HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BỘ MÔN LẬP TRÌNH NGÔN NGỮ PYTHON

BÁO CÁO VỀ BÀI TẬP LỚN

Giảng viên hướng dẫn : **KIM NGỌC BÁCH**

Lớp : **D23CQCE06-B**

Họ Tên : **LÊ ĐĂNG HIỆP**

MSV : **B23DCDT086**

Hà Nội – 2025

LỜI MỞ ĐẦU

Trong thời đại dữ liệu phát triển mạnh mẽ như hiện nay, việc thu thập, xử lý và phân tích dữ liệu đóng vai trò then chốt trong nhiều lĩnh vực, đặc biệt là trong thể thao, nơi mà mỗi con số đều có thể tác động đến chiến thuật, giá trị chuyển nhượng hay thành tích của đội bóng. Với mong muốn vận dụng những kiến thức đã học về lập trình Python vào thực tiễn, em đã thực hiện bài tập lớn với chủ đề **"Phân tích dữ liệu thống kê cầu thủ Premier League mùa giải 2024–2025"**.

Bài báo cáo này trình bày quá trình xây dựng một hệ thống thu thập và phân tích dữ liệu cầu thủ từ trang FBref.com — một trang web cung cấp số liệu thống kê bóng đá uy tín. Dữ liệu thu thập được sẽ được xử lý, phân tích thống kê cơ bản, trực quan hóa bằng biểu đồ và ứng dụng một số phương pháp học máy để dự đoán giá trị chuyển nhượng cầu thủ.

Cụ thể, bài tập được chia thành 4 bài chính:

Bài 1: Tự động thu thập dữ liệu thống kê cầu thủ Premier League có thời gian thi đấu trên 90 phút bằng Selenium và BeautifulSoup.

Bài 2: Phân tích Top 3 cầu thủ có chỉ số cao nhất và thấp nhất ở từng hạng mục, tính toán thống kê mô tả và trực quan hóa phân phối dữ liệu bằng biểu đồ.

Bài 3: Phân tích toàn bộ các chỉ số dạng số, xác định các chỉ số hợp lệ, tính toán thống kê và đánh giá đội bóng có phong độ tốt nhất dựa trên các chỉ số tấn công.

Bài 4: Thu thập dữ liệu giá trị chuyển nhượng từ các trang chuyên về bóng đá, chuẩn hóa dữ liệu và ứng dụng mô hình Random Forest để dự đoán giá trị chuyển nhượng của các cầu thủ dựa trên các chỉ số thống kê chuyên môn.

Thông qua bài tập này, em có cơ hội vận dụng kiến thức về **web scraping**, **pandas**, **matplotlib**, **machine learning** và các kỹ năng lập trình xử lý dữ liệu thực tế. Đồng thời, bài tập cũng giúp em hiểu rõ hơn về cách tổ chức và phân tích dữ liệu thể thao một cách khoa học, hỗ trợ cho việc ra quyết định trong bóng đá hiện đại.

Em xin chân thành cảm ơn thầy **Kim Ngọc Bách** đã tận tình hướng dẫn và tạo điều kiện để em thực hiện bài tập này.

Bài 1: Lấy dữ liệu cầu thủ từ trang FBref

Bước 1: Khởi động trình duyệt và truy cập trang web

- Đầu tiên cần cài đặt thư viện selenium và khởi tạo trình duyệt tự động Chrome. Ở đây dùng **ChromeDriverManager** để tự động tải đúng version trình điều khiển phù hợp với máy.

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
```

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

-> Lệnh này khởi động một trình duyệt Chrome mới bằng **Selenium**

Tiếp theo gán địa chỉ trang web Premier League Stats vào biến url:

```
url = 'https://fbref.com/en/comps/9/stats/Premier-League-Stats'
```

-> Đây là trang chứa bảng thống kê mình cần thu thập.

Bước 2: Truy cập trang và lấy HTML

- Để trình duyệt tự động truy cập trang web và lấy nội dung HTML sau khi trang đã load ta sử dụng lệnh **driver.get(url)** để truy cập trang web. Sau đó dùng **BeautifulSoup** để trích xuất source HTML của trang. Cuối cùng tạo biến table để tìm bảng thống kê có id: **stats_standard**.

```
driver.get(url)
time.sleep(5)

soup = BeautifulSoup(driver.page_source, 'html.parser')

table = soup.find('table', {'id': 'stats_standard'})
```

Bước 3: Duyệt từng hàng cầu thủ trong hàng.

- Duyệt qua từng hàng <tr> trong hàng, lấy thông tin tên cầu thủ và số phút thi đấu. Dùng **.find('tbody').find_all('tr')** để lấy từng dòng dữ liệu. Sau đó lấy tên cầu thủ ở cột **data-stat='player'** và lấy số phút ở cột **data-stats='minutes'**

```
players_over_90 = []

for row in table.find('tbody').find_all('tr'):
    name_col = row.find('th', {'data-stat': 'player'}) or row.find('td', {'data-stat': 'player'})
    minutes_col = row.find('td', {'data-stat': 'minutes'})
```

Bước 4: Lọc cầu thủ thi đấu trên 90 phút.

- Chỉ giữ lại những cầu thủ có thời gian thi đấu trên 90 phút. Phần này sẽ kiểm tra nếu có dữ liệu tên và phút thì chuyển giá trị phút từ string sang int và so sánh điều kiện thi đấu trên 90 phút.

```
if name_col and minutes_col:
    try:
        minutes = int(minutes_col.text.replace(',', ''))
        if minutes > 90:
```

Bước 5: Lấy toàn bộ chỉ số chuyên môn của cầu thủ.

- Nếu cầu thủ đủ điều kiện, tiếp tục lấy toàn bộ chỉ số thống kê khác của cầu thủ đó. Tạo **dictionary player_info** sau đó duyệt tất cả các cột <td> còn lại. Gán tên chỉ số và giá trị vào dictionary. Thêm cầu thủ vào danh sách **player_over_90**.

```
player_info = {
    'player': name_col.text.strip(),
}
for stat in row.find_all('td'):
    stat_name = stat.get('data-stat')
    if stat_name:
        player_info[stat_name] = stat.text.strip()

players_over_90.append(player_info)
```

Bước 6: Tạo DataFrame và đổi tên cột.

- Sau khi duyệt xong tất cả cầu thủ, chuyển danh sách player_over_90 sang DataFrame để dễ xử lý tiếp.

```
df = pd.DataFrame(players_over_90)
```

-> Tạo DataFrame từ list các dictionary.

- Tiếp theo ta chuẩn hoá tên các cột về dạng quen thuộc để dễ phân tích.

```
df = df.rename(columns={
            'player': 'Player',
            'nationality' : 'Nation',
            'position' : 'Pos',
            'team': 'Squad',
            'Age': 'Age',
            'Born': 'Born',
            'age': 'Age',
            'games': 'MP',
            'games_starts': 'Starts',
            'minutes' : 'Min',
            'minutes_90s': '90s',
            'goals': 'Gls',
            'assists': 'Ast',
            'goals_assists': 'G+A',
            'goals_pens': 'G-PK',
            'pens_made': 'PK',
            'pens_att': 'PKatt',
            'cards_yellow': 'CrdY',
            'cards_red': 'CrdR',
            'xg': 'xG',
            'npxg': 'npxG',
            'xg_assist': 'xAG',
            'npxg_xg_assist': 'npxG+xAG',
            'progressive_carries': 'PrgC',
            'progressive_passes': 'PrgP',
            'progressive_passes_received': 'PrgR',
            'goals_per90': 'Gls',
            'assists_per90': 'Ast',
            'goals_assists_per90': 'G+A',
            'goals_pens_per90': 'G-PK',
            'goals_assists_pens_per90': 'G+A-PK',
            'xg_per90': 'xG',
            'xg_assist_per90': 'xAG',
            'xg_xg_assist_per90': 'xG+xAG',
            'npxg_per90': 'npxG',
```

-> Đổi tên các cột quan trọng về đúng định dạng mình muốn.

Bước 7: Sắp xếp dữ liệu và lưu vào CSV.

- Để dễ đọc và xử lý tiếp, ta sắp xếp danh sách cầu thủ theo tên và lưu vào file CSV. Trong file ta sắp xếp theo tên cầu thủ và lưu dữ liệu ra file **ketqua.csv**(dùng **utf-8-sig** để đọc tiếng Việt chuẩn).

```
df = df.sort_values(by='Player', ascending=True)

print(df)

df.to_csv('ketqua.csv', index=False, encoding='utf-8-sig')
```

Bước 8: Đóng trình duyệt.

- Tắt trình duyệt tự động sau khi hoàn thành nhiệm vụ.

```
finally:
    driver.quit()
```

Bài 2: Phân tích top cầu thủ, thống kê mô tả và trực quan hoá.

Bước 1: Thiết lập các hằng số và đọc dữ liệu.

- Khai báo các hằng số tên file, số lượng cầu thủ top cần tìm, các chỉ số cần phân tích và đọc dữ liệu từ file CSV kết quả của **Bài 1**.

```
INPUT_CSV_FILE = 'ketqua.csv'
TOP_N = 3
```

```
SELECTED_STATS = ['Gls', 'Ast', 'xG', 'CrdY', 'CrdR', 'PrgP']
```

```
df = pd.read_csv(INPUT_CSV_FILE, encoding='utf-8')
```

Bước 2: Chuyển đổi dữ liệu chỉ số về dạng số.

- Các cột chỉ số trong bảng có thể đang ở dạng string. Cần chuyển về dạng số để tính toán được. Chuyển đổi từng cột trong **SELECTED_STATS** sang numeric, nếu lỗi thì trả về **NaN**.

```
for col in SELECTED_STATS:
    if col in df.columns:
        df[col] = pd.to_numeric(df[col], errors='coerce')
```

Bước 3: Tìm top 3 cầu thủ cao nhất và thấp nhất từng chỉ số.

- Lọc ra top N cầu thủ có chỉ số cao nhất và thấp nhất cho từng thống kê đã chọn.

```
top_highest = df_col_cleaned.nlargest(TOP_N, col)
f.write(f"Top {TOP_N} Highest:\n")
f.write(format_player_list(top_highest[col], col))
top_lowest = df_col_cleaned.nsmallest(TOP_N, col)
f.write(f"\nTop {TOP_N} Lowest:\n")
f.write(format_player_list(top_lowest[col], col))
f.write("\n" + "="*30 + "\n\n")
```

-> Lệnh này lấy ra top 3 cầu thủ có chỉ số cao nhất/thấp nhất trong từng cột.

- In ra file **top_3.txt** theo cấu trúc dễ đọc:

```
with open(OUTPUT_TOP_PLAYERS_FILE, 'w', encoding='utf-8') as f:
```

-> Ghi kết quả từng chỉ số vào file text.

Bước 4: Tính thống kê mô tả (trung vị, trung bình, độ lệch chuẩn)

- Tính các thông số mô tả cơ bản cho từng chỉ số: Median (Trung vị), Mean (Trung bình), Standard Deviation (Độ lệch chuẩn).

```
overall_median = df[SELECTED_STATS].median()
overall_mean = df[SELECTED_STATS].mean()
overall_std = df[SELECTED_STATS].std()
```

-> Tính cho toàn bộ giải.

Ta tiếp tục tính cho từng đội:

```
grouped_by_team = df.groupby('Squad')[SELECTED_STATS]
team_median = grouped_by_team.median()
team_mean = grouped_by_team.mean()
team_std = grouped_by_team.std()
```

-> Gom nhóm theo đội, tính thông số tương ứng.

Bước 5: Lưu kết quả thống kê vào file CSV.

- Gộp các kết quả tính toán lại thành DataFrame và lưu vào **results2.csv** để dùng cho bước sau hoặc tham khảo.

```
results_df.to_csv(OUTPUT_STATS_FILE, encoding='utf-8')
```

Bước 6: Vẽ biểu đồ phân phối.

- Để trực quan hoá phân phối từng chỉ số:

+ Vẽ histogram cho toàn giải.

+ Vẽ histogram riêng cho từng đội.

```
plt.style.use('ggplot')
plt.figure(figsize=(10, 6))
df[col].dropna().hist(bins=20)
plt.title(f'Distribution of {col} (All Players)')
plt.xlabel(col)
plt.ylabel('Frequency')
plt.tight_layout()
try:
    plt.savefig(os.path.join(OUTPUT_HISTOGRAM_DIR, f'{col}_all_players.png'))
```

-> Tạo biểu đồ cho từng chỉ số toàn giải.

- Vẽ tiếp cho từng đội:

```
plt.figure(figsize=(8, 5))
team_data = df[df['Squad'] == team][col].dropna()
if not team_data.empty:
    team_data.hist(bins=15)
    plt.title(f'Distribution of {col} ({team})')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.tight_layout()
```

-> Tạo histogram riêng cho từng đội, từng chỉ số.

Bước 7: Xác định đội dẫn đầu từng chỉ số.

- Tìm đội có trung bình cao nhất cho từng chỉ số, để biết đội nào mạnh ở mục nào.

```
top_team = teams_only_df[mean_col_name].idxmax()
top_score = teams_only_df[mean_col_name].max()
top_teams[col] = (top_team, top_score)
```

-> Lấy tên đội dẫn đầu và giá trị trung bình cao nhất.

Bước 8: Đánh giá tổng thể.

- Đếm số lần mỗi đội dẫn đầu chỉ số để xác định đội thi đấu nổi bật nhất.

```
team_mentions = pd.Series([team for team, score in top_teams.values()]).value_counts()
print("  Teams mentioned most often as highest scorer:")
print(team_mentions.head())

top_scorer_team = top_teams.get('Gls', ('N/A', 0))[0]
top_assist_team = top_teams.get('Ast', ('N/A', 0))[0]
top_xg_team = top_teams.get('xG', ('N/A', 0))[0]

print(f"\n  Based on key metrics:")
print(f"  - Team with highest avg Goals (Gls): {top_scorer_team}")
print(f"  - Team with highest avg Assists (Ast): {top_assist_team}")
print(f"  - Team with highest avg Expected Goals (xG): {top_xg_team}")

best_performing_team = team_mentions.idxmax() if not team_mentions.empty else "Undetermined"
print(f"\n  Conclusion: Based on the analyzed statistics (especially {', '.join(SELECTED_STATS)}),")
print(f"  '{best_performing_team}' appears to be performing strongly, frequently leading in average statistics.")
print(f"  However, a comprehensive analysis would require more stats and context.")
```

Bài 3: Phân tích toàn bộ các chỉ số số học.

Bước 1: Đọc dữ liệu và xác định cột số hợp lệ.

- Đọc file **ketqua.csv**. Sau đó, kiểm tra và lọc ra những cột chứa dữ liệu dạng số hợp lệ để phục vụ cho các phân tích thống kê.

```
try:
    df = pd.read_csv(INPUT_CSV_FILE, encoding='utf-8')
except FileNotFoundError:
    print(f"Error: File '{INPUT_CSV_FILE}' not found. Please make sure it's in the same directory.")
    exit()
except Exception as e:
    print(f"Error loading CSV: {e}")
    exit()

numeric_stat_cols = get_numeric_columns(df.copy())
for col in numeric_stat_cols:
    df[col] = pd.to_numeric(df[col], errors='coerce')
```

-> Hàm **get_numeric_columns()** loại bỏ các cột chứa chữ, lọc giữ lại các cột số có giá trị hợp lệ.

Bước 2: Tìm Top 3 cầu thủ cao nhất và thấp nhất ở từng chỉ số.

- Duyệt từng cột sô, lọc ra top 3 cầu thủ có giá trị cao nhất và thấp nhất rồi ghi vào file **top_3.txt**.

```
top_highest = df_col_cleaned.nlargest(TOP_N, col)
f.write(f"Top {TOP_N} Highest:\n")
f.write(format_player_list(top_highest[col], col))
top_lowest = df_col_cleaned.nsmallest(TOP_N, col)
f.write(f"\nTop {TOP_N} Lowest:\n")
f.write(format_player_list(top_lowest[col], col))
f.write("\n" + "="*30 + "\n\n")
```

-> Phần này thực hiện lần lượt với tất cả các chỉ số dạng số.

Bước 3: Tính thống kê mô tả.

- Tính các thông số trung vị, trung bình và độ lệch chuẩn cho từng chỉ số số học - tính cho cả giải và theo từng đội.

```
overall_median = df[numeric_stat_cols].median()
overall_mean = df[numeric_stat_cols].mean()
overall_std = df[numeric_stat_cols].std()

grouped_by_team = df.groupby('Squad')[numeric_stat_cols]
team_median = grouped_by_team.median()
team_mean = grouped_by_team.mean()
team_std = grouped_by_team.std()
```

-> Kết quả được lưu vào file **results2.csv**.

Bước 4: Vẽ biểu đồ phân phối từng chỉ số.

- Vẽ histogram trực quan hoá phân phối giá trị của từng chỉ số:

- Với toàn bộ cầu thủ
- Với từng đội riêng biệt

```
for col in numeric_stat_cols:
    plt.style.use('ggplot')
    plt.figure(figsize=(10, 6))
    df[col].dropna().hist(bins=20)
    plt.title(f'Distribution of {col} (All Players)')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.tight_layout()
    try:
        plt.savefig(os.path.join(OUTPUT_HISTOGRAM_DIR, f'{col}_all_players.png'))
    except Exception as e:
        print(f"    Error saving plot for {col} (all players): {e}")
    plt.close()

    teams = df['Squad'].unique()
    for team in teams:
        plt.figure(figsize=(8, 5))
        team_data = df[df['Squad'] == team][col].dropna()
        if not team_data.empty:
            team_data.hist(bins=15)
            plt.title(f'Distribution of {col} ({team})')
            plt.xlabel(col)
            plt.ylabel('Frequency')
            plt.tight_layout()
            try:
                safe_team_name = "".join(c if c.isalnum() else "_" for c in team)
                plt.savefig(os.path.join(team_hist_dir, f'{col}_{safe_team_name}.png'))
            except Exception as e:
                print(f"    Error saving plot for {col} ({team}): {e}")
        plt.close()
```

-> Tự động lưu vào thư mục **histograms/ và histograms/teams/**

Bước 5: Tìm đội dẫn đầu từng chỉ số tấn công quan trọng.

- Với các chỉ số tấn công chính (Gls, Ast, xG, xAG, PrgC, PrgP), tìm đội có trung bình cao nhất.

```
top_team = teams_only_df[mean_col_name].idxmax()
top_score = teams_only_df[mean_col_name].max()
top_teams[col] = (top_team, top_score)
```

Bước 6: Thực hiện phân cụm K-means.

- Sử dụng K-means để phân cụm cầu thủ dựa trên các chỉ số số học:

  - Loại bỏ các cột không cần thiết

- Chuẩn hóa dữ liệu
- Dùng Elbow Method và Silhouette Score để xác định số cụm tối ưu

```
os.makedirs(CLUSTERING_RESULTS_DIR, exist_ok=True)
exclude_cols = ['Player', 'Nation', 'Squad', 'Pos', 'Age']
features = df.drop(columns=exclude_cols).apply(pd.to_numeric, errors='coerce')
cols_all_nan = features.columns[features.isnull().all()]
features = features.drop(columns=cols_all_nan)
imputer = SimpleImputer(strategy='median')
features_imputed = pd.DataFrame(imputer.fit_transform(features), columns=features.columns)
scaler = StandardScaler()
features_scaled = pd.DataFrame(scaler.fit_transform(features_imputed), columns=features_imputed.columns)

Determine_Optimal_K(features_scaled)
```

-> Vẽ biểu đồ Elbow và Silhouette để chọn số cụm tối ưu.

Bước 7: Áp dụng phân cụm K-means.

- Dùng số cụm tối ưu (ở đây giả định là K=6) để gán nhãn cụm cho từng cầu thủ.

```
optimal_k = 6
cluster_labels = Apply_K_means(features_scaled, optimal_k)
df['Cluster'] = cluster_labels
features_imputed['Cluster'] = cluster_labels
```

-> Thêm cột **Cluster** vào bảng dữ liệu.

Bước 8: Giảm chiều dữ liệu bằng PCA và trực quan hoá.

- Dùng PCA giảm số chiều xuống 2 để vẽ biểu đồ phân cụm cầu thủ theo các cụm vừa phân tích.

```
pca_df = Apply_PCA(features_scaled, cluster_labels)
Plot_2D_Cluster(pca_df, optimal_k)
```

-> Lưu biểu đồ PCA vào thư mục **clustering_results/**

Bước 9: Xuất thống kê và nhận xét kết quả phân cụm.

- Tính trung bình các chỉ số tấn công chính cho từng cụm và xuất nhận xét phân tích vào **file clustering_comments.txt**

```
features_imputed['Cluster'] = cluster_labels

print("\nMean of key statistics per cluster:")
print(features_imputed.groupby('Cluster')[KEY_TEAM_STATS].mean())
```

Ghi chú nhận xét vào file:

```
with open(os.path.join(CLUSTERING_RESULTS_DIR, "clustering_comments.txt"), "w", encoding="utf-8") as f:
    f.write("\n--- Comments on Clustering Results ---\n")
    f.write(f"1. Number of Groups (K):\n")
    f.write(f"   - Based on the Elbow Method and Silhouette Score analysis, K={optimal_k} clusters were chosen.\n")
    f.write(f"   - The Elbow plot likely showed diminishing returns in WCSS reduction around K={optimal_k}.\n")
    f.write(f"   - The Silhouette Score plot might have indicated a peak or high value at K={optimal_k}, suggesting reasonable cluster separation.\n
    f.write(f"\n2. PCA and Clustering Plot:\n")
    f.write(f"   - PCA was used to reduce the features to 2 dimensions for visualization.\n")
    f.write(f"   - The 2D scatter plot shows the distribution of players based on these two principal components, colored by their assigned K-means
    f.write(f"   - Interpretation of the plot:\n")
    f.write(f"     - Observe the separation between clusters. Are they distinct or overlapping?\n")
    f.write(f"     - The spread and density of points within each cluster provide insight into the similarity of players grouped together based on t
```

-> Bao gồm nhận xét về số cụm, biểu đồ PCA và vị trí các cầu thủ trong từng cụm.

Bài 4: Dự đoán giá trị chuyển nhượng cầu thủ bằng Machine Learning.

Bước 1: Đọc dữ liệu và lọc cầu thủ thi đấu trên 900 phút.

- Đọc dữ liệu từ file **ketqua.csv**, chuyển số phút thành số thực và chỉ giữ cầu thủ có số phút thi đấu lớn hơn 900.

```
df = pd.read_csv(INPUT_CSV, encoding='utf-8-sig')
```

```
df['Min'] = pd.to_numeric(df['Min'].astype(str).str.replace(',', ''), errors='coerce')
df = df.dropna(subset=['Min'])

players_900 = df[df['Min'] > 900][['Player', team_col, 'Min']].rename(columns={team_col: 'Team'})
```

-> Chỉ lọc những cầu thủ đáng tin cậy cho dự đoán (thi đấu đủ nhiều).

Bước 2: Lấy giá trị chuyển nhượng của từng cầu thủ.

- Dùng Selenium vào **Transfermarkt** và **FootballTransfers** để lấy giá trị chuyển nhượng. Nếu trang đầu tiên lỗi, chuyển sang trang thứ hai. Nếu cả hai thất bại thì gán **'N/a'**.

```
for attempt in range(3):
    try:
        print(f"Attempting for {player}")
        driver.get(f"{TRANSFERMARKT_URL}{player.replace(' ', '+')}")
        link = WebDriverWait(driver, 20).until(
            EC.element_to_be_clickable((By.CSS_SELECTOR, "td.hauptlink a[href*='/profil/spieler/']"))
        )
        player_url = link.get_attribute('href')
        driver.get(player_url)
        transfer_value = WebDriverWait(driver, 20).until(
            EC.presence_of_element_located((By.CSS_SELECTOR, "a[href*='/marktwertverlauf/spieler/']"))
        ).text
        print(f"Scraped {player}: {transfer_value}")
        break
    except Exception as e:
        time.sleep(5)

if not transfer_value:
    for attempt in range(3):
        try:
            print(f"Attempting {player}")
            driver.get(f"{FOOTBALLTRANSFERS_URL}{player.replace(' ', '+')}")
            transfer_value = WebDriverWait(driver, 20).until(
                EC.presence_of_element_located((By.CSS_SELECTOR, "div.market-value"))
            ).text
            print(f"Scraped {player}: {transfer_value}")
            break
        except Exception as e:
            time.sleep(5)
    else:
        transfer_value = 'N/a'
```

-> Crawl dữ liệu web theo từng cầu thủ.

Kết quả được lưu vào file **transfer_values.csv**

Bước 3: Gộp dữ liệu giá trị với bảng cầu thủ.

- Ghép giá trị chuyển nhượng lấy được với danh sách cầu thủ theo tên.

```
merged_data = pd.merge(players_900, transfer_data, left_on='Player', right_on='player_name', how='left')
```

Bước 4: Chuyển giá trị tiền về dạng số.

- Giá trị chuyển nhượng thu được có thể ở dạng “€?m” hoặc “£?k”, cần chuyển thành số.

```
def clean_transfer_value(value):
    if value == 'N/a' or pd.isna(value):
        return np.nan
    try:
        value = value.replace('€', '').replace('£', '').strip()
        if 'm' in value.lower():
            return float(value.lower().replace('m', '')) * 1e6
        elif 'k' in value.lower():
            return float(value.lower().replace('k', '')) * 1e3
        return float(value)
    except:
        return np.nan
```

-> Chuyển hoá giá trị về dạng số để đưa vào model.

Bước 5: Xác định các đặc trưng (feature) cho dự đoán.

- Chọn các chỉ số thống kê chuyên môn phù hợp để làm input cho model dự đoán giá trị cầu thủ.

```
features = ['Age', 'Min', 'Gls', 'Ast', 'xG', 'xAG', 'PrgC', 'PrgP', 'PrgR', 'SoT%', 'SoT/90', 'Tkl', 'TklW', 'Blocks', 'Touches', 'Succ%', 'Fls', 'Fld', 'Won%']
```

Và lọc những cột có đủ dữ liệu:

```
available_features = get_numeric_columns(df, features)
```

Bước 6: Chuẩn hoá dữ liệu.

- Dữ liệu số học phải chuẩn hoá về cùng thang đo (trung bình = 0, độ lệch chuẩn = 1) để model hoạt động ổn định hơn.

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Bước 7: Chia dữ liệu train - test.

- Chia ngẫu nhiên dữ liệu thành 80% train và 20% test để kiểm tra độ chính xác dự đoán.

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

Bước 8: Huấn luyện mô hình Ramdom Forest.

- Dùng **RamdomForestRegressor** - một thuật toán machine learning mạnh, ít cần tinh chỉnh, dự đoán tốt với dữ liệu nhỏ và vừa.

```
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Bước 9: Dự đoán và tính sai số.

- Dùng model để dự đoán giá trị cầu thủ, kiểm tra độ chính xác bằng sai số trung bình **MSE** và hệ số **R^2**.

```
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"MSE: {mse:.2f}, R^2: {r2:.2f}")
```

Bước 10: Dự đoán toàn bộ và lưu kết quả.

- Dự đoán giá trị chuyển nhượng cho toàn bộ cầu thủ và lưu vào file **transfer_predictions.csv**.

```python
full_pred = model.predict(X_scaled)

predictions = pd.DataFrame({
    'Player': players.values,
    'Actual_Value': y.values,
    'Predicted_Value': full_pred
})
predictions.to_csv(PREDICT_CSV, index=False, encoding='utf-8-sig')
```