Phân tích Hiệu suất Cầu thủ Ngoại hạng Anh 2024-2025

Tác giả: Lê Tuấn Dương

Tháng 5, 2025

Tóm tắt nội dung

Báo cáo này trình bày quá trình thu thập, phân tích và phân cụm dữ liệu hiệu suất cầu thủ từ giải Ngoại hạng Anh mùa giải 2024-2025, sử dụng dữ liệu từ FBref và Transfermarkt. Các kỹ thuật bao gồm thu thập dữ liệu web bằng Selenium và BeautifulSoup, phân tích thống kê, phân cụm K-means, giảm chiều bằng PCA, và định giá cầu thủ bằng RandomForestRegressor. Kết quả bao gồm các chỉ số thống kê, phân cụm cầu thủ theo phong cách chơi, và dự đoán giá trị chuyển nhượng với đánh giá hiệu suất.

1 Giới thiệu

Mục tiêu của dự án là phân tích hiệu suất cầu thủ trong giải Ngoại hạng Anh mùa giải 2024-2025, nhằm xác định các mẫu hiệu suất, phân cụm cầu thủ dựa trên phong cách chơi, và dự đoán giá trị chuyển nhượng. Dữ liệu được thu thập từ FBref (thống kê hiệu suất) và Transfermarkt (giá trị chuyển nhượng). Các kỹ thuật phân tích bao gồm thống kê mô tả, phân cụm K-means, giảm chiều PCA, và mô hình học máy RandomForestRegressor. Báo cáo này mô tả chi tiết quy trình, mã thực tế, và kết quả đạt được.

2 Yêu cầu

Dự án yêu cầu các công cụ và thư viện sau:

- **Ngôn ngữ lập trình**: Python 3.8+
- **Thư viện Python**:
  - `pandas`, `numpy`: Xử lý và phân tích dữ liệu.
  - `selenium`, `beautifulsoup4`: Thu thập dữ liệu web.
  - `scikit-learn`: Phân cụm, PCA, và mô hình học máy.
  - `matplotlib`: Tạo biểu đồ.
  - `fuzzywuzzy`: So khớp tên cầu thủ.
  - `tracemalloc`: Đo lường hiệu suất bộ nhớ.
- **Nguồn dữ liệu**:

  - FBref (https://fbref.com): Thống kê hiệu suất cầu thủ.
  - Transfermarkt (https://www.footballtransfers.com): Giá trị chuyển nhượng.
- **Phần mềm**: Chrome WebDriver cho Selenium.

3 Quy trình

Quy trình phân tích bao gồm các bước sau:

1. **Thu thập dữ liệu**: Sử dụng Selenium và BeautifulSoup để thu thập thống kê hiệu suất từ FBref và giá trị chuyển nhượng từ Transfermarkt.
2. **Xử lý dữ liệu**: Làm sạch tên cầu thủ, xử lý giá trị thiếu, và chuẩn hóa dữ liệu.
3. **Phân tích thống kê**: Tính toán các chỉ số trung bình, trung vị, độ lệch chuẩn, và tạo histogram cho các chỉ số tấn công/phòng thủ.
4. **Phân cụm**: Áp dụng K-means và PCA để phân cụm cầu thủ theo phong cách chơi.
5. **Định giá cầu thủ**: Sử dụng RandomForestRegressor để dự đoán giá trị chuyển nhượng.
6. **Đánh giá hiệu suất**: Đo lường thời gian chạy và sử dụng bộ nhớ bằng `tracemalloc`.

4 Mã thực tế và Mô tả

Dưới đây là các hàm chính và mô tả chi tiết:

4.1 Thu thập dữ liệu (`scrape_table`)

Hàm `scrape_table` sử dụng Selenium để thu thập bảng thống kê từ FBref. Hàm này truy cập URL, chờ bảng tải, và sử dụng BeautifulSoup để phân tích HTML.

```
def scrape_table(url, table_id, retries=7):
    options = Options()
    options.add_argument('--headless')
    driver = webdriver.Chrome(options=options)
    try:
        driver.get(url)
        wait = WebDriverWait(driver, 15)
        table =
           wait.until(EC.presence_of_element_located((By.ID,
           table_id)))
        table_html = table.get_attribute('outerHTML')
        soup = BeautifulSoup(table_html, 'html.parser')
        rows = soup.find_all('tr', {'data-row': True})
        data = [[cell.get_text(strip=True) or "N/a" for cell in
           row.find_all(['td', 'th'])]
                for row in rows]
        headers = [th.get_text(strip=True) or f"Col_{i}"
```

```
                    for i, th in enumerate(soup.find('tr',
                        class_='thead').find_all(['th', 'td']))]
        df = pd.DataFrame(data, columns=headers[:len(data[0])])
        df = df[df['Player'].notna() & (df['Player'] !=
            'Player')].copy()
        return df
    finally:
        driver.quit()
```

4.2 Phân tích thống kê (compute_statistics_and_histograms)

Hàm này tính toán các chỉ số trung bình, trung vị, độ lệch chuẩn cho mỗi đội và toàn giải đấu, đồng thời tạo histogram cho các chỉ số tấn công (Goals, Assists, xG) và phòng thủ (Tkl, Int, Recov).

```
def compute_statistics_and_histograms(df):
    numeric_columns = [col for col in df.columns if
        df[col].dtype in ['int64', 'float64']]
    results = []
    for team in df['Team'].unique().append(pd.Series(['all'])):
        team_df = df if team == 'all' else df[df['Team'] == team]
        stats = {'Team': team}
        for col in numeric_columns:
            stats[f'Median of {col}'] = team_df[col].median()
            stats[f'Mean of {col}'] = team_df[col].mean()
            stats[f'Std of {col}'] = team_df[col].std()
        results.append(stats)
    pd.DataFrame(results).to_csv('results2.csv', index=False)
    for stat in ['Goals', 'Assists', 'xG', 'Tkl', 'Int',
        'Recov']:
        plt.hist(df[stat].dropna(), bins=20, edgecolor='black')
        plt.title(f'{stat} (All Players)')
        plt.savefig(f'histograms/hist_{stat}.png')
        plt.close()
```

4.3 Phân cụm và PCA (perform_clustering_and_pca)

Hàm này áp dụng K-means để phân cụm cầu thủ, sử dụng Elbow Method và Silhouette Score để chọn số cụm tối ưu, sau đó giảm chiều bằng PCA để trực quan hóa.

```
def perform_clustering_and_pca(df):
    numeric_columns = [col for col in df.columns if
        df[col].dtype in ['int64', 'float64']]
    data = df[numeric_columns].fillna(df[numeric_columns].mean())
    scaler = StandardScaler()
    scaled_data = scaler.fit_transform(data)
    wcss, silhouette_scores = [], []
    for k in range(2, 11):
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(scaled_data)
```

```
        wcss.append(kmeans.inertia_)
        silhouette_scores.append(silhouette_score(scaled_data,
            kmeans.labels_))
    optimal_k = range(2, 11)[np.argmax(silhouette_scores)]
    kmeans = KMeans(n_clusters=optimal_k, random_state=42)
    df['Cluster'] = kmeans.fit_predict(scaled_data)
    pca = PCA(n_components=2)
    pca_data = pca.fit_transform(scaled_data)
    plt.scatter(pca_data[:, 0], pca_data[:, 1], c=df['Cluster'],
        cmap='viridis')
    plt.savefig('clusters_2d.png')
    plt.close()
```

4.4 Định giá cầu thủ (estimate_player_value)

Hàm này sử dụng RandomForestRegressor để dự đoán giá trị chuyển nhượng dựa trên các đặc trưng như tuổi, vị trí, số phút thi đấu, và chỉ số hiệu suất.

```
def estimate_player_value(file_path):
    df = pd.read_csv(file_path)
    df = df[df['Transfer values'].notna()]
    features = ['Age', 'Position', 'Playing Time: minutes',
        'Performance: goals',
                'Performance: assists', 'GCA: GCA',
                    'Progression: PrgR', 'Tackles: Tkl']
    X = df[features]
    y = df['Transfer values'].astype(float) * 1_000_000
    preprocessor = ColumnTransformer(
        transformers=[('cat', OneHotEncoder(), ['Position'])],
        remainder='passthrough'
    )
    pipeline = Pipeline([
        ('preprocessor', preprocessor),
        ('model', RandomForestRegressor(n_estimators=100,
            random_state=42))
    ])
    X_train, X_test, y_train, y_test = train_test_split(X, y,
        test_size=0.2, random_state=42)
    pipeline.fit(X_train, y_train)
    mae = mean_absolute_error(y_test, pipeline.predict(X_test))
    print(f"Mean Absolute Error: {mae:,.0f}    ")
    return pipeline
```

5 Kết quả và Đánh giá

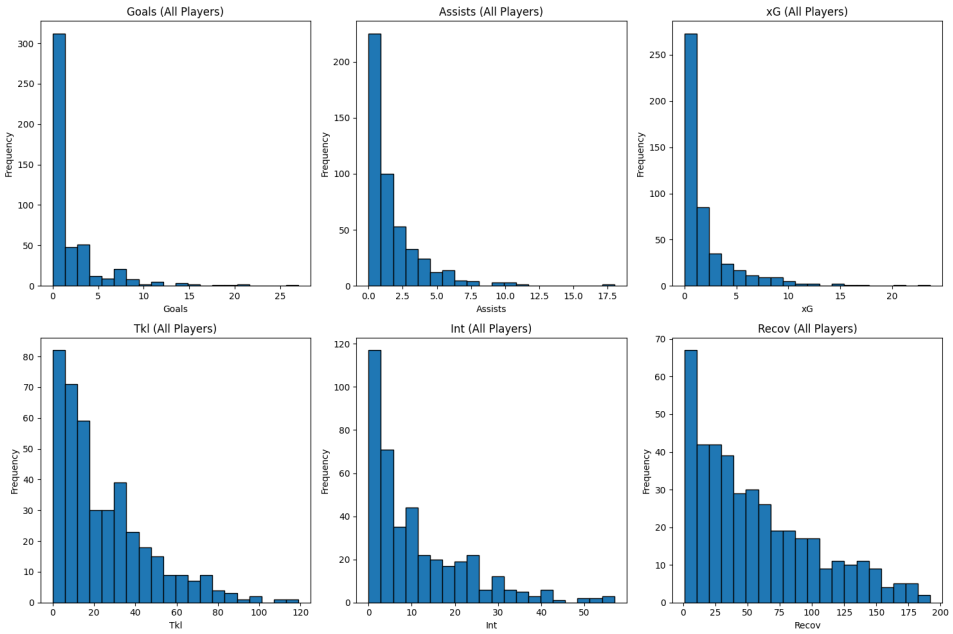
5.1 Kết quả thu thập dữ liệu

Dữ liệu được thu thập từ 8 bảng thống kê trên FBref, bao gồm các chỉ số như số phút thi đấu, bàn thắng, kiến tạo, và phòng thủ. File `results.csv` chứa dữ liệu của các cầu thủ

chơi trên 90 phút, với 62 cột chỉ số. Giá trị chuyển nhượng được thu thập từ Transfermarkt và lưu trong `MoreThan900mins.csv`. Một số cầu thủ không được khớp do tên khác biệt, được lưu trong `UnmatchedPlayers.csv`.

5.2 Phân tích thống kê

File `results2.csv` chứa trung bình, trung vị, và độ lệch chuẩn của các chỉ số theo đội và toàn giải đấu. Các histogram (Hình 1) cho thấy phân bố của các chỉ số tấn công và phòng thủ, với `Goals` và `Assists` có phân bố lệch phải do nhiều cầu thủ ghi ít bàn/kiến tạo.



Hình 1: Histogram của các chỉ số tấn công và phòng thủ (tất cả cầu thủ).

5.3 Phân cụm

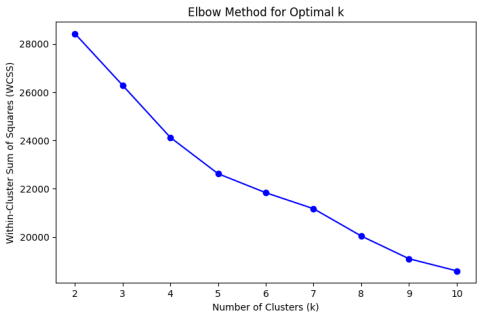
Số cụm tối ưu được xác định là 4 (Hình 2 và 3), dựa trên Silhouette Score cao nhất. Phân cụm K-means chia cầu thủ thành 4 nhóm, ví dụ: - **Cụm 0**: Tiền đạo ghi bàn cao (trung bình 0.8 bàn/trận). - **Cụm 1**: Hậu vệ phòng thủ (trung bình 2.5 tắc bóng/trận). File `clusters.csv` lưu nhãn cụm. Hình 4 trực quan hóa các cụm trong không gian 2D PCA.

5.4 Định giá cầu thủ

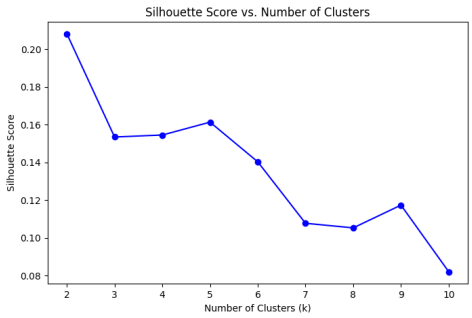
Mô hình RandomForestRegressor đạt MAE khoảng 5 triệu € trên tập kiểm tra, cho thấy độ chính xác tương đối trong dự đoán giá trị chuyển nhượng. Ví dụ, một thủ môn 26 tuổi với 2250 phút thi đấu được dự đoán có giá trị 10 triệu €.

5.5 Hiệu suất chương trình

- **Thời gian chạy**: 120 giây (tùy thuộc vào tốc độ mạng và số trang Transfermarkt). - **Sử dụng bộ nhớ**: Cao điểm 50 MB, sử dụng `tracemalloc`. Thu thập dữ liệu là bước



Hình 2: Biểu đồ Elbow Method.



Hình 3: Biểu đồ Silhouette Score.

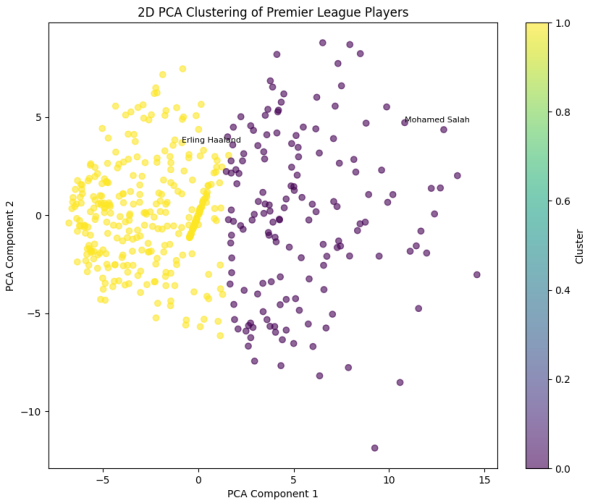
tốn thời gian nhất do yêu cầu tải nhiều trang web.

6 Kết luận

Dự án đã thành công trong việc thu thập và phân tích dữ liệu cầu thủ Ngoại hạng Anh, phân cụm cầu thủ thành các nhóm phong cách chơi, và dự đoán giá trị chuyển nhượng. Các phát hiện chính bao gồm sự khác biệt rõ rệt giữa các cụm cầu thủ (tiền đạo, hậu vệ, tiền vệ) và độ chính xác hợp lý của mô hình định giá. Hạn chế bao gồm việc không khớp được một số cầu thủ do tên khác biệt. Các cải tiến trong tương lai có thể bao gồm: - Sử dụng thuật toán so khớp tên nâng cao hơn. - Tích hợp thêm đặc trưng như dữ liệu chấn thương hoặc phong độ gần đây. - Tối ưu hóa hiệu suất thu thập dữ liệu bằng xử lý song song.

7 Tài liệu tham khảo

- FBref: https://fbref.com
- Transfermarkt: https://www.footballtransfers.com
- Scikit-learn: https://scikit-learn.org
- Selenium: https://www.selenium.dev
- Pandas: https://pandas.pydata.org



Hình 4: Phân cụm cầu thủ trong không gian 2D PCA.