Data Analysis platform for the Big Five Leagues of football

Sun Weitai

AI_ 2　　202264640330

The aim of the project is to build a Data Analysis platform to analysis the information of the Leagues of the footballs this season, and the players' information.

To construct the platform, we can divide the program into 4 parts: Get information from the website, Store the information, Provide the information when searched and GUI design.

Get information from the website: There are 3 parts of information that we are going to get. The players' information, the league table, League Shooters and Assistants Chart. In order to get the html text from the website, we use request module and provide the website we are like to get, and then we transform the response(html) to content(text),and then create a object soup.

```
content =" "
for chunk in response.iter_content(chunk_size=8192):
    if chunk:
        content += chunk.decode("utf-8",errors='ignore')
soup = BeautifulSoup(content, "html.parser")
```

We use soup.select() to extract the information we want. We find the selector of the part we want to extract and paste it into soup.select(). Repeat 20 times to get all the teams' information in the league. And transform the information into text using for loop and add them all to a list. Finally we return the list at the end of the function.

```
information_pre = [team_name, plays, wins, loses, ties, goals, lose_goals, goals_difference, points]
Team = list()
# Transform the information to text
for element in information_pre:
    if element != []:
        for s in element:
            s = s.text
        Team.append(s)
# Append the team information to the list if it isn't empty
if team_name != []:
    information.append(Team)
```

In order to access the League Shooters and Assistants Chart, the processes is much like get League table, we just have to assign different variables to different soup.select(selector), and finally return the goal list and assist list.

The format of website of the player is

https://www.dongqiudi.com/player/ +"player's id"+.html

So, in order to search the player's information by its name, we have to get player's id and match them together. To realize that, we request the League Shooters and Assistants Chart html and transform to text and to json. And then, we can extract player's id and name from the json and put them in a dict. In this way, when we enter the player's name, we can get

the player's id and get on the player's person website to collect data. If we can request the html on the website, we just have to assign different variables to different soup.select(selector), and finally a list with player's information.

```
response = requests.get(url_2, headers=header)
content = " "
for chunk in response.iter_content(chunk_size=4096):
    if chunk:
        content += chunk.decode("utf-8", errors='ignore')
data = json.loads(content)
person_ranking = data["content"]["data"]
```

```
# Distribute the information of the player
for person in person_ranking:
    person_name = person["person_name"]
    team_name = person["team_name"]
    person_id = person["person_id"]
    person_logo = person["person_logo"]
    assist = person['count']
    player_person = [person_id, person_name]

    # Add the player's id to name list into the list
    if player_person not in Players:
        Players.append(player_person)
```

```
Player = [Name, FC, Pos, number, Country, age, Preferred_foot, Height, weight, person_rating]
# return the player's information list
return Player
```

After finishing the function of getting information, we want to store the information by write them into CSV file. First, we call different functions to get list, and set headers.

```python
def store_file(league):
    # Call different functions to get list
    information = league_data(league)
    goal_list = league_goal(league)
    assist_list = league_assist(league)
    Players = id_player(league)
    # Write information to CSV file
    header_list = ['Team Name', 'Plays', 'Wins', 'loses', 'Goals scored', 'Goals Allowed', 'Goals Differential', 'Points']
    headers_list_p = ['ID', 'Name']
    headers_list_g = ['Player', 'Team Name', 'Goals']
    headers_list_a = ['Player', 'Team Name', 'Assists']
    with open(""+league+"_test.csv", "w", encoding="utf-8-sig", newline="") as f:
        writer = csv.writer(f)
        writer.writerow(header_list)
        writer.writerows(information)
    with open("" + league + "_goals_test.csv", "w", encoding="utf-8-sig", newline="") as f:
        writer = csv.writer(f)
        writer.writerow(headers_list_g)
        writer.writerows(goal_list)
    with open("" + league + "_assists_test.csv", "w", encoding="utf-8-sig", newline="") as k:
        writer = csv.writer(k)
        writer.writerow(headers_list_a)
        writer.writerows(assist_list)
    with open("Players "+str(league)+"_test.csv", "w", encoding="utf-8-sig", newline="") as p:
        writer = csv.writer(p)
        writer.writerow(headers_list_p)
        writer.writerows(Players)
```

The player's id and name are access leagues by leagues so we'd better to merge all the csv into one CSV file to make search easier. We can read all the csv file one by one and write into one csv file. In this way, when we search player's information, we can just access player's id by the total csv file.

```python
def merge_players():
    # Create headers
    headers_list_p = ['ID', 'Name']
    csv_files = ['Players Serie A_test.csv', 'Players Fußball-Bundesliga_test.csv', 'Players La Liga_test.csv', 'Players Ligue 1_test.csv',
                 'Players Premier League_test.csv']

    merged_file = 'Players_test.csv'
    # Merge multiple csv files into one, get players' id to name
    with open(merged_file, 'w', newline='') as outfile:
        writer = csv.writer(outfile)
        writer.writerow(headers_list_p)

        for filename in csv_files:
            with open(filename, 'r', encoding="utf-8-sig", newline='') as infile:
                reader = csv.reader(infile)
                next(reader)
                writer.writerows(reader)
```

Finally, we will design the GUI, first create a window to execute the program, set the size and the title of the window. Create a button to

execute the function collect_data, which collect all the leagues and players' information and write them into csv file.

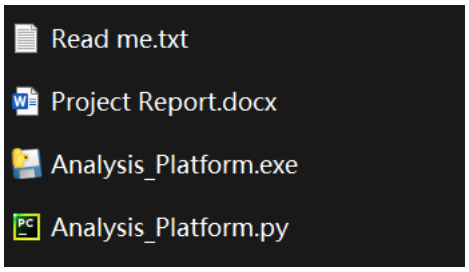
Then, we design a selection bar to choose the category that users want to know about. Create a button to execute function 'next_step', which will operate differently by the category.

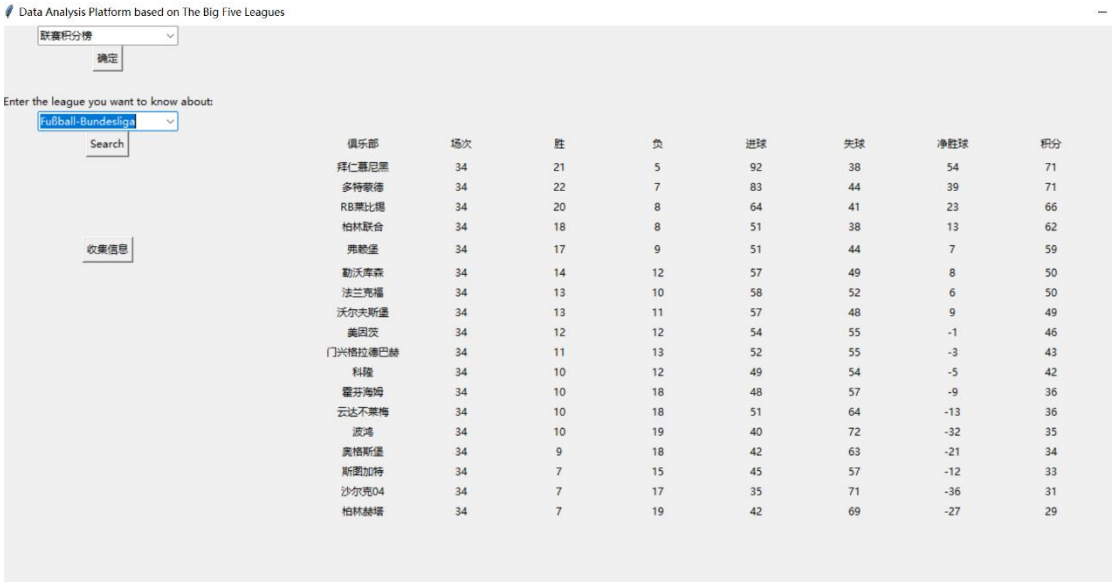
First, it will clear the previous results by using empty label. Then, it will require the user to choose from selection bar or enter player's name. And Call the league_data function to get the information, display the information for each club in the league and the column headers.

```
# Display the information for each club in the league
for i in range(0, len(information)):
    for j in range(0,len(information[i])):
        label_result = tk.Label(window, text='')
        label_result.grid(row=i+6, column=3*j+9)
        label_result.config(text=information[i][j])
```

How to display the players' information, the league table, League Shooters and Assistants Chart are similar to the operation above.

Finally, we realize all the function that we want. Now we will pack the .py file into .exe by using pyinstaller package.





俱乐部	场次	胜	负	进球	失球	净胜球	积分
拜仁慕尼黑	34	21	5	92	38	54	71
多特蒙德	34	22	7	83	44	39	71
RB莱比锡	34	20	8	64	41	23	66
柏林联合	34	18	8	51	38	13	62
弗赖堡	34	17	9	51	44	7	59
勒沃库森	34	14	12	57	49	8	50
法兰克福	34	13	10	58	52	6	50
沃尔夫斯堡	34	13	11	57	48	9	49
奥因茨	34	12	12	54	55	-1	46
门兴格拉德巴赫	34	11	13	52	55	-3	43
科隆	34	10	12	49	54	-5	42
霍芬海姆	34	10	18	48	57	-9	36
云达不莱梅	34	10	18	51	64	-13	36
波鸿	34	10	19	40	72	-32	35
奥格斯堡	34	9	18	42	63	-21	34
斯图加特	34	7	15	45	57	-12	33
沙尔克04	34	7	17	35	71	-36	31
柏林赫塔	34	7	19	42	69	-27	29