8小时学会Python数据分析与可视化

通过这个为期8小时的学习计划

你将掌握Python数据分析和可视化的基本知识和技能

1 - Python数据分析基础

- Python安装与设置
- 安装数据分析库
- 学习资源

2 - 使用Pandas进行数据操作

- Pandas介绍
- 基本操作
- 练习
  - 使用Pandas加载一个CSV文件
  - 进行基本数据操作，如查看前几行数据、统计描述等

3 - 数据清洗与预处理

- 处理缺失值
- 数据转换
- 练习
  - 处理一个数据集中存在的缺失值
  - 进行数据类型转换和重命名列操作

4 - 数据可视化基础

- 数据可视化介绍
- Matplotlib基础
- 练习
  - 使用Matplotlib创建基本图表
  - 添加标题、轴标签和图例

5 - 使用Matplotlib进行可视化

- 高级图表
- 定制化图表
- 练习
  - 创建多个子图
  - 定制化一个图表，使其更加美观

6 - 使用Seaborn进行高级可视化

- Seaborn介绍
- 高级可视化
- 练习
  - 使用Seaborn创建高级图表
  - 定制化Seaborn图表

7 - 实际数据分析项目

- 选择数据集
- 数据分析步骤
- 练习
  - 完成一个简单的数据分析项目，并使用可视化展示结果

8 - 综合练习与复习

- 综合练习
- 复习
- 扩展学习
  - 了解更多的可视化工具（如Plotly）
  - 学习更高级的数据分析技术（如机器学习）

记得多实践，巩固所学内容。

@lifeboat

第1小时：Python数据分析基础
    1. Python安装与设置
    2. 安装数据分析库
    3. 学习资源
    总结
第2小时：使用Pandas进行数据操作
    1. Pandas介绍
    2. 基本操作
    3. 练习
    总结
第3小时：数据清洗与预处理
    1. 处理缺失值
    2. 数据转换
    3. 练习
    总结
第4小时：数据可视化基础
    1. 数据可视化介绍
    2. Matplotlib基础
    3. 练习
    总结
第5小时：使用Matplotlib进行可视化
    1. 高级图表
    2. 定制化图表
    3. 练习
    总结
第6小时：使用Seaborn进行高级可视化
    1. Seaborn介绍
    2. 高级可视化
    3. 练习
    总结
第7小时：实际数据分析项目
    1. 选择数据集
    2. 数据分析步骤
    3. 练习
    总结
第8小时：综合练习与复习
    1. 综合练习
    2. 复习
    3. 扩展学习
    总结

第1小时：Python数据分析基础

目标: 了解Python数据分析的基础知识，并安装必要的库。

1. Python安装与设置

首先，需要确保已经安装了Python。如果你还没有安装，可以访问Python官方网站下载并安装适合你操作系统的版本。

检查是否已安装Python
打开终端或命令提示符，输入以下命令检查是否已经安装Python：

```
python --version
```

如果已经安装，你会看到类似于 `Python 3.x.x` 的输出。

安装Jupyter Notebook
Jupyter Notebook是一个交互式笔记本，广泛用于数据科学和数据分析。使用pip来安装：

```
pip install notebook
```

安装完成后，可以通过以下命令启动Jupyter Notebook：

```
jupyter notebook
```

2. 安装数据分析库

接下来，安装用于数据分析的主要Python库：

- **Pandas**: 强大的数据分析和操作工具
- **Numpy**: 支持高性能数组运算
- **Matplotlib**: 基础的绘图工具
- **Seaborn**: 基于Matplotlib的高级绘图工具

使用pip安装这些库：

```
pip install pandas numpy matplotlib seaborn
```

3. 学习资源

为了更好地学习和理解Python数据分析，可以参考以下资源：

- **Python Data Science Handbook by Jake VanderPlas**: 一本全面的Python数据科学手册，涵盖了Numpy、Pandas、Matplotlib和Scikit-learn等内容。

- **官方文档**:
  - Pandas官方文档
  - Numpy官方文档
  - Matplotlib官方文档
  - Seaborn官方文档

总结

在第1小时内，你需要完成Python和Jupyter Notebook的安装，并安装用于数据分析的主要库（Pandas、Numpy、Matplotlib、Seaborn）。通过参考推荐的学习资源，你可以更深入地理解和掌握Python数据分析的基础知识。完成这些步骤后，你将为后续的学习和数据分析实践打下坚实的基础。

第2小时：使用Pandas进行数据操作

目标: 学习如何使用Pandas进行数据操作。

1. Pandas介绍

什么是Pandas?
Pandas是一个强大的数据分析和操作库，提供了易于使用的数据结构和数据分析工具，特别适合处理结构化数据。Pandas最常用的数据结构是Series和DataFrame。

Pandas的数据结构：Series和DataFrame

- **Series**: 一维数据结构，类似于Python的列表或字典，但具有标签（索引）。
- **DataFrame**: 二维数据结构，类似于电子表格或SQL表格，由多个Series组成。

2. 基本操作

创建DataFrame和Series

```
import pandas as pd

# 创建Series
series = pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])
print(series)

# 创建DataFrame
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
```

```
    'Salary': [50000, 60000, 80000]
}
df = pd.DataFrame(data)
print(df)
```

导入和导出数据（CSV, Excel等）

```
# 导入CSV文件
df = pd.read_csv('data.csv')

# 导出到CSV文件
df.to_csv('output.csv', index=False)

# 导入Excel文件
df = pd.read_excel('data.xlsx')

# 导出到Excel文件
df.to_excel('output.xlsx', index=False)
```

查看数据（head, tail, info, describe）

```
# 查看前几行数据
print(df.head())

# 查看后几行数据
print(df.tail())

# 查看DataFrame的信息
print(df.info())

# 查看统计描述
print(df.describe())
```

3. 练习

练习1：使用Pandas加载一个CSV文件

```python
import pandas as pd

# 加载CSV文件
df = pd.read_csv('your_data.csv')

# 查看前几行数据
print(df.head())
```

练习2：进行基本数据操作，如查看前几行数据、统计描述等

```python
# 查看前几行数据
print(df.head())

# 查看后几行数据
print(df.tail())

# 查看DataFrame的信息
print(df.info())

# 查看统计描述
print(df.describe())
```

总结

通过这些基本操作，你可以开始使用Pandas进行数据加载、查看和分析。这些技能将为你后续的数据清洗、分析和可视化打下基础。继续多练习，熟悉Pandas的基本用法。

第3小时：数据清洗与预处理

目标: 学习如何使用Pandas清洗和预处理数据，以确保数据质量。

1. 处理缺失值

查找缺失值

缺失值是数据分析中的常见问题，可能会影响分析结果。Pandas提供了丰富的工具来查找和处理缺失值。

```
import pandas as pd

# 创建示例数据集
data = {'Name': ['Alice', 'Bob', None, 'Charlie'],
        'Age': [25, None, 30, 35],
        'Salary': [50000, 60000, None, 80000]}
df = pd.DataFrame(data)

# 查找缺失值
print(df.isnull())  # 返回布尔值DataFrame，显示哪些值是缺失的
print(df.isnull().sum())  # 每列的缺失值总数
```

填补或删除缺失值

```
# 删除包含缺失值的行
df_dropped = df.dropna()
print(df_dropped)

# 用指定值填补缺失值
df_filled = df.fillna({'Name': 'Unknown', 'Age': df['Age'].mean(),
'Salary': df['Salary'].median()})
print(df_filled)
```

2. 数据转换

数据转换是指将数据从一种形式转换为另一种形式，例如数据类型转换、重命名列和数据排序。

数据类型转换

```
# 创建示例数据集
data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': ['25', '30', '35'],
        'Salary': [50000, 60000, 80000]}
df = pd.DataFrame(data)

# 查看数据类型
print(df.dtypes)

# 将'Age'列从字符串转换为整数
df['Age'] = df['Age'].astype(int)
print(df.dtypes)
```

重命名列

```
# 重命名列
df_renamed = df.rename(columns={'Name': 'Full Name', 'Age': 'Years',
'Salary': 'Income'})
print(df_renamed)
```

数据排序和过滤

```
# 按'Age'列排序
df_sorted = df.sort_values(by='Age')
print(df_sorted)

# 过滤'Age'大于30的行
df_filtered = df[df['Age'] > 30]
print(df_filtered)
```

3. 练习

练习1：处理一个数据集中存在的缺失值

```
import pandas as pd

# 创建练习数据集
data = {'Product': ['A', 'B', 'C', None],
        'Price': [10.5, None, 25.0, 40.0],
        'Quantity': [100, 150, None, 200]}
df = pd.DataFrame(data)

# 查找缺失值
print(df.isnull().sum())

# 删除包含缺失值的行
df_cleaned = df.dropna()
print(df_cleaned)

# 用平均值填补缺失值
df_filled = df.fillna(df.mean(numeric_only=True))
print(df_filled)
```

练习2：进行数据类型转换和重命名列操作

```
# 创建练习数据集
data = {'Name': ['Alice', 'Bob', 'Charlie'],
        'Age': ['25', '30', '35'],
```

```
        'Salary': ['50000', '60000', '80000']}
df = pd.DataFrame(data)

# 将'Age'和'Salary'列转换为整数类型
df['Age'] = df['Age'].astype(int)
df['Salary'] = df['Salary'].astype(int)
print(df.dtypes)

# 重命名列
df_renamed = df.rename(columns={'Name': 'Employee Name', 'Age':
'Employee Age', 'Salary': 'Employee Salary'})
print(df_renamed)
```

总结

通过以上步骤和练习，你将学习到如何使用Pandas进行数据清洗和预处理，包括查找和处理缺失值、进行数据类型转换、重命名列以及数据排序和过滤。这些技能对于确保数据质量和进行有效的数据分析至关重要。继续多练习，巩固所学内容。

第4小时：数据可视化基础

目标: 了解数据可视化的基本概念和工具。

1. 数据可视化介绍

为什么要进行数据可视化?

数据可视化是数据分析的重要部分，因为它可以:

- 帮助理解和解释复杂的数据
- 揭示数据中的模式、趋势和异常
- 使数据分析结果更直观、更易于沟通

常用的可视化工具

- **Matplotlib**：Python最基础的绘图库，功能强大但语法较为繁琐。
- **Seaborn**：基于Matplotlib的高级绘图库，提供更简洁和美观的绘图接口。

2. Matplotlib基础

创建简单的图表

```
import matplotlib.pyplot as plt
import numpy as np

# 创建数据
x = np.linspace(0, 10, 100)
y = np.sin(x)

# 创建折线图
plt.plot(x, y)
plt.title('Sine Wave')
plt.xlabel('x values')
plt.ylabel('sin(x)')
plt.show()
```

创建柱状图

```
# 创建数据
categories = ['A', 'B', 'C', 'D']
values = [10, 24, 36, 40]

# 创建柱状图
plt.bar(categories, values)
plt.title('Category Values')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.show()
```

创建散点图

```
# 创建数据
x = np.random.rand(50)
y = np.random.rand(50)

# 创建散点图
plt.scatter(x, y)
plt.title('Random Scatter Plot')
plt.xlabel('x values')
plt.ylabel('y values')
plt.show()
```

图表的基本元素

在创建图表时，可以添加标题、标签和图例以增强图表的可读性和信息量。

```
# 折线图示例
plt.plot(x, y, label='Sine Wave')
plt.title('Sine Wave Example')  # 添加标题
plt.xlabel('X Axis')  # 添加x轴标签
plt.ylabel('Y Axis')  # 添加y轴标签
plt.legend()  # 添加图例
plt.show()
```

3. 练习

练习1：使用Matplotlib创建基本图表

```
import matplotlib.pyplot as plt
import numpy as np

# 创建数据
x = np.linspace(0, 10, 100)
y = np.sin(x)

# 创建折线图
plt.plot(x, y)
plt.title('Sine Wave')
plt.xlabel('x values')
plt.ylabel('sin(x)')
plt.show()
```

练习2：添加标题、轴标签和图例

```
import matplotlib.pyplot as plt
import numpy as np

# 创建数据
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)

# 创建折线图
plt.plot(x, y1, label='Sine Wave')
plt.plot(x, y2, label='Cosine Wave')
plt.title('Sine and Cosine Waves')  # 添加标题
plt.xlabel('x values')  # 添加x轴标签
plt.ylabel('Function values')  # 添加y轴标签
plt.legend()  # 添加图例
plt.show()
```

总结

通过这些步骤和练习，你将学习如何使用Matplotlib进行基本的数据可视化操作，包括创建折线图、柱状图和散点图，以及添加图表的基本元素。这些技能将为你后续的数据可视化分析打下坚实的基础。

第5小时：使用Matplotlib进行可视化

目标: 深入学习如何使用Matplotlib进行数据可视化。

1. 高级图表

子图（subplot）

子图可以在一个图形窗口中绘制多个图表。

```
import matplotlib.pyplot as plt
import numpy as np

# 创建数据
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)

# 创建子图
fig, axs = plt.subplots(2, 1)  # 创建2行1列的子图

# 第一个子图
axs[0].plot(x, y1)
axs[0].set_title('Sine Wave')

# 第二个子图
axs[1].plot(x, y2, 'r')
axs[1].set_title('Cosine Wave')

plt.tight_layout()  # 调整子图之间的间距
plt.show()
```

条形图（bar plot）

条形图用于展示分类数据的对比。

```
# 创建数据
categories = ['A', 'B', 'C', 'D']
values = [10, 24, 36, 40]

# 创建条形图
plt.bar(categories, values)
plt.title('Category Values')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.show()
```

直方图（histogram）

直方图用于展示数据的分布情况。

```
# 创建数据
data = np.random.randn(1000)

# 创建直方图
plt.hist(data, bins=30, edgecolor='black')
plt.title('Histogram')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```

2. 定制化图表

调整图表样式

```
# 创建数据
x = np.linspace(0, 10, 100)
y = np.sin(x)

# 调整图表样式
plt.plot(x, y, linestyle='--', color='purple', marker='o')
plt.title('Styled Sine Wave')
plt.xlabel('x values')
plt.ylabel('sin(x)')
plt.grid(True)  # 添加网格
plt.show()
```

添加文本和注释

```
# 创建数据
```

```
x = np.linspace(0, 10, 100)
y = np.sin(x)

# 创建图表
plt.plot(x, y)
plt.title('Sine Wave with Annotation')
plt.xlabel('x values')
plt.ylabel('sin(x)')

# 添加注释
plt.text(2, 0.5, 'Local Max', fontsize=12, color='red')
plt.annotate('Local Min', xy=(7, -1), xytext=(7, -0.5),
             arrowprops=dict(facecolor='black', shrink=0.05))

plt.show()
```

3. 练习

练习1：创建多个子图

```
import matplotlib.pyplot as plt
import numpy as np

# 创建数据
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.tan(x)

# 创建子图
fig, axs = plt.subplots(3, 1, figsize=(8, 12))  # 创建3行1列的子图

# 第一个子图
axs[0].plot(x, y1)
axs[0].set_title('Sine Wave')

# 第二个子图
axs[1].plot(x, y2, 'r')
axs[1].set_title('Cosine Wave')

# 第三个子图
axs[2].plot(x, y3, 'g')
axs[2].set_title('Tangent Wave')
```

```
plt.tight_layout()  # 调整子图之间的间距
plt.show()
```

练习2：定制化一个图表，使其更加美观

```
# 创建数据
x = np.linspace(0, 10, 100)
y = np.sin(x)

# 创建图表
plt.plot(x, y, linestyle='-', color='blue', marker='x')

# 调整样式
plt.title('Customized Sine Wave', fontsize=14, fontweight='bold')
plt.xlabel('x values', fontsize=12)
plt.ylabel('sin(x)', fontsize=12)
plt.grid(True)  # 添加网格
plt.axhline(0, color='black', linewidth=0.5)  # 添加水平线
plt.axvline(0, color='black', linewidth=0.5)  # 添加垂直线
plt.text(5, 0.5, 'Peak', fontsize=12, color='purple')
plt.annotate('Valley', xy=(7, -1), xytext=(7, -0.5),
             arrowprops=dict(facecolor='red', shrink=0.05))

plt.show()
```

总结

通过这些步骤和练习，你将学会如何使用Matplotlib创建高级图表（子图、条形图和直方图），以及如何调整图表样式和添加文本注释。这些技能将帮助你创建更复杂和美观的数据可视化图表，进一步提升你的数据分析能力。

第6小时：使用Seaborn进行高级可视化

目标: 学习如何使用Seaborn进行更高级的数据可视化。

1. Seaborn介绍

Seaborn与Matplotlib的区别

- **Seaborn**：基于Matplotlib构建的高级绘图库，提供了更简洁和美观的绘图接口，特别适合用于统计数据可视化。
- **Matplotlib**：功能强大、灵活性高，但语法较为复杂，需要较多的代码进行美观的图表定制。

Seaborn的高级图表类型

- 热力图（heatmap）
- 箱线图（box plot）
- 小提琴图（violin plot）

2. 高级可视化

热力图（heatmap）

热力图用于显示矩阵数据的可视化，可以很好地展示数据之间的相关性。

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

# 创建示例数据
data = np.random.rand(10, 12)
heatmap_data = pd.DataFrame(data, columns=[f'Month {i+1}' for i in
range(12)])

# 创建热力图
plt.figure(figsize=(10, 8))
sns.heatmap(heatmap_data, annot=True, cmap='viridis')
plt.title('Heatmap Example')
plt.show()
```

箱线图（box plot）

箱线图用于显示数据的分布情况，特别是分位数、异常值等信息。

```
# 创建示例数据
data = pd.DataFrame({
    'Category': ['A', 'A', 'A', 'B', 'B', 'B', 'C', 'C', 'C'],
    'Values': [10, 15, 14, 22, 24, 23, 30, 35, 34]
})

# 创建箱线图
plt.figure(figsize=(8, 6))
sns.boxplot(x='Category', y='Values', data=data)
plt.title('Box Plot Example')
plt.show()
```

小提琴图（violin plot）

小提琴图结合了箱线图和核密度估计，可以更详细地展示数据的分布情况。

```
# 创建示例数据
data = pd.DataFrame({
    'Category': ['A', 'A', 'A', 'B', 'B', 'B', 'C', 'C', 'C'],
    'Values': [10, 15, 14, 22, 24, 23, 30, 35, 34]
})

# 创建小提琴图
plt.figure(figsize=(8, 6))
sns.violinplot(x='Category', y='Values', data=data)
plt.title('Violin Plot Example')
plt.show()
```

3. 练习

练习1：使用Seaborn创建高级图表

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

# 创建示例数据
data = np.random.rand(10, 12)
heatmap_data = pd.DataFrame(data, columns=[f'Month {i+1}' for i in
range(12)])

# 创建热力图
plt.figure(figsize=(10, 8))
sns.heatmap(heatmap_data, annot=True, cmap='coolwarm')
plt.title('Heatmap Exercise')
plt.show()

# 创建示例数据
data = pd.DataFrame({
    'Category': ['A', 'A', 'A', 'B', 'B', 'B', 'C', 'C', 'C'],
    'Values': [10, 15, 14, 22, 24, 23, 30, 35, 34]
})

# 创建箱线图
plt.figure(figsize=(8, 6))
```

```
sns.boxplot(x='Category', y='Values', data=data)
plt.title('Box Plot Exercise')
plt.show()

# 创建小提琴图
plt.figure(figsize=(8, 6))
sns.violinplot(x='Category', y='Values', data=data)
plt.title('Violin Plot Exercise')
plt.show()
```

练习2：定制化Seaborn图表

```
# 创建示例数据
tips = sns.load_dataset("tips")

# 创建箱线图并定制化
plt.figure(figsize=(10, 6))
sns.boxplot(x="day", y="total_bill", hue="smoker", data=tips,
palette="Set3")
plt.title('Customized Box Plot')
plt.xlabel('Day of the Week')
plt.ylabel('Total Bill')
plt.legend(title='Smoker')
plt.show()

# 创建热力图并定制化
flights = sns.load_dataset("flights")
flights_pivot = flights.pivot("month", "year", "passengers")
plt.figure(figsize=(10, 8))
sns.heatmap(flights_pivot, annot=True, fmt="d", cmap="YlGnBu",
linewidths=.5)
plt.title('Customized Heatmap')
plt.show()
```

总结

通过这些步骤和练习，你将学会如何使用Seaborn进行高级数据可视化，包括创建热力图、箱线图和小提琴图，并对图表进行定制化处理。这些技能将帮助你创建更美观和信息丰富的数据可视化图表，提升你的数据分析能力。

第7小时：实际数据分析项目

目标: 应用所学知识进行实际数据分析项目。

1. 选择数据集

从Kaggle或其他数据源选择一个感兴趣的数据集。这里以Kaggle上的“Titanic: Machine Learning from Disaster”数据集为例，下载并导入该数据集进行分析。

2. 数据分析步骤

导入数据并进行初步探索

```
import pandas as pd

# 读取数据
df = pd.read_csv('titanic.csv')

# 查看数据前几行
print(df.head())

# 查看数据基本信息
print(df.info())

# 描述性统计
print(df.describe())
```

数据清洗和预处理

处理缺失值、数据类型转换和特征工程等。

```
# 查找缺失值
print(df.isnull().sum())

# 填补缺失值
df['Age'].fillna(df['Age'].median(), inplace=True)
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
df.drop(columns=['Cabin'], inplace=True)  # 删除缺失值较多的列

# 数据类型转换
df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})

# 重命名列
df.rename(columns={'Survived': 'Survived (1: Yes, 0: No)'},
inplace=True)
```

数据分析和可视化

探索数据之间的关系，并使用可视化展示结果。

```
import matplotlib.pyplot as plt
import seaborn as sns

# 分析性别和生存率之间的关系
plt.figure(figsize=(8, 6))
sns.countplot(x='Survived (1: Yes, 0: No)', hue='Sex', data=df)
plt.title('Survival Rate by Gender')
plt.show()

# 分析年龄和生存率之间的关系
plt.figure(figsize=(8, 6))
sns.histplot(df[df['Survived (1: Yes, 0: No)'] == 1]['Age'], kde=True,
label='Survived')
sns.histplot(df[df['Survived (1: Yes, 0: No)'] == 0]['Age'], kde=True,
label='Not Survived')
plt.title('Age Distribution by Survival Status')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```

3. 练习

完成一个简单的数据分析项目，并使用可视化展示结果

以Titanic数据集为例，通过上述步骤完成一个简单的数据分析项目。

```
# 导入数据
df = pd.read_csv('titanic.csv')

# 初步探索
print(df.head())
print(df.info())
print(df.describe())

# 数据清洗和预处理
df['Age'].fillna(df['Age'].median(), inplace=True)
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
df.drop(columns=['Cabin'], inplace=True)
df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})
df.rename(columns={'Survived': 'Survived (1: Yes, 0: No)'},
inplace=True)

# 数据分析和可视化
```

```
plt.figure(figsize=(8, 6))
sns.countplot(x='Survived (1: Yes, 0: No)', hue='Sex', data=df)
plt.title('Survival Rate by Gender')
plt.show()

plt.figure(figsize=(8, 6))
sns.histplot(df[df['Survived (1: Yes, 0: No)'] == 1]['Age'], kde=True,
label='Survived')
sns.histplot(df[df['Survived (1: Yes, 0: No)'] == 0]['Age'], kde=True,
label='Not Survived')
plt.title('Age Distribution by Survival Status')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.legend()
plt.show()

# 进一步分析
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

总结

通过上述步骤和练习，你将学会如何选择和导入数据集，进行数据清洗和预处理，并应用数据分析和可视化技术完成一个简单的数据分析项目。这些技能将帮助你在实际工作中进行数据分析，提高数据驱动的决策能力。

第8小时：综合练习与复习

目标: 综合练习和复习，巩固所学知识。

1. 综合练习

再进行一个数据分析项目

选择另一个数据集，以进一步巩固所学知识。这里以Kaggle上的“Iris”数据集为例。

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

# 读取数据
df = pd.read_csv('iris.csv')
```

```
# 查看数据基本信息
print(df.head())
print(df.info())
print(df.describe())

# 数据清洗（检查和处理缺失值）
print(df.isnull().sum())  # 检查缺失值
df.dropna(inplace=True)  # 如果有缺失值，删除缺失值行

# 数据分析和可视化
plt.figure(figsize=(8, 6))
sns.countplot(x='species', data=df)
plt.title('Count of Each Species')
plt.show()

plt.figure(figsize=(8, 6))
sns.scatterplot(x='sepal_length', y='sepal_width', hue='species',
data=df)
plt.title('Sepal Length vs Sepal Width')
plt.show()

plt.figure(figsize=(8, 6))
sns.pairplot(df, hue='species')
plt.title('Pairplot of Iris Dataset')
plt.show()

# 热力图显示相关性
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

尝试使用不同的可视化方法展示结果

```
import plotly.express as px

# 使用Plotly创建可视化
fig = px.scatter(df, x='sepal_length', y='sepal_width', color='species',
title='Sepal Length vs Sepal Width')
fig.show()

fig = px.histogram(df, x='species', title='Count of Each Species')
fig.show()

fig = px.box(df, x='species', y='petal_length', title='Box Plot of Petal
Length by Species')
fig.show()
```

2. 复习

回顾前几小时的内容

- Python数据分析基础
- 使用Pandas进行数据操作
- 数据清洗与预处理
- 数据可视化基础
- 使用Matplotlib进行可视化
- 使用Seaborn进行高级可视化
- 实际数据分析项目

查漏补缺，解决疑问

- 回顾笔记和代码
- 找出不熟悉的部分并加强练习
- 提出问题并寻找答案

3. 扩展学习

了解更多的可视化工具

- Plotly: 提供交互式和高质量的图表
- Bokeh: 用于创建互动式和可视化的大规模数据集

学习更高级的数据分析技术

- 机器学习基础：监督学习和非监督学习
- Scikit-learn: 常用的机器学习库
- 实现简单的机器学习模型（如线性回归、决策树等）

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# 数据准备
X = df.drop(columns=['species'])
y = df['species']

# 数据集拆分
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# 训练模型
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)

# 预测
y_pred = model.predict(X_test)

# 评估模型
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
```

总结

通过这8小时的学习计划，你将掌握Python数据分析的基础知识，学会如何使用Pandas进行数据操作和清洗，使用Matplotlib和Seaborn进行数据可视化，并完成实际数据分析项目。同时，通过综合练习和复习，巩固所学知识，了解更多的可视化工具和更高级的数据分析技术。继续练习和探索，逐步提高你的数据分析能力。