**中北大学计算机科学与技术学院**

数据采集技术

大作业说明书

|  |  |
| --- | --- |
| **课程名称：** | **数据采集技术** |
| **学 院：** | **计算机科学与技术学院** |
| **学 号：** | **2207054212** |
| **姓 名：** | **乔婷婷** |
| **专 业：** | **数据科学与大数据技术** |
| **设计题目：** | **中国软科大学排名可视化系统** |
| **指导教师：** | **井超** |

2025年 4 月

**中国软科大学排名可视化系统**

**摘 要**

随着中国软科大学排名的不断更新与发展，如何高效分析和展示大学排名数据，成为教育领域的重要课题。传统的人工分析方法不仅效率低下，而且难以处理大量复杂的排名数据。因此，利用数据分析与可视化技术构建智能分析系统，能够更好地呈现大学排名的变化与趋势，具有重要的现实意义。

本项目设计并实现了一个基于Python的中国软科大学排名可视化系统，系统主要包含四大模块：数据采集模块通过模拟浏览器请求，从中国软科网站抓取各大学的排名、综合评分、学科排名等数据，备份为结构化Excel文件。数据库模块采用MySQL存储结构化数据，支持高效查询，并结合Excel文件提供备份，确保数据可移植性。数据分析模块利用Pandas进行数据清洗与统计分析，包括排名趋势、学科排名分布及各项指标的相关性分析。可视化模块通过Matplotlib生成静态图表，基于Django框架构建Web界面，便于用户进行实时查询与展示。

本项目整合了数据采集、MySQL数据库、数据分析与可视化技术，形成了一套完整的数据处理流程，为类似应用场景提供了可复用的技术方案。系统能够帮助教育工作者和学生快速获取最新的大学排名信息，并为决策提供数据支持。

**关键词：**数据采集、可视化、Django框架

**China Soft Science University Ranking Visualization SystemAbstract**

As the China Academic Ranking of World Universities (ARWU) continues to evolve, the efficient analysis and presentation of university ranking data has become an important topic in the field of education. Traditional manual analysis methods are not only inefficient but also struggle to handle the large and complex ranking data. Therefore, constructing an intelligent analysis system using data analysis and visualization technologies can better present the changes and trends in university rankings, which is of significant practical importance.

This project designs and implements a Python-based China Soft Science University Ranking Visualization System, which consists of four main modules: The data collection module simulates browser requests to scrape data such as rankings, overall scores, and subject rankings from the China Soft Science website, and stores it in a MySQL database, with structured Excel backups. The database module uses MySQL to store structured data, supporting efficient queries, and combines Excel files for backup, ensuring data portability. The data analysis module uses Pandas for data cleaning and statistical analysis, including ranking trends, subject ranking distribution, and correlation analysis of various indicators. The visualization module generates static charts using Matplotlib, creates dynamic interactive charts with Plotly, and displays the university rankings and their trends intuitively. The system also builds a web interface using the Django framework, allowing users to query and display data in real-time.

This project integrates data collection, MySQL databases, data analysis, and visualization technologies, forming a complete data processing workflow that provides a reusable technical solution for similar application scenarios. The system helps educators and students quickly access the latest university ranking information and provides data support for decision-making.

**Keywords:** Data collection, visualization, Django framework

目 录

[1 引言 1](#_Toc195173970)

[1.1 研究背景 1](#_Toc195173971)

[1.2 国内外现状 1](#_Toc195173972)

[1.3 本文结构 2](#_Toc195173973)

[2 概述 4](#_Toc195173974)

[2.1 设计内容 4](#_Toc195173975)

[2.2 意义 4](#_Toc195173976)

[2.3 设计要求 4](#_Toc195173977)

[2.4 工作要求 -5](#_Toc195173978)

[3 相关技术介绍 5](#_Toc195173979)

[3.1开发环境及技术栈 5](#_Toc195173980)

[3.2 Django框架 5](#_Toc195173981)

[3.3 Requests 6](#_Toc195173982)

[3.4 数据可视化库 7](#_Toc195173983)

[4 需求分析 7](#_Toc195173984)

[4.1 项目概要 7](#_Toc195173985)

[4.2 功能需求 8](#_Toc195173986)

[4.3 系统总体用例 9](#_Toc195173987)

[4.4 可行性分析 9](#_Toc195173988)

[4.4.1 技术可行性分析 9](#_Toc195173989)

[4.4.2 经济可行性分析 10](#_Toc195173990)

[4.4.3 操作可行性分析 10](#_Toc195173991)

[4.5 未来需求 10](#_Toc195173992)

[5 概要设计 11](#_Toc195173993)

[5.1 概念结构流程图设计 11](#_Toc195173994)

[5.2 整体流程图设计 12](#_Toc195173995)

[5.3 系统模块设计 12](#_Toc195173996)

[5.3.1 数据采集设计 12](#_Toc195173997)

[5.3.2 数据存储设计 12](#_Toc195173998)

[5.3.3 数据分析设计 13](#_Toc195173999)

[5.3.4 数据可视化设计 13](#_Toc195174000)

[6 详细设计 14](#_Toc195174001)

[6.1 数据采集设计 14](#_Toc195174002)

[6.1.1 核心设计思想 14](#_Toc195174003)

[6.1.2 核心代码 15](#_Toc195174004)

[6.2 数据存储设计 15](#_Toc195174005)

[6.2.1 核心设计思想 15](#_Toc195174006)

[6.2.2 核心代码 16](#_Toc195174007)

[6.3 数据分析设计 16](#_Toc195174008)

[6.3.1 核心设计思想 16](#_Toc195174009)

[6.3.2 核心代码 16](#_Toc195174010)

[6.4 数据可视化设计 18](#_Toc195174011)

[6.4.1 核心设计思想 18](#_Toc195174012)

[6.4.2 核心代码 19](#_Toc195174013)

[7 调试分析 21](#_Toc195174014)

[7.1 数据采集 21](#_Toc195174015)

[7.2 数据存储 21](#_Toc195174016)

[7.3 数据分析与可视化 22](#_Toc195174017)

[8 总结与体会 25](#_Toc195174018)

[参 考 文 献 27](#_Toc195174019)

# 1 引言

## 1.1 研究背景

大学排名在全球教育领域中扮演着至关重要的角色，成为学生、教育工作者和研究人员选择学术机构的重要参考。各种大学排名系统中，中国软科大学排名（ARWU）是最具影响力的排名之一，尤其在中国，它根据学术声誉、师资力量、研究成果、教学质量等多个指标对高校进行全面评估。

随着中国高等教育的快速发展，对大学排名数据进行高效分析和展示的需求日益增加。然而，传统的排名分析方法通常依赖于人工数据收集和解读，这不仅耗时且容易出错。此外，每年发布的大量排名数据，需要以直观且易于理解的方式进行处理和可视化，这仍然是一个重要的挑战。

在这种背景下，利用数据分析与可视化技术为解决这一问题提供了一个有效的方案。通过自动化数据采集、应用先进的分析方法以及使用互动式可视化技术，教育机构、学生及其他相关利益方可以深入了解影响大学排名的因素，跟踪排名变化趋势，并做出更为科学的决策。

本项目旨在通过整合数据采集、分析和可视化技术，开发一套“中国软科大学排名可视化系统”，以应对上述挑战，帮助用户高效、直观地理解和展示大学排名数据。

## 1.2 国内外现状

大学排名系统作为全球高等教育评价的重要工具，近年来在国内外得到了广泛关注和应用。国外的大学排名系统，如QS世界大学排名、泰晤士高等教育世界大学排名（THE）、US News全球大学排名等，已成为衡量大学综合实力和学术影响力的重要指标。它们通过科学的方法，结合学术声誉、教学质量、科研能力、国际化程度等多维度数据，为全球的学生、学者和教育工作者提供了选择高校的重要参考。

然而，这些排名系统虽然在国际上具有较大的影响力，但在评估方法、数据来源和权重设置上存在一定的争议。例如，排名中可能过于偏重科研成果和国际声誉，而忽略了地方性大学或特定学科的优势。此外，排名的标准和权重在不同的国家和地区有所不同，这使得排名数据的可比性受到一定影响。

在中国，软科大学排名（ARWU）作为国内最具影响力的大学排名之一，逐渐成为了衡量中国高校发展水平的重要标准。软科大学排名采用了独特的评价方法，注重学术研究成果和师资力量的综合考量，具有较高的学术认可度和影响力。然而，现有的排名展示和分析手段仍然相对传统，且大多数数据分析和排名展示依赖人工操作，难以满足日益增长的实时数据需求和多维度分析需求。

随着大数据、人工智能及数据可视化技术的快速发展，国内外越来越多的高校排名系统开始探索自动化、智能化的排名数据处理与展示方法。比如，越来越多的高校排名机构开始采用数据可视化技术，通过交互式图表、趋势分析等方式，使得排名数据的呈现更加直观和易于理解。同时，基于大数据技术的实时更新与分析也成为未来排名系统发展的趋势。

尽管如此，国内的大学排名数据可视化系统仍然处于起步阶段，尤其是针对中国软科大学排名的数据可视化分析，仍缺乏成熟的应用系统。因此，构建一个基于数据采集、分析和可视化技术的“中国软科大学排名可视化系统”，不仅可以弥补当前系统的不足，也能为广大用户提供更加高效、直观的排名信息查询与分析工具。

## 1.3 本文结构

本文共分为八个主要部分：

第一章为引言部分，本章主要介绍课题的研究背景与意义。随着高等教育信息化的发展，大学排名数据的公开透明和智能化展示成为高校管理与学生择校的重要参考依据。传统的排名数据展示方式存在交互性差、更新不及时、分析手段单一等问题。因此，构建一个集数据采集、分析与可视化于一体的系统具有现实需求。本章还通过对国内外高校排名及其可视化研究的现状进行梳理，归纳已有成果，指出存在的不足，并提出本项目的研究目标与创新之处。

第二章为概述部分，本章对系统进行整体规划，明确项目的核心功能，包括排名数据的采集、结构化存储、趋势分析与图形化展示。结合中国软科大学排名的特点，阐述本系统在教育信息服务领域的实际应用价值。同时，本章还提出了项目的主要技术路线、开发目标与阶段性成果安排，为后续开发奠定方向。

第三章为相关技术介绍部分，本章对项目开发中涉及的核心技术进行深入解析，包括Python编程语言、Django Web框架、MySQL数据库、数据可视化库（如Matplotlib）、数据处理工具（如Pandas、NumPy）等。通过对各技术的选型原因及其在本项目中的应用场景进行说明，帮助读者全面了解系统技术基础。

第四章为需求分析部分，本章从用户角度出发，梳理系统的功能性与非功能性需求，明确数据采集范围（如年份、学校、指标等）、用户交互方式以及系统响应速度、兼容性等性能指标。结合用例图和功能模块图展示系统整体交互逻辑，并从技术可行性、经济可行性和操作可行性三方面进行分析，为系统开发提供理论支撑。

第五章为概要设计部分，本章构建系统的整体架构模型，采用模块化设计思路划分各个子系统的职责。介绍数据采集模块、数据管理模块、分析模块与可视化展示模块的基本结构，并通过架构图与流程图描绘系统各模块之间的交互关系，确保系统逻辑清晰、结构合理。

第六章为详细设计部分，本章深入讲解各模块的具体实现方式，包括数据抓取流程、数据库结构设计、分析算法实现及图表渲染逻辑。展示部分核心代码与接口设计，明确系统数据流转路径及前后端联动方式，使开发人员能够依据该设计内容完成系统搭建。

第七章为调试分析部分，本章对系统各功能模块进行全面测试，验证数据采集准确性、数据库连接稳定性、分析结果的正确性及图表展示的完整性。同时对用户界面进行功能测试与兼容性测试，提出改进建议，并通过性能分析对系统进行优化，提升整体运行效率与用户体验。

第八章为总结与体会部分，本章对整个项目开发过程进行总结，归纳在技术实现、系统设计及实际应用中取得的成果与经验，分析项目中遇到的问题及应对策略。最后，对系统的未来发展方向进行展望，如引入实时更新机制、拓展更多维度的排名指标、支持移动端访问等，提出后续优化与扩展建议。

此外，文末参考文献部分列出本文所参考的相关文献信息，为进一步深入了解相关领域提供参考资料。

# 2 概述

## 2.1 设计内容

使用Python web主流开发框架Django，以Pycharm作为开发和运行测试环境：

1. 准备数据：从中国软科大学排名网站爬取2021至2024年前300所大学排名数据，数据量约为2000条约；
2. 预处理数据：对爬取的数据进行清洗和转换，包括格式统一，根据学校名称对比各年数据，确保数据的准确性；
3. 设计架构：设计架构：根据系统功能分析，分段设计功能模块，并绘制系统用例图；
4. 实现系统：现系统：准备开发环境，搭建基于 MySQL 的数据库，实现后端数据接口和前端界面显示，包括大学排名数据表和图表可视化；
5. 代码复现：确保 Python 环境安装完善，包括 Django、Pandas、Matplotlib等依赖库，运行爬虫代码获取数据，启动 Django 项目；
6. 确定设计方案，确定程序模块，并进行详细的设计说明；
7. 写出大作业说明书。

## 2.2 意义

随着信息化进程的加速，高级教育资源分布及其影响力成为社会关注的焦点。本系统采集中国软科大学排名数据，并培育用户对大学排名及其变化的观念。

通过 Django 框架构建系统后端，实现数据接入和管理；并利用 Matplotlib 和 Seaborn 展示指标变化趋势、排名高低分布、看看省份分布、类型分布等统计结果，为高考生、教育部门和研究者提供数据依据和准确分析基础。

## 2.3 设计要求

在设计大学排名可视化系统时，需综合考虑数据爬取、数据处理、分析以及前后端开发技术，确保数据准确、系统稳定、功能完善。

首先，需熟悉 Python 爬虫技术，包括 requests 等库的使用，同时用 Pandas 对数据进行清洗、序列排列、格式转换。

然后，培育 Django 框架的完整系统，设计后端接口、前端页面和统计图展示功能，并结HTML/CSS/JavaScript 实现响应式前端布局。统计分析模块需包括排名变化、省份分布、总分差异分析等，利用 Matplotlib 和 Seaborn 实现统计图表。

最后，代码编写需符合规范，注重模块化和可读性，确保系统易于维护和升级。整体设计需兼顾实用性与交互性，为用户提供清晰、直观的数据洞察，助力健康科普内容的优化与传播。

## 2.4 工作要求

项目初期，需完成需求分析，明确系统目标，规划数据爬取和处理方案，实现基本 Django 框架初始化。

中期，要实现数据爬取和编码处理，进行数据库存储，实现基本的后端模型、规划架构，并利用 Matplotlib/ Seaborn 展示数据分析图。

后期，完善代码结构，进行模块集成和运行测试，确保各功能正常运行；同时编写完整的项目说明文档和系统设计文档，供数据分析和开发者参考和使用。

# 3 相关技术介绍

## 3.1开发环境及技术栈

前端开发语言：HTML、CSS、JavaScript；

前端开发平台：Pycharm；

前端开发框架：Django框架；

后端开发语言：Python；

后端开发平台：Pycharm；

## 3.2 Django框架

Django 是一个用 Python 编写的高级 Web 应用开发框架，它遵循 MTV 架构模式，即模型（Model）、模板（Template）和视图（View），以“用最少的代码做更多的事情”为核心理念。Django 提供了强大的 ORM（对象关系映射）系统，开发者可以通过定义 Python 类与数据库进行交互，而无需直接编写 SQL 语句，大大提高了开发效率和代码的可维护性。它内置的管理后台可以自动生成数据管理界面，无需开发人员额外编写后台代码，这一点在很多企业级应用开发中极具实用价值。

Django 框架自带众多功能模块，包括用户认证、权限管理、会话控制、表单处理、缓存管理、国际化支持等，开发者可以快速搭建起完整的 Web 系统。其路由系统灵活可配，支持函数视图和类视图（CBV），便于实现复杂的业务逻辑。在安全性方面，Django 默认启用了防范 SQL 注入、XSS 攻击、CSRF 攻击等机制，保障 Web 应用在网络环境下的安全运行。同时，Django 的模板系统功能强大，支持模板继承、变量传递、自定义过滤器等特性，使得前后端协作更加高效。

Django 拥有良好的扩展性和活跃的开发社区，支持丰富的第三方库和插件，涵盖图表生成、数据可视化、REST API、任务调度等多种应用场景，能够满足从中小型网站到企业级系统的开发需求。它广泛应用于新闻门户、内容管理系统、教育平台、电子商务网站、数据可视化平台等领域。总的来说，Django 是一个结构清晰、功能齐全、开发高效的 Web 框架，非常适合对开发效率、安全性和系统稳定性有较高要求的项目。

## 3.3 Requests

Requests是Python中最受欢迎的HTTP库之一，它提供了一种简单、优雅的方式来发送HTTP请求和处理响应。作为一个第三方库，Requests极大地简化了与Web服务的交互过程，使开发者能够轻松完成GET、POST、PUT、DELETE等常见HTTP操作，而无需关注底层的复杂细节。它的设计遵循“人性化”理念，通过简洁的API和清晰的文档，让网络请求变得直观易用。

Requests的核心优势在于其高度封装的功能和丰富的特性。它自动处理URL编码、会话保持、SSL证书验证等繁琐任务，并支持Cookie持久化、超时设置、文件上传等高级功能。此外，Requests内置的JSON解析器能够直接将响应内容转换为Python字典，显著提升了开发效率。其灵活的请求参数配置（如headers、params、data等）也使得定制化请求变得非常简单。

在实际应用中，Requests被广泛用于爬虫开发、API调用、数据采集等场景。无论是与RESTful服务交互，还是模拟表单提交，Requests都能以极少的代码实现复杂的需求。与其他HTTP库相比，它的低学习门槛和强大功能使其成为Python开发者的首选工具。通过结合异常处理机制，Requests还能有效应对网络波动或服务错误，确保程序的健壮性。总之，Requests以其高效、可靠的特点，成为现代Python生态中不可或缺的组件。

## 3.4 数据可视化库

在数据分析和探索性研究（EDA）中，可视化是理解数据分布、趋势和关联性的关键手段。Python生态提供了多种强大的可视化工具，如Matplotlib、Seaborn它们各具特色，适用于不同的场景。

Matplotlib 是 Python 语言中最为广泛使用的图表绘制工具之一，特别适用于静态图表的生成。Matplotlib 提供了一系列灵活的功能，可以绘制从基本的线图、散点图、柱状图到更复杂的直方图、饼图、等高线图等多种类型的图表。同时，Matplotlib 允许对图表进行高度定制，用户可以自定义图表的各个方面，包括轴标签、标题、图例、颜色、线型、标注等，使得生成的图表不仅符合用户的需求，也可以在不同的输出环境下（如网页、PDF、PNG等）显示出最佳效果。Matplotlib 支持与 Pandas、NumPy 等数据处理库无缝集成，能够直接操作 DataFrame 数据，并将数据转化为各种图形输出。因此，Matplotlib 成为了数据分析和科学研究中不可或缺的可视化工具。

Seaborn 是基于 Matplotlib 库之上的一个数据可视化工具，旨在简化复杂的图形绘制过程，并增强图表的美观性。Seaborn 提供了丰富的内置主题、颜色调色板以及高级的图形类型，使得用户在创建统计图形时可以获得更加清晰和美观的视觉效果。与 Matplotlib 不同，Seaborn 更加专注于统计数据的可视化，尤其是在处理复杂的数据关系时非常有效。例如，Seaborn 提供了直接绘制热力图、箱线图、条形图、分布图等功能，并且能够方便地显示数据之间的相关性、分布和趋势。Seaborn 使得用户能够快速而高效地创建统计图形，特别是在处理 Pandas DataFrame 类型的数据时，用户可以直接利用 DataFrame 的列作为参数传入，简化了数据的处理和图形的绘制。它内建的主题和颜色配置使得生成的图表更加美观，符合现代设计审美，同时也增强了图表的可读性。

这两者的结合使用，不仅在生成基础的图表时提供了强大的功能，还能通过定制化的样式、主题和颜色，提升图表的美观性和易读性，从而使得图表更加符合实际需求，并且在展示时提供更好的视觉体验。

# 4 需求分析

## 4.1 项目概要

中国软科大学排名可视化系统是一个基于Python Django框架开发的Web应用，旨在对中国软科大学排名数据进行采集、处理、分析和可视化展示。系统通过爬虫从中国软科大学排名网站获取大学排名数据，包括排名、学校名称、类型、总分等关键信息，并利用多种可视化工具生成直观的图表，帮助用户快速洞察大学排名的变化趋势、各省份大学分布、学校类型分布等。

本项目采用模块化设计，分为数据采集、数据处理、数据分析和Web展示四大模块。数据采集模块使用Python爬虫抓取中国软科大学排名网站的数据，存储至MySQL数据库，并根据需要导出为Excel文件；数据处理模块利用Pandas进行数据清洗和预处理，结合Matplotlib、Seaborn生成静态图表；数据分析模块对大学排名的变化趋势、学校类型分布、各省份大学的分布情况等进行深入分析，帮助用户发现排名变化的规律和影响因素；Web展示模块基于Django框架构建后端服务，前端采用Bootstrap实现响应式布局，并通过Jinja2模板动态渲染分析结果。此外，系统在设计时考虑到中文适配与异常处理，确保系统稳定运行并优化用户体验。。

该系统通过量化分析中国软科大学排名数据，帮助教育机构、学生和公众更好地了解大学排名的动态变化，为选择学校提供有力的数据支持，最终促进教育决策的科学性和透明度。

## 4.2 功能需求

中国软科大学排名可视化系统的功能需求包括但不限于以下内容：

数据采集与存储：

通过Python爬虫从中国软科大学排名网站抓取大学排名数据，包括学校名称、排名、类型、得分等结构化数据，导出为Excel文件供后续分析。爬虫需支持分页抓取、请求头模拟、随机延迟防封禁、数据去重和异常处理。

数据分析与可视化：

（1）排名分析：该图展示了各省大学在 2021 年与 2024 年之间的排名变化趋势，主要通过堆叠条形图来表示大学排名的变化（上升、下降或稳定）。

（2）省份分布分析：不同省份的大学数量分布，采用了柱状图形式，图中展示了各省的大学数量，可以直观反映各省在大学排名中的分布情况。

（3）类型分布分析：展示了每年不同类型大学的分布情况，通过饼图展示每种类型所占的比例。

（4）相关性热力图：各大学排名指标（如综合排名、学科排名等）之间的相关性，使用热力图来展示各指标间的线性关系。

Web展示与交互：

1. 前端页面：前端页面主要通过Django模板系统（Jinja2）展示数据和图表。在数据展示部分，系统通过后端传递的大学排名数据生成HTML表格，展示各年份、排名、学校类型等信息。在图表展示部分，后端生成的图表（如排名变化趋势、各省大学数量分布等）以图片的形式保存在静态目录，并通过HTML标签嵌入到页面中。前端页面使用了基本的HTML和CSS来确保图表和数据的整齐展示。
2. 后端服务：后端基于Django框架，负责处理前端请求，调用数据分析函数生成图表并保存到静态目录。生成的图表和数据会通过Django视图传递给前端，在视图中，使用Django的模板引擎将图表路径、数据表格和相应的描述文本嵌入到HTML页面中。
3. 用户交互：系统通过Django的路由机制提供不同页面之间的切换，用户可以点击导航栏切换到不同的数据展示或图表展示页面。在图表展示页面，用户可以查看不同年份的大学排名变化趋势、各省大学数量分布等图表。在图表和数据表格上，用户通过简单的页面交互进行查看，图表和数据可以直接显示，无需复杂的交互功能。

## 4.3 系统总体用例

系统总体用例图如图4.1所示。

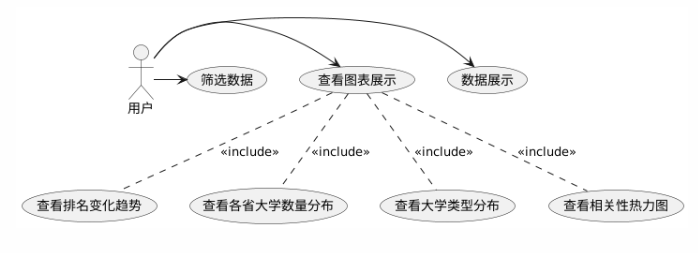


图4.1 系统总体用例图

## 4.4 可行性分析

### 4.4.1 技术可行性分析

本系统采用成熟稳定的技术栈实现，具有充分的技术可行性。Python生态系统提供了完整的解决方案：Django作为轻量级Web框架可快速搭建服务接口，Pandas库具备强大的数据处理能力，Matplotlib和Seaborn的组合既能生成静态图表又可实现交互式可视化。数据采集方面，Requests技术组合能够有效抓取目标网站的数据。当前技术方案已覆盖从数据采集、存储、分析到展示的全流程需求，团队成员掌握的Python+Django开发技能也能有效降低技术风险。

### 4.4.2 经济可行性分析

本项目具有显著的成本效益优势。开发阶段主要依赖开源技术栈，无需支付商业软件授权费用。硬件方面，系统初期可在普通服务器甚至云主机上部署运行，基础设施投入较低。爬虫模块替代了传统人工数据收集方式，大幅降低了数据获取成本。数据分析自动化减少了专业数据分析师的人力依赖，长期使用将产生可观的人力成本节约。系统提供的可视化分析能力可帮助运营团队快速定位优质内容方向，避免无效的内容投入，间接提升了资源利用效率。考虑到健康科普领域的持续发展需求，系统建成后的边际使用成本趋近于零，投资回报率具有吸引力。

### 4.4.3 操作可行性分析

本系统设计充分考虑了实际使用场景，具有良好的操作可行性。前端界面采用响应式布局和CSS框架，确保在不同设备上都能获得良好的浏览体验。可视化图表提供交互功能，用户无需专业培训即可通过鼠标操作获取详细数据。管理员后台集成数据更新、系统监控等常用功能，操作流程设计符合常规认知。系统采用渐进式加载策略，即使在网络条件不佳时也能保证基本功能可用。文档和工具提示的完善设计进一步降低了使用门槛，使不具备技术背景的运营人员也能独立完成数据分析任务。现有的使用习惯调研表明，目标用户群体对类似系统的接受度和适应性较高。

## 4.5 未来需求

未来该系统可根据用户反馈与技术发展进行持续优化和功能拓展。首先，在数据采集方面，可拓展为自动化定时更新功能，确保大学排名数据的实时性与完整性。其次，数据存储可以引入分布式数据库或数据仓库，以应对数据规模扩大和并发访问的需求。在数据分析模块中，可以结合机器学习算法实现对大学排名变动趋势的预测与建模，提升数据的决策支持能力。而在可视化方面，可引入交互式图表（如ECharts、Plotly）以及移动端适配，使用户在不同终端上都能便捷获取所需信息。此外，还可考虑用户个性化功能，如收藏高校、比较多校排名曲线等，进一步提升系统的智能化与用户体验。

# 5 概要设计

## 5.1 概念结构流程图设计

系统整体分为四个模块：数据采集、数据存储、数据分析和数据可视化。系统概念结构设计流程图如图5.1所示。

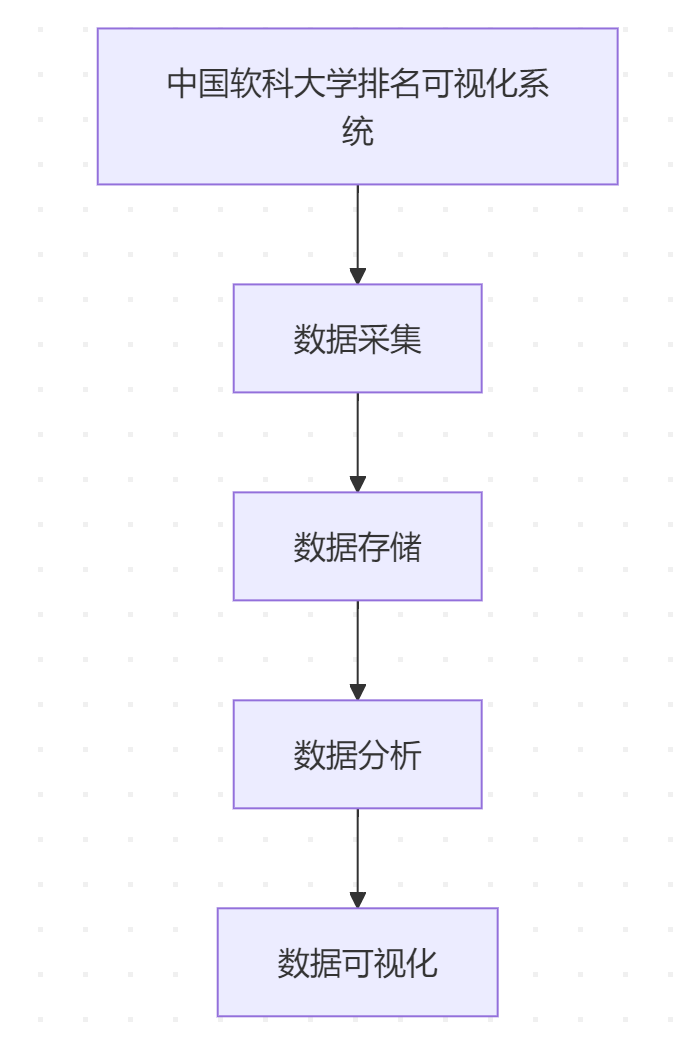


图5.1 系统概念结构设计流程图

## 5.2 整体流程图设计



图5.2 系统整体流程图

## 5.3 系统模块设计

### 5.3.1 数据采集设计

数据采集模块是中国软科大学排名可视化系统的核心组成部分，主要负责从目标网站（如中国软科大学排名）抓取与大学排名相关的数据。该模块利用Python编写的爬虫程序自动化采集数据，确保数据的及时性和准确性。爬虫程序通过模拟浏览器请求，抓取网页中的大学名称、排名、省份、学校类型等信息。数据采集完成后，导出excel，便于后续的数据处理和分析。为了避免封禁，爬虫程序会设置随机延迟，并采取请求头模拟等反爬虫措施，确保稳定运行。

### 5.3.2 数据存储设计

数据存储模块负责存储从数据采集模块获取的所有信息，并为后续的数据处理和分析提供可靠的数据支持。在中国软科大学排名可视化系统中，采用MySQL数据库作为主要的存储方案，确保数据的结构化存储和高效访问。

数据库设计中，包括了多个表格用于存储不同的大学排名数据。每个表格都会包含大学名称、排名、年度、省份、学校类型等字段。此外，数据表将进行适当的索引设计，以优化查询性能。为了保证数据的完整性和一致性，数据采集过程中的每一条记录都会进行严格的验证，避免错误数据进入系统。

### 5.3.3 数据分析设计

数据分析设计模块是中国软科大学排名可视化系统的重要组成部分，旨在对存储在数据库中的大学排名数据进行深入分析，并为可视化展示提供支撑。数据分析模块主要使用Python中的数据分析库，如Pandas、Numpy等，来处理、分析和提取有价值的信息。

在数据分析设计中，首先进行数据清洗和预处理，确保数据的质量和一致性。常见的数据清洗操作包括去除缺失值、处理重复数据、标准化数据格式等。处理后的数据会通过各种统计方法进行分析，如排名变化趋势分析、大学类型和地区分布分析等。

具体分析包括以下几个方面：

排名变化分析：通过对比不同年份的排名数据，分析大学排名的波动情况，识别排名上升或下降的趋势。这有助于揭示影响排名变化的因素，并为教育机构的策略调整提供数据支持。

省份分布分析：分析不同省份的大学数量及其排名分布，帮助识别各省高校在全国的竞争力和发展趋势。

学校类型分布分析：对不同类型的学校进行分类统计，分析综合性大学、理工大学、医学类大学等不同类型学校在排名中的表现及分布。

相关性分析：通过计算排名与其他指标（如总分、学科分数等）之间的相关性，揭示大学排名变化的潜在因素。

数据分析模块的设计目标是提供清晰的数据洞察，帮助用户理解排名背后的规律和趋势，为决策提供数据支持。。

### 5.3.4 数据可视化设计

数据可视化设计模块是中国软科大学排名可视化系统中的关键部分，旨在通过图表和图形的形式将数据分析结果直观地展示给用户，帮助用户更清晰地理解大学排名的分布和变化趋势。该模块使用Python的可视化工具，如Matplotlib、Seaborn等，将分析结果以图表形式展示在Web界面上。在具体实现中，排名变化可视化通过折线图或柱状图展示不同年份的大学排名变化，帮助用户清楚地看到排名上升或下降的趋势。省份分布可视化则通过柱状图或地图展示各省份的大学数量及其排名分布，让用户一目了然地看到各省高校在全国范围内的竞争态势。学校类型分布通过饼图或条形图展示不同类型学校的分布情况，分析不同类型大学的表现差异。相关性热力图用于展示排名和其他因素（如总分、学科分数等）之间的相关性，帮助用户理解排名变化背后的潜在因素。数据可视化设计的核心目标是通过简洁、直观的图表，帮助用户快速理解复杂的数据，发现大学排名背后的规律和趋势，从而为决策提供依据。。

# 6 详细设计

## 6.1 数据采集设计

### 6.1.1 核心设计思想

数据采集设计模块的核心目标是从中国软科大学排名网站自动获取所需的大学排名数据，并将数据存储到指定的文件中。该模块的实现使用了Python编写，采用了requests库进行网络请求，获取排名数据的API接口为https://www.shanghairanking.cn/api/pub/v1/bcur?bcur\_type=11&year={year}，该接口可以返回指定年份的大学排名数据。模块通过get\_university\_data函数，从指定年份（2021-2024年）获取大学排名数据。每次请求时，构建正确的请求头并发送GET请求以防止被反爬机制封禁，获取的数据为JSON格式，需要进行解析，提取其中的大学排名信息。数据从API中获取后，利用process\_data函数将每个年份的数据进行处理，生成统一格式的字典。每个大学的多个信息（如学校名称、排名、省份、类型、总分等）都被整合到一个字典里，并最终转换为Pandas的DataFrame结构，便于后续的数据处理与分析。然后将整理好的DataFrame存储到本地的Excel文件中，确保数据的持久化存储，便于后续分析和可视化。数据存储位置为D:\univ\_site\_package\_full\data\2021-2024年前300所学校(一个sheet)(7个特征).xlsx，如果目录不存在，程序会自动创建。整个数据采集模块设计注重自动化执行，能够定期抓取并更新数据，数据在抓取过程中经过清洗和处理，以生成标准格式的数据文件，便于后续分析与可视化，同时也具备错误处理机制，确保在出现异常时能及时反馈并不中断整个流程。。

### 6.1.2 核心代码

def get\_university\_data(year, num=300):

url = f'https://www.shanghairanking.cn/api/pub/v1/bcur?bcur\_type=11&year={year}'

headers = {

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safari/537.36 Edg/101.0.1210.53'

}

try:

response = requests.get(url, headers=headers, timeout=30)

response.raise\_for\_status()

data = json.loads(response.text)

return data['data']['rankings'][:num]

except Exception as e:

print(f"获取{year}年数据失败: {e}")

return None6.2 数据存储设计

## 6.2 数据存储设计

### 6.2.1 核心设计思想

数据存储的核心思想是在从爬虫获取的原始数据基础上，进行格式化处理，将其转换为结构化数据，并将其保存到本地文件中，以便后续分析和可视化。在代码中，首先通过 get\_university\_data 函数获取每个年份的大学排名数据。随后，使用 process\_data 函数处理每年的数据，将其转化为统一格式的 DataFrame。处理好的数据被存储为 Excel 文件，具体操作通过 pandas 库中的 to\_excel 方法完成，确保数据能够持久化存储在本地文件系统中。文件路径由代码中的 os.makedirs 确保正确创建，避免因目录不存在导致的存储失败。在存储过程中，异常会被捕获并打印，确保数据存储过程中出错时能及时反馈错误信息。

### 6.2.2 核心代码

def save\_to\_excel(df, filename):

try:

df.to\_excel(filename, index=False)

print(f"数据已成功保存到 {filename}")

except Exception as e:

print(f"保存文件失败: {e}")

## 6.3 数据分析设计

### 6.3.1 核心设计思想

数据分析设计的核心思想是对从爬虫获取并存储的数据进行深入分析，提取出具有价值的统计信息和趋势，帮助用户更好地理解大学排名的变化及其背后的因素。在实现过程中，首先是通过 process\_data 函数对所有年份的大学排名数据进行清洗和整理，确保数据格式统一。随后，针对不同的数据维度，如省份、学校类型和排名变化等，使用 pandas 进行数据的分组、聚合和统计。通过这种方式，可以对不同年份的排名数据、类型分布、地域分布等进行分析，识别出每个指标的趋势变化和关联性。在分析过程中，利用 matplotlib 和 seaborn 等可视化工具生成图表，帮助用户更直观地了解数据背后的规律。最终，数据分析部分的目标是为可视化展示提供支持，并帮助用户发现潜在的规律和趋势，从而提高决策的科学性和准确性。。

### 6.3.2 核心代码

def process\_data(all\_data):

"""

处理所有年份的数据并生成DataFrame

:param all\_data: 包含所有年份数据的字典

:return: 处理好的DataFrame

"""

records = []

for i in range(len(all\_data['2021'])):

record = {

'Unnamed: 0': i + 1, # 序号

'2024排名': all\_data['2024'][i]['rankOverall'] if i < len(all\_data['2024']) else '',

'学校名称': all\_data['2021'][i]['univNameCn'], # 假设学校名称不变

'2024省份': all\_data['2024'][i]['province'] if i < len(all\_data['2024']) else '',

'2024类型': all\_data['2024'][i]['univCategory'] if i < len(all\_data['2024']) else '',

'2024总分': all\_data['2024'][i]['score'] if i < len(all\_data['2024']) else '',

'2024英文名': all\_data['2024'][i]['univNameEn'] if i < len(all\_data['2024']) else '',

'2024学校标签': all\_data['2024'][i]['univTags'] if i < len(all\_data['2024']) else '',

'2023排名': all\_data['2023'][i]['rankOverall'] if i < len(all\_data['2023']) else '',

'2023省份': all\_data['2023'][i]['province'] if i < len(all\_data['2023']) else '',

'2023类型': all\_data['2023'][i]['univCategory'] if i < len(all\_data['2023']) else '',

'2023总分': all\_data['2023'][i]['score'] if i < len(all\_data['2023']) else '',

'2023英文名': all\_data['2023'][i]['univNameEn'] if i < len(all\_data['2023']) else '',

'2023学校标签': all\_data['2023'][i]['univTags'] if i < len(all\_data['2023']) else '',

'2022排名': all\_data['2022'][i]['rankOverall'] if i < len(all\_data['2022']) else '',

'2022省份': all\_data['2022'][i]['province'] if i < len(all\_data['2022']) else '',

'2022类型': all\_data['2022'][i]['univCategory'] if i < len(all\_data['2022']) else '',

'2022总分': all\_data['2022'][i]['score'] if i < len(all\_data['2022']) else '',

'2022英文名': all\_data['2022'][i]['univNameEn'] if i < len(all\_data['2022']) else '',

'2022学校标签': all\_data['2022'][i]['univTags'] if i < len(all\_data['2022']) else '',

'2021排名': all\_data['2021'][i]['rankOverall'],

'2021省份': all\_data['2021'][i]['province'],

'2021类型': all\_data['2021'][i]['univCategory'],

'2021总分': all\_data['2021'][i]['score'],

'2021英文名': all\_data['2021'][i]['univNameEn'],

'2021学校标签': all\_data['2021'][i]['univTags']

}

records.append(record)

return pd.DataFrame(records)

## 6.4 数据可视化设计

### 6.4.1 核心设计思想

数据可视化设计的设计思想聚焦于通过静态图表清晰展示大学排名数据和分析结果，帮助用户更好地理解各类信息，并支持多维度的数据探索。根据“中国软科大学排名可视化系统”的需求，数据可视化部分包括四种主要图表形式，分别用于展示排名分析、省份分布、大学类型分布和相关性分析。设计的核心思想是通过简洁直观的图表帮助用户快速洞察数据背后的规律，同时提供清晰的分析和易于操作的界面。

图表设计遵循直观、简洁的原则，保证每一类数据都能通过合适的图表形式进行展示。例如，使用柱状图展示不同年份各省大学数量分布，清晰地反映各省的教育资源分布情况；使用饼图展示大学类型分布，帮助用户了解不同类型大学的数量比例；利用折线图展示各省大学排名变化趋势，揭示排名的动态变化；最后，使用热力图展示各个数值型指标之间的相关性，便于揭示不同因素之间的内在关系。

所有图表均为静态生成，以便于用户直观地查看分析结果，同时系统设计保证了图表展示的清晰度和易用性。例如，通过设置合适的标题、轴标签和数据注释，使得每个图表在视觉上都具有良好的可读性。此外，为了满足用户多样化的需求，图表之间的布局遵循统一的设计规范，确保不同图表风格的一致性，从而提供一种和谐的视觉体验。

### 6.4.2 核心代码

（1）各省大学数量分布图

print("Generating chart 1: 各省大学数量分布")

fig, axs = plt.subplots(2, 2, figsize=(18, 15))

for i, year in enumerate(years):

counts = df[f'{year}省份'].value\_counts()

ax = counts.plot(kind='bar', color='skyblue', ax=axs[i // 2, i % 2],

title=f'{year}年各省大学数量分布')

ax.set\_xlabel('省份' if i // 2 == 1 else '')

ax.set\_ylabel('数量')

ax.set\_xticklabels(counts.index, rotation=90, ha='center')

for p in ax.patches:

ax.annotate(f"{p.get\_height():.0f}", (p.get\_x() + 0.2, p.get\_height() + 0.5))

plt.tight\_layout()

plt.savefig(os.path.join(output\_dir, 'univ\_by\_province.png'))

plt.close()

（2）大学类型分布图（饼图）

print("Generating chart 2: 大学类型分布（饼图）")

fig, axs = plt.subplots(2, 2, figsize=(12, 10))

for ax, year in zip(axs.flat, years):

type\_counts = df[f'{year}类型'].value\_counts()

type\_counts.plot(kind='pie', autopct='%1.1f%%', ax=ax, startangle=90)

ax.set\_title(f'{year}年类型分布')

ax.set\_ylabel('')

plt.tight\_layout()

plt.savefig(os.path.join(output\_dir, 'univ\_type\_pie.png'))

plt.close()

（3）各省大学排名变化趋势图

print("Generating chart 3: 各省大学排名变化趋势")

df['Ranking\_Change'] = df.apply(

lambda row: '上升' if row['2024排名'] < row['2021排名'] else

('下降' if row['2024排名'] > row['2021排名'] else '稳定'),

axis=1

)

rank\_change = df.groupby(['2021省份', 'Ranking\_Change']).size().unstack()

rank\_change.plot(kind='bar', stacked=True, figsize=(14, 6))

plt.title('各省大学排名变化趋势（2021 vs 2024）')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.savefig(os.path.join(output\_dir, 'ranking\_change\_by\_province.png'))

plt.close()

（4）相关性热力图

print("Generating chart 4: 相关性热力图")

numeric\_cols = df.select\_dtypes(include=np.number).columns

plt.figure(figsize=(12, 8))

sns.heatmap(df[numeric\_cols].corr(), annot=True, cmap='coolwarm', fmt=".2f")

plt.title('排名指标相关性')

plt.tight\_layout()

plt.savefig(os.path.join(output\_dir, 'heatmap\_corr.png'))

plt.close()

# 7 调试分析

## 7.1 数据采集

在调试数据采集的过程中，首先需要确保API请求能够正常发送并接收到有效的响应。可以通过打印出response.status\_code来确认请求是否成功。如果状态码不是200，那么可能是API服务不可用或发生了错误。在请求中设置适当的timeout，并在出现异常时能够捕捉错误信息并打印出来，方便排查问题。其次，要验证API返回的数据结构，确保从data['data']['rankings']中提取的数据格式正确且符合预期。如果API返回的字段结构发生变化，代码将无法正常运行，因此要在调试时打印出返回的原始数据进行验证。此外，在数据存储时，确保每个字段的数据类型与期望的一致，比如学校排名应为整数类型，学校名称应为字符串类型。处理数据时，如果遇到缺失值或格式不符的情况，可以加入额外的检查和清理步骤。在get\_university\_data函数中的异常处理非常重要，通过捕捉requests.exceptions.RequestException等异常，可以避免程序因网络问题或API错误崩溃。在数据采集的过程中，为了提高代码的健壮性，可以增加日志输出，便于跟踪每个步骤的执行情况。如果出现问题时打印出请求的URL、响应内容以及关键的返回数据，帮助定位问题。

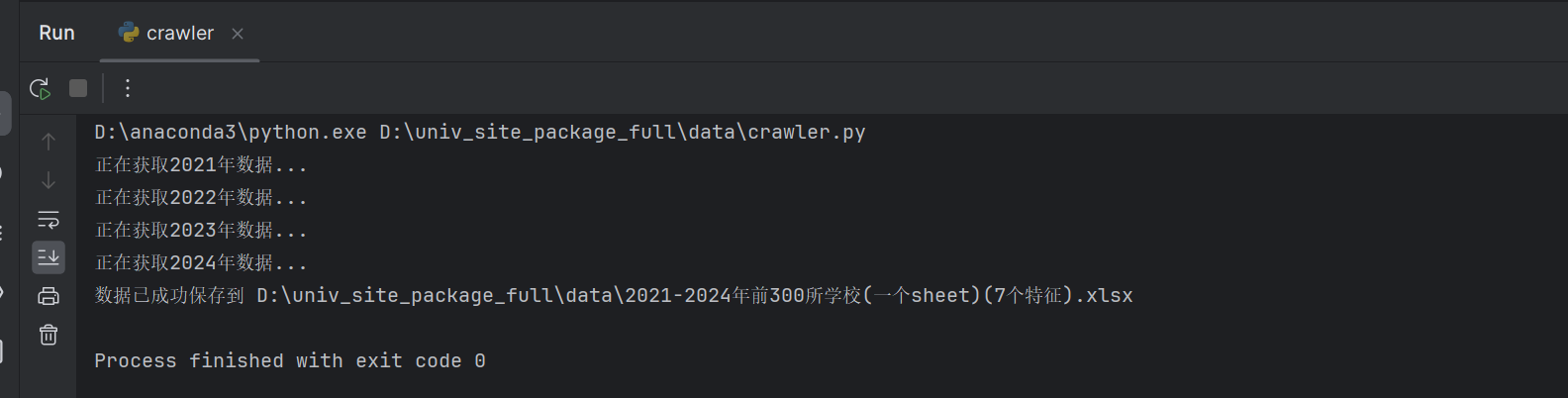


图7.1 爬取网站数据

## 7.2 数据存储

在数据存储的设计思想中，核心目标是确保从外部数据源（如API）获取到的数据能够准确、完整地存储到本地数据库或文件中，供后续使用。在这个过程中，首先需要处理和整理数据，确保每个字段的格式一致且完整。获取到的数据经过整理后会存入一个统一的格式中，这个格式通常是DataFrame，能够方便地进行进一步分析和操作。

为了保证数据能够成功存储，我们需要确保文件存储路径存在，如果路径不存在，必须通过创建路径的方式来避免文件存储失败。在数据存储时，应当检查数据的质量，避免保存缺失或格式错误的数据，这可能导致后续的数据分析和可视化出现问题。

在存储过程中，除了关注数据内容的准确性，还要注意存储方式的效率和可维护性。对于较大的数据集，可以使用压缩文件格式或数据库存储方式以提高性能。在文件存储时，通常会使用标准的格式如Excel或CSV，以便于后续读取和处理。

综上所述，数据存储设计的核心思想是确保数据的正确性、完整性和便于后续使用，同时考虑存储路径和文件格式的选择，以确保系统的稳定性和数据的易用性。

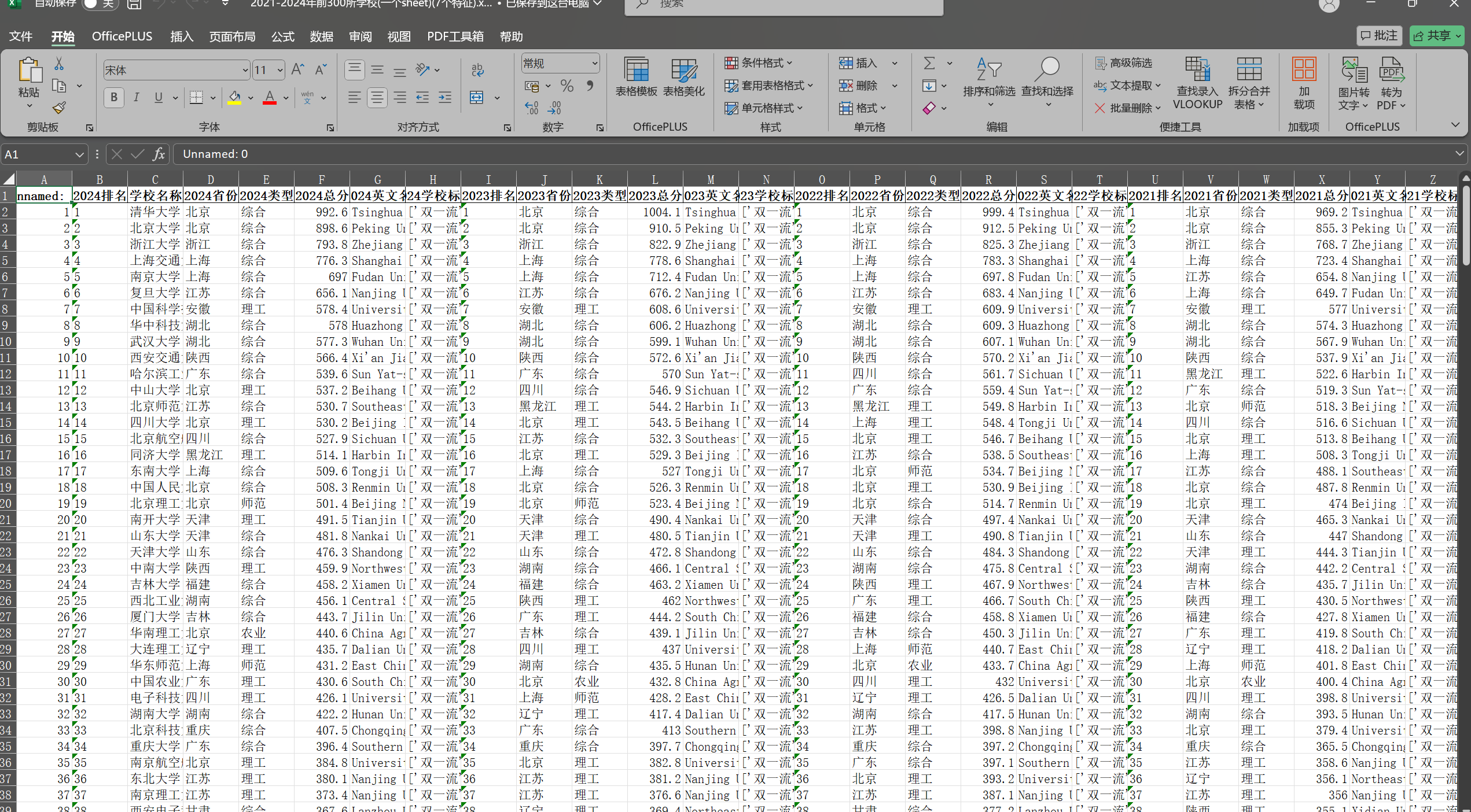


图7.2 Excel中的数据

## 7.3 数据分析与可视化

数据分析可视化的调试主要围绕数据处理的准确性和可视化效果的呈现。在数据分析阶段，使用pandas库处理数据时，要对数据的类型和值进行检查。例如，在计算统计量时，确保数据类型为数值型，避免因数据类型错误导致计算结果异常。在进行数据可视化时，首先要验证图表的类型是否选择合适。

各省大学数量分布图（柱状图）：这个图表可以帮助用户清晰地了解各省大学的数量分布情况。通过柱状图展示各省份的大学数量，可以直观地看到哪些省份的高等教育资源较为集中，哪些省份的大学数量较少。这对于了解中国各地区大学的分布、教育资源的区域差异具有重要意义。

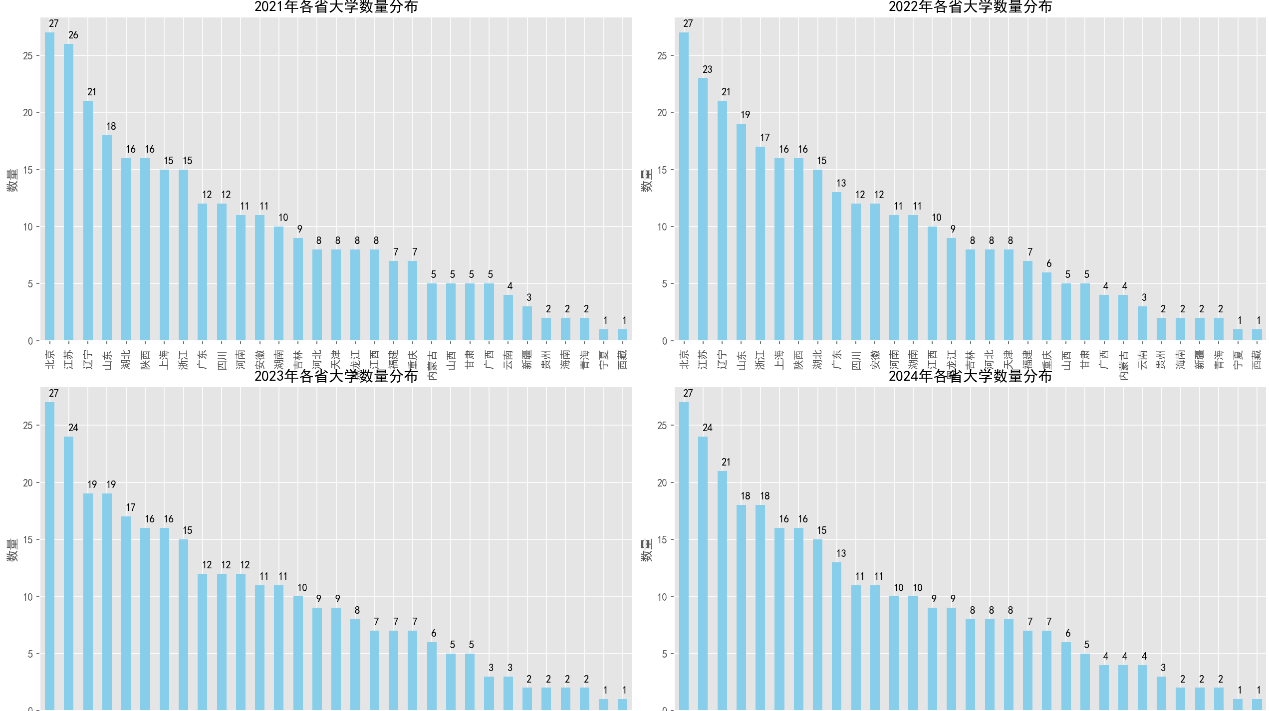


图7.3 各省大学数量分布图

大学类型分布图（饼图）： 饼图能够清晰地显示每年不同类型大学的占比，展示哪些类型的大学更为常见。通过饼图，用户能够直观地看到不同类型大学的数量比例，比如综合性大学与理工类大学的分布情况，有助于分析大学的多样性。

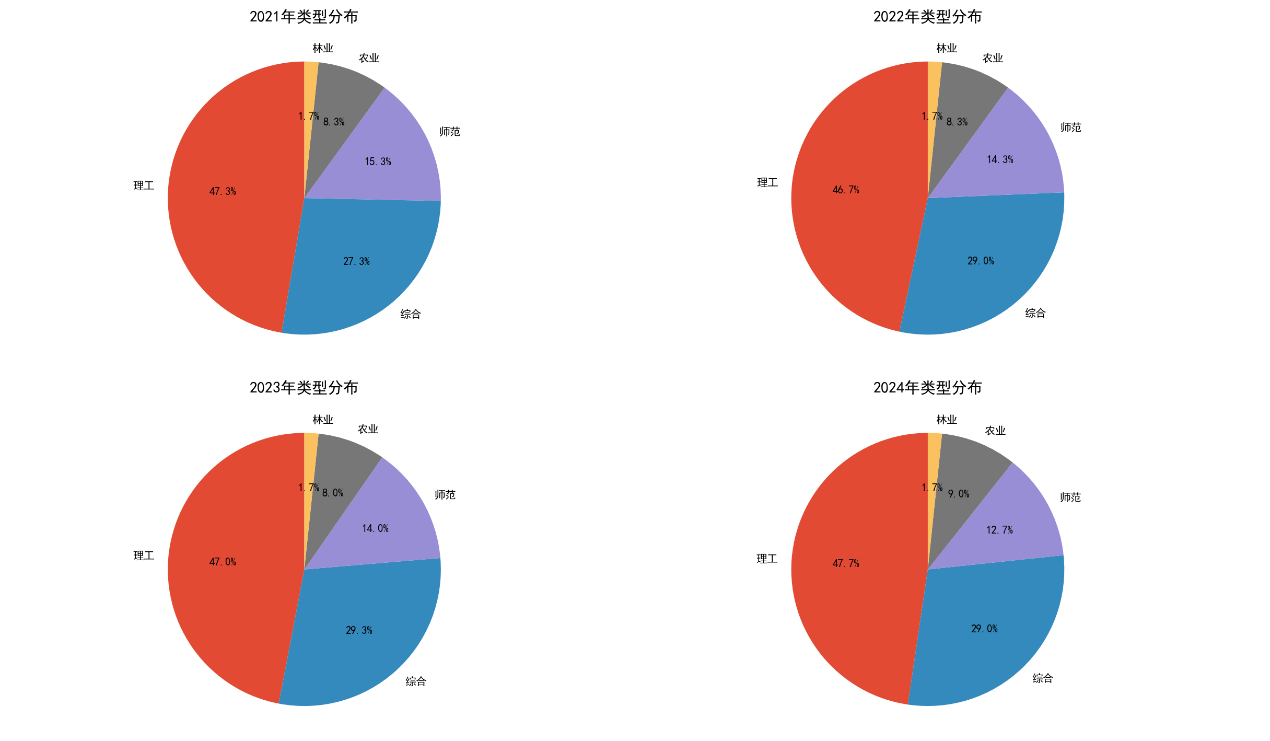


图7.4 大学类型分布图（饼图）

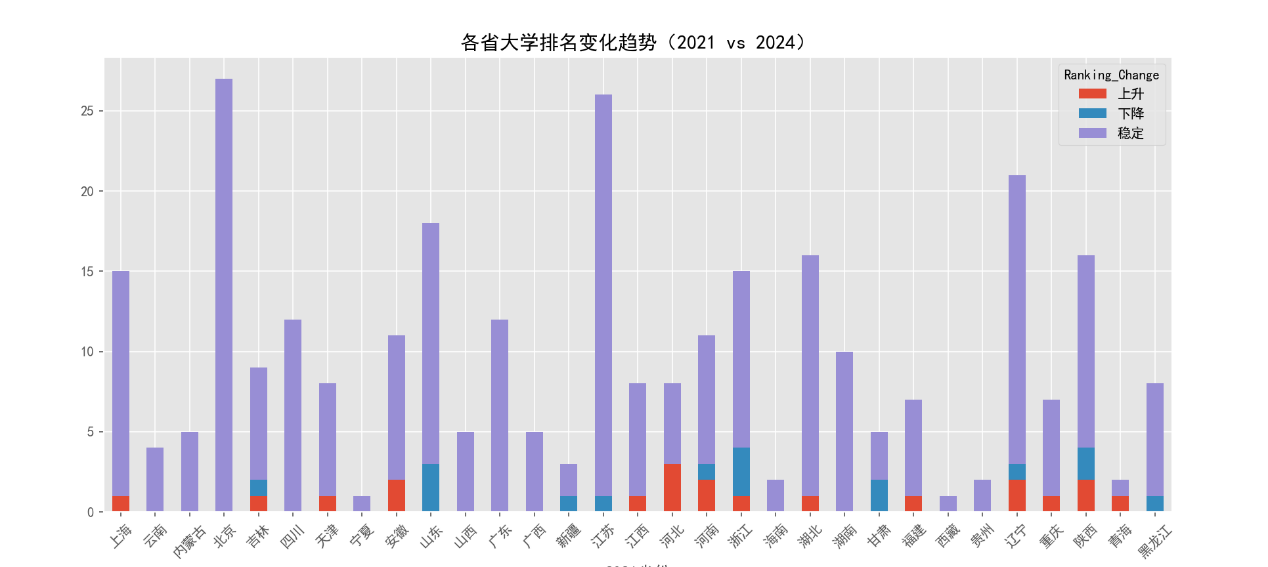
各省大学排名变化趋势图（条形图）： 这个图表的目的是展示不同省份的大学在2021年到2024年期间的排名变化。通过展示排名的变化趋势，用户可以更好地了解大学在这几年的竞争力变化，哪些大学排名上升，哪些排名下降。这可以帮助用户了解中国高等教育的整体发展趋势以及具体省份的变化。

图7.5 各省大学排名变化趋势图

相关性热力图： 相关性热力图展示了不同指标（如总分、排名等）之间的相关性，帮助用户了解各项指标之间的内在关系。例如，排名与总分之间的相关性、学校类型与排名的关系等。通过热力图，用户可以快速识别哪些因素对大学排名有较大影响，从而帮助进行更加深入的分析。。

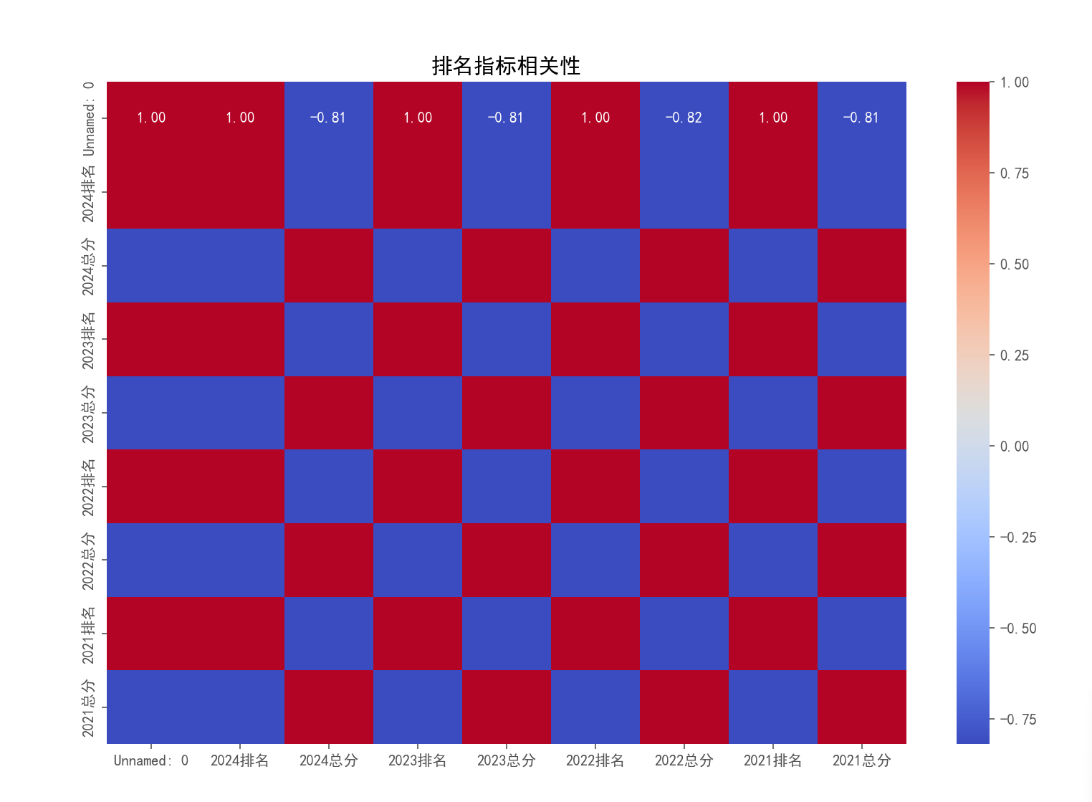


图7.6相关性热力图

在使用Django框架搭建Web应用时，检查网页的布局和样式是否符合预期，图表是否能正常嵌入到网页中。通过不断调试，确保数据分析可视化的结果能够准确、直观地展示给用户。



图7.7 数据展示界面

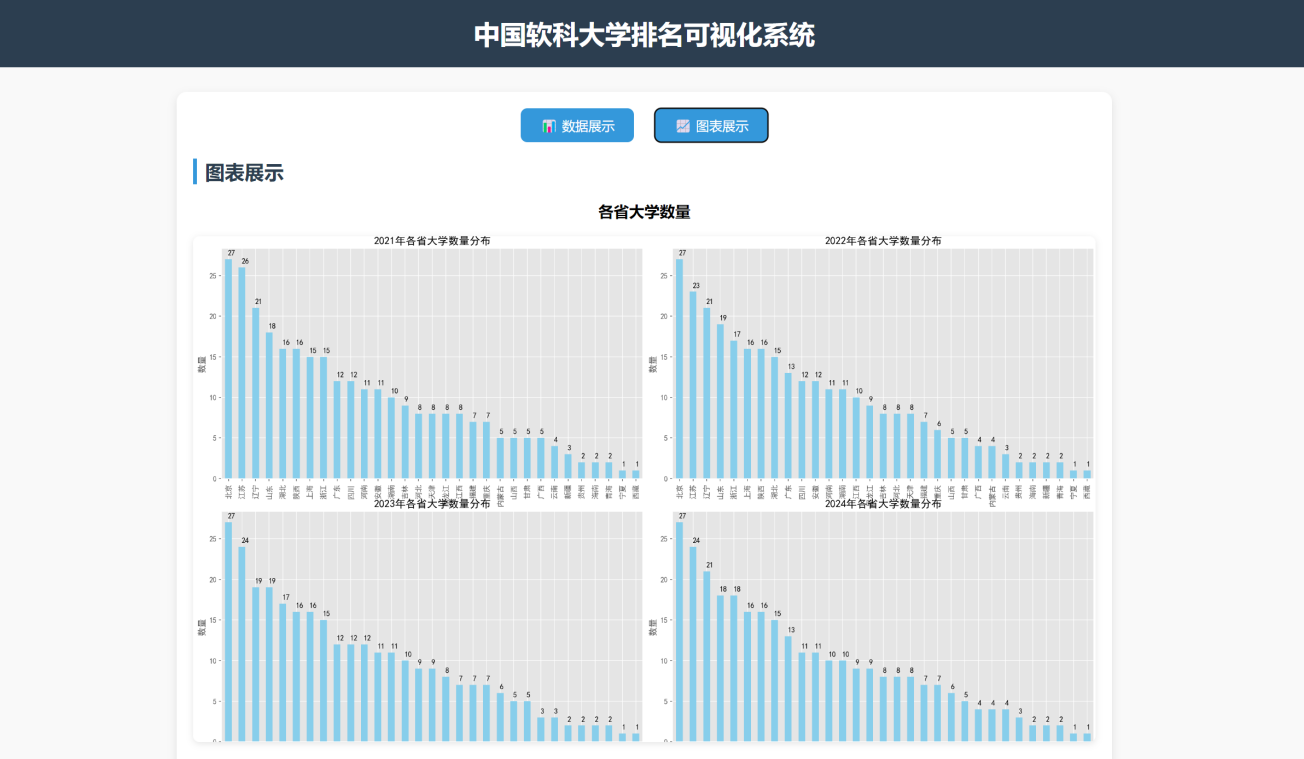


图7.8 图表展示界面

# 8 总结与体会

在完成中国软科大学排名可视化系统的过程中，我深刻体会到了数据分析和可视化在实际项目中的重要性，以及如何利用不同的技术工具和方法来处理复杂的数据。数据采集是整个项目的基础，通过编写爬虫从上海软科获取数据并进行处理，我确保了数据的结构化格式，以便后续的分析和可视化能够顺利进行。数据处理阶段，我特别注意了缺失值的处理和数据的清洗，确保所有的数据都是准确和完整的，这为后续分析提供了坚实的基础。

在数据分析和可视化的设计上，我选择了合适的图表来展示不同维度的数据。通过柱状图、饼图、条形图和热力图等方式，能够直观地展示大学的分布、排名趋势、得分变化等信息。每一个图表的设计都力求简洁明了，既要展示数据的趋势和关系，又要确保用户能够轻松地理解其中的信息。这让我更加认识到可视化设计中的挑战，既需要艺术性的布局，也需要精确的技术实现，确保信息的准确传达。

前后端的紧密配合在整个项目中起到了至关重要的作用。前端通过HTML和JavaScript展示图表和数据，后端则通过Django框架处理请求并生成图表。在这一过程中，我深刻感受到前后端之间需要紧密协作，确保数据从服务器到客户端的流畅传输和展示。这一过程不仅提升了我的编程能力，也让我学会了如何在前后端之间架构高效的数据流和交互机制。

此外，用户体验的设计也是我在开发过程中不断思考的一个方面。我注重图表的布局和交互设计，确保用户能够在图表上获得清晰的视图和相关的提示信息。同时，通过数据筛选功能，用户可以按需查看不同维度的数据，极大地提升了系统的可用性和灵活性。这使我意识到，数据可视化不仅要关注图表的美观和技术实现，还需要考虑到用户的需求，提供直观、易用的操作界面。

在项目开发的过程中，我也面临了许多挑战，尤其是在调试和优化环节。每一个环节的实现都伴随着问题的出现和解决，从数据抓取时遇到的API限制，到图表展示时遇到的显示问题，每一个细节都需要精心调整和优化。这些挑战让我不断反思和总结经验，不仅提升了我的技术能力，也让我更深刻地理解了数据可视化的价值和复杂性。

通过这个项目，我不仅学到了如何通过爬虫获取和处理大量数据，也更加深入地理解了数据可视化的重要性。数据分析和可视化并不仅仅是对数字的处理，更是通过合适的工具和手段将数据转化为易于理解的、有价值的信息。这一过程不仅帮助用户更好地理解复杂的数据，还能为决策提供重要依据，最终促使整个项目的成功完成。

# 参 考 文 献

[1] 王锋, 赵洁. 数据挖掘与分析的应用研究 [M]. 北京: 电子工业出版社, 2023.

[2] 黑马程序员. Python网络爬虫基础教程 [M]. 北京: 人民邮电出版社, 2022.

[3] 李强, 刘婷婷. 基于Python的爬虫技术及数据处理应用 [J]. 计算机工程与设计, 2022, 43(5): 1102-1109.

[4] 张亮, 陈瑞. 大数据分析与数据挖掘技术应用 [M]. 北京: 机械工业出版社, 2021.

[5] 李艳, 张华. 基于Python的文本数据分析与可视化技术研究 [J]. 数据分析与知识发现, 2023, 7(4): 45-52.

[6] 张文. 基于Python数据可视化的研究与应用 [J]. 电脑编程技巧与维护, 2023, (11): 3-5+12.

[7] 徐磊, 王静. 基于Python的网络爬虫在数据分析中的应用 [J]. 计算机与数字工程, 2022, 50(6): 112-116.

[8] 曾繁荣, 何晓阳. 数据科学中的爬虫技术与数据分析方法 [M]. 北京: 科学出版社, 2021.

[9] 刘宇航, 李涛. 数据可视化技术综述与应用 [J]. 信息与控制, 2023, 52(2): 135-142.

[10] 赵鹏飞. Python数据分析与可视化技术 [J]. 数据技术与应用, 2022, 15(9): 85-92.

[11] 蔡昉, 李鹏. 网络爬虫及其应用研究 [M]. 上海: 上海交通大学出版社, 2022.