

课程设计 1：NBA 球员战绩数据爬取与可视化（lxml）

一、准备过程

NBA 作为全球知名的篮球联赛，其球员信息、赛事数据等数据常常被研究者、开发者和数据分析师用作数据分析。本实验旨在通过编写 Python 爬虫脚本，实现对某 NBA 网站页面的爬取，获取球员数据并将其保存到本地。

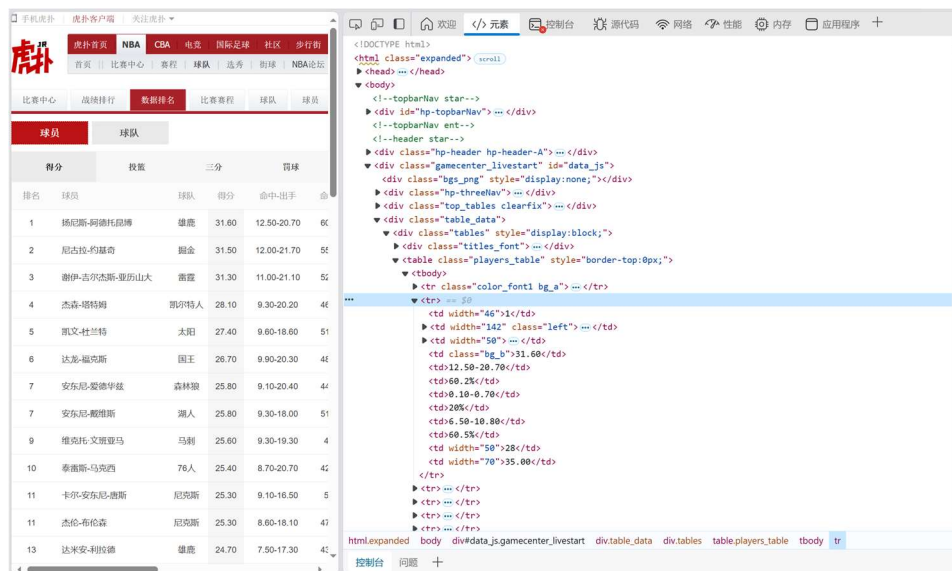


The screenshot shows the NBA website's player statistics page. The table lists the top 5 players based on points per game. The columns include Rank, Player Name, Team, Points, Field Goals, Three-Pointers, Free Throws, Rebounds, Assists, Steals, Blocks, and Minutes.

排名	球员	球队	得分	投篮	三分	罚球	篮板	助攻	盖帽	抢断	上场时间
1	扬尼斯-阿德托昆博	雄鹿	31.60	12.50-20.70	60.2%	0.10-0.70	20%	6.50-10.80	60.5%	28	35.00
2	尼古拉-约基奇	掘金	31.50	12.00-21.70	55.3%	2.30-4.80	47.3%	5.20-6.50	80.6%	31	37.10
3	谢伊-吉尔杰斯-亚历山大	雷霆	31.30	11.00-21.10	52.4%	2.20-6.00	36%	7.10-7.90	88.8%	35	34.50
4	杰森-塔图姆	凯尔特人	28.10	9.30-20.20	46.3%	3.80-10.50	36.2%	5.70-7.00	80.3%	34	36.40
5	凯文-杜兰特	太阳	27.40	9.60-18.60	51.7%	2.30-5.70	40.8%	5.80-6.80	84.2%	25	36.00

图 1：所要爬取的网站

首先，目标网站的内容是允许爬取，实验中所使用的 NBA 网站并不涉及敏感信息，且其内容是公开的，因此可以放心进行爬取。目标页面的数据是结构化的，即表格或列表形式。通过查看源代码，可以发现球员的相关信息（如球员姓名、所属球队、基本数据等）是以<table>标签的形式进行展示的。



The screenshot shows the NBA website's player statistics page with the HTML source code open. The table lists the top 13 players based on points per game. The columns include Rank, Player Name, Team, Points, Field Goals, Three-Pointers, Free Throws, Rebounds, Assists, Steals, Blocks, and Minutes.

排名	球员	球队	得分	投篮	三分	罚球	篮板	助攻	盖帽	抢断	上场时间
1	扬尼斯-阿德托昆博	雄鹿	31.60	12.50-20.70	60.2%	0.10-0.70	20%	6.50-10.80	60.5%	28	35.00
2	尼古拉-约基奇	掘金	31.50	12.00-21.70	55.3%	2.30-4.80	47.3%	5.20-6.50	80.6%	31	37.10
3	谢伊-吉尔杰斯-亚历山大	雷霆	31.30	11.00-21.10	52.4%	2.20-6.00	36%	7.10-7.90	88.8%	35	34.50
4	杰森-塔图姆	凯尔特人	28.10	9.30-20.20	46.3%	3.80-10.50	36.2%	5.70-7.00	80.3%	34	36.40
5	凯文-杜兰特	太阳	27.40	9.60-18.60	51.7%	2.30-5.70	40.8%	5.80-6.80	84.2%	25	36.00
6	达米安-利拉德	开拓者	26.70	9.80-20.30	46%	3.80-8.00	46.3%	6.50-10.80	80.3%	34	36.40
7	安东尼-戴维斯	鹈鹕	25.80	9.10-20.40	44%	3.80-8.00	46.3%	6.50-10.80	80.3%	34	36.40
8	詹姆斯-哈登	火箭	25.80	9.30-18.00	51%	3.80-8.00	46.3%	6.50-10.80	80.3%	34	36.40
9	维克托-文班亚马	马刺	25.60	9.30-19.30	4%	3.80-8.00	46.3%	6.50-10.80	80.3%	34	36.40
10	泰雷斯-马克西	76人	25.40	8.70-20.70	42%	3.80-8.00	46.3%	6.50-10.80	80.3%	34	36.40
11	卡尔-安东尼-唐斯	尼克斯	25.30	9.10-16.50	5%	3.80-8.00	46.3%	6.50-10.80	80.3%	34	36.40
12	杰伦-布朗	凯尔特人	25.30	8.60-18.10	47%	3.80-8.00	46.3%	6.50-10.80	80.3%	34	36.40
13	达米安-利拉德	开拓者	24.70	7.50-17.30	44%	3.80-8.00	46.3%	6.50-10.80	80.3%	34	36.40

图二：网站的结构

在进行数据爬取时，其中最常用的包括 requests 和 lxml。requests 库用于发送 HTTP 请求。lxml 用于解析 HTML 页面，可以方便地通过 XPath 提取目标数据，能方便地定位和提取网页中的特定数据。

二、实现过程

首先导入必要的库，pandas 用来数据处理，matplotlib, seaborn 用来数据可视化，warnings 用来屏蔽警告信息，最后一行用来设置绘图的字體。

```
import requests
from lxml import etree
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import matplotlib
matplotlib.rcParams['font.sans-serif'] = ['SimHei']
```

url1 是网站链接，headers 是请求头信息，使用 requests 的 get 请求方式请求网页信息。

```
# 网站以及请求头信息
url1 = 'https://nba.hupu.com/stats/players/pts'
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 \ (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36'}
# 请求数据
request1 = requests.get(url1, headers=headers)
```

使用 lxml 中的 etree 进行网页解析，并结合 xpath 语法找到所要的 NBA 球员数据。找到数据后将数据保存到 csv 文件中。

```
nba1_etree = etree.HTML(request1.content)
# 解析并写入文件
result =
nba1_etree.xpath('/html/body/div/div/div/table/tbody/tr/td/text()') #
球员数据
result2 =
nba1_etree.xpath('/html/body/div/div/div/table/tbody/tr/td/a/text()')
# 球员姓名和球队
# 保存数据
with open('nba.csv', 'w') as f: # 将球员数据保存到 nba.csv 中
    m = 1
    for i in result[12:]:
        f.write(i)
        m += 1
        if m < 11:
            f.write(',')
        else:
            f.write('\n')
            m = 1
```

```
with open('nba_name.csv','w') as f: # 将球员及球队保存到 nba_name.csv 中
    m = 1
    for i in result2:
        f.write(i)
        m += 1
        if m < 3:
            f.write(',')
        else:
            f.write('\n')
            m = 1
```

三、实验结果

在上个步骤中，爬取得到两份文件，一个是球员的姓名和球队，另一个是球员的数据，将他们合并到一个文件中。

```
# 合并两个表的数据
data = pd.read_csv('nba.csv',header=None)
name = pd.read_csv('nba_name.csv',header=None)
all = pd.concat([name,data],axis=1)
all.head()
```

数据详情如下：

	0	1	0	1	2	3	4	5	6	7	8	9
0	扬尼斯-阿德托昆博	雄鹿	1	32.4	12.70-21.00	60.6%	0.20-0.70	22.2%	6.80-11.40	60.3%	26	35.1
1	谢伊-吉尔杰斯-亚历山大	雷霆	2	31.2	11.00-21.00	52.5%	2.20-6.00	35.6%	7.00-7.90	88.5%	34	34.4
2	尼古拉-约基奇	掘金	3	31.0	11.80-21.30	55.3%	2.30-4.70	47.9%	5.20-6.50	80.5%	30	36.9
3	杰森-塔特姆	凯尔特人	4	28.2	9.40-20.20	46.4%	3.90-10.70	37%	5.50-6.90	79.6%	32	36.3
4	达龙-福克斯	国王	5	26.7	9.90-20.30	48.6%	2.00-6.30	32.1%	4.90-6.10	80.8%	35	37.3

图三：数据详情

下面是对数据进行简单的清洗。首先修改列名。其次对命中率，三分命中率，罚球命中率进行浮点数处理，用于后面的可视化。最后将清洗好的数据保存为 csv 文件。

```
# 修改列名称
columns_n = ['姓名','球队','排名','得分','命中-出手','命中率','命中-三分','三分命中率','命中-罚球','罚球命中率','场次','上场时间']
all.columns = columns_n
# 数据清洗
all['命中率'] = all['命中率'].str.replace('%','').astype(float) / 100
all['三分命中率'] = all['三分命中率'].str.replace('%','').astype(float) / 100
all['罚球命中率'] = all['罚球命中率'].str.replace('%','').astype(float) / 100
# 将清洗好的数据保存在 nba_all.csv 中
all.to_csv('nba_all.csv', index=False)
```

最后进行数据的可视化。fig 创建一个画布，包含两张绘制在同一张图中的子图，设置标题，对 x 轴坐标上的文字旋转 90 度，绘制背景网格，创建颜色。

```
# 数据可视化
fig, ax1 = plt.subplots(figsize=(12, 6), dpi=300)
plt.title('NBA 战绩排行 Top50 数据')
plt.xticks(rotation=90)
plt.grid(True,axis='both', alpha=0.5)
color = sns.color_palette('rocket',n_colors=7)
```

绘制场次，得分，上场时间条形图。设置标签，调整字体大小，显示图例。

```
# 绘制第一个画布
sns.barplot(x='姓名', y='场次', data=all, ax=ax1, color=color[4],
label='场次')
sns.barplot(x='姓名', y='得分', data=all, ax=ax1, color=color[5],
label='场均得分')
sns.barplot(x='姓名', y='上场时间', data=all, ax=ax1, color=color[6],
fill=False, label='上场时间')
ax1.set_ylabel('得分/场均得分/上场时间',size=12)
ax1.set_xlabel('球员',size=12)

ax1.legend(bbox_to_anchor=(0.2,0.15))
```

绘制命中率，三分命中率，罚球命中率散点图。绘制坐标轴，显示图例。

```
# 绘制第二个画布
ax2 = ax1.twinx()
sns.scatterplot(x='姓名', y='命中率',
                data=all, ax=ax2,
                color=color[0], marker='o',
                label='综合命中率')
sns.scatterplot(x='姓名', y='三分命中率',
                data=all, ax=ax2,
                color=color[1], marker='d',
                label='三分命中率')
sns.scatterplot(x='姓名', y='罚球命中率',
                data=all, ax=ax2,
                color=color[2], marker='v',
                label='罚球命中率')
ax2.set_ylabel('命中率',size=12)
ax2.legend(loc='upper left', bbox_to_anchor=(0.85,0.15))
```

可视化结果图下图所示：

