HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÁO CÁO BÀI TẬP LỚN

MÔN HỌC: LẬP TRÌNH PYTHON

NHÓM MÔN HỌC: 11

Giảng viên: Kim Ngọc Bách

Sinh viên: Dương Hải Lưu – B2DCCN513

Hà Nội 2023-2024

Mục lục

I. Viết chương trình Python thu thập dữ liệu phân tích cầu thủ

- Định nghĩa hàm GetDataFromWeb, sử dụng Selenium để truy cập vào một trang web, thu thập dữ liệu được yêu cầu từ hai bảng khác nhau (xác định bởi các XPath), xử lí và trả về dữ liệu đã thu thập.

```
def GetDataFromWeb(url, Xpath_player, Xpath_squad, Data_Name):  10 usages
    driver = webdriver.Chrome()
    driver.get(url)

    resultPlayerData = []
    resultSquadData = []
    try:
        time.sleep(10)
        table2 = driver.find_element(By.XPATH, Xpath_player)
        rows2 = table2.find_elements(By.TAG_NAME, value: 'tr')

        for row in rows2:  # Bỏ qua hàng tiêu đề
            cols = row.find_elements(By.TAG_NAME, value: 'td')
            data = []
            for id, play in enumerate(cols[:-1]):
                if id == 1:
                    a = play.text.strip().split()
                    if len(a) == 2:
                        data.append(a[1])
                    else:
                        data.append(play.text.strip())
                else:
                    s = play.text.strip()
                    if id >= 4:
                        s = s.replace( _old: ",", _new: "")
                        s = validdata(s)
                    data.append(s)
            if len(data) != 0: resultPlayerData.append(data)
```

(thu nhập dữ liệu của các cầu thủ)

```
        table1 = driver.find_element(By.XPATH, Xpath_squad)
        rows1 = table1.find_elements(By.TAG_NAME, value: 'tr')

        for row in rows1[2:]:  # Bỏ qua hàng tiêu đề
            data = []
            name = row.find_element(By.TAG_NAME, value: 'th')
            data.append(name.text.strip())

            cols = row.find_elements(By.TAG_NAME, value: 'td')
            for id, value in enumerate(cols):
                s = value.text.strip()
                if id >= 4:
                    s = s.replace( _old: ",", _new: "")
                    s = validdata(s)
                data.append(s)
            if len(data) != 0: resultSquadData.append(data)

    finally:
        driver.quit()
        print("Finish Page " + DataName)
    return resultPlayerData, resultSquadData
```

(Thu thập dữ liệu của các đội)

- Sử dụng hàm chỉ số nào không có hoặc không áp dụng thì để là N/a

```
def validdata(n):  2 usages
    if n == '': return "N/a"
    return float(n)
```

- Truy cập vào trang web Premier League của FBRef để lấy dữ liệu cầu thủ và đội bóng, rồi xử lý và lưu trữ dữ liệu trong các đối tượng Player và Squad thông qua các hàm trong player_manager và squad_manager

```
url = "https://fbref.com/en/comps/9/2023-2024/stats/2023-2024-Premier-League-Stats"
xpath_player = '//*[@id="stats_standard"]'
xpath_Squad = '//*[@id="stats_squads_standard_for"]'
DataName = "Standard"
list_player_result, list_Squad_result = GetDataFromWeb(url,xpath_player, xpath_Squad, DataName)

for i in list_player_result:
    p = player_manager.findPlayerByNameandTeam(i[0],i[3])
    if p == None:
        new_p=Player(i[0],i[1],i[2],i[3],i[4])
        #bo qua i5 vi i5 la nam sinh
        new_p.setPlaying_time(i[6:9])
        #bo qua i9
        new_p.setPerformance([i[13],i[14],i[11],i[16],i[17]])
        new_p.setExpected(i[18:21])
        #bo qua i22
        new_p.setProgression(i[22:25])
        new_p.setPer90(i[25:])
        player_manager.add_Player(new_p)
```

```
#squad data
for i in list_Squad_result:
    s = squad_manager.findSquadByName(i[0])
    if s == None:
        new_s=Squad(*i[0:4])
        new_s.setPlaying_time(i[4:7])
        #bo qua i9
        new_s.setPerformance([i[11],i[12],i[9],i[14],i[15]])
        new_s.setExpected(i[16:19])
        #bo qua i22
        new_s.setProgression(i[20:22])
        new_s.setPer90(i[22:])
        squad_manager.add_Squad(new_s)
```

(áp dụng tương tự với các trang khác để có thể lấy hết tất cả các cột dữ liệu)

- Thứ tự các cầu thủ sắp xếp theo thứ tự tên (First Name), nếu trùng tên thì xếp theo độ tuổi từ lớn đến nhỏ.(sử dụng hàm sortingByName đã được tạo sẵn trong class Player_Manager)

```
player_manager.sortingByName()
```

- Thu thập dữ liệu của tất cả các cầu thủ có số phút thi đấu cần nhiều hơn 90 phút (Sử dụng hàm filtering trong class Player_Manager để lọc)

```
player_manager.filtering()
```

- Tạo file CSV (result.csv) để lưu trữ dữ liệu của cầu thủ theo cấu trúc yêu cầu từ các đối tượng trong player_manager

```
import csv
from bai1.tieu_de import header, row

with open('E:/World/Code/Python/BTL/bai1/file/result.csv', mode='w', newline='', encoding='utf-8') as file:
    writer = csv.writer(file)
    writer.writerow(header)

    for player in player_manager.list_player:
        r = row(player)
        writer.writerow(r)
print("Exam 1 Success")
```

II. (2đ)

1. Tìm top 3 cầu thủ có điểm cao nhất và thấp nhất ở mỗi chỉ số.
   - Lấy dữ liệu các cầu thủ từ file player_data.txt
   -

```
def read_player_data_from_txt(file_path):  1 usage
    players = []
    with open(file_path, 'r', encoding='utf-8') as file:
        # Bỏ qua dòng tiêu đề
        header = file.readline().strip().split("\t")

        # Đọc từng dòng dữ liệu
        for line in file:
            # Tách các giá trị bằng dấu tab và thêm vào danh sách
            values = line.strip().split("\t")
            players.append(values)
    return players

# Sử dụng hàm để đọc dữ liệu cầu thủ từ file player_data.txt
player_data = read_player_data_from_txt('E:/World/Code/Python/BTL/bai1/file/player_data.txt')
```

- Với mỗi chỉ số chúng ta sort để xắp xếp thứ tự và top 3 cầu thủ có điểm cao nhất và thấp nhất vào class max_min:

```
from tieu_de import header
from max_min_player import *

list_mm = max_min()

for index, value in enumerate(header):
    if index < 3: continue
    player_data = sorted(player_data, key=lambda x: x[index])
    list_mm.add_max_min([value, player_data[0][0], player_data[1][0], player_data[2][0],
                         player_data[-3][0],player_data[-2][0], player_data[-1][0]])
```

- Lưu dữ liệu vào file result.xlsx

```
import openpyxl
# Lưu dữ liệu vào file Excel cho player_manager
wb = openpyxl.Workbook()
ws = wb.active
ws.title = "Players"

# Ghi tiêu đề (header)
ws.append(header_max_min)
# Ghi dữ liệu của các cầu thủ
for player in list_mm.list_max_min:
    ws.append(player)

# Lưu vào file Excel
wb.save('E:/World/Code/Python/BTL/bai2/file/result.xlsx')
print("Exam 1 Success - Player Data Saved")
```

2. Tìm trung vị của mỗi chỉ số. Tìm trung bình và độ lệch chuẩn của mỗi chỉsố cho các cầu thủ trong toàn giải và của mỗi đội. Ghi kết quả ra file results2.csv
   - Tải dữ liệu và tạo hàm để thêm dữ liệu vào cho mỗi chỉ số

```
# Tải dữ liệu từ file CSV vào DataFrame df
df = pd.read_csv('E:/World/Code/Python/BTL/bai2/file/result.csv')

def add_statistics_for_team(team_name, numeric_df, table):  2 usages
    table['Team'].append(team_name)
    for att in numeric_df.columns:
        table['Median of ' + att].append(float(numeric_df[att].median()))
        table['Mean of ' + att].append(float(numeric_df[att].mean()))
        table['Std of ' + att].append(float(numeric_df[att].std()))
```

- Tính toán dữ liệu cho từng cầu thủ trong giải:

```
numeric_df = df.select_dtypes(include=['float', 'int'])
# Tạo dictionary để lưu kết quả
table = {'Team': []}
# Khởi tạo các cột cho Median, Mean, Std của từng thuộc tính số
for att in numeric_df.columns:
    table['Median of ' + att] = []
    table['Mean of ' + att] = []
    table['Std of ' + att] = []
    # Tính toán cho tất cả các đội (team 'all')
add_statistics_for_team( team_name: 'all', numeric_df, table)
teams=['all']
teams.extend(df['team'].unique())
```

- Tính toán dữ liệu cho từng đội trong giải:

```
    # Tính toán cho từng đội
for team in teams[1:]:
        # Bỏ qua 'all' vì đã tính toán trước
    filtered_df = df[df['team'] == team]
    numeric_df = filtered_df.select_dtypes(include=['float', 'int'])
    add_statistics_for_team(team, numeric_df, table)
  # Tạo DataFrame từ dictionary 'table'
result2 = pd.DataFrame(table)
```

3. Vẽ historgram phân bố của mỗi chỉ số của các cầu thủ trong toàn giải và mỗi đội.
   - Lấy dữ liệu cần thiết và tạo các thư mục để lưu cái file .png

```
# Đọc dữ liệu
df = pd.read_csv('E:/World/Code/Python/BTL/bai2/file/result.csv')

# Tạo thư mục để lưu biểu đồ nếu chưa tồn tại
output_folder = 'E:/World/Code/Python/BTL/bai2/output_histograms'
os.makedirs(output_folder, exist_ok=True)
```

   - Vẽ historgram cho các cầu thủ trong giải đấu

```
# Vẽ histogram cho toàn bộ giải đấu
numeric_df = df.select_dtypes(include=['float', 'int'])
num_attributes = len(numeric_df.columns)
num_cols = 4
num_rows = math.ceil(num_attributes / num_cols)

plt.figure(figsize=(16, 10))  # Tăng chiều cao
for idx, att in enumerate(numeric_df.columns, 1):
    plt.subplot( *args: num_rows, num_cols, idx)
    plt.hist(numeric_df[att], bins=10, alpha=0.7, color='blue', edgecolor='black')
    plt.title( label: f'Histogram of {att} (All Teams)', fontsize=10)  # Giảm kích thước font tiêu đề
    plt.xlabel(att, fontsize=8)  # Giảm kích thước font nhãn
    plt.ylabel( ylabel: 'Frequency', fontsize=8)  # Giảm kích thước font nhãn
    plt.grid(axis='y', alpha=0.75)

plt.subplots_adjust(hspace=0.5, wspace=0.3)  # Điều chỉnh khoảng cách giữa các ô con
plt.savefig(os.path.join(output_folder, 'all_teams_histogram.png'))
plt.close()
```

   - Vẽ historgram cho các cầu thủ trong mỗi đội

```
# Vẽ histogram cho từng đội
teams = df['team'].unique()
for team in teams:
    team_df = df[df['team'] == team].select_dtypes(include=['float', 'int'])
    num_attributes_team = len(team_df.columns)
    num_rows_team = math.ceil(num_attributes_team / num_cols)

    plt.figure(figsize=(16, 10))  # Tăng chiều cao
    for idx, att in enumerate(team_df.columns, 1):
        plt.subplot( *args: num_rows_team, num_cols, idx)
        plt.hist(team_df[att], bins=10, alpha=0.7, color='green', edgecolor='black')
        plt.title( label: f'Histogram of {att} ({team})', fontsize=10)  # Giảm kích thước font tiêu đề
        plt.xlabel(att, fontsize=8)  # Giảm kích thước font nhãn
        plt.ylabel( ylabel: 'Frequency', fontsize=8)  # Giảm kích thước font nhãn
        plt.grid(axis='y', alpha=0.75)

    plt.subplots_adjust(hspace=0.5, wspace=0.3)  # Điều chỉnh khoảng cách giữa các ô con
    plt.savefig(os.path.join(output_folder, f'{team}_histogram.png'))
    plt.close()
```

4. Tìm đội bóng có chỉ số điểm số cao nhất ở mỗi chỉ số. Theo bạn đội nào có phong độ tốt nhất giải ngoại Hạng Anh mùa 2023-2024

   - Chuẩn bị dữ liệu và lọc thêm tên cái đội bóng để tìm kiếm chỉ số có giá trị lớn nhất

```python
df2 = pd.read_csv('E:/World/Code/Python/BTL/bai2/file/result2.csv')
df2.drop(index=0, inplace=True)
numeric_columns = df2.select_dtypes(include=['float', 'int']).columns

# Tạo danh sách để lưu kết quả cho mỗi thuộc tính
highest_team_per_stat = []

for column in numeric_columns:
    # Tìm chỉ số dòng có giá trị lớn nhất cho thuộc tính
    max_idx = df2[column].idxmax()

    # Lấy tên đội bóng và giá trị của thuộc tính tại chỉ số này
    max_team = df2.loc[max_idx, 'Team']
    max_score = df2.loc[max_idx, column]

    # Thêm kết quả vào danh sách
    highest_team_per_stat.append({
        'Attribute': column,
        'Team': max_team,
        'Highest Score': max_score
    })
```

   - Lưu lại dữ liệu của bảng highest_team_per_stat vào result3.csv

```python
# Chuyển đổi danh sách kết quả thành DataFrame
highest_team_per_stat_df = pd.DataFrame(highest_team_per_stat)

    # lưu vào result3.csv
highest_team_per_stat_df.to_csv( path_or_buf: 'E:/World/Code/Python/BTL/bai2/file/result3.csv', index=False)
```

   - In ra mỗi đội bóng có bao nhiêu chỉ số cao nhất theo thứ tự giảm dần

```
# đếm  xem mỗi đội bóng có bao nhiêu chỉ số điểm cao nhất
print(highest_team_per_stat_df['Team'].value_counts())
```

```
Team
Manchester City    107
Liverpool           50
Fulham              33
Arsenal             20
Everton             16
Bournemouth         15
West Ham            14
Tottenham           13
Newcastle Utd       11
Crystal Palace       7
Chelsea              6
Luton Town           5
Aston Villa          5
Wolves               3
Manchester Utd       2
Sheffield Utd        2
Brighton             1
Burnley              1
Brentford            1
```

⇨ Theo dữ liệu thu nhập được thì Manchester City đang có phong độ cao nhất với 107 chỉ số đứng đầu

III. (3đ)

1. Sử dụng thuật toán K-means để phân loại các cầu thủ thành các nhóm có chỉ số giống nhau
   - Xử lý các giá trị thiếu nếu có

```
numeric_columns = df.select_dtypes(include=[np.number]).columns
df[numeric_columns] = df[numeric_columns].fillna(df[numeric_columns].mean())
```

   - Chuẩn hóa dữ liệu số

```
numerical_data = df.select_dtypes(include=[float, int])
scaler = StandardScaler()
X = scaler.fit_transform(numerical_data)
```

- Tìm số lượng cụm tối ưu

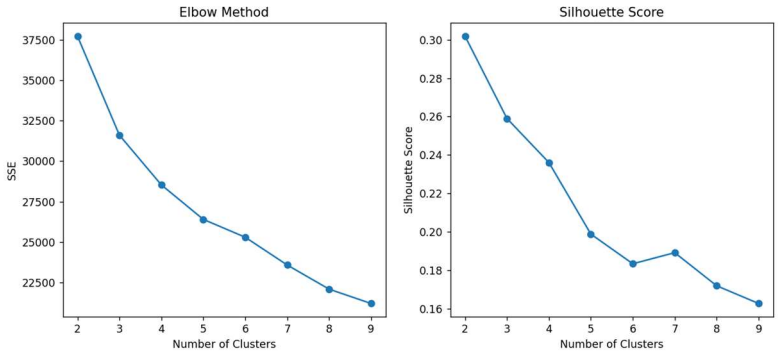
```
sse = []
silhouette_scores = []
for k in range(2, 10):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X)
    sse.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(X, kmeans.labels_))

plt.figure(figsize=(12, 5))
```

- Phương pháp Elbow để xác định SSE

```
plt.subplot( *args: 1, 2, 1)
plt.plot( *args: range(2, 10), sse, marker='o')
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.title("Elbow Method")
```

- Đánh giá Silhouette Score

```
plt.subplot( *args: 1, 2, 2)
plt.plot( *args: range(2, 10), silhouette_scores, marker='o')
plt.xlabel("Number of Clusters")
plt.ylabel("Silhouette Score")
plt.title("Silhouette Score")
plt.show()
```

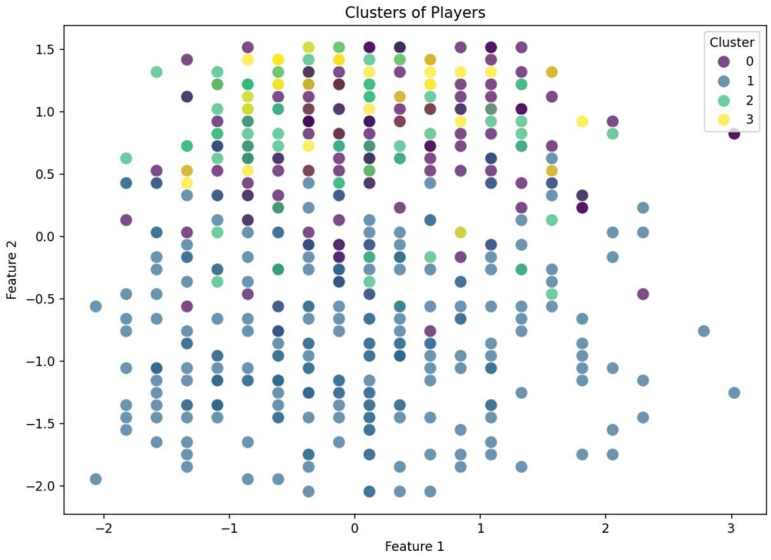


⇨ *Sử dụng số cụm = 4 dựa trên kết quả từ đồ thị*

```
optimal_k = 4
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
df['Cluster'] = kmeans.fit_predict(X)
```

- Trực quan hóa các cụm

```
plt.figure(figsize=(10, 7))
sns.scatterplot(x=X[:, 0], y=X[:, 1], hue=df['Cluster'], palette='viridis', s=100, alpha=0.7)
plt.title('Clusters of Players')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend(title='Cluster')
plt.show()
```



⇨ **<u>Nhận xét:</u>**

Dựa vào hai biểu đồ, bạn chọn **số cụm tối ưu là 4**, có thể vì:

- Điểm "khuỷu tay" trong biểu đồ SSE xuất hiện xung quanh **k=4**, sau đó SSE giảm chậm lại.
- Silhouette Score tại k=4 có thể đạt mức cao tương đối, cho thấy các cụm khá rõ ràng.
- Cân đối về mặt dữ liệu, giúp phân nhóm mà vẫn giữ sự khác biệt rõ ràng giữa các cụm.

2. Sử dụng thuật toán PCA, giảm số chiều dữ liệu xuống 2 chiều, vẽ hình phân cụm các điểm dữ liệu trên mặt 2D.
    - Giảm số chiều dữ liệu xuống 2 chiều

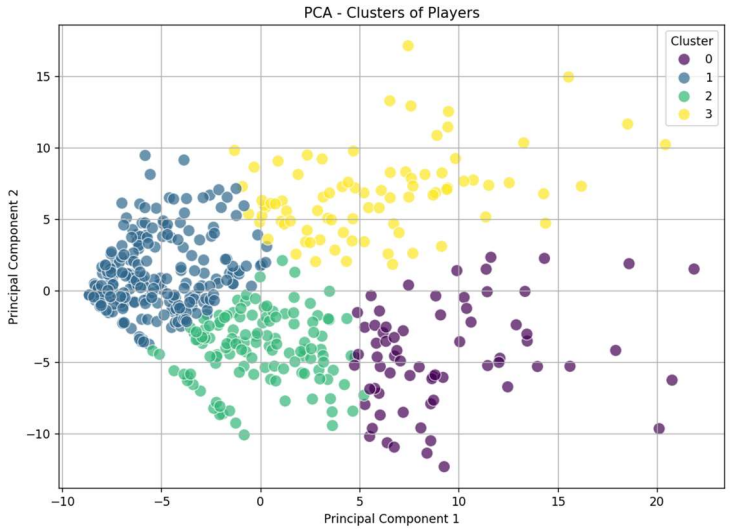
```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
```

    - Tạo DataFrame với các thành phần chính và cụm

```
df_pca = pd.DataFrame(data=X_pca, columns=['PC1', 'PC2'])
df_pca['Cluster'] = df['Cluster']
```

    - Vẽ biểu đồ phân cụm

```
plt.figure(figsize=(10, 7))
sns.scatterplot(x='PC1', y='PC2', hue='Cluster', data=df_pca, palette='viridis', s=100, alpha=0.7)
plt.title('PCA - Clusters of Players')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title='Cluster')
plt.grid(True)
plt.show()
```



3. Viết chương trình python vẽ biểu đồ rada (radar chart) so sánh cầu thủ
    - Viết hàm create_radar_chart tạo một biểu đồ radar cho một cầu thủ dựa trên các thuộc tính của họ.

```python
def create_radar_chart(ax, player, attributes):  2 usages

    df = pd.read_csv(args.file)

    # Lấy dữ liệu cho cầu thủ
    values = df.loc[df['name'] == player, attributes].values.flatten()

    # Số lượng thuộc tính
    num_vars = len(attributes)

    # Tạo một bảng cho biểu đồ radar
    angles = np.linspace( start: 0, 2 * np.pi, num_vars, endpoint=False).tolist()

    values = np.concatenate((values, [values[0]]))
    angles += angles[:1]

    colors = ['b', 'r', 'g', 'm', 'y']

    ax.set_facecolor('#f0f0f0')

    # Vẽ biểu đồ radar với màu sắc cho từng thuộc tính
    for i in range(num_vars):
        ax.plot(angles[i:i + 2], values[i:i + 2], color=colors[i % len(colors)], linewidth=2)
        ax.fill(angles, values, facecolor=colors[i % len(colors)], alpha=0.25)

    ax.set_yticklabels([])
    ax.set_xticks(angles[:-1])
    ax.set_xticklabels(attributes)

    ax.set_title(f'Radar Chart for: {player}', weight='bold', size='medium', position=(0.5, 1.1),
                 horizontalalignment='center', verticalalignment='center')
```

    - Chuẩn bị dữ liệu và danh sách chỉ số cần so sánh

```python
parser = argparse.ArgumentParser(description="Draw radar charts to compare players.")
parser.add_argument( *name_or_flags: '--file', type=str, default='E:/World/Code/Python/BTL/bai3/file/result.csv')
args = parser.parse_args()

# Đọc dữ liệu từ file
df = pd.read_csv(args.file)

# Lấy tên của hai cầu thủ từ hai dòng đầu tiên
player1 = df['name'].iloc[0]
player2 = df['name'].iloc[1]

# Đặt thuộc tính mặc định để so sánh
default_attributes = ['age', 'matches_played', 'PrgP', 'Pass_Cmp', 'Medium_Cmp', 'Pass_Live']
```

- Tạo biểu đồ vào lưu biểu đồ vào file radar_charts.png

```python
# Tạo hình ảnh với hai biểu đồ radar
fig, axs = plt.subplots( nrows: 1,  ncols: 2, figsize=(16, 8), subplot_kw=dict(polar=True))

# Vẽ biểu đồ cho mỗi cầu thủ
create_radar_chart(axs[0], player1, default_attributes)
create_radar_chart(axs[1], player2, default_attributes)

# Lưu hình ảnh vào tệp PNG
plt.savefig( *args: 'E:/World/Code/Python/BTL/bai3/file/radar_charts.png', bbox_inches='tight')
plt.show()
```

IV. Thu thập giá chuyển nhượng của các cầu thủ trong mùa 2023-2024

- Hàm GetDataFromWeb dùng Selenium để truy cập trang web và lấy dữ liệu và xử lí dữ liệu lấy được từ một bảng chứa thông tin cầu thủ.

```
def GetDataFromWeb(url, Data_Name):  2 usages
    driver = webdriver.Chrome()
    driver.get(url)
    player_data = []
    try:
        WebDriverWait(driver, timeout: 10).until(
            EC.presence_of_element_located((By.XPATH, '//*[@id="content-part"]/section/div/div/div[1]/div/div[1]')))
        div = driver.find_element(By.XPATH, value: '//*[@id="content-part"]/section/div/div/div[1]/div/div[1]')

        rows = div.find_elements(By.TAG_NAME, value: 'tr')

        for row in rows[1:]:
            data = []
            # Lấy dữ liệu từ các cột
            cols = row.find_elements(By.TAG_NAME, value: 'td')
            for index, play in enumerate(cols):
                if index == 0:
                    HaiLuu = play.find_element(By.CLASS_NAME, value: "text")
                    a = HaiLuu.text.strip().split()
                    data.append(str(a[0]) + ' ' + str(a[1]))
                elif index == 1:
                    a = play.text.strip().split('\n')
                    data.append(a[0])
                    data.append(a[1])
                elif index == 2:
                    continue
                else:
                    data.append(play.text.strip())
            player_data.append(data)

    finally:
        driver.quit()
        print("Finish Page " + Data_Name)
    return player_data
```

- Tạo 1 mảng để lưu trữ tất cả dữ liệu lấy được từ 18 web con

```
url = 'https://www.footballtransfers.com/us/leagues-cups/national/uk/premier-league/transfers/2023-2024'
Get_Player = GetDataFromWeb(url, Data_Name: '1')

for index in range(2, 19):
    x = GetDataFromWeb(url + '/' + str(index), str(index))
    Get_Player += x
```

- Thêm tiêu đề và lưu dữ liệu vào file result.csv

```
import csv
from bai4.tieu_de import header

with open('E:/World/Code/Python/BTL/bai4/file/result.csv', mode='w', newline='', encoding='utf-8') as file:
    writer = csv.writer(file)
    writer.writerow(header)
    for player in Get_Player:
        writer.writerow(player)
print("Exam 1 Success")
```

⇨ Đề xuất phương pháp định giá cầu thủ:

- So sánh cầu thủ với những người có cùng vị trí, độ tuổi, phong độ và đóng góp gần đây. Các cầu thủ tương đương về số liệu và thành tích có thể giúp dự đoán giá trị thị trường.
- Cân nhắc yếu tố "giá trị thị trường hiện tại" từ các nguồn như Transfermarkt, và so sánh các cầu thủ dựa trên các tiêu chí như kinh nghiệm và đóng góp cá nhân.