

# Assignment 2: 使用大语言模型进行 数据分析和可视化

Assignment 2 Report

大模型应用与实践

Zhejiang University

团队成员及学号:

潘越 (3220100912)

吴海槟 (3220106042)

周楠 (3220102535)

Date: September 4<sub>th</sub>, 2024

Teacher: Zhu Linchao

# 目录

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Code Specification</b>	<b>3</b>
2.1	app.py 实现 . . . . .	3
2.1.1	如何与本地大模型交流 . . . . .	3
2.1.2	如何保存文件和展示图片 . . . . .	5
2.1.3	如何运行代码，并解决 <code>exec</code> 导入库函数失败的问题 . . . . .	5
2.2	index.html 实现 . . . . .	8
2.2.1	如何和主程序交互，传输信息 . . . . .	8
2.2.2	如何识别 AI 生成代码 . . . . .	9
2.2.3	如何自动执行代码 . . . . .	10
<b>3</b>	<b>Prompts and Outputs</b>	<b>12</b>
3.1	数据处理 . . . . .	12
3.2	数据分析 . . . . .	17
3.2.1	计算近 10 年各地区的人口均值、最大值和最小值 . . . . .	17
3.2.2	可视化近 10 年各地区的平均人口比例 . . . . .	21
3.2.3	可视化近 10 年浙江省的人口变化趋势 . . . . .	24
<b>4</b>	<b>Analysis And Comments</b>	<b>28</b>
<b>5</b>	<b>Summary</b>	<b>28</b>

# 1 Introduction

## 任务概述

1. 数据处理：设计合适的 prompt，指示 LLM 生成输出，读取文本文件，去除表头表尾，另存为一个新 CSV 文件。
2. 数据分析：读取预处理后的 CSV 文件，设计合适的 prompt，指示 LLM 生成 Python 代码，对单个 CSV 文件进行增删改查以及数据可视化。
  - 计算近 10 年各地区的人口均值、最大值和最小值，按列添加到表的最后。
  - 以饼状图的形式，可视化近 10 年各地区的平均人口比例。
  - 以折线图的形式，可视化近 10 年浙江省的人口变化趋势。
3. 仅使用本地大语言模型。

本次我们采用的模型是 `llava-llama-3-8b-v1_1-int4.gguf 4.92GB`

本地大模型部署在 dell 的 win11 笔记本上，3060 显卡，16G 显存

# 2 Code Specification

项目的文件架构如下：

```
.
├── README.md
├── app.py
├── templates
│   └── index.html
└── uploads
```

## 2.1 app.py 实现

`app.py` 实现了一个基于 Flask 的 Web 应用，主要功能包括与本地大模型交流、保存文件和展示图片，以及运行用户提交的代码。以下是对这段代码的详细介绍，重点说明了实现这些功能的方式。

### 2.1.1 如何与本地大模型交流

代码通过 OpenAI 客户端与本地大模型进行交流。以下是相关代码片段：

```
from openai import OpenAI

# 初始化 OpenAI 客户端
client = OpenAI(base_url="http://localhost:5001/v1/", api_key="sk-xxxx")
```

```

@app.route('/chat', methods=['POST'])
def chat():
    user_input = request.form.get('user_input')
    messages = request.form.getlist('messages')
    file = request.files.get('file')

    # 处理文件上传
    if file:
        file_path = os.path.join('uploads', file.filename)
        file.save(file_path)
    else:
        file_path = None

    # 添加用户消息到消息列表
    messages.append({"role": "user", "content": user_input})

    response = None
    if user_input != "":
        # 添加用户消息到消息列表
        messages.append({"role": "user", "content": user_input})

        # 调用 OpenAI API 生成响应
        completion = client.chat.completions.create(
            temperature=0,
            model="local",
            messages=messages
        )
        # 获取模型的响应
        response = completion.choices[0].message.content

    # 如果有文件上传，包含文件路径信息
    if file_path:
        if response == None:
            response = f"文件已保存到: {file_path}"
        else:
            response += f"\n文件已保存到: {file_path}"

    return jsonify({"response": response, "messages": messages})

```

在 /chat 路由中，用户输入和消息列表通过 POST 请求发送到服务器。然后，代码调用 OpenAI 客户端的 `chat.completions.create` 方法与本地大模型进行交流，并返回模型的响应。

### 2.1.2 如何保存文件和展示图片

代码通过 Flask 的文件上传功能保存文件，并通过生成 HTML 标签展示图片。以下是相关代码片段：

```
@app.route('/chat', methods=['POST'])
def chat():
    user_input = request.form.get('user_input')
    messages = request.form.getlist('messages')
    file = request.files.get('file')

    # 处理文件上传
    if file:
        file_path = os.path.join('uploads', file.filename)
        file.save(file_path)
    else:
        file_path = None

    # ... 省略部分代码 ...

@app.route('/uploads/<filename>')
def uploaded_file(filename):
    return send_from_directory('uploads', filename)
```

在 /chat 路由中，代码检查是否有文件上传，如果有，则将文件保存到 uploads 目录。通过 /uploads/<filename> 路由，可以访问和下载这些文件。

### 2.1.3 如何运行代码，并解决 exec 导入库函数失败的问题

代码通过 exec 函数运行用户提交的代码，并动态导入所需的库。以下是相关代码片段：

```
@app.route('/execute', methods=['POST'])
def execute():
    data = request.json
    code = data.get('code')

    try:
        # 打印接收到的代码
        print("Received code:", code)

        # 切换到 uploads 目录并执行代码
        os.chdir('uploads')
        exec_globals = {}
        exec_locals = {}
```

```

# 捕获标准输出
stdout = io.StringIO()
sys.stdout = stdout

# 在执行代码之前导入所有在代码中出现的库
import_statements = re.findall(r'^\s*import\s+(\S+)(?:\s+as\s+
    +(\S+))?', code, re.MULTILINE)
from_import_statements = re.findall(r'^\s*from\s+(\S+)\s+
    import\s+(\S+)(?:\s+as\s+(\S+))?', code, re.MULTILINE)

for module, alias in import_statements:
    if alias:
        exec(f"import {module} as {alias}", exec_globals,
            exec_locals)
        exec_globals[alias] = exec_locals[alias]
    else:
        exec(f"import {module}", exec_globals, exec_locals)
        exec_globals[module] = exec_locals[module]

for module, submodule, alias in from_import_statements:
    if alias:
        exec(f"from {module} import {submodule} as {alias}",
            exec_globals, exec_locals)
        exec_globals[alias] = exec_locals[alias]
    else:
        exec(f"from {module} import {submodule}", exec_globals,
            exec_locals)
        exec_globals[submodule] = exec_locals[submodule]

# 记录现有的文件列表
existing_files = set(os.listdir('.'))

exec(code, exec_globals, exec_locals)

# 恢复标准输出
sys.stdout = sys.__stdout__

# 获取执行结果
result = stdout.getvalue()
print("Result:", result)

```

```

# 生成图像
if 'plt.show()' in code:
    plt.gcf().canvas.draw()
    # 生成唯一的文件名
    timestamp = time.strftime("%Y%m%d-%H%M%S")
    random_str = ''.join(random.choices(string.ascii_lowercase
        + string.digits, k=6))
    filename = f'output_{timestamp}_{random_str}.png'
    plt.savefig(filename)
    plt.close()
    if os.path.exists(filename):
        result += f"\n<img src='/uploads/{filename}' />"

# 检查是否生成了新文件
new_files = set(os.listdir('.')) - existing_files
for file in new_files:
    if os.path.isfile(file):
        result += f"\n<a href='/uploads/{file}' download class
            ='download-link'>{file}</a>"

os.chdir('..')
return jsonify({"success": True, "result": result})
except Exception as e:
    return jsonify({"success": False, "error": str(e)})

```

在 `/execute` 路由中，代码接收用户提交的代码，并使用 `exec` 函数执行。为了确保导入的库能够在 `exec` 中正确使用，代码首先通过正则表达式提取所有 `import` 和 `from ... import ...` 语句，并动态导入这些库。然后，代码记录现有的文件列表，执行用户代码，并检查是否生成了新文件。如果生成了新文件，则将其添加到结果中。

通过这些步骤，代码解决了 `exec` 导入库函数失败的问题，并确保所有导入的库在执行代码时都可用。

## 代码结尾部分

以下是代码的结尾部分：

```

os.chdir('..')
return jsonify({"success": True, "result": result})
except Exception as e:
    return jsonify({"success": False, "error": str(e)})

@app.route('/uploads/<filename>')
def uploaded_file(filename):

```

```

    return send_from_directory('uploads', filename)

if __name__ == '__main__':
    if not os.path.exists('uploads'):
        os.makedirs('uploads')
    app.run(debug=True)

```

## 2.2 index.html 实现

这个 HTML 文件实现了一个与 AI 进行聊天的网页界面，用户可以在此界面输入消息、上传文件，并接收 AI 的响应。以下是对这个 HTML 文件的详细介绍，重点解释如何与主程序交互、识别 AI 生成的代码以及自动执行代码。

### 2.2.1 如何和主程序交互，传输信息

网页通过 JavaScript 的 fetch API 与主程序进行交互，传输用户输入的信息和文件。以下是相关代码片段：

```

<script>
    let messages = [];

    function sendMessage() {
        const userInput = document.getElementById('user-input').value;
        const fileInput = document.getElementById('file-input').files[0];
        if (!userInput && !fileInput) return;

        const chatBox = document.getElementById('chat-box');
        const userMessage = marked.parse(userInput);
        chatBox.innerHTML += `<div class="message user" data-text="${
            userInput}"><strong>YOU:</strong> ${userMessage} <button
            class="copy-btn" onclick="copyText(this)">Copy</button></div>`;

        chatBox.scrollTop = chatBox.scrollHeight; // 滚动到底部
        document.getElementById('user-input').value = '';
        document.getElementById('file-input').value = '';

        const formData = new FormData();
        formData.append('user_input', userInput);
        if (fileInput) {
            formData.append('file', fileInput);
        }
    }

```



```

    fetch('/chat', {
      method: 'POST',
      body: formData
    })

    .then(response => response.json())
    .then(data => {
      const aiMessage = marked.parse(data.response);
      const aiMessageElement = document.createElement('div')
      ;
      aiMessageElement.className = 'message assistant';
      aiMessageElement.setAttribute('data-text', data.
        response);
      aiMessageElement.innerHTML = `AI:</strong> ${
        aiMessage} <button class="copy-btn" onclick="
        copyText(this)">Copy</button>`;

      chatBox.appendChild(aiMessageElement);
      chatBox.scrollTop = chatBox.scrollHeight; // 滚动到底
        部
      messages = data.messages;

      // 自动执行代码
      const codeBlocks = extractCodeBlocks(data.response);
      if (codeBlocks.length > 0) {
        const combinedCode = codeBlocks.join('\n');
        executeCode(combinedCode, aiMessageElement);
      }
    });
  }
</script>

```

在 `sendMessage` 函数中，用户输入的消息和上传的文件通过 `FormData` 对象打包，并通过 `fetch` API 发送到 `/chat` 路由。服务器处理请求并返回响应，网页解析响应并更新聊天框。

### 2.2.2 如何识别 AI 生成代码

网页通过正则表达式识别 AI 生成的代码块。以下是相关代码片段：

```

<script>
function extractCodeBlocks(response) {
  const codeBlockPattern = /```(?:python)?\n([\s\S]*?)```/g;
  const codeBlocks = [];
  let match;

```

```

        while ((match = codeBlockPattern.exec(response)) !== null) {
            codeBlocks.push(match[1]);
        }

        return codeBlocks;
    }
</script>

```

`extractCodeBlocks` 函数使用正则表达式 `/`(?:python)?([$]*?)`/g` 匹配响应中的代码块，并将其提取到 `codeBlocks` 数组中。

### 2.2.3 如何自动执行代码

网页通过 `fetch` API 将提取到的代码块发送到 `/execute` 路由，并在服务器执行代码后更新聊天框。以下是相关代码片段：

```
<script>
function executeCode(codeBlock, aiMessageElement) {
  fetch('/execute', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ code: codeBlock })
  })
    .then(response => response.json())
    .then(data => {
      if (data.success) {
        const resultMessage = `${data.result}\n`;
        const renderedMessage = marked.parse(resultMessage);
        aiMessageElement.innerHTML = aiMessageElement.
          innerHTML.replace(codeBlock, renderedMessage);
      } else {
        const errorMessage = `\\`\\`\\`\\`\\nFailed to execute
          code: ${data.error}\\n\\`\\`\\`\\`;
        const renderedMessage = marked.parse(errorMessage);
        ;
        aiMessageElement.innerHTML = aiMessageElement.
          innerHTML.replace(codeBlock, renderedMessage);
      }
      const chatBox = document.getElementById('chat-box');
      chatBox.scrollTop = chatBox.scrollHeight; // 滚动到底部
    })
    .catch(err => {

```

```
        console.error('Failed to execute code: ', err);
    });
}
</script>
```

`executeCode` 函数将代码块发送到 `/execute` 路由，并处理服务器返回的响应。如果代码执行成功，网页将执行结果插入到 AI 消息元素中；如果执行失败，网页将错误信息插入到 AI 消息元素中。

通过这些步骤，网页实现了与主程序的交互、识别 AI 生成的代码以及自动执行代码的功能。最终的网站页面为：

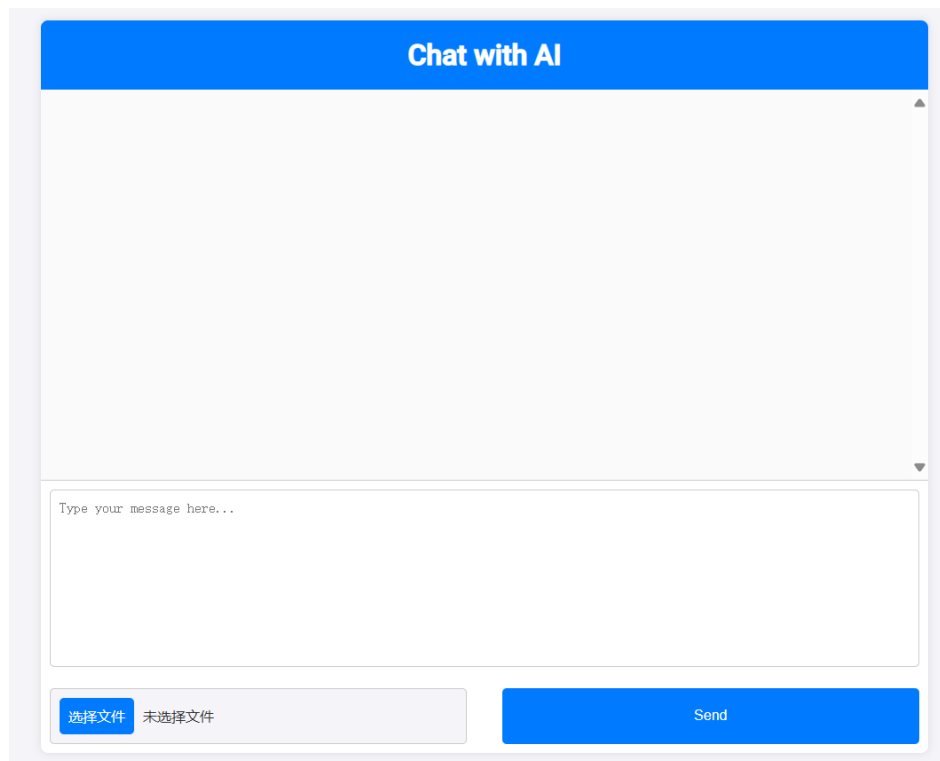


图 1: 网站页面

## 3 Prompts and Outputs

### 3.1 数据处理

**Prompt 内容如下:**

we have a .txt file named '年末常住人口.txt'

数据库: 分省年度数据

指标: 年末常住人口(万人)

时间: 最近10年

地区, 2023年, 2022年, 2021年, 2020年, 2019年, 2018年, 2017年, 2016年, 2015年, 2014年

北京市, 2186, 2184, 2189, 2189, 2190, 2192, 2194, 2195, 2188, 2171

天津市, 1364, 1363, 1373, 1387, 1385, 1383, 1410, 1443, 1439, 1429

河北省, 7393, 7420, 7448, 7464, 7447, 7426, 7409, 7375, 7345, 7323

山西省, 3466, 3481, 3480, 3490, 3497, 3502, 3510, 3514, 3519, 3528

内蒙古自治区, 2396, 2401, 2400, 2403, 2415, 2422, 2433, 2436, 2440, 2449

辽宁省, 4182, 4197, 4229, 4255, 4277, 4291, 4312, 4327, 4338, 4358

吉林省, 2339, 2348, 2375, 2399, 2448, 2484, 2526, 2567, 2613, 2642

黑龙江省, 3062, 3099, 3125, 3171, 3255, 3327, 3399, 3463, 3529, 3608

上海市, 2487, 2475, 2489, 2488, 2481, 2475, 2466, 2467, 2458, 2467

江苏省, 8526, 8515, 8505, 8477, 8469, 8446, 8423, 8381, 8315, 8281

浙江省, 6627, 6577, 6540, 6468, 6375, 6273, 6170, 6072, 5985, 5890

安徽省, 6121, 6127, 6113, 6105, 6092, 6076, 6057, 6033, 6011, 5997

福建省, 4183, 4188, 4187, 4161, 4137, 4104, 4065, 4016, 3984, 3945

江西省, 4515, 4528, 4517, 4519, 4516, 4513, 4511, 4496, 4485, 4480

山东省, 10123, 10163, 10170, 10165, 10106, 10077, 10033, 9973, 9866, 9808

河南省, 9815, 9872, 9883, 9941, 9901, 9864, 9829, 9778, 9701, 9645

湖北省, 5838, 5844, 5830, 5745, 5927, 5917, 5904, 5885, 5850, 5816

湖南省, 6568, 6604, 6622, 6645, 6640, 6635, 6633, 6625, 6615, 6611

广东省, 12706, 12657, 12684, 12624, 12489, 12348, 12141, 11908, 11678, 11489

广西壮族自治区, 5027, 5047, 5037, 5019, 4982, 4947, 4907, 4857, 4811, 4770

海南省, 1043, 1027, 1020, 1012, 995, 982, 972, 957, 945, 936

重庆市, 3191, 3213, 3212, 3209, 3188, 3163, 3144, 3110, 3070, 3043

四川省, 8368, 8374, 8372, 8371, 8351, 8321, 8289, 8251, 8196, 8139

贵州省, 3865, 3856, 3852, 3858, 3848, 3822, 3803, 3758, 3708, 3677

云南省, 4673, 4693, 4690, 4722, 4714, 4703, 4693, 4677, 4663, 4653

西藏自治区, 365, 364, 366, 366, 361, 354, 349, 340, 330, 325

陕西省, 3952, 3956, 3954, 3955, 3944, 3931, 3904, 3874, 3846, 3827

甘肃省, 2465, 2492, 2490, 2501, 2509, 2515, 2522, 2520, 2523, 2531

青海省, 594, 595, 594, 593, 590, 587, 586, 582, 577, 576

宁夏回族自治区, 729, 728, 725, 721, 717, 710, 705, 695, 684, 678

新疆维吾尔自治区, 2598, 2587, 2589, 2590, 2559, 2520, 2480, 2428, 2385, 2325

注：常住人口包括：居住在本乡镇街道且户口在本乡镇街道或户口待定的人；居住在本乡镇街道且离开户口登记地所在的乡镇街道半年以上的人；户口在本乡镇街道且外出不满半年或在境外工作学习的人。

数据来源：国家统计局

#### 1. Read the text file:

- Use the `open()` function in read mode ( `'r'`) to open the 年末常住人口.txt file.
- Use `file.readlines()` to read all lines from the file and store them in a list called `lines`.
- Ensure the file is read using `encoding='utf-8'` to handle any potential Chinese characters

#### 2. Remove the header and footer:

- The first 3 lines of the file contain metadata (e.g., database, indicator, and time), which should be removed from the dataset.
- The last 2 lines usually contain footnotes and should also be removed.
- Use list slicing, i.e., `lines[3:-2]`, to remove these unnecessary lines and retain the valid data in the middle.

#### 3. Save the cleaned data as a CSV file:

- Use the `open()` function in write mode ( `'w'`) to open a new file (e.g., 年末常住人口.csv) to save the processed data.
- **To avoid garbled text for Chinese characters, specify `encoding='utf-8-sig'` when writing the file.**
- **To avoid extra blank lines in the CSV file, specify the `newline=''` parameter when opening the file.** This ensures that no additional line breaks are added between rows when writing.
- Use `csv.writer()` to create a CSV writer object named `csvwriter`.
- Iterate through the cleaned `lines` list line by line:
  - Use the `strip()` function to remove any trailing newline characters.
  - Use `split(',')` to split the data by commas, converting each line into a list format.
  - Call the `csvwriter.writerow(row)` method to write each line of data into the CSV file.

#### 4. Main function:

- Wrap the above steps into a main function named `convert\_txt\_to\_csv`, which accepts two parameters: the input file path and the output file path.
- In the main function, sequentially call the helper functions to achieve the conversion from .txt to .csv.

## 5. Call the function to perform the file conversion:

- Call 'convert\_txt\_to\_csv('年末常住人口.txt', '年末常住人口.csv')' to execute the file conversion.

得到如下代码:

```
import os

def convert_txt_to_csv(input_file_path, output_file_path):
    # Read the input file
    with open(input_file_path, 'r', encoding='utf-8') as f:
        lines = f.readlines()

    # Remove the header and footer
    lines = lines[3:-2]

    # Write the cleaned data to the output file
    with open(output_file_path, 'w', encoding='utf-8-sig', newline='')
        as f:
            for line in lines:
                f.write(line)

if __name__ == "__main__":
    input_file_path = '年末常住人口.txt'
    output_file_path = '年末常住人口.csv'

    convert_txt_to_csv(input_file_path, output_file_path)
```

运行结果如下:

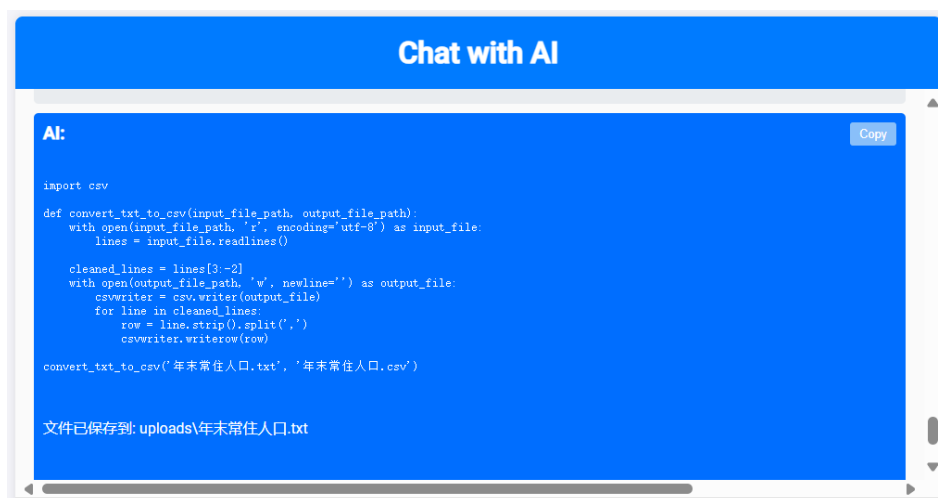


图 2: prompt

得到代码并自动执行:

```
import csv

def convert_txt_to_csv(input_file_path, output_file_path):
    with open(input_file_path, 'r', encoding='utf-8') as input_file:
        lines = input_file.readlines()

    cleaned_lines = lines[3:-2]
    with open(output_file_path, 'w', newline='') as output_file:
        csvwriter = csv.writer(output_file)
        for line in cleaned_lines:
            row = line.strip().split(',')
            csvwriter.writerow(row)

convert_txt_to_csv('年末常住人口.txt', '年末常住人口.csv')
```

生成的年末常住人口.csv 自动存储在 upload 文件夹中  
打开 csv 表格后结果如下：

年末常住人口										
地区	2023年	2022年	2021年	2020年	2019年	2018年	2017年	2016年	2015年	2014年
北京市	2186	2184	2189	2189	2190	2192	2194	2195	2188	2171
天津市	1364	1363	1373	1387	1385	1383	1410	1443	1439	1429
河北省	7393	7420	7448	7464	7447	7426	7409	7375	7345	7323
山西省	3466	3481	3480	3490	3497	3502	3510	3514	3519	3528
内蒙古自治区	2396	2401	2400	2403	2415	2422	2433	2436	2440	2449
辽宁省	4182	4197	4229	4255	4277	4291	4312	4327	4338	4358
吉林省	2339	2348	2375	2399	2448	2484	2526	2567	2613	2642
黑龙江省	3062	3099	3125	3171	3255	3327	3399	3463	3529	3608
上海市	2487	2475	2489	2488	2481	2475	2466	2467	2458	2467
江苏省	8526	8515	8505	8477	8469	8446	8423	8381	8315	8281
浙江省	6627	6577	6540	6468	6375	6273	6170	6072	5985	5890
安徽省	6121	6127	6113	6105	6092	6076	6057	6033	6011	5997
福建省	4183	4188	4187	4161	4137	4104	4065	4016	3984	3945
江西省	4515	4528	4517	4519	4516	4513	4511	4496	4485	4480
山东省	10123	10163	10170	10165	10106	10077	10033	9973	9866	9808
河南省	9815	9872	9883	9941	9901	9864	9829	9778	9701	9645
湖北省	5838	5844	5830	5745	5927	5917	5904	5885	5850	5816
湖南省	6568	6604	6622	6645	6640	6635	6633	6625	6615	6611
广东省	12706	12657	12684	12624	12489	12348	12141	11908	11678	11489
广西壮族自治区	5027	5047	5037	5019	4982	4947	4907	4857	4811	4770
海南省	1043	1027	1020	1012	995	982	972	957	945	936
重庆市	3191	3213	3212	3209	3188	3163	3144	3110	3070	3043
四川省	8368	8374	8372	8371	8351	8321	8289	8251	8196	8139
贵州省	3865	3856	3852	3858	3848	3822	3803	3758	3708	3677
云南省	4673	4693	4690	4722	4714	4703	4693	4677	4663	4653
西藏自治区	365	364	366	366	361	354	349	340	330	325
陕西省	3952	3956	3954	3955	3944	3931	3904	3874	3846	3827
甘肃省	2465	2492	2490	2501	2509	2515	2522	2520	2523	2531
青海省	594	595	594	593	590	587	586	582	577	576
宁夏回族自治区	729	728	725	721	717	710	705	695	684	678
新疆维吾尔自治区	2598	2587	2589	2590	2559	2520	2480	2428	2385	2325

图 3: 年末常住人口.csv



## 3.2 数据分析

### 3.2.1 计算近 10 年各地区的人口均值、最大值和最小值

**prompt 内容如下：**

You have a CSV file named 年末常住人口.csv containing population data for various regions over the last 10 years. The first column represents the region names, and the next 10 columns represent the population for each year.

The goal is to write a Python script that reads this CSV file, calculates the mean, maximum, and minimum population for each region over the past 10 years, and then saves these results to a new CSV file.

The steps to achieve this are as follows:

1. Import the pandas Library:
2. Define a Function calculate\_population\_statistics:

- **Step 1: Read the CSV File:**
- **Step 2: Identify Relevant Columns:**

Extract the column names corresponding to the last 10 years of data. This can be done by slicing the columns of the DataFrame using `df.columns[-10:]` and storing them in a variable named `last_10_years`.

- **Step 3: Calculate Population Statistics:**

Use pandas vectorized operations to calculate the following statistics for each region over the last 10 years: Mean population using `df[last_10_years].mean(axis=1)` and assign the result to a new column named 均值. Maximum population using `df[last_10_years].max(axis=1)` and assign the result to a new column named 最大值. Minimum population using `df[last_10_years].min(axis=1)` and assign the result to a new column named 最小值.

- **Step 4: Save the Results to a New CSV File:**

After calculating the statistics, use `df.to_csv(output_csv, index=False)` to save the updated DataFrame to the output CSV file. The `index=False` argument ensures that the row indices are not included in the CSV file.

3. Example Call:

Finally, call the `calculate_population_statistics` function, passing the input CSV file 年末常住人口.csv and specifying `calculate_population_with_statistics.csv` as the output file. This will generate a new CSV file with the original data and the calculated statistics for each region.

By following these steps and function calls, you will create a script that processes population data and generates a CSV file with additional statistical columns for each region.

网站显示输出结果如下：

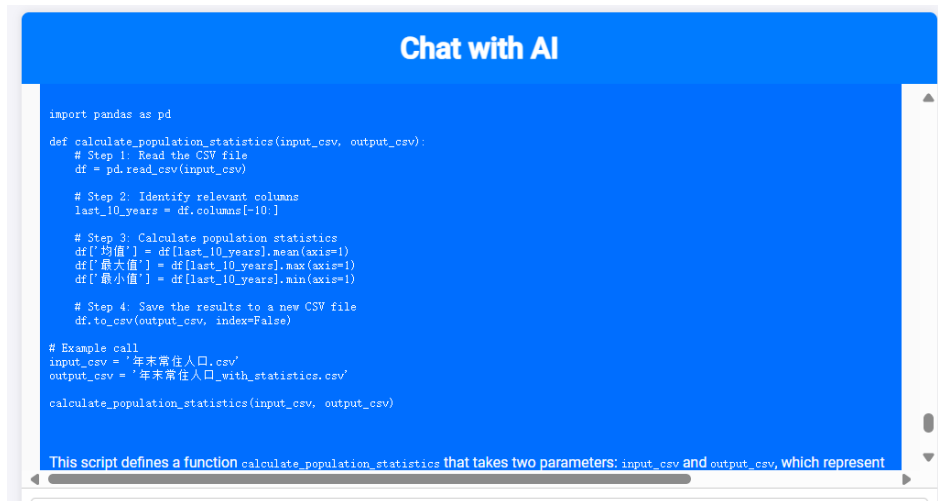


图 4: prompt

对应的终端输出为：

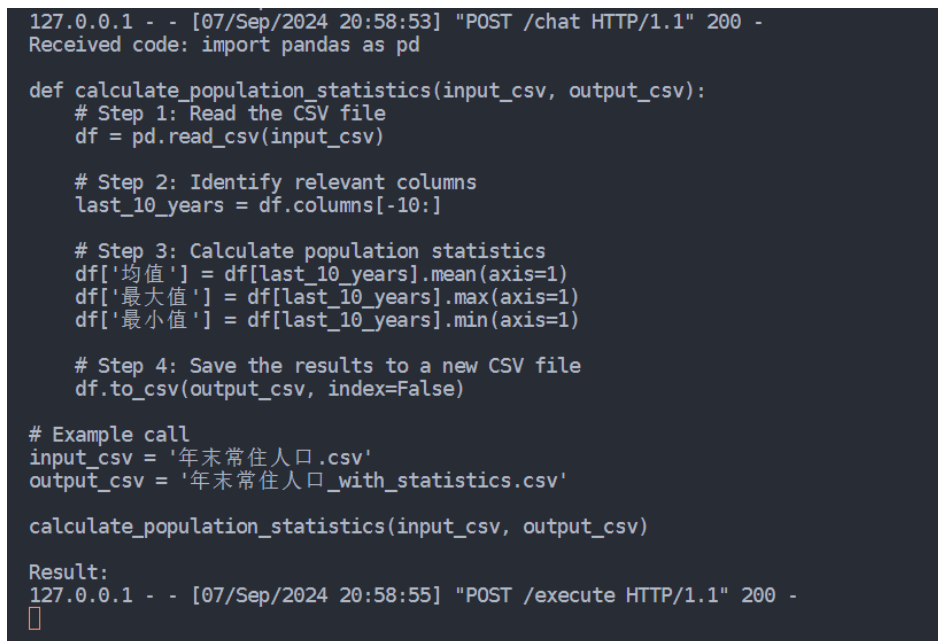


图 5: result

得到如下代码:

```
import pandas as pd

def calculate_population_statistics(input_csv, output_csv):
    # Step 1: Read the CSV file
    df = pd.read_csv(input_csv)

    # Step 2: Identify relevant columns
    last_10_years = df.columns[-10:]

    # Step 3: Calculate population statistics
    df['均值'] = df[last_10_years].mean(axis=1)
    df['最大值'] = df[last_10_years].max(axis=1)
    df['最小值'] = df[last_10_years].min(axis=1)

    # Step 4: Save the results to a new CSV file
    df.to_csv(output_csv, index=False)

# Example call
input_csv = '年末常住人口.csv'
output_csv = '年末常住人口_with_statistics.csv'

calculate_population_statistics(input_csv, output_csv)
```

自动执行后，得到输出结果‘年末常住人口 \_with\_statistics.csv’，其自动保存在 upload 文件夹中

年末常住人口_with_statistics													
地区	2023年	2022年	2021年	2020年	2019年	2018年	2017年	2016年	2015年	2014年	均值	最大值	最小值
北京市	2186	2184	2189	2189	2190	2192	2194	2195	2188	2171	2187.8	2195	2171
天津市	1364	1363	1373	1387	1385	1383	1410	1443	1439	1429	1397.6	1443	1363
河北省	7393	7420	7448	7464	7447	7426	7409	7375	7345	7323	7405.0	7464	7323
山西省	3466	3481	3480	3490	3497	3502	3510	3514	3519	3528	3498.7	3528	3466
内蒙古自治区	2396	2401	2400	2403	2415	2422	2433	2436	2440	2449	2419.5	2449	2396
辽宁省	4182	4197	4229	4255	4277	4291	4312	4327	4338	4358	4276.6	4358	4182
吉林省	2339	2348	2375	2399	2448	2484	2526	2567	2613	2642	2474.1	2642	2339
黑龙江省	3062	3099	3125	3171	3255	3327	3399	3463	3529	3608	3303.8	3608	3062
上海市	2487	2475	2489	2488	2481	2475	2466	2467	2458	2467	2475.3	2489	2458
江苏省	8526	8515	8505	8477	8469	8446	8423	8381	8315	8281	8433.8	8526	8281
浙江省	6627	6577	6540	6468	6375	6273	6170	6072	5985	5890	6297.7	6627	5890
安徽省	6121	6127	6113	6105	6092	6076	6057	6033	6011	5997	6073.2	6127	5997
福建省	4183	4188	4187	4161	4137	4104	4065	4016	3984	3945	4097.0	4188	3945
江西省	4515	4528	4517	4519	4516	4513	4511	4496	4485	4480	4508.0	4528	4480
山东省	10123	10163	10170	10165	10106	10077	10033	9973	9866	9808	10048.4	10170	9808
河南省	9815	9872	9883	9941	9901	9864	9829	9778	9701	9645	9822.9	9941	9645
湖北省	5838	5844	5830	5745	5927	5917	5904	5885	5850	5816	5855.6	5927	5745
湖南省	6568	6604	6622	6645	6640	6635	6633	6625	6615	6611	6619.8	6645	6568
广东省	12706	12657	12684	12624	12489	12348	12141	11908	11678	11489	12272.4	12706	11489
广西壮族自治区	5027	5047	5037	5019	4982	4947	4907	4857	4811	4770	4940.4	5047	4770
海南省	1043	1027	1020	1012	995	982	972	957	945	936	988.9	1043	936
重庆市	3191	3213	3212	3209	3188	3163	3144	3110	3070	3043	3154.3	3213	3043
四川省	8368	8374	8372	8371	8351	8321	8289	8251	8196	8139	8303.2	8374	8139
贵州省	3865	3856	3852	3858	3848	3822	3803	3758	3708	3677	3804.7	3865	3677
云南省	4673	4693	4690	4722	4714	4703	4693	4677	4663	4653	4688.1	4722	4653
西藏自治区	365	364	366	366	361	354	349	340	330	325	352.0	366	325
陕西省	3952	3956	3954	3955	3944	3931	3904	3874	3846	3827	3914.3	3956	3827
甘肃省	2465	2492	2490	2501	2509	2515	2522	2520	2523	2531	2506.8	2531	2465
青海省	594	595	594	593	590	587	586	582	577	576	587.4	595	576
宁夏回族自治区	729	728	725	721	717	710	705	695	684	678	709.2	729	678
新疆维吾尔自治区	2598	2587	2589	2590	2559	2520	2480	2428	2385	2325	2506.1	2598	2325

图 6: 年末常住人口 \_with\_statistics.csv

### 3.2.2 可视化近 10 年各地区的平均人口比例

prompt 如下:

You have a CSV file named 年末常住人口\_with\_statistics.csv containing data on the population of various regions over the past 10 years. The goal is to visualize the average population proportion of each region over these years using a pie chart.

- **Step 1 - Set Up the Environment for Chinese Text Display:**

Configure Matplotlib to support Chinese characters by setting the font to "SimHei". Additionally, ensure that the text.usetex parameter is set to False to avoid issues with LaTeX rendering.

Functions: plt.rcParams['font.family'], plt.rcParams['text.usetex']

- **Step 2 - Load and Read Data:**

- **Step 3 - Select Relevant Columns:**

From the DataFrame, extract unique region names using df['地区'].unique() and calculate the sum of the average population for each region using df.groupby('地区')['均值'].sum().

- **Step 4 - Create a Pie Chart, Increase Figure Size, and Set Decimal Precision:**

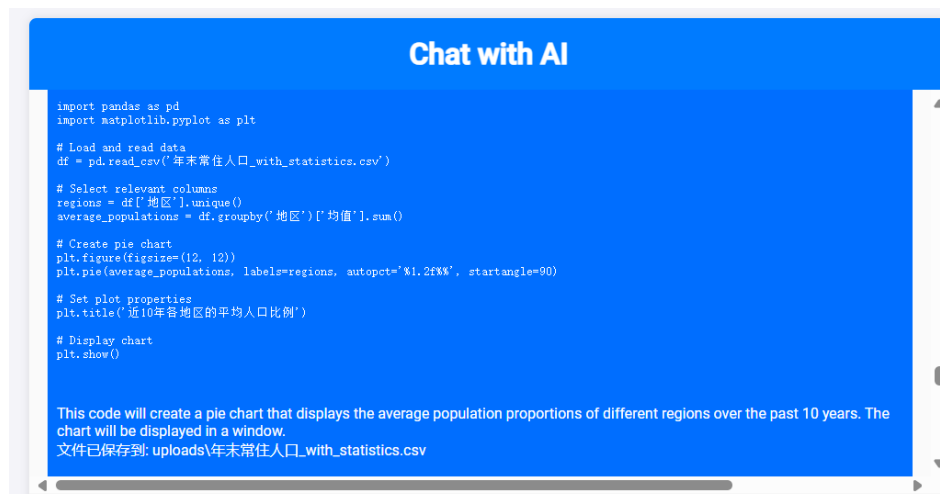
Use the plt.figure() function to increase the overall figure size (e.g., to 12x12 inches) before creating the pie chart with plt.pie. Pass the average population data as the sizes of the slices and the regions as the labels. Ensure that the percentage values displayed on the pie chart are formatted to two decimal places by using autopct=

Functions: plt.figure(figsize=(width, height)), plt.pie(autopct=

- **Step 5 - Set Plot Properties:**

- **Step 6 - Display the Chart:**

网站输出结果如下:



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

# Load and read data
df = pd.read_csv('年末常住人口_with_statistics.csv')

# Select relevant columns
regions = df['地区'].unique()
average_populations = df.groupby('地区')['均值'].sum()

# Create pie chart
plt.figure(figsize=(12, 12))
plt.pie(average_populations, labels=regions, autopct='%1.2f%%', startangle=90)

# Set plot properties
plt.title('近10年各地区的平均人口比例')

# Display chart
plt.show()
```

This code will create a pie chart that displays the average population proportions of different regions over the past 10 years. The chart will be displayed in a window.

文件已保存到: uploads\年末常住人口\_with\_statistics.csv

图 7: prompt

得到代码如下:

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

# Load and read data
df = pd.read_csv('年末常住人口_with_statistics.csv')

# Select relevant columns
regions = df['地区'].unique()
average_populations = df.groupby('地区')['均值'].sum()

# Create pie chart
plt.figure(figsize=(12, 12))
plt.pie(average_populations, labels=regions, autopct='%1.2f%%',
        startangle=90)

# Set plot properties
plt.title('近10年各地区的平均人口比例')

# Display chart
plt.show()
```

代码自动运行后, 得到如下结果

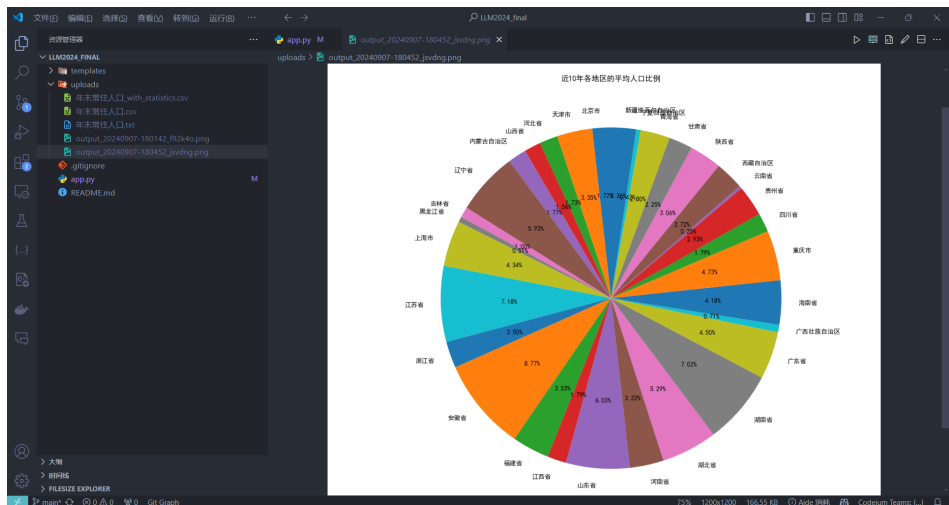


图 8: result

得到饼状图

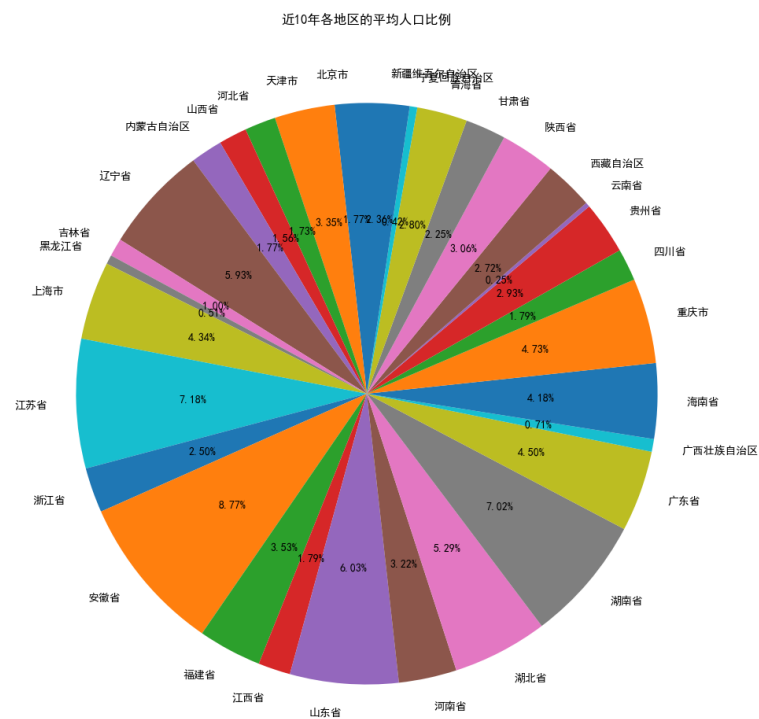


图 9: 饼状图

### 3.2.3 可视化近 10 年浙江省的人口变化趋势

**Prompt 内容如下:**

Generate Python code that reads population data from a CSV file named '年末常住人口.csv'. The code should focus on visualizing the population trend in Zhejiang Province over the past 10 years using a line plot.

The code should follow these steps:

1. **Import the necessary libraries:**

2. **Set up the environment for Chinese text display:**

- Use `plt.rcParams['font.family'] = 'SimHei'` to set the font family to SimHei, enabling the display of Chinese characters.
- Use `plt.rcParams['text.usetex'] = False` to disable LaTeX rendering.

3. **Load the data from a CSV file:**

4. **Filter data for a specific region (Zhejiang Province):**

- Use `data[data['地区'] == '浙江省']` to filter the DataFrame and keep only the rows where the 地区 column equals '浙江省'.

5. **Define the years and extract the corresponding population data:**

- Create a list of years using `years = ['2014年', '2015年', ..., '2023年']`.
- Use `zhejiang_data[years].values.flatten()` to extract the population data for these years from the filtered DataFrame and flatten it into a one-dimensional array.

6. **Plot the data:**

- Use `plt.figure(figsize=(width, height))` to initialize a new figure with the specified size.
- Use `plt.plot(x, y, marker='o', linestyle='-', color='b')` to plot the data with years on the x-axis and population on the y-axis.
- Use `plt.title('title')` to set the title of the plot.
- Use `plt.grid(True)` to add a grid to the plot.
- Use `plt.xticks(rotation=angle)` to rotate the x-axis labels for better readability.

7. **Display the plot:**

- Use `plt.tight_layout()` to adjust the layout to accommodate the rotated labels.

网站输出结果如下:



### Chat with AI

Zhejiang Province over the past 10 years using a line plot:

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

# Set up the environment for Chinese text display
plt.rcParams[ 'font.family' ] = 'SimHei'
plt.rcParams[ 'text.usetex' ] = False

# Load the data from a CSV file
data = pd.read_csv('年末常住人口.csv')

# Filter data for a specific region (Zhejiang Province)
zhejiang_data = data[data['地区'] == '浙江省']

# Define the years and extract the corresponding population data
years = ['2014年', '2015年', '2016年', '2017年', '2018年', '2019年', '2020年', '2021年', '2022年', '2023年']
zhejiang_data = zhejiang_data[years].values.flatten()

# Plot the data
plt.figure(figsize=(10, 6))
plt.plot(years, zhejiang_data, marker='o', linestyle='-', color='b')
plt.title('浙江省人口趋势')
plt.xlabel('年份')
plt.ylabel('人口')
```

图 10: code

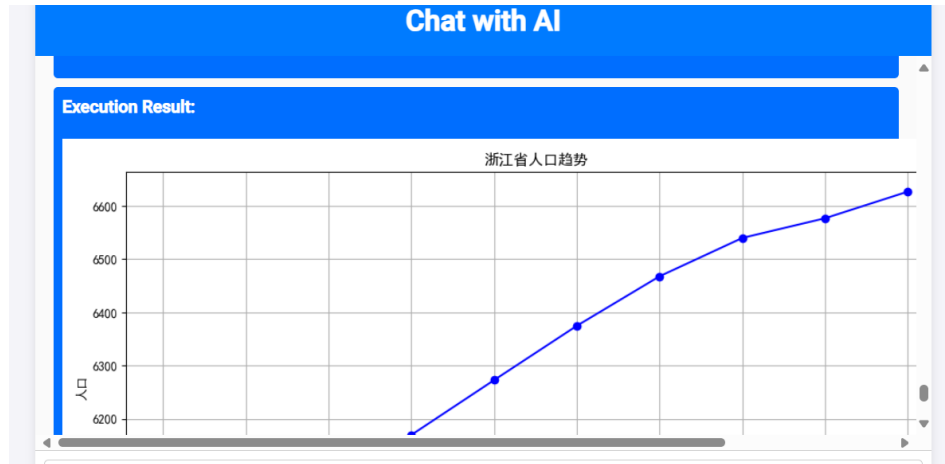


图 11: graph

得到代码如下:

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

# Set up the environment for Chinese text display
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['text.usetex'] = False

# Load the data from a CSV file
data = pd.read_csv('年末常住人口.csv')

# Filter data for a specific region (Zhejiang Province)
zhejiang_data = data[data['地区'] == '浙江省']

# Define the years and extract the corresponding population data
years = ['2014年', '2015年', '2016年', '2017年', '2018年', '2019年',
        '2020年', '2021年', '2022年', '2023年']
zhejiang_data = zhejiang_data[years].values.flatten()

# Plot the data
plt.figure(figsize=(10, 6))
plt.plot(years, zhejiang_data, marker='o', linestyle='-', color='b')
plt.title('浙江省人口趋势')
plt.xlabel('年份')
plt.ylabel('人口')
plt.grid(True)
plt.xticks(rotation=45)

# Display the plot
plt.tight_layout()
plt.show()
```

代码自动执行后，得到结果图片自动保存在 upload 文件夹中

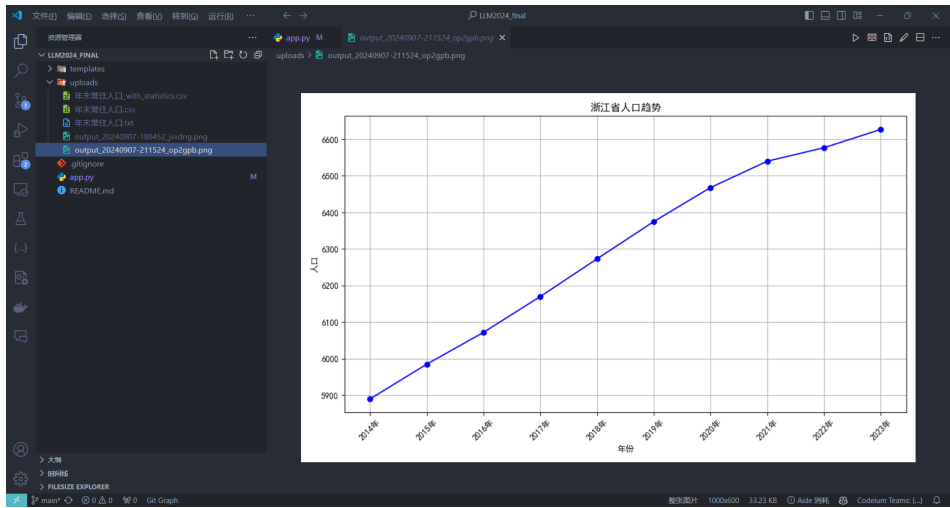


图 12: result

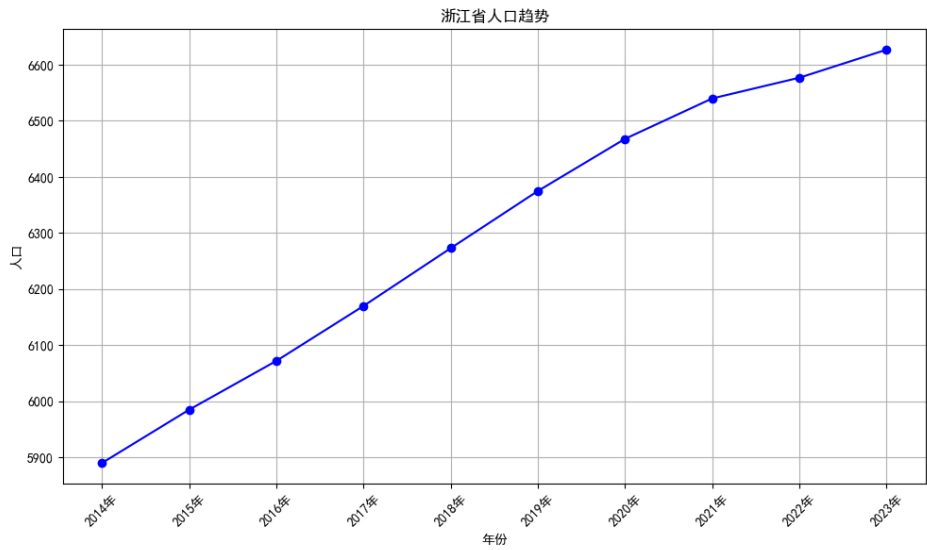


图 13: Zhejiang Province Population Trend Over 10 Years

## 4 Analysis And Comments

在这次 assignment 中，我们尝试了不同的本地大模型进行训练，我们发现这些大模型运行的时间或长或短，其代码质量也各不相同，最终我们选定了一个本地大模型 `llava-llama-3-8b-v1_1-int4.gguf` 4.92GB，对这个大模型不断地进行 prompt 的调试，在调试完成后，我们编写了一个与本地大模型交互的网页，能够自动识别大模型生成的代码并执行返回相关结果，完成了本次 assignment。

我们在实验中尝试了多种方法和策略，以优化代码生成的效果。具体而言，我们探讨了以下几种输入方式对生成结果的影响：

1. 直接输入整体任务目标：将完整的任务描述输入模型，要求模型生成全部代码。这种方法尽管简洁，但在代码复杂度较高的情况下，生成结果往往不理想，代码质量较低，且错误较多。
2. 拆分细化任务目标：将任务目标分解为多个步骤，逐步引导模型生成代码。此方法较为有效，模型生成的代码质量有所提升，但仍存在一定的不准确性。
3. 输入相关代码参考：提供参考代码片段，以指导模型生成类似结构的代码。此方法在代码结构较为清晰的情况下表现较好，但当任务要求较高时，模型依然难以完全满足需求。
4. 将 txt 内容全部输入模型：尝试将大量文本内容直接输入模型，以期获得较为详尽的代码生成结果。此方法在处理大量数据时效果有限，生成代码质量参差不齐。

经过多轮实验，我们最终发现，采用逐步输入代码步骤及相应函数的方法效果最佳。具体做法是，将代码生成任务拆解为多个小步骤，明确每一步的具体要求与所需函数，并逐一输入模型。这种“保姆式教学”输入方式，能够显著提高模型生成代码的质量和通过率。

在网站的构建过程中，我们通过 OpenAI 客户端和本地大模型进行交流，并通过 Flank 框架的文件上传功能保存文件，通过 HTML 标签展示图片，通过 `exec` 函数执行代码动态导入所需库，最终搭建起网页平台，其最终结果已经呈现在上文。

## 5 Summary

在这次作业中，我们利用本地的大语言模型 (LLMs) 自动进行数据处理和可视化任务，实现了大模型的多模态输出功能，并通过实践共计完成了三个主要目标：处理数据以计算人口统计指标，使用 Python 代码分析数据，以及通过图表可视化趋势。

总体而言，这次作业展示了本地大语言模型在执行复杂的数据分析和可视化任务方面的潜力以及 prompt 工程的重要性，通过合适的 prompt 可以让大模型输出令人满意的结果。