HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

BỘ MÔN NGÔN NGỮ LẬP TRÌNH PYTHON

BÁO CÁO BÀI TẬP LỚN

Giảng viên hướng dẫn	**: Kim Ngọc Bách**
Họ và tên sinh viên	**: Lê Minh Đức**
Mã sinh viên	**: B23DCVT095**
Lớp	**: D23CQCE06-B**
Hệ đào tạo	**: Đại học chính quy**

Hà Nội – 2025

MỤC LỤC

LỜI MỞ ĐẦU

Trong thời đại dữ liệu hiện nay, việc thu thập và phân tích dữ liệu không chỉ giới hạn trong các lĩnh vực như tài chính, y tế hay thương mại điện tử, mà còn đóng vai trò quan trọng trong thể thao – một lĩnh vực đòi hỏi sự chính xác, chiến lược và tối ưu hóa hiệu suất. Bóng đá, với tính chất cạnh tranh cao và sự quan tâm rộng rãi từ cộng đồng, đã trở thành một trong những lĩnh vực tiên phong trong việc áp dụng khoa học dữ liệu để hỗ trợ ra quyết định, phân tích hiệu suất cầu thủ và xây dựng chiến lược thi đấu. Nhằm giúp sinh viên tiếp cận với ứng dụng thực tế của lập trình Python trong lĩnh vực này, bài tập số 1 trong học phần "Lập trình Python" yêu cầu thực hiện một loạt các tác vụ liên quan đến thu thập, xử lý, phân tích và trực quan hóa dữ liệu cầu thủ tại Giải Ngoại hạng Anh mùa giải 2024–2025.

Bài tập đặt ra yêu cầu xây dựng một chương trình Python có khả năng tự động thu thập dữ liệu thống kê của các cầu thủ từ trang web chuyên biệt fbref.com, xử lý và lưu trữ dữ liệu có cấu trúc, đồng thời thực hiện các phân tích thống kê cơ bản như trung bình, trung vị, độ lệch chuẩn theo từng đội bóng và trên toàn giải đấu. Ngoài ra, sinh viên còn được yêu cầu triển khai các công cụ phân tích dữ liệu nâng cao như biểu đồ phân phối, thuật toán phân cụm K-means, kỹ thuật giảm chiều PCA, và đề xuất phương pháp ước lượng giá trị chuyển nhượng cầu thủ dựa trên các đặc trưng thống kê thu thập được.

Bài tập không chỉ giúp sinh viên rèn luyện kỹ năng lập trình, mà còn thúc đẩy tư duy phân tích dữ liệu và hiểu biết sâu hơn về cách dữ liệu thể thao được sử dụng để hỗ trợ ra quyết định trong thực tiễn. Qua đó, sinh viên có thể hình dung rõ ràng hơn vai trò của Python như một công cụ mạnh mẽ trong việc khai phá thông tin từ dữ liệu lớn và phát triển các mô hình có giá trị ứng dụng cao. Việc thực hiện bài tập còn giúp nâng cao khả năng xử lý dữ liệu thực tế, thao tác với dữ liệu thiếu, không đồng nhất và phát triển khả năng tư duy hệ thống trong việc thiết kế và triển khai quy trình phân tích hoàn chỉnh.

BÀI TẬP 1

1. Cấu hình đầu vào

a. Đường dẫn và ID các bảng cần lấy

```
TABLE_LINKS = {
    'Standard Stats': ('https://fbref.com/en/comps/9/stats/Premier-League-Stats', 'stats_standard'),
    'Shooting': ('https://fbref.com/en/comps/9/shooting/Premier-League-Stats', 'stats_shooting'),
    'Passing': ('https://fbref.com/en/comps/9/passing/Premier-League-Stats', 'stats_passing'),
    'Goal and Shot Creation': ('https://fbref.com/en/comps/9/gca/Premier-League-Stats', 'stats_gca'),
    'Defense': ('https://fbref.com/en/comps/9/defense/Premier-League-Stats', 'stats_defense'),
    'Possession': ('https://fbref.com/en/comps/9/possession/Premier-League-Stats', 'stats_possession'),
    'Miscellaneous': ('https://fbref.com/en/comps/9/misc/Premier-League-Stats', 'stats_misc'),
    'Goalkeeping': ('https://fbref.com/en/comps/9/keepers/Premier-League-Stats', 'stats_keeper')
}
```

-Mỗi bảng dữ liệu được xác định bởi một tên, URL và ID của bảng HTML trên trang.

-Các bảng như: Thống kê cơ bản, Sút, Chuyền bóng, Phòng ngự, Kiểm soát bóng, Thủ môn, v.v.

b. Cột dữ liệu cần thiết

```
REQUIRED_COLUMNS = {
    'Standard Stats': ['Player', 'Nation', 'Squad', 'Pos', 'Age', 'Playing Time Min', 'Playing Time MP', 'Playing Time Starts',
                       'Performance Gls', 'Performance Ast', 'Performance CrdY', 'Performance CrdR',
                       'Expected xG', 'Expected xAG', 'Progression PrgC', 'Progression PrgP', 'Progression PrgR',
                       'Per 90 Minutes Gls', 'Per 90 Minutes Ast', 'Per 90 Minutes xG', 'Per 90 Minutes xAG'],
    'Shooting': ['Player', 'Standard SoT%', 'Standard SoT/90', 'Standard G/Sh', 'Standard Dist'],
    'Passing': ['Player', 'Total Cmp', 'Total Cmp%', 'Total TotDist', 'Short Cmp', 'Medium Cmp', 'Long Cmp',
                'KP', 'Passing 1/3', 'PPA', 'CrsPA', 'PrgP'],
    'Goal and Shot Creation': ['Player', 'SCA', 'SCA SCA90', 'GCA', 'GCA GCA90'],
    'Defense': ['Player', 'Tackles Tkl', 'Tackles TklW', 'Challenges Att', 'Challenges Lost',
                'Blocks', 'Blocks Sh', 'Blocks Pass', 'Int'],
    'Possession': ['Player', 'Touches', 'Touches Def Pen', 'Touches Def 3rd', 'Touches Mid 3rd', 'Touches Att 3rd', 'Touches Att Pen',
                'Take-Ons Att', 'Take-Ons Succ%', 'Take-Ons Tkld%',
                'Carries', 'Carries PrgDist', 'Carries PrgC', 'Carries 1/3', 'Carries CPA', 'Carries Mis', 'Carries Dis',
                'Receiving Rec', 'Receiving PrgR'],
    'Miscellaneous': ['Player', 'Performance Fls', 'Performance Fld', 'Performance Off', 'Performance Crs', 'Performance Recov',
                      'Aerial Duels Won', 'Aerial Duels Lost', 'Aerial Duels Won%'],
    'Goalkeeping': ['Player', 'Performance GA90', 'Performance Save%', 'Performance CS%', 'Penalty Kicks Save%']
}
```

-Xác định rõ cột nào cần được lấy từ mỗi bảng dữ liệu.

-Giúp lọc bỏ các cột không cần thiết, làm nhẹ dữ liệu và đơn giản hóa xử lý.

c. Cột xuất hiện trong kết quả cuối cùng

```
FINAL_COLUMNS = [
    'Player', 'Nation', 'Squad', 'Pos', 'Age', 'Playing Time Min', 'Playing Time MP', 'Playing Time Starts',
    'Performance Gls', 'Performance Ast', 'Performance CrdY', 'Performance CrdR',
    'Expected xG', 'Expected xAG', 'Progression PrgC', 'Progression PrgP', 'Progression PrgR',
    'Per 90 Minutes Gls', 'Per 90 Minutes Ast', 'Per 90 Minutes xG', 'Per 90 Minutes xAG',
    'Performance GA90', 'Performance Save%', 'Performance CS%', 'Penalty Kicks Save%',
    'Standard SoT%', 'Standard SoT/90', 'Standard G/Sh', 'Standard Dist',
    'Total Cmp', 'Total Cmp%', 'Total TotDist', 'Short Cmp', 'Medium Cmp', 'Long Cmp',
    'KP', 'Passing 1/3', 'PPA', 'CrsPA', 'PrgP',
    'SCA', 'SCA SCA90', 'GCA', 'GCA GCA90',
    'Tackles Tkl', 'Tackles TklW', 'Challenges Att', 'Challenges Lost',
    'Blocks', 'Blocks Sh', 'Blocks Pass', 'Int',
    'Touches', 'Touches Def Pen', 'Touches Def 3rd', 'Touches Mid 3rd', 'Touches Att 3rd', 'Touches Att Pen',
    'Take-Ons Att', 'Take-Ons Succ%', 'Take-Ons Tkld%',
    'Carries', 'Carries PrgDist', 'Carries PrgC', 'Carries 1/3', 'Carries CPA', 'Carries Mis', 'Carries Dis',
    'Receiving Rec', 'Receiving PrgR',
    'Performance Fls', 'Performance Fld', 'Performance Off', 'Performance Crs', 'Performance Recov',
    'Aerial Duels Won', 'Aerial Duels Lost', 'Aerial Duels Won%'
]
```

-Tập hợp tất cả các thống kê quan trọng cần có trong bảng kết quả sau cùng.

2. Thu thập dữ liệu bảng bằng Selenium + BeautifulSoup

```
def fetch_table(driver, url, table_id, name):
    """Fetch and parse table from the given URL."""
    print(f'\nFetching data: {name}')
    driver.get(url)
    try:
        WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.ID, table_id)))
    except Exception as e:
        print(f'Loading table error {name}: {e}')
        return None

    soup = BeautifulSoup(driver.page_source, 'html.parser')
    table = soup.find('table', id=table_id)
    if not table:
        print(f'Unable to find table {name}')
        return None

    df = pd.read_html(StringIO(str(table)))[0]
    return df
```

-Mở trang web bằng trình điều khiển Chrome.

-Đợi bảng HTML có ID cụ thể tải xong (dùng WebDriverWait).

-Dùng BeautifulSoup để lấy HTML bảng rồi chuyển thành pandas.DataFrame.

3. Xử lý tiêu đề cột phức tạp

```
def process_columns(df, name):
    """Process MultiIndex and clean column names."""
    if isinstance(df.columns, pd.MultiIndex):
        new_cols = []
        for col in df.columns:
            group = col[0].strip() if col[0].strip() and 'Unnamed' not in col[0] else ''
            subgroup = col[1].strip() if col[1].strip() else col[0].strip()
            col_name = f"{group} {subgroup}" if group and group != subgroup else subgroup
            new_cols.append(col_name.strip())
        df.columns = new_cols
    else:
        df.columns = [col.strip() for col in df.columns]

    print(f"Original columns {name}: {list(df.columns)}")
    return df
```

-Một số bảng có tiêu đề 2 dòng (MultiIndex), ví dụ: ('Performance', 'Goals').

-Hàm này sẽ:

+Gộp hai cấp tiêu đề thành một chuỗi duy nhất (ví dụ: Performance Gls).

+Loại bỏ ký tự thừa, làm tiêu đề dễ đọc và xử lý hơn.

4. Làm sạch dữ liệu và chọn cột quan trọng

```
def clean_data(df, name):
    """Clean data and select required columns."""
    player_col = next((col for col in df.columns if 'player' in col.lower() and 'rk' not in col.lower()), None)
    if not player_col:
        print(f"ERROR: Can't find column 'Player' in {name}")
        return None

    df = df.loc[df[player_col].notna() & (df[player_col] != player_col)].drop_duplicates(subset=player_col)
    df = df.rename(columns={player_col: 'Player'})
    if name == 'Passing':
        df = df.rename(columns={'1/3': 'Passing 1/3'})

    required_cols = REQUIRED_COLUMNS.get(name, ['Player'])
    selected_cols = [col for col in required_cols if col in df.columns]
    if len(selected_cols) <= 1:
        print(f"Can't find the required columns for {name}, pass!")
        return None

    df = df[selected_cols]
    print(f"Selected columns {name}: {list(df.columns)}")
    return df
```

-Loại bỏ các dòng lặp tiêu đề (thường nằm trong dữ liệu).

-Đảm bảo chỉ giữ các cột cần thiết (theo REQUIRED_COLUMNS).

-Chuẩn hóa tên cột "Player" để thống nhất cho việc gộp dữ liệu.

5. Gộp các bảng thành một bảng tổng

```
def merge_data(merged_df, df, name):
    """Merge data into the main DataFrame."""
    if name == 'Goalkeeping':
        return df
    if merged_df.empty:
        return df
    df = df.drop(columns=[col for col in df.columns if col in merged_df.columns and col != 'Player'])
    return pd.merge(merged_df, df, on='Player', how='outer')
```

-Ghép dữ liệu của bảng mới vào bảng tổng merged_df theo cột "Player".

-Tránh trùng lặp cột đã tồn tại.

-Trường hợp đặc biệt: Nếu là bảng "Goalkeeping", không gộp ngay mà xử lý riêng.

6. Lọc cầu thủ có thời gian thi đấu trên 90 phút

```
def filter_players(merged_df):
    """Filter players with more than 90 minutes played."""
    min_col = next((col for col in merged_df.columns if 'playing time min' in col.lower()), None)
    if min_col:
        merged_df = merged_df[merged_df[min_col].notna()]
        try:
            merged_df[min_col] = merged_df[min_col].str.replace(',', '', regex=False).astype(float)
            merged_df = merged_df[merged_df[min_col] > 90]
            print(f"Found {len(merged_df)} players who played over 90 minutes")
        except Exception as e:
            print(f"Filtering Error 'Playing Time Min': {e}")
    else:
        print("ERROR: Can't find column 'Playing Time Min'")
    return merged_df
```

-Chỉ giữ lại các cầu thủ đã thi đấu tổng cộng >90 phút trong mùa giải.

-Chuyển giá trị phút (ví dụ: "1,234") thành số (int hoặc float) để so sánh.

7. Xử lý dữ liệu thủ môn (Goalkeepers)

```python
def process_goalkeeping(merged_df, goalkeeping_df):
    """Process goalkeeping data and merge with main DataFrame."""
    if goalkeeping_df.empty or merged_df.empty:
        return merged_df
    goalkeeping_df = goalkeeping_df[goalkeeping_df['Player'].isin(merged_df['Player'])]
    pos_col = next((col for col in merged_df.columns if 'pos' in col.lower()), None)
    if pos_col:
        goalkeeping_df['Pos'] = goalkeeping_df['Player'].map(merged_df.set_index('Player')[pos_col])
        for col in goalkeeping_df.columns:
            if col not in ['Player', 'Pos']:
                goalkeeping_df[col] = goalkeeping_df.apply(
                    lambda row: row[col] if pd.notna(row['Pos']) and 'GK' in row['Pos'] else 'N/A', axis=1
                )
        goalkeeping_df.drop(columns='Pos', inplace=True)
        merged_df = pd.merge(merged_df, goalkeeping_df, on='Player', how='left')
    else:
        print("ERROR: Can't find column 'Pos'")
    return merged_df
```

-Lấy dữ liệu thủ môn từ bảng riêng.

-Áp dụng thống kê thủ môn **chỉ cho những cầu thủ có vị trí là "GK"**.

-Cầu thủ không phải thủ môn sẽ có các cột thủ môn là "N/A".

8. Ghi dữ liệu ra tệp CSV

```python
def save_output(merged_df):
    """Save the final DataFrame to CSV."""
    output_file = 'results.csv'
    try:
        merged_df.to_csv(output_file, index=False, encoding='utf-8-sig')
        print(f'Extracted file {output_file} successfully')
    except Exception as e:
        output_file = 'results_backup.csv'
        print(f'Error when saving to results.csv: {e}, try saving to {output_file}')
        merged_df.to_csv(output_file, index=False, encoding='utf-8-sig')
        print(f'Extracted file {output_file} successfully')
```

-Ghi bảng dữ liệu tổng đã xử lý ra results.csv.

-Nếu xảy ra lỗi (do encoding, file đang mở, v.v.) sẽ ghi vào results_backup.csv.

9. Hàm điều phối main()

```
def main():
    """Main function to scrape and process Premier League stats."""
    merged_df = pd.DataFrame()
    goalkeeping_df = pd.DataFrame()

    with webdriver.Chrome(service=Service(ChromeDriverManager().install())) as driver:
        driver.set_page_load_timeout(30)
        for name, (url, table_id) in TABLE_LINKS.items():
            df = fetch_table(driver, url, table_id, name)
            if df is None:
                continue

            df = process_columns(df, name)
            df = clean_data(df, name)
            if df is None:
                continue

            if name == 'Goalkeeping':
                goalkeeping_df = df.copy()
            else:
                merged_df = merge_data(merged_df, df, name)
```

```
    merged_df = filter_players(merged_df)
    merged_df = process_goalkeeping(merged_df, goalkeeping_df)

    must_have = ['Player', 'Nation', 'Squad', 'Pos', 'Age']
    available_columns = [col for col in FINAL_COLUMNS if col in merged_df.columns]
    if not available_columns:
        print("Error: Can't find any in the required columns")
        print(f"Columns available in merged_df: {list(merged_df.columns)}")
        return

    final_columns = must_have + [col for col in available_columns if col not in must_have]
    final_columns = [col for col in final_columns if col in merged_df.columns]
    merged_df = merged_df[final_columns]
    print(f"Final columns: {final_columns}")

    merged_df.fillna('N/A', inplace=True)
    save_output(merged_df)
```

Toàn bộ quy trình được điều phối trong hàm chính:

-Khởi tạo trình duyệt bằng webdriver.Chrome.

-Duyệt qua từng bảng, gọi các hàm:

+fetch_table()

+process_columns()

+clean_data()

+merge_data()

-Lọc cầu thủ đủ điều kiện thi đấu.

-Xử lý bảng thủ môn.

-Chọn đúng cột cần thiết từ FINAL_COLUMNS.

-Ghi kết quả ra tệp CSV.

BÀI TẬP 2

1. Cấu hình và tham số cố định

```
np.random.seed(42)  # Seed ngẫu nhiên
plt.rcParams.update({'font.size': 12})  # Cố định font size
HISTOGRAM_BINS = 20  # Số bins cố định cho biểu đồ
COLOR_PALETTE = {'attack': 'steelblue', 'defense': 'orange'}  # Bảng màu cố định
```

-Cố định các tham số để biểu đồ có tính nhất quán.
-COLOR_PALETTE: Xác định màu sắc cho biểu đồ tấn công và phòng ngự.

2. Cấu hình đường dẫn và tên tệp

```
TOP_N = 3
TOP_PLAYERS_FILE = 'top_3.txt'
RESULTS_FILE = 'results2.csv'
HISTOGRAM_DIR = 'histograms'
ENCODING = 'utf-8-sig'
CSV_FILE = 'results.csv'
```

-results.csv: Dữ liệu đầu vào (từ phần thu thập).
-top_3.txt: Lưu cầu thủ có chỉ số cao và thấp nhất.
-results2.csv: Lưu kết quả thống kê cho từng đội.
-histograms : Thư mục chứa biểu đồ trực quan.

3. Các chỉ số phân tích

```
attacking_cols = ['Standard SoT/90', 'Standard G/Sh', 'Standard Dist']
defensive_cols = ['Tackles Tkl', 'Tackles TklW', 'Blocks']
```

-Nhóm các chỉ số thành “tấn công” và “phòng ngự” để phân tích riêng biệt.
-Giúp biểu đồ dễ phân biệt màu sắc và ý nghĩa.

4. Hàm load_data() – Đọc và chuẩn hóa dữ liệu

```python
def load_data():
    """Đọc và chuẩn hóa dữ liệu với xử lý lỗi chi tiết"""
    try:
        df = pd.read_csv(CSV_FILE, encoding=ENCODING)
        print(f"✅ Đọc file '{CSV_FILE}' thành công")

        # Chuẩn hóa dữ liệu số
        for col in df.columns:
            if col not in ['Player', 'Nation', 'Squad', 'Pos', 'Age']:
                if df[col].dtype == 'object' and df[col].str.contains('%', na=False).any():
                    df[col] = df[col].str.replace('%', '', regex=False)
                df[col] = pd.to_numeric(df[col], errors='coerce')
        return df

    except FileNotFoundError:
        print(f"❌ Lỗi: Không tìm thấy file '{CSV_FILE}'. Vui lòng kiểm tra lại đường dẫn.")
        exit()
    except Exception as e:
        print(f"❌ Lỗi đọc file: {str(e)}")
        exit()
```

-Đọc dữ liệu từ tệp results.csv.
-Chuẩn hóa các giá trị dạng chuỗi thành số (loại bỏ dấu % nếu có).
-Nếu file không tồn tại hoặc đọc lỗi sẽ thông báo và kết thúc chương trình.

5. Hàm save_top_players() – Lưu top/bottom 3 cầu thủ

```python
def save_top_players(df, numeric_columns):
    """Lưu top players với sắp xếp ổn định"""
    try:
        with open(TOP_PLAYERS_FILE, 'w', encoding=ENCODING) as f:
            for col in numeric_columns:
                sorted_df = df[['Player', col]].dropna().sort_values(
                    by=[col, 'Player'],
                    ascending=[False, True]
                )
                f.write(f"📊 Thống kê: {col}\n")
                f.write("=" * 50 + "\n")
                top_3 = sorted_df.head(TOP_N)
                f.write("🏆 Top 3:\n")
                f.write(top_3.to_string(index=False, header=False) + "\n\n")
                bottom_3 = df[['Player', col]].dropna().sort_values(
                    by=[col, 'Player'],
                    ascending=[True, True]
                ).head(TOP_N).iloc[::-1]
                f.write("🔻 Bottom 3:\n")
                f.write(bottom_3.to_string(index=False, header=False) + "\n")
                f.write("-" * 50 + "\n\n")
    except Exception as e:
        print(f"❌ Lỗi khi lưu top players: {str(e)}")
```

-Duyệt qua từng chỉ số số học (numeric_columns).

-Xác định 3 cầu thủ có giá trị cao nhất (Top 3) và thấp nhất (Bottom 3) cho mỗi chỉ số.
-Lưu kết quả vào top_3.txt.

6. Hàm generate_plots() – Tạo biểu đồ histogram

```
def generate_plots(df, numeric_columns, prefix='all'):
    """Tạo biểu đồ với style cố định"""
    try:
        out_dir = os.path.join(HISTOGRAM_DIR, prefix)
        os.makedirs(out_dir, exist_ok=True)
        for col in numeric_columns:
            safe_col = re.sub(r'[^\w]', '_', col)
            plt.figure(figsize=(10, 6), dpi=100)
            color = COLOR_PALETTE['attack'] if col in attacking_cols else COLOR_PALETTE['defense']
            plt.hist(df[col].dropna(), bins=HISTOGRAM_BINS, color=color, edgecolor='black')
            plt.title(f"Phân bố {col} ({prefix})")
            plt.xlabel(col)
            plt.ylabel("Số lượng")
            plt.tight_layout()
            plt.savefig(os.path.join(out_dir, f"{safe_col}_{prefix}.png"))
            plt.close()
    except Exception as e:
        print(f"❌ Lỗi khi tạo biểu đồ: {str(e)}")
```

-Tạo histogram cho từng chỉ số.
-Gán màu sắc theo nhóm tấn công/phòng ngự để trực quan hơn.
-Các biểu đồ được lưu dưới tên: histograms/{prefix}/{ten_chi_so}.png.

7. Hàm main() – Điều phối toàn bộ quá trình xử lý

a. Đọc và kiểm tra dữ liệu

```
def main():
    # ========== XỬ LÝ DỮ LIỆU ==========
    df = load_data()

    required_cols = ['Player', 'Squad'] + attacking_cols + defensive_cols
    for col in required_cols:
        if col not in df.columns:
            print(f"❌ Thiếu cột quan trọng: '{col}'. Vui lòng kiểm tra file CSV.")
            exit()

    numeric_columns = attacking_cols + defensive_cols
```

-Đảm bảo các cột cần phân tích đều có mặt trong dữ liệu.
-Nếu thiếu sẽ dừng chương trình để tránh lỗi về sau.

b. Gọi hàm save_top_players() để lưu thống kê cầu thủ

```
    save_top_players(df, numeric_columns)
    print(f"✅ Đã lưu kết quả top players vào '{TOP_PLAYERS_FILE}'")
```

c. Tạo biểu đồ cho toàn giải và từng đội

```
# 2. Tạo biểu đồ ALL PLAYERS (theo cầu thủ)
generate_plots(df, numeric_columns, prefix='all_players')
print(f"✅ Đã tạo biểu đồ all_players trong '{os.path.join(HISTOGRAM_DIR, 'all_players')}'")

for team, grp in df.groupby('Squad'):
    safe_team = re.sub(r'[^\w]', '_', team)
    generate_plots(grp, numeric_columns, prefix=safe_team)
```

-Tạo biểu đồ phân phối cho tất cả cầu thủ (all_players) và từng đội bóng.
-Biểu đồ giúp hình dung mức độ phân tán chỉ số.
d. Phân tích thống kê theo từng đội

```
team_stats = []
for team, group in df.groupby('Squad'):
    stats = {'Squad': team}
    for col in numeric_columns:
        stats[f'{col}_median'] = group[col].median()
        stats[f'{col}_mean'] = group[col].mean()
        stats[f'{col}_std'] = group[col].std()
    team_stats.append(stats)
team_df = pd.DataFrame(team_stats)
```

-Tính toán trung vị, trung bình và độ lệch chuẩn cho từng đội theo từng chỉ số.
-Lưu kết quả vào team_df.
e. Tính trung bình của toàn bộ giải dựa trên các đội

```
avg_stats = {'Squad': 'allplayers'}
for col in numeric_columns:
    # Lấy cột mean của team_df
    mean_col = f'{col}_mean'
    if mean_col in team_df.columns:
        avg_stats[f'{col}_median'] = team_df[f'{col}_median'].mean()
        avg_stats[f'{col}_mean'] = team_df[f'{col}_mean'].mean()
        avg_stats[f'{col}_std'] = team_df[f'{col}_std'].mean()
# Append row
team_df = pd.concat([team_df, pd.DataFrame([avg_stats])], ignore_index=True)
```

-Dòng allplayers được thêm vào cuối để đại diện cho toàn giải.
-Được tính bằng trung bình của các đội (không phải toàn bộ cầu thủ).
f. Lưu kết quả phân tích vào file CSV

```
team_df.to_csv(RESULTS_FILE, index=False, encoding=ENCODING)
print(f"✅ Đã lưu thống kê đội (kèm allplayers) vào '{RESULTS_FILE}'")
```

-Kết quả được lưu vào results2.csv để sử dụng cho báo cáo.

BÀI TẬP 3

1. Cấu hình tham số

```
CSV_FILE = 'results.csv'
ENCODING = 'utf-8-sig'
MAX_K = 10
OPTIMAL_K = 3  # Giữ nguyên kết quả đầu ra
```

-results.csv: Tệp chứa dữ liệu đã được xử lý từ bước trước.
-MAX_K: Số lượng cụm tối đa để vẽ biểu đồ Elbow (dò tìm số cụm tối ưu).
-OPTIMAL_K: Số cụm được chọn để phân nhóm cầu thủ (dựa trên phương pháp Elbow).

2. Hàm load_and_prepare() – Chuẩn bị dữ liệu đầu vào

```
def load_and_prepare(file_path, encoding):
    """Đọc file CSV, xử lý thiếu giá trị và chuẩn hóa dữ liệu số"""
    df = pd.read_csv(file_path, encoding=encoding)
    numeric_cols = df.select_dtypes(include=[np.number]).columns.tolist()
    if not numeric_cols:
        raise ValueError("Không có cột số để phân tích.")
    # Impute missing values với trung bình cột
    imputer = SimpleImputer(strategy='mean')
    imputed = imputer.fit_transform(df[numeric_cols])
    df[numeric_cols] = imputed
    # Chuẩn hóa
    scaler = StandardScaler()
    scaled = scaler.fit_transform(df[numeric_cols])
    return df, numeric_cols, scaled
```

-Đọc tệp CSV thành DataFrame.
-Tách các cột dạng số (numeric_cols) để đưa vào mô hình học máy.
-Xử lý dữ liệu thiếu bằng cách điền giá trị trung bình (SimpleImputer).
-Chuẩn hóa dữ liệu (đưa về cùng thang đo) bằng StandardScaler.

3. Hàm plot_elbow() – Vẽ biểu đồ Elbow để chọn số cụm tối ưu

```
def plot_elbow(inertia, max_k, optimal_k):
    plt.figure(figsize=(8, 6))
    ks = range(1, max_k + 1)
    plt.plot(ks, inertia, 'o-', linewidth=2)
    plt.axvline(optimal_k, color='red', linestyle='--', label=f'Elbow at k={optimal_k}')
    plt.annotate(
        'Elbow Point',
        xy=(optimal_k, inertia[optimal_k - 1]),
        xytext=(optimal_k + 1, inertia[optimal_k - 1] + (inertia[0] - inertia[-1]) * 0.05),
        arrowprops=dict(arrowstyle='->', color='black')
    )
    plt.title('Elbow Chart: Inertia vs. Number of Clusters')
    plt.xlabel('Number of clusters (k)')
    plt.ylabel('Inertia')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

-Biểu đồ Elbow vẽ quan hệ giữa số cụm k và giá trị inertia (độ chệch nội tại).
-Điểm "gấp khúc" (elbow) cho thấy k tối ưu vì sau đó inertia giảm không đáng kể.

4. Hàm perform_kmeans() – Huấn luyện mô hình Kmeans

```
def perform_kmeans(data, k, random_state=42):
    model = KMeans(n_clusters=k, random_state=random_state)
    labels = model.fit_predict(data)
    return labels, model
```

-Huấn luyện mô hình phân cụm KMeans với số cụm k.
-Trả về **nhãn phân cụm** (labels) cho mỗi cầu thủ và mô hình đã huấn luyện.

5. Hàm summarize_clusters() – Tóm tắt từng cụm

```
def summarize_clusters(df, labels, numeric_cols):
    df['Cluster'] = labels
    summary = df.groupby('Cluster')[numeric_cols].mean().round(2)
    print("\nCluster Summary (Mean values):")
    print(summary)
    return df
```

-Gắn nhãn cụm cho từng cầu thủ trong DataFrame.
-Tính trung bình các chỉ số số học theo từng cụm để mô tả đặc điểm.

6. Hàm plot_pca_clusters() – Biểu diễn cụm bằng PCA 2D

```python
def plot_pca_clusters(data, labels):
    pca = PCA(n_components=2)
    comps = pca.fit_transform(data)
    df_pca = pd.DataFrame(comps, columns=['PCA1', 'PCA2'])
    df_pca['Cluster'] = labels

    plt.figure(figsize=(10, 8))
    for c in sorted(df_pca['Cluster'].unique()):
        subset = df_pca[df_pca['Cluster'] == c]
        plt.scatter(subset['PCA1'], subset['PCA2'], s=80, alpha=0.7, label=f'Cluster {c}')
    plt.title('PCA 2D Cluster Plot')
    plt.xlabel('PCA Component 1')
    plt.ylabel('PCA Component 2')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

-Sử dụng PCA để giảm số chiều dữ liệu xuống còn 2.
-Vẽ biểu đồ 2D thể hiện phân cụm trực quan.
-Mỗi điểm đại diện một cầu thủ, màu sắc đại diện cụm.

7. Hàm main() – Điều phối toàn bộ quy trình

a. Đọc và chuẩn hóa dữ liệu

```python
try:
    df, numeric_cols, scaled = load_and_prepare(CSV_FILE, ENCODING)
    print(f"✅ Đã đọc và chuẩn hóa dữ liệu từ '{CSV_FILE}'")
except Exception as e:
    print(f"❌ Lỗi khi chuẩn bị dữ liệu: {e}")
    return
```

b. Vẽ biểu đồ Elbow để chọn số cụm

```python
inertia = []
for k in range(1, MAX_K + 1):
    _, model = perform_kmeans(scaled, k)
    inertia.append(model.inertia_)
plot_elbow(inertia, MAX_K, OPTIMAL_K)

labels, _ = perform_kmeans(scaled, OPTIMAL_K)
df = summarize_clusters(df, labels, numeric_cols)
plot_pca_clusters(scaled, labels)
```

c. Phân cụm với k=3

```python
labels, _ = perform_kmeans(scaled, OPTIMAL_K)
```

d. Tóm tắt các cụm

```python
df = summarize_clusters(df, labels, numeric_cols)
```

e. Trực quan hóa kết quả phân cụm bằng PCA

```python
plot_pca_clusters(scaled, labels)
```

f. Đưa ra bình luận dựa trên kết quả phân cụm

```
print("\nComment:")
print("Chọn k=3 cho KMeans dựa trên Elbow Method, vì sau k=3, inertia giảm chậm lại, cho thấy ba cụm phân biệt tốt nhất.")
print("Cluster 0: Defensive players (defenders, midfielders, goalkeepers) với chỉ số tấn công thấp.")
print("Cluster 1: Substitute/lower-contribution players với điểm số trung bình thấp.")
print("Cluster 2: Attacking players với số bàn thắng và kiến tạo cao.")
```

BÀI TẬP 4

1. Thu thập dữ liệu ETV từ trang web

a. Khởi tạo Selenium và lấy dữ liệu theo trang

```
def data(filtered_df):
    players = {}
    for name in filtered_df['Player']:
        players[name] = ''
    driver = webdriver.Chrome()
    url = 'https://www.footballtransfers.com/us/players/uk-premier-league/'
    driver.get(url)
    names = []
    values = []
    cnt = 0
    while cnt < 22:
        time.sleep(1)
        page_source = driver.page_source
        soup = BeautifulSoup(page_source, 'html.parser')
        table = soup.find('table', class_='table table-hover no-cursor table-striped leaguetable mvp-table similar-players-table mb-0')
        if table is None:
            print("Không tìm thấy bảng dữ liệu.")
            break
```

-Sử dụng Selenium để tự động mở trang web Premier League trên *footballtransfers.com*.

-Duyệt qua tối đa 22 trang để thu thập thông tin cầu thủ.

b. Trích xuất tên và giá trị chuyển nhượng

```
        name_tags = table.find_all('div', class_='text')
        price_tags = table.find_all('span', class_='player-tag')
```

-name_tags: chứa tên cầu thủ (trong thẻ <a>).

-price_tags: chứa giá trị chuyển nhượng dạng văn bản, ví dụ "€45m".

c. Gán giá trị vào từ điển players

```
for i in range(len(names)):
    if names[i] in players:
        players[names[i]] = values[i]
return players
```

-Chỉ giữ lại những cầu thủ xuất hiện trong filtered_df (có >900 phút thi đấu).

-Kết quả là một dictionary players: name → ETV.

2. Gán ETV vào DataFrame và lưu ra file

```
market_value_df = filtered_df[['Player', 'ETV']]
market_value_df.to_csv('market value.csv', index=False, encoding='utf-8-sig')
print("Đã lưu tệp 'market value.csv'.")
```

Thêm cột ETV vào bảng dữ liệu chính.

-Lưu danh sách cầu thủ kèm giá trị chuyển nhượng vào file market value.csv.

3. Chuẩn bị dữ liệu huấn luyện cho mô hình

a. Lọc đặc trưng có thể dùng được

```
features = ['Performance Ast', 'Age', 'Playing Time Min', 'Playing Time MP', 'Performance Gls', 'Progression PrgR', 'GCA']
```

-Chọn các đặc trưng thống kê liên quan đến khả năng kiến tạo, tuổi, thời gian thi đấu, ghi bàn, đóng góp tấn công...

-Hàm get_numeric_columns() lọc ra các cột có thể chuyển sang số được và không bị thiếu dữ liệu quá nhiều.

b. Xây dựng tập dữ liệu cho mô hình

```
model_df = filtered_df.dropna(subset=available_features + ['ETV'])
print(f"Số cầu thủ có ETV hợp lệ cho mô hình: {len(model_df)}")

X = model_df[available_features]
y = model_df['ETV']
```

-Loại bỏ các cầu thủ chưa có dữ liệu ETV hoặc thiếu một trong các đặc trưng.

-X: ma trận đặc trưng đầu vào.

-y: giá trị mục tiêu (ETV).

4. Huấn luyện mô hình hồi quy tuyến tính

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

model = LinearRegression()
model.fit(X_train, y_train)
```

-Dữ liệu được chia 80% để huấn luyện và 20% để kiểm tra.

-Sử dụng mô hình hồi quy tuyến tính (LinearRegression) từ thư viện sklearn.

5. Đánh giá mô hình

```
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Sai số bình phương trung bình: {mse:.2f}')
print(f'Điểm R²: {r2:.2f}')
```

-MSE (Mean Squared Error): đo sai số trung bình giữa giá trị dự đoán và thực tế.

-R^2 (R-squared): đo mức độ mô hình giải thích được phương sai của dữ liệu. Giá trị càng gần 1 càng tốt.

6. Phân tích kết quả

```
print('Hệ số:', model.coef_)
print('Hằng số:', model.intercept_)
```

-Các hệ số cho biết mức độ ảnh hưởng của từng đặc trưng đến giá trị chuyển nhượng.

-Hệ số dương → đặc trưng càng cao thì ETV càng tăng (ví dụ: bàn thắng).

-Hệ số âm → đặc trưng càng cao thì ETV giảm (có thể là tuổi hoặc số trận ít).