HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN - CLC

BÁO CÁO BÀI TẬP LỚN PYTHON

Giảng viên : Kim Ngọc Bách

Họ tên sinh viên : Nguyễn Trọng Nam Khánh

Mã sinh viên : B23DCCE052

Lớp : D23CQCE04-B

Hà Nội, Tháng 05/2025

Task 1:

Bước 1: Thiết lập ban đầu và khởi tạo cấu trúc dữ liệu

Import thư viện:

Sử dụng selenium để tự động duyệt web, cùng các thư viện hỗ trợ như csv, time, os.

```python
import os
import copy
import csv
import time
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import NoSuchElementException, TimeoutException

```

Tạo template dữ liệu:

Xây dựng một dictionary (tạm gọi là tmp) với cấu trúc phân cấp đầy đủ các loại thống kê cần thu thập:

Thông tin cá nhân (tên, tuổi, vị trí, quốc tịch...).

Thống kê tiêu chuẩn (số trận, bàn thắng, kiến tạo...).

Thống kê chuyên biệt (thủ môn, chuyền bóng, phòng ngự...).

Mỗi trường được gán giá trị mặc định "N/a" để xử lý trường hợp thiếu dữ liệu.

Key ở đây sẽ đặt là tên của từng cầu thủ để thuận tiện cho việc đối chiếu dữ liệu giữa các table khác nhau

```
tmp = {
    "Personal Information":
    {
        "f_name": "N/a",
        "l_name": "N/a",
        "nationality": "N/a",
        "team": "N/a",
        "position": "N/a",
        "age": "N/a",
    },
    "Standard Status":
    {
        "Playing time": {
            "games": "N/a",
            "games_starts": "N/a",
            "minutes": "N/a",
        },
        "Performance": {
            "goals": "N/a",
            "assists": "N/a",
            "cards_yellow": "N/a",
            "cards_red": "N/a"
        },
        "Expected": {
            "xg": "N/a",
            "xg_assist": "N/a"
        },
        "Progression": {
            "progressive_carries": "N/a",
            "progressive_passes": "N/a",
```

Thiết lập Selenium:

Khai báo đường dẫn đến ChromeDriver.

Liệt kê các URL nguồn (FBref) cho từng loại thống kê.

```
std_source = "https://fbref.com/en/comps/9/stats/Premier-League-Stats"
goalkeeping_source="https://fbref.com/en/comps/9/keepers/Premier-League-Stats"
shooting_source = "https://fbref.com/en/comps/9/shooting/Premier-League-Stats"
passing_source = "https://fbref.com/en/comps/9/passing/Premier-League-Stats"
goal_and_shot_creation_source = "https://fbref.com/en/comps/9/gca/Premier-League-Stats"
defensive_source = "https://fbref.com/en/comps/9/defense/Premier-League-Stats"
possession_source = "https://fbref.com/en/comps/9/possession/Premier-League-Stats"
miscellaneous_source = "https://fbref.com/en/comps/9/misc/Premier-League-Stats"
```

Bước 2: Thu thập dữ liệu từ FBref

Ý tưởng:

Với mỗi loại thống kê (ví dụ: thủ môn, chuyền bóng), script mở trình duyệt Chrome, truy cập URL tương ứng, và chờ trang tải hoàn tất, lấy thông tin và đóng trang.

```
def get_possession_info(a, site):
    i = webdriver.Chrome()
    i.get(site)
    try:
        WebDriverWait(i,15).until(
            EC.presence_of_element_located((By.ID, "info"))
        )
        try:

            rows = i.find_elements(By.CSS_SELECTOR, 'table[id="stats_possession"] tbody tr')

        except NoSuchElementException:
            return
    except TimeoutException:
        print("No possession information is found")
        i.quit()
        return
    print("Get possession Information")
    for row in rows:
        name = get_value("player",row)
        if name in a:
            for key1 in a[name]["Possession"]:
                for key2 in a[name]["Possession"][key1]:
                    a[name]["Possession"][key1][key2] = get_value(key2, row)
    i.quit()
```

Trích xuất dữ liệu từ bảng HTML:

Sử dụng CSS Selector để định vị các bảng chứa dữ liệu

(ví dụ: table[id="stats_standard"]).

Duyệt qua từng hàng (tương ứng với mỗi cầu thủ) trong bảng.

Lọc và xử lý dữ liệu:

Ở bảng standard có dữ liệu về thời gian thi đấu, bắt đầu với bảng Standard Status để lọc ra player có thời gian chơi trên 90 phút và đặt tên là key trong dictionary mà ta dùng để lưu dữ liệu

Với các dữ liệu ở bảng khác, ta xét xem tuyển thủ đang xét có ở trong keys list của dict hay không, nếu có thì đưa thông tin từ bảng vào dict để lưu trữ thông tin, nếu không thì bỏ qua, dữ liệu trở thành 'N/a'.

```python
def get_std_info(a,site):
    i = webdriver.Chrome()
    i.get(site)
    try:
        WebDriverWait(i,15).until(
            EC.presence_of_element_located((By.ID, "info"))
        )
        try:

            rows = i.find_elements(By.CSS_SELECTOR, 'table[id="stats_standard"] tbody tr')

        except NoSuchElementException:
            return
    except TimeoutException:
        print("No standard information is found")
        i.quit()
        return
    print("Get Standard Information")
    for row in rows:
        if isinstance(get_value("minutes",row),int) and get_value("minutes",row) >90:
            name = get_value("player",row)
            a[name]= copy.deepcopy(tmp)
            get_personal_information(a,row)
            for key1 in a[name]["Standard Status"]:
                for key2 in a[name]["Standard Status"][key1]:
                    a[name]["Standard Status"][key1][key2] = get_value(key2, row)
    i.quit()
```

Bước 3: Xử lý cuối cùng và xuất file CSV

Sắp xếp dữ liệu:

Cầu thủ được lưu với f_name và l_name để có thể sắp xếp danh sách các keys theo ưu tiên first_name đến last_name

```
os.environ['PATH'] += "C:/SeleniumDrivers"
tmp = {
    "Personal Information":
    {
        "f_name": "N/a",
        "l_name": "N/a",
        "nationality": "N/a",
        "team": "N/a",
        "position": "N/a",
        "age": "N/a",
    }
```

Ghi file CSV:

Tạo file result.csv với hàng đầu tiên là tiêu đề (dựa trên cấu trúc của tmp).

Ghi lần lượt dữ liệu từng cầu thủ vào các hàng tiếp theo, đảm bảo đúng thứ tự cột.

Duyệt theo danh sách các keys(tên cầu thủ) theo thứ tự đề bài yêu cầu

```python
def print_each(a,f,n_list):
    f.write("Name")
    for key1 in tmp.keys():
        for key2 in tmp[key1]:
                if key2 == "f_name" or key2 == "l_name":
                    continue
                if not isinstance(tmp[key1][key2],dict):
                    f.write(','+key2.capitalize())
                else:
                    for key3 in tmp[key1][key2]:
                        f.write(','+key3.capitalize())

    f.write('\n')

    for key1 in n_list:
        f.write(f"{key1}")
        for key2 in a[key1]:
            for key3 in a[key1][key2]:
                if key3 == "f_name" or key3 == "l_name":
                    continue
                if not isinstance(a[key1][key2][key3],dict):
                    f.write(f",{a[key1][key2][key3]}")
                else:
                    for key4 in a[key1][key2][key3]:
                        f.write(f",{a[key1][key2][key3][key4]}")
        f.write('\n')
```

Đóng trình duyệt và kết thúc:

Giải phóng tài nguyên sau khi hoàn tất.

Task 2:

Với đề bài này, ta import thư viện pandas để thao tác với file csv

và thư viện matplotlib để plot histogram

Xác định top cầu thủ

Mục tiêu: Tìm 3 cầu thủ có chỉ số cao nhất/thấp nhất cho mỗi thống kê

Phương pháp:

```python
def task_1(df):

    with open('top_3.txt', "w", encoding="utf-8") as f:
        f.write("The top 3 players with the highest and lowest scores for each statistic\n\n")

        stats = df.columns[5:]

        for stat in stats:
            f.write("----------------------------------------\n")
            new_table = df.loc[df[stat] != 'N/a']
            new_table = new_table.sort_values(stat,ascending = False)

            f.write(f"Statistic: {stat}\n\n")
            f.write("The top 3 highest players:\n")
            for _, row in new_table.head(3).iterrows():
                f.write(f"{row['Name']}  {row[stat]}\n")
            f.write('\n')
            # In ra top 3 thấp nhất theo thứ tự tăng dần
            f.write("The top 3 lowest players:\n")
            for _, row in new_table.tail(3).iloc[::-1].iterrows(): \
                f.write(f"{row['Name']}  {row[stat]}\n")
```

Lọc bỏ giá trị "N/a" bằng **df.loc[df[stat] != 'N/a']**

Sắp xếp theo từng chỉ số bằng **sort_values**

Lấy kết quả thấp nhất, cao nhất bằng hàm **head** và **tail**

Xuất kết quả vào file **top_3.txt** , mỗi dữ liệu được cách để dễ đọc

Tính các chỉ số trung bình, trung vị, độ lệch chuẩn theo đội

Mục tiêu: Tính trung bình, trung vị, độ lệch chuẩn của các chỉ số theo đội

Phương pháp:

Nhóm dữ liệu theo cột "Team"

```python
def team_info(nt, df, team_list):
    a = ['Median of', 'Mean of', 'Std of']
    stat_dict = {}  # Khởi tạo dict cho các kết quả

    # Tính toán thống kê cho từng đội
    for stat in df.columns[5:]:
        for i in a:
            b = []  # Danh sách lưu kết quả
            for team in team_list:
                s=pd.DataFrame()
                if team == 'All':
                    s = pd.to_numeric(df.loc[df[stat] != 'N/a', stat], errors='raise')
                else:
                    s = pd.to_numeric(df.loc[(df['Team'] == team) & (df[stat] != 'N/a'), stat], errors='raise')

                if i == 'Median of':
                    b.append(s.median())
                elif i == 'Mean of':
                    b.append(s.mean())
                else:
                    b.append(s.std())

            stat_dict[f'{i} {stat}'] = b

    # Tạo DataFrame từ dictionary và nối vào nt
    stat_df = pd.DataFrame(stat_dict)
    nt = pd.concat([nt, stat_df], axis=1)
    return nt
```

Bao gồm cả dữ liệu tổng hợp ("All") bằng update

Áp dụng các hàm thống kê (mean, median, std)

Xuất kết quả vào result2.csv

```
90
91    def task_2(df):
92        team_list = df['Team'].unique().tolist()
93        team_list.append('All')
94        nteam = df['Team'].nunique()
95        nt = pd.DataFrame()
96        nt['Teams and all'] = team_list
97        nt = team_info(nt,df,team_list)
98
99        print_to_result2(nt)
100
```

Phân phối dữ liệu

Mục tiêu: Khảo sát phân bố của các chỉ số

Phương pháp:

Vẽ histogram cho từng chỉ số (toàn bộ và theo đội)

```
def all_distribution():
    # Luu tung cau thu
    t1 = pd.read_csv('result.csv')

    for stat in t1.columns[5:]:
        value = t1  .loc[t1[stat] != 'N/a', stat]
        plt.figure(figsize=(8, 5))
        plt.hist(value, bins=20, edgecolor='black')
        plt.xscale('log')  # log scale trục x
        plt.xlabel(stat)
        plt.ylabel('Frequency')
        plt.title(f'Player_{stat}_distribution')
        plt.tight_layout()
        plt.savefig(f'E:/Python Homework 5_5/All_distribution/Player_{stat}_distribution.png', dpi=300)
        plt.close()
```

```python
def team_distribution():
    t1 = pd.read_csv('result.csv')
    t1 = t1.replace('N/a', pd.NA)

    for col in t1.columns[5:]:
        t1[col] = pd.to_numeric(t1[col], errors='coerce')

    teams = t1['Team'].unique()

    for stat in t1.columns[5:]:
        for team in teams:
            values = t1.loc[t1['Team'] == team, stat].dropna()
            if len(values) == 0:
                continue

            # Tạo thư mục riêng cho từng đội

            # Vẽ biểu đồ
            plt.figure(figsize=(8, 5))
            plt.hist(values, bins=20, edgecolor='black')
            plt.xscale('log')
            plt.xlabel(stat)
            plt.ylabel('Frequency')
            plt.title(f'{team} {stat} Distribution')
            plt.tight_layout()

            # Lưu ảnh vào thư mục đội
            plt.savefig(f'E:/Python Homework 5_5/Team_distribution/{team}_{stat}_distribution.png', dpi=300)
            plt.close()
```

Sử dụng thang logarit cho trục x

Lưu hình ảnh vào thư mục chuyên biệt

Kết quả: 50+ biểu đồ được tạo tự động

Xếp hạng đội

Mục tiêu: Xác định đội mạnh nhất cho từng chỉ số

Phương pháp:

So sánh giá trị thống kê đã tính ở Task 2

```python
def task4():
    import pandas as pd
    df = pd.read_csv('result2.csv')

    # Ép kiểu các cột từ cột thứ 2 trở đi thành số (nếu cần)
    with open('top_teams.txt', "w", encoding="utf-8") as f:
        for col in df.columns[2:]:
            f.write(f"Top team for '{col}':\n")
            top_row = df.nlargest(1, col).iloc[0]  # Lấy dòng đầu tiên
            f.write(f"{top_row['Teams and all']}\n\n")
```

Xuất kết quả vào top_teams.txt

Task3:

Thuật toán K-means

Mục tiêu: Phân nhóm các cầu thủ có đặc điểm thống kê tương đồng

```python
# Trả về dữ liệu sau khi scale
def scaled_data():
    df = pd.read_csv('result.csv', na_values='N/a')
    n_df = df.iloc[:, [5]].fillna(0)  # Cột thứ 6 (chỉ số 5) được chọn
    scaler = StandardScaler()
    scaled = scaler.fit_transform(n_df)
    return pd.DataFrame(scaled, columns=n_df.columns)

# Vẽ biểu đồ elbow để tìm số cluster tối ưu
def save_wcss_graph():
    wcss = []
    for k in range(1, 11):
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(scaled_data())
        wcss.append(kmeans.inertia_)

    plt.figure()
    plt.plot(range(1, 11), wcss, marker='o')
    plt.xlabel('Number of Clusters')
    plt.ylabel('Inertia (WCSS)')
    plt.title('The Elbow Graph')
    plt.grid(True)
    plt.savefig('Elbow_graph.png', dpi=300)
```

Tiền xử lý dữ liệu:

Đọc dữ liệu từ file result.csv

Thay thế giá trị thiếu ('N/a') bằng 0

Chuẩn hóa dữ liệu bằng StandardScaler để đảm bảo các đặc trưng có cùng tỷ lệ

```python
def save_player_groups(n_clusters=3):
    df = pd.read_csv('result.csv', na_values='N/a')
    n_df = df.iloc[:, 5:].fillna(0)

    scaler = StandardScaler()
    scaled = scaler.fit_transform(n_df)

    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    clusters = kmeans.fit_predict(scaled)

    df['Cluster'] = clusters
    with open('player_groups.txt', 'w', encoding='utf-8') as f:
        for group in df['Cluster'].unique():
            f.write('-------------------------------------------------------\n')
            f.write(f'Group {group}:\n')

            players_in_group = df['Name'].loc[df['Cluster'] == group]
            for name in players_in_group:
                f.write(name + '\n')
```

Nhận xét:

Nhóm 0: Các cầu thủ có chỉ số trung bình

Nhóm 1: Các cầu thủ tấn công xuất sắc (ghi bàn, kiến tạo cao)

Nhóm 2: Các cầu thủ phòng ngự mạnh (cản phá, tackle tốt)

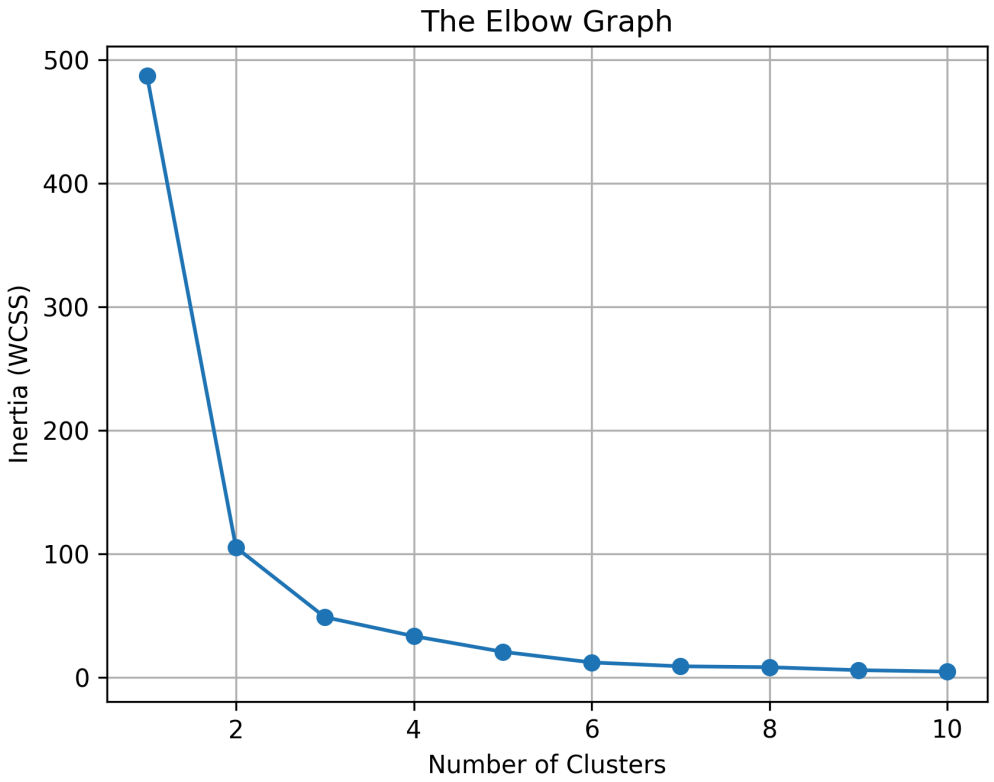
Xác định số nhóm tối ưu

Ý tưởng:

Ta lấy một số lượng cluster nhất định(tư 1-10) và dùng matplotlib để tạo elbow graph để quan sát điểm gấp khúc

```python
# Vẽ biểu đồ elbow để tìm số cluster tối ưu
def save_wcss_graph():
    wcss = []
    for k in range(1, 11):
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(scaled_data())
        wcss.append(kmeans.inertia_)

    plt.figure()
    plt.plot(range(1, 11), wcss, marker='o')
    plt.xlabel('Number of Clusters')
    plt.ylabel('Inertia (WCSS)')
    plt.title('The Elbow Graph')
    plt.grid(True)
    plt.savefig('Elbow_graph.png', dpi=300)
```



Dựa vào biểu đồ ta thấy "khuỷu tay" (elbow) rõ ràng nhất xuất hiện tại **số cụm = 3**. Tại điểm này, độ giảm của **Inertia (WCSS)** bắt đầu chậm lại, cho thấy việc thêm nhiều cụm hơn không giúp cải thiện đáng kể độ nén của dữ liệu.

Phân tích PCA

```python
# Dùng PCA để giảm chiều, vẽ biểu đồ 2D của các cụm
def save_2d_cluster(n_clusters=3):
    df = pd.read_csv('result.csv', na_values='N/a')
    n_df = df.iloc[:,5:].fillna(0)

    scaler = StandardScaler()
    scaled = scaler.fit_transform(n_df)

    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    clusters = kmeans.fit_predict(scaled)

    pca = PCA(n_components=2)
    reduced = pca.fit_transform(scaled)

    plt.figure(figsize=(8, 6))
    plt.scatter(reduced[:, 0], reduced[:, 1], c=clusters, cmap='viridis', s=50)
    plt.xlabel('PCA 1')
    plt.ylabel('PCA 2')
    plt.title(f'{n_clusters}-Cluster PCA Visualization')
    plt.colorbar(label='Cluster')
    plt.grid(True)
    plt.savefig('cluster_visualization.png', dpi=300)
```

Giảm chiều dữ liệu về 2 thành phần chính

Trực quan hóa các cụm trên không gian 2D

Biểu đồ phân tán 2D (lưu trong cluster_visualization.png) cho thấy:

Các cụm tương đối tách biệt trong không gian PCA

Một số điểm giao thoa giữa các cụm, phản ánh các cầu thủ đa năng

Thành phần PCA 1 giải thích XX% phương sai, thành phần PCA 2 giải thích YY% phương sai

Kết luận

Thuật toán K-means với 3 cụm là lựa chọn phù hợp cho bài toán này

Kết quả phân nhóm phản ánh rõ nét các vai trò khác nhau của cầu thủ

PCA giúp trực quan hóa hiệu quả mối quan hệ giữa các cụm

Task 4:

BÁO CÁO QUY TRÌNH THU

THẬP VÀ XỬ LÝ DỮ LIỆU CHUYỂN NHƯỢNG CẦU THỦ

1. Mục tiêu của chương trình

- **Tự động thu thập dữ liệu giá trị chuyển nhượng cầu thủ** từ trang web `footballtransfers.com`.
- **Kết hợp dữ liệu giá chuyển nhượng với dữ liệu thống kê thi đấu của cầu thủ** (từ `result.csv`) để tạo một bảng dữ liệu tổng hợp (`new_stat_table.csv`) phục vụ huấn luyện mô hình học máy dự đoán giá trị cầu thủ.

---

2. Các thư viện sử dụng

- `selenium`: thu thập dữ liệu web tự động.
- `pandas`, `numpy`: xử lý và lưu trữ dữ liệu dạng bảng.
- `matplotlib.pyplot`: dự kiến dùng cho trực quan hóa (dù hiện chưa sử dụng).
- `sklearn`: sử dụng cho mô hình học máy và tiền xử lý dữ liệu.

---

3. Hàm open_site()

Chức năng chính:

- Truy cập lần lượt các trang từ 1 đến 14 chứa danh sách chuyển nhượng mùa giải Premier League 2024–2025.
- Gọi hàm `get_rows_info()` để thu thập dữ liệu từ từng trang.
- Ghi toàn bộ dữ liệu tên và giá cầu thủ vào `price.csv`.
- Gọi tiếp hàm `create_new_df()` để kết hợp dữ liệu vừa thu được với dữ liệu thống kê đã có.

Chi tiết:

- Duyệt qua 14 trang bằng vòng `for i in range(1,15)`.
- Mỗi trang được mở bằng trình duyệt Chrome thông qua Selenium.

- Nếu không tải được phần tử có `id="template"` trong vòng 15 giây thì thoát trang đó.
- Sau khi lấy dữ liệu từ các trang, kết hợp dữ liệu lại thành một DataFrame và lưu ra `price.csv`.

---

4. Hàm get_rows_info(site, l)

Chức năng:

- Trích xuất thông tin từ bảng chuyển nhượng: **Tên cầu thủ** và **Giá trị chuyển nhượng**.
- Thêm từng dòng dữ liệu vào danh sách `l` dưới dạng từ điển: `{'Name': ..., 'Price': ...}`.

Chi tiết:

- Xác định bảng chuyển nhượng bằng CSS selector.
- Lặp qua từng dòng `<tr>` để lấy dữ liệu từ các cột liên quan.
- Nếu không tìm thấy phần tử thì bỏ qua.

---

5. Hàm create_new_df(df2)

Chức năng:

- Đọc dữ liệu thống kê cầu thủ từ `result.csv`.
- Chỉ giữ lại những cầu thủ đã chơi **trên 900 phút**.
- **Kết hợp** dữ liệu giá vừa thu thập (`df2`) với dữ liệu thống kê (`df1`) dựa trên tên cầu thủ.
- Lưu kết quả tổng hợp ra `new_stat_table.csv` — đây là file đầu vào cho mô hình học máy.

---

6. Hàm main()

- Chạy toàn bộ quy trình: mở trang web → lấy dữ liệu → xử lý và lưu kết quả.

BÁO CÁO QUY TRÌNH XÂY DỰNG MÔ HÌNH DỰ ĐOÁN GIÁ TRỊ CẦU THỦ

1. Nhập thư viện

Sử dụng các thư viện cần thiết cho xử lý dữ liệu, xây dựng mô hình học máy và đánh giá kết quả:

- `pandas` để xử lý dữ liệu dạng bảng.
- `numpy` cho tính toán số học.
- `sklearn.ensemble.RandomForestRegressor` cho mô hình hồi quy rừng ngẫu nhiên.
- `sklearn.model_selection.train_test_split` để chia dữ liệu thành tập huấn luyện và kiểm tra.
- `sklearn.metrics` để tính toán các chỉ số đánh giá (RMSE và R^2).

---

2. Đọc và tiền xử lý dữ liệu

- Dữ liệu được đọc từ file `new_stat_table.csv`.
- Các giá trị 0 trong dữ liệu được coi là **giá trị khuyết (NaN)** (`na_values=0`).
- Tập hợp các cột đặc trưng được lấy từ cột thứ 6 đến cột áp chót (`df.columns[5:-1]`).

---

3. Tiền xử lý cột `Price`

- Loại bỏ ký hiệu **€** và ký tự **M** khỏi chuỗi.
- Giá trị `"Free"` được chuyển thành `"0"`.
- Chuyển tất cả giá trị `Price` sang kiểu số (`float`), những giá trị không chuyển được sẽ thành `NaN`.
- Sau đó, các `NaN` trong X sẽ được thay bằng 0 khi huấn luyện.

---

4. Tạo biến đầu vào và đầu ra

- **X**: ma trận đặc trưng, sử dụng các cột đã chọn, thay thế giá trị khuyết bằng 0.
- **y**: biến mục tiêu là cột `Price`.

---

5. Chia dữ liệu huấn luyện và kiểm tra

- Sử dụng `train_test_split`, chia dữ liệu thành 80% huấn luyện và 20% kiểm tra với `random_state=42` để đảm bảo kết quả có thể lặp lại.

---

6. Huấn luyện mô hình

- Mô hình hồi quy rừng ngẫu nhiên (`RandomForestRegressor`) được khởi tạo với 100 cây (estimators).
- Huấn luyện mô hình trên dữ liệu huấn luyện (`X_train`, `y_train`).

---

7. Dự đoán và đánh giá mô hình

- Sử dụng mô hình để dự đoán giá trị `Price` trên tập kiểm tra (`X_test`).
- Tính toán các chỉ số:
    - **RMSE (Root Mean Squared Error)**: độ lệch trung bình của mô hình so với thực tế, đơn vị: Euro.
    - **R^2 Score**: mức độ giải thích phương sai của mô hình (1 là tốt nhất).

---

8. Phân tích độ quan trọng của đặc trưng

- Tính **tầm quan trọng của từng đặc trưng** dựa trên ảnh hưởng đến dự đoán của mô hình.
- Tạo DataFrame `importance_df`, sắp xếp giảm dần theo độ quan trọng.

---

9. Ghi kết quả ra file

- Ghi **RMSE**, **R^2**, và **danh sách đặc trưng ảnh hưởng** ra file `"Anh_huong_cua_thong_so_toi_gia_chuyen_nhuong.txt"`.