데이터마이닝 프로젝트 보고서

프리미어리그 선수의 시장 가치 추정 모델

2024. 12. 20

조 번 호	11조
조이름	SIUUUUU
팀원	소프트웨어학부, 권진용, 2021041097 컴퓨터공학과, 박덕열, 2021084046

I. 프로젝트 개요

1 프로젝트 주제

본 프로젝트의 주제는 '프리미어리그 선수의 시장 가치 추정 모델'이다.

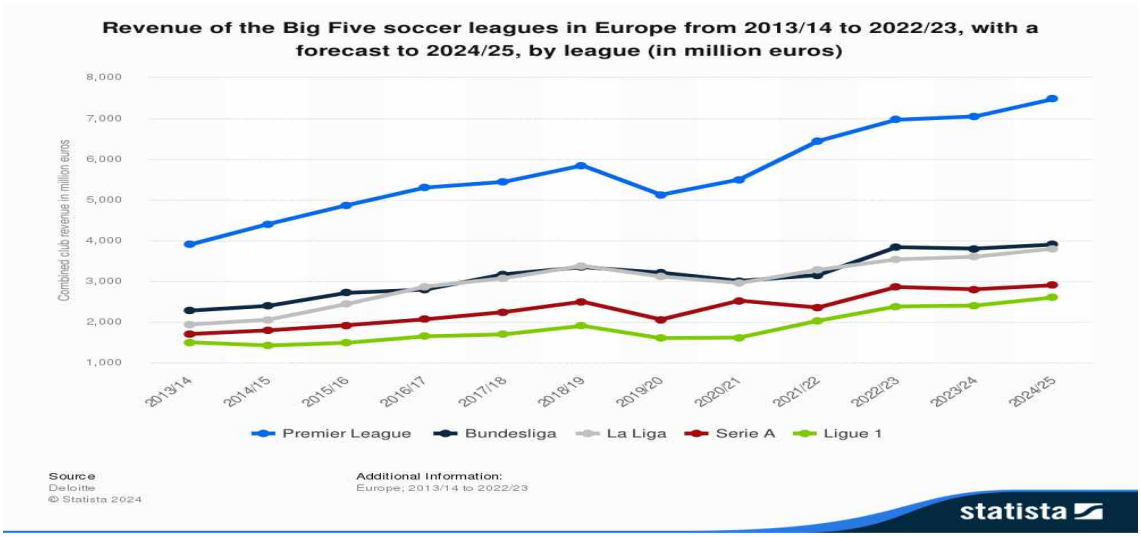
(1) 주제 선정 동기

○ 개인적으로 같은 흥미 분야(축구)를 가지고 있었고 한 학기 동안 실습하고 배운 데이터 마이닝 기법들로 축구 통계 사이트, 게임 데이터 등에서 데이터들을 확보하고 진행할 수 있는 주제라고 판단했기 때문에 ' 프리미어리그 선수의 시장 가치 추정 모델'로 주제를 선정했다.

○

○ 가장 인기 있는 축구 리그

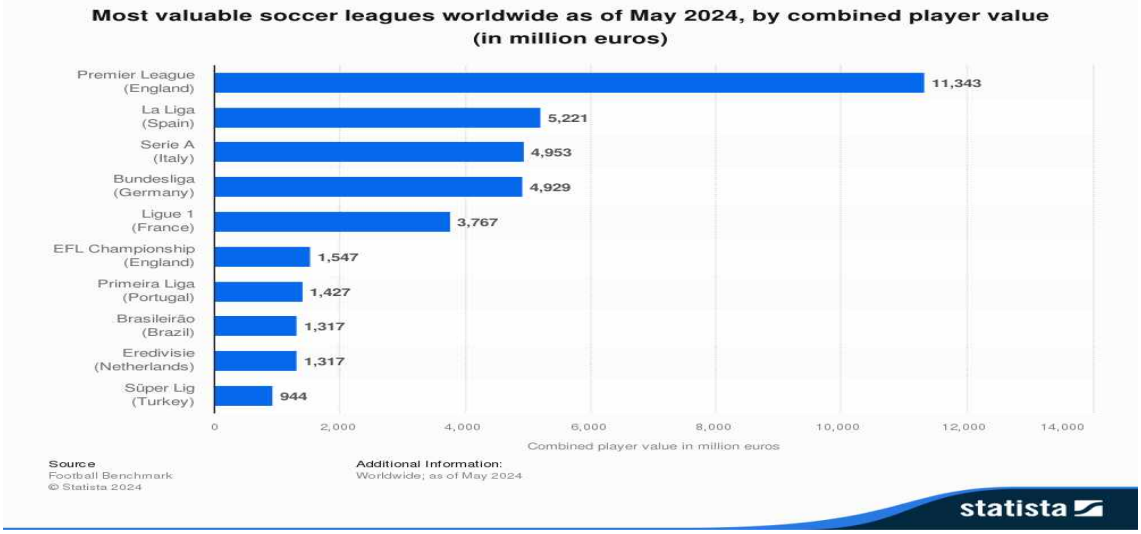
축구는 가장 인기 있는 스포츠 중 하나이다. 그중에서도 프리미어리그는 세계에서 가장 크고 인기 있는 축구 리그 중 하나이다. 아래의 자료를 보면 지난 10여 년간, 리그의 수익이 다른 4대 리그(분데스리가, 라리가, 세리에A, 리그1)에 비해 압도적으로 높다.



출처: https://www.statista.com/statistics/261218/big-five-european-soccer-leagues-revenue/

○ 리그에 따른 시장 가치의 차이

인기와 수익이 가장 큰 프리미어리그의 팀들은 타 리그에 비해 선수의 영입과 시설 운영에 그만큼 많은 자본을 투자할 수 있다. 따라서, 팀에 필요한 선수라면 타 리그와의 영입 경쟁에서 우위에 설 수 있다. 이 과정에서 선수에 대한 시장 가치는 아래와 같이 높아지게 된다. 결과적으로, 프리미어리그와 타 리그의 선수에 대한 가치 평가의 기준은 차이가 있다. 이러한 원인 때문에 프리미어리그만을 선정하여 선수의 시장 가치를 추정하기로 하였다.



출처: https://www.statista.com/statistics/1454070/soccer-leagues-aggregate-player-value/

(2) 주제 소개

○ 선수의 리그 기록으로 시장 가치를 추정

선수의 포지션에 따라 선수의 기록을 입력한다. 포지션에 따라 입력받는 기록의 feature 명과 수는 차이가 있다. 입력한 기록에 대한 프리미어리그 선수의 적절한 시장 가치를 반환한다.

2 프로젝트 목적

현재 프리미어리그의 주급 협상에서 선수의 국적, 리그의 특성(예: 잉글랜드 프리미엄, 홈 그로운 등) 으로 인해 상대적으로 고평가/저평가 받는 선수가 존재한다. 본 프로젝트의 수행을 통해서 팀에서 필요한 포지션 영입, 타 팀과의 영입 경쟁으로 인한 시장 가치 과열, 선수의 남은 계약 기간 등 외부 요인을 배제하고 오직 선수의 기록을 기반으로 한 시장 가치를 제시한다.

3 팀원별 담당 업무 및 기여사항

팀원별 담당 업무 및 기여사항은 다음과 같습니다.

[표 1] 팀원별 담당 업무 및 기여사항

이름	담당 업무	기여사항	비고
권진용	모델 구현	• 분석 모델 구현 및 평가	
박덕열	데이터 수집	• FotMob 데이터 수집 • 전처리 과정 수행	

II. 데이터 수집 및 전처리

1 데이터 수집 방법

본 프로젝트의 수행을 위하여 FotMob 웹페이지에서 프리미어리그 선수들의 기록을 동적 크롤링으로 데이터를 수집하였다. 자세한 데이터의 수집 과정은 아래와 같다.

☐ **collect_data.py**

① get_team_url(result)

FotMob의 프리미어리그 메인 페이지에 프리미어리그 20개 구단의 순위표가 존재한다. 여기서 각 구단의 페이지로 이동할 수 있는 주소(예: https://www.fotmob.com/ko/teams/8650/overview/liverpool)를 얻을 수 있다. 하지만 팀의 스쿼드(감독과 선수)를 확인할 수 있는 주소로 진입을 위해 overview를 squad로 변환하여 result 리스트에 저장했다.

② get_player_url(GK, DF, MF, FW, team_url)

다음으로 구단별 스쿼드 페이지(team_url)에서 각 구단의 선수들의 포지션을 확인하여 각각 GK, DF, MF, FW 리스트에 선수의 주소 파라미터를 저장한 후 반환하였다. 이때 감독은 제외하였고 다음과 같이 저장하였다. (FW 리스트의 손흥민 파라미터: /ko/players/212867/heung-min-son)

선수의 주소 파라미터는 ko/players/player_serial_code/player_name와 같은 형태이다.

③ get_meaningful_data(result, codes, real_codes)

각 선수의 URL 주소(codes)에 진입했을 때 HTML 소스가 대부분 Javascript 이벤트 처리로 구성되어 있었다. 그래서 selenium.webdriver.common.by 라이브러리를 삽입하여 리그의 90분당 기록을 보여주는 버튼을 클래스로 식별했다. 버튼 클릭 이후 이벤트로 나타난 HTML 소스에서 선수의 기록을 가져왔다.

이때 출전 시간이 적은 선수의 경우에는 기록이 비어있는 문제가 있었다. 그래서 선수의 기록이 존재하지 않는 경우, 오류가 발생하는 경우 다음 선수로 넘어가도록 예외 처리를 진행했다. 그리고 기록이 존재하는 선수의 주소 파라미터를 real_code 리스트에 저장했다.

마지막으로 선수의 90분당 기록은 그 선수의 특성을 잘 보여주는 기들로 구성되어 있어서 선수마다 모두 달랐다. 예를 들면, 같은 공격수라도 드리블을 잘하는 선수의 경우에 드리블 성공 횟수와 드리블 성공률을, 공중볼에 강한 선수의 경우에 공중볼 경합 횟수와 공중볼 경합 성공률, 창의적인 패스를 잘하는 선수는 kill pass와 big chance created 등 특성이 모두

달랐다. 그래서 포지션별 선수의 모든 특성을 result 딕셔너리에 저장하여 n명의 선수들에 대해 특성이 몇 번 나타나는지 count 하였다.

④ make_url_csv(GK, DF, MF, FW)

충분한 경기를 소화하여 기록이 존재하는 선수들의 주소 파라미터가 있는 리스트들을 csv 파일로 저장했다.

⑤ make_evaluation_items_csv(GK, DF, MF, FW)

포지션별 특성을 count한 딕셔너리들을 입력받아 모든 선수에 대해 드러나는 특성인지를 검사한다. 기록이 존재하는 모든 선수들에게서 단 한 명에게서도 누락되지 않은 특성으로는 "경고"와 "퇴장"이 있었다. (기록이 존재하는 선수의 수 = 딕셔너리[경고]의 값) 따라서 p 포지션에서 "경고" 특성과 a 특성의 수가 같은 경우, a 특성을 p 포지션의 리스트에 저장하고 이 리스트를 csv 파일로 저장했다.

⑥ main()

위 ① ~ ⑤ 함수를 실행한다.

□ **get_players_data.py**

① save_in_val_list(gk, df, mf, fw)

collect_data.py에서 저장한 csv파일을 포지션별로 gk, df, mf, fw 리스트로 변환한다. 이 함수는 각 포지션별 선수의 주소 파라미터를 저장한 csv 파일과 각 포지션별 모든 선수들이 가진 특성을 저장한 csv 파일에 대해 리스트 변환을 진행하기 때문에 2번 호출한다.

② get_player_info(result, urls, eval)

이 함수는 선수의 90분당 기록을 가져와서 result 함수에 저장한다.먼저, 선수의 주소 파라미터(urls)에서 선수의 주소를 가져와서 collect_data.py의 get_meaningful_data() 함수처럼 동적으로 웹페이지를 실행시켜 temp 리스트에 선수 이름, 키, 나이, 시장 가치, 출전 시간을 저장하고 이어서 각 포지션별 모든 선수들이 가진 특성(eval)을 찾아 해당 특성의 값도 temp 리스트에 저장한다. 그리고 temp 리스트를 result 리스트에 저장한다.

③ main()

①, ② 함수를 실행하고 포지션별 리스트를 csv 파일로 저장한다.

2 데이터 전처리 과정

데이터를 수집하는 과정에서 경기 기록이 없는 선수들, 선수에 따라 존재하지 않는 특성들을 이미 식별하여 제거했다.
하지만 저장된 선수들의 기록에서 키, 나이, 시장 가치의 특성에 대한 값들이 수치가 아니었다. 그리고 일부 데이터의 경우 “1,080”과 같이 수치형 데이터가 아닌 경우가 있어 이에 대해 전처리 과정을 진행했다.
(예: Virgil van Dijk
193cm,33년,€3500만,"1,080",0.08,0.50,0.08,0.02,76.67,91.3%,94.17,0.17,4.33,66.7%,0.08,0.00)

(1) 데이터 타입을 모두 수치 데이터로 통일

□ **data_pre2.ipynb**

① '나이' 열에서 '년', ‘신장’열에서 ‘㎝’, ‘몸값’열에서 ‘€’, ‘만’ 글자를 빈 문자열로 대체 및 앞뒤 공백 제거

② 이후 정수형으로 변환 및 ‘€’, ‘만’ 단위를 실제 몸값 데이터에 적용

III. 분석 모델

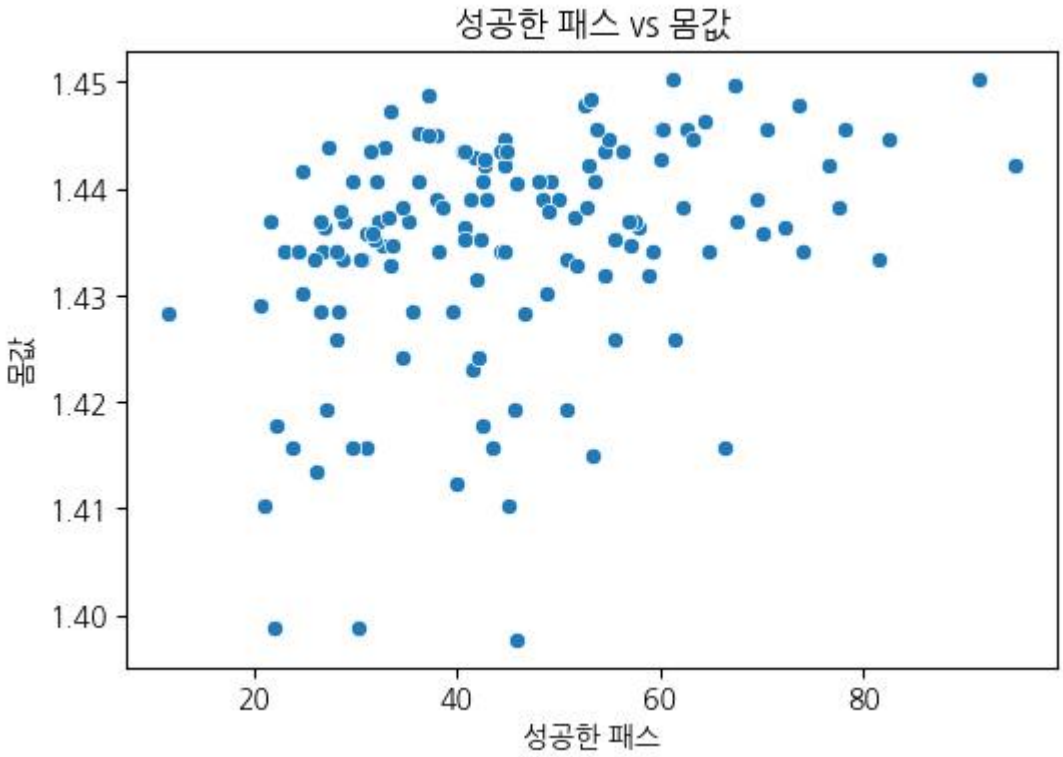
1 사용한 분석 기법

해당 프로젝트의 수행을 위해 랜덤 포레스트 분석을 모델로 선정하였다.

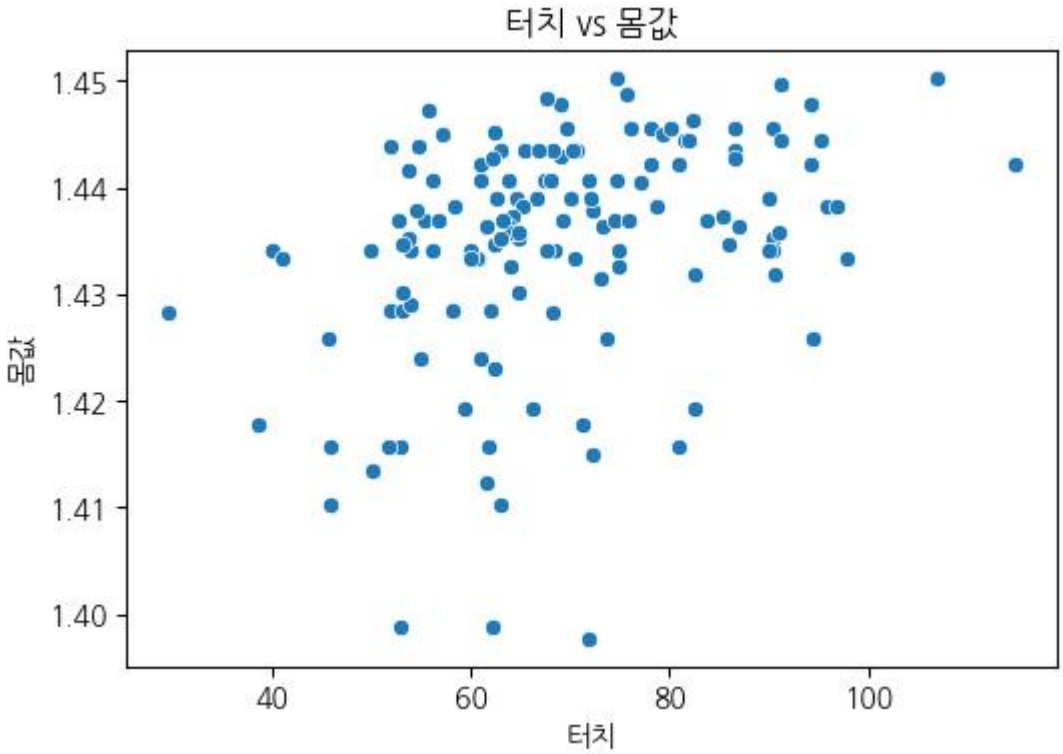
랜덤 포레스트 분석 기법은 여러 개의 의사결정 트리를 결합하여 예측 성능을 높이는 앙상블 학습 방법이다. 해당 분석 기법은 독립 변수와 종속 변수간의 관계가 비선형적이거나 복잡한 경우 그리고 변수의 상호작용이 있는 데이터에 적합하다.

해당 분석 모델을 선택한 이유는 종속 변수(몸값)와 독립 변수(나이, 신장, 골, 경고, 터치 등) 간의 관계가 비선형적 관계 경향을 보이기 때문이었다.

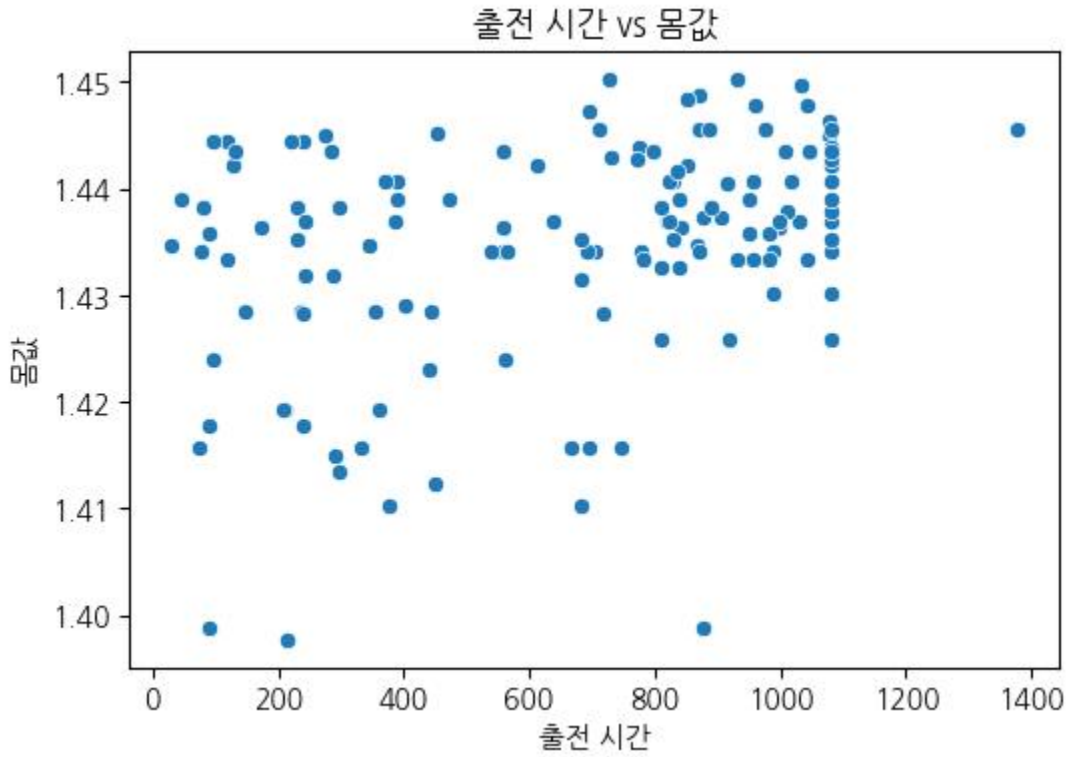
[표 2] 표 목차



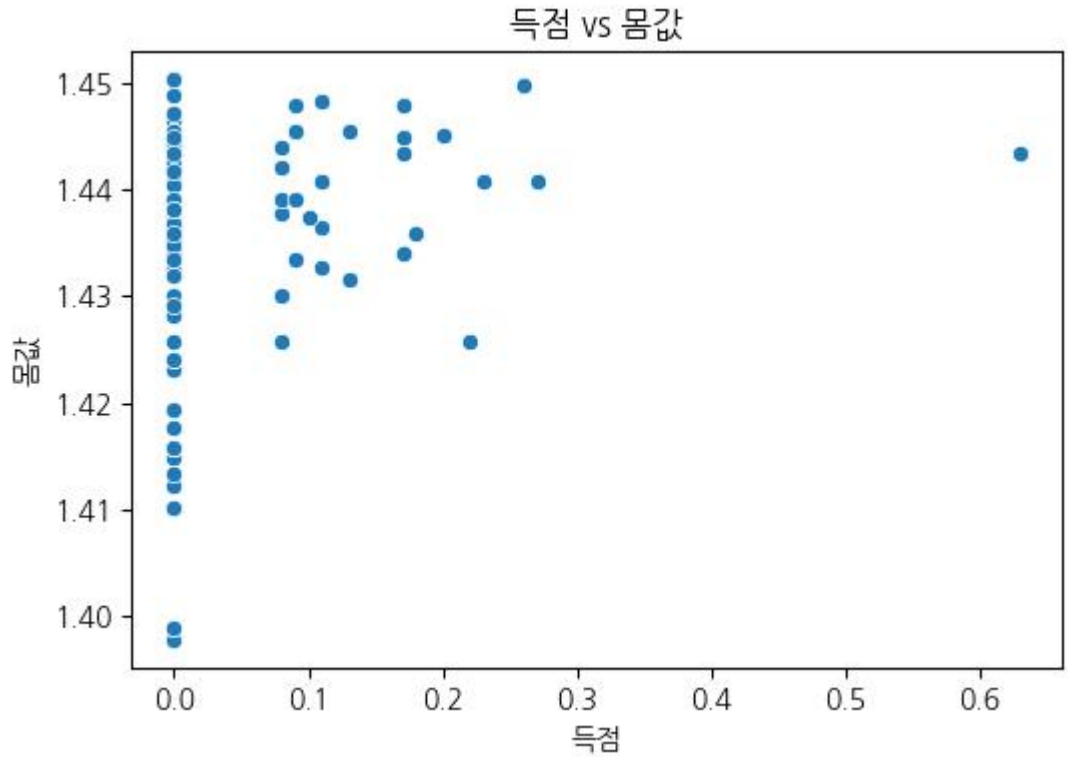
[1] 독립 변수(성공한 패스)와 종속 변수(몸값)의 산점도



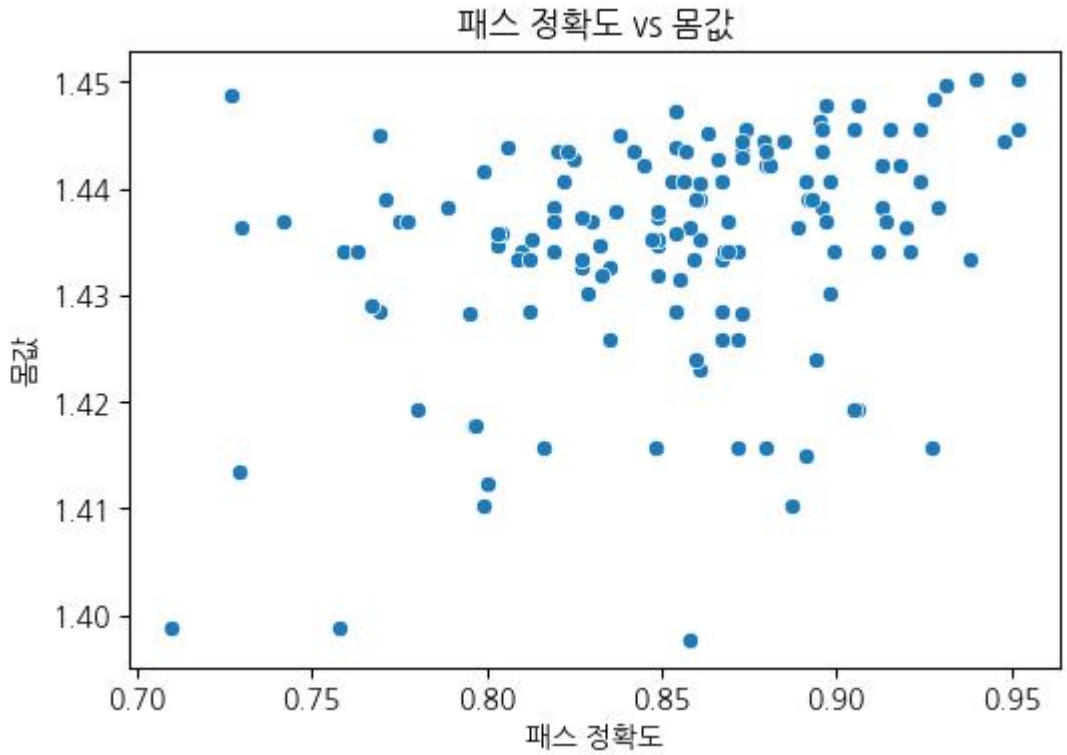
[2] 독립 변수(터치)와 종속 변수(몸값)의 산점도



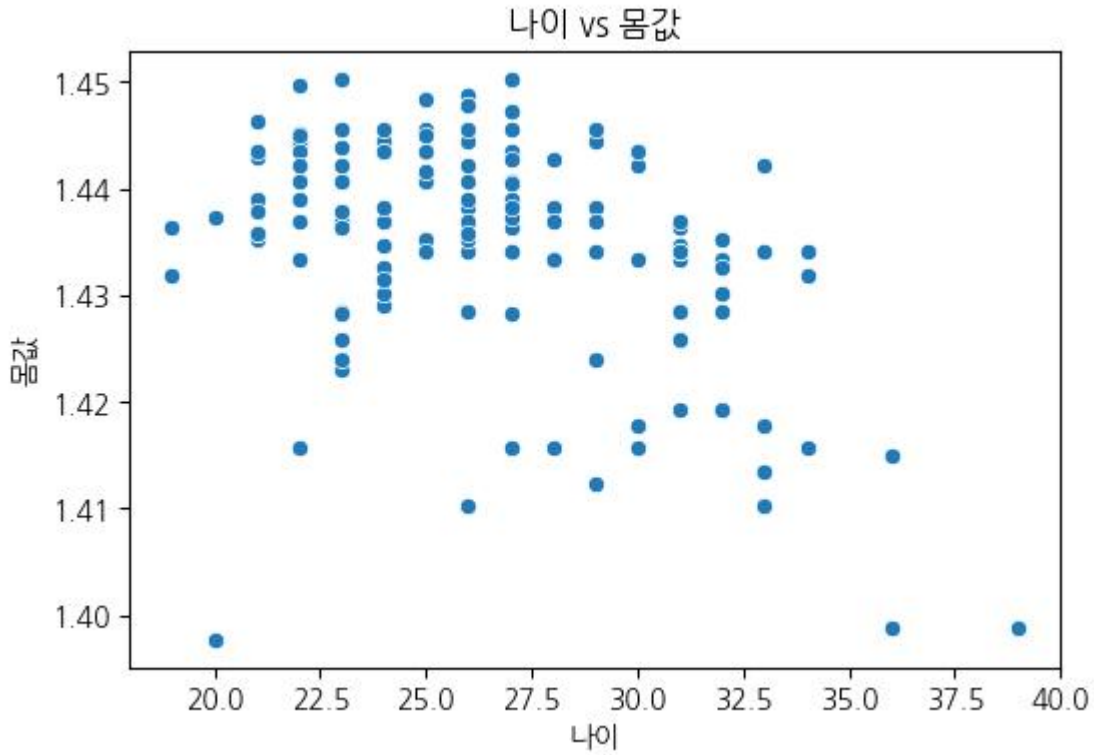
[3] 독립 변수(출전 시간)와 종속 변수(몸값)의 산점도



[4] 독립 변수(득점)와 종속 변수(몸값)의 산점도



[5] 독립 변수(패스 정확도)와 종속 변수(몸값)의 산점도



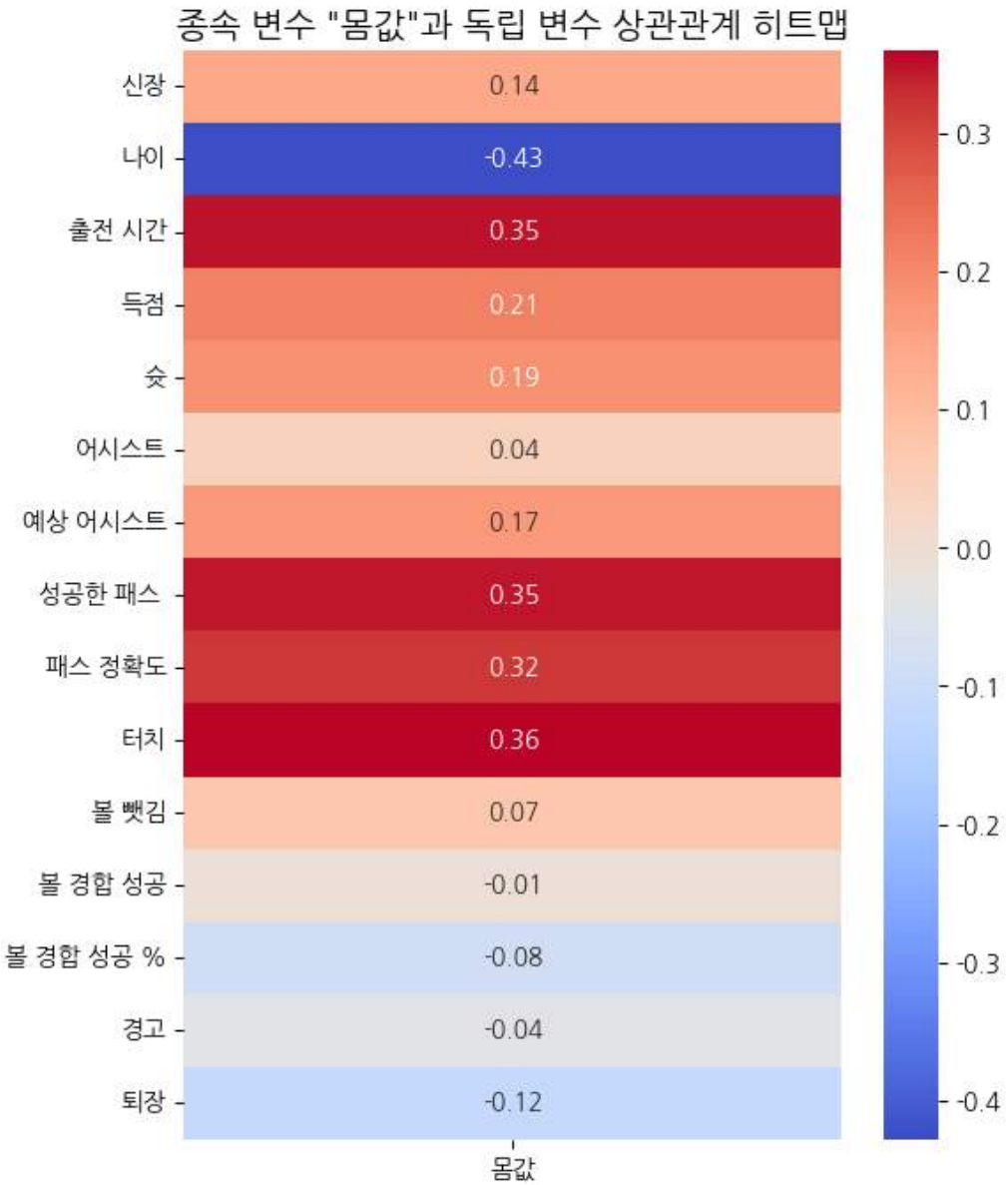
[6] 독립 변수(나이)와 종속 변수(몸값)의 산점도

위의 산점도 결과들을 참고하면, 독립 변수들과 종속 변수(몸값) 간의 산점도에서 완벽한 직선 형태의 선형 관계가 보이지 않았다.

특히, 나이와 몸값의 관계는 **U자형 비선형 패턴**을 보인다. 다른 변수들도 일부 극단값으로 인해 선형 관계가 약하다는 것을 파악할 수 있었다.

또 다른 이유는 상관계수가 약하다.

상관관계 분석에서 독립 변수와 종속 변수 간의 상관관계가 0.3~0.4 수준으로 약하며, 이는 선형적 관계가 강하지 않음을 알 수 있었다.

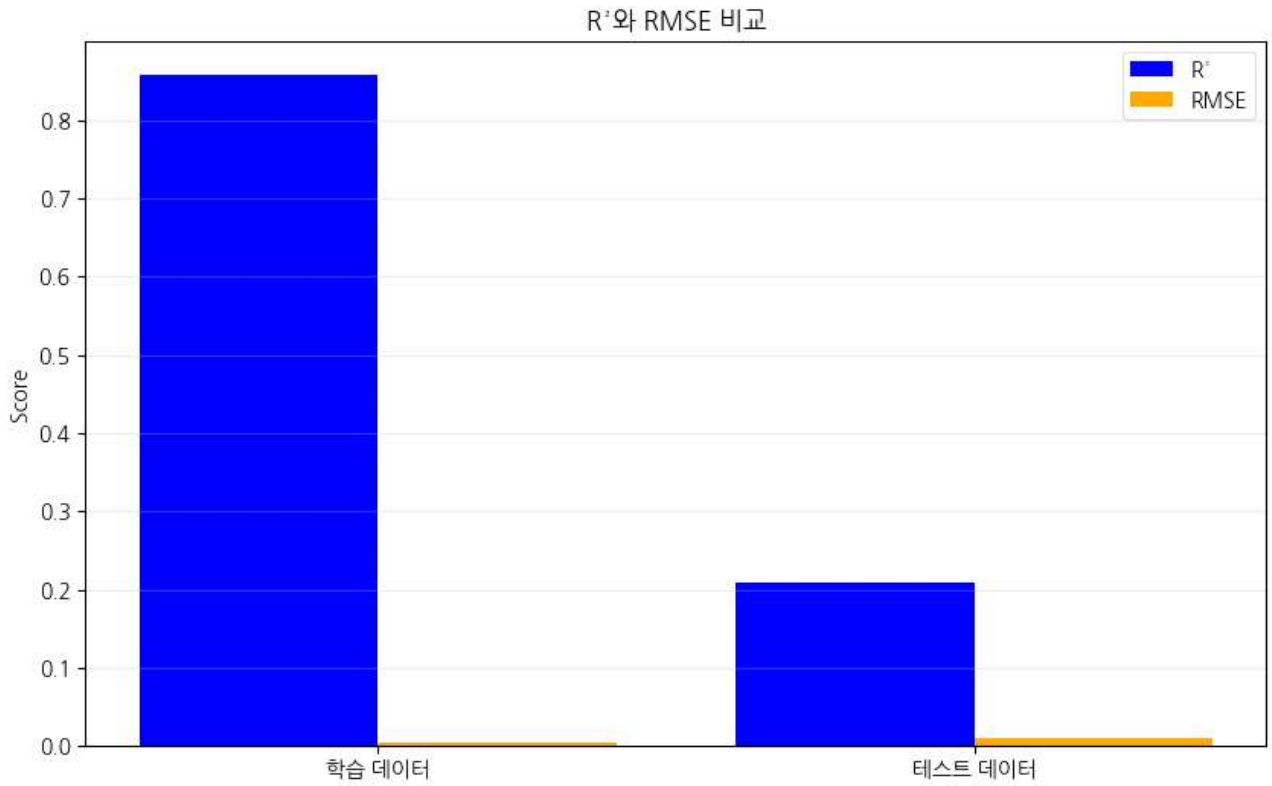


[7] 독립 변수와 종속 변수(몸값) 상관관계 히트맵

IV. 분석 모델 평가 및 분석 결과

1 RMSE, R2 SCORE

랜덤 포레스트 모델을 선택하고 학습한 뒤 도출해낸 r2 score 와 rmse 지표이다.



[8] 랜덤 포레스트 모델의 R2_SCORE, RMSE 막대그래프

R^2 = 0.857, RMSE = 0.004

학습 데이터에서 모델이 매우 높은 성능을 보이고 있음을 알 수 있다.

그러나, 테스트 데이터에서 RMSE = 0.010 는 평균 오차가 0.010으로 학습 데이터에 비해 약 2.5배 높은 수준이지만 R^2 = 0.209를 달성했다. 이는 학습 데이터와 비교하여 설명력이 크게 떨어졌음을 의미한다. 모델이 테스트 데이터에서 일반화 성능이 부족하다는 것을 나타낸다.

V. 결론

1 결론

학습 데이터에서 높은 성능을 보여주지만, 반면 테스트 데이터에서는 성능이 크게 저하되었다. 이는 모델이 학습 데이터의 세부적인 패턴에 과도하게 맞추어져 새로운 데이터(테스트 데이터)를 잘 일반화하지 못했음을 알 수 있다.

나이, 출전 시간, 득점, 터치, 패스 정확도 성공한 패스들 등의 독립변수 상관관계 값이 0.5(절댓값)보다 큰 값이 없다. 언급되지 않은 다른 독립 변수들의 상관 관계값은 0에 가깝거나 준하기 때문에 제외하였다.

이에 결론은 데이터의 한계가 있다고 판단했다. Fotmob 사이트를 기반으로 크롤링한 데이터들만으론 선수들의 기록을 학습하고 새로운 데이터에 대한 결과값 (예측되는 몸값)을 도출하기엔 부족하다. 몸값에 영향을 미치는 외부 요인(예: 마케팅 가치 등) 및 추가 데이터가 확보되면 모델 성능을 개선할 수 있다고 판단된다.

[별첨] 소스코드

Github 주소: https://github.com/jinyong1837/Data_Mining

collect_data.py(1)

```
from bs4 import BeautifulSoup
import urllib.request
import pandas as pd
import datetime
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
import time
import csv

mainUrl = "https://www.fotmob.com"
wd = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

def get_team_url(result):
    wd.get(mainUrl + "/ko/leagues/47/overview/premier-league")
    time.sleep(1)

    teams = wd.find_elements(By.CLASS_NAME, 'css-jlnv70-TeamLink.exhos731')
    for team in teams:
        team_url = team.get_attribute('href')
        team_url = team_url.replace('/overview', '/squad')
        result.append(team_url)
    return


def get_player_url(GK, DF, MF, FW, team_url):
    for team in team_url:
        html = urllib.request.urlopen(team)
        soupTeam = BeautifulSoup(html, 'html.parser')
        playerPosition = soupTeam.select("div.css-10a1gry-SquadTilesWrapper.e1kl3u1z2")

        for i in range(1, 5):
            playerCodes = playerPosition[i].select("a")
            if i == 1:
                for code in playerCodes:
                    GK.append(code['href'])
            elif i == 2:
                for code in playerCodes:
                    DF.append(code['href'])
            elif i == 3:
                for code in playerCodes:
                    MF.append(code['href'])
            elif i == 4:
                for code in playerCodes:
                    FW.append(code['href'])
    return
```

collect_data.py(2)

```
def make_url_csv(GK, DF, MF, FW):
    with open('url/gk_url.csv', mode='w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        for code in GK:
            writer.writerow([code])
    with open('url/df_url.csv', mode='w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        for code in DF:
            writer.writerow([code])
    with open('url/mf_url.csv', mode='w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        for code in MF:
            writer.writerow([code])
    with open('url/fw_url.csv', mode='w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        for code in FW:
            writer.writerow([code])
    return


def get_meaningful_data(result, codes, real_codes):
    for code in codes:
        wd.get(mainUrl + code)
        time.sleep(1)

        try:
            # 버튼 클릭 대기
            button = wd.find_element(By.CLASS_NAME, 'css-a7bq51-FilterButton.e1tb2kvp1')
            button.click()

            # 페이지 소스 가져오기
            soupPlayer = BeautifulSoup(wd.page_source, 'html.parser')
            stats = soupPlayer.select("div.css-17js6f6-PlayerPageGridCSS.e17ysukt0 > div.css-14y4cbw-Column-LeftColumnCSS.e17ysukt1
> div.css-1wb2t24-CardCSS.e1mlfzv61")
            stats = stats[3]
            stats = stats.select("div.css-2duihq-StatTitle.e1uibvo11")

            for stat in stats:
                if stat.string in result:
                    result[stat.string] += 1
                else:
                    result[stat.string] = 1

            if len(stats) != 0:
                real_codes.append(code)

        except Exception as e:
            pass
    return
```

collect_data.py(3)

```
def make_evaluation_items_csv(GK, DF, MF, FW):
    gktemp = []; dftemp = []; mftemp = []; fwtemp = []
    for key, value in GK.items():
        if value == GK['경고']:
            gktemp.append(key)
    for key, value in DF.items():
        if value == DF['경고']:
            dftemp.append(key)
    for key, value in MF.items():
        if value == MF['경고']:
            mftemp.append(key)
    for key, value in FW.items():
        if value == FW['경고']:
            fwtemp.append(key)
    with open('evaluation items/gk.csv', mode='w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        for item in gktemp:
            writer.writerow([item])
    with open('evaluation items/df.csv', mode='w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        for item in dftemp:
            writer.writerow([item])
    with open('evaluation items/mf.csv', mode='w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        for item in mftemp:
            writer.writerow([item])
    with open('evaluation items/fw.csv', mode='w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        for item in fwtemp:
            writer.writerow([item])
    return


def main():
    teamCode = []
    TempGKCodes = []; TempDFCodes= []; TempMFCodes = []; TempFWCodes = []
    GKCodes = []; DFCodes= []; MFCodes = []; FWCodes = []
    playerStats = []
    GKStatDic = {}; DFStatDic = {}; MFStatDic = {}; FWStatDic = {}

    get_team_url(teamCode)
    get_player_url(TempGKCodes, TempDFCodes, TempMFCodes, TempFWCodes, teamCode)

    get_meaningful_data(GKStatDic, TempGKCodes, GKCodes)
    get_meaningful_data(DFStatDic, TempDFCodes, DFCodes)
    get_meaningful_data(MFStatDic, TempMFCodes, MFCodes)
    get_meaningful_data(FWStatDic, TempFWCodes, FWCodes)

    make_url_csv(GKCodes, DFCodes, MFCodes, FWCodes)
    make_evaluation_items_csv(GKStatDic, DFStatDic, MFStatDic, FWStatDic)

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

get_players_data.py(1)

```
from bs4 import BeautifulSoup
import urllib.request
import pandas as pd
import datetime
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
import time
import csv

mainUrl = "https://www.fotmob.com"
wd = webdriver.Chrome(service=Service(ChromeDriverManager().install()))


def save_in_val_list(gk, df, mf, fw):
    with open('evaluation items/gk.csv', newline='', encoding='utf-8') as csvfile:
        reader = csv.reader(csvfile)
        for row in reader:
            gk.append(row[0])
    with open('evaluation items/df.csv', newline='', encoding='utf-8') as csvfile:
        reader = csv.reader(csvfile)
        for row in reader:
            df.append(row[0])
    with open('evaluation items/mf.csv', newline='', encoding='utf-8') as csvfile:
        reader = csv.reader(csvfile)
        for row in reader:
            mf.append(row[0])
    with open('evaluation items/fw.csv', newline='', encoding='utf-8') as csvfile:
        reader = csv.reader(csvfile)
        for row in reader:
            fw.append(row[0])
    return


def save_in_url_list(gk, df, mf, fw):
    with open('url/gk_url.csv', newline='', encoding='utf-8') as csvfile:
        reader = csv.reader(csvfile)
        for row in reader:
            gk.append(row[0])
    with open('url/df_url.csv', newline='', encoding='utf-8') as csvfile:
        reader = csv.reader(csvfile)
        for row in reader:
            df.append(row[0])
    with open('url/mf_url.csv', newline='', encoding='utf-8') as csvfile:
        reader = csv.reader(csvfile)
        for row in reader:
            mf.append(row[0])
    with open('url/fw_url.csv', newline='', encoding='utf-8') as csvfile:
        reader = csv.reader(csvfile)
        for row in reader:
            fw.append(row[0])
    return
```

get_players_data.py(2)

```
def get_player_info(result, urls, eval):
    for url in urls:
        wd.get(mainUrl + url)

        try:
            # 버튼 클릭 대기
            button = wd.find_element(By.CLASS_NAME, 'css-a7bq51-FilterButton.e1tb2kvp1')
            button.click()

            # 페이지 소스 가져오기
            soupPlayer = BeautifulSoup(wd.page_source, 'html.parser')
            source = soupPlayer.select("div.css-17js6f6-PlayerPageGridCSS.e17ysukt0 > div.css-14y4cbw-Column-LeftColumnCSS.e17ysukt1 > div.css-1wb2t24-CardCSS.e1mlfzv61")

            # 선수 정보를 담을 리스트
            temp = []

            # 선수 이름, 키, 나이, 시장 가치 가져오기
            info = source[0]
            name = info.select("div.css-1l2h5po-NameAndTeam.e1uunyvp4 > h1.css-zt63wq-PlayerNameCSS.e1uunyvp1")
            temp.append(name[0].string)
            height_and_age = info.select("div.css-to3w1c-StatValueCSS.e55tcbm4")
            height = height_and_age[0].select("span")
            temp.append(height[0].string)
            age = height_and_age[2].select("span")
            temp.append(age[0].string)
            value = height_and_age[5].select("span")
            temp.append(value[0].string)

            # 선수 출전 시간 가져오기
            time = source[1]
            time = time.select("div.css-170fd60-StatValue.e1ahduwc5 > span")
            temp.append(time[4].string)

            # 선수 기록 가져오기
            stats = source[3]
            records = stats.select("div.css-17zw5kc-StatCSS.e1uibvo13 > div.css-jb6lgd-StatValue.e1uibvo12 > span")
            stats = stats.select("div.css-2duihq-StatTitle.e1uibvo11")

            for stat, record in zip(stats, records):
                if stat.string in eval:
                    temp.append(record.string)
                else:
                    pass

            result.append(temp)

        except Exception as e:
            pass
    return
```

get_players_data.py(3)

```
def main():
    gkeval = []; dfeval = []; mfeval = []; fweval = []
    gkurl = []; dfurl = []; mfurl = []; fwurl = []
    gkstats = []; dfstats = []; mfstats = []; fwstats = []

    save_in_val_list(gkeval, dfeval, mfeval, fweval)
    save_in_url_list(gkurl, dfurl, mfurl, fwurl)

    get_player_info(gkstats, gkurl, gkeval)
    get_player_info(dfstats, dfurl, dfeval)
    get_player_info(mfstats, mfurl, mfeval)
    get_player_info(fwstats, fwurl, fweval)

    with open('players info/gk.csv', mode='w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        for item in gkstats:
            writer.writerow(item)
    with open('players info/df.csv', mode='w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        for item in dfstats:
            writer.writerow(item)
    with open('players info/mf.csv', mode='w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        for item in mfstats:
            writer.writerow(item)
    with open('players info/fw.csv', mode='w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file)
        for item in fwstats:
            writer.writerow(item)

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

data_pre2.ipynb

```
import pandas as pd
import numpy as np

# 데이터셋 불러오기
df = pd.read_excel('/content/drive/MyDrive/Colab Notebooks/datamining/df.xlsx')

print("데이터셋 크기 :",df.shape)
df.info()

# 추가 데이터 전처리

# '나이' 열에서 '년'이라는 글자를 빈 문자열로 대체
df['나이'] = df['나이'].astype(str)
df['나이'] = df['나이'].str.replace('년', '', regex=False)
df['나이'] = df['나이'].str.strip()  # 앞뒤 공백 제거

df['신장']=df['신장'].astype(str)
df['신장']=df['신장'].str.replace('cm', '', regex=False)
df['신장']=df['신장'].str.strip()  # 앞뒤 공백 제거

df['몸값'] = df['몸값'].str.replace('만', '', regex=False)
df['몸값'] = df['몸값'].str.replace('€', '', regex=False)
df['몸값'] = df['몸값'].str.strip()  # 앞뒤 공백 제거

# 이후 정수형으로 변환
df['나이'] = df['나이'].astype(int)
df['신장'] = df['신장'].astype(int)
df['몸값'] = df['몸값'].astype(int) * 10000000

# 추가 전처리한 데이터셋을 해당 경로로 설정하여 저장
df.to_excel('/content/drive/MyDrive/Colab Notebooks/datamining/processed_data.xlsx', index=False)
```

data_model_team.ipynb (1)

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split

df = pd.read_excel('/content/drive/MyDrive/Colab Notebooks/datamining/processed_data.xlsx')

# 1. 독립 변수와 종속 변수 설정

# 열 이름에서 공백 제거
df.columns = df.columns.str.strip()

X = df[['신장', '나이', '출전 시간', '득점', '슛', '어시스트', '예상 어시스트',
    '성공한 패스', '패스 정확도', '터치', '볼 뺏김', '볼 경합 성공', '경고', '퇴장']]

df['몸값'] = np.log1p(df['몸값']) # 몸값의 데이터값의 범위가 너무 크기 때문에 log 변환을 사용한다.
y = df['몸값']

# 종속 변수 로그 변환
y_log = np.log1p(y)  # log(1 + y) 변환

# 2. 데이터 분할
X_train, X_test, y_train_log, y_test_log = train_test_split(X, y_log, test_size=0.4, random_state=42)

# 3. 랜덤 포레스트 모델 생성 및 학습
rf_model = RandomForestRegressor(n_estimators= 200 # 트리 수 증가
                                 , max_depth = 10 # 트리 깊이 제한
                                 , min_samples_split = 5
                                 , random_state = 42)
rf_model.fit(X_train, y_train_log)

# 4 예측 및 지수 변환(로그 복원)
y_train_pred_log = rf_model.predict(X_train)
y_test_pred_log = rf_model.predict(X_test)
y_train_pred = np.expm1(y_train_pred_log)  # 예측값 복원
y_test_pred = np.expm1(y_test_pred_log)
```

data_model_team.ipynb (2)

```
# 5. R^2 점수와 RMSE 계산
y_train_actual = np.expm1(y_train_log)  # 실제값 복원
y_test_actual = np.expm1(y_test_log)    # 실제값 복원

train_r2 = r2_score(y_train_actual, y_train_pred)
test_r2 = r2_score(y_test_actual, y_test_pred)

train_mse = mean_squared_error(y_train_actual, y_train_pred)
test_mse = mean_squared_error(y_test_actual, y_test_pred)

train_rmse = np.sqrt(train_mse)
test_rmse = np.sqrt(test_mse)

print(f"학습 데이터 R²: {train_r2:.3f}, RMSE: {train_rmse:.3f}")
print(f"테스트 데이터 R²: {test_r2:.3f}, RMSE: {test_rmse:.3f}")

# 산점도, 히트맵 시각화
!pip install koreanize-matplotlib

import matplotlib.pyplot as plt
import seaborn as sns
import koreanize_matplotlib
%matplotlib inline

# '이름' 열 제거
try:
    df_no_name = df.drop(columns=['이름'])
except KeyError:
    df_no_name = df

# '몸값'과 다른 변수 간 상관관계 계산
target_corr = df_no_name.corr()['몸값'].drop('몸값')

# 히트맵 시각화
plt.figure(figsize=(6, 8))
sns.heatmap(target_corr.to_frame(), annot=True, cmap='coolwarm', fmt=".2f", cbar=True)
plt.title('종속 변수 "몸값"과 독립 변수 상관관계 히트맵', fontsize=14)
plt.xticks(rotation=0, fontsize=10)
plt.yticks(fontsize=10)
plt.show()
```

data_model_team.ipynb (3)

```
for feature in ['성공한 패스', '터치', '패스 정확도', '출전 시간', '득점', '나이']:
    plt.figure(figsize=(6, 4))
    sns.scatterplot(x=df[feature], y=df['몸값'])
    plt.title(f'{feature} vs 몸값')
    plt.xlabel(feature)
    plt.ylabel('몸값')
    plt.show()

# 데이터
datasets = ['학습 데이터', '테스트 데이터']
r2_scores = [0.857, 0.209]
rmse_scores = [0.004, 0.010]

x = np.arange(len(datasets))

# 막대 그래프
plt.figure(figsize=(10, 6))

# R²
plt.bar(x - 0.2, r2_scores, width=0.4, label='R²', color='blue')
# RMSE
plt.bar(x + 0.2, rmse_scores, width=0.4, label='RMSE', color='orange')

plt.xticks(x, datasets)
plt.ylabel("Score")
plt.title("R²와 RMSE 비교")
plt.legend()
plt.grid(axis='y', alpha=0.3)
plt.show()
```

참고문헌

[1] 김지희, 오진희, 김명진, 임양규.(2021). 인공지능 감정분석 기술을 이용한 관객 참여형 공연에서의 실감형 콘텐츠 생성 방식에 관한 연구. 방송공학회논문지, 26(5), 533-542.
[2] 최종후, & 서두성. (1999). 데이터마이닝 의사결정나무의 응용. 통계분석연구, 4(1), 61-83.
[3] 김진화, & 민진영. (2004). 연속발생 데이터를 위한 실시간 데이터 마이닝 기법. 한국경영과학회지, 29(4), 41-60.