BỘ THÔNG TIN VÀ TRUYỀN THÔNG

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

------📖-----

BÁO CÁO BÀI TẬP LỚN

MÔN: LẬP TRÌNH PYTHON

Giảng viên hướng dẫn:	Kim Ngọc Bách
Sinh viên:	Nguyễn Việt Anh
Mã sinh viên:	B23DCCE009
Lớp:	D23CQCE06-B
Niên khoá:	2023-2028
Hệ đào tạo:	Đại học chính quy

Hà Nội, Tháng 5/2025

Mục lục

Chương trình 4.1: Thu thập phí chuyển nhượng thực tế từ trang web

- **Mục đích chính:** Đọc dữ liệu cầu thủ từ tệp `result.csv`, lọc ra những cầu thủ có số phút thi đấu nhất định, sau đó truy cập trang web `footballtransfers.com` để tìm kiếm và thu thập phí chuyển nhượng đã *được xác nhận* cho mùa giải 2024-2025. Cuối cùng, chương trình ghép nối thông tin phí chuyển nhượng này với dữ liệu cầu thủ ban đầu và lưu vào một tệp CSV mới.
- **Các bước hoạt động chi tiết:**

1.Nhập thư viện

Đây là bước nhập các công cụ cần thiết từ các thư viện phổ biến trong Python cho khoa học dữ liệu, xử lý file và web scraping.

Python

```
import pandas as pd # Để làm việc với cấu trúc dữ liệu bảng (DataFrame)
import os # Để tương tác với hệ thống file
from fuzzywuzzy import fuzz, process # Để so khớp chuỗi mờ
from selenium import webdriver # Để tự động hóa trình duyệt web
from selenium.webdriver.chrome.service import Service # Quản lý service của ChromeDriver
from selenium.webdriver.common.by import By # Xác định vị trí phần tử trên web
from selenium.webdriver.chrome.options import Options # Cấu hình tùy chọn trình duyệt
from selenium.webdriver.support.ui import WebDriverWait # Chờ điều kiện xảy ra trên web
from selenium.webdriver.support import expected_conditions as EC # Các điều kiện chờ đợi phổ biến
from webdriver_manager.chrome import ChromeDriverManager # Tự động tải và quản lý ChromeDriver
```

2.Định nghĩa đường dẫn file

Các biến này lưu trữ đường dẫn đến tệp CSV đầu vào, thư mục và tệp đầu ra.

Python

```
# Đường dẫn file CSV
csv_file = r'D:\python project\report\csv\result.csv'
output_dir = r'D:\python project\report\csv'
output_file = os.path.join(output_dir, 'transfer_fee.csv')
```

3.Hàm shorten_name(name)

Hàm này rút gọn tên cầu thủ, giúp quá trình so khớp tên giữa dữ liệu cục bộ và dữ liệu web chính xác hơn.

Python

```
# Hàm rút gọn tên cầu thủ
def shorten_name(name):
    """Rút gọn tên thành 2 từ đầu tiên."""
```

```
    parts = name.strip().split()
    return " ".join(parts[:2]) if len(parts) >= 2 else name
```

4.Hàm is_valid_transfer_fee(fee)

Hàm này kiểm tra xem chuỗi phí chuyển nhượng có chứa các ký hiệu tiền tệ hoặc là "Free" hay không để xác định đây là một giá trị hợp lệ cần thu thập.

Python

```
# Hàm kiểm tra phí chuyển nhượng hợp lệ
def is_valid_transfer_fee(fee):
    """Kiểm tra nếu phí chuyển nhượng là giá trị tiền tệ hoặc
Free."""
    if not fee or fee.lower() in ["n/a", "not found", ""]:
        return False
    return fee.lower() == "free" or any(currency in fee for
currency in ["€", "£", "$"])
```

5.Hàm load_players_over_900_minutes(csv_file)

Hàm này đọc dữ liệu từ tệp CSV đầu vào, xử lý các giá trị thiếu (`na_values="N/A"`) và lọc ra những cầu thủ có số phút thi đấu (`Minutes`) lớn hơn 900.

Python

```
# Hàm đọc và lọc cầu thủ từ result.csv
def load_players_over_900_minutes(csv_file):
    """Đọc file CSV và lọc cầu thủ có hơn 900 phút thi đấu."""
    try:
        df = pd.read_csv(csv_file, encoding='utf-8-sig',
na_values=["N/A"])
        print(f"Đã đọc file CSV: {csv_file}")
        print(f"Số dòng: {df.shape[0]}, Số cột: {df.shape[1]}")

        # Lọc cầu thủ có hơn 900 phút
        df_filtered = df[df['Minutes'] > 900][['Player',
'Team', 'Minutes']].copy()
        print(f"Số cầu thủ thi đấu > 900 phút:
{len(df_filtered)}")

        return df_filtered
    except FileNotFoundError:
        print(f"Không tìm thấy file {csv_file}. Vui lòng kiểm
tra đường dẫn.")
        exit() # Thoát chương trình nếu file không tìm thấy
    except Exception as e:
        print(f"Lỗi khi đọc file CSV: {e}")
        exit() # Thoát chương trình nếu có lỗi đọc file
```

6.Hàm scrape_transfer_values(players)

Đây là phần cốt lõi thực hiện việc cào dữ liệu web.

Python

```
# Hàm thu thập giá trị chuyển nhượng từ web
```

```
def scrape_transfer_values(players):
    """Thu thập giá trị chuyển nhượng từ
footballtransfers.com."""
    # Cấu hình trình duyệt headless (chạy ẩn, không mở cửa sổ)
    options = Options()
    options.add_argument("--headless")
    options.add_argument("--no-sandbox")
    options.add_argument("--disable-dev-shm-usage")

    # Khởi tạo ChromeDriver, sử dụng webdriver_manager để tự
động tải
    driver =
webdriver.Chrome(service=Service(ChromeDriverManager().install(
)), options=options)

    # Tạo danh sách URL từ trang 1 đến 14 của mục confirmed
transfers
    base_url =
"https://www.footballtransfers.com/us/transfers/confirmed/2024-
2025/uk-premier-league/"
    urls = [f"{base_url}{i}" for i in range(1, 15)]

    # Tạo danh sách tên rút gọn từ dữ liệu cục bộ để so khớp
    player_names = [shorten_name(name) for name in
players['Player'].str.strip()]

    # Tạo dict để tra cứu thông tin cầu thủ gốc (Team, Minutes)
dựa vào tên đầy đủ
    player_info = dict(zip(players['Player'].str.strip(),
zip(players['Team'], players['Minutes'])))

    transfer_results = [] # Danh sách lưu kết quả tìm thấy
    not_found = [] # Danh sách lưu cầu thủ không tìm thấy/không
khớp

    try:
        # Lặp qua từng URL để cào dữ liệu
        for url in urls:
            print(f"Đang cào dữ liệu từ: {url}")
            driver.get(url) # Tải trang web

            try:
                # Đợi bảng chuyển nhượng xuất hiện trên trang
(để đảm bảo nội dung đã tải)
                table = WebDriverWait(driver, 10).until(

EC.presence_of_element_located((By.CLASS_NAME, "transfer-
table"))
                )

                # Tìm tất cả các hàng trong bảng
                rows = table.find_elements(By.TAG_NAME, "tr")

                # Lặp qua từng hàng để trích xuất dữ liệu
                for row in rows:
                    cols = row.find_elements(By.TAG_NAME, "td")
# Tìm các cột trong hàng
                    if cols and len(cols) >= 3: # Đảm bảo hàng
có đủ cột cần thiết
                        # Lấy tên cầu thủ từ cột đầu tiên, xử
lý trường hợp tên có xuống dòng
                        player_name =
cols[0].text.strip().split("\n")[0].strip()
```

```
                        shortened_player_name =
shorten_name(player_name) # Rút gọn tên cào được

                        # Lấy phí chuyển nhượng từ cột cuối
cùng
                        transfer_fee = cols[-1].text.strip() if
cols[-1].text.strip() else "N/A"

                        # Lấy loại chuyển nhượng (giả định cột
thứ 2 chứa thông tin loại)
                        transfer_type = cols[1].text.strip() if
len(cols) > 1 and cols[1].text.strip() else "Unknown"

                        # Sử dụng fuzzywuzzy để so khớp tên cầu
thủ cào được với danh sách tên rút gọn cục bộ
                        best_match =
process.extractOne(shortened_player_name, player_names,

scorer=fuzz.token_sort_ratio) # Dùng thuật toán
token_sort_ratio

                        # Nếu tìm thấy khớp tốt (độ tương đồng
>= 85)
                        if best_match and best_match[1] >= 85:
                            matched_name = best_match[0] # Tên
rút gọn đã khớp từ danh sách cục bộ
                            # Tìm tên đầy đủ tương ứng trong dữ
liệu gốc
                            for full_name, short_name in
zip(players['Player'].str.strip(), player_names):
                                if short_name == matched_name:
                                    team, minutes =
player_info[full_name] # Lấy thông tin Team, Minutes
                                    # Chỉ thêm vào kết quả nếu
phí chuyển nhượng hợp lệ
                                    if
is_valid_transfer_fee(transfer_fee):

transfer_results.append({
                                            'Player':
full_name, # Tên đầy đủ gốc
                                            'Team': team,
                                            'Minutes': minutes,
                                            'Transfer Fee':
transfer_fee,
                                            'Transfer Type':
transfer_type
                                        })
                                    else:

not_found.append(full_name) # Không tìm thấy phí hợp lệ
                                    break # Đã tìm thấy tên đầy
đủ, thoát vòng lặp nội bộ
                        # Nếu không tìm thấy khớp tốt (không có
best_match hoặc score < 85)
                        else:
                            # Cần xử lý trường hợp này nếu muốn
ghi lại cầu thủ cào được nhưng không khớp
                            pass # Hiện tại không làm gì nếu
không khớp tốt

            except Exception as e:
                print(f"Lỗi khi xử lý {url}: {str(e)}") # Báo
cáo lỗi nhưng tiếp tục với URL khác
```

```
        # Sau khi cào hết các trang, tìm các cầu thủ trong danh
sách ban đầu mà không được xử lý
        found_players = {result['Player'] for result in
transfer_results}
        for player in players['Player']:
            if player not in found_players and player not in
not_found:
                not_found.append(player) # Thêm vào danh sách
không tìm thấy
    finally:
        driver.quit() # Đảm bảo đóng trình duyệt

    return transfer_results, not_found # Trả về kết quả và danh
sách không tìm thấy
```

7.Hàm save_results_to_csv(transfer_results, not_found, total_players)

Hàm này tạo thư mục đầu ra (nếu cần), chuyển kết quả tìm thấy thành DataFrame và lưu vào tệp CSV. Cuối cùng, nó in ra bảng tóm tắt thống kê về quá trình tìm kiếm.

Python

```
# Hàm lưu kết quả vào file CSV
def save_results_to_csv(transfer_results, not_found,
total_players):
    """Lưu kết quả vào file CSV."""
    os.makedirs(output_dir, exist_ok=True) # Tạo thư mục đầu ra
nếu chưa tồn tại

    # Chuyển kết quả thành DataFrame
    df_results = pd.DataFrame(transfer_results)

    try:
        if not df_results.empty:
            df_results.to_csv(output_file, index=False,
encoding='utf-8-sig') # Lưu file CSV
            print(f"Kết quả đã được lưu vào {output_file} với
{len(df_results)} cầu thủ")
        else:
            print(f"Không có cầu thủ nào thỏa mãn điều kiện.
Không tạo file {output_file}")
    except Exception as e:
        print(f"Lỗi khi ghi file CSV: {e}")

    # In thống kê ra console
    output_content = "Phân tích giá trị chuyển nhượng cầu thủ
Premier League 2024-2025 (Thi đấu > 900 phút và có giá trị
chuyển nhượng)\n"
    output_content += "=" * 80 + "\n\n"

    output_content += "Thống kê:\n"
    output_content += f"- Số cầu thủ thi đấu > 900 phút:
{total_players}\n"
    output_content += f"- Số cầu thủ tìm thấy giá trị chuyển
nhượng (bao gồm Free): {len(transfer_results)}\n"
    output_content += f"- Số cầu thủ không tìm thấy giá trị
chuyển nhượng: {len(not_found)}\n"
```

```
    print(output_content)
```

8.Hàm main()

Hàm này là điểm điều phối chính, gọi tuần tự các hàm con để thực hiện toàn bộ quy trình.

Python

```
# Hàm chính
def main():
    # Đọc và lọc cầu thủ từ result.csv
    players = load_players_over_900_minutes(csv_file)
    total_players = len(players) # Tổng số cầu thủ sau khi lọc
phút thi đấu

    # Thu thập giá trị chuyển nhượng từ web
    transfer_results, not_found =
scrape_transfer_values(players)

    # Lưu kết quả vào file CSV và in thống kê
    save_results_to_csv(transfer_results, not_found,
total_players)
```

9.Khối if __name__ == "__main__":

Điểm bắt đầu thực thi khi script được chạy trực tiếp.

Python

```
if __name__ == "__main__":
    main()
```

- **Tóm tắt:** Chương trình 4.1 là một script cào dữ liệu có mục tiêu cụ thể là thu thập các giao dịch chuyển nhượng đã hoàn tất từ một trang web. Nó sử dụng Selenium để tương tác với trang web động và `fuzzywuzzy` để giải quyết vấn đề khớp tên không chính xác giữa dữ liệu cục bộ và dữ liệu trên web. Kết quả được lưu vào một tệp CSV để phân tích thêm.

Chương trình 4.2: Ước tính giá trị chuyển nhượng cầu thủ bằng Học máy

- **Mục đích chính:** Đọc dữ liệu cầu thủ từ `result.csv`, truy cập một phần khác của trang web `footballtransfers.com` để thu thập *ước tính giá trị chuyển nhượng (Estimated Transfer Value - ETV)* của các cầu thủ Premier League. Chương trình sau đó ghép nối ETV này với dữ liệu thống kê cầu thủ cục bộ, xây dựng các mô hình Học máy (Hồi quy Tuyến tính) *riêng biệt* cho từng vị trí chơi (Thủ môn, Hậu vệ, Tiền vệ, Tiền đạo) để dự đoán giá trị chuyển nhượng dựa trên các chỉ số hiệu suất. Cuối cùng, nó lưu kết quả (ETV thực tế và ETV dự đoán) vào một tệp CSV mới.
- **Các bước hoạt động chi tiết:**

1.Nhập thư viện

Tương tự chương trình 4.1, cộng thêm các thư viện của `sklearn` cho học máy.

Python

```
import pandas as pd
import numpy as np
import os
import re # Để làm việc với biểu thức chính quy
from fuzzywuzzy import process, fuzz
from sklearn.model_selection import train_test_split # Chia dữ
liệu
from sklearn.linear_model import LinearRegression # Mô hình hồi
quy tuyến tính
from sklearn.preprocessing import StandardScaler, OneHotEncoder
# Tiền xử lý: chuẩn hóa, mã hóa One-Hot
from sklearn.compose import ColumnTransformer # Áp dụng các
bước tiền xử lý khác nhau cho các cột khác nhau
from sklearn.pipeline import Pipeline # Kết hợp các bước tiền
xử lý và mô hình
from sklearn.metrics import mean_squared_error, r2_score # Đánh
giá mô hình
# Các thư viện Selenium và webdriver_manager tương tự 4.1
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as
EC
from webdriver_manager.chrome import ChromeDriverManager
```

2.Định nghĩa đường dẫn file

Đường dẫn đến file input và output.

Python

```
# Đường dẫn thư mục và file
csv_dir = r"D:\python project\report\csv"
result_path = os.path.join(csv_dir, "result.csv") # File input
output_path = os.path.join(csv_dir,
"estimate_transfer_fee.csv") # File output
```

3.standard_output_columns

Danh sách các cột chuẩn mà file CSV đầu ra mong muốn có.

Python

```
# Các cột đầu ra chuẩn
standard_output_columns = [
    'Player', 'Team', 'Nation', 'Position',
'Actual_Transfer_Value_M', 'Predicted_Transfer_Value_M'
]
```

4.positions_config

Dictionary quan trọng định nghĩa cấu hình cho từng vị trí, bao gồm cách lọc cầu thủ và danh sách các chỉ số (features) sẽ được sử dụng để huấn luyện mô hình cho vị trí đó.

Python

```
# Cấu hình cho các vị trí
positions_config = {
    'GK': { # Thủ môn
        'position_filter': 'GK',
        'features': [
            'Save%', 'CS%', 'GA90', 'Minutes', 'Age', 'PK
Save%', 'Team', 'Nation'
        ],
        'important_features': ['Save%', 'CS%', 'PK Save%'] #
Các chỉ số quan trọng hơn
    },
    'DF': { # Hậu vệ
        'position_filter': 'DF',
        'features': [
            'Tkl', 'TklW', 'Int', 'Blocks', 'Recov', 'Minutes',
'Team', 'Age', 'Nation', 'Aerl Won%',
            'Aerl Won', 'Cmp', 'Cmp%', 'PrgP', 'LongCmp%',
'Carries', 'Touches', 'Dis', 'Mis'
        ],
        'important_features': ['Tkl', 'TklW', 'Int', 'Blocks',
'Aerl Won%', 'Aerl Won', 'Recov']
    },
    'MF': { # Tiền vệ
        'position_filter': 'MF',
        'features': [
            'Cmp%', 'KP', 'PPA', 'PrgP', 'Tkl', 'Ast', 'SCA',
'Touches', 'Minutes', 'Team', 'Age', 'Nation',
            'Pass into 1_3', 'xAG', 'Carries 1_3', 'ProDist',
'Rec', 'Mis', 'Dis'
        ],
        'important_features': ['KP', 'PPA', 'PrgP', 'SCA',
'xAG', 'Pass into 1_3', 'Carries 1_3']
    },
    'FW': { # Tiền đạo
        'position_filter': 'FW',
        'features': [
            'Gls', 'Ast', 'Gls per 90', 'xG per 90', 'SoT%', 'G
per Sh', 'SCA90', 'GCA90',
            'PrgC', 'Carries 1_3', 'Aerl Won%', 'Team', 'Age',
'Minutes'
        ],
        'important_features': ['Gls', 'Ast', 'Gls per 90', 'xG
per 90', 'SCA90', 'GCA90']
    }
}
```

5.Hàm shorten_name(name)

Hàm rút gọn tên, có thêm một số trường hợp đặc biệt được xử lý riêng.

Python

```
# Hàm rút ngắn tên
def shorten_name(name):
    if name == "Manuel Ugarte Ribeiro": return "Manuel Ugarte"
    elif name == "Igor Júlio": return "Igor"
    elif name == "Igor Thiago": return "Thiago"
    elif name == "Felipe Morato": return "Morato"
    elif name == "Nathan Wood-Gordon": return "Nathan Wood"
    elif name == "Bobby Reid": return "Bobby Cordova-Reid"
    elif name == "J. Philogene": return "Jaden Philogene
Bidace"
    parts = name.strip().split(" ")
    # Lấy từ đầu tiên và từ cuối cùng cho tên có 3 từ trở lên,
ngược lại giữ nguyên
    return parts[0] + " " + parts[-1] if len(parts) >= 3 else
name
```

6.Hàm parse_etv(etv_text)

Hàm chuyển đổi chuỗi ETV (có ký hiệu tiền tệ, 'M', 'K') thành giá trị số thực.

Python

```
# Hàm chuyển đổi giá trị chuyển nhượng (ETV) từ chuỗi sang số
def parse_etv(etv_text):
    if pd.isna(etv_text) or etv_text in ["N/A", ""]:
        return np.nan # Trả về NaN nếu giá trị thiếu hoặc không
hợp lệ
    try:
        # Loại bỏ ký hiệu tiền tệ và khoảng trắng, chuyển sang
chữ hoa
        etv_text = re.sub(r'[€£]', '',
etv_text).strip().upper()
        # Xác định hệ số nhân dựa trên 'M' (triệu) hoặc 'K'
(ngàn)
        multiplier = 1000000 if 'M' in etv_text else 1000 if
'K' in etv_text else 1
        # Loại bỏ 'M', 'K' và chuyển phần còn lại thành float,
sau đó nhân với hệ số
        value = float(re.sub(r'[MK]', '', etv_text)) *
multiplier
        return value
    except (ValueError, TypeError):
        return np.nan # Trả về NaN nếu không thể parse
```

7.Hàm fuzzy_match_name(name, choices, score_threshold=80)

Hàm so khớp tên mờ, sử dụng tên rút gọn và trả về cả tên đã khớp và điểm số tương đồng.

Python

```
# Hàm so khớp tên mờ
def fuzzy_match_name(name, choices, score_threshold=80):
    if not isinstance(name, str): # Bỏ qua nếu tên không phải
chuỗi
        return None, None
    shortened_name = shorten_name(name).lower() # Rút gọn và
chuyển tên cần khớp về chữ thường
    # Rút gọn và chuyển danh sách các lựa chọn về chữ thường
```

```
    shortened_choices = [shorten_name(c).lower() for c in
choices if isinstance(c, str)]
    # Tìm tên rút gọn có độ tương đồng cao nhất trong danh sách
lựa chọn rút gọn
    match = process.extractOne(
        shortened_name,
        shortened_choices,
        scorer=fuzz.token_sort_ratio, # Thuật toán so khớp
        score_cutoff=score_threshold # Ngưỡng điểm tối thiểu để
chấp nhận là khớp
    )
    if match is not None:
        # Nếu có khớp, tìm chỉ mục của tên rút gọn đã khớp
trong danh sách rút gọn
        matched_idx = shortened_choices.index(match[0])
        # Trả về tên gốc đầy đủ từ danh sách choices ban đầu và
điểm số
        return choices[matched_idx], match[1]
    return None, None # Trả về None nếu không tìm thấy khớp tốt
```

8.Hàm scrape_transfer_values()

Hàm cào dữ liệu ETV từ trang web.

Python

```
# Hàm cào dữ liệu giá trị chuyển nhượng ước tính (ETV) từ web
def scrape_transfer_values():
    # Đọc dữ liệu cầu thủ cục bộ để lấy danh sách tên và vị trí
    df_players = pd.read_csv(result_path)
    # Tạo dict để tra cứu vị trí và tên đầy đủ gốc từ tên rút
gọn
    player_positions =
dict(zip(df_players['Player'].str.strip().apply(shorten_name),
df_players['Position']))
    player_original_names =
dict(zip(df_players['Player'].str.strip().apply(shorten_name),
df_players['Player'].str.strip()))
    player_names = list(player_positions.keys()) # Danh sách
tên rút gọn để so khớp

    # Cấu hình và khởi tạo Selenium tương tự 4.1
    options = Options()
    options.add_argument("--headless")
    options.add_argument("--no-sandbox")
    options.add_argument("--disable-dev-shm-usage")
    driver =
webdriver.Chrome(service=Service(ChromeDriverManager().install(
)), options=options)

    # Tạo danh sách URL cho trang danh sách cầu thủ Premier
League
    base_url =
"https://www.footballtransfers.com/us/players/uk-premier-
league/"
    urls = [f"{base_url}{i}" for i in range(1, 23)] # Cào từ
trang 1 đến 22
    all_data = [] # Danh sách lưu dữ liệu cào được

    try:
        for url in urls:
            driver.get(url) # Tải trang web
```

```
            print(f"Scraping: {url}")
            try:
                # Chờ bảng dữ liệu cầu thủ xuất hiện
                table = WebDriverWait(driver, 10).until(
EC.presence_of_element_located((By.CLASS_NAME, "similar-
players-table"))
                )
                rows = table.find_elements(By.TAG_NAME, "tr") #
Tìm các hàng

                for row in rows:
                    cols = row.find_elements(By.TAG_NAME, "td")
# Tìm các cột
                    # Đảm bảo hàng có đủ cột và không phải hàng
tiêu đề
                    if cols and len(cols) >= 2 and
cols[1].text.strip().split("\n")[0].strip() != 'Name':
                        player_name =
cols[1].text.strip().split("\n")[0].strip() # Lấy tên cầu thủ
cào được
                        shortened_player_name =
shorten_name(player_name) # Rút gọn tên cào được
                        etv = cols[-1].text.strip() if
len(cols) >= 3 else "N/A" # Lấy ETV từ cột cuối

                        # So khớp tên rút gọn cào được với danh
sách tên rút gọn cục bộ
                        best_match =
process.extractOne(shortened_player_name, player_names,
scorer=fuzz.token_sort_ratio)

                        if best_match and best_match[1] >= 80:
# Nếu có khớp tốt (score >= 80)
                            matched_name = best_match[0] # Tên
rút gọn đã khớp
                            # Lấy tên đầy đủ gốc và vị trí từ
dữ liệu cục bộ
                            original_name =
player_original_names.get(matched_name, matched_name)
                            position =
player_positions.get(matched_name, "Unknown")
                            print(f"Match found: {player_name}
-> {original_name} (score: {best_match[1]}, Position:
{position})")
                            # Thêm thông tin vào danh sách
                            all_data.append([original_name,
position, etv])
                        else:
                            print(f"No match for: {player_name}
(best match: {best_match[0] if best_match else 'None'}, score:
{best_match[1] if best_match else 'N/A'})")

            except Exception as e:
                print(f"Error processing {url}: {str(e)}") #
Báo cáo lỗi nhưng tiếp tục

    finally:
        driver.quit() # Đảm bảo đóng trình duyệt

    if all_data:
        df_all = pd.DataFrame(all_data, columns=['Player',
'Position', 'Price']) # Tạo DataFrame từ dữ liệu cào được
        print("Dữ liệu cào được đã sẵn sàng để xử lý.")
```

```
        return df_all
    else:
        print("No matching players found.")
        return pd.DataFrame(columns=['Player', 'Position',
'Price']) # Trả về DataFrame rỗng nếu không có dữ liệu
```

9.Hàm process_position(position, config, df_etv)

Hàm xử lý dữ liệu, huấn luyện mô hình và dự đoán cho MỘT vị trí cụ thể.

Python

```
# Hàm xử lý dữ liệu theo vị trí, xây dựng và huấn luyện mô hình
ML
def process_position(position, config, df_etv):
    try:
        df_result = pd.read_csv(result_path) # Tải dữ liệu cầu
thủ gốc
    except FileNotFoundError as e:
        print(f"Lỗi: Không tìm thấy tệp result.csv - {e}")
        return None, None # Trả về None nếu không tìm thấy file

    # Tạo cột vị trí chính và lọc DataFrame theo vị trí đang xử
lý
    df_result['Primary_Position'] =
df_result['Position'].astype(str).str.split(r'[,/]').str[0].str
.strip()
    df_result =
df_result[df_result['Primary_Position'].str.upper() ==
config['position_filter'].upper()].copy()

    player_names = df_etv['Player'].dropna().tolist() # Danh
sách tên cầu thủ từ dữ liệu ETV đã cào
    df_result['Matched_Name'] = None
    df_result['Match_Score'] = None
    df_result['ETV'] = np.nan # Cột lưu ETV dạng số, khởi tạo
là NaN

    # Ghép nối ETV từ dữ liệu cào được vào DataFrame cầu thủ
gốc theo vị trí
    for idx, row in df_result.iterrows():
        matched_name, score = fuzzy_match_name(row['Player'],
player_names) # So khớp tên cầu thủ gốc với danh sách tên từ
ETV
        if matched_name:
            df_result.at[idx, 'Matched_Name'] = matched_name
            df_result.at[idx, 'Match_Score'] = score
            # Tìm dòng tương ứng trong dữ liệu ETV đã cào
            matched_row = df_etv[df_etv['Player'] ==
matched_name]
            if not matched_row.empty:
                etv_value =
parse_etv(matched_row['Price'].iloc[0]) # Chuyển đổi ETV từ
chuỗi sang số
                df_result.at[idx, 'ETV'] = etv_value # Gán ETV
số vào cột 'ETV'

    # Tạo DataFrame chỉ chứa cầu thủ đã khớp ETV thành công
    df_filtered =
df_result[df_result['Matched_Name'].notna()].copy()
    # Loại bỏ các dòng trùng lặp dựa trên tên đã khớp (nếu có
cầu thủ có nhiều entry trong result.csv?)
```

```
    df_filtered =
df_filtered.drop_duplicates(subset='Matched_Name')
    # Xác định các cầu thủ ban đầu theo vị trí này mà không tìm
thấy khớp ETV
    unmatched =
df_result[df_result['Matched_Name'].isna()]['Player'].dropna().
tolist()
    if unmatched:
        print(f"Cầu thủ {position} không khớp: {len(unmatched)}
cầu thủ.")
        # print(unmatched) # Có thể in danh sách cụ thể nếu cần

    features = config['features'] # Danh sách các chỉ số
(features) cho vị trí này
    target = 'ETV' # Cột mục tiêu là ETV

    # Tiền xử lý các cột đặc trưng (features)
    for col in features:
        if col in ['Team', 'Nation']:
            df_filtered[col] =
df_filtered[col].fillna('Unknown') # Điền 'Unknown' cho NaN
trong cột phân loại
        else:
            # Chuyển cột sang kiểu số, lỗi chuyển đổi thành NaN
            df_filtered[col] = pd.to_numeric(df_filtered[col],
errors='coerce')
            # Điền NaN trong cột số bằng giá trị trung vị
            median_value = df_filtered[col].median()
            df_filtered[col] =
df_filtered[col].fillna(median_value if not
pd.isna(median_value) else 0)

    # Áp dụng biến đổi logarit cho các cột số (trừ Team,
Nation, Age)
    numeric_features = [col for col in features if col not in
['Team', 'Nation']]
    for col in numeric_features:
         # Áp dụng log(1+x) để xử lý các giá trị bằng 0 hoặc
phân phối lệch, cắt giá trị âm về 0 trước khi log
        df_filtered[col] =
np.log1p(df_filtered[col].clip(lower=0))

    # Áp dụng trọng số cho các đặc trưng quan trọng và
tuổi/phút
    for col in config['important_features']:
        if col in df_filtered.columns:
            df_filtered[col] = df_filtered[col] * 2.0 # Nhân
đôi trọng số cho đặc trưng quan trọng
    if 'Minutes' in df_filtered.columns:
        df_filtered['Minutes'] = df_filtered['Minutes'] * 1.5 #
Tăng trọng số cho Minutes
    if 'Age' in df_filtered.columns:
        df_filtered['Age'] = df_filtered['Age'] * 0.5 # Giảm
trọng số cho Age

    # Chuẩn bị dữ liệu cho mô hình ML: chỉ lấy các dòng có ETV
hợp lệ
    df_ml = df_filtered.dropna(subset=[target]).copy()
    if df_ml.empty:
        print(f"Lỗi: Không có dữ liệu ETV hợp lệ cho
{position}.")
        return None, unmatched # Trả về None nếu không có dữ
liệu huấn luyện/kiểm tra
```

```
    X = df_ml[features] # Đặc trưng (features)
    y = df_ml[target] # Mục tiêu (target)

    # Chia tập huấn luyện và tập kiểm tra (80/20)
    if len(df_ml) > 5: # Chỉ chia nếu có đủ dữ liệu
        X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.2, random_state=42)
    else: # Nếu không đủ dữ liệu, sử dụng toàn bộ làm tập huấn
luyện/kiểm tra
        print(f"Cảnh báo: Không đủ dữ liệu cho {position} để
chia tập huấn luyện/kiểm tra.")
        X_train, y_train = X, y
        X_test, y_test = X, y # Sử dụng X, y làm tập kiểm tra
để tính metric (sẽ là metric trên tập huấn luyện)

    # Định nghĩa các bước tiền xử lý cho các loại cột khác nhau
    categorical_features = [col for col in features if col in
['Team', 'Nation']]
    preprocessor = ColumnTransformer(
        transformers=[
            ('num', StandardScaler(), [col for col in
numeric_features if col != 'Age']), # Chuẩn hóa cột số (trừ
Age)
            # Age không được chuẩn hóa trong pipeline, nó đã
được xử lý NaN và trọng số
            ('age', 'passthrough', ['Age']) if 'Age' in
numeric_features else ('age', 'drop', []), # Giữ nguyên cột Age
nếu có
            ('cat', OneHotEncoder(handle_unknown='ignore',
sparse_output=False), categorical_features) # Mã hóa One-Hot
cột phân loại
        ],
        remainder='passthrough' # Giữ lại các cột khác không
được liệt kê (không có trong trường hợp này)
    )
    # CẬP NHẬT LOGIC PHÍA TRÊN: StandardScaler nên áp dụng cho
*tất cả* numeric_features (bao gồm Age) SAU khi log/weight.
    # Logic hiện tại hơi phức tạp. Giả định rằng Age được xử lý
riêng.
    # Cải tiến: Định nghĩa lại numeric_features = [col for col
in features if col not in ['Team', 'Nation']] VÀ StandardScaler
áp dụng cho toàn bộ numeric_features.
    # Code hiện tại áp dụng StandardScaler chỉ cho
numeric_features KHÔNG phải Age, và Age được 'passthrough'.
    # Cần xem xét lại logic xử lý Age và numeric_features trong
ColumnTransformer.
    # Tiếp tục phân tích theo code hiện có:
numeric_features_no_age được chuẩn hóa, Age được giữ nguyên.

    # Tạo Pipeline kết hợp tiền xử lý và mô hình hồi quy
    pipeline = Pipeline([
        ('preprocessor', preprocessor), # Bước tiền xử lý
        ('regressor', LinearRegression()) # Mô hình hồi quy
tuyến tính
    ])

    # Huấn luyện pipeline trên tập huấn luyện
    pipeline.fit(X_train, y_train)

    # Đánh giá mô hình trên tập kiểm tra (nếu có)
    if len(X_test) > 0:
        y_pred = pipeline.predict(X_test)
        rmse = np.sqrt(mean_squared_error(y_test, y_pred)) #
Tính RMSE
```

```
        r2 = r2_score(y_test, y_pred) # Tính R-squared
        print(f"Đánh giá mô hình {position} - RMSE: {rmse:.2f},
R²: {r2:.2f}")

    # Dự đoán ETV cho TẤT CẢ cầu thủ đã khớp ETV cho vị trí này
    df_filtered['Predicted_Transfer_Value'] =
pipeline.predict(df_filtered[features])
    # Cắt bớt giá trị dự đoán về phạm vi hợp lý
    df_filtered['Predicted_Transfer_Value'] =
df_filtered['Predicted_Transfer_Value'].clip(lower=100_000,
upper=200_000_000)
    # Chuyển giá trị ETV sang đơn vị triệu ($M hoặc €M)
    df_filtered['Predicted_Transfer_Value_M'] =
(df_filtered['Predicted_Transfer_Value'] / 1_000_000).round(2)
    df_filtered['Actual_Transfer_Value_M'] =
(df_filtered['ETV'] / 1_000_000).round(2)

    # Đảm bảo DataFrame kết quả có đủ các cột chuẩn, điền NaN
hoặc rỗng nếu thiếu
    for col in standard_output_columns:
        if col not in df_filtered.columns:
            df_filtered[col] = np.nan if col in
['Actual_Transfer_Value_M', 'Predicted_Transfer_Value_M'] else
''

    df_filtered['Position'] = position # Đặt lại cột vị trí
chính xác
    # Chọn và sắp xếp lại các cột theo chuẩn đầu ra
    result = df_filtered[standard_output_columns].copy()

    # Thực hiện biến đổi ngược (Inverse Transform) cho các cột
chỉ số (features) trong DataFrame kết quả
    # Mục đích là để các chỉ số hiển thị trong file CSV đầu ra
ở thang đo gốc, dễ đọc hơn
    numeric_features_no_age = [col for col in features if col
in result.columns and col not in ['Team', 'Nation', 'Age']]
    for col in numeric_features_no_age:
         # Áp dụng exp(x)-1 để đưa về thang đo gốc
        result[col] = np.expm1(result[col]).round(2)

    # Xử lý riêng cột Age (nếu có)
    if 'Age' in features and 'Age' in result.columns:
         # Áp dụng exp(x)-1 cho Age, làm tròn và chuyển kiểu
        result['Age'] = np.expm1(result['Age']).round(0)
        median_age = result['Age'].median()
        result['Age'] =
result['Age'].fillna(median_age).astype(int) # Điền NaN bằng
trung vị và chuyển sang int

    # Trả về DataFrame kết quả cho vị trí này và danh sách cầu
thủ không khớp ETV
    return result, unmatched
```

10.Luồng thực thi chính

Phần này điều phối việc cào dữ liệu ETV, sau đó lặp qua từng vị trí để xử lý và dự đoán, cuối cùng kết hợp và lưu kết quả.

Python

```
# Main execution
print("Bắt đầu cào dữ liệu giá trị chuyển nhượng...")
```

```
df_etv = scrape_transfer_values() # Cào dữ liệu ETV từ web

all_results = [] # Danh sách lưu DataFrame kết quả của từng vị
trí
all_unmatched = [] # Danh sách lưu cầu thủ không khớp từ tất cả
các vị trí

# Lặp qua từng vị trí và cấu hình đã định nghĩa
for position, config in positions_config.items():
    print(f"\nĐang xử lý {position}...")
    # Gọi hàm xử lý cho vị trí này
    result, unmatched = process_position(position, config,
df_etv)
    # Nếu có kết quả trả về (không phải None)
    if result is not None:
        all_results.append(result) # Thêm kết quả của vị trí
này vào danh sách chung
    # Nếu có cầu thủ không khớp cho vị trí này
    if unmatched:
         # Thêm danh sách không khớp vào danh sách chung, kèm
theo vị trí
        all_unmatched.extend([(position, player) for player in
unmatched])

# Sau khi xử lý tất cả các vị trí
if all_results: # Nếu có bất kỳ kết quả nào được thu thập
    try:
        # Nối tất cả các DataFrame kết quả lại với nhau
        combined_results = pd.concat(all_results,
ignore_index=True)
        # Sắp xếp kết quả cuối cùng theo giá trị dự đoán giảm
dần
        combined_results =
combined_results.sort_values(by='Predicted_Transfer_Value_M',
ascending=False)
        combined_results.to_csv(output_path, index=False) # Lưu
kết quả cuối cùng vào file CSV
        print(f"Kết quả đã được lưu vào '{output_path}'")
    except ValueError as e:
        print(f"Lỗi khi nối: {e}")
        # In thông tin cột để debug lỗi nối DataFrame
        print("Các cột trong mỗi DataFrame kết quả:")
        for i, df in enumerate(all_results):
            print(f"Cột vị trí
{list(positions_config.keys())[i]}: {df.columns.tolist()}")

if all_unmatched: # Nếu có bất kỳ cầu thủ nào không khớp
    print("\nDanh sách cầu thủ không khớp:")
    for position, player in all_unmatched:
        print(f"Vị trí: {position}, Cầu thủ: {player}")
```

- **Tóm tắt:** Chương trình 4.2 là một pipeline phức tạp hơn, tích hợp web scraping ETV, tiền xử lý dữ liệu chuyên sâu cho từng vị trí (bao gồm biến đổi logarit và trọng số đặc trưng), xây dựng và huấn luyện các mô hình Hồi quy Tuyến tính riêng biệt, và cuối cùng là dự đoán giá trị chuyển nhượng ước tính. Nó thể hiện cách áp dụng học máy để phân tích dữ liệu bóng đá và đưa ra dự đoán dựa trên các chỉ số hiệu suất. Việc xử lý riêng cho từng vị trí và việc sử dụng Pipeline của scikit-learn là những điểm đáng chú ý trong thiết kế.