HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN I

o0o

BÁO CÁO BÀI TẬP LỚN 1

NGÔN NGỮ LẬP TRÌNH PYTHON

Giảng viên hướng dẫn:	**Kim Ngọc Bách**
Sinh viên:	**Lê Như Quỳnh**
Mã sinh viên:	**B23DCCE081**
Lớp:	**D23CQCEO6-B**
Niên khóa:	**2023 - 2028**
Hệ đào tạo:	**Đại học chính quy**

Hà Nội, 2025

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN I

o0o

BÁO CÁO BÀI TẬP LỚN 1

NGÔN NGỮ LẬP TRÌNH PYTHON

Giảng viên hướng dẫn: Kim Ngọc Bách
Sinh viên: Lê Như Quỳnh
Mã sinh viên: B23DCCE081
Lớp: D23CQCEO6-B
Niên khóa: 2023 - 2028
Hệ đào tạo: Đại học chính quy

Hà Nội, 2025

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Điểm: **(Bằng chữ:)**

Hà Nội, ngày tháng năm 20...

Giảng viên

Mục lục

Danh sách hình vẽ

MỞ ĐẦU

Trong thời đại dữ liệu bùng nổ, khả năng xử lý và phân tích dữ liệu là một kỹ năng quan trọng đối với bất kỳ lập trình viên hay nhà khoa học dữ liệu nào. Ngôn ngữ `Python`, với hệ sinh thái thư viện phong phú, cho phép người học tiếp cận dễ dàng với các bài toán thực tế, đặc biệt trong lĩnh vực thể thao – nơi dữ liệu được thu thập, phân tích và ứng dụng mạnh mẽ.

Bài tập số 1 của học phần Lập trình `Python` đặt ra bài toán thu thập và phân tích dữ liệu cầu thủ tại giải Ngoại hạng Anh mùa giải 2024–2025. Báo cáo này trình bày quy trình xử lý toàn diện – từ thu thập dữ liệu, phân tích thống kê, trực quan hóa, đến áp dụng các thuật toán học máy. Nội dung được xây dựng xoay quanh bốn câu hỏi lớn. Tất cả quá trình thu thập dữ liệu này được thực hiện vào ngày 6 tháng 5 năm 2025. Do đó, mọi phân tích và kết quả trình bày trong báo cáo này phản ánh tình hình và số liệu của các cầu thủ tính đến thời điểm cụ thể đó của mùa giải. Nội dung cụ thể của bài báo cáo gồm các phần chính:

Chương 1: Thu thập dữ liệu cầu thủ từ fbref.com:
Chương này trình bày phương pháp thu thập dữ liệu từ trang web `fbref.com`, lọc các cầu thủ có thời gian thi đấu trên 90 phút, xử lý dữ liệu thô và lưu trữ có hệ thống vào tệp `results.csv`, sẵn sàng cho các bước phân tích tiếp theo.

Chương 2: Phân tích và Trực quan hóa Dữ liệu:
Chương này tập trung vào phân tích thống kê mô tả. Dựa trên dữ liệu thu thập được, báo cáo xác định top 3 cầu thủ có giá trị cao nhất và thấp nhất cho mỗi thống kê, tính toán các đại lượng trung bình, trung vị, độ lệch chuẩn và trình bày phân bố dữ liệu dưới dạng biểu đồ.

Chương 3: Phân cụm Cầu thủ bằng K-Means và PCA:
Từ việc tổng hợp các chỉ số theo từng đội, chương này sử dụng biểu đồ và so sánh trực tiếp để đưa ra đánh giá khách quan về hiệu suất thi đấu của các câu lạc bộ trong mùa giải, từ đó xác định đội bóng đang có phong độ tốt nhất.

Chương 4: Ước tính giá trị cầu thủ:
Chương cuối kết hợp các kỹ thuật học máy để phân cụm cầu thủ theo đặc điểm thống kê, đồng thời đề xuất một mô hình ước lượng giá trị chuyển nhượng dựa trên hiệu suất thi đấu. Việc lựa chọn đặc trưng, xây dựng mô hình và phân tích kết quả sẽ được trình bày chi tiết để chứng minh khả năng ứng dụng thực tiễn của dữ liệu.

Chương 1

Phần I: Thu thập dữ liệu cầu thủ từ Fbref.com

1.1 Phân tích yêu cầu đề bài

Phần này có yêu cầu là xây dựng một chương trình Python để tự động thu thập dữ liệu thống kê của các cầu thủ bóng đá. Các yêu cầu cụ thể bao gồm:

- **Đối tượng:** Cầu thủ thi đấu tại giải Ngoại hạng Anh (English Premier League) mùa giải 2024-2025
- **Nguồn dữ liệu:** Trang web fbref.com
- **Điều kiện lọc:** Chỉ thu thập dữ liệu cho những cầu thủ đã thi đấu nhiều hơn 90 phút
- **Dữ liệu cần thu thập:** Nation, Team, Position, Age, Playing Time(matches played, starts, minutes), Performance(goals, assists, yellow cards, red cards), Expected(xG, XAG), Progression(PrgC, PrgP, PrgR), Per 90 minutes(Gls, Ast, xG, XGA), Goalkeeping [(Performance: goals against per 90mins (GA90), Save%, CS%), Penalty Kicks: penalty kicks Save%], Shooting, Passing, Goal and Shot Creation, Defensive Actions, Possession, Miscellaneous Stats
- **Định dạng đầu ra:** Lưu kết quả vào tệp `results.csv`

Cấu trúc tệp results.csv:

- Mỗi cột tương ứng với một chỉ số thống kê theo đúng thứ tự và tên gọi như trong danh sách yêu cầu của đề bài
- Mỗi hàng đại diện cho một cầu thủ
- Các cầu thủ phải được sắp xếp theo thứ tự trong bảng chữ cái của tên (First Name)
- Các giá trị thống kê không có sẵn hoặc không áp dụng đánh dấu là "N/a"

1.2 Cấu trúc chương trình

Để thực hiện yêu cầu đề bài, em đã chia thành 3 module Python chính:

- **config.py**: Module trung tâm chứa các cấu hình, hằng số và định nghĩa ánh xạ (mapping) dữ liệu
- **scraper.py**: Module chứa logic và các hàm/lớp cần thiết cho việc thu thập dữ liệu (web scrapping)
- **main.py**: Module chính, đóng vai trò là điểm khởi chạy và điều phối hoạt động của toàn bộ chương trình

1.3 Lựa chọn thư viện

Để có kết quả như yêu cầu của đề, em đã lựa chọn các thư viện Python sau:

- **Selenium**: Em chọn Selenium vì trang Fbref.com sử dụng JavaScript để tải dữ liệu trong các bảng thống kê. Nếu chỉ dùng thư viện như requests, sẽ không lấy được đầy đủ HTML. Selenium giúp em điều khiển trình duyệt (Chrome) tự động, đợi các thành phần động tải xong rồi mới lấy mã nguồn trang.
- **Beautiful Soup (bs4)**: Sau khi Selenium lấy được HTML đầy đủ, em dùng BeautifulSoup để phân tích cấu trúc HTML này. Nó rất mạnh trong việc tìm kiếm các thẻ HTML (như `<table>`, `<tr>`, `<td>`) dựa trên các thuộc tính (ví dụ: `id = "stat_standard"`, `data-stat = "goals"`) và trích xuất nội dung text bên trong.
- **Pandas**: Thư viện này rất cần thiết ở bước cuối. Sau khi trích xuất dữ liệu thô, em dùng Pandas để tạo một DataFrame (bảng dữ liệu), đảm bảo đúng cấu trúc và thứ tự cột theo yêu cầu, sau đó dễ dàng lưu DataFrame này thành tệp `results.csv`.

1.4 Quy trình thực hiện tổng thể

Chương trình của em thực hiện việc thu thập dữ liệu theo các bước chính sau:

1. **Khởi tạo**: Bắt đầu chạy từ file `main.py`. Chương trình đọc các cài đặt cần thiết từ file `config.py` (như địa chỉ web, id các bảng, cách ánh xạ tên cột, ngưỡng lọc 90 phút).
2. **Lấy dữ liệu gốc**: Chương trình dùng `main.py` điều khiển `scraper.py` truy cập vào trang chứa bảng thống kê chính (standard) trên Fbref. Sử dụng Selenium và BeautifulSoup, nó lấy dữ liệu của tất cả cầu thủ trong bảng này.
3. **Lọc cầu thủ**: Ngay sau khi lấy dữ liệu gốc, chương trình lọc ra những cầu thủ có số phút thi đấu (minutes) lớn hơn 90. Những cầu thủ hợp lệ này được lưu lại (dưới dạng đối tượng Player).
4. **Lấy dữ liệu bổ sung**: Chương trình tiếp tục truy cập vào các trang chứa các bảng thống kê phụ khác (thủ môn, sút bóng, chuyền bóng, phòng ngự, v.v.) theo danh sách đã định nghĩa trong `config.py`.

5. **Cập nhật dữ liệu**: Với dữ liệu từ mỗi bảng phụ, chương trình tìm đến các cầu thủ đã có trong danh sách gốc (khớp theo tên) và cập nhật thêm các chỉ số mới từ bảng phụ vào thông tin của cầu thủ đó.

6. **Sắp xếp**: Sau khi đã lấy và cập nhật đủ thông tin từ tất cả các bảng, chương trình sắp xếp lại danh sách cầu thủ theo thứ tự tên (First Name).

7. **Tạo và xuất file**: Cuối cùng, chương trình dùng thư viện Pandas để tạo một bảng DataFrame từ danh sách cầu thủ đã sắp xếp, đảm bảo các cột theo đúng thứ tự yêu cầu. Thực hiện làm sạch dữ liệu số lần cuối (loại bỏ dấu phẩy) và lưu bảng này thành file `results.csv`.

1.5 Phân tích chi tiết các thành phần

1.5.1 Cấu hình và Ánh xạ dữ liệu (`config.py`)

Dây là nơi em định nghĩa tất cả các thông số và cấu trúc dữ liệu cố định để chương trình chạy đúng và dễ dàng chỉnh sửa hơn.

1. **Thông tin cơ bản:** URL, tên file, ngưỡng phút

```python
FBREF_BASE_URL = 'https://fbref.com/en/comps/9'
PL_SUFFIX = '/stats/Premier-League-Stats'
OUT_FILE = 'results.csv'
MIN_MINUTES = 90
```

Hình 1.1: Định nghĩa thông tin cơ bản trong config.py.

2. **ID và URL các bảng:** Mã nhận dạng và đường dẫn của các bảng, giúp chương trình xác định và tìm kiếm đúng bảng dữ liệu cần sử dụng.

```python
TABLE_IDS = {
    'standard': 'stats_standard',
    'keeper': 'stats_keeper',
    'shooting': 'stats_shooting',
    'passing': 'stats_passing',
    'gca': 'stats_gca',
    'defense': 'stats_defense',
    'possession': 'stats_possession',
    'misc': 'stats_misc',
}

TABLE_URLS = {
    TABLE_IDS['standard']: PL_SUFFIX,
    TABLE_IDS['keeper']: '/keepers/Premier-League-Stats',
    TABLE_IDS['shooting']: '/shooting/Premier-League-Stats',
    TABLE_IDS['passing']: '/passing/Premier-League-Stats',
    TABLE_IDS['gca']: '/gca/Premier-League-Stats',
    TABLE_IDS['defense']: '/defense/Premier-League-Stats',
    TABLE_IDS['possession']: '/possession/Premier-League-Stats',
    TABLE_IDS['misc']: '/misc/Premier-League-Stats',
}
```

Hình 1.2: Định nghĩa ID và URL các bảng trong config.py.

3. **Ánh xạ (`HEADER_MAP`):** Nối tên cột trong CSV với `data-stat` trong HTML.

```
HEADER_MAP = {
    'Player': 'player',
    'Nation': 'nationality',
    'Team': 'team',
    #...(toàn bộ ánh xạ)...
}
```

Hình 1.3: Định nghĩa ánh xạ HEADER_MAP trong config.py (đoạn trích).

4. **Thứ tự cột (`HEADER_ORDER`):** Đảm bảo file CSV có cấu trúc cột đúng yêu cầu

```
HEADER_ORDER = [
    'Player', 'Nation', 'Team',... ,
    'Miscellaneous: Aerial Duels: Won%'
]
```

Hình 1.4: Định nghĩa thứ tự cột HEADER_ORDER trong config.py (đoạn trích).

5. **Danh sách trường xuất (`EXPORT_STATS`) & Nhóm theo bảng (`STATS_BY_TABLE`):** Các biến này được tạo tự động từ `HEADER_ORDER` và `HEADER_MAP` để giúp `main.py` biết chính xác cần lấy `data-stat` nào từ `scraper.py` và từ bảng nào.

```
EXPORT_STATS = []
fetched_stats = set()
for col in HEADER_ORDER:
    stat = HEADER_MAP.get(col)
    if stat:
        EXPORT_STATS.append(stat)
        fetched_stats.add(stat)
    else:
        print(f"[ERROR] Config: Không tìm thấy mapping data-stat cho cột CSV mẫu: '{col}'.")

STATS_BY_TABLE = {
    TABLE_IDS['standard']: [
        'player', 'nationality', 'position', 'team', 'age', 'games', 'games_starts', 'minutes',
        'goals', 'assists', 'cards_yellow', 'cards_red', 'xg', 'xg_assist',
        'progressive_carries', 'progressive_passes', 'progressive_passes_received',
        'goals_per90', 'assists_per90', 'xg_per90', 'xg_assist_per90'
    ],
```

Hình 1.5: Tạo EXPORT_STATS và STATS_BY_TABLE trong config.py (đoạn trích).

1.5.2 Xử lý và Làm sạch dữ liệu

Việc xử lý dữ liệu được thực hiện ở nhiều chỗ trong chương trình:

1. **Chuyển đổi Số An toàn (`scraper.safe_cast_int`):** Hàm này được dùng khi lấy số phút và các chỉ số khác từ HTML. Nó bỏ dấu phẩy, chuyển sang int, và nếu gặp lỗi hoặc "N/a"thì trả về 0 và báo lỗi.

```
def safe_cast_int(value_str: Optional[str], default: int = 0) -> int:
    if not value_str or value_str == 'N/a':
        return default
    try:
        cleaned_str = value_str.replace(',', '')
        return int(cleaned_str)
    except (ValueError, TypeError):
        print(f"[LOI] Khong the chuyen '{value_str}' sang so nguyen.")
        return default
```

Hình 1.6: Hàm safe_cast_int trong scraper.py.

2. **Xử lý Thiếu khi Parse (`scraper.py` - logic trong các hàm):** Khi `scraper.py` đọc từng ô `<td>`, nếu không tìm thấy ô tương ứng với `data-stat` thì nó sẽ gán giá trị 'N/a'.

3. **Lọc theo Phút (`scraper.get_players_from_table`):** Chỉ những cầu thủ có `minutes > 90` mới được giữ lại sau khi lấy từ bảng 'standard'.

4. **Xử lý Thiếu khi Export (`scraper.Player.export`):** Khi xuất dữ liệu ra CSV, nếu một cầu thủ không có chỉ số nào đó (do không có ở bảng gốc hoặc bảng phụ), phương thức `export` sẽ tự động điền 'N/a' vào vị trí đó.

5. **Làm sạch Số trong DataFrame (`main.clean_numeric_commas`):** Trước khi lưu file CSV, hàm này chạy qua các cột DataFrame, tìm cột kiểu object (chuỗi), bỏ dấu phẩy và cố gắng chuyển cả cột thành kiểu số.

```python
def clean_numeric_commas(df: pd.DataFrame) -> pd.DataFrame:
    # Làm sạch dấu ',' nếu là dấu ngăn cách hàng nghìn
    for col in df.columns:
        if df[col].dtype == object:
            # Xóa dấu ',' trong các chuỗi
            df[col] = df[col].str.replace(',', '', regex=False)
            # Cố gắng chuyển sang số nếu hợp lệ
            df[col] = pd.to_numeric(df[col], errors='ignore')
    return df
```

Hình 1.7: Hàm clean_numeric_commas trong main.py.

1.5.3 Các hàm thu thập cốt lõi (`scraper.py`)

File này chứa các hàm và lớp thực hiện việc "cào"dữ liệu.

1. **Lớp Player:** Lưu dữ liệu cầu thủ bằng dictionary, có hàm `update()` và `export()`.

```python
class Player:
    def __init__(self, **kwargs: Any):
        self.data: Dict[str, Any] = kwargs

    def update(self, **kwargs: Any) -> None:
        self.data.update(kwargs)

    def export(self, export_keys: List[str]) -> List[Any]:
        return [self.data.get(key, 'N/a') for key in export_keys]

    def __repr__(self) -> str:
        name = self.data.get('player', 'Unknown Player')
        team = self.data.get('team', 'Unknown Team')
        age = self.data.get('age', 'N/A')
        return f"<Player: {name} ({age} - {team})>"
```

Hình 1.8: Định nghĩa lớp Player trong scraper.py.

2. **Hàm get_soup(...):** Dùng Selenium để mở web, đợi, cuộn trang, lấy HTML và parse bằng BeautifulSoup, có cơ chế thử lại.

```python
def get_soup(url: str, table_id: str, retries: int = 3, delay: int = 5) -> Optional[BeautifulSoup]:
    options = Options()
```

Hình 1.9: Định nghĩa hàm get_soup trong scraper.py (chỉ phần khai báo).

3. **Hàm get_players_from_table(...)**: Lấy danh sách Player ban đầu từ bảng 'standard' và lọc theo số phút.

```python
def get_players_from_table(url: str, table_id: str, fetch_fields: List[str], min_mins: int) -> List[Player]:
    print(f"[THONG TIN] Dang lay du lieu cau thu tu bang '{table_id}' tai {url}...")
    soup = get_soup(url, table_id)
```

Hình 1.10: Định nghĩa hàm get_players_from_table trong scraper.py (đoạn trích).

4. **Hàm update_players(...)**: Cập nhật thêm dữ liệu từ các bảng phụ vào danh sách Player gốc.

```python
def update_players(players: List[Player], url: str, table_id: str, update_fields: List[str]) -> None:
    print(f"[THONG TIN] Dang cap nhat du lieu tu bang '{table_id}' tai {url}...")
    soup = get_soup(url, table_id)
```

Hình 1.11: Định nghĩa hàm update_players trong scraper.py (đoạn trích).

1.5.4 Hàm chính thực hiện (`main.py`)

Hàm `main()` trong file `main.py` điều phối tất cả các bước.

```python
import pandas as pd
import time
# ... (imports khác và các hàm tiện ích)
from config import *
from scraper import *
def main():
    start_time = time.time()
    print("[INFO] Bat dau...")
    # 1. Chuẩn bị: Xác định các trường cần lấy từ config
    # ... (code xác định base_req_fields, addtl_tables) ...
    # 2. Gọi hàm lấy dữ liệu gốc
    players = get_players_from_table(...)
    # ... (Kiểm tra lỗi) ...
    # 3. Gọi hàm cập nhật lặp qua các bảng phụ
    for table_info in addtl_tables:
        update_players(...)
        print("-" * 30)
    # 4. Sắp xếp
    sorted_players = sorted(players, ...)
    # 5. Tạo DataFrame
    data_to_export = [p.export(EXPORT_STATS) for p in sorted_players]
    df = pd.DataFrame(data_to_export, columns=HEADER_ORDER)
    # 6. Làm sạch DataFrame
    df = clean_numeric_commas(df)
    # 7. Lưu CSV
    try:
        df.to_csv(OUT_FILE, index=False, encoding='utf-8-sig')
        print(f"[SUCCESS] Luu file {OUT_FILE} thanh cong!")
        # ... (In kết quả ra console) ...
    except Exception as e:
        print(f"[ERROR] Loi khi luu CSV: {e}")
if __name__ == '__main__':
    main()
```

Hình 1.12: Hàm main trong main.py.

1.6 Kết quả

Sau khi chạy thành công file `main.py`, em thu được tệp kết quả chính của Bài 1 là `results.csv`. Tệp này chứa dữ liệu của 494 cầu thủ đã thi đấu trên 90 phút, kèm đầy đủ các thống kê theo yêu cầu.
Dưới đây là một đoạn mẫu từ file results.csv khi mở bằng Excel:

Player	Nation	Team	Position	Age	Playing Time: matches played	Playing Time: starts	Playing Time: minutes
Aaron Cresswell	eng ENG	West Ham	DF	35-144	15	8	676
Aaron Ramsdale	eng ENG	Southampton	GK	26-359	27	27	2430
Aaron Wan-Bissaka	eng ENG	West Ham	DF	27-163	33	32	2884
Abdoulaye Doucouré	ml MLI	Everton	MF	32-127	30	29	2425
Abdukodir Khusanov	uz UZB	Manchester City	DF	21-068	6	6	503
Abdul Fatawu Issahaku	gh GHA	Leicester City	FW	21-061	11	6	579
Adam Armstrong	eng ENG	Southampton	FWMF	28-087	20	15	1248
Adam Lallana	eng ENG	Southampton	MF	36-363	14	5	361
Adam Smith	eng ENG	Bournemouth	DF	34-009	22	17	1409
Adam Webster	eng ENG	Brighton	DF	30-124	11	8	617
Adam Wharton	eng ENG	Crystal Palace	MF	20-340	20	16	1318
Adama Traoré	es ESP	Fulham	FWMF	29-103	33	16	1592
Albert Grønbaek	dk DEN	Southampton	FWMF	23-350	4	2	143
Alejandro Garnacho	ar ARG	Manchester Utd	MFFW	20-311	34	23	2146

Hình 1.13: Đoạn mẫu từ file results.csv.

Chương 2

Phần II: Phân tích và Trực quan hóa dữ liệu

Sau khi đã hoàn thành việc thu thập và tạo ra file results.csv ở Phần I, Bài tập 2 chuyển sang các bước phân tích và trực quan hóa bộ dữ liệu. Nội dung này nhằm giúp làm rõ đặc điểm của dữ liệu, làm rõ các cầu thủ và đội bóng nổi bật, đồng thời xem xét sự phân phối của một số chỉ số quan trọng.

2.1 Cấu trúc chương trình Chương 2

Tương tự Phần I, phần II cũng chia thành các module Python chính:

- **main2.py**: Đọc results.csv, tìm 3 cầu thủ cao nhất và thấp nhất cho mỗi chỉ số số liệu, ghi kết quả vào top_3.txt.
- **part_2.py**: Đọc results.csv, tính toán Trung vị (Median), Trung bình (Mean), Độ lệch chuẩn (Std Dev) cho mỗi chỉ số theo toàn giải ('all') và theo từng đội, lưu kết quả vào results2.csv.
- **part_3.py**: Đọc results.csv, vẽ biểu đồ histogram (phân phối tổng thể và theo từng đội) cho một số chỉ số được chọn (selected_stats), lưu các biểu đồ thành file ảnh.
- **part_4.py**: Đọc results2.csv, phân tích để tìm ra đội bóng dẫn đầu ở nhiều chỉ số trung bình nhất, và in kết quả đánh giá này ra màn hình.

2.2 Lựa chọn thư viện

Để thực hiện các yêu cầu phân tích và trực quan hóa trong Bài 2, em đã sử dụng các thư viện Python chính sau:

- **Pandas**: Em chọn thư viện này vì nó là công cụ mạnh để làm việc với dữ liệu dạng bảng. Trong Bài 2, Pandas giúp em đọc các file CSV (results.csv và results2.csv), làm sạch và chuẩn bị dữ liệu (như chuyển đổi kiểu số, xử lý dấu phẩy), lựa chọn cột, lọc bỏ giá trị thiếu (dropna), sắp xếp dữ liệu để tìm top/bottom (sort_values), nhóm dữ liệu theo đội (groupby) và tính toán các thống kê tổng hợp (agg), cuối cùng là tạo DataFrame kết quả (results2.csv).

- **NumPy**: Thư viện này là nền tảng cho Pandas, hỗ trợ mạnh mẽ các tính toán số học. Em dùng nó (thường là thông qua Pandas) để xử lý các giá trị NaN và thực hiện các phép tính toán trên các cột số liệu một cách hiệu quả.

- **Matplotlib (pyplot)**: Em chọn thư viện này vì nó là tiêu chuẩn để vẽ đồ thị trong Python. Trong part_3.py, em dùng Matplotlib để thiết lập các thông số cơ bản của biểu đồ (kích thước, tiêu đề, nhãn trục) và để lưu các biểu đồ histogram thành file ảnh.

- **Seaborn**: Em dùng Seaborn vì nó được xây dựng trên Matplotlib nhưng cung cấp các hàm vẽ đồ thị thống kê đẹp và đơn giản hơn. Cụ thể trong part_3.py, em dùng seaborn.histplot để vẽ biểu đồ phân phối tần suất và seaborn.FacetGrid để dễ dàng tạo ra một lưới các biểu đồ histogram cho từng đội bóng.

- **OS**: Thư viện này giúp em tương tác với hệ điều hành, chủ yếu là để quản lý đường dẫn file. Em dùng os.path.join, os.path.dirname để tạo đường dẫn chính xác đến các file dữ liệu đầu vào (results.csv, results2.csv) và thư mục/tệp đầu ra (top_3.txt, thư mục chứa histograms) một cách linh hoạt, không phụ thuộc vào việc chạy code trên máy nào.

- **Các thư viện phụ trợ khác**: Ngoài ra, em có dùng traceback để giúp hiển thị lỗi chi tiết khi chương trình gặp sự cố, re (Regular Expressions) và math cho một số tiện ích nhỏ trong part_3.py (làm sạch tên file, tính toán layout), và typing để code rõ ràng hơn.

2.3 Quy trình thực hiện tổng thể

Để phân tích và trực quan hóa dữ liệu từ file results.csv (kết quả Bài 1), em đã thực hiện các bước công việc chính sau đây, tương ứng với các script Python khác nhau:

1. **Chuẩn bị dữ liệu nền tảng**: Bước đầu tiên cho mọi phân tích là đọc file results.csv bằng Pandas, xác định các cột chứa dữ liệu số và thực hiện các bước làm sạch cơ bản như loại bỏ dấu phẩy, xử lý giá trị 'N/a' thành NaN, và chuyển đổi các cột này sang kiểu dữ liệu số (numeric) để có thể tính toán.

2. **Tìm cầu thủ nổi bật (Top/Bottom 3)**: Em chạy script `main2.py`. Script này sẽ lặp qua từng cột số liệu đã chuẩn bị, loại bỏ các giá trị NaN, sau đó sắp xếp để tìm ra 3 cầu thủ có giá trị cao nhất và 3 cầu thủ có giá trị thấp nhất. Kết quả của bước này được ghi chi tiết vào file `top_3.txt`.

3. **Tính toán Thống kê Mô tả**: Em chạy script `part_2.py`. Script này tính toán các giá trị thống kê quan trọng là Trung vị (Median), Trung bình (Mean), và Độ lệch chuẩn (Standard Deviation) cho từng chỉ số. Việc tính toán này được thực hiện ở hai cấp độ: một là cho toàn bộ cầu thủ trong giải (tạo ra hàng 'all'), hai là tính riêng cho từng đội bóng (dùng groupby). Kết quả cuối cùng được tổng hợp và lưu vào file `results2.csv` với cấu trúc cột rõ ràng.

4. **Vẽ Biểu đồ Phân phối**: Em chạy script `part_3.py`. Script này tập trung vào việc vẽ biểu đồ histogram bằng Seaborn và Matplotlib cho một số chỉ số tấn công và phòng ngự/kiến thiết tiêu biểu mà em đã chọn lọc (selected_stats). Nó vẽ biểu

đồ phân phối tổng thể cho mỗi chỉ số, và quan trọng hơn là vẽ biểu đồ dạng lưới (FacetGrid) để so sánh phân phối của chỉ số đó giữa các đội bóng khác nhau. Các biểu đồ này được lưu dưới dạng các file ảnh PNG.

5. **Đánh giá Sơ bộ Hiệu suất Đội bóng**: Cuối cùng, em chạy script `part_4.py`. Script này đọc file `results2.csv` (kết quả của bước 3), tập trung vào các cột giá trị Trung bình (Mean) của từng đội. Nó tìm ra đội có giá trị trung bình cao nhất cho mỗi chỉ số (hoặc thấp nhất cho chỉ số tiêu cực như GA90), rồi đếm xem đội nào dẫn đầu nhiều chỉ số nhất. Kết quả và kết luận sơ bộ về đội có hiệu suất tốt nhất theo tiêu chí này được in ra màn hình console.

2.4 Phân tích chi tiết các thành phần

2.4.1 Chuẩn bị dữ liệu phân tích

Đây là bước nền tảng được thực hiện ở đầu các script (main2.py, part_2.py, part_3.py) để đảm bảo dữ liệu sẵn sàng cho phân tích.

- **Đọc và làm sạch cơ bản**: Em dùng Pandas đọc file results.csv từ Bài 1, tự động nhận diện 'N/a' là giá trị thiếu (NaN), và làm sạch khoảng trắng thừa ở tên cột và tên đội.
- **Xác định và chuẩn hóa cột số**: Em lọc ra các cột chứa chỉ số thống kê (loại bỏ cột định danh). Với các cột này, em thực hiện:
  - Loại bỏ dấu phẩy trong các số lớn (`.str.replace(',', '')`).
  - Ép kiểu sang dạng số bằng `pd.to_numeric(errors='coerce')`. Các giá trị lỗi sẽ thành NaN.
  - Lưu tên các cột số hợp lệ vào danh sách (stat_columns hoặc numeric_cols).

```
stat_columns = []
for col in df.columns:
    if col in non_stat_cols:
        continue

    # Loại bỏ dấu ',' rồi chuyển sang kiểu số
    clean_col = df[col].astype(str).str.replace(',', '', regex=False)
    numeric_col = pd.to_numeric(clean_col, errors='coerce')

    if not numeric_col.isnull().all():
        df[col] = numeric_col
        stat_columns.append(col)
```

Hình 2.1: Trích đoạn mã chuẩn hóa cột số.

2.4.2 Tìm Top/Bottom 3 cầu thủ (`main2.py`)

Phần này em tìm ra 3 cầu thủ xuất sắc nhất và yếu nhất cho mỗi chỉ số. **Thực hiện**:

1. Import thư viện (pandas, numpy, os, traceback). Xác định đường dẫn file input (results.csv) và output (top_3.txt).
2. Đọc và chuẩn bị dữ liệu như mục 2.4.1.
3. Mở file `top_3.txt` để ghi.
4. Vòng lặp chính: Lặp qua từng `stat` trong `stat_columns`.
   - Ghi tên chỉ số.
   - Tạo `df_stat` chỉ chứa 'Player' và `stat`, loại bỏ NaN (`.dropna()`).
   - Kiểm tra và xử lý nếu dữ liệu rỗng hoặc chỉ có 1 giá trị duy nhất (`nunique() == 1`).
   - Dùng `sort_values()` và `head(3)` để lấy top 3 (cao nhất) và bottom 3 (thấp nhất).
   - Ghi kết quả (Tên, Giá trị) vào file.

```
# --- Buoc 4: Xu ly top 3 va ghi ra file ---
try:
    with open(output_path, 'w', encoding='utf-8') as f:
        print("Dang xu ly va ghi top 3...")
        for stat in stat_columns:
            f.write(f"Chi so: {stat}\n")
            df_stat = df[[player_column, stat]].dropna()
            if df_stat.empty:
                f.write("Khong co du lieu hop le\n\n")
                continue
            if df_stat[stat].nunique() == 1:
                f.write(f"Tat ca cau thu cung gia tri: {df_stat[stat].iloc[0]}\n")
                top_3 = bottom_3 = df_stat.head(3)
            else:
                top_3 = df_stat.sort_values(by=stat, ascending=False).head(3)
                bottom_3 = df_stat.sort_values(by=stat, ascending=True).head(3)
            f.write("3 cau thu diem cao nhat:\n")
            for _, row in top_3.iterrows():
                f.write(f"   {row[player_column]}: {row[stat]}\n")
            f.write("3 cau thu diem thap nhat:\n")
            for _, row in bottom_3.iterrows():
                f.write(f"   {row[player_column]}: {row[stat]}\n")
            f.write("\n")
    print(f"Hoan thanh. Da luu ket qua vao: {output_path}"
except Exception as e:
    print("Loi khi ghi file:")
    traceback.print_exc()
```

Hình 2.2: Trích đoạn mã xử lý và ghi Top/Bottom 3 trong main2.py.

Kết quả: Sau khi chạy xong chương trình, kết quả được lưu vào file `top_3.txt`.

```
Chi so: Playing Time: matches played
3 cau thu diem cao nhat:
Tyrick Mitchell: 35
David Raya: 35
Dean Henderson: 35
3 cau thu diem thap nhat:
Ayden Heaven: 2
Ben Godfrey: 2
Billy Gilmour: 2

Chi so: Playing Time: starts
3 cau thu diem cao nhat:
Youri Tielemans: 35
Tyrick Mitchell: 35
Dean Henderson: 35
3 cau thu diem thap nhat:
Ben Chilwell: 0
Sergio Reguilón: 0
Solly March: 0
...
```

2.4.3 Tính toán thống kê mô tả (Median, Mean, Std Dev) - part_2.py

Mục tiêu là tính toán trung vị (median), trung bình (mean) và độ lệch chuẩn cho từng chỉ số, áp dụng cho toàn bộ cầu thủ và cho từng đội.
Thực hiện:

- **Xác định Cột Số (`numeric_cols`)**:Trong bước chuẩn bị dữ liệu (Bước 2 trong file), script thực hiện việc nhận diện và chuyển đổi kiểu dữ liệu cho các cột quan trọng sang dạng số. Quá trình này có kèm theo lệnh in thông tin DEBUG để ghi nhận trạng thái và kết quả của từng bước chuyển đổi.

- **Tính toán**:

  - *Overall*: Tính các thống kê tổng thể (median, mean, std) cho cột số bằng `df[numeric_cols].agg([...]).T`. Phương thức `.agg()` tính đồng thời 3 giá trị, `.T` chuyển vị để mỗi hàng là một chỉ số.
  - *Per Team*: Nhóm dữ liệu theo `'Team'` rồi dùng `.agg([...],numeric_only=True)` để tính median, mean, std cho các cột số. Kết quả là `team_stats`, một DataFrame có `MultiIndex`.

- **Tạo DataFrame Kết quả**:

  - `all_row_df`: Đoạn code này tạo một dictionary `all_row_data`, sau đó lặp qua `numeric_cols`. Với mỗi col, nó tạo tên cột mới (ví dụ `f'Median of {col}'`) và lấy giá trị tương ứng từ `overall_stats.loc[col, 'median']` (hoặc 'mean', 'std'). Cuối cùng tạo DataFrame từ dictionary này.

- `team_results_df`: Đoạn code này phức tạp hơn do phải xử lý MultiIndex từ `team_stats`. Lặp qua các cột số (`numeric_cols`), tạo key `MultiIndex` để truy cập cột tương ứng trong `team_stats`. Dùng `reindex(team_names).values` để lấy mảng giá trị theo đúng thứ tự đội, gán vào `team_results_data` (xử lý key thiếu bằng `pd.NA`). Cuối cùng, tạo `DataFrame` từ `dictionary` `team_results_data`.
- `final_results = pd.concat(...)`: Ghép `all_row_df` và `team_results_df` theo chiều dọc.

- **Lưu file**: Sử dụng `final_results.to_csv(STATS_SUMMARY_FILE, ...)` để lưu kết quả. Tham số `float_format='%.3f'` giúp làm tròn số thập phân.

Kết quả: Sau khi chạy, kết quả được lưu vào file `results2.csv`.

Team	Median of Playing Time: matches played	Mean of Playing Time: matches played	Std of Playing Time: matches played
all	23	21.223	9.976
Arsenal	23.5	23.273	8.066
Aston Villa	20.5	19.429	10.333
Bournemou	26	22.217	10.18
Brentford	28	23.524	11.383
Brighton	21	19.536	10.035
Chelsea	18	19.692	11.263
Crystal Pala	30	24.048	10.632
Everton	24	22.409	9.287
Fulham	27	24.5	9.329
Ipswich Tov	18.5	18.2	9.015
Leicester Ci	21.5	20.231	9.717
Liverpool	28	25.19	8.925
Manchester	23	19.88	8.974
Manchester	20	17.667	11.454
Newcastle U	27	23.217	10.144
Nott'ham Fo	30	24.5	10.888
Southamptc	20	18.655	10.379
Tottenham	22	19.333	9.232
West Ham	20	21.56	8.53
Wolves	26	22.783	8.924

Hình 2.3: Đoạn mẫu từ file results2.csv

2.4.4 Trực quan hóa phân phối (`part_3.py`)

Ở phần này em vẽ biểu đồ histogram. **Lựa chọn Chỉ số**: Em đã chọn 6 chỉ số sau để vẽ biểu đồ histogram, đại diện cho các khía cạnh quan trọng của tấn công và phát triển lối chơi:

```
selected_stats = [
    'Expected: xG',
    'Per 90 minutes: Gls',
    'Shooting: Standard: SoT%',
    'Performance: assists',
    'Passing: Expected: KP',
    'Passing: Expected: PrgP'
]
```

Lý do chọn:

- **Expected: xG** (Bàn thắng kỳ vọng): Chỉ số này đo lường chất lượng cơ hội mà cầu thủ đội tạo ra, không chỉ đếm số bàn thắng thực tế. Nó giúp đánh giá khả năng tấn công một cách bền vững, xem liệu đội có đang tạo ra các cơ hội "ngon ăn"hay không, độc lập với khả năng dứt điểm nhất thời. Phân phối của xG cho thấy mức độ nguy hiểm trung bình mà các cầu thủ đội tạo ra.

- **Per 90 minutes: Gls** (Bàn thắng/90 phút): Đây là thước đo hiệu quả ghi bàn trực tiếp nhất, đã được chuẩn hóa theo thời gian thi đấu. Nó cho biết tần suất một cầu thủ "nổ súng"khi họ có mặt trên sân. Phân phối chỉ số này giúp so sánh khả năng săn bàn giữa các cầu thủ đội một cách công bằng hơn.
- **Shooting: Standard: SoT%** (Tỷ lệ sút trúng đích): Chỉ số này thể hiện độ chính xác của các cú sút. Tỷ lệ cao cho thấy cầu thủ đội thường xuyên đưa bóng đi trúng mục tiêu, gây khó khăn cho thủ môn đối phương. Phân phối của nó cho thấy mức độ hiệu quả chuyển hóa cơ hội thành cú sút nguy hiểm.
- **Performance: assists** (Kiến tạo): Đo lường trực tiếp khả năng tạo ra bàn thắng cho đồng đội. Phân phối số kiến tạo cho thấy những cầu thủ đội nào có khả năng kiến thiết lối chơi và tạo ra những đường chuyền quyết định tốt.
- **Passing: Expected: KP** (Key Passes): Số đường chuyền trực tiếp dẫn đến một cú sút của đồng đội. Chỉ số này thể hiện khả năng tạo ra cơ hội (không nhất thiết thành bàn). Phân phối KP cho thấy tần suất các cầu thủ đội đặt đồng đội vào vị trí có thể dứt điểm.
- **Passing: Expected: PrgP** (Progressive Passes): Số đường chuyền thành công đưa bóng lên phía trước một cách đáng kể (ví dụ: vào 1/3 sân đối phương hoặc tiến một khoảng cách nhất định). Chỉ số này đo lường khả năng phát triển bóng, phá vỡ cấu trúc phòng ngự đối phương. Phân phối PrgP cho thấy đội nào có xu hướng và khả năng đưa bóng lên nhanh và hiệu quả.

Thực hiện:

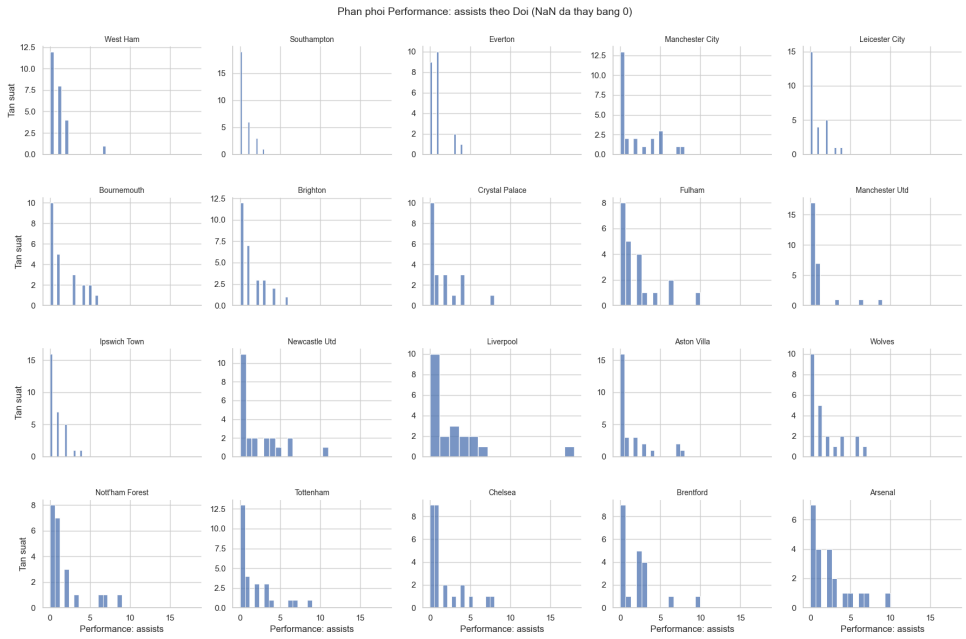
- **Thiết lập:** Import thư viện, định nghĩa `selected_stats`, `OUTPUT_FOLDER`...
- **Hàm tiện ích:** `create_folder`, `sanitize_filename`, `read_csv_file`.
- **Chuẩn bị dữ liệu:** Đọc `results.csv`. Chọn cột, ép kiểu số, và quan trọng là `fillna(0)`.

```
if df is not None:
    missing_required = [col for col in [TEAM_COLUMN] + selected_stats if col not in df.columns]
    if TEAM_COLUMN not in df.columns:
        print(f"Loi nghiem trong: Khong tim thay cot doi '{TEAM_COLUMN}'. Khong the tiep tuc.")
        df = None
    elif missing_required:
        print(f"Canh bao: Cac cot sau khong tim thay trong CSV va se bi bo qua: {', '.join(missing_required)}")
        selected_stats = [stat for stat in selected_stats if stat in df.columns]

    if df is not None and selected_stats:
        required_columns = [TEAM_COLUMN] + selected_stats
        try:
            cleaned_data = df[required_columns].copy()
            print(f"Bat dau xu ly {len(selected_stats)} chi so: {', '.join(selected_stats)}")
            for stat in selected_stats:
                numeric_col = pd.to_numeric(cleaned_data[stat], errors='coerce')
                if not numeric_col.isnull().all():
                    print(f"  -> Xu ly cot '{stat}': Chuyen sang so va fillna(0).")
                    cleaned_data[stat] = numeric_col.fillna(0)
                    valid_stat_columns.append(stat)
                else:
                    print(f"  -> Canh bao: Cot '{stat}' khong chua du lieu so hop le hoac toan NaN. Da loai bo.")
                    if stat in cleaned_data.columns:
                        cleaned_data.drop(columns=[stat], inplace=True)
            if not valid_stat_columns:
                print("Khong con cot chi so nao hop le de ve bieu do.")
                cleaned_data = None
            else:
                print(f"Cac cot chi so hop le se duoc ve: {', '.join(valid_stat_columns)}")
```

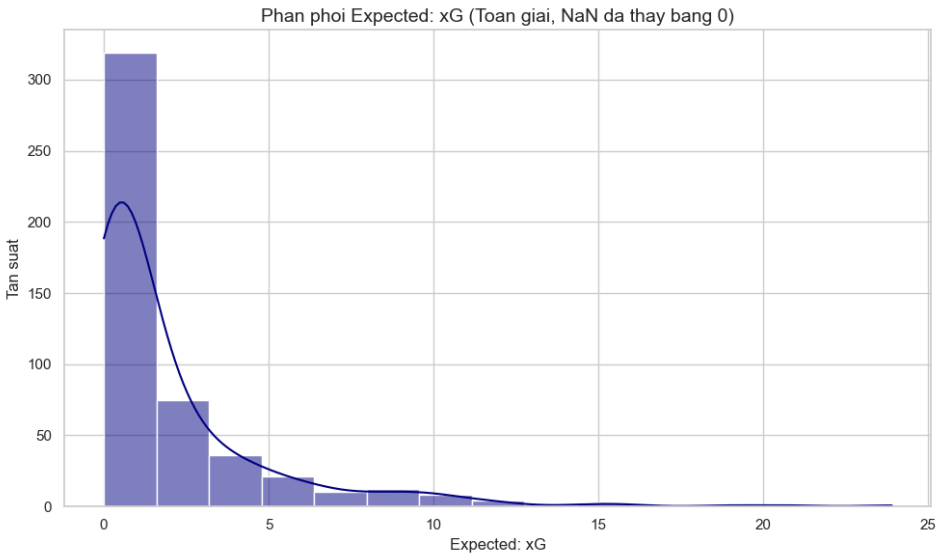
Hình 2.4: Trích đoạn mã chuẩn bị dữ liệu trong part_3.py.

- **Hàm vẽ:**
  - `plot_histogram_all_players_facet`: Vẽ histogram tổng thể.



Hình 2.5: Kết quả của histogram_assists theo đội.

  - `plot_histograms_per_team_facet`: Vẽ lưới histogram theo đội.



Hình 2.6: Phân phối Expected toàn giải.

Kết quả: Các tệp ảnh PNG được lưu trong thư mục 'histograms'.

2.4.5 Xác định Đội bóng Hiệu suất Tốt nhất (Theo từng chỉ số trung bình) - `part_4.py`

Xác định đội dẫn đầu theo từng chỉ số trung bình: Đối với mỗi chỉ số thống kê, em xác định đội bóng có giá trị trung bình (mean) cao nhất. Việc sử dụng giá trị trung bình giúp phản ánh hiệu suất "trung bình"của các cầu thủ trong đội đó ở chỉ số tương ứng.
Thực hiện:

1. **Đọc dữ liệu:** Tải dữ liệu từ tệp `results2.csv` vào một pandas DataFrame.
2. **Tiền xử lý:**
   - Đặt cột chứa tên đội ('Team') làm chỉ mục (index) của DataFrame để dễ dàng truy xuất theo tên đội.
   - Loại bỏ hàng 'all' (nếu có) để chỉ phân tích dữ liệu của các đội cụ thể.
3. **Xác định các cột chỉ số trung bình:** Lọc ra danh sách các cột có tên bắt đầu bằng "Mean of "trong DataFrame.
4. **Lặp và tìm giá trị cao nhất:**
   - Duyệt qua từng cột trong danh sách các cột trung bình.
   - Lấy tên gốc của chỉ số (ví dụ: từ "Mean of Performance: goals"lấy "Performance: goals").
   - Chuyển đổi dữ liệu của cột sang dạng số, xử lý các giá trị không hợp lệ thành NaN (`pd.to_numeric, errors='coerce'`).
   - Sử dụng hàm `idxmax(skipna=True)` để tìm tên đội (index) có giá trị lớn nhất trong cột số liệu hiện tại (bỏ qua NaN).
   - Sử dụng hàm `max(skipna=True)` để lấy giá trị lớn nhất đó.
   - Lưu trữ tên đội dẫn đầu và giá trị tương ứng vào một dictionary.
   - Xử lý các trường hợp không có dữ liệu hợp lệ (toàn NaN).
5. **Hiển thị danh sách dẫn đầu:** In ra danh sách chi tiết, cho biết đội nào dẫn đầu từng chỉ số trung bình và giá trị cụ thể là bao nhiêu.
6. **Thống kê số lần dẫn đầu:**
   - Trích xuất tất cả các tên đội dẫn đầu đã lưu.
   - Sử dụng `pandas.Series.value_counts()` để đếm số lần mỗi đội xuất hiện trong danh sách này.
   - In ra bảng thống kê số lần dẫn đầu của mỗi đội.
7. **Xác định đội tốt nhất :**
   - Tìm tên đội có số lần dẫn đầu cao nhất từ kết quả thống kê ở bước 6 (`idxmax()`).
   - In ra kết luận ban đầu về đội xuất sắc nhất dựa trên số lần dẫn đầu nhiều nhất.

8. **Kiểm tra các chỉ số quan trọng:**

   - Đối với một danh sách các chỉ số quan trọng được định nghĩa trước (như goals, assists, xG, GA90, Save%), hiển thị đội dẫn đầu (cao nhất hoặc thấp nhất tùy chỉ số).
   - Đặc biệt, đối với Goalkeeping: Performance: GA90, sử dụng `idxmin()` và `min()` để tìm đội có giá trị thấp nhất (tốt nhất).

9. **Hoàn thành:** Kết thúc quá trình phân tích và hiển thị kết quả ra console.

Kết quả:

```
Doi co diem trung binh cao nhat moi chi so:
Playing Time: matches played: Liverpool (24.48)
Playing Time: starts: Brentford (17.81)
Playing Time: minutes: Liverpool (1596.38)
Performance: goals: Liverpool (3.76)
Performance: assists: Liverpool (2.81)
Performance: yellow cards: Bournemouth (3.78)
Performance: red cards: Arsenal (8.23)
Expected: xG: Liverpool (3.63)
Expected: xAG: Liverpool (2.63)
Progression: PrgC: Manchester City (40.56)
Progression: PrgP: Liverpool (81.38)
Progression: PrgR: Liverpool (80.62)
Per 90 minutes: Gls: Manchester City (0.18)
Per 90 minutes: Ast: Liverpool (0.15)
Per 90 minutes: xG: Aston Villa (8.19)
Per 90 minutes: xAG: Chelsea (0.15)
Goalkeeping: Performance: GA90: Leicester City (2.73)
Goalkeeping: Performance: Save%: Bournemouth (80.00)
Goalkeeping: Performance: CS%: Brentford (59.10)
Goalkeeping: Penalty Kicks: Save%: Everton (100.00)
Shooting: Standard: SoT%: Nott'ham Forest (38.99)
Shooting: Standard: SoT/90: Fulham (9.54)
Shooting: Standard: G/Sh: Arsenal (8.14)
Shooting: Standard: Dist: Nott'ham Forest (19.09)
Passing: Total: Cmp: Liverpool (778.18)
Passing: Total: Cmp%: Manchester City (86.55)
Passing: Total: TotDist: Liverpool (4445.05)
Passing: By Distance: Short Cmp%: Manchester City (92.15)
Passing: By Distance: Medium Cmp%: Manchester City (89.58)
Passing: By Distance: Long Cmp%: Liverpool (60.32)
Passing: Expected: KP: Liverpool (22.00)
Passing: Expected: 1/3: Liverpool (67.71)
Passing: Expected: PPA: Liverpool (18.62)
Passing: Expected: CrsPA: Fulham (4.23)
Passing: Expected: PrgP: Liverpool (81.38)
Goal and Shot Creation: SCA: SCA: Liverpool (49.81)
```

Goal and Shot Creation: SCA: SCA90: Liverpool (2.63)
Goal and Shot Creation: GCA: GCA: Liverpool (6.48)
Goal and Shot Creation: GCA: GCA90: Liverpool (0.35)
Defensive Actions: Tackles: Tkl: Crystal Palace (32.00)
Defensive Actions: Tackles: TklW: Crystal Palace (18.62)
Defensive Actions: Challenges: Att: Liverpool (28.29)
Defensive Actions: Challenges: Lost: Crystal Palace (13.48)
Defensive Actions: Blocks: Blocks: Brentford (20.38)
Defensive Actions: Blocks: Sh: Brentford (8.38)
Defensive Actions: Blocks: Pass: Crystal Palace (14.14)
Defensive Actions: Blocks: Int: Bournemouth (14.00)
Possession: Touches: Touches: Liverpool (1102.05)
Possession: Touches: Def Pen: Brentford (142.62)
Possession: Touches: Def 3rd: Brentford (357.33)
Possession: Touches: Mid 3rd: Liverpool (497.52)
Possession: Touches: Att 3rd: Manchester City (343.48)
Possession: Touches: Att Pen: Liverpool (55.81)
Possession: Take-Ons: Att: Arsenal (29.73)
Possession: Take-Ons: Succ%: Liverpool (54.91)
Possession: Take-Ons: Tkld%: Leicester City (48.30)
Possession: Carries: Carries: Manchester City (643.48)
Possession: Carries: PrgDist: Manchester City (1994.16)
% PrgC thiếu trong đoạn PDF này
Possession: Carries: 1/3: Manchester City (30.36)
Possession: Carries: CPA: Manchester City (14.04)
Possession: Carries: Mis: Nott'ham Forest (23.80)
Possession: Carries: Dis: Newcastle Utd (17.65)
Possession: Receiving: Rec: Liverpool (769.67)
Possession: Receiving: PrgR: Liverpool (80.62)
Miscellaneous: Performance: Fls: Bournemouth (19.83)
Miscellaneous: Performance: Fld: Newcastle Utd (17.61)
Miscellaneous: Performance: Off: Nott'ham Forest (3.73)
Miscellaneous: Performance: Crs: Fulham (36.82)
Miscellaneous: Performance: Recov: Bournemouth (71.44)
Miscellaneous: Aerial Duels: Won: Brentford (26.76)
Miscellaneous: Aerial Duels: Lost: Crystal Palace (26.57)
Miscellaneous: Aerial Duels: Won%: Southampton (54.17)

2.4.6 Xác định đội bóng có hiệu suất tổng thể tốt nhất - `part_4.py`

Từ kết quả ở mục 2.4.5, số lần mỗi đội dẫn đầu về chỉ số trung bình được thống kê như sau:

Team	**Số lần dẫn đầu**
Liverpool	27
Manchester City	10
Brentford	7
Bournemouth	5
Crystal Palace	5
Nott'ham Forest	4
Fulham	3
Arsenal	3
Newcastle Utd	2
Leicester City	2
Aston Villa	1
Chelsea	1
Everton	1
Southampton	1

Bảng 2.1: Thống kê số lần dẫn đầu chỉ số trung bình của các đội.

Dựa trên số chỉ số trung bình dẫn đầu, 'Liverpool' là đội xuất hiện nhiều nhất.

Kiểm tra các chỉ số quan trọng:

- Cao nhất Performance: goals: Liverpool (3.76)
- Cao nhất Performance: assists: Liverpool (2.81)
- Cao nhất Goalkeeping: Performance: Save%: Bournemouth (80.00)

Kết luận: Dựa trên kết quả phân tích số lượng chỉ số thống kê có giá trị trung bình cao nhất, Liverpool là đội dẫn đầu ở nhiều hạng mục nhất (27 lần) và do đó được xem là đội có hiệu suất tốt nhất trong mùa giải 2024-2025 theo phương pháp đánh giá này.

2.5 Kết quả

Sau khi chạy xong chương trình, kết quả của phần II được lưu trữ như sau:

- **top_3.txt**: danh sách top 3 cầu thủ cao nhất và thấp nhất cho từng chỉ số thống kê
- **results2.csv**: bảng tổng hợp các giá trị trung vị, trung bình, độ lệch chuẩn cho từng chỉ số tính toán trên toàn giải và theo từng đội
- **Thư mục histograms**: chứa các tệp hình ảnh png là các biểu đồ histogram phân phối của các chỉ số đã chọn, cho toàn giải và từng đội.
- **Output trên console:** hiển thị đội có điểm cao nhất cho từng chỉ số và kết quả đội có tổng thể cao nhất

Chương 3

Phân cụm và Giảm chiều dữ liệu

3.1 Phân tích đề bài

Theo yêu cầu của đề bài phần III, nhiệm vụ chính bao gồm:

- **Phân cụm cầu thủ:** Sử dụng thuật toán K-means để tự động nhóm các cầu thủ có đặc điểm thống kê tương đồng vào cùng một cụm.
- **Xác định số cụm tối ưu:** Tìm ra số lượng cụm (k) phù hợp nhất cho bộ dữ liệu và giải thích cơ sở cho lựa chọn đó.
- **Giảm chiều và Trực quan hóa:** Áp dụng phân tích thành phần chính (PCA) để giảm số chiều của dữ liệu xuống còn 2, sau đó vẽ biểu đồ 2D để trực quan hóa các cụm cầu thủ đã phân loại.
- **Đánh giá và Bình luận:** nhận xét về kết quả phân cụm và trực quan hóa thu được.

3.2 Lựa chọn thư viện

Để thực hiện các yêu cầu trên, em đã sử dụng các thư viện:

- **Pandas:** dùng để đọc xử lý và thao tác với dữ liệu cầu thủ dưới dạng bảng (DataFrame)
- **NumPy:** cung cấp các cấu trúc mảng và hàm toán học nền tảng cho các tính toán số.
- **Scikit-learn:** Cung cấp các thuật toán học máy và công cụ tiền xử lý (KMeans, PCA, StandardScaler, OneHotEncoder, SimpleImputer, Pipeline, Column Transformer).
- **Matplotlib & Seaborn:** Dùng để tạo các biểu đồ trực quan hóa, bao gồm biểu đồ Elbow và biểu đồ phân cụm 2D.
- **os:** Hỗ trợ quản lý đường dẫn file.

3.3 Quy trình tổng thể

Quá trình phân tích và thực hiện em đã chia làm 2 module: `kmeans_pca.py` và `plotting.py`:

1. **Nạp dữ liệu:** tải dữ liệu từ file results.csv (kết quả của bài 1)
2. **Tiền xử lý dữ liệu:** thực hiện các bước làm sạch, chuẩn hóa và mã hóa cần thiết để chuẩn bị dữ liệu cho các thuật toán học máy.
3. **Xác định số cụm (k) bằng phương pháp Elbow:** Chạy K-means với nhiều giá trị k khác nhau và vẽ biểu đồ Elbow (elbow_plot.png) để tìm ra điểm "khuỷu tay".
4. **Phân cụm K-means:** áp dụng K-means với số cụm tối ưu đã chọn để gán nhãn cụm cho từng cầu thủ. Lưu kết quả vào `results_clustered.csv`.
5. **Giảm chiều bằng PCA:** Áp dụng PCA với n_components = 2 để giảm chiều dữ liệu đã tiền xử lý. Lưu kết quả PCA vào `pca_clusters.csv`.
6. **Trực quan hóa kết quả phân cụm:** dùng plotting.py để vẽ biểu đồ scatter 2D, lưu kết quả vào file `kmeans_pca_cluster_plot_with_variance.jpg`, thể hiện các cầu thủ trên mặt phẳng PCA và tô màu theo cụm đã phân loại.

3.4 Phân tích chi tiết

3.4.1 Tiền xử lý dữ liệu (`kmeans_pca.py`)

Đây là bước nền tảng mà em thực hiện để chuẩn bị dữ liệu từ results.csv cho các thuật toán K-means và PCA. Quá trình này bao gồm nhiều giai đoạn:

- **Tải dữ liệu:** Dữ liệu được nạp vào DataFrame Pandas từ file results.csv.
- **Xác định kiểu dữ liệu cột:** Em đã cấu hình script để tự động xác định các cột có kiểu dữ liệu số (np.number) và xác định các cột phân loại dự kiến (`categorical_cols = ['Nation', 'Team', 'Position']`). Cột định danh cầu thủ (cột đầu tiên) cũng được em loại bỏ khỏi tập dữ liệu dùng để phân tích (features).
- **Làm sạch dữ liệu số:** Em đã xử lý các cột số bằng cách lặp qua chúng để:
  - Thay thế giá trị 'N/a' bằng NaN.
  - Nếu cột đang ở dạng object (chuỗi), loại bỏ dấu phẩy (,) ngăn cách hàng nghìn.
  - Chuyển đổi cột sang kiểu số học bằng `pd.to_numeric`, các giá trị không thể chuyển đổi sẽ thành NaN (`errors='coerce'`).

```
for col in numerical_cols:
    if col in features.columns:
        features[col] = features[col].replace('N/a', np.nan)
        if not pd.api.types.is_numeric_dtype(features[col]):
            print(f"  Xu ly cot so dang chuoi/object: {col}")
            features[col] = features[col].astype(str).str.replace(',', '', regex=False)
            features[col] = pd.to_numeric(features[col], errors='coerce')
        else:
            features[col] = pd.to_numeric(features[col], errors='coerce')
```

Hình 3.1: Trích đoạn mã làm sạch cột số trong kmeans_pca.py.

- **Xử lý giá trị thiếu (Imputation):** Em đã sử dụng SimpleImputer từ Scikit-learn:
  - Đối với các cột số, các giá trị NaN được thay thế bằng giá trị trung bình (`strategy='mean'`) của cột đó.
  - Đối với các cột phân loại, các giá trị NaN được thay thế bằng giá trị xuất hiện nhiều nhất (`strategy='most_frequent'`) trong cột đó.
- **Chuẩn hóa dữ liệu số (Scaling):** Áp dụng StandardScaler cho tất cả các cột số đã được xử lý giá trị thiếu. Bước này biến đổi dữ liệu sao cho mỗi cột số có giá trị trung bình là 0 và độ lệch chuẩn là 1, đảm bảo các đặc trưng đóng góp công bằng vào kết quả.
- **Mã hóa dữ liệu phân loại (Encoding):** Em đã áp dụng OneHotEncoder cho các cột phân loại ('Nation', 'Team', 'Position') để chuyển đổi chúng thành dạng số nhị phân mà thuật toán có thể xử lý.
- **Kết hợp các bước bằng Pipeline và Column Transformer:**
  - Tạo hai Pipeline riêng biệt được tạo: một cho xử lý cột số (Imputer -> Scaler) và một cho xử lý cột phân loại (Imputer -> OneHotEncoder).
  - Em đã sử dụng Column Transformer để áp dụng đúng pipeline xử lý cho đúng nhóm cột và kết hợp kết quả lại. Các cột không được chỉ định sẽ bị loại bỏ (`remainder='drop'`).

```
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore', sparse_output=False))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_cols),
        ('cat', categorical_transformer, categorical_cols)
    ],
    remainder='drop'
)
```

Hình 3.2: Định nghĩa Pipeline và ColumnTransformer trong kmeans_pca.py.

- **Kết quả tiền xử lý:** Đầu ra của toàn bộ quá trình này là biến `processed_data`, một mảng NumPy chứa dữ liệu đã được em chuẩn bị sẵn sàng cho các bước phân cụm và giảm chiều tiếp theo.

3.4.2 Xác định số cụm tối ưu (Phương pháp Elbow) (kmeans_pca.py)

Để xác định số lượng cụm (k) phù hợp, phương pháp Elbow đã được em áp dụng.

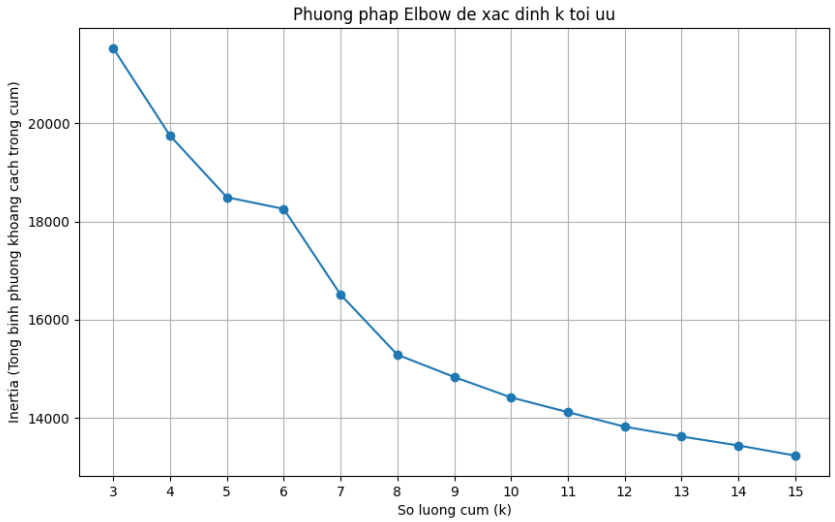
Thực hiện: Em đã lặp qua một khoảng giá trị k (từ 3 đến 16). Với mỗi giá trị k, thuật toán KMeans được chạy (`n_init= 'auto'`, `random_state=42`). Giá trị `inertia_` (WCSS) tương ứng với mỗi k được ghi lại.

Trực quan hóa: Các giá trị inertia vẽ lên đồ thị Elbow và lưu vào file `elbow_plot.png`.

```
# Ve bieu do Elbow
try:
    plt.figure(figsize=(10, 6))
    plt.plot(k_range, inertia, marker='o')
    plt.title('Phuong phap Elbow de xac dinh k toi uu')
    plt.xlabel('So luong cum (k)')
    plt.ylabel('Inertia (Tong binh phuong khoang cach trong cum)')
    plt.xticks(k_range)
    plt.grid(True)
    elbow_plot_path = os.path.join(current_dir, 'elbow_plot.png')
    plt.savefig(elbow_plot_path)
    print(f"Da luu bieu do Elbow vao '{elbow_plot_path}'. Hay xem bieu do nay va chon gia tri 'k' tai diem 'khuyu tay'.")
    plt.close()
except Exception as plot_e:
    print(f"Loi khi ve hoac luu bieu do Elbow: {plot_e}")
```

Hình 3.3: Vẽ biểu đồ Elbow.

Lựa chọn k: Sau khi biểu đồ `elbow_plot.png` được tạo và lưu lại từ kết quả tính toán inertia, bước tiếp theo là em tiến hành quan sát trực quan đồ thị này để xác định số cụm tối ưu (k). Mục tiêu của việc quan sát là tìm ra điểm "khuỷu tay"(elbow point) trên đường cong biểu diễn giá trị inertia theo số cụm k. Điểm "khuỷu tay"này là điểm mà tại đó, đường cong thay đổi độ dốc một cách rõ rệt nhất – từ dốc đứng (inertia giảm nhanh khi tăng k) sang thoai thoải hơn (inertia giảm chậm lại đáng kể khi tiếp tục tăng k).



Hình 3.4: Hình vẽ elbow để xác định k tối ưu.

Phân tích biểu đồ cho thấy Inertia giảm mạnh khi k tăng từ 3 đến 5, sau đó tốc độ giảm chậm lại đáng kể. Điểm k=6 được em xác định là "điểm khuỷu tay"(elbow point) vì đây là nơi đường cong bắt đầu phẳng ra, thể hiện sự cân bằng tốt nhất giữa việc giảm thiểu phương sai trong cụm và việc tránh tạo ra số lượng cụm quá lớn. Việc tăng k vượt quá 6 chỉ mang lại lợi ích giảm dần về Inertia, do đó, em đã chọn k=6 là số cụm tối ưu để phân loại các nhóm cầu thủ.

3.4.3 Thực hiện và Phân tích các cụm

Thực hiện phân cụm K-Means (`kmeans_pca.py`)

Sau khi đã xác định số cụm tối ưu là k=6 từ phương pháp Elbow, thuật toán K-means được em áp dụng trên bộ dữ liệu đã tiền xử lý (`processed_data`).

- **Khởi tạo và Huấn luyện**: Em đã khởi tạo một đối tượng `KMeans` từ thư viện Scikit-learn với các tham số `n_clusters=6`, `random_state=42` và `n_init='auto'` (hoặc giá trị n_init cụ thể như 10 để đảm bảo tính ổn định). Phương thức `fit_predict()` được gọi trên `processed_data` để huấn luyện mô hình và gán nhãn cụm (từ 0 đến 5) cho mỗi cầu thủ.
- **Lưu kết quả**: Mảng `cluster_labels` chứa nhãn cụm (0, 1, 2, 3, 4, hoặc 5) cho từng cầu thủ được em thêm vào DataFrame gốc (`data`) dưới dạng một cột mới có tên là 'Cluster'. DataFrame cập nhật này sau đó được lưu vào file `results_clustered.csv`. File này là cơ sở dữ liệu để em thực hiện phân tích đặc điểm chi tiết của 6 cụm ở mục tiếp theo.

Phân tích và Diễn giải các cụm

Dựa trên kết quả phân cụm K-Means với k=6 và các số liệu thống kê tính toán cho từng cụm, em đưa ra nhận xét sơ bộ về đặc điểm của 6 cụm như sau:

- **Cụm 0 (21 cầu thủ): Thủ Môn (GK)** Đây là cụm có ít cầu thủ nhất với chỉ 21 người (chiếm 4.28% tổng số cầu thủ). Phần lớn các cầu thủ trong nhóm này chơi ở vị trí Thủ môn (GK), có vai trò quan trọng trong việc bảo vệ khung thành và không tham gia trực tiếp vào các pha tấn công. Những cầu thủ trong cụm này thường có khả năng phản xạ xuất sắc và giúp đội bảo vệ khung thành vững chắc.
- **Cụm 1 (59 cầu thủ): Hậu Vệ (DF)** Cụm này có 59 cầu thủ, chiếm 12.02% tổng số cầu thủ. Phần lớn các cầu thủ trong cụm này chơi ở vị trí Hậu vệ (DF), là những người bảo vệ khu vực phòng ngự và ngăn chặn các pha tấn công của đối thủ. Các cầu thủ trong cụm này có khả năng đọc trận đấu tốt và hỗ trợ từ phía sau cho các pha tấn công.
- **Cụm 2 (100 cầu thủ): Hậu Vệ (DF)** Đây là một trong những cụm lớn nhất với 100 cầu thủ, chiếm 20.37% tổng số cầu thủ. Tương tự như cụm 1, phần lớn các cầu thủ trong cụm này chơi ở vị trí Hậu vệ (DF). Cụm này đại diện cho những cầu thủ có nhiệm vụ bảo vệ khu vực giữa sân và hỗ trợ trong các tình huống không chiến.
- **Cụm 3 (99 cầu thủ): Tấn Công và Hỗ Trợ (FW, MF)** Với 99 cầu thủ (chiếm 20.16%), đây là một cụm khá đa dạng, bao gồm cả các cầu thủ chơi ở vị trí Tiền

đạo (FW) và Tiền vệ (MF). Cụm này có thể gồm các cầu thủ tham gia vào việc tổ chức tấn công và ghi bàn. Những cầu thủ trong cụm này có vai trò quan trọng trong việc tạo ra cơ hội ghi bàn hoặc tham gia vào các pha phản công.

- **Cụm 4 (148 cầu thủ): Hậu Vệ (DF)** Đây là cụm lớn nhất với 148 cầu thủ, chiếm 30.14% tổng số cầu thủ. Phần lớn các cầu thủ trong cụm này chơi ở vị trí Hậu vệ (DF). Cụm này đại diện cho những cầu thủ có khả năng phòng ngự xuất sắc và đóng vai trò quan trọng trong việc bảo vệ khung thành. Họ giúp đội nhà duy trì sự an toàn và vững chắc trong suốt trận đấu.

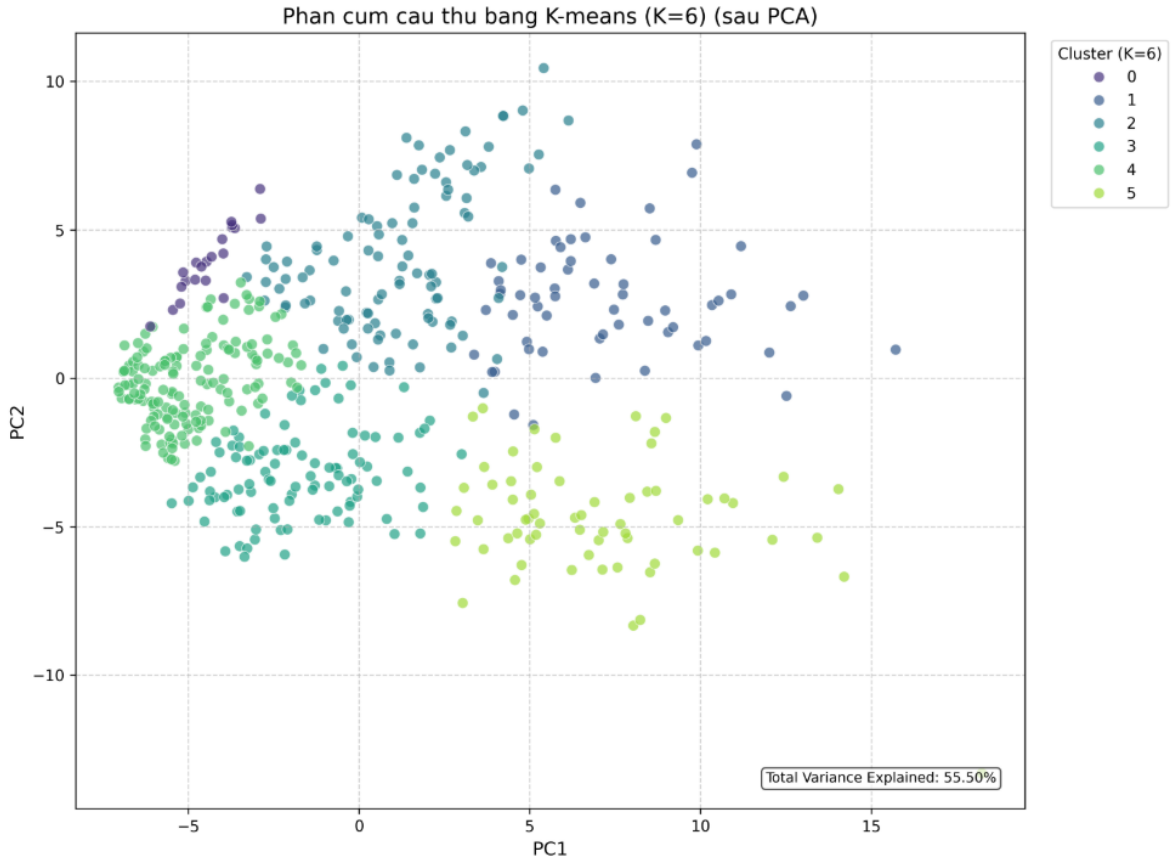
- **Cụm 5 (64 cầu thủ): Tiền Đạo (FW)** Cụm này có 64 cầu thủ, chiếm 13.03% tổng số cầu thủ. Chủ yếu các cầu thủ trong cụm này chơi ở vị trí Tiền đạo (FW). Đây là những cầu thủ có khả năng ghi bàn xuất sắc và luôn sẵn sàng tấn công đối thủ. Những cầu thủ này có vai trò quan trọng trong việc mang lại chiến thắng cho đội nhà thông qua các pha ghi bàn quyết định.

3.4.4 Giảm chiều dữ liệu và trực quan hóa bằng PCA

Sau khi đã phân nhóm cầu thủ thành 6 cụm ở mục 4.3, em tiến hành giảm chiều dữ liệu và trực quan hóa các cụm này bằng Phân tích Thành phần Chính (PCA) để có cái nhìn rõ hơn về sự phân bố của chúng.

- **Các bước thực hiện**:
  - Áp dụng PCA để giảm chiều (`kmeans_pca.py`): Sử dụng PCA từ Scikit-learn để giảm số chiều của bộ dữ liệu đặc trưng đã tiền xử lý (`processed_data`) xuống còn 2 thành phần chính (PC1 và PC2), giữ lại `n_components=2`. Kết quả là `pca_components`, chứa tọa độ mới của mỗi cầu thủ.
  - Chuẩn bị dữ liệu cho trực quan hóa (`kmeans_pca.py`): Tạo DataFrame `pca_df` từ `pca_components`, bổ sung nhãn cụm. Dữ liệu này được lưu vào file `pca_clusters.csv`.
  - Vẽ biểu đồ trực quan hóa (`plotting.py`): Sử dụng seaborn và matplotlib đọc từ `pca_clusters.csv` để vẽ biểu đồ tán xạ. Trục hoành là PC1, trục tung là PC2. Mỗi điểm đại diện cho một cầu thủ, màu sắc thể hiện cụm (0 đến 5) mà cầu thủ đó thuộc về.

- **Biểu đồ PCA**:



Hình 3.5: Phân cụm cầu thủ bằng K-means (K=6).

Chương 4

Phần IV: Ước tính Giá trị Cầu thủ bằng Học máy

4.1 Thu thập và Chuẩn bị Dữ liệu Giá trị Chuyển nhượng

Để xây dựng cơ sở dữ liệu phục vụ cho việc ước tính giá trị cầu thủ theo phương pháp học máy, công việc đầu tiên cần thực hiện là thu thập thông tin về giá trị chuyển nhượng của các cầu thủ. Dữ liệu này bao gồm giá trị ước tính hiện tại (Current ETV) và giá trị ước tính cao nhất (Highest ETV), được lấy từ nguồn `footballtransfers.com` và áp dụng cho các cầu thủ đáp ứng điều kiện về thời gian thi đấu tối thiểu.

4.1.1 Quy trình Thu thập Dữ liệu Tự động (Web Scraping)

Để thực hiện yêu cầu ước tính giá trị cầu thủ, bước đầu tiên và quan trọng là thu thập dữ liệu về giá trị chuyển nhượng của các cầu thủ đang thi đấu tại giải Ngoại hạng Anh mùa giải 2024-2025. Dữ liệu này được lấy từ trang web footballtransfers.com và được lọc theo tiêu chí thời gian thi đấu tối thiểu được quy định trong đề bài.

- **Thư viện và Lý do lựa chọn:**
  - **Selenium:** Đây là nền tảng cho việc tự động hóa tương tác với trình duyệt web (Chrome), bao gồm điều hướng, tìm kiếm phần tử và trích xuất dữ liệu từ các trang web động.
  - **Pandas:** Công cụ chính cho việc đọc dữ liệu đầu vào (từ tệp CSV), lưu trữ dữ liệu thu thập được dưới dạng DataFrame, cũng như thực hiện các thao tác lọc và xuất kết quả ra tệp CSV mới.
  - **re:** Sử dụng để phân tích và chuẩn hóa các chuỗi văn bản chứa giá trị chuyển nhượng (ETV), chuyển đổi chúng thành định dạng số.
  - **os:** Cung cấp các chức năng tương tác với hệ thống tệp, cụ thể là kiểm tra và tạo thư mục lưu trữ kết quả.
  - **time:** Cho phép tạo các khoảng dừng trong kịch bản, quản lý thời gian tải trang và tránh gửi yêu cầu quá nhanh.

- **traceback:** Giúp ghi lại thông tin chi tiết khi có lỗi xảy ra, phục vụ cho việc gỡ lỗi.
- **random:** Được dùng để tạo khoảng thời gian chờ ngẫu nhiên nhỏ, giúp quá trình scraping mô phỏng hành vi người dùng tự nhiên hơn.

- **Các bước thực hiện:**

1. Đầu tiên, script xử lý tệp results.csv để lọc những cầu thủ thi đấu trên 900 phút, đồng thời chuẩn hóa tên của họ và lưu vào một danh sách eligible_players_set dùng cho việc đối chiếu sau này.
2. Kế tiếp, Selenium được dùng để mở trang web `footballtransfers.com` và tự động duyệt qua tất cả các trang con chứa danh sách cầu thủ.
3. Từ mỗi trang danh sách, các thông tin cơ bản của cầu thủ như Tên, URL trang chi tiết, Đội, Tuổi, Vị trí và Giá trị Chuyển nhượng Hiện tại (Current ETV) dạng văn bản thô sẽ được thu thập.
4. Sau đó, script truy cập trang chi tiết của từng cầu thủ để lấy Giá trị Chuyển nhượng Cao nhất (Highest ETV) dạng văn bản. Cả hai loại giá trị ETV này sau đó được hàm `parse_etv` chuyển đổi sang dạng số (triệu Euro).
5. Cuối cùng, toàn bộ dữ liệu đã xử lý của mỗi cầu thủ (Tên, Đội, Tuổi, Vị trí, và hai giá trị ETV dạng số) được tập hợp vào danh sách `all_results` để chuẩn bị cho việc lưu trữ.

4.1.2 Chuẩn bị và Lưu trữ Dữ liệu Cuối cùng

Sau khi hoàn tất quá trình duyệt web và thu thập thông tin giá trị chuyển nhượng, dữ liệu được lưu trong danh sách `all_results` sẽ được xử lý và lưu trữ vào các tệp CSV.

1. **Chuyển đổi sang DataFrame:** Danh sách `all_results`, chứa các dictionary thông tin của từng cầu thủ, được chuyển đổi thành một đối tượng Pandas DataFrame. Thao tác này giúp cấu trúc hóa dữ liệu dưới dạng bảng, thuận tiện cho việc sắp xếp, lọc và lưu trữ.

2. **Lựa chọn và Sắp xếp Cột:** Để đảm bảo tính nhất quán và dễ đọc của dữ liệu đầu ra, các cột cần thiết được lựa chọn và sắp xếp theo một thứ tự cụ thể: Player, Team, Age, Position, TransferValue_EUR_Millions, Highest_ETV_EUR_Millions. Script đảm bảo chỉ các cột này được giữ lại trong DataFrame cuối cùng.

3. **Kiểm tra và Tạo Thư mục Output:** Trước khi tiến hành lưu tệp, script kiểm tra xem thư mục đích (ví dụ: Report/OUTPUT_BAI4/ từ các biến `ALL_PLAYERS_FILE` và `OVER_900_FILE`) có tồn tại không. Nếu thư mục chưa tồn tại, nó sẽ được tạo tự động bằng `os.makedirs()` để tránh lỗi khi ghi tệp.

4. **Lưu Dữ liệu ra Tệp CSV:** Sau khi thu thập, dữ liệu của tất cả cầu thủ (đã chọn lọc các cột cần thiết) được lưu vào tệp `all_players_scraped.csv` (biến `ALL_PLAYERS_FILE`). Tiếp đó, nếu danh sách cầu thủ trên 900 phút (`eligible_players_set`) có sẵn, DataFrame này sẽ được lọc: tên cầu thủ được chuẩn hóa để so khớp với `eligible_players_set`, chỉ giữ lại các cầu thủ phù hợp. Kết quả lọc cuối cùng này được lưu vào tệp `players_over_900_filtered.csv` (biến `OVER_900_FILE`).

4.2 Xử lý dữ liệu

Sau khi quá trình thu thập dữ liệu giá trị chuyển nhượng thô từ trang footballtransfers.com được thực hiện (`main_4.py`), dữ liệu này cần được xử lý và lọc để đảm bảo tính chính xác và phù hợp với yêu cầu của bài toán. Kết quả của quá trình thu thập ban đầu, chứa thông tin của tất cả các cầu thủ cào được, được lưu vào tệp `all_players_scraped.csv`. Mục tiêu chính của giai đoạn xử lý này là tạo ra một bộ dữ liệu chỉ bao gồm những cầu thủ đã thi đấu trên 900 phút trong mùa giải và đã có thông tin giá trị chuyển nhượng. Quá trình này được thực hiện trong cùng `main_4.py`, sau khi đã hoàn tất việc cào dữ liệu.

4.2.1 Tải danh sách cầu thủ đủ điều kiện ($>$ 900 phút)

Để có cơ sở cho việc lọc, trước tiên cần xác định những cầu thủ nào đã đáp ứng tiêu chí về thời gian thi đấu.

1. **Đọc dữ liệu Phần I:** Hàm `load_eligible_players` được gọi để đọc tệp results.csv (biến `RESULTS_CSV_PATH`). Tệp này chứa dữ liệu thống kê hiệu suất chi tiết của tất cả cầu thủ từ fbref.com đã chơi trên 90 phút.

2. **Xử lý cột thời gian thi đấu:** Trong tệp `results.csv`, cột Playing Time: minutes được xử lý để đảm bảo nó ở định dạng số.

3. **Lọc theo ngưỡng 900 phút:** Dữ liệu sau đó được lọc để chỉ giữ lại những cầu thủ có số phút thi đấu lớn hơn 900 (`min_minutes=900`).

4. **Chuẩn hóa và tạo tập hợp tên:**Tên của các cầu thủ đủ điều kiện (từ cột Player) được trích xuất. Để đảm bảo việc so khớp tên sau này được chính xác, hàm `normalize_name` (đã định nghĩa trong main_4.py) được áp dụng. Hàm này chuyển tên về chữ thường, loại bỏ dấu phụ và các khoảng trắng thừa. Kết quả là một set (tập hợp) chứa tên các cầu thủ đã được chuẩn hóa và đủ điều kiện. Việc dùng set giúp tối ưu hóa tốc độ kiểm tra ở bước sau. Nếu tệp results.csv không tìm thấy hoặc có lỗi, một set rỗng sẽ được trả về và thông báo lỗi được in ra.

4.2.2 Lọc dữ liệu giá trị chuyển nhượng

Sau khi đã có set tên các cầu thủ đủ điều kiện (`eligible_players_set`), script tiến hành lọc DataFrame chứa tất cả dữ liệu cầu thủ đã cào được từ footballtransfers.com (được lưu trong biến `df_all_to_save`, đọc từ `ALL_PLAYERS_FILE`).

1. **Chuẩn hóa tên trong dữ liệu đã cào:** Tên cầu thủ trong cột 'Player' của `df_all_to_save` cũng được chuẩn hóa bằng hàm `normalize_name` và lưu vào một cột tạm thời.

2. **Thực hiện lọc:** DataFrame `df_all_to_save` được lọc bằng cách sử dụng phương thức `.isin()` của Pandas. Chỉ những hàng mà giá trị trong cột 'normalized_player' (tên đã chuẩn hóa từ footballtransfers.com) nằm trong `eligible_players_set` (tên đã chuẩn hóa từ fbref.com và đã lọc $>$900 phút) mới được giữ lại.

3. **Loại bỏ cột tạm:** Cột 'normalized_player' được loại bỏ sau khi hoàn tất việc lọc.

4. **Chọn các cột cần thiết:** Kết quả lọc (`df_filtered`) sau đó chỉ giữ lại các cột đã định nghĩa trong `columns_to_keep` (bao gồm "Player", "Team", "Age", "Position", "TransferValue_EUR_Millions", "Highest_ETV_EUR_Millions").

4.2.3 Lưu kết quả đã xử lý

DataFrame chứa thông tin giá trị chuyển nhượng của các cầu thủ đã được lọc chính xác theo yêu cầu (> 900 phút thi đấu) được lưu vào tệp Report/OUTPUT_BAI4/players_over_900_filtered.csv (biến `OVER_900_FILE` trong code). Tệp này là đầu ra quan trọng của `main_4.py` và sẽ được sử dụng làm một trong những nguồn dữ liệu chính cho `main_4.2.py` (module huấn luyện mô hình).

	A	B	C	D	E	F
1	Player	Team	Age	Position	TransferValue_EUR_Millions	Highest_ETV_EUR_Millions
2	Erling Haaland	Man City	24	FW	198.8	199.6
3	Martin Ødegaard	Arsenal	26	MF	126.5	134.5
4	Alexander Isak	Newcastle Utd.	25	FW	120.3	120.3
5	Cole Palmer	Chelsea	23	MF	115.4	119
6	Declan Rice	Arsenal	26	MF	107.8	120
7	Alexis Mac Allister	Liverpool	26	MF	106.1	117
8	Phil Foden	Man City	24	MFFW	105.7	138.6
9	Bukayo Saka	Arsenal	23	FWMF	101.3	127.5
10	Ryan Gravenberch	Liverpool	22	MF	85.3	85.5
11	Bruno Guimarães	Newcastle Utd.	27	MF	83.2	83.2
12	Moisés Caicedo	Chelsea	23	MFDF	80.7	86.1
13	William Saliba	Arsenal	24	DF	79.5	79.5
14	Omar Marmoush	Man City	26	FWMF	79.1	79.1
15	Joško Gvardiol	Man City	23	DF	78.7	86.6

Hình 4.1: Đoạn kết quả mẫu từ file players_over_900_filtered.csv

4.3 Đề xuất phương pháp ước tính giá trị cầu thủ main_4_2.py

Sau khi đã thu thập và xử lý dữ liệu giá trị chuyển nhượng của các cầu thủ ở mục 4.2, bước tiếp theo là đề xuất và triển khai một phương pháp học máy để ước tính giá trị của họ. Mục tiêu là xây dựng một mô hình có khả năng dự đoán biến TransferValue_EUR_Millions dựa trên các thông số thống kê hiệu suất thi đấu và thông tin cá nhân của cầu thủ. Toàn bộ quy trình này được thực hiện trong tệp `main_4.2.py`.

4.3.1 Giới thiệu quy trình tổng thể

Quy trình ước tính giá trị cầu thủ được em thực hiện bao gồm các bước chính sau:

1. **Tải và kết hợp dữ liệu:** Nạp dữ liệu thống kê cầu thủ từ results.csv và dữ liệu giá trị chuyển nhượng đã lọc từ mục 4.2. Hai bộ dữ liệu này được hợp nhất (`merge`) dựa trên tên cầu thủ.

2. **Tiền xử lý dữ liệu và Tạo đặc trưng (Feature Engineering):**

    - Xử lý biến mục tiêu: Áp dụng phép biến đổi log (`np.log1p`) cho cột giá trị chuyển nhượng để giảm độ lệch và xử lý outliers bằng kỹ thuật Winsorizing.

- Loại bỏ các cột định danh không cần thiết cho việc huấn luyện.
- Làm sạch và chuyển đổi kiểu dữ liệu cho các cột.
- Tạo các đặc trưng mới (advanced features) từ các đặc trưng hiện có để tăng cường khả năng biểu diễn của dữ liệu. Các đặc trưng này bao gồm:
  - Binning (chia khoảng) cho các biến liên tục như Age và Playing Time: minutes bằng `KBinsDiscretizer`.
  - Tạo các đặc trưng tương tác giữa các biến (ví dụ: Age_x_Minutes, Age_Sq, tương tác giữa tuổi/chỉ số với vị trí thi đấu Is_FW, Is_MF).
  - Tạo các đặc trưng tỷ lệ (ví dụ: Gls_div_xG).
- Xử lý các giá trị thiếu (NaN) và outliers cho các đặc trưng.
- Chuẩn bị dữ liệu cho mô hình: Chia dữ liệu thành tập huấn luyện (train) và tập kiểm tra (test). Xác định các cột số và cột hạng mục.

3. **Xây dựng Pipeline tiền xử lý:** Sử dụng `ColumnTransformer` để áp dụng các bước xử lý riêng biệt cho từng loại đặc trưng:
   - Đặc trưng số: `KNNImputer` (điền giá trị thiếu), `PolynomialFeatures` (tạo đặc trưng đa thức và tương tác bậc hai), và `StandardScaler` (chuẩn hóa).
   - Đặc trưng hạng mục: `SimpleImputer` (điền giá trị thiếu) và `OneHotEncoder` (chuyển đổi sang dạng số).
4. **Lựa chọn Đặc trưng bằng RFE:** Mã có bao gồm tùy chọn sử dụng Recursive Feature Elimination (RFE) với một mô hình GradientBoostingRegressor cơ sở để chọn ra một tập con các đặc trưng quan trọng nhất trước khi huấn luyện mô hình cuối cùng.
5. **Huấn luyện mô hình:** Sử dụng `GradientBoostingRegressor` từ `sklearn.ensemble`. Các siêu tham số tốt nhất cho mô hình được tìm kiếm bằng `RandomizedSearchCV` trên một không gian tham số rộng, tối ưu theo chỉ số R^2.
6. **Đánh giá mô hình:** Dự đoán trên tập kiểm tra và đánh giá hiệu suất bằng các chỉ số R^2 (trên cả thang đo log và thang đo gốc), Root Mean Squared Error (RMSE), và Mean Absolute Error (MAE).
7. **Trực quan hóa kết quả:** Vẽ các biểu đồ bao gồm Mức độ quan trọng của đặc trưng, Giá trị dự đoán so với Giá trị thực tế, và Biểu đồ Phần dư.
8. **Lưu trữ:** Lưu mô hình đã huấn luyện và đối tượng preprocessor để có thể tái sử dụng.

4.3.2 Lựa chọn đặc trưng (Feature Selection)

Việc lựa chọn và tạo đặc trưng có ảnh hưởng lớn đến hiệu suất của mô hình.

1. **Dữ liệu đầu vào:**
   - Dữ liệu thống kê từ `results.csv` (Phần I).
   - Thông tin cơ bản và giá trị chuyển nhượng từ `players_over_900_filtered.csv` (Phần IV, mục 4.3).

- Cột Highest_ETV_EUR_Millions được xem xét sử dụng để tạo đặc trưng mới như Value_vs_HighestETV_Ratio nhằm phản ánh giá trị hiện tại so với giá trị đỉnh cao của cầu thủ.

2. **Quá trình xử lý và tạo đặc trưng trong prepare_features_target và create_more_advanced_features**:
   - Các cột định danh (Player, Nation, Team, Team_TransferSite) được loại bỏ khỏi tập đặc trưng huấn luyện (X).
   - Các cột dạng object được cố gắng chuyển đổi sang dạng số sau khi làm sạch.
   - Các đặc trưng mới được tạo ra như đã mô tả ở mục 4.4.1 (Binning, Tương tác, Tỷ lệ) để làm phong phú thêm thông tin cho mô hình.
   - Outliers trong các đặc trưng số được xử lý bằng phương pháp Winsorizing.
   - Các cột có tỷ lệ giá trị thiếu cao (ví dụ: >30-40%) sẽ bị loại bỏ để đảm bảo chất lượng dữ liệu.
3. **Lựa chọn đặc trưng bằng RFE (Tùy chọn):** Nếu được kích hoạt (`use_rfe = True`), RFE sẽ được sử dụng để tự động lựa chọn một số lượng đặc trưng (`n_features_to_select_rfe`, ví dụ 40 hoặc 80) dựa trên mô hình Gradient Boosting cơ sở. Điều này giúp giảm số chiều, loại bỏ nhiễu và có thể cải thiện khả năng tổng quát hóa của mô hình cuối cùng. Danh sách các đặc trưng được chọn bởi RFE sẽ được sử dụng để huấn luyện mô hình chính.

4.3.3 Lựa chọn mô hình (Model Selection)

Em sử dụng mô hình GradientBoostingRegressor (`sklearn.ensemble.GradientBoostingRegressor`) cho bài toán ước tính giá trị cầu thủ. Lựa chọn này dựa trên những ưu điểm sau:

- **Hiệu suất cao:** Gradient Boosting thường nằm trong nhóm các thuật toán cho kết quả dự đoán tốt nhất cho các bài toán hồi quy với dữ liệu dạng bảng.
- **Khả năng xử lý mối quan hệ phi tuyến:** Bằng cách xây dựng tuần tự các cây quyết định, mô hình này có thể nắm bắt các mối quan hệ phức tạp và tương tác giữa các đặc trưng mà các mô hình tuyến tính có thể bỏ qua.
- **Mạnh mẽ với các kiểu dữ liệu khác nhau:** Có thể hoạt động tốt với cả đặc trưng số và hạng mục (sau khi được mã hóa phù hợp).
- **Khả năng điều chỉnh (Regularization):** Cung cấp nhiều siêu tham số (như subsample, max_features, min_samples_leaf, learning_rate) giúp kiểm soát overfitting.
- **Hàm Loss linh hoạt:** Có thể chọn các hàm loss khác nhau (ví dụ: 'squared_error', 'huber') để phù hợp với đặc điểm của dữ liệu và giảm ảnh hưởng của outliers.

Việc tìm kiếm siêu tham số tối ưu cho GradientBoostingRegressor được thực hiện bằng `RandomizedSearchCV`. Kỹ thuật này duyệt qua một không gian các siêu tham số được định nghĩa trước (`param_dist`), thực hiện đánh giá chéo (`cv=5`) và chọn ra bộ tham số cho kết quả tốt nhất dựa trên chỉ số `scoring='r2'`.

4.3.4 Quy trình huấn luyện tổng thể và tiền xử lý trong Pipeline

Quy trình từ lúc có dữ liệu đến khi huấn luyện mô hình được đóng gói và tự động hóa:

- **Tải và kết hợp dữ liệu:** Nạp dữ liệu thống kê cầu thủ từ results.csv và dữ liệu giá trị chuyển nhượng đã lọc từ mục 4.2. Hai bộ dữ liệu này được hợp nhất (`merge`) dựa trên tên cầu thủ.
- **Chuẩn bị X và y:** Tách đặc trưng và biến mục tiêu, áp dụng log transform cho y.
- **Chia tập Train/Test:** Dữ liệu được chia với tỷ lệ 80% cho huấn luyện và 20% cho kiểm tra (`test_size=0.2`).
- **Xây dựng Preprocessor:** Một đối tượng `ColumnTransformer` được tạo ra để áp dụng các pipeline xử lý riêng biệt cho từng nhóm cột:
  - `numeric_pipeline`: Bao gồm `KNNImputer` (điền giá trị thiếu), `PolynomialFeatures(degree=2, interaction_only=True)` (tạo đặc trưng tương tác bậc hai giữa các biến số), và `StandardScaler` (chuẩn hóa z-score).
  - `categorical_pipeline`: Bao gồm `SimpleImputer` (điền giá trị thiếu bằng hằng số 'Missing' hoặc giá trị phổ biến nhất) và `OneHotEncoder` (chuyển đổi biến hạng mục thành các cột nhị phân, với `handle_unknown='infrequent_if_exist'` để xử lý các giá trị mới/hiếm và `min_frequency` để nhóm các hạng mục quá ít xuất hiện).

  Preprocessor này được `fit_transform` trên `X_train` và `transform` trên `X_test`.
- **Huấn luyện:**
  - Dữ liệu `X_train_processed` (sau khi qua preprocessor) được đưa vào `train_gbr_model`.
  - Nếu RFE được kích hoạt, `X_train_processed` sẽ được RFE lựa chọn đặc trưng trước.
  - `RandomizedSearchCV` được thực hiện để tìm siêu tham số tốt nhất và huấn luyện mô hình GradientBoostingRegressor cuối cùng.

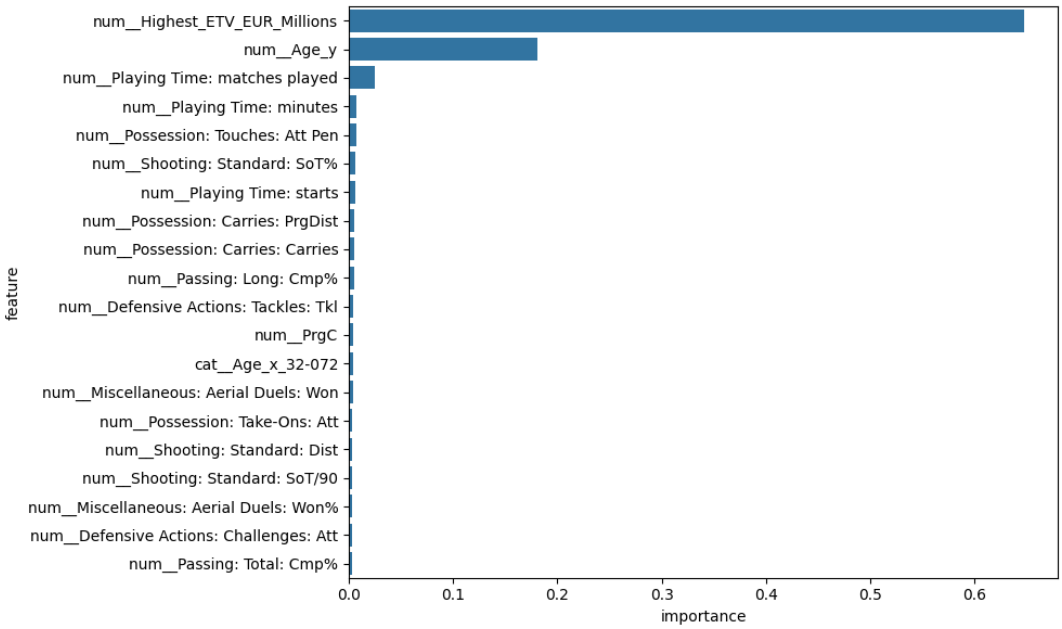
4.3.5 Kết quả và đánh giá

Sau khi thực hiện quy trình huấn luyện và đánh giá, em thu được các kết quả sau trên tập kiểm tra:

- R^2: 0.894
- RMSE :8.15
- MAE : 5.61

Các chỉ số này cho thấy mô hình có khả năng giải thích được khoảng 89.4% sự biến động trong giá trị chuyển nhượng của cầu thủ trên thang đo gốc. Giá trị RMSE cho biết sai số trung bình của dự đoán là khoảng 8.15 triệu EUR.

Biểu đồ Mức độ Quan trọng của Đặc trưng:

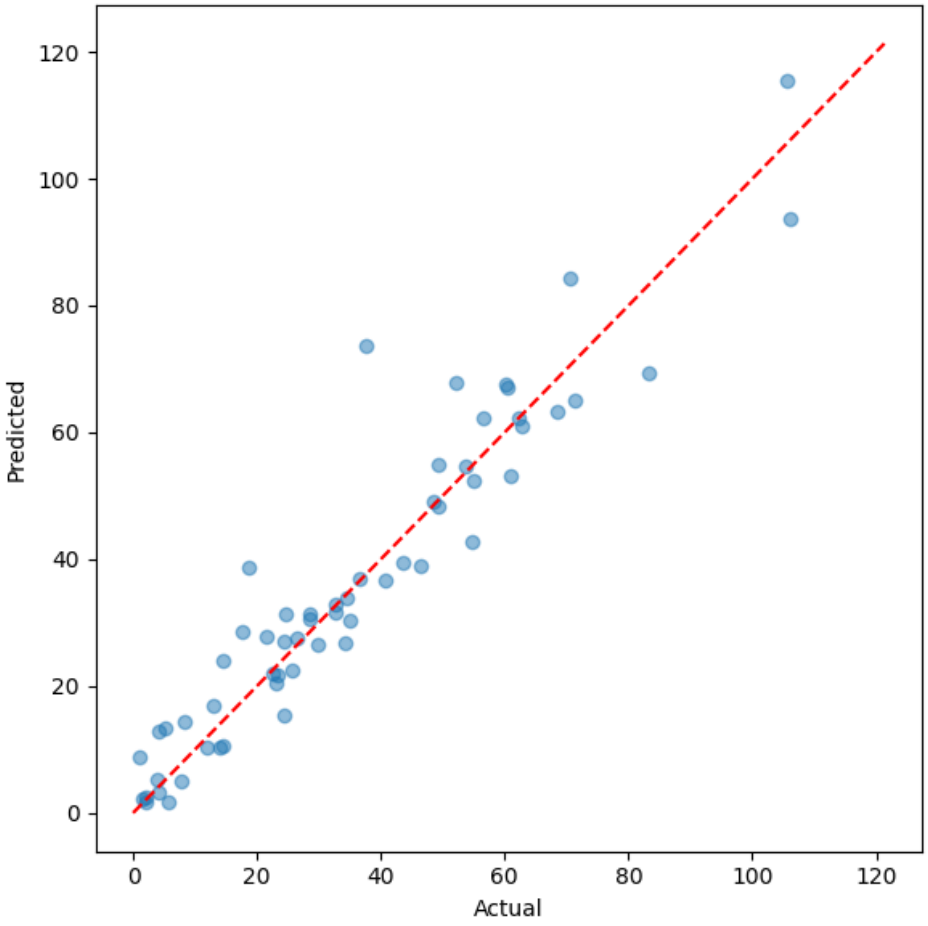


Hình 4.2: Mức độ quan trọng của đặc trưng.

Phân tích biểu đồ: Biểu đồ Mức độ Quan trọng của Đặc trưng (Hình 4.1) cho thấy những yếu tố có ảnh hưởng lớn nhất đến dự đoán giá trị cầu thủ của mô hình Gradient Boosting.

- **num_Highest_ETV_EUR_Millions** (Giá trị chuyển nhượng cao nhất lịch sử): Quan trọng nhất, phản ánh đẳng cấp và tiềm năng đã được công nhận của cầu thủ.
- **num_Age_y** (Tuổi): Rất quan trọng, ảnh hưởng đến tiềm năng phát triển và giai đoạn sự nghiệp, từ đó tác động đến giá trị.
- **num_Playing Time: matches played** (Số trận đã thi đấu): Cho thấy kinh nghiệm và mức độ đóng góp, ảnh hưởng đến nhận định giá trị.

Biểu đồ Giá trị Dự đoán và Thực tế:

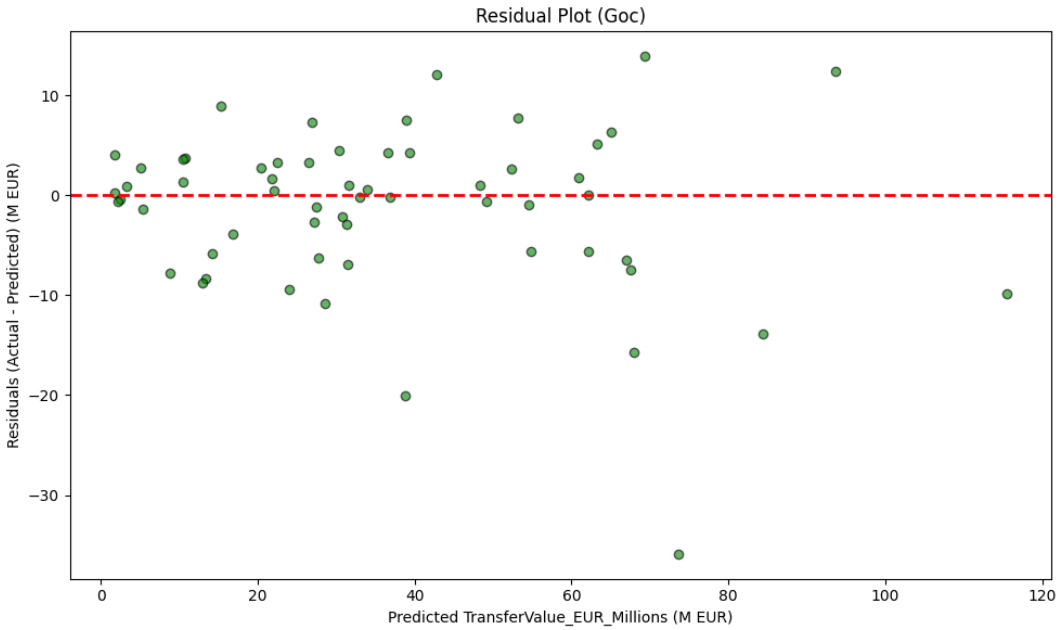


Hình 4.3: Biểu đồ Giá trị Dự đoán và Thực .tế

Phân tích biểu đồ: Hình 4.2 thể hiện mối quan hệ giữa giá trị chuyển nhượng thực tế của cầu thủ (trục hoành) và giá trị được dự đoán bởi mô hình (trục tung).

- Xu hướng chung: Các điểm dữ liệu có xu hướng phân bổ quanh đường chéo y=x. Điều này cho thấy mô hình nắm bắt được phần lớn mối mối quan hệ giữa các đặc trưng và giá trị chuyển nhượng, dự đoán có độ tương quan dương với giá trị thực.
- Khoảng dự đoán tốt: Mô hình thể hiện khả năng dự đoán tốt nhất đối với các cầu thủ có giá trị thực tế trong khoảng từ 0 đến khoảng 80 triệu Euro. Trong phạm vi này, các điểm dữ liệu nằm gần đường chéo y=x hơn, với độ lệch thấp.
- Dự đoán ở giá trị thấp: Đối với một số cầu thủ có giá trị thực tế rất thấp (gần 0), mô hình có vẻ dự đoán giá trị hơi cao hơn một chút so với thực tế (overestimation), thể hiện qua một vài điểm nằm phía trên đường chéo.
- Dự đoán ở giá trị cao: Với các cầu thủ có giá trị thực tế cao (ví dụ, trên 80-100 triệu Euro), mô hình có xu hướng dự đoán giá trị thấp hơn so với thực tế (underestimation). Các điểm dữ liệu ở vùng này thường nằm phía dưới đường chéo y=x.

Biểu đồ Phần dư (Residuals Plot):



Hình 4.4: Biểu đồ Phần dư

Phân tích biểu đồ: Biểu đồ Phần dư (Hình 4.4) thể hiện sai số giữa giá trị thực tế và giá trị dự đoán cho một điểm dữ liệu.

- Phân bố quanh 0: Phần lớn các điểm phần dư phân bố khá ngẫu nhiên quanh đường 0, cho thấy mô hình không có thiên lệch hệ thống rõ ràng.
- Khuôn mẫu (Heteroscedasticity): Không có hình phễu rõ rệt, tuy nhiên phương sai của phần dư có vẻ hơi tăng nhẹ ở các giá trị dự đoán cao hơn (ví dụ, từ 60 triệu EUR trở lên), ngụ ý độ không chắc chắn của dự đoán có thể lớn hơn cho cầu thủ giá trị cao.
- Ngoại lệ: Vẫn có một vài điểm phần dư lớn, cho thấy các trường hợp cá biệt mà mô hình dự đoán chưa tốt.

4.3.6 Đánh giá tổng thể phương pháp đề xuất

Phương pháp mà em đã đề xuất để ước tính giá trị cầu thủ, bao gồm các bước từ thu thập dữ liệu, kết hợp dữ liệu thống kê, tiền xử lý kỹ lưỡng, tạo đặc trưng nâng cao (bao gồm binning, tương tác, tỷ lệ, và đặc trưng đa thức), tùy chọn lựa chọn đặc trưng bằng RFE, và sử dụng mô hình Gradient Boosting Regressor với việc tối ưu hóa siêu tham số bằng RandomizedSearchCV, đã cho thấy những kết quả khả quan.

Điểm mạnh của phương pháp:

- Quy trình tiền xử lý dữ liệu được thiết kế để xử lý các vấn đề phổ biến như giá trị thiếu (sử dụng KNNImputer, SimpleImputer), chuẩn hóa (StandardScaler), và mã hóa biến hạng mục (OneHotEncoder).
- Việc tạo đặc trưng (Feature Engineering) với PolynomialFeatures và các tương tác thủ công giúp mô hình có khả năng nắm bắt các mối quan hệ phức tạp hơn trong dữ liệu.
- Gradient Boosting là một thuật toán mạnh mẽ, và việc sử dụng RandomizedSearchCV giúp tìm ra bộ siêu tham số tốt, tối ưu cho chỉ số R^2.
- Việc biến đổi log biến mục tiêu và xử lý outliers giúp cải thiện sự ổn định và hiệu suất của mô hình.

Kết quả: Với chỉ số R^2 trên thang đo gốc đạt 0.894 - đây là một kết quả tốt, cho thấy mô hình đã thể hiện được khả năng giải thích một phần đáng kể sự biến động của giá trị cầu thủ. Các biểu đồ trực quan hóa cũng cung cấp cái nhìn sâu sắc về hiệu suất và các yếu tố ảnh hưởng đến dự đoán.

Tài liệu tham khảo

[1] GeeksforGeeks. (2025, March 11). *ML / Gradient Boosting*.

[2] Rawat, A. (2025, February 16). *Clustering and dimensionality reduction techniques to simplify complex data*. Interview Kickstart.

[3] Scikit-learn Development Team. (2025). *Scikit-learn: Machine Learning in Python*

[4] Scikit-learn Development Team. (2025). *sklearn.impute.KNNImputer*. Scikit-learn User Guide.

[5] Selenium Committers. (2025). *Selenium WebDriver Documentation*.