

OOF 프로젝트

유럽 축구 리그 기반 경기, 선수 예측 및 Chatbot 서비스 구현

| | |
|---------|-----------------------------------|
| 팀명 | Adios |
| 팀원 | 김동휘, 김성일, 임경란, 정현수, 서한울, 신대근, 오지영 |
| 프로젝트 기간 | '24. 04. 18 ~ 24. 04. 25 |

CONTENTS

01



[프로젝트 진행 결과](#)

02



[개요](#)

- R&R
- [프로젝트 수립 배경](#)
- [프로젝트 진행 목표](#)

03



[프로젝트 수행 내용](#)

- DATA EDA
- AI 모델링
- Langchain

04



[중점 및 미흡 사항](#)

05



[부록](#)

- 사용 라이브러리
- [코드](#)



01.

프로젝트 진행 결과 : OOF 플랫폼 구축 프로젝트

| | | |
|---------------|--|---|
| 프로젝트명 | OOF(Oracle of Football) 플랫폼 구축 프로젝트 | |
| 기획 목표 | <p>유럽 5대 축구 리그의 선수 및 경기 내용 Data를 기반으로 구현한 기능 서비스를 통해 축구 스카우터, 게임 유저 및 팬 등에게 경쟁력 있는 서비스 제공</p> | |
| 투입 인원 및 소요 기간 | 총 7명 / 8일 (24.04.18 ~ 24.04.25) | |
| 기능 서비스 | <p>선수 정보</p> <p>유망주 여부 예측</p> <p>포지션별 유사 선수 분류 조회</p> <p>선수별 속성 비교 시각화</p> <p>Langchain 선수 정보 데이터 분석 Chatbot</p> | <p>경기 정보</p> <p>딥러닝 기반 승부 예측</p> <p>Langchain 기반 경기 내용 조회 Chatbot</p> <p>Langchain 기반 Foot-Ball RSS(Rich Site Summary) 요약 봇</p> |
| Web 배포 | https://sayoof-adios.streamlit.app/ | |

02-1. 개요

R&R



02-2. 개요 서비스 기획 배경

1 기획 배경

[24년도 기준, 아직 관련 뉴스 기사]

이적료 역대 1위, 주급 1위... 아무도 반박할 수 없는 **최악의 영입**, 매각도 불가
'1,934억 최악의 막튀' 기록, 6년 동안 안 깨졌네... 역대 MF 최고 이적료는?
토트넘 **1080억 헛돈** → '최악 막튀' 몸값 160억 급추락...
투자금 회수마저 '푼돈' 위기

문제점

AI기반 프로세스로 선수 영입을 하고 있으나,
해당 선수의 현지리스에서 활약과 성장 가능성 판별 미지수

Needs

구단 입장에서 성공적인 영입은 현재 경쟁력보다
선수의 성장 가능성에 따른 종합 능력치 + 가성비 좋은 선수

2 기획 배경

[FIFA 모바일 월 다운로드 수 증감률]



문제점

축구 관련 게임 **다운로드 수는 지속적 증가**
But, 게임사의 과금 유도 프로세스와 지나친 뽑기 확률로
유저수 감소 中

Needs

무과금으로도 게임 플레이를 할 수 있는 **환경 및 수단 필요**

02-3. 개요

OOF 플랫폼 서비스



우프 플랫폼 [OOF(Oracle of Football)]

“영화 매트릭스의 오라클은 예언자로 매트릭스에서 일어날 일들을 모두 관찰하여,
주인공 네오에게 조언과 예측으로 길을 이끌어주고 선택하게 만든다.”

이처럼 'Oracle of Football'은
유럽 리그별 팀 스카우터들이 기록한 실제 데이터를 바탕으로
~~축구와 관련된 모든 사용자들에게 아래와 같은 서비스 제공~~

| | 서비스기능 | 타겟 |
|-------|---|--|
| 선수 정보 | 신규 선수 속성에 대한 유망주 여부 예측 신규 선수 속성에 대한 포지션별 유사 선수 분류 조회 선수별 속성 비교 시각화 Langchain 기반 선수 정보 데이터 분석 Chatbot | · 축구팀 스카우터 |
| 경기 정보 | 딥러닝(Softmax) 기반 승부 예측 Langchain 기반 경기 내용 조회 Chatbot Langchain 기반 Foot-Ball RSS(Rich Site Summary) 요약 봇 | · 축구 게임(FIFA, FM, FC 등) 유저 · 해외 및 국내 축구 팬 |

03-1. 프로젝트 수행 내용

DATA 정보

1. FM Inside

| | |
|-----------|---|
| 데이터소개 | 1. 전세계 축구리그 및 팀 소속 스카우들이 선수에 대한 포지션별 속성을 수치화한 데이터 2. 포지션별 속성 분류상(주 속성: Technical, Mental, Physical, GoalKeeping) |
| 데이터출처 | FM Inside (https://fminside.net) |
| 수집방법 및 건수 | 크롤링(Selenium, BeautifulSoup) / 25,862건 |
| 활용방안 | LangChain 기반 선수분석 Chatbot / 유망주 & 선수별 잠재력 예측 포지션별 유사선수 클러스터링 |

Kevin De Bruyne



95
95

Manchester City
Belgium
MC, AMC

| PLAYER INFO | |
|--------------|-----------------|
| Name | Kevin De Bruyne |
| Age | 32 |
| Position(s) | MC, MR, ML, AMC |
| Foot | Right |
| Height | 181 CM |
| Weight | 68 KG |
| Caps / Goals | 99 / 26 |
| Unique ID | 18004457 |

Attributes

| TECHNICAL | MENTAL | PHYSICAL | | | |
|------------------|--------|---------------|----|-----------------|----|
| Corners | 14 | Aggression | 12 | Acceleration | 16 |
| Crossing | 19 | Anticipation | 14 | Agility | 13 |
| Dribbling | 16 | Bravery | 13 | Balance | 14 |
| Finishing | 16 | Composure | 14 | Jumping Reach | 10 |
| First Touch | 16 | Concentration | 15 | Natural Fitness | 16 |
| Free Kick Taking | 17 | Decisions | 18 | Pace | 14 |
| Heading | 6 | Determination | 17 | Stamina | 16 |
| Long Shots | 17 | Flair | 16 | Strength | 13 |
| Long Throws | 7 | Leadership | 13 | | |
| Marking | 9 | Off the Ball | 15 | | |
| Passing | 18 | Positioning | 10 | | |
| Penalty Taking | 15 | Teamwork | 14 | | |
| Tackling | 9 | Vision | 20 | | |
| Technique | 18 | Work Rate | 15 | | |

| player_nm | player_overall | player_potential | player_team | player_country | player_position | player_age | player_foot | player_height | player_weight | aerial-reach | command-of-area |
|-----------------------|----------------|------------------|-----------------|----------------|-----------------|------------|-------------|---------------|---------------|--------------|-----------------|
| Thibaut Courtois | 91 | 91 | Real Madrid | Belgium | GK | 31 | Left | 200 | 96 | 20 | 16 |
| Ederson | 90 | 90 | Manchester City | Brazil | GK | 29 | Left | 188 | 86 | 13 | 16 |
| Alisson | 88 | 90 | Liverpool | Brazil | GK | 30 | Right | 191 | 91 | 15 | 13 |
| Marc-André ter Stegen | 88 | 92 | FC Barcelona | Germany | GK | 31 | Right | 187 | 85 | 13 | 15 |
| Jan Oblak | 86 | 89 | Atlético Madrid | Slovenia | GK | 30 | Right | 189 | 84 | 15 | 15 |

〈FM Inside 사이트〉

〈웹 크롤링 → Data Frame 변환〉

03-1. 프로젝트 수행 내용

DATA 정보

2. FOTMOB

| | |
|------------|--|
| 데이터 소개 | 23~4년도 유럽 5대 축구 리그별 경기 결과 & 출전 선수 라인업 |
| 데이터 출처 | FotMob (https://fotmob.com) |
| 수집 방법 및 건수 | 웹크롤링(Selenium, BeautifulSoup) / 1,471건 |
| 활용 방안 | Langchain 기반 경기 내용 조회 Chatbot 팀, 포지션별 속성 수치 기반으로 승부 예측 |

| leagueNation | league | homeTeam | awayTeam | homeScore | awayScore | matchResult |
|--------------|--------|------------------------|-------------------|-----------|-----------|-------------|
| England | EPL | Burnley | Manchester City | 0 | 3 | Away |
| England | EPL | Arsenal | Nottingham Forest | 2 | 1 | Home |
| England | EPL | AFC Bournemouth | West Ham United | 1 | 1 | Draw |
| England | EPL | Brighton & Hove Albion | Luton Town | 4 | 1 | Home |
| England | EPL | Everton | Fulham | 0 | 1 | Away |
| England | EPL | Sheffield United | Crystal Palace | 0 | 1 | Away |
| England | EPL | Newcastle United | Aston Villa | 5 | 1 | Home |
| England | EPL | Brentford | Tottenham Hotspur | 2 | 2 | Draw |
| England | EPL | Chelsea | Liverpool | 1 | 1 | Draw |

3. Transfer Market

| | |
|------------|--|
| 데이터 소개 | 24년도 유럽 5대 축구리그 1~5부 선수별 Market Value |
| 데이터 출처 | Transfer Markt (https://transfermarket.com) |
| 수집 방법 및 건수 | 웹크롤링(Selenium, BeautifulSoup) / 12,913건 |
| 활용 방안 | 포지션별 유사선수 클러스터링, 선수 Market Value 예측 |

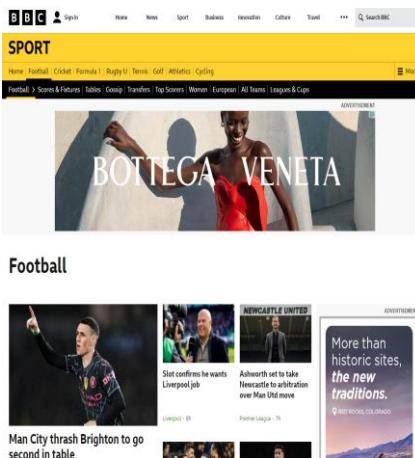
| team name | num | name | age | nationality | position | market value |
|-----------------------|-----|----------------|-----|-------------|-------------|--------------|
| Squad Manchester City | 31 | Ederson | 30 | Brazil | Goalkeeper | €40.00m |
| Squad Manchester City | 18 | Stefan Ortega | 31 | Germany | Goalkeeper | €9.00m |
| Squad Manchester City | 33 | Scott Carson | 38 | England | Goalkeeper | €200k |
| Squad Manchester City | 3 | Rúben Dias | 26 | Portugal | Centre-Back | €80.00m |
| Squad Manchester City | 24 | Josko Gvardiol | 22 | Croatia | Centre-Back | €75.00m |
| Squad Manchester City | 25 | Manuel Akanji | 28 | Switzerland | Centre-Back | €42.00m |
| Squad Manchester City | 6 | Nathan Aké | 29 | Netherlands | Centre-Back | €40.00m |
| Squad Manchester City | 5 | John Stones | 29 | England | Centre-Back | €38.00m |
| Squad Manchester City | 21 | Sergio Gómez | 23 | Spain | Left-Back | €10.00m |
| Squad Manchester City | 82 | Rico Lewis | 19 | England | Right-Back | €38.00m |

03-1. 프로젝트 수행 내용

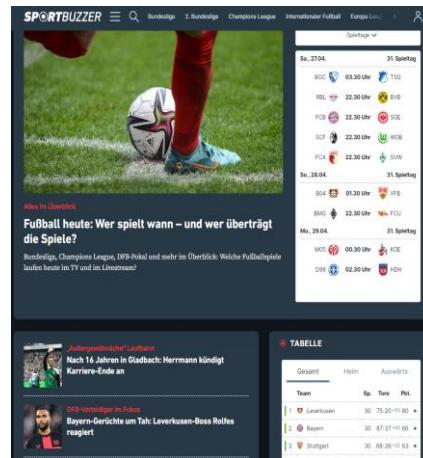
DATA 정보

4. 유럽 현지뉴스 기사

| | |
|------------|--|
| 데이터소개 | 24년도 유럽 축구리그 기사사이트 |
| 데이터출처 | (1) 영국 : BBC (https://www.bbc.com/sport/football) (2) 독일 : Sportbuzzer (https://www.sportbuzzer.de) (3) 스페인 : ABC Deportes (http://www.abc.es/deportes/futbol-formula1-tenis.asp) (4) 이탈리아 : Gianluca Dimarzio (http://gianlucadimarzio.com) |
| 수집 방법 및 건수 | 크롤링(Selenium, BeautifulSoup) / 시간별 40건 |
| 활용 방안 | 랭체인 기반 Foot-Ball RSS(Rich Site Summary) 요약 챗봇 |



〈영국뉴스기사〉



〈독일뉴스기사〉



〈스페인뉴스기사〉



〈이탈리아뉴스기사〉

03-2. 프로젝트 수행 내용

AI 모델링(1) : ML / 능력치 및 잠재력 예측

1. 유망주&선수별 잠재력 예측

- 활용 데이터 : FMInside사이트의 Players 속성 Values
- 활용 용도 : 데이터에 의해 학습된 머신러닝 모델로 능력치 및 잠재력 예측 모델 구축
능력치에 의거한 유망한 선수 판별 가능

구현 코드 방식

```
# pycaret을 이용해서 최적의 모델 구하기
X_train, X_test = train_test_split(ss_DC, train_size = 0.7, random_state = 123)
potential_df = setup(data = X_train, target = 'player_potential', test_data = X_test, session_id = 123)

# RMSE 지표를 기준으로 최적의 모델 선정
best = compare_models(n_select=30, sort='RMSE')

# 모델 생성하기(Gradient Boosting Regressor)
GBR = create_model('gbr')

# 모델 튜닝
tuned_gbr = tune_model(GBR, optimize = 'RMSE', n_iter = 100)

# 마지막 학습(Finalize)시키기
# R2 0.6211
final_model = finalize_model(tuned_gbr)
```

▲ Gradient Boosting Regressor 모델 사용

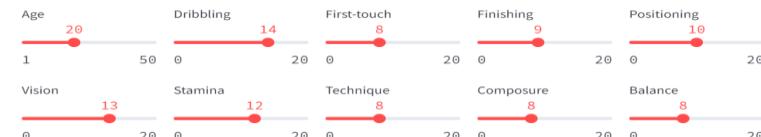
실제 적용 사례

Player Ability Prediction Model

- 1 선수의 전반적인 능력과 잠재력이 궁금하신가요?

Forward Midfielder Defender

선수의 능력치를 입력하세요



선수 능력 및 잠재력 예측하기

Click!

Overall Value

53.95

Potential Value

57.32

▲ Streamlit 구현 예시

03-2. 프로젝트 수행 내용 AI 모델링(2) : ML / 유사선수 분류

2. 포지션별 유사선수 분류, 조회

- 활용 데이터 : FMInside사이트의 Players 속성 Values
- 활용 용도 : 포지션별 주요 능력치를 KNN 모델에 학습
주요 능력치와 유사한 10명의 선수 분류, 조회

구현 코드 방식

```
# K-최근접 이웃 모델 생성 (n_neighbors=10)
knn_DC = KNeighborsClassifier(n_neighbors=10)

# 모델 학습 타겟값 중요하지 않으니 그냥 다 0으로, 인풋값은 포지션과 관련된 속성값
knn_DC.fit(DC, np.zeros(len(DC)))

# X는 포지션별 속성값들을 2차원 배열로 변환한 값
X = np.array(player_DC).reshape(1, -1)
# 찾기
distances, indices = knn_DC.kneighbors(X)

# 선수 이름 마지막 칼럼에 붙이기
UNGK_DC_name = pd.read_csv('UNGK_DC.csv', encoding='utf-16')
UNGK_DC_name.dropna(axis=0, inplace=True)
UNGK_DC_name.reset_index(drop=True, inplace=True)

DC['player_real_name'] = UNGK_DC_name.iloc[:,1]

#print(f'{ss_DC.iloc[row_index, -1]}와 주변 선수들마다의 거리:', distances)
print('주변 선수들마다의 거리:', distances)
# 해당 인덱스에 해당하는 선수들의 이름 출력
print('플레이어와 가장 유사한 선수 10명:')
for i, idx in enumerate(indices[0], 1):
    print(f'{i}. {DC.loc[idx, "player_real_name"]}'')
```

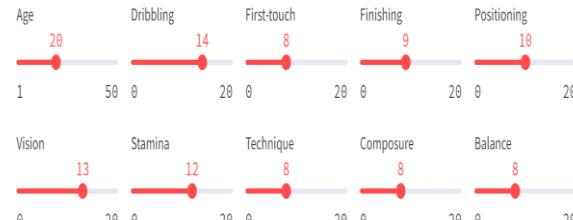
▲ KNN 모델 사용

Player Ability Prediction Model

1 선수의 전반적인 능력과 잠재력이 궁금하신가요?

Forward Midfielder Defender

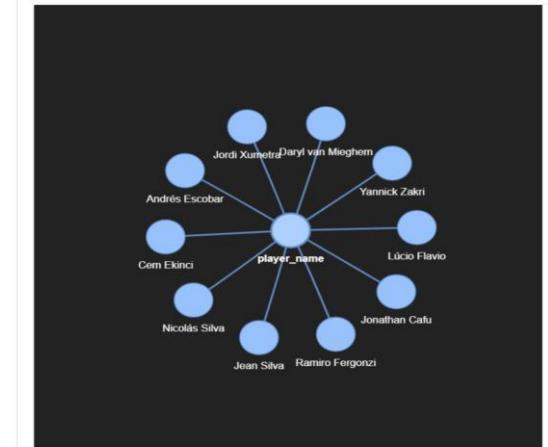
선수의 능력치를 입력하세요



실제 적용 사례

Similar Player Visualizations

2 입력하신 선수와 비슷한 선수는 다음과 같아요!



▲ Streamlit 구현 예시

03-2. 프로젝트 수행 내용

AI 모델링(3) : ML / Market Value 예측

3. 선수 Market Value 예측

- 활용 데이터 : FMInside사이트의 Players 속성 Values, Transform Market의 24년도 유럽 5대 촉구리그 1~5부 선수별 Market Value
- 활용 용도 : 포지션별 능력치, 잠재력 속성을 24년도 선수별 Market Value와 맵핑해 선수별 Market Value 예측 모델 구축

구현 코드 방식

| | player_overall | player_potential | market value |
|-----------------------|----------------|------------------|--------------|
| 1 | 77 | 78.0 | 42000000.0 |
| 2 | 78 | 82.0 | 40000000.0 |
| 4 | 66 | 79.0 | 10000000.0 |
| 5 | 72 | 84.0 | 38000000.0 |
| 6 | 82 | 82.0 | 13000000.0 |
| ... | ... | ... | ... |
| 2172 | 60 | 60.0 | 100000.0 |
| 2173 | 54 | 59.0 | 100000.0 |
| 2174 | 59 | 70.0 | 600000.0 |
| 2175 | 57 | 65.0 | 600000.0 |
| 2176 | 53 | 57.0 | 150000.0 |
| 2108 rows × 3 columns | | | |

▲ Data Merge 결과

```
# pycaret을 이용해서 최적의 모델 구하기
X_train, X_test = train_test_split(Market_value_DC, train_size = 0.7, random_state = 123)
Marketvalue = setup(data = X_train, target = 'market value', test_data = X_test, session_id = 123)
# 모델들 비교
best = compare_models()
# 모델 생성하기
GBR = create_model('gbm', cross_validation = False)
# 모델 튜닝
tuned_GBR = tune_model(GBR, optimize = 'RMSE', n_iter = 200)
# 마지막 학습(Finalize)시키기
# R2 0.7134
final_model = finalize_model(GBR)
```

▲ Gradient Boosting Regressor 모델 사용

실제 적용 사례

Player Ability Prediction Model

1 선수의 전반적인 능력과 잠재력이 궁금하신가요?

Forward Midfielder Defender

선수의 능력치를 입력하세요



Player Market Price Prediction

3 선수의 추정 몸값은?

Player Market Value

520000.0€

▲ Streamlit 구현 예시

03-2. 프로젝트 수행 내용 AI 모델링(4) : DL / 승부 예측

4. 딥러닝 기반승부예측

- 활용 데이터 : FMInside사이트의 Players 속성 Values, FotMob 23~4년도 유럽 5대 축구 리그별 경기 결과 & 출전 선수 라인업
- 활용 용도 : 포지션별 평균 능력치와 23~4년도 경기 결과를 토대로 승부예측 딥러닝 모델 학습

구현코드 방식

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

model = Sequential()
model.add(Dense(8, input_dim = X_train.shape[1], activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.summary()

model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=20)
history = model.fit(X_train, y_train, epochs=30, batch_size = 10,
                     validation_split=0.25, verbose=1, callbacks=[early_stopping_callback])

# 모델 평가
loss, accuracy = model.evaluate(X_test, y_test)

# 평가 결과 출력
print(f"손실(loss): {loss:.4f}")
print(f"정확도(accuracy): {accuracy:.4f}")

9/9 0s 911us/step - accuracy: 0.4026 - loss: 1.1156
손실(loss): 1.1086
정확도(accuracy): 0.4143
```

▲ 딥러닝 모델 사용

실제 적용 사례

2 Home/Away 선수선택 및 승부예측

Home Team

Home 팀 포메이션 : 4-3-3

Away Team

Away 팀 포메이션 : 4-3-3

골키퍼
Thibaut Courtois
91

좌풀백
Luke Shaw
80

좌측 미드필더
Kevin De Bruyne
95

좌측 윙어
Vinicius Junior
92

좌선더백
Virgil van...
89

중앙 미드필더
Rodri
90

중앙 공격수
Erling Haaland
94

우선더백
Ruben Dias
87

우측 미드필더
Bernardo Silva
89

우측 윙어
Lionel Messi
93

우풀백
Jules Kounde
79

좌측 미드필더
Jude Bellingham
89

우측 미드필더
Declan Rice
88

우선더백
Theo Her...
79

중앙 미드필더
John Stones
83

우풀백
Ronald A...
85

우측 미드필더
Achraf H...
81

좌선더백
John Sto...
83

우측 윙어
Joshua Kimmich
87

3 Predict Result

Home Win!!

Input 데이터프레임

| home_gk_mean | home_dc_mean | home_mid_mean | home_st_min | away_gk_mean | away_dc_mean | away_mid_mean | away_st_min |
|--------------|--------------|---------------|-------------|--------------|--------------|---------------|-------------|
| 91 | 83.75 | 91.3333 | 93 | 90 | 82 | 88 | 88.6667 |

Output: 3개 클래스로 분류될 확률

0: 무승부 -> 24.0 %

1: 어웨이팀 승리 -> 32.0 %

2: 홈팀 승리 -> 44.0 %

▲ Streamlit 구현 예시

03-3. 프로젝트 수행 내용

LangChain을 활용한 챗봇 서비스(1) : 선수 및 팀 정보

1. Football Analyze bot

- 활용 데이터 : FMInside사이트의 Players 속성 Values
- 활용 용도 : 데이터정보 요약•시각화 사용 가능
팀 및 개인 선수별 능력치 파악 및 비교 가능

구현 코드 방식

```
if user_input := st.chat_input("분석할 내용을 입력해주세요.")  
    # 사용자가 입력한 내용  
    st.chat_message("user").write(f"{user_input}")  
    # LLM 사용하여 AI 답변 생성  
    # 랭체인  
    # L 랭체인 MODEL 생성  
    llm = ChatOpenAI(model="gpt-3.5-turbo-0125",  
                     temperature = 0,  
                     api_key = myOpenAI_key)  
  
    # 생성한 모델 적용  
    agent_executor = create_pandas_dataframe_agent(  
        llm,  
        all_player,  
        agent_type="openai-tools",  
        verbose=True, # 분석로그  
        return_intermediate_steps=True) # 중간 과정
```

▲ LLM, Zero-shot, Agent 기법 사용

실제 적용 사례

분석할 내용을 입력해주세요.

안녕? 해당 데이터는 무슨 정보를 담고 있어?

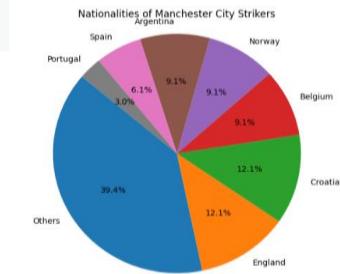
안녕하세요! 이 데이터는 축구 선수들의 다양한 특징과 능력치에 관한 정보를 담고 있습니다. 각 행은 각각의 선수에 대한 정보를 제공하며, 선수의 이름, 팀, 국적, 포지션, 나이, 신체적 특징(키, 몸무게), 그리고 다양한 능력치(페스, 슈팅, 수비 등)가 포함되어 있습니다. 이 데이터를 통해 선수들의 능력을 비교하고 분석할 수 있습니다.

맨체스터시티 팀의 공격수 포지션에 해당하는 선수들에 대한 국가 정보를 파이 차트로 시각화해서 보여줘

Here is a pie chart showing the nationalities of the strikers playing for Manchester City:

- Belgium
- Norway
- England
- Argentina

Each slice of the pie represents the proportion of players from each country.



▲ Streamlit 구현 예시

03-3. 프로젝트 수행 내용

LangChain을 활용한 챗봇 서비스(2) : 경기 결과

2. Match Record bot

- 활용 데이터 : Fotmob사이트의 경기 일정 & 결과 / Soccerdata API (유럽 5대 리그 정보 데이터프레임 형식으로 제공)
- 활용 용도 : 유럽 5대 리그 경기 결과 관련 질의에 대한 답변 가능
(홈/어웨이 팀에 따른 승률 확인 가능)

구현 코드 방식

템플릿 만들기
template = """
당신은 주어진 질문에 친절하게 답변하는 챗봇입니다. 답변 과정에 있어 다음의 요청에 맞춰 답변해주세요.
요청:
1. 주어진 데이터를 기반으로 답변해주세요.
2. team 이름에 대해 한글로 질문해도 영어로 번역하여 해당하는 속성과 그에 대한 답을 찾으세요.
3. 모르는 질문에 대해서는 친절하게 모른다고 답변해주세요.
4. 질문에 대해 정확한 수치로 답변해주세요
5. 답변은 반드시 한글로 해주세요.

CONTEXT: {question}

SUMMARY: """

```
# 프롬프트 생성
prompt = ChatPromptTemplate.from_template(template)
# llm 객체 생성
llm = ChatOpenAI(temperature=0, model_name='gpt-3.5-turbo-0125', api_key = myOpenAI_key)
# chain 생성
chain = ({"question": RunnablePassthrough()} | prompt | llm | StrOutputParser())
```

▲ Data Merge 결과

▲ RAG, Few-shot, Prompt 사용

실제 적용 사례

질문 내용을 입력해주세요.

안녕? 해당 데이터로 무엇을 확인할 수 있는지 알려줘

해당 데이터를 통해 팀 이름, 승리 횟수, 패배 횟수, 득점, 실점, 승률 등을 확인할 수 있습니다. 원하는 정보가 있으면 물어보주세요.

Manchester City팀은 2023년도 경기 중에 홈팀으로 총 몇 번 이겼어?

Manchester City팀은 2023년도 경기 중에 홈팀으로 15번 이겼습니다.

▲ Streamlit 구현 예시

03-3. 프로젝트 수행 내용

LangChain을 활용한 챗봇 서비스(3) : 현지 실시간 뉴스 요약

3. RSS GlobalNews Summary bot

- 활용 데이터 : 유럽 현지의 축구 도메인 사이트 웹크롤링
- 활용 용도 : 영국, 독일, 스페인, 이탈리아 축구 리그에 대해 실시간 기사를 번역 및 요약 가능

구현 코드 방식

```
#chunk 작업 단위로 Template 지정
template = "너는 유럽 축구리그의 모든 정보를 아는 축구 전문가야. 다음 내용을 한국어로 요약해줘.
{text}
"
Kr_arrange_template = "text"
요약 결과는 다음과 같은 형식으로 작성해줘.

(1) 기사제목 : 최대 20자
(2) 기사내용 : 기사 최대 두줄까지 요약한 내용
(3) 작성자 : 기사를 작성한 이름
(4) 작성일자 : 해당 기사가 등록된 년도, 월, 일) ex)yyyy-mm-dd

"
# Prompt 템플릿
sub_prompt = PromptTemplate(template=template, input_variables=["text"])
last_prompt = PromptTemplate(template=Kr_arrange_template, input_variables=["text"])

# llm 모듈 가져오기
llm = ChatOpenAI(temperature = 0,
                 model_name = "gpt-3.5-turbo-0125",
                 api_key = myOpenAI_key)

# 요약정의
chain = load_summarize_chain(
    llm,
    map_prompt=sub_prompt,
    combine_prompt=last_prompt,
    chain_type="map_reduce",
    verbose=False)
```

▲ load_summarize_chain, One-shot, Prompt 사용

실제 적용 사례

- (1) 기사제목: 금요일의 유언: 올리세, 브란스웨이트, 구티에레즈, 귀마레스, 주비엔디 - BBC스포츠
- (2) 기사내용: 맨체스터 유나이티드, 아스널, 맨체스터 시티가 각각 올리세, 브란스웨이트, 귀마레스에 대한 이적 거래를 추진 중이며, 주비엔디는 레알 소시에다에서 행복하다고 밝힘.
- (3) 작성자: BBC 스포츠
- (4) 작성일자: 2024-03-15

- (1) 기사제목: 아르네 슬롯, 리버풀 감독 후임으로 유력한 후보로 떠올라
- (2) 기사내용: 아르네 슬롯이 유르겐 클롭의 후임으로 논의 중이며, 그의 공격적인 스타일과 선수 발전 능력이 매력적으로 평가됨
- (3) 작성자: 베질 반 다이크
- (4) 작성일자: 2023-05-18

▲ Streamlit 구현 예시

중점 및 미흡 사항

중점 사항

- 주 사용자, 즉 고객 입장에서의 서비스/기능 정의 및 구현에 주력
- WBS 상시 확인▶ 과업별 진도파악을 통한 기한내 프로젝트 완성에 주력
- 코드 컨벤션 기반 코딩주력▶ 생산성 향상 및 코드 인계간 빠른 의사결정으로 과업진행
- 데이터 EDA 및 리뷰 강조
- 기능별 구현을 위한 다양한 함수(LangChain, 머신러닝, 딥러닝 내장 함수) 적용 및 리뷰, 선정
- 기능구현 별 코드 최적화를 위한 상시 코드 디버깅 작업

미흡 사항

- 함수 정의 및 호출 기반의 코딩 습관 부족
- 라이브러리 별 버전 충돌에 대한 상황 조치의 숙달 정도 부족

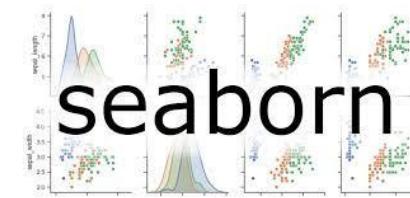
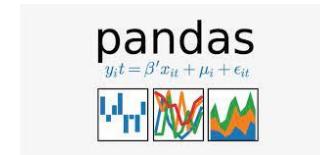
05. 부록

사용 라이브러리 및 버전

버전

Selenium: 4.10.0
Beautifulsoup4: 4.12.3
Chromadb: 0.5.0
Colorama: 0.4.6
Langchain: 0.1.16
langchain-community: 0.0.34
langchain-core: 0.1.45
langchain-experimental: 0.0.57
langchain-openai: 0.1.3
langchain-text-splitters: 0.0.1
Matplotlib: 3.8.4
Numpy: 1.26.4
Openai: 1.23.5
Pandas: 2.2.2
Streamlit: 1.33.0
streamlit-chat: 0.1.1
requests
Sklearn: 0.23.2
Pycaret: 2.3.10

사용 라이브러리



Chroma



05. 부록

코드

[24년도 전세계 축구 선수 데이터 및 Market value, 5대 리그별 경기 기록 크롤링 코드]

```
5 # 셀레늄 설치
6 from selenium import webdriver # Selenium의 웹 드라이버를 사용하기 위한 모듈을 임포트
7 from selenium.webdriver.common.by import By # Selenium에서 사용하는 By 클래스를 임포트합니다. 이 클래스는 웹 요소를 선택하는 데 사용
8 from selenium.webdriver.common.keys import Keys # 키코드. 원하는 키를 제어하기 위한 Keys 클래스를 임포트
9 from selenium.webdriver.chrome.service import Service # Chrome 드라이버 서비스를 사용하기 위한 모듈을 임포트
10 from selenium.webdriver.chrome.options import Options # Chrome 드라이버 옵션을 설정하기 위한 Options 클래스를 임포트
11 from webdriver_manager.chrome import ChromeDriverManager # Chrome 드라이버를 자동으로 설치 및 관리하는 데 사용되는 패키지를 임포트
12 from selenium.webdriver.support.ui import WebDriverWait
13 from selenium.webdriver.support import expected_conditions as EC
14 from selenium.common.exceptions import TimeoutException, ElementClickInterceptedException
15 import time # 시간 걸릴 줄 알기 위해 time 모듈을 임포트
16
17 # 브라우저 설정
18 sel_opt = Options()
19 sel_opt.add_experimental_option("excludeSwitches", ["enable-automation"]) # 드라이버 시작 시 브라우저 표시 여부도 설정
20 sel_opt.add_experimental_option("excludeSwitches", ["enable-logging"]) # 터미널의 브라우저 표시 여부도 설정
21 # mo_option.add_argument("--headless") # 드라이버 실행时不경
22 sel_opt.add_argument("--start-maximized") # 창 크기 최대화
23 sel_opt.add_argument("--disable-gpu") # GPU 캐싱 해제
24 sel_opt.add_experimental_option("detach", True) # 웹크래퍼 방지 플래그 추가
25 sel_opt.add_argument("--incognito") # 시크릿 모드로 진입
26
27 # 드라이버 세팅
28 srt_sevice = Service(ChromeDriverManager().install()) # 드라이버 설정
29 core_driver = webdriver.Chrome(service=srt_sevice, options=sel_opt)
30
31 # 유튜브 정보 수집
32 target_Url = "https://fminside.net/players#google_vignette"
33 core_driver.get(target_Url)
34 time.sleep(10)
35
36 # 무한 루프 시작
37 while True:
38     try:
39         # 'Loadmore' 클래스를 가진 버튼 찾기
40         more_btn = WebDriverWait(core_driver, 10).until(EC.visibility_of_element_located((By.CLASS_NAME, "loadmore")))
41         # 버튼 클릭
42         more_btn.click()
43         # 3초 대기(페이지 토스트가 끝나기 까지)
44         time.sleep(3)
45     except TimeoutException:
46         # 만약에 'Loadmore' 버튼이 없으면 break로 멈춤
47         break
48     except ElementClickInterceptedException:
49         # 클릭할 수 있는 경우, 페이지가 로드될 때까지 대기 후 다시 시도
50         time.sleep(3)
51
52     # 데이터
53 raw_data = list()
54 player_href = core_driver.find_elements(By.TAG_NAME, "a") # 해당페이지의 모든 링크를 리스트로 저장
55 for v in player_href: # 링크를 통해 원하는 외부HTML 코드 추출 및 뷰티풀소스에 추가
56     raw = v.get_attribute("outerHTML")
57     raw_data.append(raw)
58
59 # outerHTML 오스를 split 해서 링크 주소 추출 및 해야 되지만 주소를 통해 유상한 주소로 전처리
60 back_rul = v.split("href=")[1].split(">***>")[0]
61 full_rul = "https://fminside.net"+back_rul
62 print(full_rul)
```

```
1 # 경기 브리거 정보
2 from bs4 import BeautifulSoup
3 import pandas as pd
4 import requests
5 # 파일 경로 설정
6 raw_file = open("./player_link_url.txt", "r")
7 raw_list = raw_file.read().split("\n")
8
9 print(len(raw_list))
10 # 선수별 세부정보 변수
11 real_link = raw_list
12
13 # 데이터 프레임에 사용할 리스트
14 basic_field = ["player_nm", "player_overall", "player_potential", "player_team", "player_country", "player_position", "player_age", "player_height", "player_weight", "player_dribbling", "player_finishing", "player_heading", "player_longshots", "player_aerial_reach", "player_commandofarea", "player_communication", "player_eccentricity", "player_firsttouch", "player_handling", "player_kicking", "player_oneonones"]
15 nonGK_field_m = ["corners", "crossing", "dribbling", "finishing", "first-touch", "free-kick-taking", "heading", "long-shots", "long-throws", "passing", "shooting"]
16 GK_field_m = ["aerial-reach", "command-of-area", "communication", "eccentricity", "first-touch", "handling", "kicking", "one-on-ones"]
17
18 # 선수 정보를 저장할 데이터프레임 초기화
19 Basic_mong_df = pd.DataFrame(columns=basic_field)
20 Basic_gk_df = pd.DataFrame(columns=basic_field)
21 nonGK_df = pd.DataFrame(columns=nonGK_field_m)
22 GK_df = pd.DataFrame(columns=GK_field_m)
23
24 # 데이터프레임 풀어서 만들기
25 now_num = 1
26 for link in real_link[:30]:
27     print(now_num * real_link.index(link))
28     get_info = requests.get(link)
29     all_data = get_info.text
30     myparser = BeautifulSoup(all_data, 'html.parser') # 해당 코드 html로
31
32     # 프로필
33     target_area = myparser.select("div#player_info")
34
35     for lt in target_area:
36         try:
37             player_position = lt.select_one("div#player > div.title > div.meta > ul > li > span.value > span.desktop_positions").text
38             player_nm = lt.select_one("div#player > div.title > h1").text
39             player_overall = lt.select_one("div#player > div.title > div.meta > span#ability").text
40             player_potential = lt.select_one("div#player > div.title > div.meta > span#potential").text
41             player_team = lt.select_one("div#player > div.title > div.meta > ul > li > a > span.value").text
42             player_country = lt.select_one("div#player > div.title > div.meta > ul > li > span.value > a").text
43
44             # 축구 정보 선수 프로필
45             for ld in target_area:
46                 player_column = ld.select("div#player > div.column > ul > li")
47                 for ld_v in player_column:
48                     if ld_v.select_one("span.key").text == "Age":
49                         player_age = ld_v.select_one("span.value").text
50                     elif ld_v.select_one("span.key").text == "Foot":
51                         player_foot = ld_v.select_one("span.value").text
52                     elif ld_v.select_one("span.key").text == "Height":
53                         player_Height = ld_v.select_one("span.value").text.split("*")[-1]
54                     elif ld_v.select_one("span.key").text == "Weight":
55                         player_Weight = ld_v.select_one("span.value").text.split("*")[-1]
56
57             # 우측 프로필
58             stat_columns = myparser.select("div#right_column > div.block.stats > div.column > table > tr > td[class='stat value_']")
```

```
url = "https://www.transfermarkt.com/premier-league/marktwerteverein/wettbewerb/GB1"
driver.get(url)

# 리그에 속하는 팀 리스트 길이 확인
league_all_team = driver.find_elements(By.XPATH, '//*[@id="yw1"]/table/tbody/tr[@class="odd" or @class="even"]')

# 빈 리스트 만들기
team_name = []
team_list = []

# 각 팀에 해당하는 전체 데이터 프레임 크롤링
for i in range(len(league_all_team)):
    # 각 팀에 대한 링크 들어가기
    driver.find_element(By.XPATH, f'//*[@id="yw1"]/table/tbody/tr[{i+1}]/td[3]/a').click()

# 각 팀에 대한 테이블 리스트 길이 구하기
player_table = driver.find_elements(By.XPATH, '//*[@id="yw1"]/table/tbody/tr[@class="odd" or @class="even"]')

# 각 팀에 대한 크롤링(팀 이름 + 데이터)
for i in range(len(player_table)):
    html = driver.page_source
    soup = BeautifulSoup(html, 'html.parser')

    # 팀 이름 크롤링
    name = soup.select('#main > main > div.row.vereinsstartseite > div.large-8.columns > div.box > h2')

    # 데이터 프레임 크롤링
    inform_table = soup.select(f'#{name[0].text} table > tbody > tr:nth-child({i+1})')
    team_name.append(name[0].text)
    team_list.append(inform_table)

driver.back()
time.sleep(1)

# team_list : 이중리스트이기에 단일리스트로 변환
import itertools
team_player_list = list(itertools.chain(*team_list))
team_name_list = list(itertools.chain(*team_name))

# 2개의 리스트 결합
total_player_list = list(zip(team_name_list, team_player_list))
```

05. 부록

코드

[경기기록분석 및 질의응답 Bot 코드]

```
# 데이터 로드
loader = CSVLoader('./data/matchResult_merge.csv')
matchResult = loader.load()

# 텍스트 나누기
text_splitter = RecursiveCharacterTextSplitter(separators = "\n",
                                             chunk_size = 1000, # 1000자씩 split
                                             chunk_overlap=0) # 겹치는 부분 없게

matchPrediction = text_splitter.split_documents(matchResult)

# Chroma 저장
vectorstore = Chroma.from_documents(documents = matchPrediction, embedding=OpenAIEmbeddings())

# 검색 설정
retriever = vectorstore.as_retriever()

# 템플릿 만들기
template = """
다면 과정에 있어 다음의 요청에 따라 답변해주세요.
요청:
1. 주어진 내용을 바탕으로 답변해주세요
2. team 이름에 대해 한글로 질문해도 영어로 번역하여 해당하는 속성과 그에 대한 답을 찾으세요.
3. 해당 데이터 내에서만 답변을 진행하며, 모르는 질문에 대해서는 모른다고 답변하세요.
4. 답변은 반드시 한글로 하주세요.

\n\nCONTEXT: {question}\n\nSUMMARY:"""

# 프롬프트 생성
prompt = ChatPromptTemplate.from_template(template)

# llm 객체 생성
llm = ChatOpenAI(temperature=0, model_name='gpt-3.5-turbo-0125')

# chain 생성
chain = ({"question" : RunnablePassthrough() | prompt | llm | StrOutputParser()}

question = input('질문을 입력 : ')
response = chain.invoke(question)
print(response)
```

[선수데이터 정보분석 Bot 코드]

```
1 #!/usr/bin/python
2 import streamlit as st
3 from streamlit_chat import message
4 import pandas as pd
5 import numpy as np
6 import unicodedata
7 import matplotlib.pyplot as plt
8 import networkx as nx
9 import matplotlib.patches as mpc
10 from unidecode import unidecode as ucd
11 # 주어진 경로에 있는 파일을 읽어온다.
12 # 파일은 경로에 있는 파일을 읽어온다.
13 # 파일은 경로에 있는 파일을 읽어온다.
14 # 파일은 경로에 있는 파일을 읽어온다.
15 # 파일은 경로에 있는 파일을 읽어온다.
16 # 파일은 경로에 있는 파일을 읽어온다.
17 # 파일은 경로에 있는 파일을 읽어온다.
18 # 파일은 경로에 있는 파일을 읽어온다.
19 # 파일은 경로에 있는 파일을 읽어온다.
20 # 파일은 경로에 있는 파일을 읽어온다.
21 # 파일은 경로에 있는 파일을 읽어온다.
22 # 파일은 경로에 있는 파일을 읽어온다.
23 # 파일은 경로에 있는 파일을 읽어온다.
24 # 파일은 경로에 있는 파일을 읽어온다.
25 myOpenAI_key = st.secrets["myOpenAI"]
26 # myOpenAI_key = os.getenv("myOpenAI")
27 # 파일은 경로에 있는 파일을 읽어온다.
28 # 파일은 경로에 있는 파일을 읽어온다.
29 all_player = pd.read_csv("./useData/total_all_position.csv", encoding="utf-16", index_col=0) # 모든 선수를 위한 index_col 사용
30 # 파일은 경로에 있는 파일을 읽어온다.
31 # 파일은 경로에 있는 파일을 읽어온다.
32 # 파일은 경로에 있는 파일을 읽어온다.
33 # 파일은 경로에 있는 파일을 읽어온다.
34 st.set_page_config(layout="wide")
35 # 파일은 경로에 있는 파일을 읽어온다.
36 st.header("Data Analyze with Ovis 🎉")
37 tab_1, tab_2, tab_3 = st.tabs(["Talk with Chat-bot", "Searching and Compare Player", "code"])
38 # 파일은 경로에 있는 파일을 읽어온다.
39 with tab_1:
40     st.subheader("Talk with Ovis")
41     st.dataframe(all_player, use_container_width=True, hide_index=True)
42     if "messages" not in st.session_state:
43         st.session_state["messages"] = []
44     with st.container(border=True):
45         if user_input := st.chat_input("분석할 내용을 입력해주세요."):
46             st.chat_message("user").write(f"

{user_input}

")
47             # LLM 사용 후에 AI로 보내기
48             # 파일은 경로에 있는 파일을 읽어온다.
49             # 파일은 경로에 있는 파일을 읽어온다.
50             llm = ChatOpenAI(model="gpt-3.5-turbo-0125",
51                             temperature = 0,
52                             api_key = myOpenAI_key)
53             # 파일은 경로에 있는 파일을 읽어온다.
54             agent_executor = create_pandas_dataframe_agent(
55                 llm,
56                 all_player,
57                 handle_openai_tools=True,
58                 verbose=True,
59                 return_intermediate_steps=True)
59             # 파일은 경로에 있는 파일을 읽어온다.
60             try:
61                 response = agent_executor.invoke(user_input)
62             except openai.error.InvalidRequestError:
63                 st.write("질문에 대한 답변을 찾을 수 없습니다. 다른 질문을 해주세요.")
64             else:
65                 intermediate_step = response["intermediate_steps"][-1]
66                 if len(response["intermediate_steps"]) == 0: # response['intermediate_steps']에 있는 내용은 빈 배열입니다.
67                     AResponse = response["output"]
68                     st.session_state["messages"].append(ChatMessage(role="user", content=user_input))
69                     st.session_state["messages"].append(ChatMessage(role="assistant", content=AResponse))
70                     with st.chat_message("assistant"):
71                         st.write(AResponse)
72             # 파일은 경로에 있는 파일을 읽어온다.
73             # 파일은 경로에 있는 파일을 읽어온다.
74             # 파일은 경로에 있는 파일을 읽어온다.
75             visual_query = response["intermediate_steps"][-1].tool.input["query"] # response['intermediate_steps']에 있는 내용은 빈 배열입니다.
76             if visual_query:
77                 st.image(visual_query)
78             # 파일은 경로에 있는 파일을 읽어온다.
79             # 파일은 경로에 있는 파일을 읽어온다.
80             # 파일은 경로에 있는 파일을 읽어온다.
81             # 파일은 경로에 있는 파일을 읽어온다.
82             # 파일은 경로에 있는 파일을 읽어온다.
83             # 파일은 경로에 있는 파일을 읽어온다.
84             # 파일은 경로에 있는 파일을 읽어온다.
85             # 파일은 경로에 있는 파일을 읽어온다.
86             # 파일은 경로에 있는 파일을 읽어온다.
87             # 파일은 경로에 있는 파일을 읽어온다.
88             # 파일은 경로에 있는 파일을 읽어온다.
89             # 파일은 경로에 있는 파일을 읽어온다.
90             # 파일은 경로에 있는 파일을 읽어온다.
91             # 파일은 경로에 있는 파일을 읽어온다.
92             # 파일은 경로에 있는 파일을 읽어온다.
93             # 파일은 경로에 있는 파일을 읽어온다.
94             # 파일은 경로에 있는 파일을 읽어온다.
95             # 파일은 경로에 있는 파일을 읽어온다.
96             # 파일은 경로에 있는 파일을 읽어온다.
97             # 파일은 경로에 있는 파일을 읽어온다.
98             # 파일은 경로에 있는 파일을 읽어온다.
99             # 파일은 경로에 있는 파일을 읽어온다.
```

[RSS(Rich Site Summary)] 코드]

```
1 #!/usr/bin/python
2 import streamlit as st
3 from streamlit_chat import message
4 import pandas as pd
5 import numpy as np
6 import unicodedata
7 import matplotlib.pyplot as plt
8 import networkx as nx
9 import matplotlib.patches as mpc
10 from unidecode import unidecode as ucd
11 # 주어진 경로에 있는 파일을 읽어온다.
12 # 파일은 경로에 있는 파일을 읽어온다.
13 # 파일은 경로에 있는 파일을 읽어온다.
14 # 파일은 경로에 있는 파일을 읽어온다.
15 # 파일은 경로에 있는 파일을 읽어온다.
16 # 파일은 경로에 있는 파일을 읽어온다.
17 # 파일은 경로에 있는 파일을 읽어온다.
18 # 파일은 경로에 있는 파일을 읽어온다.
19 # 파일은 경로에 있는 파일을 읽어온다.
20 # 파일은 경로에 있는 파일을 읽어온다.
21 # 파일은 경로에 있는 파일을 읽어온다.
22 # 파일은 경로에 있는 파일을 읽어온다.
23 # 파일은 경로에 있는 파일을 읽어온다.
24 # 파일은 경로에 있는 파일을 읽어온다.
25 myOpenAI_key = st.secrets["myOpenAI"]
26 # myOpenAI_key = os.getenv("myOpenAI")
27 # 파일은 경로에 있는 파일을 읽어온다.
28 # 파일은 경로에 있는 파일을 읽어온다.
29 all_player = pd.read_csv("./useData/total_all_position.csv", encoding="utf-16", index_col=0) # 모든 선수를 위한 index_col 사용
30 # 파일은 경로에 있는 파일을 읽어온다.
31 # 파일은 경로에 있는 파일을 읽어온다.
32 # 파일은 경로에 있는 파일을 읽어온다.
33 # 파일은 경로에 있는 파일을 읽어온다.
34 st.set_page_config(layout="wide")
35 # 파일은 경로에 있는 파일을 읽어온다.
36 st.header("Data Analyze with Ovis 🎉")
37 tab_1, tab_2, tab_3 = st.tabs(["Talk with Chat-bot", "Searching and Compare Player", "code"])
38 # 파일은 경로에 있는 파일을 읽어온다.
39 with tab_1:
40     st.subheader("Talk with Ovis")
41     st.dataframe(all_player, use_container_width=True, hide_index=True)
42     if "messages" not in st.session_state:
43         st.session_state["messages"] = []
44     with st.container(border=True):
45         if user_input := st.chat_input("분석할 내용을 입력해주세요."):
46             st.chat_message("user").write(f"

{user_input}

")
47             # LLM 사용 후에 AI로 보내기
48             # 파일은 경로에 있는 파일을 읽어온다.
49             # 파일은 경로에 있는 파일을 읽어온다.
50             llm = ChatOpenAI(model="gpt-3.5-turbo-0125",
51                             temperature = 0,
52                             api_key = myOpenAI_key)
53             # 파일은 경로에 있는 파일을 읽어온다.
54             agent_executor = create_pandas_dataframe_agent(
55                 llm,
56                 all_player,
57                 handle_openai_tools=True,
58                 verbose=True,
59                 return_intermediate_steps=True)
59             # 파일은 경로에 있는 파일을 읽어온다.
60             try:
61                 response = agent_executor.invoke(user_input)
62             except openai.error.InvalidRequestError:
63                 st.write("질문에 대한 답변을 찾을 수 없습니다. 다른 질문을 해주세요.")
64             else:
65                 intermediate_step = response["intermediate_steps"][-1]
66                 if len(response["intermediate_steps"]) == 0: # response['intermediate_steps']에 있는 내용은 빈 배열입니다.
67                     AResponse = response["output"]
68                     st.session_state["messages"].append(ChatMessage(role="user", content=user_input))
69                     st.session_state["messages"].append(ChatMessage(role="assistant", content=AResponse))
70                     with st.chat_message("assistant"):
71                         st.write(AResponse)
72             # 파일은 경로에 있는 파일을 읽어온다.
73             # 파일은 경로에 있는 파일을 읽어온다.
74             # 파일은 경로에 있는 파일을 읽어온다.
75             visual_query = response["intermediate_steps"][-1].tool.input["query"] # response['intermediate_steps']에 있는 내용은 빈 배열입니다.
76             if visual_query:
77                 st.image(visual_query)
78             # 파일은 경로에 있는 파일을 읽어온다.
79             # 파일은 경로에 있는 파일을 읽어온다.
80             # 파일은 경로에 있는 파일을 읽어온다.
81             # 파일은 경로에 있는 파일을 읽어온다.
82             # 파일은 경로에 있는 파일을 읽어온다.
83             # 파일은 경로에 있는 파일을 읽어온다.
84             # 파일은 경로에 있는 파일을 읽어온다.
85             # 파일은 경로에 있는 파일을 읽어온다.
86             # 파일은 경로에 있는 파일을 읽어온다.
87             # 파일은 경로에 있는 파일을 읽어온다.
88             # 파일은 경로에 있는 파일을 읽어온다.
89             # 파일은 경로에 있는 파일을 읽어온다.
90             # 파일은 경로에 있는 파일을 읽어온다.
91             # 파일은 경로에 있는 파일을 읽어온다.
92             # 파일은 경로에 있는 파일을 읽어온다.
93             # 파일은 경로에 있는 파일을 읽어온다.
94             # 파일은 경로에 있는 파일을 읽어온다.
95             # 파일은 경로에 있는 파일을 읽어온다.
96             # 파일은 경로에 있는 파일을 읽어온다.
97             # 파일은 경로에 있는 파일을 읽어온다.
```

05. 부록 코드

[선수 Market value 예측 코드]

```

# 필요한 라이브러리
import glob
import pandas as pd
import re
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import numpy as np
import os
import warnings
warnings.filterwarnings('ignore')
from pycaret.regression import *
import seaborn as sns
import matplotlib.pyplot as plt

# 데이터 불러오기
import glob
import pandas as pd
import re
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import numpy as np
import os
import warnings
warnings.filterwarnings('ignore')
from pycaret.regression import *
import seaborn as sns
import matplotlib.pyplot as plt

# 파일 경로, 파일 경로의 CSV 파일들을 불러온다.
file_pattern = '*/Total.csv'
csv_files = glob.glob(file_pattern)

# 각 CSV 파일을 읽어들이고 리스트에 저장
dfs = []
for csv_file in csv_files:
    df = pd.read_csv(csv_file, encoding='utf-8')
    dfs.append(df)

# 리스트에 저장된 모든 파일들을 하나로 결합
Transfer_df = pd.concat(dfs, ignore_index=True)

# 결합한 데이터프레임은 CSV 파일로 저장
Transfer_df.to_csv('Transfer_value_data.csv', index=False)
Transfer_df = pd.read_csv('Transfer_value_data.csv', encoding='utf-8')
Transfer_df

# 위치별로 데이터프레임 나누기

# 골키퍼
Transfer_df_GK = Transfer_df[Transfer_df['position']=='Goalkeeper']
Transfer_df_GK.to_csv('Transfer_df_GK.csv', index=False)

# 수비수
Transfer_df['position'].unique()

array(['Goalkeeper', 'Centre-Back', 'Left-Back', 'Right-Back',
       'Defensive Midfield', 'Central Midfield', 'Attacking Midfield',
       'Left Winger', 'Right Winger', 'Second Striker', 'Centre-Forward',
       'Left Midfield', 'Right Midfield', 'Midfielder', 'Striker',
       'Defender'], dtype=object)

# 수비수 블록(나누기)
for i, x in enumerate(Transfer_df['position']):
    if x in ['Centre-Back', 'Left-Back', 'Right-Back', 'Defender']:
        Transfer_df.loc[i, 'P'] = 'DC'
    elif x in ['Defensive Midfield', 'Central Midfield', 'Attacking Midfield', 'Left Winger', 'Right Winger', 'Left Midfield', 'Right Midfield', 'Midfielder']:
        Transfer_df.loc[i, 'P'] = 'MD'
    elif x in ['Second Striker', 'Centre-Forward', 'Striker']:
        Transfer_df.loc[i, 'P'] = 'ST'
    else:
        Transfer_df.loc[i, 'P'] = 'GK'
Transfer_df['position']

Transfer_df_GK = Transfer_df[Transfer_df['position']=='GK']
Transfer_df_GK.to_csv('Transfer_df_GK.csv', index=False)

Transfer_df_DC = Transfer_df[Transfer_df['position']=='DC']
Transfer_df_DC.to_csv('Transfer_df_DC.csv', index=False)

Transfer_df_MID = Transfer_df[Transfer_df['position']=='MD']
Transfer_df_MID.to_csv('Transfer_df_MID.csv', index=False)

Transfer_df_ST = Transfer_df[Transfer_df['position']=='ST']
Transfer_df_ST.to_csv('Transfer_df_ST.csv', index=False)

```

수비수 품값 예측

```

# Transfer_df_all 과 H position을 나눈 데이터를 수비수는 DC의 품값 예측
Market_value_DC = pd.read_csv('Market_value_DC.csv')
Market_value_DC
Market_value_DC = pd.DataFrame(Market_value_DC[['player_overall', 'player_potential', 'market_value']])

Market_value_DC
# 단위로 설정해 주자
f, axes = plt.subplots(1, 1, figsize=(10,10))
sns.scatterplot(x=Market_value_DC['player_overall'], y=Market_value_DC['market_value'])
plt.show()
plt.tight_layout()

# overall 65 이하이고 market value가 10000000이상인 값을 삭제
# overall 75 이상이고 market value가 30000000 이하인 값을 삭제
# overall 100 이상이고 market value가 30000000 이상인 값을 삭제
# overall 10000000 이상이고 market value가 3000000000 이상인 값을 삭제
Market_value_DC.drop(Market_value_DC[(Market_value_DC['player_overall'] <= 65) & (Market_value_DC['market_value'] > 10000000)].index, axis=0, inplace=True)
Market_value_DC.drop(Market_value_DC[(Market_value_DC['player_overall'] >= 75) & (Market_value_DC['market_value'] <= 30000000)].index, axis=0, inplace=True)
Market_value_DC.drop(Market_value_DC[(Market_value_DC['player_overall'] >= 80) & (Market_value_DC['market_value'] >= 30000000)].index, axis=0, inplace=True)
Market_value_DC
# 결측치 확인
Market_value_DC.isna().sum()

# 결측치가 존재하는 market value의 결측값으로 대체함 --- R2
mean_market_value = np.mean(Market_value_DC['market_value'])
Market_value_DC['market_value'].fillna(mean_market_value, inplace=True)
# 결측치 대체 확인
Market_value_DC.isna().sum()

# 결측치 대체 확인
Market_value_DC.dropna(axis=0, inplace=True)
# 결측치 대체 확인
Market_value_DC.isna().sum()
# 결측치 대체 확인
Market_value_DC.info()
Market_value_DC['player_overall'] = Market_value_DC['player_overall'].astype(float)
Market_value_DC.info()
# pycaret을 이용하여 품값을 모델 구하기
X_train, X_test = train_test_split(Market_value_DC, train_size = 0.7, random_state=42)
MarketValue = setup(data=X_train, target = 'market value', test_data = X_test, session_id = 123)
# 모델을 비교
best = compare_models()
# 모델 성능평가
GBM = create_model('gbm', cross_validation = False)
# 모델 훈련
tuned_GBM = tune_model(GBM, optimize = 'RMSE', n_iter = 200)
# 디비전 평가(Finalize) 하기
# R2 0.7134
final_model = finalize_model(tuned_GBM)
# 모델을 비교
best = compare_models()
best
# 모델 훈련
GBM = create_model('gbm', cross_validation = False)
# 모델 훈련
tuned_GBM = tune_model(GBM, optimize = 'RMSE', n_iter = 200)
tuned_GBM
# 디비전 평가(Finalize) 시키기
# R2 0.7334
# finalize_model(): 결측 값을 대체 후 마지막 훈련
# predict_model(): 예측 결과를 'label' 변수에 저장
final_model = finalize_model(tuned_GBM)
pred = predict_model(final_model, data = X_test)
pred.head()
# Final을 Training Score와 Cross Validation Score 사이로
plot_model(final_model)

Transfer_df_DC = pd.read_csv('Transfervalue_df_DC.csv', encoding='utf-8')
Transfer_df_DC
Transfer_df_DC.all = pd.concat([Transfer_df_DC, Transfer_df_MID, Transfer_df_ST], axis=0)
Transfer_df_DC.all.reset_index(inplace=True, drop=True)
Transfer_df_DC.all.to_csv('Transfervalue_df_all.csv', index=False)

```

수비수 team name칼럼으로 맵핑해서 fm데이터 끌어다 붙이기

```

Transfer_df_DC = pd.read_csv('Transfer_df_DC.csv', encoding = 'utf-8')
Transfer_df_DC

leagueNation          league      team name  num   name   age nationality  position  market value
0     England        EPL  Manchester City   3  Rúben Dias  26    Portugal    DC    €80.00
1     England        EPL  Manchester City  24  Josko Gvardiol  22    Croatia    DC    €75.00m
2     England        EPL  Manchester City  25  Manuel Akanji  28    Switzerland    DC    €42.00m
3     England        EPL  Manchester City  29  Nathan Aké  29    Netherlands    DC    €40.00m
4     England        EPL  Manchester City   5  John Stones  29    England    DC    €38.00m
...   ...
4283    Spain Segunda Federación - Grupo I  CD Cayón   3 Álvaro Diez  21    Spain    DC    €25k
4284    Spain Segunda Federación - Grupo I  CD Cayón  15  Riki Fernández  23    Spain    DC    €100k
4285    Spain Segunda Federación - Grupo I  CD Cayón   2  Gabi Fernández  24    Spain    DC    €50k
4286    Spain Segunda Federación - Grupo I  CD Cayón  17 Álvaro Santamaría  19    Spain    DC    €25k
4287    Spain Segunda Federación - Grupo I  CD Cayón  21 Fernando Resines  32    Spain    DC    €25k
4288 rows x 9 columns

```