Code Review Report

Họ và tên : Nguyễn Việt Tiến - MSV: B22DCCN726
Lớp: D22CQCN06-B
Lớp học : INT13162-20241-11
Giảng viên : Kim Ngọc Bách

Câu 1:

Giới thiệu

Tài liệu "cau1.py" là một tập lệnh Python được thiết kế để thu thập và tổ chức dữ liệu từ trang web **FBRef**, một nguồn thông tin nổi tiếng về thống kê bóng đá. Mục tiêu của tập lệnh này là thu thập dữ liệu chi tiết về cầu thủ và đội bóng trong giải đấu **Premier League** mùa giải **2023-2024**. Việc thu thập dữ liệu này có thể phục vụ cho nhiều mục đích khác nhau, bao gồm phân tích hiệu suất cầu thủ, so sánh đội bóng và nghiên cứu xu hướng thi đấu.

- Để chạy đoạn code, mình phải tải thư viện cần thiết :
  - pip install selenium
  - pip install beautifulsoup4
  - pip install webdriver-manager

Các phần chính của mã

1. Nhập khẩu thư viện:

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
import time
from bs4 import BeautifulSoup
```

- **Selenium**: Thư viện này cho phép tự động hóa trình duyệt web, giúp thực hiện các tác vụ như nhấp chuột, điền biểu mẫu và thu thập dữ liệu.
- **BeautifulSoup**: Thư viện này được sử dụng để phân tích cú pháp HTML và XML, giúp dễ dàng trích xuất thông tin từ các trang web.
- **webdriver_manager**: Hỗ trợ tự động tải và cài đặt driver cho trình duyệt Chrome, giúp giảm thiểu công sức chuẩn bị môi trường.

2. Khởi tạo trình duyệt:

```
driver =
webdriver.Chrome(service=Service(ChromeDriverManager().install()))
```

Dòng mã này khởi tạo một phiên bản trình duyệt Chrome bằng cách sử dụng driver đã được cài đặt tự động. Điều này cho phép mã truy cập các trang web cần thiết để thu thập dữ liệu.

3. Hàm validdata:

```
def validdata(n):
        if n=='':   return "N/a"
        return float(n)
```

Hàm này kiểm tra giá trị đầu vào. Nếu giá trị là chuỗi rỗng, nó trả về "N/a" để biểu thị rằng không có dữ liệu. Nếu không, nó chuyển đổi giá trị sang kiểu float, giúp chuẩn hóa dữ liệu để dễ dàng xử lý sau này.

4. Hàm `getDataFromWeb` :

Hàm này thực hiện các bước sau:

- **Truy cập vào URL đã cung cấp**: Sử dụng `driver.get(url)` để mở trang web.
- **Chờ tải trang**: Sử dụng `time.sleep(3)` để đảm bảo trang đã tải hoàn toàn.
- **Phân tích HTML**: Sử dụng BeautifulSoup để phân tích nội dung HTML của trang.
- **Trích xuất dữ liệu**:
    - **Dữ liệu cầu thủ**: Tìm kiếm bảng có id là `idPlayerTable`, sau đó trích xuất từng hàng dữ liệu cầu thủ.
    - **Dữ liệu đội bóng**: Tìm kiếm bảng có id là `idSquadTable` và trích xuất thông tin đội bóng.

Kết quả được lưu vào hai danh sách: `resultPlayerData` và `resultSquadData`.

5. Thu thập dữ liệu:

Mã thực hiện một loạt các cuộc gọi đến hàm `getDataFromWeb` với các URL khác nhau để thu thập dữ liệu từ nhiều khía cạnh khác nhau, bao gồm:

- **Thời gian thi đấu**: Thống kê về số trận đã chơi, số lần đá chính, số phút thi đấu.
- **Thống kê chuẩn**: Thông tin về số bàn thắng, kiến tạo, thẻ vàng, thẻ đỏ.
- **Thống kê thủ môn**: Thông tin về số bàn thua, số lần cứu thua, tỷ lệ cứu thua.
- **Thống kê về sút bóng**: Số cú sút, số cú sút trúng đích, tỷ lệ sút trúng đích.
- **Thống kê về chuyền bóng**: Thống kê về số đường chuyền thành công, số đường chuyền không thành công.
- **Thống kê phòng ngự**: Thông tin về số pha tắc bóng, số pha cản phá.
- **Thống kê quyền sở hữu**: Thống kê về số lần chạm bóng, số lần giữ bóng.

6. Cập nhật dữ liệu:

Sau khi thu thập dữ liệu, mã cập nhật thông tin cho các đối tượng `Player` và `Squad`.

- **Đối với cầu thủ**: Nếu cầu thủ chưa tồn tại trong hệ thống, mã sẽ tạo mới một đối tượng `Player` và thêm thông tin về thời gian thi đấu, hiệu suất và các thống kê khác.
- **Đối với đội bóng**: Tương tự, mã thêm hoặc cập nhật thông tin cho đối tượng `Squad`.

7. Lưu trữ dữ liệu:

Dữ liệu sau khi được thu thập và cập nhật được lưu vào:

- **Tệp pickle**: Sử dụng để lưu trữ các đối tượng Python một cách dễ dàng (`squads.pkl`).
- **Tệp CSV**: Dữ liệu cuối cùng được xuất ra tệp CSV (`result.csv`) để dễ dàng truy cập và phân tích sau này.

8. Kết quả:

Mã in ra số lượng đội bóng và cầu thủ đã được thu thập, cùng với thông báo hoàn thành việc ghi dữ liệu vào tệp CSV và mở tệp CSV để xem kết quả.

Câu 2:

Giới thiệu

Tài liệu "cau2.py" là một tập lệnh Python được thiết kế để phân tích và trực quan hóa dữ liệu thu thập từ giải bóng đá Premier League mùa giải 2023-2024. Tập lệnh này sử dụng các thư viện như `pickle`, `csv`, `statistics`, `pandas`, và `matplotlib` để xử lý và phân tích dữ liệu, cũng như tạo các biểu đồ giúp trực quan hóa thông tin.

- Để chạy đoạn code, mình phải tải thư viện cần thiết :
  - pip install matplotlib

Các phần chính của mã

1. Nhập khẩu thư viện:

```
import pickle
import csv
import statisticsfrom common
import header, row, row2, header2, rowsquad
import pandas as pd
import os
```

- **pickle**: Được sử dụng để lưu trữ và tải dữ liệu đã được tuần tự hóa.
- **csv**: Thư viện để đọc và ghi tệp CSV.

- **statistics**: Cung cấp các hàm để tính toán các chỉ số thống kê như trung bình, trung vị, và độ lệch chuẩn.
- **pandas**: Thư viện mạnh mẽ để xử lý và phân tích dữ liệu.
- **matplotlib**: Thư viện để tạo biểu đồ trực quan hóa dữ liệu.

2. Tải dữ liệu:

```
list_squad = []
with open("squads.pkl", "rb") as file:
    list_squad = pickle.load(file)

df = pd.read_csv('result.csv')
df.replace("N/a", 0, inplace=True)
```

Mã bắt đầu bằng việc tải danh sách đội bóng từ tệp `squads.pkl` và dữ liệu kết quả từ tệp `result.csv`. Các giá trị "N/a" được thay thế bằng 0 để dễ dàng xử lý.

3. Phân tích dữ liệu:

3.1. Tìm Top 3 và Bottom 3:

```
for i in range(5, ATTR_NUMBER):
        top_3_rows = df.nlargest(3, df.columns[i])
        print("Top 3 cao nhất thuộc tính", header[i])
        print(top_3_rows.iloc[:, 0].values)
        print("Top 3 thấp nhất thuộc tính", header[i])
        bot_3_rows = df.nsmallest(3, df.columns[i])
        print(bot_3_rows.iloc[:, 0].values)
```

Mã tìm và in ra 3 đội bóng có giá trị cao nhất và thấp nhất cho từng thuộc tính trong dữ liệu. Điều này giúp xác định những đội bóng nổi bật cũng như những đội bóng có hiệu suất kém hơn.

3.2. Tính toán các chỉ số thống kê:

```
mean_value_list = [0] * (ATTR_NUMBER - 5)
median_value_list = [0] * (ATTR_NUMBER - 5)
std_dev_list = [0] * (ATTR_NUMBER - 5)

for index, arr in enumerate(all_attr):
    mean_value_list[index] = statistics.mean(arr)
    median_value_list[index] = statistics.median(arr)
    std_dev_list[index] = statistics.stdev(arr)
```

Các chỉ số thống kê như trung bình, trung vị và độ lệch chuẩn được tính toán cho từng thuộc tính. Những kết quả này được ghi vào tệp CSV `result2.csv`.

4. Trực quan hóa dữ liệu:

4.1. Vẽ biểu đồ histogram cho tất cả các thuộc tính:

```
plt.hist(arr, bins=30, color='blue', alpha=0.7, edgecolor='black')
```

Mã tạo và lưu các biểu đồ histogram cho từng thuộc tính trong dữ liệu. Các biểu đồ này giúp người dùng dễ dàng hình dung phân phối giá trị cho từng thuộc tính.

4.2. Vẽ biểu đồ cho từng đội bóng:

```
for squad in squads:
        output_squad_directory = f'histogramsof{squad}'
        os.makedirs(output_squad_directory, exist_ok=True)
```

Tạo thư mục riêng cho từng đội bóng và lưu biểu đồ histogram cho các thuộc tính của đội đó. Điều này cho phép so sánh trực tiếp giữa các đội bóng.

5. Tìm đội bóng có chỉ số cao nhất:

```
for index, value in enumerate(arr):
    if value != "N/a" and value > biggest_attr_value[index]:
        biggest_attr_value[index] = value
        best_team_in_one_attr[index] = i.name
```

Mã tìm kiếm đội bóng có chỉ số cao nhất cho từng thuộc tính và in ra kết quả. Bên cạnh đó, nó cũng xác định đội bóng ấn tượng nhất trong giải đấu dựa trên các chỉ số này.