化入门	16
第六章	综合项目：电商销售数据分析	20
附录 A	常用函数速查表	25
附录 B	常见报错与解决方案	27

第一章 环境搭建与工具准备

1.1 Python 环境安装

推荐使用 Anaconda 发行版，它自带 Python 解释器、包管理器 conda 以及 Jupyter Notebook，免去手动配置环境变量的麻烦。访问 Anaconda 官网下载对应操作系统的安装包，按默认选项安装即可。

1.2 创建虚拟环境

为避免不同项目的依赖版本冲突，建议为每个数据分析项目创建独立的虚拟环境：

```
# 创建名为 da_env 的虚拟环境，指定 Python 3.11
conda create -n da_env python=3.11

# 激活虚拟环境
conda activate da_env

# 安装核心库
conda install numpy pandas matplotlib seaborn jupyter -c conda-forge
```

1.3 启动 Jupyter Notebook

在终端中输入 `jupyter notebook`，浏览器会自动打开本地工作目录页面。点击 New → Python 3 即可新建 Notebook 文件，所有代码和注释保存在同一个 `.ipynb` 文件中。

1.4 验证安装

新建一个 Cell，输入以下代码并运行（Shift + Enter），若输出版本号则说明安装成功：

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

print('NumPy:', np.__version__)
print('Pandas:', pd.__version__)
print('Matplotlib:', plt.matplotlib.__version__)
print('Seaborn:', sns.__version__)
```

> 提示：建议将常用的 `import` 语句放在 Notebook 第一个 Cell 中，每次启动先运行它。

第二章 NumPy 数值计算基础

2.1 为什么需要 NumPy

Python 原生列表（list）在进行大规模数值运算时效率低下，且不支持向量化操作。NumPy（Numerical Python）提供了高效的多维数组对象 ndarray，以及丰富的数学函数库，是 Pandas、Scikit-learn 等更高级库的底层基础。

2.2 创建数组

从列表创建数组：

```
import numpy as np

# 一维数组
arr1d = np.array([1, 2, 3, 4, 5])
print(arr1d) # [1 2 3 4 5]

# 二维数组（矩阵）
arr2d = np.array([[1, 2, 3], [4, 5, 6]])
print(arr2d)
# [[1 2 3]
# [4 5 6]]
```

常用创建函数：

函数	作用	示例输出
np.zeros((3,4))	创建全 0 矩阵	3行4列全为0
np.ones((2,3))	创建全 1 矩阵	2行3列全为1
np.arange(0,10,2)	类似 range，生成数组	[0 2 4 6 8]
np.linspace(0,1,5)	等间距取5个点	[0. 0.25 0.5 0.75 1.]
np.random.rand(3,3)	3x3 均匀分布随机数	0~1之间随机值
np.random.randn(3,3)	3x3 正态分布随机数	均值0方差1

2.3 数组属性与索引

每个数组都有 shape（形状）、dtype（数据类型）、ndim（维度数）等属性：

```
arr = np.array([[1, 2, 3], [4, 5, 6]])
print(arr.shape) # (2, 3) — 2行3列
print(arr.dtype) # int64
```

```
print(arr.ndim) # 2
print(arr.size) # 6 — 元素总数

# 切片：取第0行所有列
print(arr[0, :]) # [1 2 3]
# 取所有行的第1列
print(arr[:, 1]) # [2 5]
# 布尔索引
print(arr[arr > 3]) # [4 5 6]
```

2.4 向量化运算

NumPy 的核心优势是向量化运算——无需编写循环即可对整个数组执行操作，速度比 Python 循环快 10–100 倍。

```
a = np.array([1, 2, 3, 4])
b = np.array([10, 20, 30, 40])

print(a + b) # [11 22 33 44]
print(a * b) # [10 40 90 160] — 对应元素相乘
print(a ** 2) # [1 4 9 16] — 每个元素平方
print(np.dot(a, b)) # 300 — 点积

# 常用统计函数
print(a.mean()) # 2.5 — 均值
print(a.sum()) # 10
print(a.max()) # 4
print(a.std()) # 1.118 — 标准差
```

> 常见坑：数组乘法 * 是逐元素相乘，矩阵乘法请用 @ 或 np.dot()，不要混淆！

第三章 Pandas 核心数据结构

3.1 Series 与 DataFrame

Pandas 是基于 NumPy 构建的数据分析库，提供两种核心数据结构：

Series：一维带标签数组，类似于一列数据；

DataFrame：二维表格型数据结构，相当于一张 Excel 表，是最常用的对象。

3.2 创建 DataFrame

```
import pandas as pd

# 从字典创建
data = {
'姓名': ['张三', '李四', '王五', '赵六'],
'年龄': [22, 25, 28, 24],
'专业': ['计算机', '统计', '金融', '计算机'],
'成绩': [85, 92, 78, 88]
}
df = pd.DataFrame(data)
print(df)
```

3.3 读取外部数据

实际工作中，数据通常保存在 CSV、Excel 或数据库中。Pandas 提供了便捷的读取函数：

```
# 读取 CSV
df = pd.read_csv('data.csv', encoding='utf-8')

# 读取 Excel
df = pd.read_excel('data.xlsx', sheet_name='Sheet1')

# 查看前5行
df.head()
# 查看后5行
df.tail()
# 查看数据概览
df.info()
# 描述性统计
df.describe()
```

3.4 数据选择与筛选

Pandas 提供了多种方式从 DataFrame 中选取数据，最常用的是 loc（按标签）和 iloc（按位置）：

```
# 选择单列
df['成绩']

# 选择多列
df[['姓名', '成绩']]

# 按行标签选择
df.loc[0] # 第0行

# 按位置切片
df.iloc[1:3, :2] # 第1-2行，前2列

# 条件筛选
df[df['成绩'] >= 85] # 成绩>=85的所有行
df[(df['专业']=='计算机') & (df['成绩']>=80)]
```

3.5 新增与修改列

```
# 新增一列：判断是否及格
df['是否及格'] = df['成绩'] >= 60

# 使用 apply 按行计算
df['等级'] = df['成绩'].apply(lambda x: '优' if x>=90 else ('良' if x>=80 else '中'))

# 修改某列的值
df.loc[df['姓名']=='张三', '年龄'] = 23
```

> 记忆口诀：loc 用名字，iloc 用位置，布尔筛选用 []，链式条件加括号。

第四章 数据清洗与预处理

4.1 处理缺失值

真实数据中经常存在缺失值（NaN），常见处理方式有三种：删除、填充、插值。

```
# 查看每列缺失数量
df.isnull().sum()

# 1. 删除含有缺失值的行
df.dropna(axis=0, inplace=True)

# 2. 用固定值填充
df.fillna(0, inplace=True)
df['成绩'].fillna(df['成绩'].mean(), inplace=True)

# 3. 前向填充（用前一个有效值填充）
df.fillna(method='ffill', inplace=True)
```

4.2 处理重复值

```
# 检查重复行
df.duplicated().sum()

# 删除重复行
df.drop_duplicates(inplace=True)

# 按某列去重，保留最后出现的
df.drop_duplicates(subset='姓名', keep='last', inplace=True)
```

4.3 数据类型转换

读取数据后经常需要修正列的数据类型，例如日期列被识别为字符串：

```
# 转换为数值类型
df['年龄'] = pd.to_numeric(df['年龄'], errors='coerce')

# 转换为日期类型
df['日期'] = pd.to_datetime(df['日期'], format='%Y-%m-%d')

# 转换为分类类型
df['专业'] = df['专业'].astype('category')
```

4.4 字符串处理

Pandas 的 str 属性提供了大量字符串处理方法：

```
# 去除首尾空格
df['姓名'] = df['姓名'].str.strip()

# 转为大写
df['专业'] = df['专业'].str.upper()

# 替换字符
df['邮箱'] = df['邮箱'].str.replace('@qq.com', '@email.com')

# 提取（正则）
df['年份'] = df['日期列'].str.extract(r'(\d{4})')
```

4.5 分组聚合（GroupBy）

GroupBy 是数据分析中最常用的操作之一，类似于 SQL 的 GROUP BY：

```
# 按专业分组，计算平均成绩
df.groupby('专业')['成绩'].mean()

# 多列分组 + 多种聚合
df.groupby(['专业', '是否及格']).agg({
'成绩': ['mean', 'max', 'count'],
'年龄': 'median'
})
```

常见坑：groupby 后得到的是 GroupBy 对象，必须再调用聚合函数（mean/sum/count等）才能得到结果。

第五章 数据可视化入门

5.1 Matplotlib 基础

Matplotlib 是 Python 最基础的绘图库，Seaborn 是其高级封装，二者通常配合使用。

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.rcParams['font.sans-serif'] = ['Noto Sans CJK SC'] # 解决中文乱码
plt.rcParams['axes.unicode_minus'] = False # 解决负号显示
```

5.2 常用图表类型

（1）折线图——展示趋势

```
months = ['1月', '2月', '3月', '4月', '5月', '6月']
sales = [120, 150, 180, 160, 210, 250]

plt.figure(figsize=(8, 4))
plt.plot(months, sales, marker='o', color='#2563EB', linewidth=2)
plt.title('上半年销售额趋势')
plt.xlabel('月份')
plt.ylabel('销售额（万元）')
plt.grid(alpha=0.3)
plt.show()
```

（2）柱状图——对比类别

```
majors = ['计算机', '统计', '金融']
avg_score = [86, 89, 80]

plt.figure(figsize=(6, 4))
plt.bar(majors, avg_score, color=['#2563EB', '#10B981', '#F59E0B'])
plt.title('各专业平均成绩对比')
plt.ylabel('平均分')
plt.ylim(0, 100)
plt.show()
```

（3）直方图——分布情况

```
plt.figure(figsize=(6, 4))
plt.hist(df['成绩'], bins=10, color='#2563EB', edgecolor='white')
plt.title('成绩分布直方图')
plt.xlabel('成绩')
plt.ylabel('人数')
plt.show()
```

（4）散点图——两个变量关系

```
plt.figure(figsize=(6, 4))
plt.scatter(df['年龄'], df['成绩'], c='#2563EB', alpha=0.6)
plt.title('年龄与成绩关系')
plt.xlabel('年龄')
plt.ylabel('成绩')
plt.show()
```

5.3 Seaborn 热力图（相关系数）

热力图常用于展示变量之间的相关系数矩阵，快速发现哪些变量相关性强：

```
plt.figure(figsize=(8, 6))
corr = df.corr(numeric_only=True)
sns.heatmap(corr, annot=True, cmap='Blues', fmt='.2f')
plt.title('变量相关系数热力图')
plt.show()
```

> 可视化原则：一张图只表达一个核心信息，标题要直接写结论而非描述图表内容。

第六章 综合项目：电商销售数据分析

6.1 项目背景

假设我们拿到了某电商平台 2025 年的销售订单数据，包含订单号、日期、商品类别、销售额、数量、用户年龄等字段。目标是通过数据分析回答以下业务问题：

1. 全年销售额趋势如何？哪个月份是销售旺季？

2. 哪些商品类别贡献了最多的销售额？

3. 用户年龄分布与消费金额有什么关系？

4. 复购率是多少？哪些用户是高价值客户？

6.2 数据读取与概览

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.rcParams['font.sans-serif'] = ['Noto Sans CJK SC']
plt.rcParams['axes.unicode_minus'] = False

df = pd.read_csv('orders.csv', encoding='utf-8')
print(df.shape) # 查看行列数
print(df.dtypes) # 查看各列类型
print(df.isnull().sum()) # 查看缺失值
df.head()
```

6.3 数据清洗

```
# 转换日期格式
df['订单日期'] = pd.to_datetime(df['订单日期'])
# 提取月份、星期
df['月份'] = df['订单日期'].dt.month
df['星期'] = df['订单日期'].dt.dayofweek

# 删除缺失值
df.dropna(subset=['销售额', '商品类别'], inplace=True)

# 剔除异常订单（销售额<=0）
df = df[df['销售额'] > 0]
```

6.4 分析一：月度销售趋势

```
monthly = df.groupby('月份')['销售额'].sum()

plt.figure(figsize=(10, 5))
plt.plot(monthly.index, monthly.values, marker='o', color='#2563EB')
plt.title('2025年月度销售额趋势')
plt.xlabel('月份')
plt.ylabel('总销售额（元）')
plt.xticks(range(1, 13))
plt.grid(alpha=0.3)
plt.show()

print(f"最高月份：{monthly.idxmax()}月，销售额{monthly.max():,.0f}元")
print(f"最低月份：{monthly.idxmin()}月，销售额{monthly.min():,.0f}元")
```

6.5 分析二：品类销售占比

```
category = df.groupby('商品类别')['销售额'].sum().sort_values(ascending=False)

plt.figure(figsize=(8, 8))
plt.pie(category.values, labels=category.index, autopct='%1.1f%%',
colors=sns.color_palette('Blues_r', len(category)))
plt.title('各商品类别销售额占比')
plt.show()
```

6.6 分析三：用户价值分层（RFM）

RFM 模型是经典的用户价值分析方法：R（最近一次消费）、F（消费频率）、M（消费金额）。

```
# 计算每个用户的 R、F、M
snapshot = df['订单日期'].max() # 以最新订单日为基准
rfm = df.groupby('用户ID').agg({
'订单日期': lambda x: (snapshot - x.max()).days, # R
'订单号': 'count', # F
'销售额': 'sum' # M
}).rename(columns={'订单日期': 'R', '订单号': 'F', '销售额': 'M'})

# 按五分位打分（5分最好）
rfm['R_score'] = pd.qcut(rfm['R'], 5, labels=[5,4,3,2,1])
rfm['F_score'] = pd.qcut(rfm['F'].rank(method='first'), 5, labels=[1,2,3,4,5])
rfm['M_score'] = pd.qcut(rfm['M'], 5, labels=[1,2,3,4,5])

rfm['RFM_total'] = rfm['R_score'].astype(int) + rfm['F_score'].astype(int) +
rfm['M_score'].astype(int)
print(rfm.head())
```

6.7 项目小结

通过这个综合项目，我们完整走了一遍数据分析标准流程：提出问题 → 获取数据 → 清洗数据 → 探索分析 → 可视化呈现 → 得出结论。掌握这套流程后，遇到新的数据集可以按同样的框架拆解。

附录 A 常用函数速查表

功能	Pandas 函数	说明
查看前N行	df.head(n)	默认前5行
查看行列数	df.shape	返回（行数，列数）
查看列类型	df.dtypes	每列的数据类型
缺失统计	df.isnull().sum()	每列缺失值数量
描述统计	df.describe()	均值、标准差、分位数
删除缺失	df.dropna()	axis=0删行，axis=1删列
填充缺失	df.fillna(value)	用指定值填充
删除重复	df.drop_duplicates()	subset指定列
排序	df.sort_values(by="列")	ascending=False降序
分组聚合	df.groupby("列").agg()	常用mean/sum/count
合并表	pd.merge(df1, df2)	how=left/inner/outer
透视表	pd.pivot_table()	index/values/aggfunc
读取CSV	pd.read_csv()	encoding指定编码
写出CSV	df.to_csv()	index=False不写行号

附录 B 常见报错与解决方案

报错信息	原因	解决方案
UnicodeDecodeError	CSV编码不对	read_csv加encoding="gbk"或"utf-8"
KeyError	列名拼写错误/有空格	用df.columns查看真实列名
ValueError: could not convert	列中混入非数字字符	pd.to_numeric(errors="coerce")
SettingWithCopyWarning	对切片结果赋值	用.loc[][]或.copy()
中文乱码	Matplotlib默认字体不支持中文	设置font.sans-serif为中文字体
MemoryError	数据量太大内存不够	分块读取chunksize或改用Dask

—— 全文完 ——

如有疑问欢迎在论坛帖中留言讨论，看到会回复。