FC온라인 승리 요인 분석과 유저 게임 /과금 전략 제안

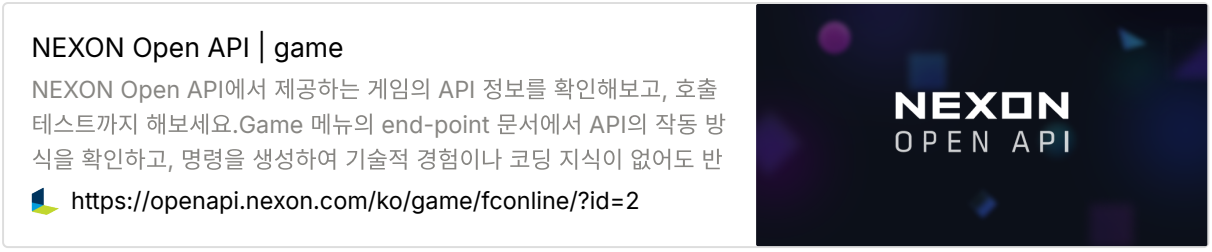
1. 프로젝트 소개 및 주제 선정 이유

1.1 프로젝트 소개 및 주제 선정

FC온라인 축구게임은 동아시아에서 5천3백만명이 넘는 누적 가입자수를 보유하고 있으며, 56억 이상의 플레이 시간을 자랑하는 아시아권의 대표 인기 축구 시뮬레이션 게임입니다. 이 게임은 실제 축구 리그와 선수들을 기반으로 한 현실감 넘치는 게임플레이를 제공하며 전세계적으로 사랑받고 있습니다. 저희 팀은 넥슨에서 제공한 OPEN API를 활용해 유저들의 다양한 경기 상황과 플레이어 행동 데이터를 분석하여 승리 요인을 다각도로 파악하였습니다. 또한, 승리 예측 모델을 개발하였습니다. 저희의 분석 결과를 토대로 FC온라인 유저들의 게임 플레이 시 승률 상승을 위한 가이드라인을 제작, 이를 통해 FC온라인의 유저 경험을 향상시키고자 하였습니다. 더불어, 기업 입장에서 유저의 과금을 유도할 수 있는 액션 아이템을 제안해 게임 과금 전략을 제시하고자 합니다.

2. 데이터 ETL 및 전처리

2.1 데이터 수집 절차 (NEXON OPEN API, FC 온라인 홈페이지 크롤링)



데이터 수집은 넥슨의 OPEN API와 FC 온라인 홈페이지 크롤링을 활용하였습니다. 넥슨 OPEN API에서 게임 당 3개의 개발용 API키를 발급 받을 수 있으며, 1개의 API 키당 1000개의 데이터를 추출할 수 있습니다. 웹 크롤링의 경우, API를 통해 얻어진 nickname을 활용하여 전적 검색 창에 입력시키고, 입력된 새로운 창에서 BP(구단 가치) 데이터를 얻어오는 형식으로 크롤링을 시도하였습니다.

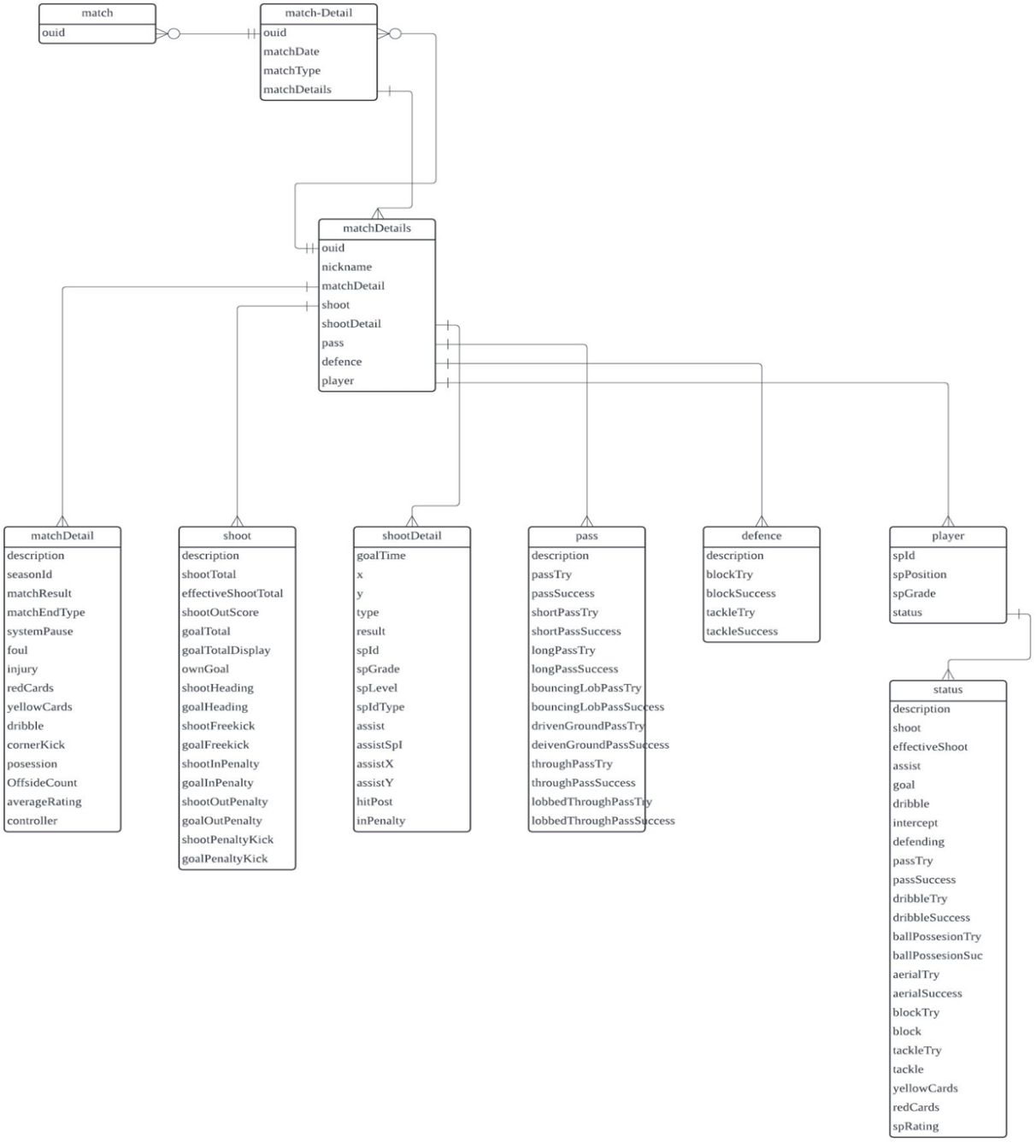
- 데이터 수집 일자: 7/10일 오후 1시 ~ 오후 5시
- 대상: FC 온라인 유저들의 리그 친선 경기 매치 데이터

EA SPORTS FC ONLINE

https://fconline.nexon.com/main/index

2.1.1 넥슨 Open API 데이터 구조

- 이 프로젝트에서 사용된 라이엇에서 제공하는 API 데이터 구조는 다음과 같습니다:



- 이 API에서 데이터를 수집하는 과정은 다음과 같이 진행됩니다:
  1. **MATCH API**
     - **목적**: FC 온라인 유저들의 고유 ouid를 가져옵니다.
     - **출력**: ouid 리스트
  2. **MATCH Detail API**
     - **입력**: MATCH API에 있는 ouid를 입력합니다.
     - **목적**: ouid를 입력하여 유저들의 게임 경기 정보를 가져옵니다.
     - **출력**: ouid, matchDate, matchType, matchDetails
- 이후 matchDetails에 속해있는 ouid, nickname, matchDetail, shoot, shootDetail, pass, defence, player을 출력하고, matchDetail, shoot, shootDetail, pass, defence, player은 matchDetails에 속해있고 위의 ERD처럼 각자 고유의 컬럼을 가지게 됩니다.

2.1.2 데이터 추출(Extraction)

- **Nexon Open API**: 모든 FC온라인 유저들을 대상으로 매치 데이터를 추출합니다.
- **FC 온라인 웹 크롤링:** FC 온라인 유저들의 구단 가치 데이터를 추출합니다.
  - 크롤링의 경우, 다음 코드를 이용하여 BP(구단가치) 데이터를 추출합니다.

```
 # 피파 온라인 페이지에서 구단 BP 정보를 가져와 크롤링하는 코드1
#match_df에 있는 닉네임을 가져와서 리스트화 하고, 휴면계정, 그리고 이름

name_list = match_df['nickname'].to_frame()['nickname'].to_l
name_list = [name for name in name_list if name != "설폡"]
name_list = [name for name in name_list if name != "맨유풋볼클
name_list = [name for name in name_list if name != "yourface
name_list = [name for name in name_list if name != "Dokkaebi
name_list = [name for name in name_list if name != "알개"]
name_list = [name for name in name_list if name != "nasdh85"
name_list = [name for name in name_list if name != "자은생제리
name_list = [name for name in name_list if "휴면" not in name
name_list = [name for name in name_list if name != "탑가드인자
name_list = [name for name in name_list if name != "김아린남편
name_list = [name for name in name_list if name != "the1ast"
name_list = [name for name in name_list if name != "요요이2"]
name_list = [name for name in name_list if name != "체코페이크
```

```
name_list = [name for name in name_list if name != "레드데빌마

# 중복 제거 (고유 요소만 유지)
unique_name_list = list(set(name_list))
name_list = pd.read_csv('C:/Users/82103/Desktop/name_list.cs

# 피파 온라인 페이지에서 구단 BP 정보를 가져와 크롤링하는 코드2
# chrome 웹 드라이버를 가져와서 창이 열릴 수 있도록 20초를 기다려준다,
driver = webdriver.Chrome()
driver.get('https://fconline.nexon.com/main/index')
driver.implicitly_wait(20)

# 피파 온라인 페이지에서 구단 BP 정보를 가져와 크롤링하는 코드3
# 피파 온라인 페이지 접속 - 우측 중단에 구단주 클릭 -> 닉네임 입력 -> 검
count = 0
bp_list = []

for i in range(0,7793):
    try:
        # 드롭다운 버튼 누르기
        driver.find_element(By.XPATH, '//*[@id="middle"]/div
        # 구단주로 변경 버튼 누르기
        driver.find_element(By.XPATH, '//*[@id="middle"]/div
        # 닉네임 칸 클릭 및 닉네임 입력
        click = driver.find_element(By.XPATH, '//*[@id="txtS
        click.clear()  # 이전 텍스트를 지움
        click.send_keys(name_list[i])
        driver.find_element(By.XPATH, '//*[@id="middle"]/div
        # 잠시 대기 후, 새로운 창으로 전환
        time.sleep(1)  # 페이지 로딩 시간을 고려하여 충분히 대기
        # 경고 창 감지
        try:
            alert = driver.switch_to.alert
            alert_text = alert.text
            alert.accept()
            print(f"Alert detected: {alert_text}")
            continue  # 이름이 존재하지 않을 경우 다음 루프로 넘어감
        except (NoAlertPresentException, UnexpectedAlertPres
            pass
```

```
            # 모든 창 핸들 얻기
            all_handles = driver.window_handles
            # 새로운 창으로 전환
            driver.switch_to.window(all_handles[-1])
            # 새로운 창에서 작업 수행
            driver.find_element(By.XPATH, '//*[@id="profilePop"]
            user_bp = driver.find_element(By.XPATH, '//*[@id="pr
            print(f'{count}번째 인덱스', name_list[i], user_bp)
            bp_list.append([name_list[i],user_bp])
            count += 1
            # 새로운 창 닫기
            driver.close()
            # 원래 창으로 전환
            driver.switch_to.window(all_handles[0])
        except NoSuchElementException as e:
            print(f"Error encountered for name {name_list[i]}: {
            continue  # 다음 루프로 넘어감
        except TimeoutException as e:
            print(f"Timeout error for name {name_list[i]}: {e}")
            continue  # 다음 루프로 넘어감

bp_data = pd.DataFrame(bp_list)
bp_data.to_csv('bp_data.csv', index=False)

# 크롤링을 통해 가져온 닉네임과 구단 가치 csv 파일을 가져온다
bp_data1 = pd.read_csv('/content/drive/MyDrive/Team_Minsoo_t
bp_data2 = pd.read_csv('/content/drive/MyDrive/Team_Minsoo_t
bp_data3 = pd.read_csv('/content/drive/MyDrive/Team_Minsoo_t
match_df = pd.read_csv('/content/drive/MyDrive/Team_Minsoo_t
```

2.1.3 데이터 변환(Transformation)

- **JSON 파싱: pd.json_normalize를 이용하여** 데이터 프레임화 시켜줍니다.

2.1.4 데이터 로드(Load)

- **CSV 파일 저장**: 최종적으로 병합된 데이터프레임을 CSV 파일 형식으로 저장하여 분석을 위한 데이터셋을 완성합니다.

2.2 데이터 설명

2.2.1 데이터 전처리

```
#seasonId는 drop한다
temp.drop(columns=['matchDetail.seasonId'], inplace = True)
```

- seasonId는 승리를 예측하는 것에 필요가 없다고 생각되어 drop 하였습니다.

```
#매치 타입은 드랍 시킨다. 왜냐면, 승리에 연관이 없는 변수이다.
temp.drop(columns=['matchDetail.matchEndType'], inplace = True)
```

- matchEndType은 리그 친선경기 데이터만 불러왔으므로, drop 하였습니다.

```
#승,무,패를 라벨 인코딩 해준다. 승의 경우 1, 무의 경우 0, 패의 경우 -1로 
temp['matchDetail.matchResult'] = temp['matchDetail.matchResult
```

- 승, 무, 패의 경우 각각 1, 0, -1로 라벨 인코딩해주었습니다.

```
#컨트롤러의 여부의 경우 pd.get_dummies를 활용하여 원핫인코딩을 실행한다.
controller = pd.get_dummies(temp, columns=['matchDetail.control
```

- controller의 경우, keyboard, gamepad로 나누어져있으므로, 순서에 지장을 주지 않는 One-Hot Encoding을 실시하였습니다.

```
#결측치인 데이터는 모두 제거한다
# 이유는 데이터의 결측치에 해당하는 열의 모든 정보들이 아예 없었기 때문에
temp.dropna(axis=0)
```

- 결측치의 경우 모두 제거해주었습니다.
    - 결측치 데이터를 모두 제거한 이유는 매치 기록에 대해서 유추할 수 있는 어떠한 데이터도 남아있지 않았기 때문에, 원본 데이터에서 제거해주었습니다.

```
# 자책골을 1골만 넣는 경우에는 그럴수도 있지만, 2골의 경우에는 거의 고의라고
# 자책골을 진짜 2골을 우연히 넣는 경우는 있지만, 지려고 의도하지 않는 이상 그
# 따라서 제거해준다.
# 어차피 이상치나 값을 제거하는 과정에서 사라진다. 승 무 패가 값이 딱 떨어지
print(temp[temp['shoot.ownGoal']>=1]['matchDetail.matchResult']
```

```
print(temp[temp['shoot.ownGoal']>=2]['matchDetail.matchResult']
temp = temp[temp['shoot.ownGoal']>=2]
```

- 자책골을 넣은 경우는 2골 이상부터 제거하였습니다. 2골 이상 자책골을 넣는 경우, 승리를 하기 위한 플레이, 게임에 정상적으로 임하기 위한 플레이가 아닌 이벤트 참여 목적으로 게임에 참여하거나 일부러 게임을 패배하기 위해 자책골을 넣는 비정상적 게임 플레이로 유추될 수 있으므로, 분석, 모델링의 해석의 편향성을 없애기 위해 제거하였습니다.

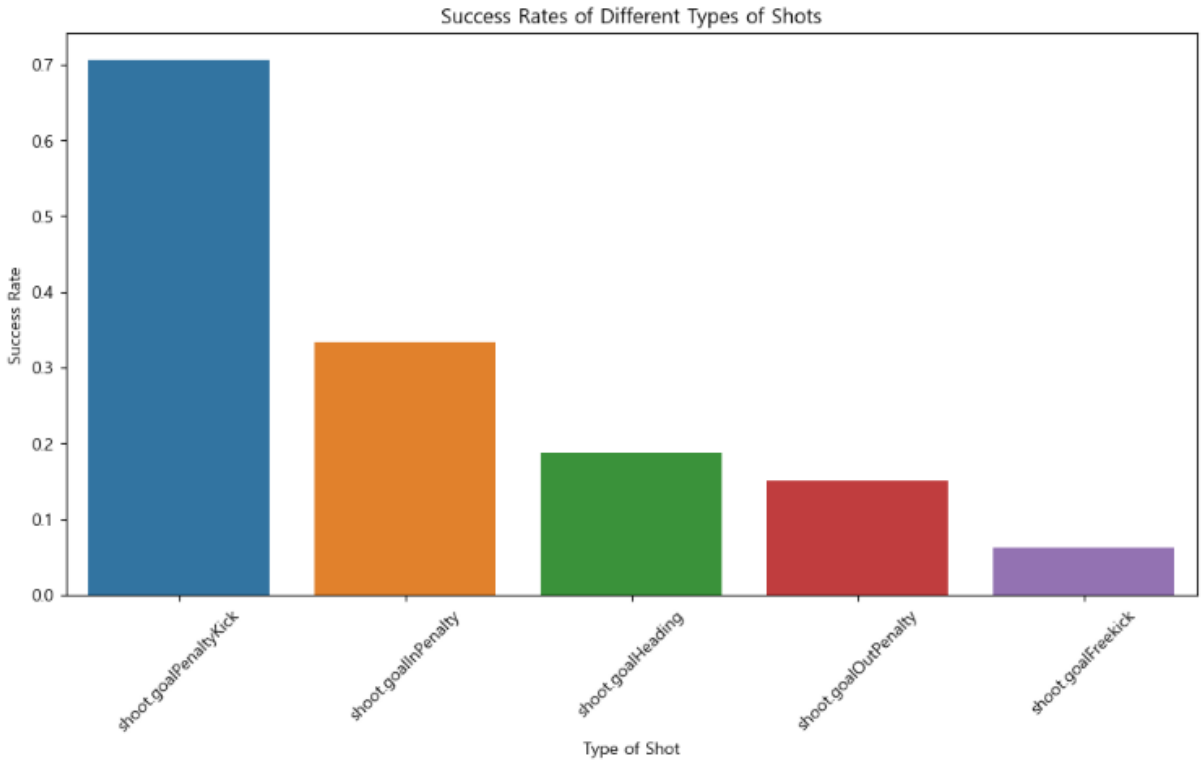
2.2.2 데이터 특징

3. 데이터 분석

3.1 상준 가설

3.1.1 EDA

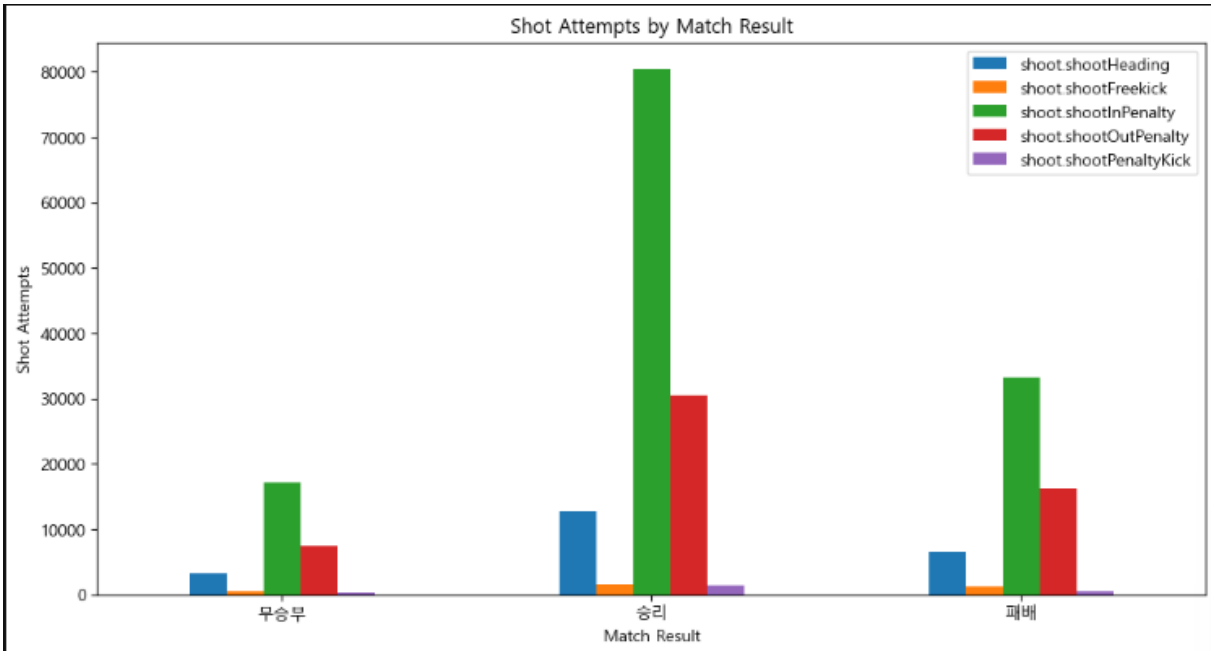
- **결과 상관없이 어떠한 형태의 슛(헤딩,프리킥,패널티인, 패널티 아웃)이 골로 많이 이어졌는가?**

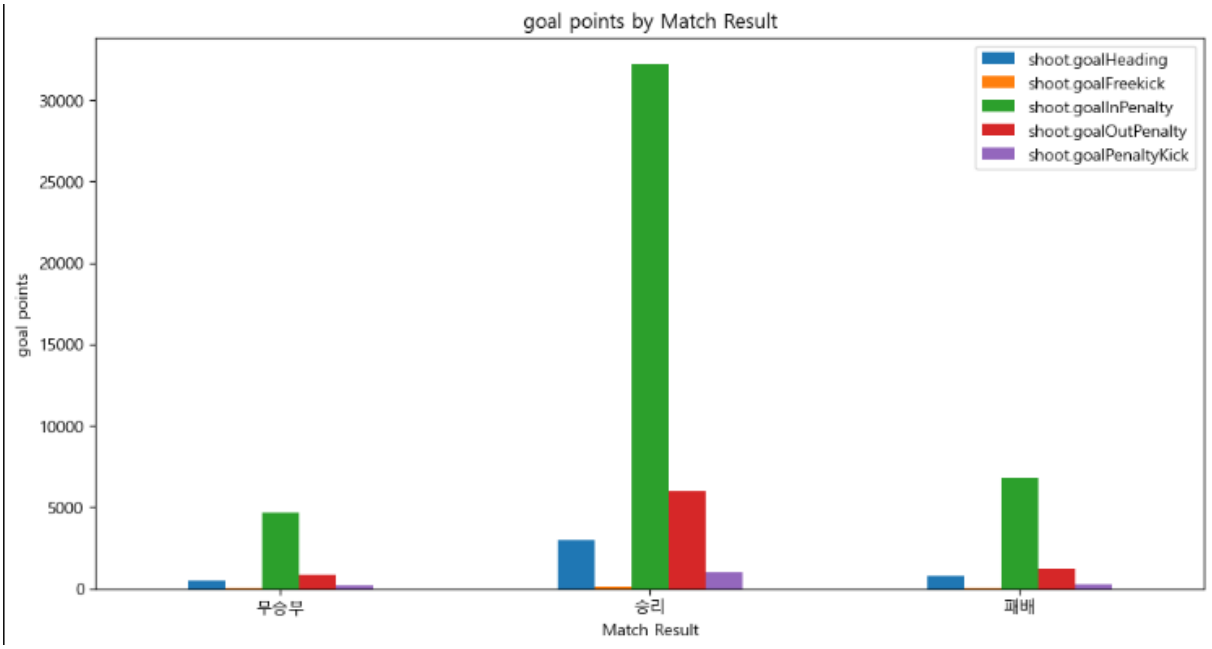


패널티킥의 성공률이 70.63%로 가장 높은 이유는 게임 내에서 플레이어가 정확하게 컨트롤할 수 있으며, 골키퍼의 반응 패턴을 예측하여 쉽게 득점할 수 있습니다. 패널티 박스 내 슛은

33.38%의 성공률을 보이며, 이는 골대와 가까운 거리에서 시도되며 상대 수비가 약화된 상황에서 주로 발생하기 때문으로 보입니다. 헤딩의 성공률은 18.83%로 낮은데, 이는 타이밍과 정확성이 요구되며 수비수와의 경합이 있기 때문입니다. 패널티 박스 외 슛의 성공률은 14.99%로, 거리와 골키퍼의 반응 시간 때문에 낮습니다. 프리킥의 성공률은 6.23%로 가장 낮으며, 이는 정교한 조작과 높은 기술력이 요구되고 수비벽과 골키퍼의 위치를 고려해야 하기 때문입니다.

- **결과별(승,패,무) 슛, 골 시도는 어떤 식으로 이루어졌는가?**



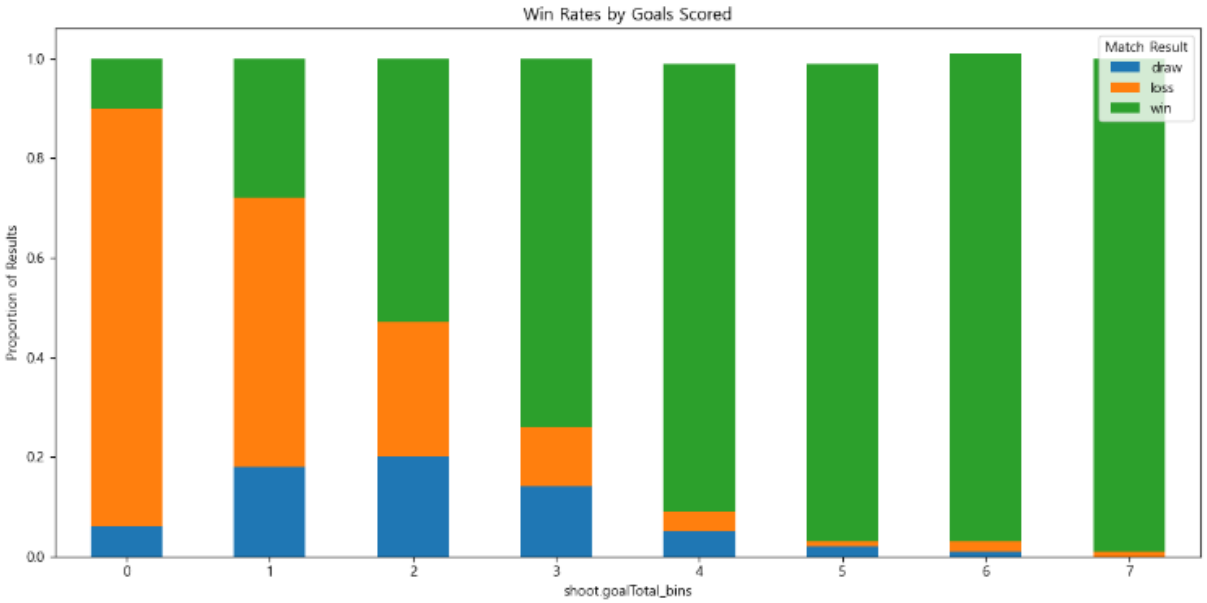


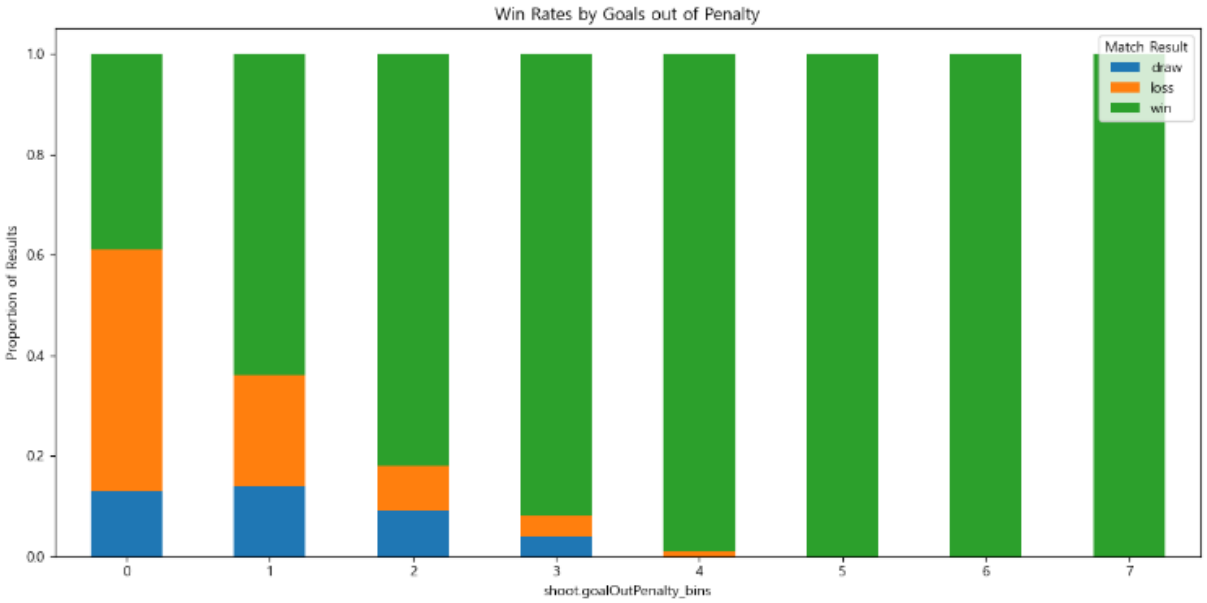
경기 결과에 따른 다양한 슛 유형별 골 성공 수치를 추가로 분석하여, 각 경기 결과(무승부, 승리, 패배)에 따라 다양한 슛 유형이 골로 이어진 횟수를 분석하였습니다. 경기에서 승리한 경우, 모든 슛 유형에서 골 수가 가장 많았으며, 이는 공격 기회의 빈도와 성공률이 높았음을 보여줍니다. 특히, 패널티 박스 내와 외에서의 슛, 그리고 패널티킥 상황에서 많은 득점이 이루어졌다는 점이 두드러집니다. 이는 승리한 경기에서 플레이어들이 효율적으로 득점 기회를 창출하고, 이를 성공적으로 마무리했다는 것을 의미합니다.

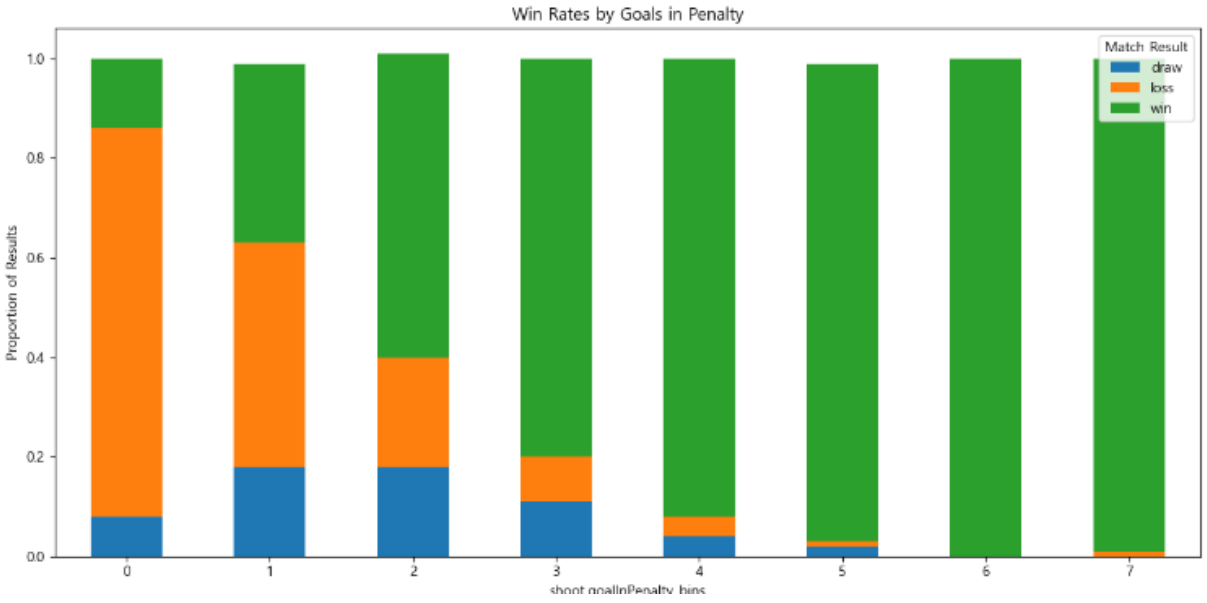
반면, 패배한 경기에서는 모든 슛 유형에서 골 수가 현저히 적었습니다. 이는 공격 기회가 적거나, 득점 성공률이 낮았음을 시사합니다. 패널티 박스 내에서의 슛과 패널티킥에서도 골 수가 적다는 것은 패배한 경기에서 플레이어들이 효과적인 공격을 펼치지 못했거나, 결정적인 순간에 득점하지 못했다는 것을 나타냅니다.

무승부의 경우, 골 수는 모든 슛 유형에서 중간 정도로 나타났습니다. 이는 특정 슛 유형에서의 우위를 점하지 못하고, 전체적으로 균형 잡힌 경기 양상을 보였음을 의미합니다. 이는 무승부 경기가 공격과 수비의 균형이 맞춰진 경기였거나, 결정적인 득점 기회가 양 팀 모두에게 충분하지 않았음을 시사합니다. 득점한 슛 상황에서 역시 슛시도에 비해 수가 적을 뿐, 비슷한 비율을 보이고 있습니다.

- 총 골 수와 페널티 안/밖에서의 골이 어느 정도 이상일 때 승리할 확률이 높을까?


Win Rates by Goals Scored
Match Result
draw
loss
win
Proportion of Results
shoot.goalTotal_bins


Win Rates by Goals out of Penalty
Match Result
draw
loss
win
Proportion of Results
shoot.goalOutPenalty_bins


Win Rates by Goals in Penalty
Match Result
draw
loss
win
Proportion of Results
shoot.goalInPenalty_bins

총 골 수에 따른 경기 결과
0~1골: 승리 확률이 낮고 패배 확률이 높습니다.
2~3골: 승리 확률이 증가하고 패배 확률이 감소합니다.
4골 이상: 승리 확률이 매우 높아지며, 특히 5골 이상에서는 거의 모든 경기를 승리합니다..

패널티 박스 외 골 수에 따른 경기 결과
0골: 승리와 패배 확률이 비슷하며, 무승부 확률도 상당히 높습니다.
1골: 승리 확률이 대폭 증가하며, 패배 확률이 크게 감소합니다.
2골 이상: 승리 확률이 매우 높아지며, 특히 3골 이상에서는 거의 모든 경기를 승리합니다.

패널티 박스 내 골 수에 따른 경기 결과
0~1골: 패배 확률이 매우 높습니다.
2골: 승리 확률이 크게 증가하며, 패배 확률이 급격히 감소합니다.
3골 이상: 승리 확률이 매우 높아지며, 특히 4골 이상에서는 거의 모든 경기를 승리합니다.

총 골 수, 패널티 박스 내 및 1골 정도만 높아져도,득점 수가 높을수록 승리 확률이 증가하는 경향을 보였습니다. 이는 플레이어들이 공격 기회를 효과적으로 활용하고, 다양한 득점 전략을 통해 경기에서 승리할 수 있음을 시사합니다. 이러한 인사이트를 토대로, 득점 전략을 더욱 세부적으로 분석하여 각 슛 유형과 그 빈도, 결과가 경기 승패에 어떻게 영향을 미치는지 이해하기 위해 아래와 같은 가설을 세우고 추가 분석을 진행하였습니다.

3.1.2 Drill Down: 선수가 시도한 슛의 종류, 슛의 빈도, 슛의 결과는 유의미하게 매치 결과에 영향을 줄 것이며, 기술 난이도가 높은 슛을 성공했을 시 승률이 더 높을 것이다.

피파 온라인 4의 유저가 조작하여 시도할 수 있는 슛은 다음과 같은 슛이 있으며, 일반적인 난이도는 다음과 같습니다.

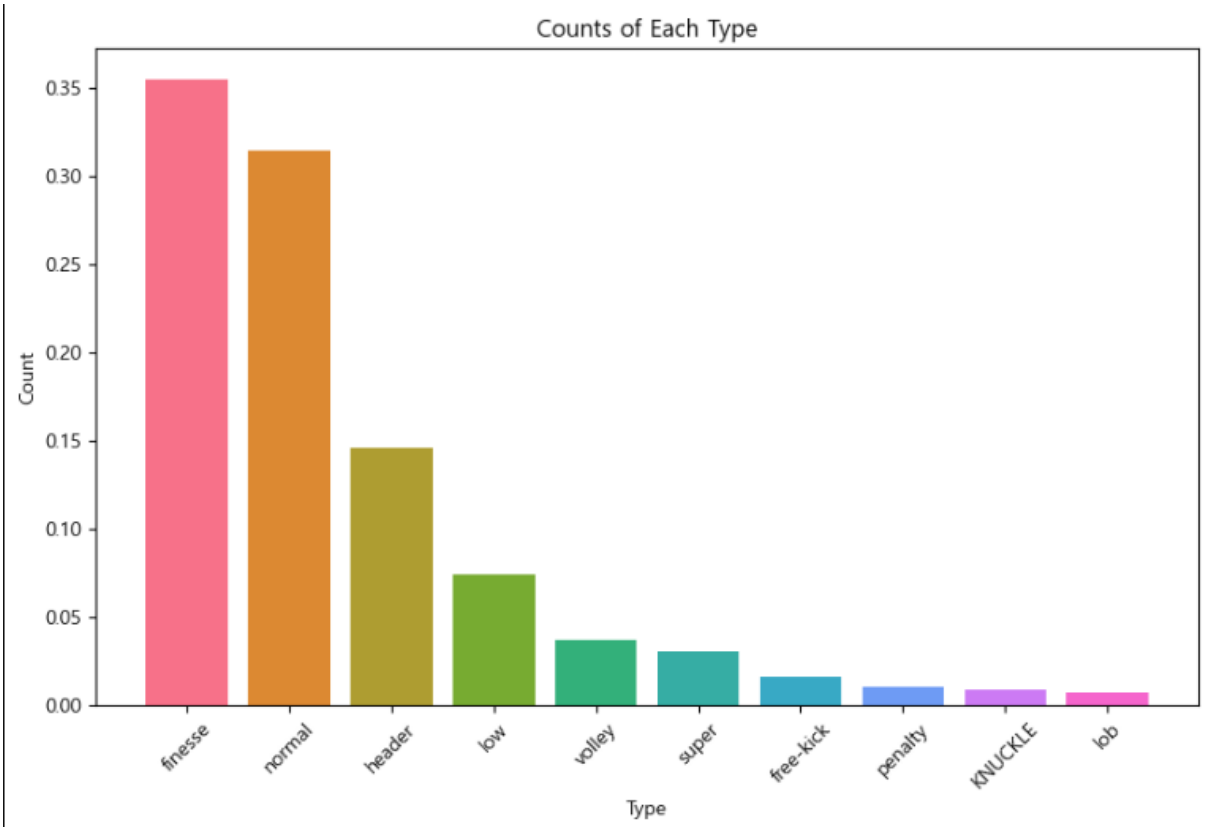
피파 온라인 4 슛 유형 난이도 평가

슛 이름	난이도	간략한 설명
Normal	낮음	가장 기본적인 슛으로, 특별한 기술이 필요하지 않습니다.
Finesse	중간	공을 감아차서 골키퍼를 속이는 슛으로, 정확도가 필요합니다.
Header	중간	공중에서 머리로 하는 슛으로, 타이밍과 위치 선정이 중요합니다.

Lob	중간	골키퍼를 넘겨서 골을 노리는 슛으로, 정확한 힘 조절과 타이밍이 필요합니다.
Flare	높음	화려한 기술을 사용한 슛으로, 높은 기술력과 상황 판단이 요구됩니다.
Low	중간	낮은 슛으로, 주로 골키퍼의 발 밑을 노립니다. 정확성과 빠른 반응이 필요합니다.
Volley	높음	공이 땅에 닿기 전에 차는 슛으로, 타이밍과 기술이 매우 중요합니다.
Free-kick	높음	정지된 상황에서 차는 슛으로, 골키퍼와 수비벽을 넘기기 위한 높은 기술과 정확성이 필요합니다.
Penalty	중간	11미터 거리에서 골키퍼와 1:1로 차는 슛으로, 심리적 압박이 큽니다.
Knuckle	높음	공이 회전 없이 직선으로 날아가는 슛으로, 공의 임팩트와 정확한 킥이 필요합니다.
Bicycle	매우 높음	공중에서 몸을 뒤로 젖혀 발리하는 슛으로, 매우 높은 기술력과 정확성이 요구됩니다.
Super	높음	매우 강력한 힘으로 차는 슛으로, 정확성과 함께 힘의 조절이 중요합니다.

- 시도할 수 있는 슈팅 기술 중 `Bicycle`과 `Flare`의 경우 전체 데이터 중 전부 4건이기 때문에, 분석에서 제외합니다.
- 또한 22건의 페널티 존에서 기록된 `Free-kick` 역시 제외합니다.

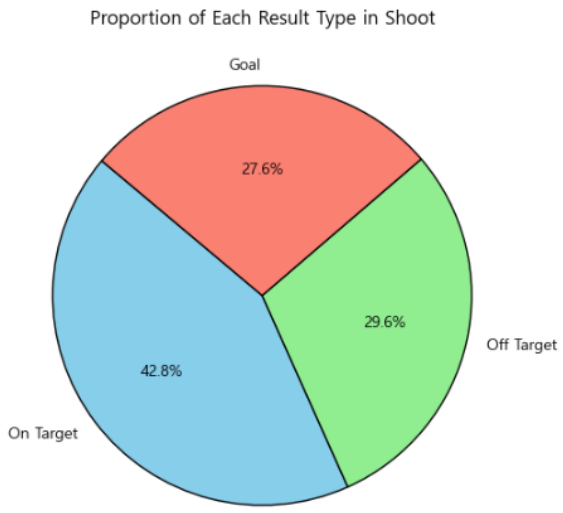
1. 난이도가 높은 기술일수록 시도횟수가 적을 것이라고 판단하고, 이를 시각화하였습니다.

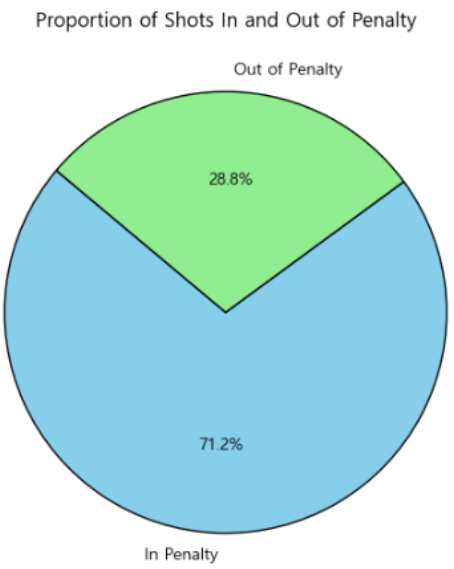


몇 가지 눈에 띄는 기술을 살펴보자면 `Finesse(감아차기 슛)`는 상대적으로 난이도가 중간 정도로, 많은 플레이어들이 자주 시도합니다. 높은 성공률과 적당한 난이도로 인해 선호도가 높습니다. `Normal(기본 슛)`은 난이도가 낮으며, 가장 기본적인으로, 특별한 기술이 필요하지 않아 모든 상황에서 사용 가능합니다 난이도가 높은 `super(파워샷)`, `Free-kick(프리킥)` 등 기술은 시도 횟수는

`Finesse`와 `normal`을 합친 비율(67%)보다 적게 시도됩니다.

2. **또한 난이도가 높은 기술은 주로 어떤 결과(Result)를 시사하고, 어떤 상황(InOutPenalty)에서 발생하는 지 알아보기 위해,결과별, 상황별 슈팅 시도 비율을 시각화 하였습니다.**

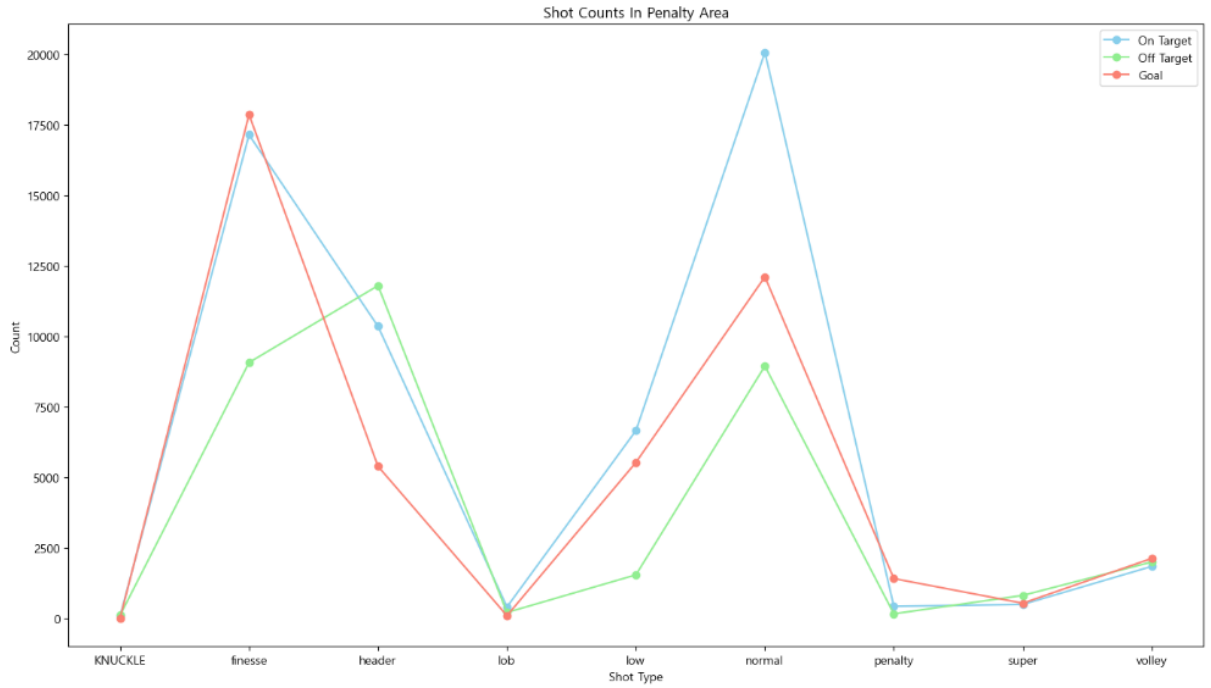


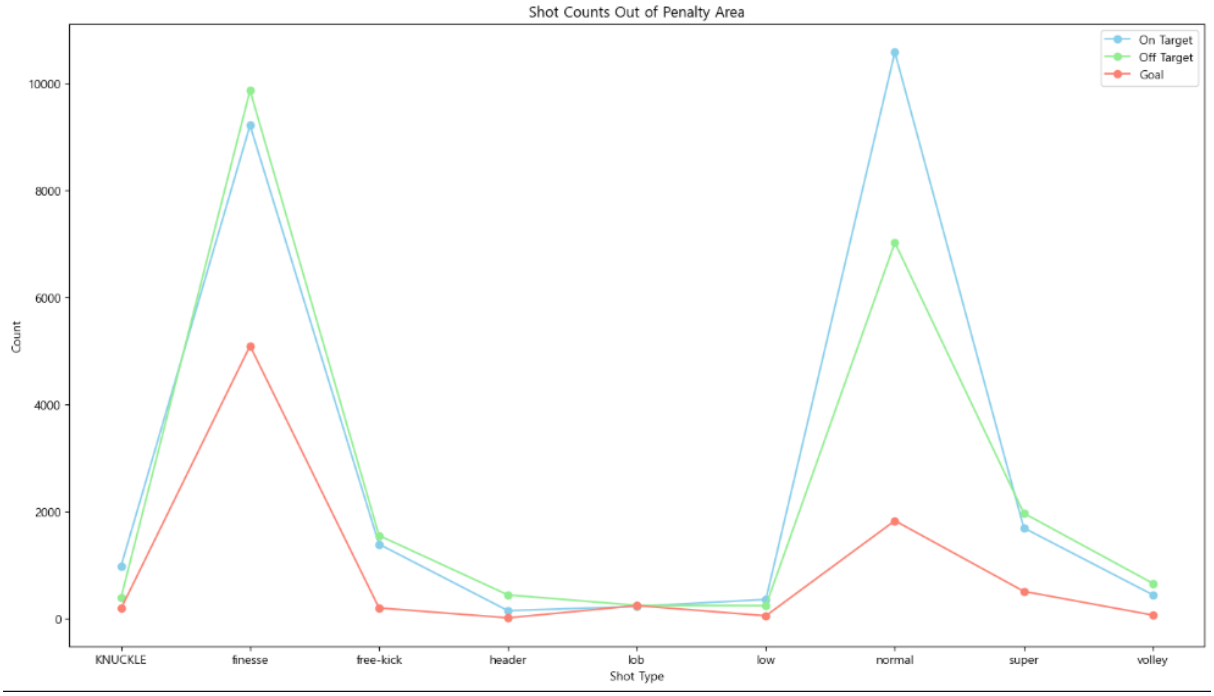


전체 슈팅 시도 중, `Goal`로 이어진 슈팅의 비율은 27.6%로, 전체 1/3에 못미치는 수준입니다. 골문을 향해 정확히 날아간 슛의 경우(득점 포함x) 42.81%, 골문을 벗어난 슛은 29.58%로 나타납니다. 대부분의 슛시도가 패널티 지역 내에서 발생하고, 이는 패널티 박스 내에서 슛 시도가 많고(71.2%), 공격이 주로 이 지역에서 이루어진다는 것을 나타냅니다. 패널티 박스 외에서도 슛 시도가 상당히 이루어지지만, 패널티 박스 내에서의 시도(28.8%)에 비해 적게 시도됩니다.

난이도가 높은 슛 기술은 일반적으로 성공률이 낮고, 정확도가 떨어질 수 있습니다. 높은 기술력과 정확성을 요구하지만, 성공 확률이 낮아 골문을 벗어나거나(Off Target), 골키퍼에게 막히는(On Target) 경우가 많을 것이며, 안정성을 높이기 위해 패널티 지역에서 더 많이 시도할 것이라고 생각하고, 다음 분석을 시도하였습니다.

3. 어떤 상황(InOutPenalty)에서 특정 슛이 시도되는지 먼저 살펴보았습니다.

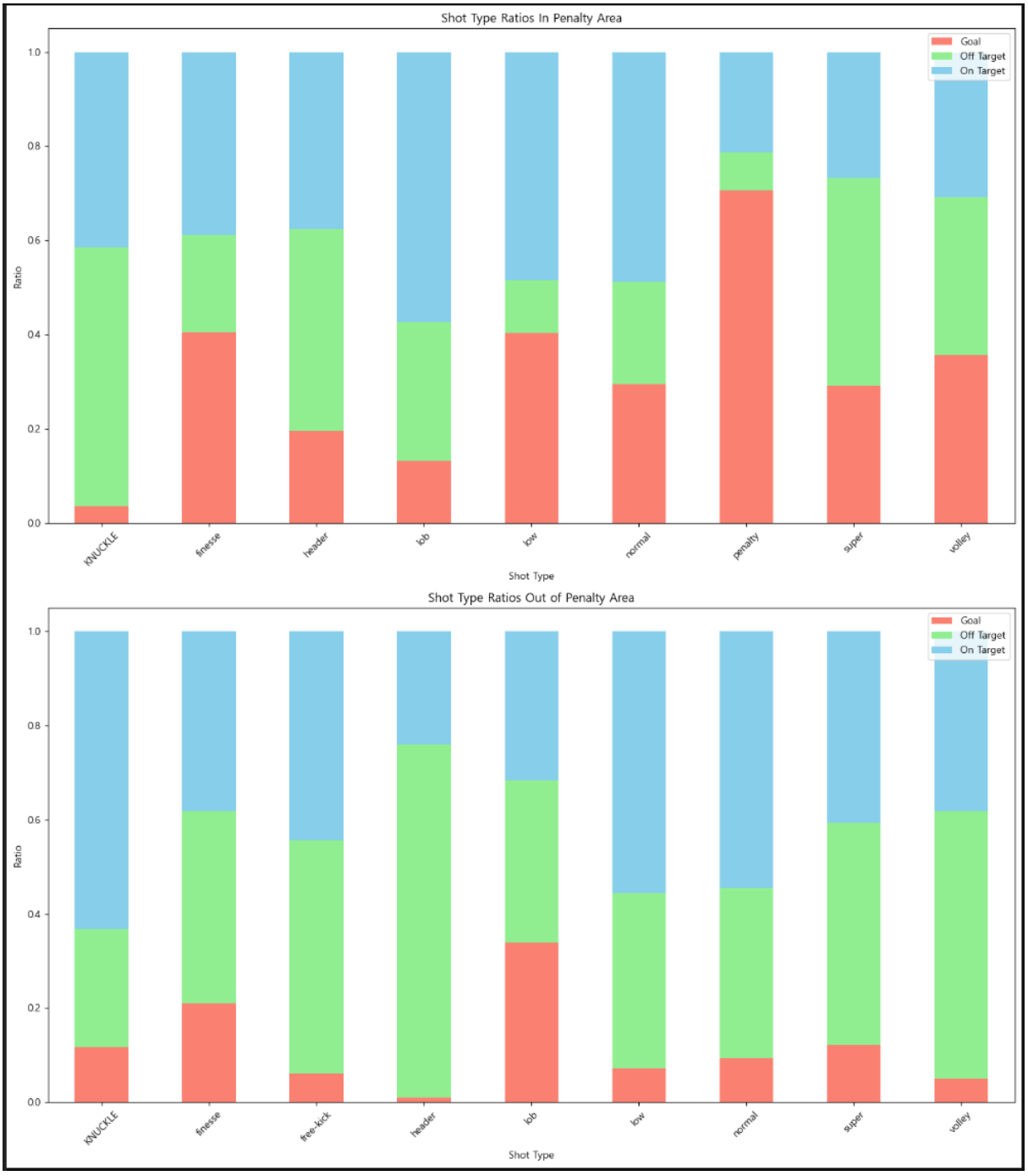




페널티 존에서 난이도가 높은 `KNUCKLE`은 잘 시도되지 않습니다. 거리와 타이밍 조절이 중요하기에 페널티 존 외에서는 무회전슛의 시도가 많습니다. 중간 난이도의 `finesse`의 경우, 골대와 가까운 거리에서 효과적이기에, 페널티 존 내에서 가장 많이 시도된 슛 유형으로 나타납니다. `header`의 경우 페널티 존 외에서는 헤딩 시도가 거의 이루어지지 않습니다. `lob`은 페널티 존 외에서 약간 더 많이 시도됩니다. 골키퍼의 발 밑을 노리는 슛으로, `low`는 페널티 존 내에서 많이 시도됩니다. `normal`은 모든 상황에서 사용할 수 있는 슛이며, 모든 상황에서 많이 시도되지만, 주로 페널티 존에서 많이 시도됩니다. `penalty`와 `freekick`은 각각 페널티 안/밖에서만 시도됩니다. 이는 축구 규칙

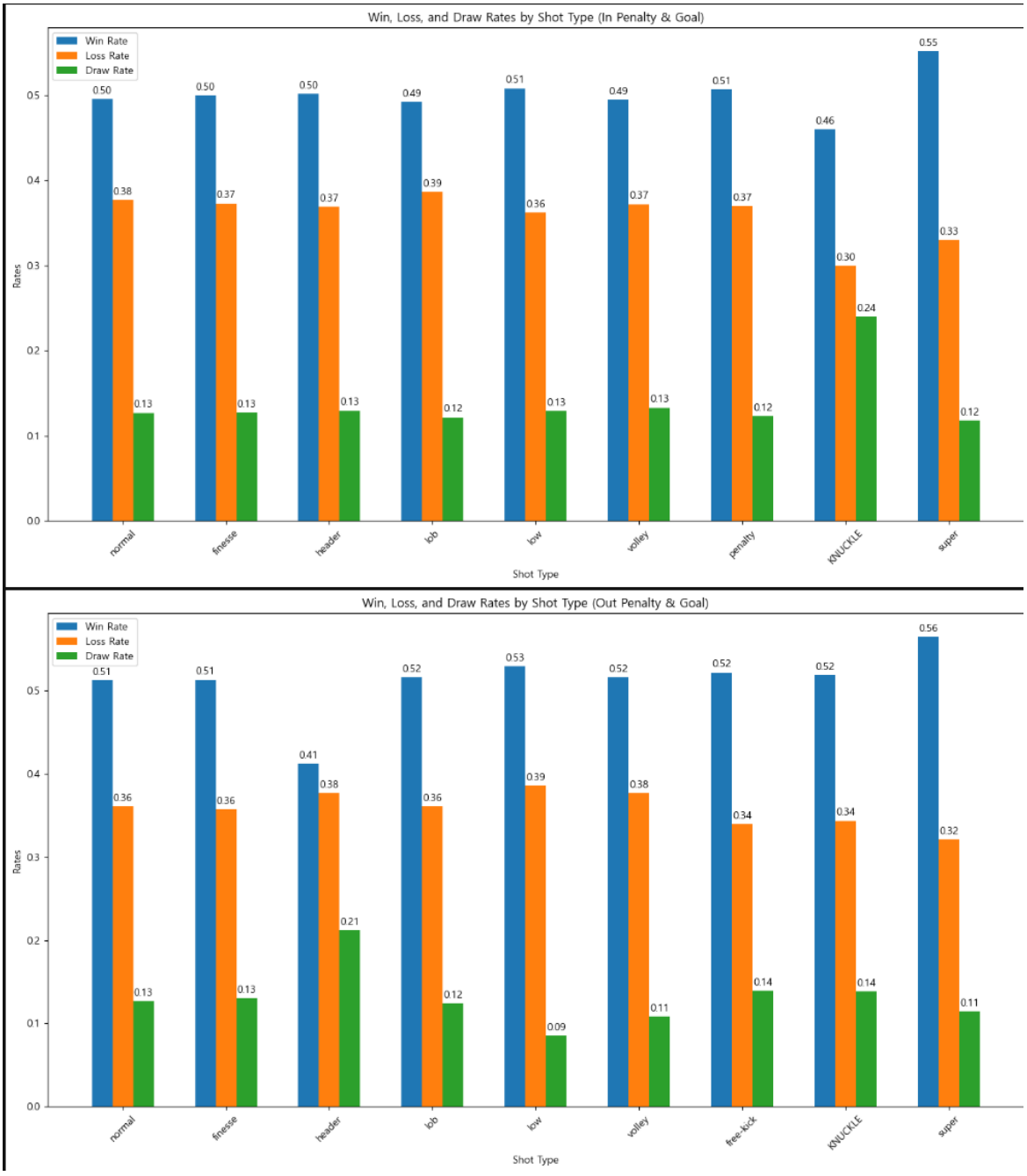
상 해당 슛들이 특정 위치에서 시작되기 때문입니다. `super`와 `volley` 모두 난이도가 높은 슛이지만, 전자는 일반적으로 멀리서 강력하게 차는 슛으로 패널티 존 밖에서 시도가 많은 반면, 후자는 높은 기술력과 정확성을 요구하기에 보다 안정권인 패널티 존 안에서 많이 시도됩니다.

4. 상황마다 어떤 결과(Result)를 시사하는지도 살펴보았습니다.



`KNUCKLE`은 기술 특성 상 페널티 존 안에서 시도될 때, 유효하지 않은 슛의 비율이 크게 증가하며, 골 비율 역시 감소합니다. 반면 마찬가지로 고난도지만 `finesse`의 경우 정확도가 중요하기 때문에, 페널티 존 안에서 득점률이 크게 증가합니다. `header`의 경우 역시 기술의 특성 상 페널티 존안에서 유효한 슛의 비율이 높고, 득점률 또한 높습니다. `lob` 의 경우 페널티 존 내에서 시도될 때와 페널티 존 외에서 시도될 때 각각 다른 특성을 보입니다. 페널티 존 내에서는 골키퍼를 넘기기 어려워 득점률이 낮지만, 시도 자체는 정확하게 이루어져 유효슈팅률 비율이 높습니다. 반면, 페널티 존 외에서는 골키퍼를 넘기는 데 성공하면 득점으로 이어질 가능성이 높아 득점률이 높지만, 거리의 영향으로 인해 유효슈팅률은 낮습니다. 낮은 슛인 `low` 는 안정적인 페널티 존 안에서 유효슈팅과 득점의 비율이 높습니다. `normal` 역시 비슷한 비율을 보입니다. `super`는 페널티 존 내에서 강한 힘으로 인해 높은 득점률을 보이지만, 정확도가 떨어져 많은 시도가 골문을 벗어납니다. 페널티 존 외에서는 힘이 강해도 정확도가 더 떨어져 득점률이 낮아집니다. `Volley` 는 페널티 존 내에서 높은 득점률을 보이지만, 공중에서의 슛이기 때문에 정확한 타이밍과 위치 선정이 어려워 많은 시도가 골문을 벗어납니다. 페널티 존 외에서는 매우 낮은 득점률을 보이며, 이는 먼 거리에서 공중에 떠 있는 공을 정확하게 타격하는 것이 매우 어렵기 때문입니다.

5. **지금까지 분석한 정보를 토대로, 특정 상황(InOutPenalty)에서 특정 결과(Result)를 보이는 슛 기술이 그렇다면 '전체 경기에서 어떤 최종 결과를 가져오는 지' 알아보았습니다.**



먼저 득점을 했을 때, 최종 매치의 승률이 어떻게 되는 지 시각화하였습니다.

대체적으로 고난도의 기술로 득점을 하였을 때, 높은 승률을 보이고 있습니다. **(고난도 최고 승률: 56%, 중간 난이도 최고 승률: 53%, 낮은 난이도 최고 승률 51%)**

`normal`의 경우 가장 많이 시도되는 슛으로, 난이도는 낮은 편입니다. 일반적으로 많이 시도되고, 페널티 존에서 많이 기술이 사용되지만, 페널티 존 밖에서 득점하였을 때 승률이 소폭 올라가는 것으로 확인됩니다. 중간 난이도의 `finesse`의 경우 역시 많이 사용되는 기술로, `normal`과 비슷한 지표를 보입니다. `header`의 경우 중간 난이도의 공중에서 머리를 사용하는 기술로, 페널티 존 내에서 시도되었을 때 밖에서의 승률보다 무려 9% 높게 나타납니다. 또 다른 중간

난이도의 `lob`의 경우 앞서 살펴 본 것처럼 페널티 존 외에서의 득점률이 높은 것처럼, 승률이 역시 더 높게 보이고 있습니다. 낮은 슛을 시도하는 `low`의 경우 앞서 페널티 존에서 많이 시도되고, 득점률도 높게 나왔던 것과 달리, 페널티 존 외에서 오히려 높은 승률을 보이고 있습니다. 높은 난이도의 `volley` 역시 `low` 가 기존에 페널티 존에서 좋은 지표를 보였던 것과 달리, 페널티 존 외에서 오히려 높은 승률을 보입니다. 높은 난이도의 `KNUCLE` 과 `super` 모두 기술의 특성상 페널티 존 외에서 득점 하였을 때 높은 승률을 보이며, `KNUCLE`의 경우 그 차이가 6% 가량 더 차이가 발생합니다.

3.1.3 결론

- 특정 상황(InOutPenalty)과 특정 결과(Result)에 따라 발생한 슛의 빈도와 비율이 어느 정도 매치의 승률과 일치한다는 것을 파악했습니다. 하지만 예상과 달리, 페널티 존 내에서 좋은 지표를 보였던 기술이 페널티 존 밖에서 아주 약간 더 승률이 높게 나타난 부분도 있습니다.

- 이러한 현상은 숙련도 등 요인에 의해 설명될 수 있습니다:
  - 유저의 숙련도(일반적인 기준)
    - 고숙련 유저: 고숙련 유저들은 게임 내에서 다양한 슛 기술을 매우 잘 활용할 수 있으며, 특히 페널티 존 밖에서 고난도의 슛 기술을 성공시키는 능력이 뛰어납니다.
    - 초보 유저: 반면, 초보 유저들은 상대적으로 쉬운 슛 기술을 선호하며, 페널티 존 내에서 더 안전하게 슛을 시도하는 경향이 있습니다.

- 즉, 페널티 존 내에서 좋은 지표를 보였던 기술들이 페널티 존 밖에서 약간 더 높은 승률을 보이는 현상은 유저의 숙련도, 슛의 특성, 게임 메커니즘, 전략적 접근 등의 여러 요인에 의해 발생할 수 있습니다.

- **분석 결과, 시도한 슛의 종류, 빈도, 결과는 경기 결과에 유의미한 영향을 미치며, 특히 기술 난이도가 높은 슛을 성공했을 때 승률이 더 높다는 것을 확인할 수 있었습니다.**

- 경기 상황에 맞는 슛 기술을 전략적으로 활용하여 득점 기회를 극대화, 유저들의 슛 기술을 향상시키고, 경기 전략을 개선할 필요가 있습니다.

3.1.4 전략 제시

피파온라인 4 유저 슈팅 활용 가이드

입문자 가이드

슛 기술	추천 상황	이유
Normal	모든 상황	기본적인 슛으로, 사용이 간편하고 모든 상황에서 효과적입니다. 페널티 박스 내외에서 모두 사용할 수 있어 다양한 슛 상황에 적합합니다.
Finesse	페널티 박스 안	감아차기 슛은 상대적으로 난이도가 중간 정도로, 페널티 박스 내에서 정확하고 효과적인 슛을 날릴 수 있습니다. 골대와 가까운 거리에서 특히 유효합니다.
Header	페널티 박스 안	헤딩 슛은 공중볼 상황에서 매우 유효하며, 특히 코너킥이나 크로스 상황에서 많이 시도됩니다. 정확한 타이밍이 중요하며, 페널티 박스 내에서 높은 유효 슈팅률과 득점률을 보여줍니다.
Low	페널티 박스 안	골키퍼의 발 밑을 노리는 낮은 슛은 페널티 박스 내에서 매우 효과적입니다. 안정적이고 성공 확률이 높은 슛 기술로, 페널티 박스 내에서 사용 시 높은 득점률을 보입니다.

숙련자 가이드

슛 기술	추천 상황	이유
Knuckle	페널티 박스 외	무회전슛은 거리와 타이밍 조절이 중요한 고난도 기술로, 페널티 박스 외에서 강력하고 예측 불가능한 궤적으로 골키퍼를 혼란시킬 수 있습니다. 페널티 박스 내에서는 유효하지 않은 슛의 비율이 증가하므로 외부에서 시도하는 것이 좋습니다.
Super	페널티 박스 외	강력한 파워샷은 먼 거리에서 사용될 때 매우 효과적입니다. 높은 난이도지만, 슛이 성공할 경우 득점으로 이어질 가능성이 높습니다. 페널티 박스 외에서 힘을 실어 슛을 시도하면, 골문을 벗어나거나 골키퍼에게 막히는 경우가 많습니다.
Volley	페널티 박스 안	발리슛은 공중에서 바로 차는 슛으로, 정확한 타이밍과 위치 선정이 필요합니다. 페널티 박스 내에서 사용하면 높은 득점 확률을 가지며, 먼 거리에서 공중에 떠 있는 공을 정확하게 타격하는 것이 매우 어렵기 때문에 페널티 박스 내에서 시도합니다.
Lob	페널티 박스 외	로빙슛은 골키퍼를 넘기는 슛으로, 페널티 박스 외에서 사용하면 효과적입니다. 거리의 영향을 받지만 성공 시 높은 득점 확률을 가집니다. 페널티 박스 내에서는 골키퍼를 넘기기 어려워 유효 슈팅률이 높지만 득점률이 낮습니다.
Finesse	페널티 박스 안	감아차기 슛은 정확도를 요구하지만, 페널티 박스 내에서 사용하면 득점률이 크게 증가합니다. 골대와 가까운 거리에서 특히

		효과적이며, 페널티 박스 내에서 가장 많이 시도된 슛 유형입니다.
Low	페널티 박스 내외	낮은 슛은 안정적이고 성공 확률이 높은 슛 기술로, 페널티 박스 내외에서 모두 유효합니다. 페널티 박스 내에서 사용 시 유효 슈팅과 득점의 비율이 높으며, 골키퍼의 발 밑을 노리는 슛으로 많은 득점 기회를 제공합니다.

3.2 승훈 가설

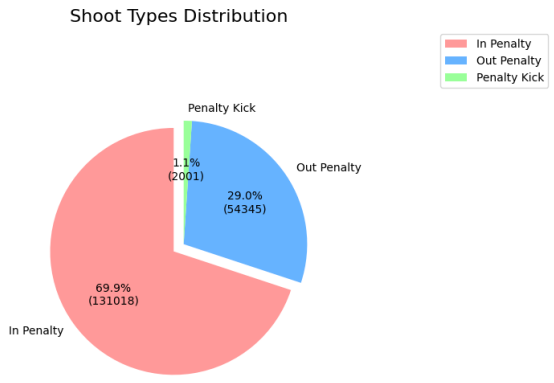
3.2.1 플레이어들은 패널티 구역 내&외 어느 곳에서 어떤 슈팅을 많이 할까?

📊 FC online(FCO)를 플레이 할 때, 개인적으로 패널티 구역 바깥에서 감아차기(finesse)와 , 일반슛(normal), 파워 슛(super)을 많이 시도하는데 다른 플레이어들의 양상은 어떻게 나타나는지에 대한 호기심과 이에 대한 전략을 제시하기 위해 분석을 진행하였습니다.

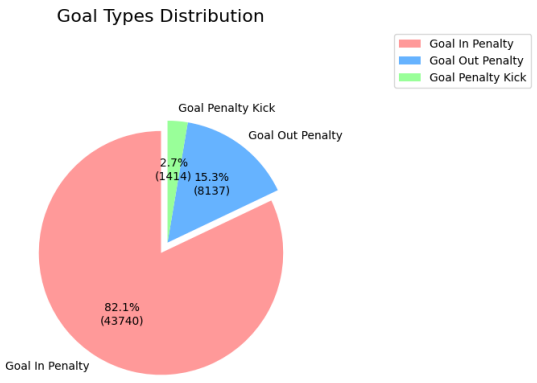
전처리 과정에서 메인 데이터 프레임에 JSON파일로 구성된 shootDetail을 데이터 프레임으로 만들었고 이것을 통해 데이터 중 슈팅 위치를 나타내는 좌표값인 x, y 컬럼을 사용하여 패널티 구역 내외 슈팅 위치을 알아보았습니다.

먼저 총 슈팅 수로 이루어지는 패널티 구역 내외 슈팅, 패널티 킥 중 어떤 슈팅을 많이 시도했는지에 대한 비율과 총 득점에 대한 패널티 구역 내외, 패널티 킥 중 어떤 것이 득점을 많이 했는지에 대한 비율을 수치로 확인 후 시각화 해보았습니다.

Shoot



Goal



이를 통해 슈팅은 패널티 구역 내에서 많이 이루어지고 득점으로 이어지는 슈팅 또한 패널티 구역 내에서 많이 발생하는 것을 알 수 있습니다. 그럼 어떤 위치에서 슈팅이 많이 이루어지고 득점까지 이어지는가에 대해 알아보기 위해 좌표를 통해 시각화해보았습니다.

시각화에 앞서 FCO에서는 각 x, y 값의 비율을 FIFA 공식 사이즈 규정하고 있으므로 다음과 같이 기준을 정하고 좌표를 경기장에 표현할 수 있도록 하였습니다.

FIFA 공식 경기장 사이즈 규정사이즈

- 길이(터치라인): 100-110m
- 너비(골라인): 64-75m

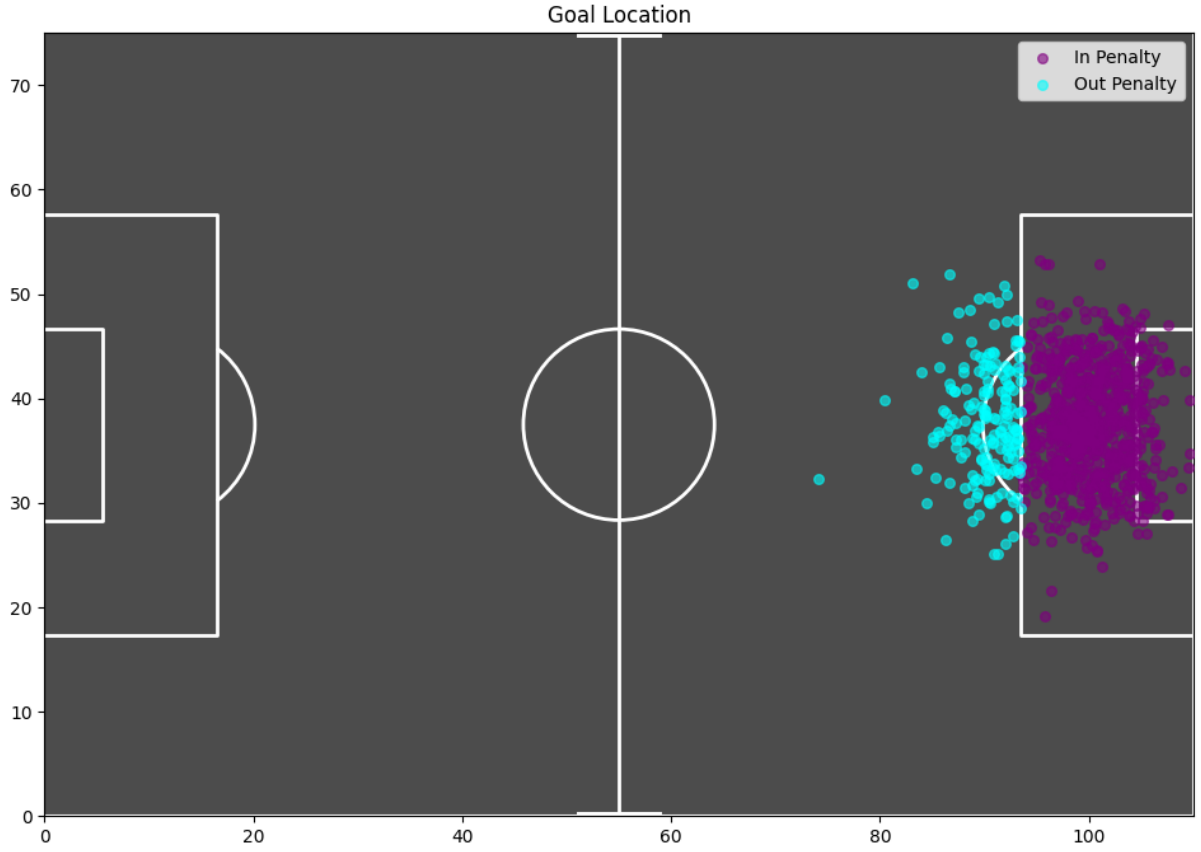
여기서는 큰 경기장의 기준인 110m, 75m를 기준으로 사용하였으며, 이 x, y 좌표를 계산하면 다음과 같이 표현할 수 있다.

x 좌표 (너비): x * 75m ≈ X_wm

y 좌표 (길이): y * 110m ≈ Y_lm

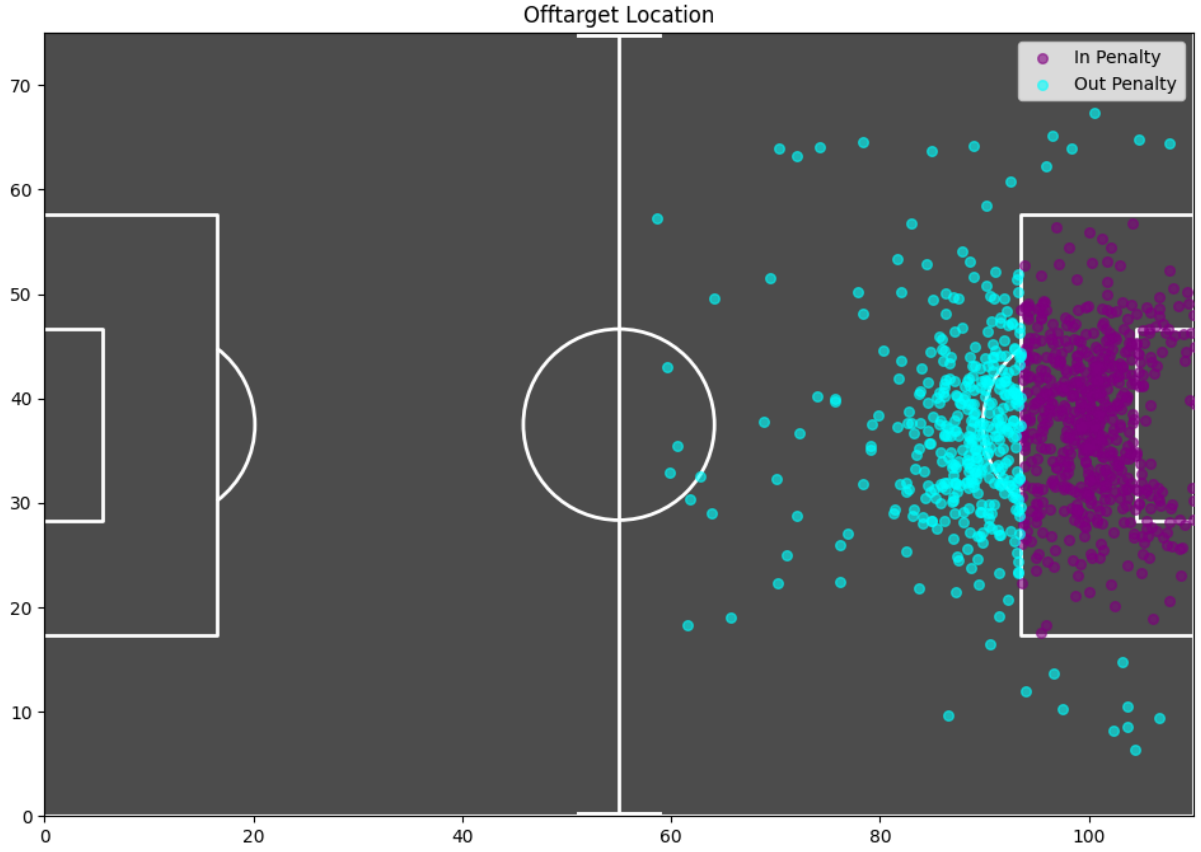
좌표의 값은 0, 1 사이의 값으로 이루어져 있으므로 왼쪽 아래 코너를 기준으로 너비와 길이에 대한 거리 지점의 백분율이 소수점 표기법으로 좌표 변환되어 있다는 것을 알 수 있습니다.

이 기준으로 시각화하면 다음과 같이 구성할 수 있습니다.



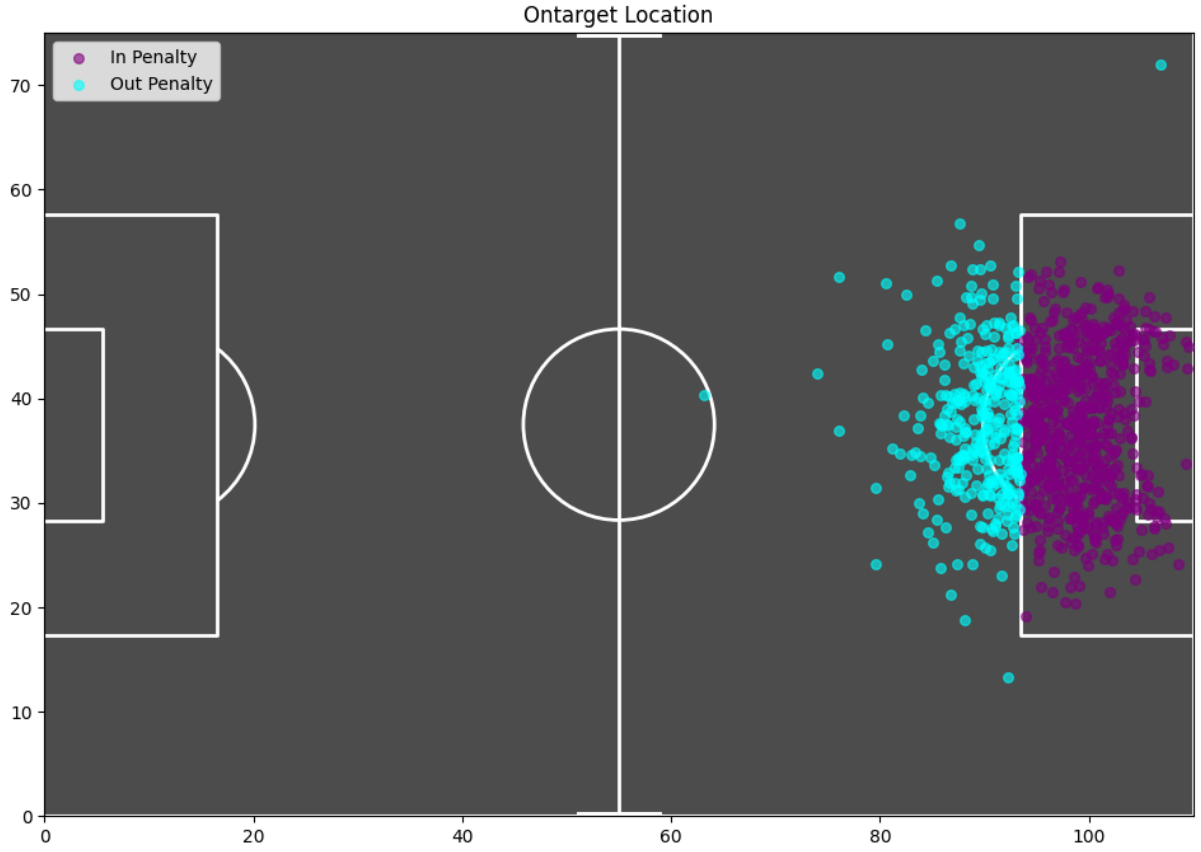
득점에 성공한 슈팅 좌표 1000개를 시각화한 모습입니다.

패널티 에어리어 안의 점 개수: 820 (82.00%)
패널티 에어리어 밖의 점 개수: 180 (18.00%)



득점에 성공하지 못한 슈팅(비유효슈팅) 좌표 1000개를 시각화한 모습입니다.

패널티 에어리어 안의 점 개수: 597 (59.70%)
패널티 에어리어 밖의 점 개수: 403 (40.30%)



득점에 성공하지 못했지만 유효슈팅으로 이어진 슈팅 좌표 1000개를 시각화한 모습입니다.

패널티 에어리어 안의 점 개수: 663 (66.30%)
패널티 에어리어 밖의 점 개수: 337 (33.70%)

(유효슈팅: 슈팅이 골대 안으로 향해서 골키퍼가 막아내지 않았다면 득점으로 이어지는 슛**)**

시각화 결과로 보아 앞서 확인했던 패널티 구역 내외 비율과 근사한 모습을 나타내는 것을 알 수 있습니다.

3.2.2 패널티 구역 내외에서 가장 많이 시도한 슛 타입과 가장 득점이 많은 슛 타입이 무엇인지 확인해보았습니다.

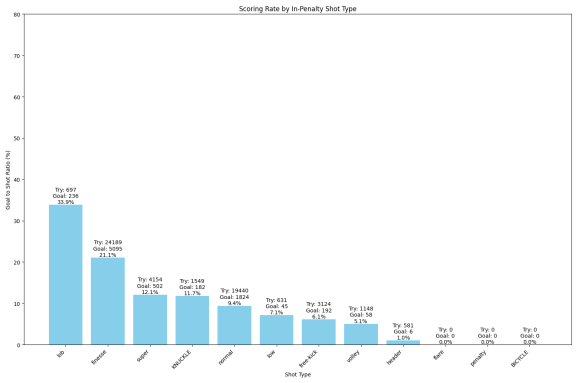
슛 타입은 FCO에서 다음과 같이 정의하고 있습니다.

- 1: **normal**(기본슛)
- 2: **finesse**(감아차기 슛)
- 3: **header**(헤딩슛)
- 4: **lob**(로빙슛)

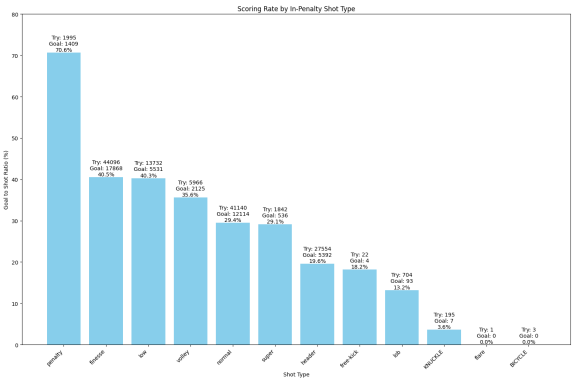
- 5: **flare**(플레어슛)
- 6: **low**(낮은 슛)
- 7: **volley**(발리슛)
- 8: **free-kick**(프리킥)
- 9: **penalty**(페널티킥)
- 10: **KNUCKLE**(무회전슛)
- 11: **BICYCLE**(바이시클킥)
- 12: **super**(파워샷)

이 기준을 가지고 패널티 구역 내 & 외 슈팅 대비 득점의 비율을 확인하고 시각화하였습니다.

Out_Penalty



In_Penalty

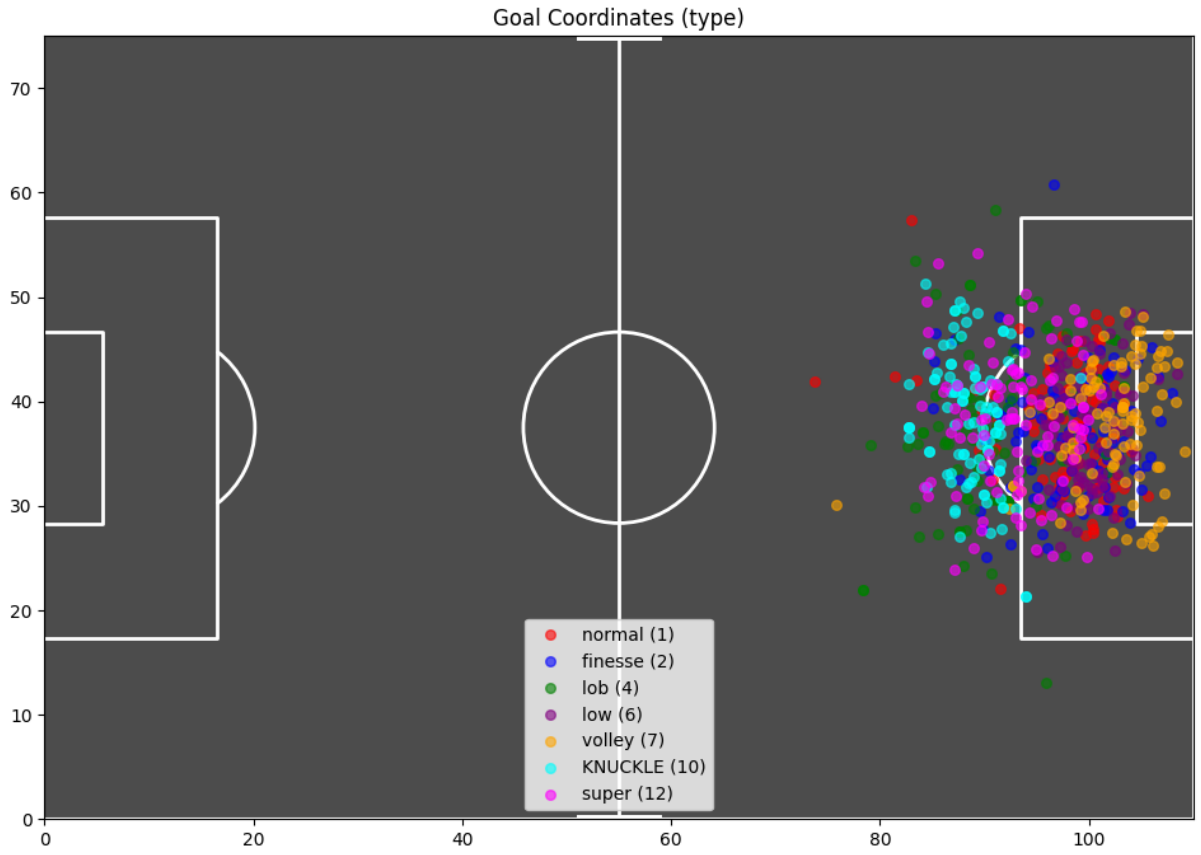


Out_Penalty 에서는 lob, finesse가 비교적 높게 나타났고, 다음으로 super, KNUCKLE, normal이 비슷한 득점 성공률을 보이고 있으며, 시도한 횟수 대비 성공률이 낮다고 판단되는 finesse, normal도 확인해볼 수 있었습니다.

In_Penalty 에서는 penalty kick이 압도적으로 성공률이 높은걸 알수 있고 일반적으로 finesse, low, vollye를 선두로 normal, super 의 순으로 득점 성공률이 나타나는 걸 볼수 있습니다. 또한, Out_Penalty와는 비교되는 부분이 있는데 우선 마찬가지로 finesse, normal을 가장 많이 시도하고 있고 시도한 횟수 대비 성공률도 finesse는 약 2배, normal은 약 3배 가량 높은 득점 성공률을 확인해볼 수 있었습니다.

(In_Penalty에서도 free-kick이 존재하는데 이는 간접프리킥으로 판단됩니다.)

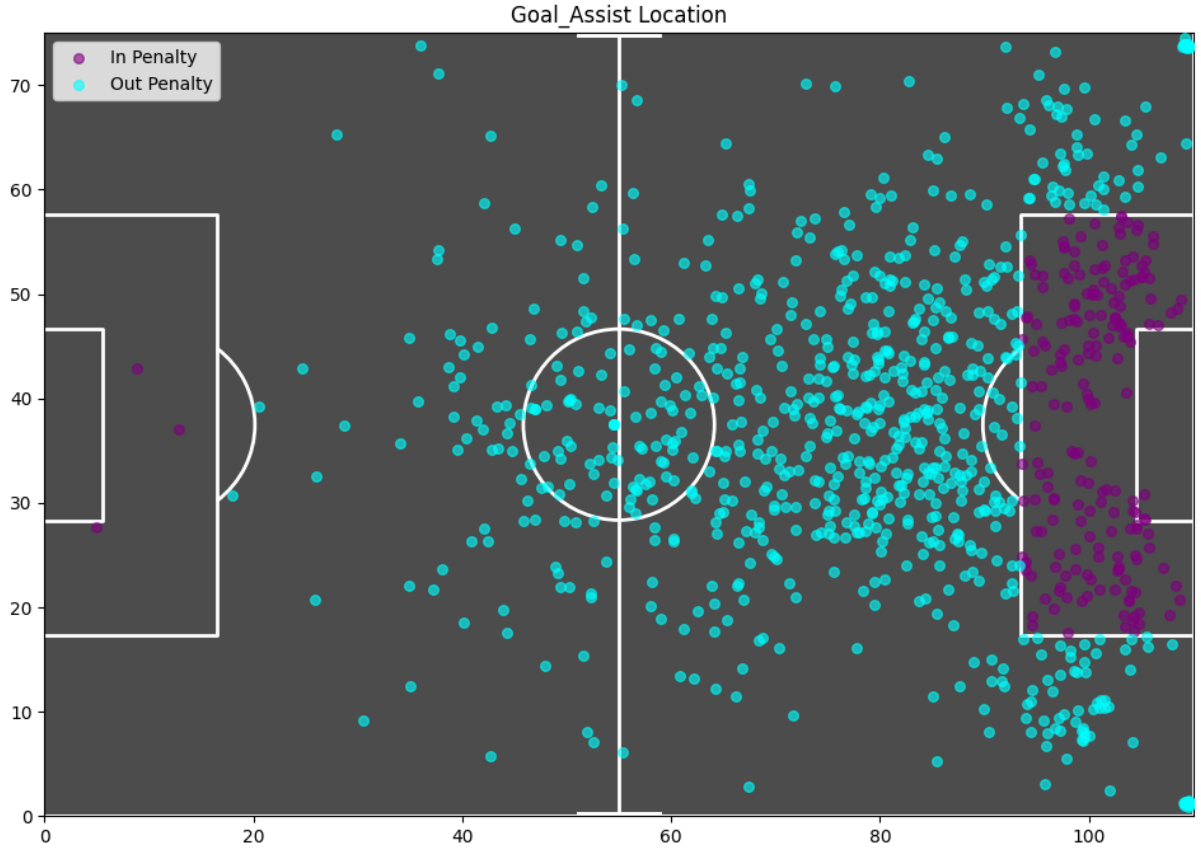
이를 바탕으로 가장 선호되는 슈팅 타입이 어떤 위치에서 득점에 성공되는지에 대한 시각화를 진행하였습니다.



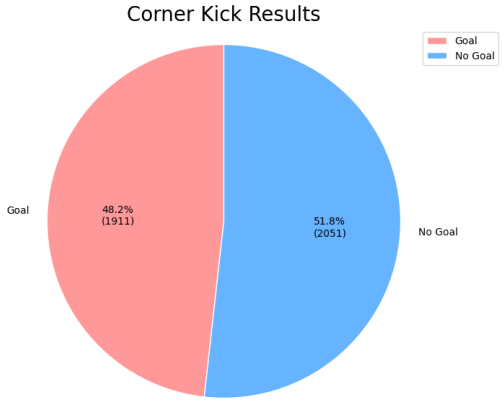
슈팅 타입별로 구역을 가지는 모습을 확인할 수 있습니다.

3.2.3 패널티 구역 내에서 슛이 많은데 중요한 세트피스 중 하나인 코너킥에 의한 어시스트로 득점이 어느 정도의 비율을 가지는지 확인해 보았습니다.

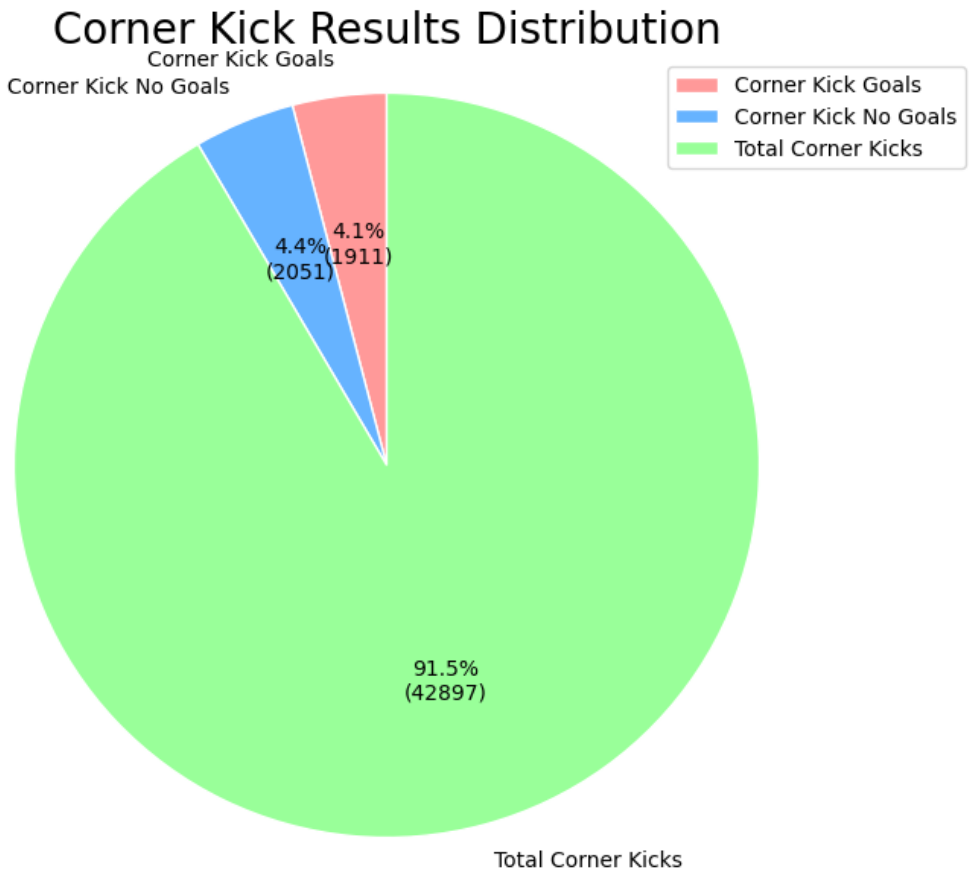
다음은 어시스트가 득점으로 이어진 경우의 어시스트 좌표를 시각화한 것입니다.



코너킥이 어시스트가 된 경우는 3962회이며 이에 대한 득점 비율은 다음과 같습니다.



약 48%의 성공률을 보이고 있으며, 코너킥이 어시스트가 된 경우가 아닌 총 코너킥 수 대비 어느 정도의 득점 성공률을 보이는지 확인해보았습니다.



전체 코너킥 대비 코너킥이 어시스트로 이어진 경우는 약 8.5%이며, 득점으로 이어진 코너킥 어시스트는 약 4.1%의 득점 성공률을 보이고 있으며, 이는 아주 근소한 수치임을 알 수 있습니다.

결과적으로 상황에 맞는 슈팅을 하는 것이 효율적인데 패널티 구역 내에서는 normal, low, volley 를 시도하는 것이 좋고, 패널티 구역 밖에서는 lob, KNUCKLE 를, 패널티 구역 내외를 구분하지 않는다면 super, finesse가 가장 좋은 선택이 될 것 입니다.

추가적으로 패널티 구역 내에서 슛이 많이 발생하지만 코너킥은 큰 영향을 미치지 못하니 큰 기대를 갖지 않아야하며 오히려 상대에게 역습을 제공할 수 있으니 주의할 필요가 있습니다. 그리고 장지현 해설위원의 말을 인용하자면 '두드리면 열린다.' 라는 마인드를 가지고 주저없이 슛을 하는 것이 좋겠습니다.

3.2.4 Goal 예측 모델 - Random Forest

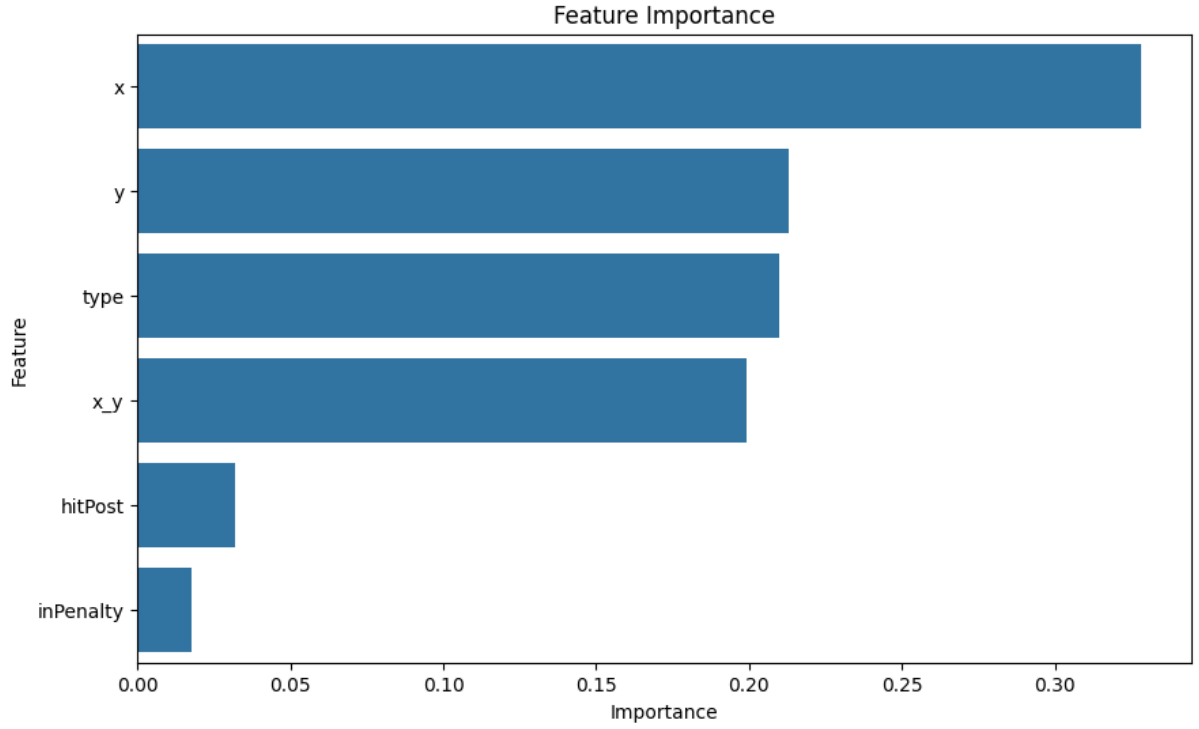
패널티 구역 내외, 슈팅 타입에 따른 득점 성공률을 예측하기 위해 예측 모델 분석을 진행하였습니다.

특성과 라벨을 분리하고 학습60%, 테스트20%, 예측 20%로 데이터를 구분하였습니다. 랜덤 포래스트 분류모델을 사용하고 클래스가 불균형할 때 각 클래스에 가증치를 주어 학습이 편향되지 않도록 하였으며, 그리드 서치로 하이퍼파라미터를 조정하여 데이터 샘플링하여 전체 데이터 셋이 아니라 데이터의 일부로 빠르게 최적의 하이퍼파라미터를 찾도록 하여 최종 모델을 학습 시킵니다.

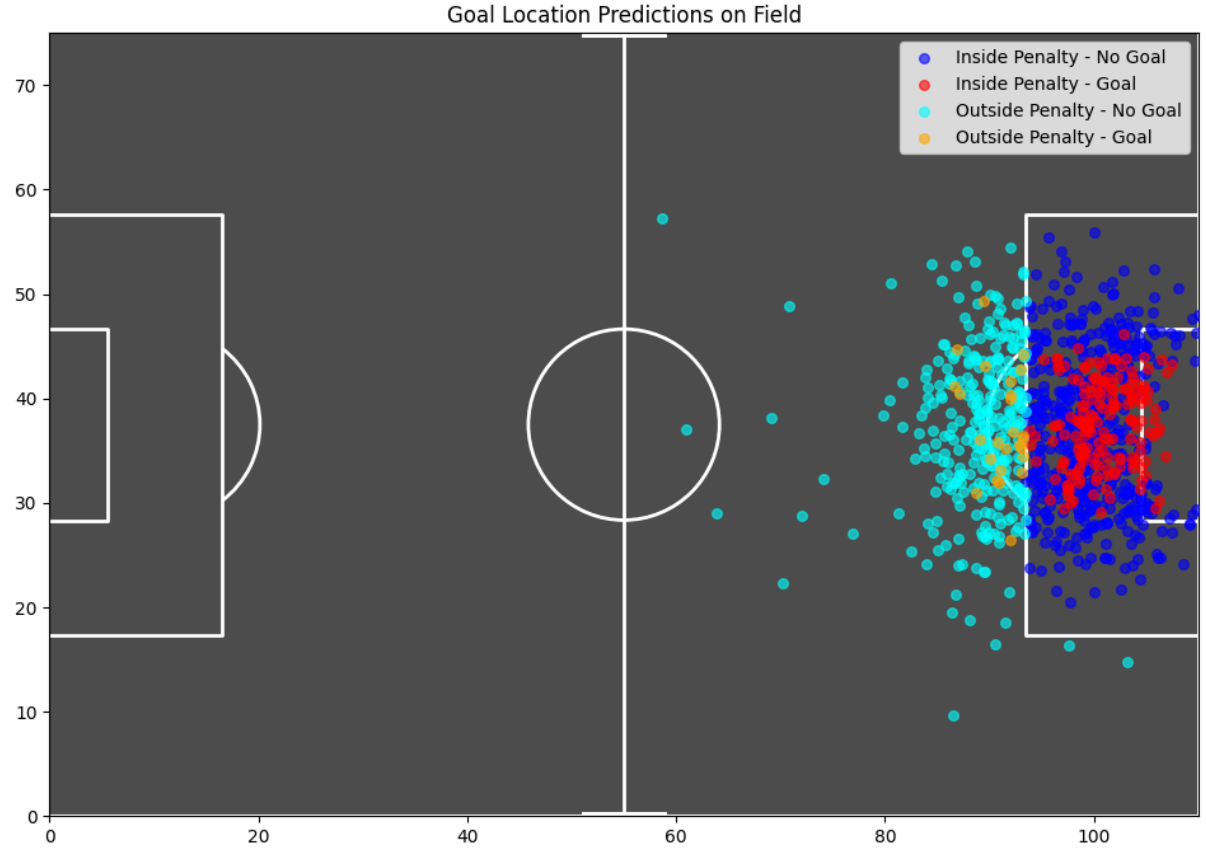
정확도(accuracy), 정밀도(precision), 재현율(recall)에 대한 결과는결과는 다음과 같습니다.

```
Random Forest - Accuracy: 0.72, Precision: 0.50, Recall: 0.35
```

피처 중요도는 다음과 같습니다.



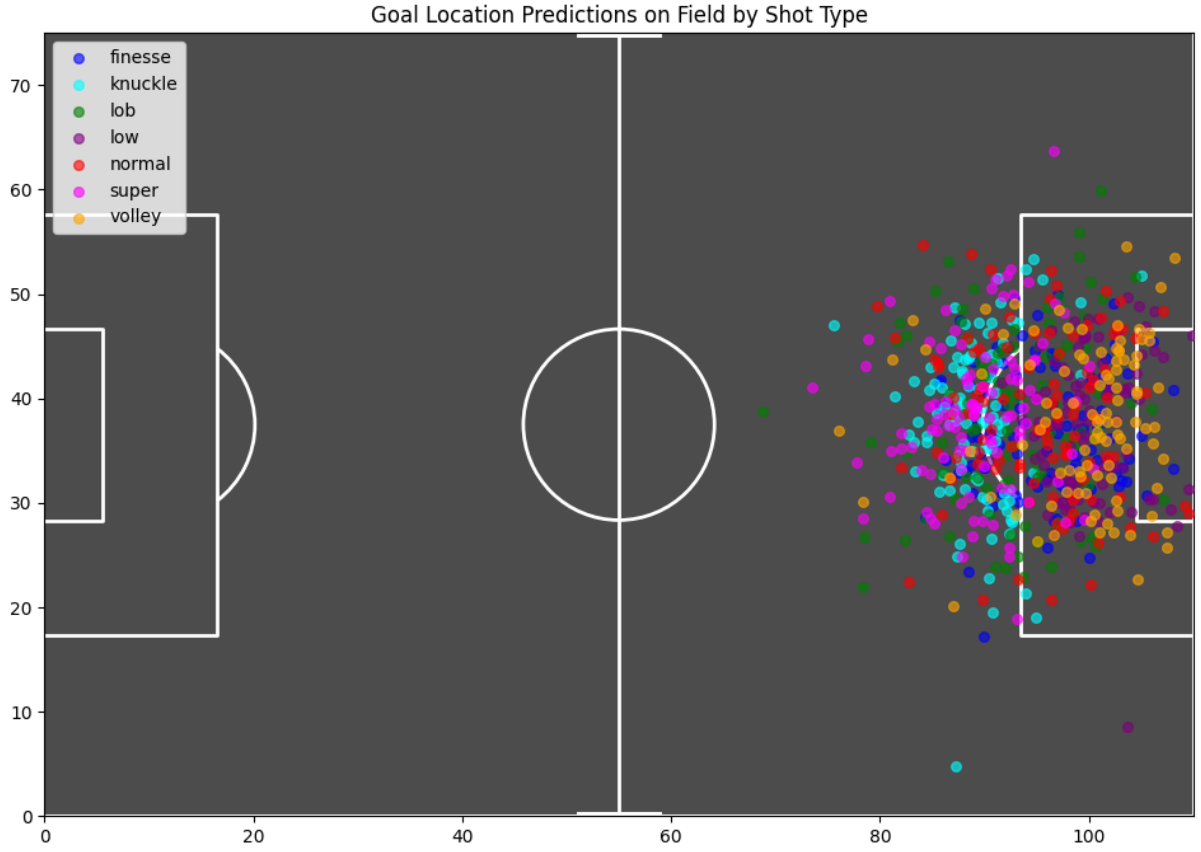
예측 결과를 바탕으로 패널티 구역 내외 슛 좌표를 시각화하였습니다.




```
골 비율: 21.00%
노골 비율: 79.00%
패널티 에어리어 안 비율: 67.20%
패널티 에어리어 밖 비율: 32.80%
```

실제 총 슈팅 대비 득점 비율인 28.41%와 7.4% 정도의 차이가 나는 것을 확인할 수 있습니다.

예측결과에 대해 슛 타입별 시각화도 진행하였습니다.




```
골 비율: 19.75%
노골 비율: 80.25%
finesse - 슈팅 시도: 13574, 득점: 3413
finesse - 골 비율: 25.14%, 노골 비율: 74.86%
knuckle - 슈팅 시도: 359, 득점: 5
knuckle - 골 비율: 1.39%, 노골 비율: 98.61%
lob - 슈팅 시도: 286, 득점: 26
lob - 골 비율: 9.09%, 노골 비율: 90.91%
low - 슈팅 시도: 2824, 득점: 968
low - 골 비율: 34.28%, 노골 비율: 65.72%
normal - 슈팅 시도: 12191, 득점: 1878
normal - 골 비율: 15.40%, 노골 비율: 84.60%
super - 슈팅 시도: 1215, 득점: 71
super - 골 비율: 5.84%, 노골 비율: 94.16%
volley - 슈팅 시도: 1435, 득점: 367
volley - 골 비율: 25.57%, 노골 비율: 74.43%
```

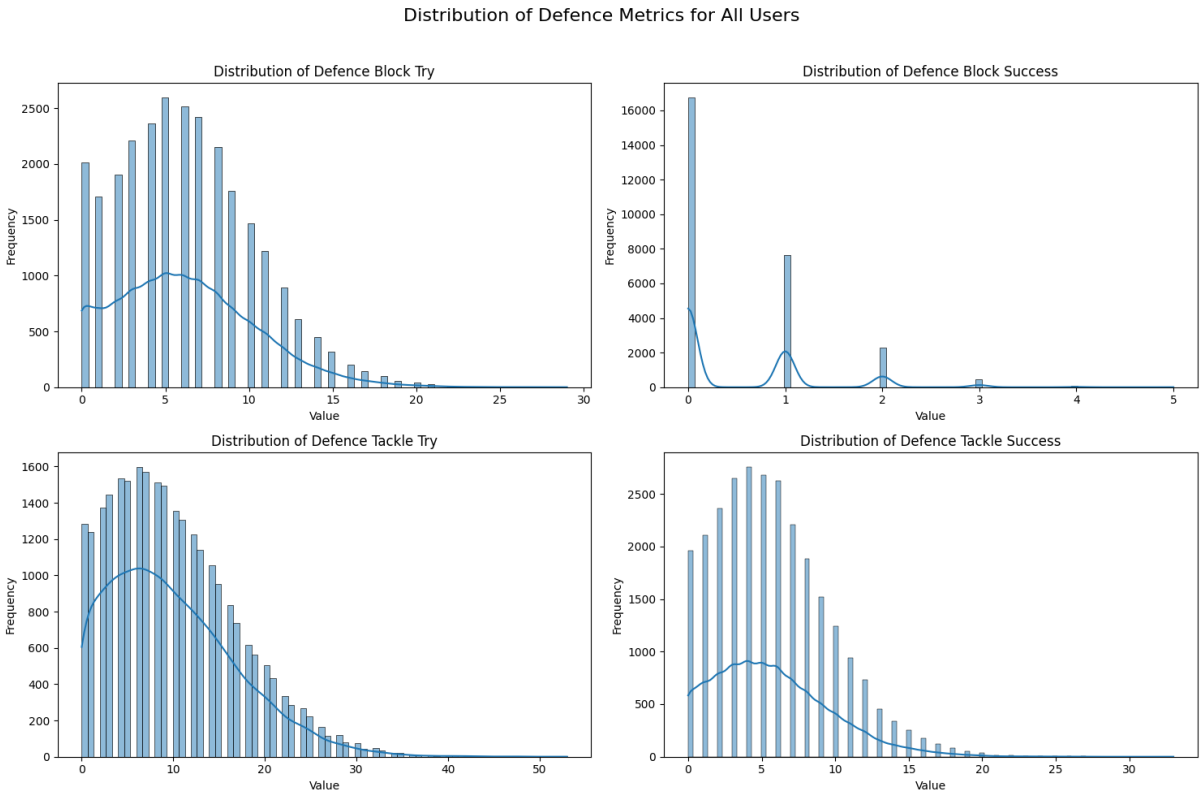
실제 득점 성공율과는 차이가 있지만 슈팅 타입별 위치는 대체로 비슷한 것을 확인할 수 있습니다.

3.3 희령 가설

축거 경기에서 수비수들의 퍼포먼스는 팀의 승패에 중요한 영향을 미칠수 있습니다. 특히, 태클 성공률과 압박 지표 등은 수비수들의 실력을 평가하는 주요 지표로 활용됩니다. Defence

부분의 4가지 컬럼인 태클 시도(tackleTry), 태클 성공(tackleSuccess), 블록 시도(blockTry), 블록 성공(blockSuccess) 컬럼을 바탕으로, 수비 능력이 팀 승률에 미치는 영향을 탐구하였습니다.

3.3.1 수비지표 EDA

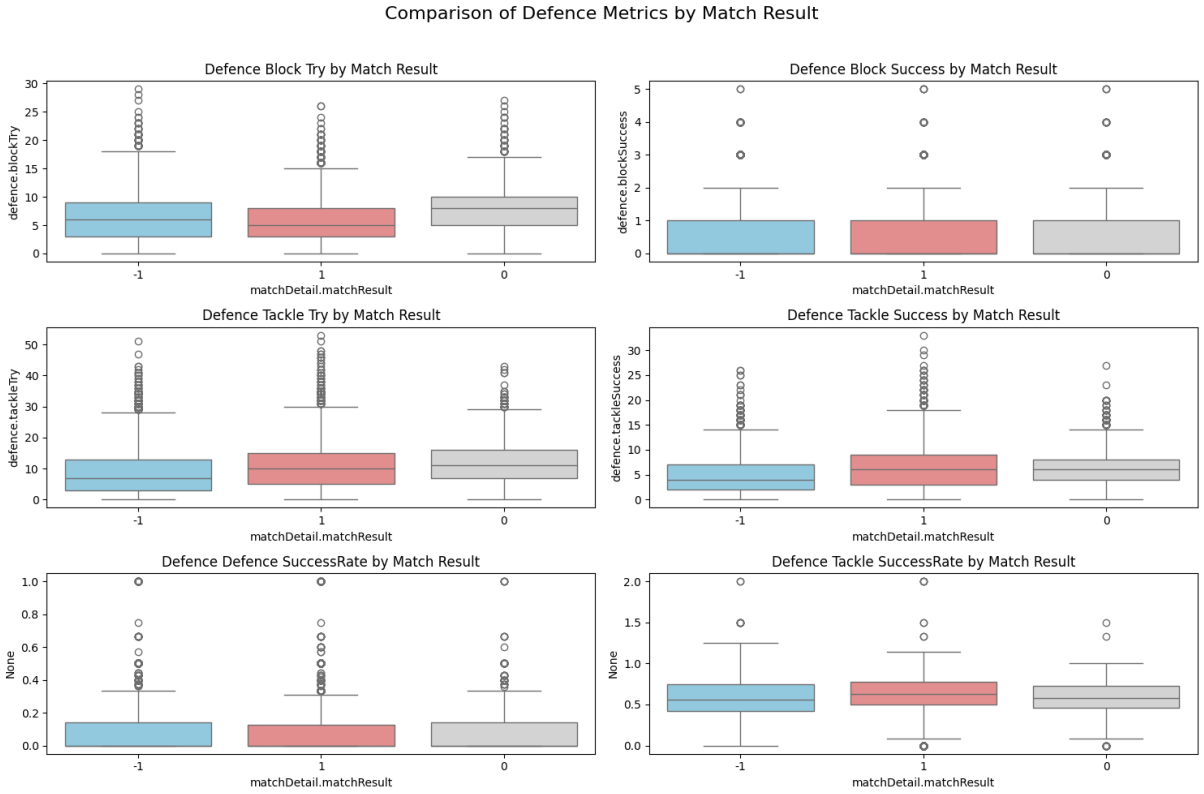


[그림 3.3.1 blockTry, blockSuccess, tackleTry , tackleSuccess 분포]

유저들은 경기당 평균 6.18회의 블록 시도를 합니다. 분포는 오른쪽 꼬리가 긴 비대칭 분포를 보이며, 대부분의 블록시도 횟수는 10회 이하입니다. 블록 성공 횟수는 평균 0.51회로, 매우 낮은 성공확률을 보입니다. 분포는 대부분 0또는 1회에 집중되어 있습니다. 태클 시도 횟수는 평균 9.96회로, 블록시도에 비해 더 빈번하게 발생하였고 태클 성공 횟수는 평균 5.66회로, 태클 시도 대비 비교적 높은 성공률을 보였습니다.

유저들의 블록과 태클 지표의 분포를 통해 블록은 성공 확률이 매우 낮고 태클은 비교적 높은 성공률을 보이는 것을 확인할 수 있었습니다. EDA한 결과를 기반으로 가설을 세워 분석을 진행하였습니다.

3.3.2 블록과 태클의 시도, 성공횟수가 높을수록 승률이 높을 것이다.



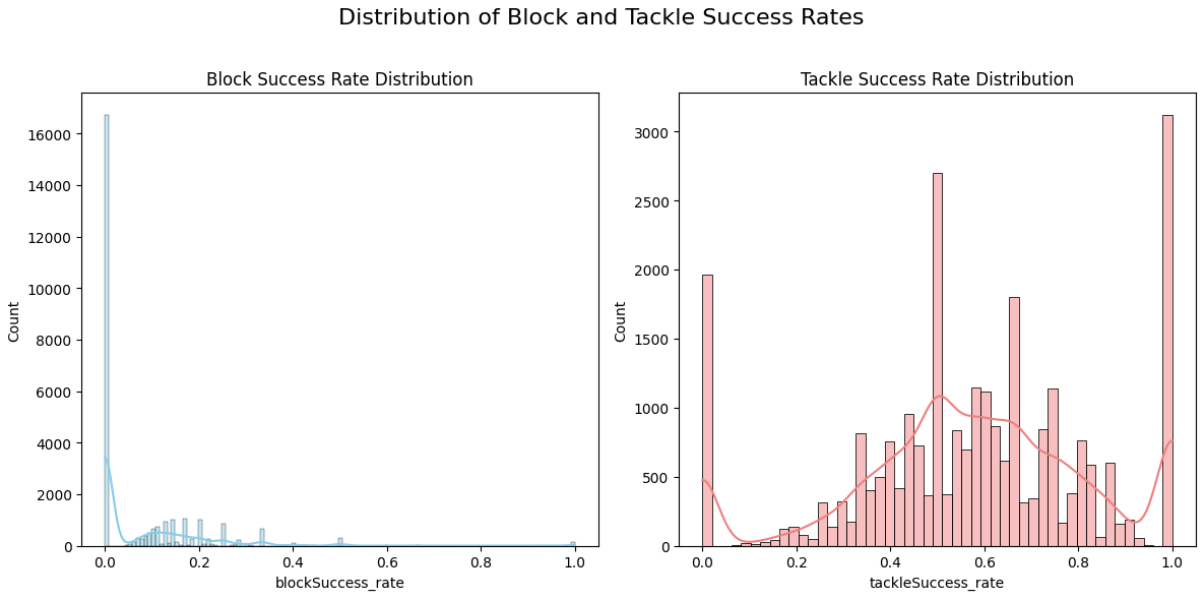
[그림 3.3.2 태클, 블락- 매치Result 관계 시각화]

위의 가설을 토대로 유저들의 태클 및 블록 지표가 경기 결과(승:1, 패:-1 무:0) 에 미치는 영향을 분석하였습니다. 블록 시도의 경우, 무승부를 기록한 유저들이 평균 7.9회로 가장 많았스며, 패배한 유저들이 평균 6.3회 , 승리한 유저들이 평균 5.5회로 낮았고, 반면 블록 성공 횟수는 거의 0에 가까운 값으로, 경기 결과에 따른 평균 횟수 차이가 거의 없었습니다. 이러한 결화는 블록 시도 횟수가 승률에 부정적인 영향을 미친다는 것을 시사합니다. 즉, 블록 시도가 많을수록 경기 결과에 부정적인 영향을 미칠 수 있으며, 높은 실패 확률을 고려할 때 블록 시도를 줄이는 것이 좋을 것으로 보입니다.

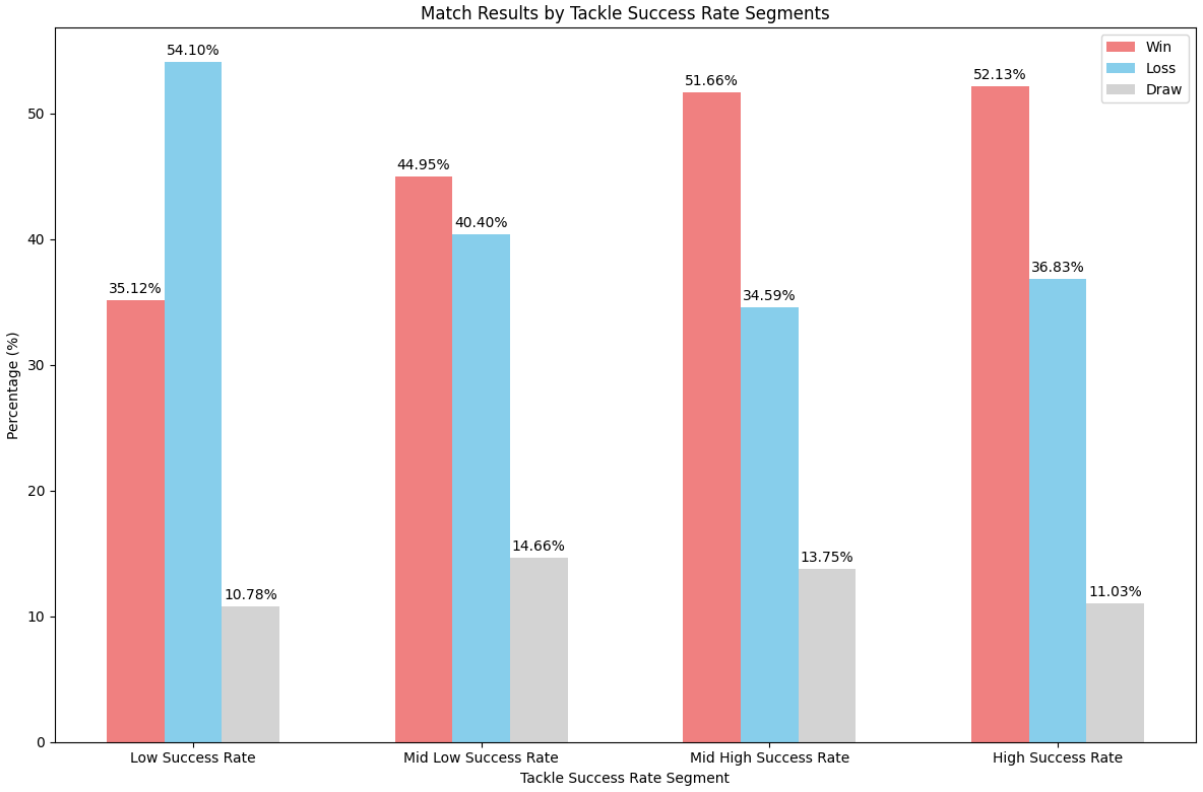
결론적으로, 블록 시도 및 성공 횟수가 높을수록 승률이 높아질 것이라는 가설은 틀린 것으로 확인 되었습니다.

태클 시도의 경우, 무승부를 기록한 유저들이 평균 11.6회로 가장 많았으며, 승리한 유저들이 평균 10.6회, 패배한 유저들이 평균 8.7회로 나타났습니다. 태클 성공 횟수는 무승부를 기록한 유저들이 평균 6.4회 승리한 유저들이 평균 6.3회,패배한 유저들이 평균 4.6회로 나타났습니다. 이는 태클 시도가 경기 결과에 긍정적인 영향을 미칠 수 있음을 시사하며, 태클을 시도하면서 성공률을 높이는 것이 승리에 기여할 수 있다는 것을 유추하여 태클 성공률을 박스플롯을 시각화 한결과 승리한 유저들의 태클 성공률이 높은 것으로 확인되었습니다.

3.3.3 태클 성공률이 경기결과에 미치는 영향



[그림 3.3.3-1 블록 성공률 , 태클 성공률 분포]



[그림 3.3.3-2 태클 성공률과 경기 결과 시각화]

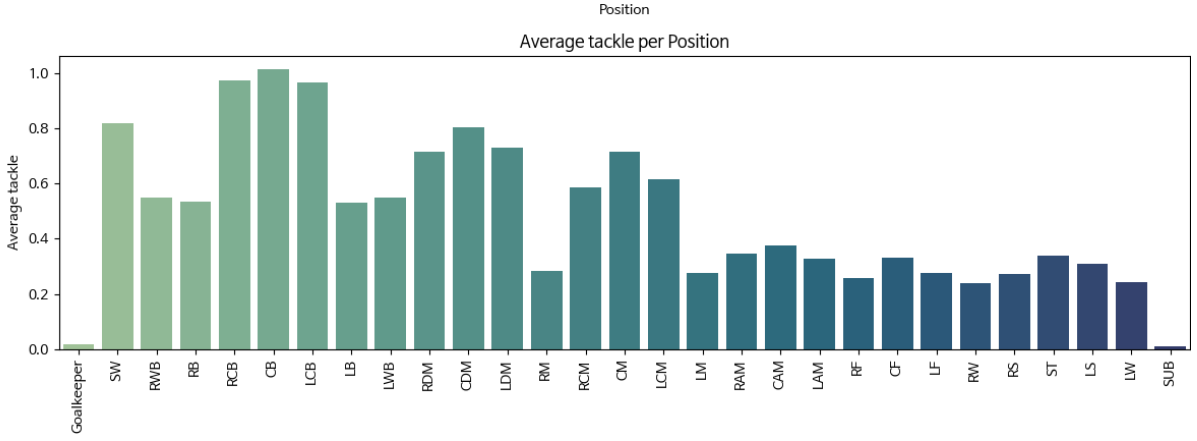
Try 컬럼과 Success 컬럼을 활용해 블록, 태클 각각 SuccessRate 컬럼을 만들어 분석하였습니다. [그림 3.3.3-1]은 전체 유저들의 블록 성공률과 태클 성공률 분포를 나타냅니다. 블록 성공률의 평균은 0.073, 중앙값도 비슷한 값으로 전반적으로 성공률이 매우 낮음을 보입니다. 태클 성공률은 평균 0.57, 중앙값도 0.57로 나타나 전반적으로 많은 플레이어가 태클 시도에서 절반이상(57%)을 성공하는 경향을 보였습니다.

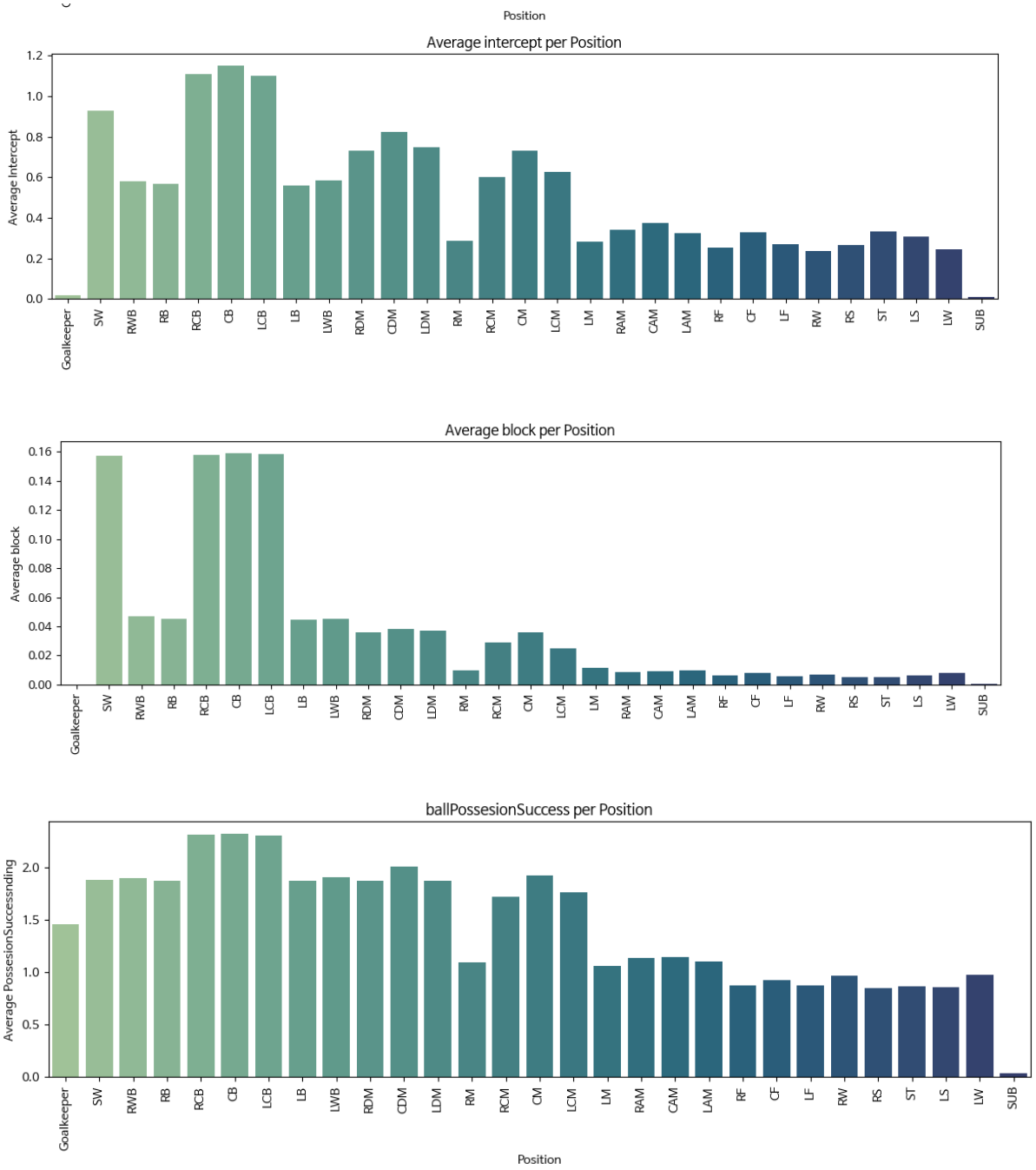
블록 성공률이 거의 0에가까운 점을 고려해 블록성공률은 분석에서 제외하고, 태클 성공률을 기준으로 사분위수 기반 세그먼트를 설정하여 상위 그룹, 중상위 그룹, 중하위 그룹, 하위그룹의 4가지 그룹으로 나누어 분석을 진행하엿습니다. 분석 결과, 태클 성공률이 높은 상위 그룹일수록 승리 비율이 35.1% 에서 52.13% 까지 증가하는 경향을 보엿고 태클 성공률이 낮은 그룹일수록 패배 비율이 36.8% 에서 54.1% 까지 증가하는 경향을 보였습니다.

결론적으로 첫째, 블록 시도는 성공률이 낮을 뿐만 아니라 시도를 많이 할수록 패배할 가능성이 높아집니다. 따라서 블록 시도를 적게 사용하는 것이 경기결과에 유리할 것으로 판단됩니다.

둘째, 태클의 경우 겅공률이 경기 결과에 중요한 영향을 미치는 것으로 나타났습니다. 이러한 결과는 태클이 경기 흐름을 끊고 이를 통해 경기 방향을 유리하게 가져오는 중요한 역할을 할 수 있다는 점을 시사합니다. 따라서 태클 성공률이 높은 선수를 수비수로 배치하는 것이 팀의 승률을 높이는 데 긍정적인 효과를 미칠 것으로 보입니다.

3.3.4 수비수들의 주요 스탯 알아보기



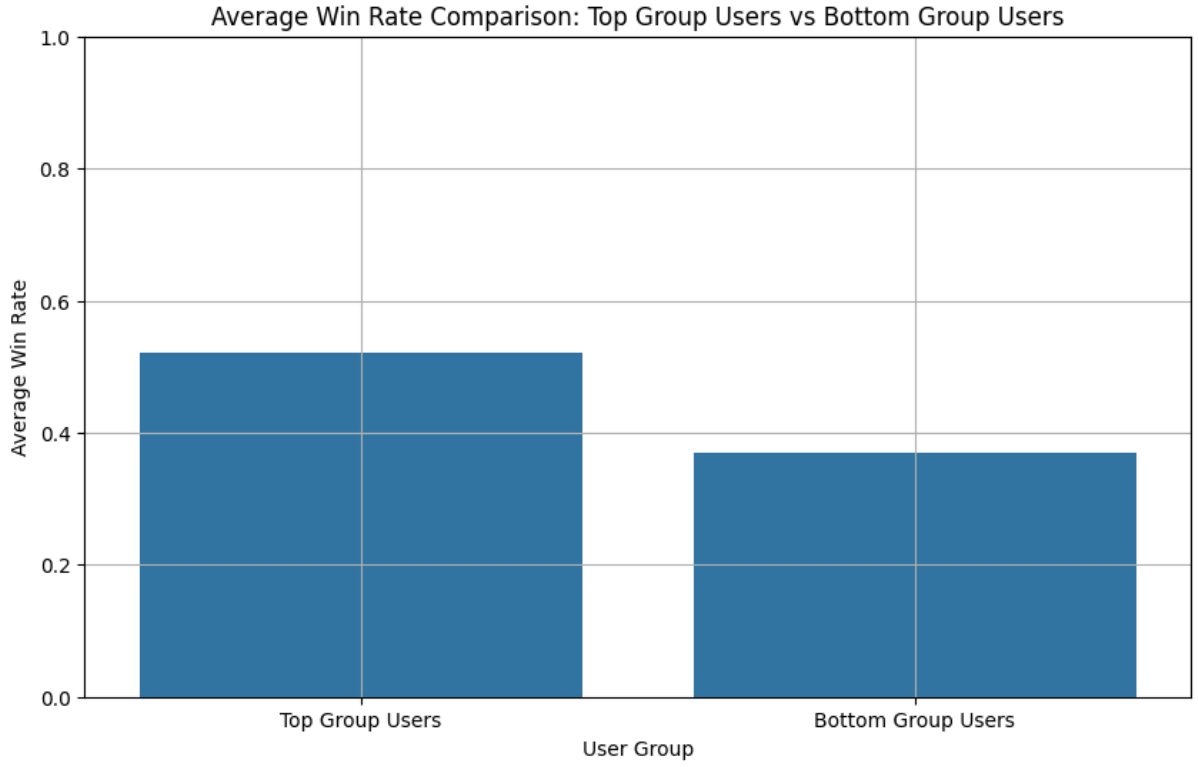


[그림 3.3.4 선수 포지션별 평균 분포]

유저들이 상요한 선수들의 상세 테이블을 활영하여 포지션 중 수비수 관련 지표를 분석하고자 하였습니다. FC온라인에서 활용되는 29가지 포지션들의 중 SW 부터 SWB 까지가 수비수 포지션에 해당함을 확인하였습니다. 상관관계 계수를 확인해 여러 지표들을 비교해보자 하였고 수비수들의 주요 스탯을 분석한 결과 ballPossessionSuccess(볼 소유 성공수), intercept(입터셉트 수), tackle(태클 성공 수), block(블록 성공 수) 등이 다른 포지션에 비해 평균적으로 높은것으로 나타났습니다. 그러나 block은 0에 가까운 데이터가 많아 분석에서 제

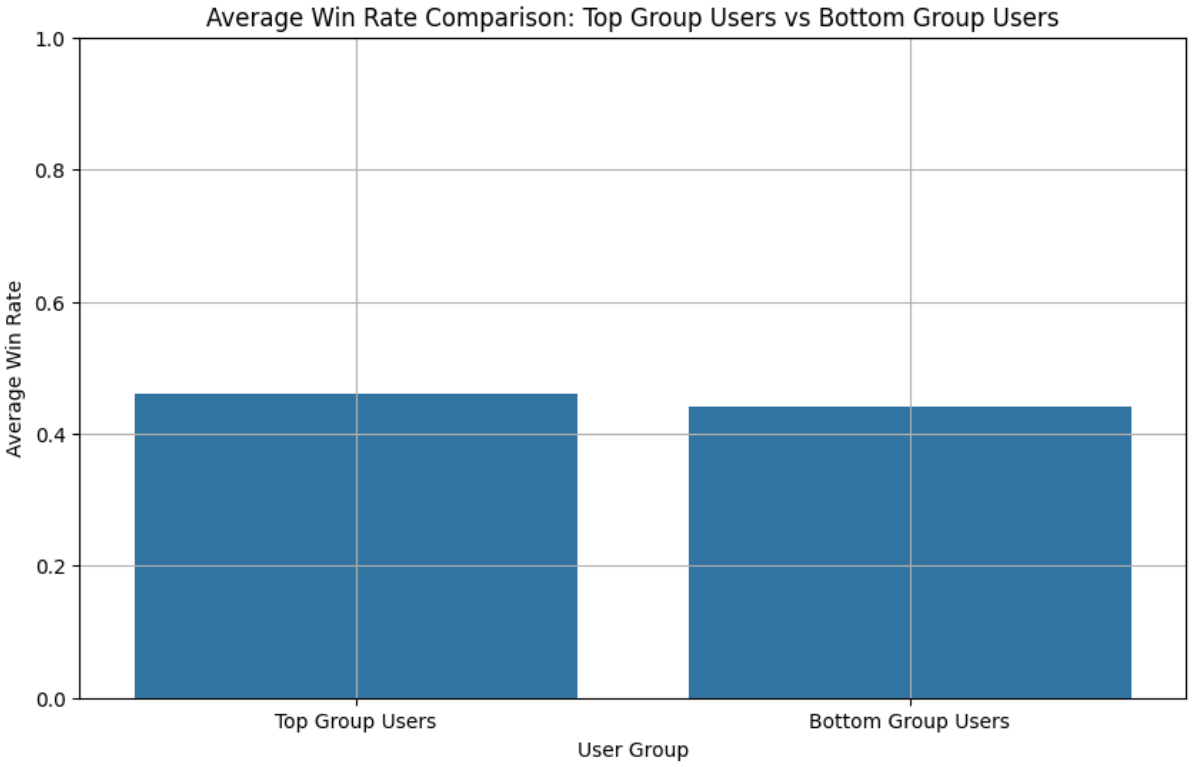
외하였으며 , 나머지 3개의 컬럼이랑 선수 평가 지표인 spGrade 4가지 지표로 사분위수 25% 이상을 능력치가 좋은 top수비수, 하위 25% 를 bottom 수비수로 지정하여 매치 결과 승률을 확인해 보았습니다.

3.3.5 수비수 주요 스탯과 매치 결과 승률 확인



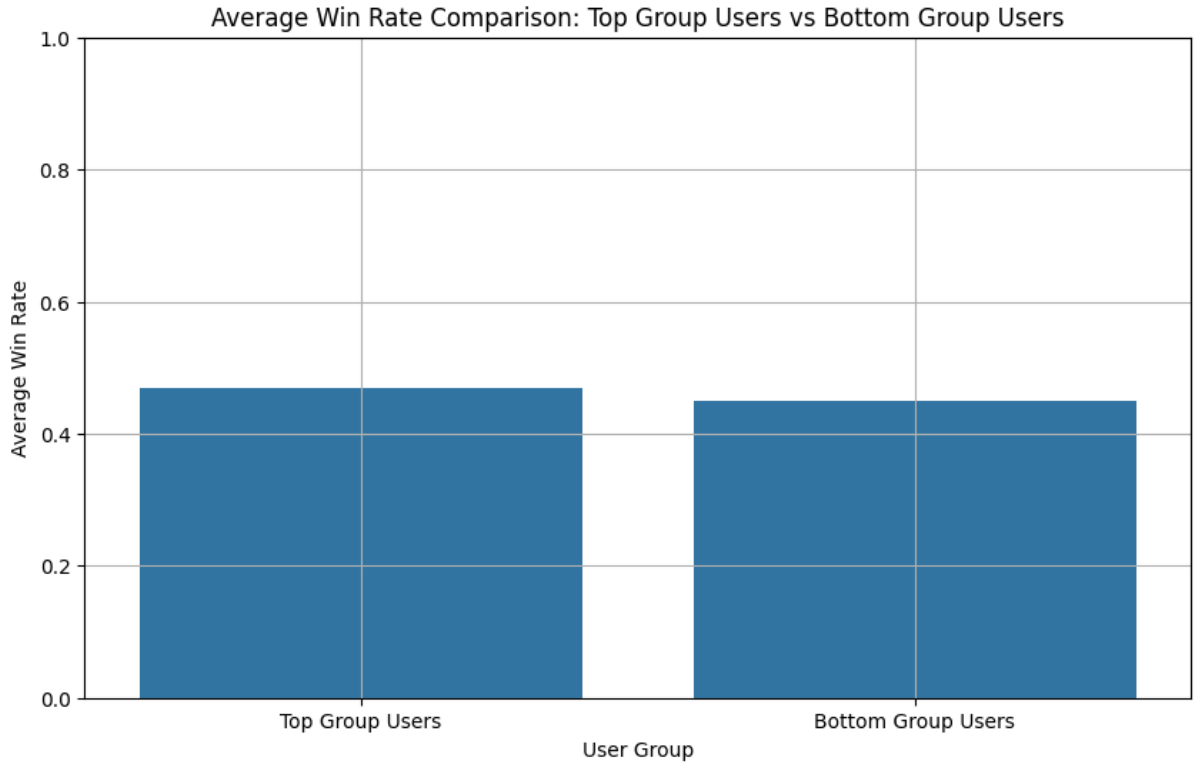
[그림 3.3.5-1 ballPossesionSuccess, tackle, intercept, spRating 지표 그룹별 수비수들 승률 차이]

[그림 3.3.5-1]는 top 그룹과 bottom 그룹 수비수들을 사용하는 유저들의 평균 승률을 비교한 결과를 나타냅니다. 분석 결과, Top 스탯 수비수들을 사용하는 유저들의 평균 승률은 0.51인 반면 , bottom 스탯 수비수들을 사용하는 유저들의 평균 승률은 0.38로 나타났습니다. 이는 수비수의 주요 지표인 ballPossessionSuccess, intercept, tackle, spGrade 이높은 수비수를 기용한 팀이 그렇지 않은 팀보다 높은 승률을 기록했음을 시사합니다.



[그림 3.3.5-2 ballPossessionSuccess, intercept, tackle, block 지표 그룹별 수비수들 승률 차이]

[그림 3.3.5-2] 는 수비지표를 ballPossesionSuccess, tackle, intercept, block 4가지로 지정해 top그룹과 bottom 그룹 수비수들을 사용하는 평균 승률을 시각화한 결과입니다. top 스탯 수비수들의 평균 승률은 0.46 , bottom 스탯 수비수들의 평균 승률은 0.44 로 나타났습니다.



[그림 3.3.5-3 ballPossesionTry, passTry, intercept, tackle 지표 그룹별 수비수들 승률 차이]

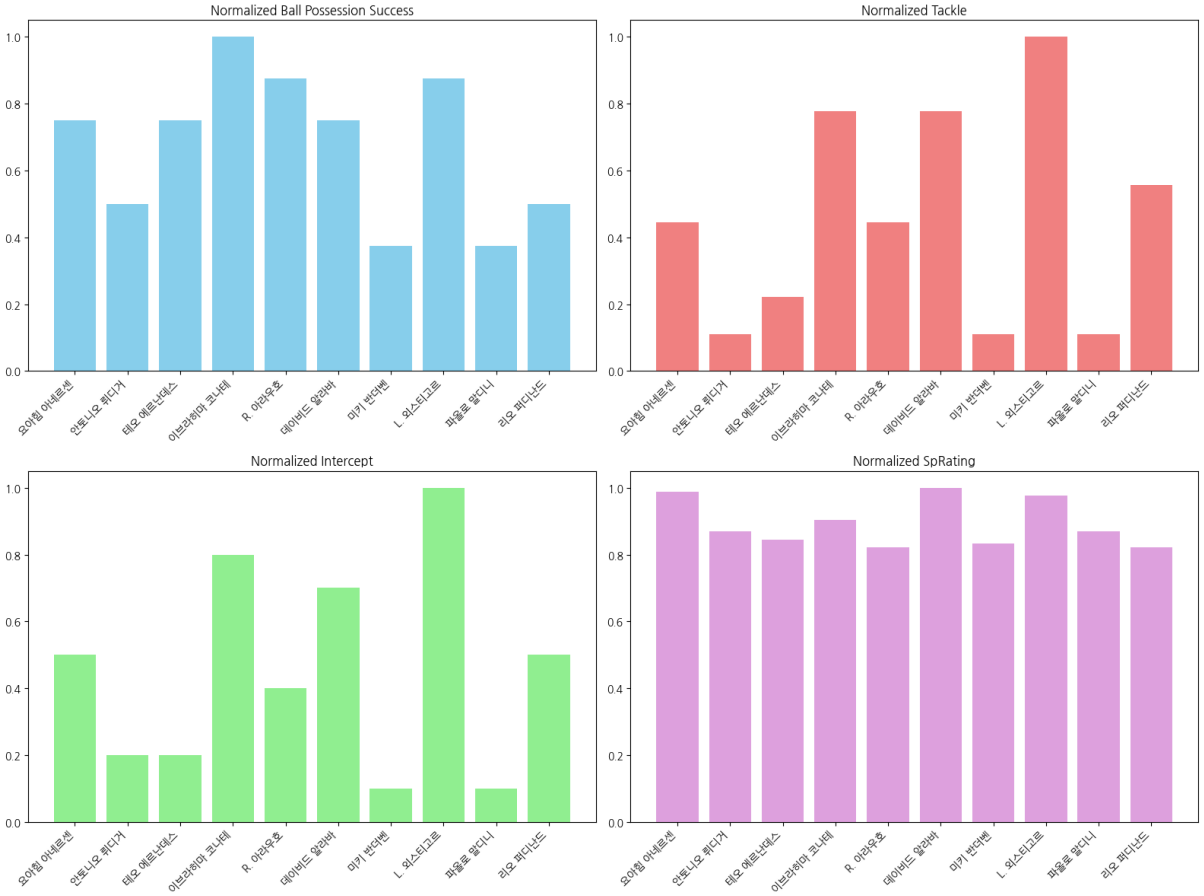
위의 그림은 ballPossesionTry (볼 소유 시도) passTry (패스시도) intercept, tackle 을 주요지표로 선정해 top 그룹과 bottom 그룹 수비수들 평균 승률을 살펴보았고 top스탯 수비수들의 평균 승률은 0.47, bottom스탯 수비수들의 평균 승률은 0.45 로 나타났습니다.

수비수들의 주요스탯과 경기 승률을 확인해본 결과 ballPossesionSuccess, intercept, tackle, spRating 4가지 컬럼을 기준으로 선정했을때 경기 승률에 긍정적인 영향을 미친다는 결론을 도출 할 수 있었습니다.

3.3.5 Top 스탯 수비수들의 분석

	name	className	counts	ballPossesionSuccess	tackle	intercept	spRating
0	요아힘 아네르센	23HW (23 Hard Worker)	3	6.0	4.0	5.0	8.3
67	안토니오 뤼디거	SPL (Spotlight)	2	4.0	1.0	2.0	7.3
73	테오 에르난데스	24 TOTY (24 Team Of The Year)	2	6.0	2.0	2.0	7.1
95	이브라히마 코나테	22 UCL (22 UEFA Champions League)	2	8.0	7.0	8.0	7.6
93	R. 아라우호	23 UCL (23 UEFA Champions League)	2	7.0	4.0	4.0	6.9
88	데이비드 알라바	GR (Golden Rookies)	2	6.0	7.0	7.0	8.4
87	미키 반더벤	23 LIVE (23 LIVE)	2	3.0	1.0	1.0	7.0
10	L. 외스티고르	23 UCL (23 UEFA Champions League)	2	7.0	9.0	10.0	8.2
9	파울로 말디니	DC (Decade)	2	3.0	1.0	1.0	7.3
191	리오 퍼디난드	ICON (ICON)	1	4.0	5.0	5.0	6.9

[그림 3.3.5-1 top10 수비수]



[그림 3.3.5-2 top10 선수들 주요 지표 시각화]

유저들이 사용한 수비수들의 주요 지표를 통해 top10 선수들의 목록과 [그림 3.3.5-2] 는 상위선수들의 앞서 선정한 4가지의 주요지표를 시각화한 결과물입니다. 이러한 지표들을 바탕으로, 특정한 한 가지 스탯에 치중하지 않고, 다양한 지표에서 균형 3잡힌 수비 능력을 보유한 수비수들을 사용하고자 하는 안정적인 플레이 전략을 구사하는 유저들에게 선수 카드를 추천함으로써 , 보다 전략적인 팀 구성을 위한 핵심 카드로 활용 될 수 있음을 시사합니다 .

3.4 수민 가설

3.4.1 도전적인 패스를 많이 시도한 사람들은 승률이 높았을까?

가설 배경

패스는 여러가지 종류가 있습니다. 축구를 보면 그 중에 상대방에게 좋은 패스를 공급해주는 플레이메이커들이 게임에 끼치는 영향은 대단합니다. 대한민국 축구 경기에서 이강인 선수가 축구를 하는 경우만 보더라도 그의 존재가 얼마나 대단한 지 느낄 수 있습니다. 이러한 플레이 메이커들은 경기를 뒤집을 도전적인 패스를 시도하게 되고, 그 패스가 이어지면 골을 넣는 찬스로 이어지곤 합니다. 골을 넣는다면, 경기에서 이길 확률은 당연히 높아집니다. 그렇다면 축구와 비슷한 환경을 구사하는 FC 온라인에서도 도전적인 패스를 시도했을 때 도전적인 패스를 많이 한 사람들은 승리할 확률이 높았을 것이라는 점을 가설로 삼고 분석을 진행하였습니다. 따라서, 본 분석은 FC 온라인에서 게임의 승리를 원하는 사람들에게 패스 전략을 제시해 줄 것입니다.

가설의 중요성

축구 게임에서 도전적인 패스는 공격 전술의 핵심 요소로 작용합니다. 이강인 선수와 같은 플레이메이커들은 뛰어난 시야와 패스 능력으로 경기를 변화시키며, 그들의 패스가 성공하면 직접적인 골 찬스로 이어질 가능성이 큽니다. 실제 축구에서 이러한 전술이 성공적임을 확인할 수 있습니다.

FC 온라인과 같은 축구 시뮬레이션 게임에서도 이러한 전략을 적용함으로써 플레이어들은 더 높은 승률을 기대할 수 있습니다. 도전적인 패스를 시도하는 것은 리스크를 수반하지만, 성공할 경우 팀의 공격력과 득점 가능성을 크게 높입니다. 따라서, 이 가설은 플레이어들이 승률을 높이기 위한 구체적인 전술을 제공함으로써 게임 플레이의 질을 향상시키는 데 중요한 역할을 할 것입니다.

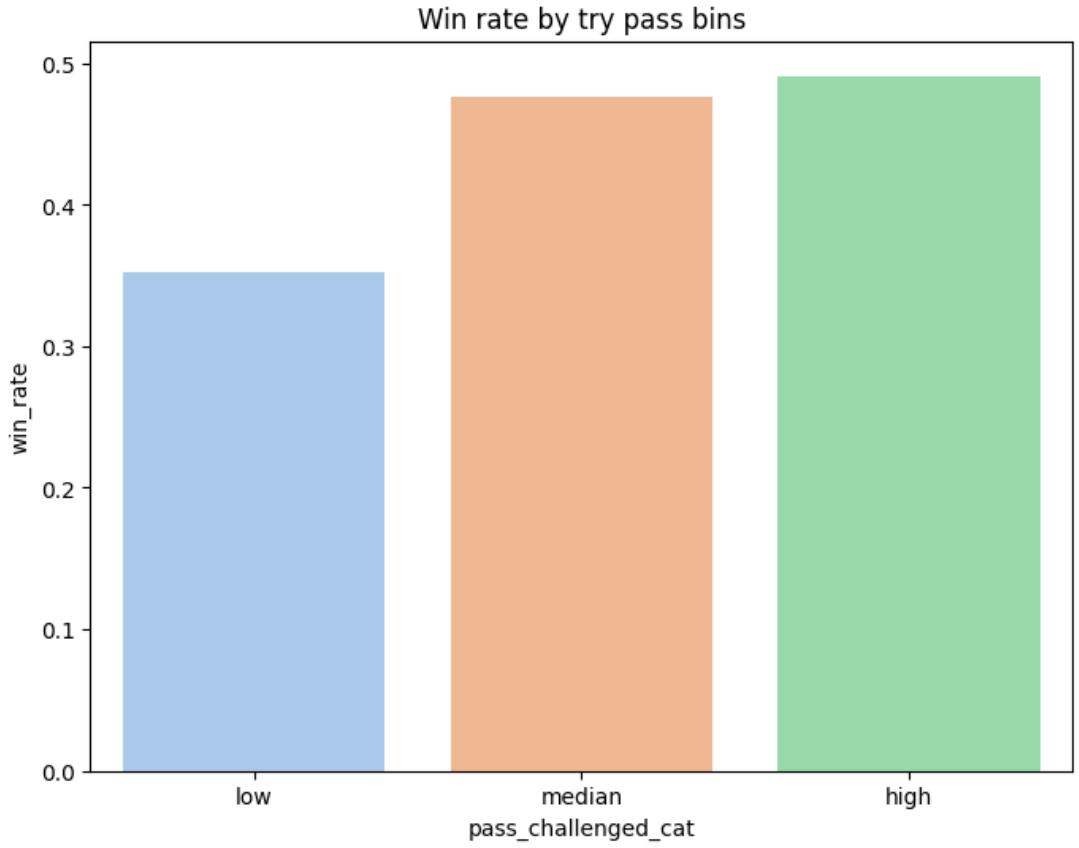
정의 요약

- **도전적인 패스의 정의:** 일반 패스의 경우는 끊길 위험이 거의 없는 패스를 말하는 즉, 숏패스와 롱패스를 의미하고 도전적인 패스의 정의는 상대 선수에게 차단될 확률이 높은 패스거나, 상대의 빈공간을 향해 찔러주는 패스로 정의하였습니다.
- **도전적인 패스를 많이 하는 사람의 기준:** FC 온라인 사이트 내 통계를 활용하였습니다. 7월 첫째 주 기준, **전반적인 전체 패스의 평균 횟수는 106회, 스루 패스는 19회, 평균 드라이빙 땅볼 패스는 3.5회, 로빙 스루 패스는 0.9회로, 스루패스 + 드라이빙 땅볼 패스 + 로빙 스루 패스/전체 패스 횟수를 하면 평균 도전적인 패스 비율은 22%임을 알 수 있습니다.** 따라서, 평균에 +- 30%를 적용하여, 도전적인 패스 비율이 16% ~ 28% 사이일 때, 도전

적인 패스를 평균적으로 하는 사람, 16% 미만을 도전적인 패스를 선호하지 않는 사람, **28% 초과를 도전적인 패스를 많이 하는 사람이라고 정의하였습니다.**

- **스루패스를 가장 많이 성공시킨다의 기준: FC 온라인 공식 홈피(07/12)를 기준으로 성공률의 평균치는 85%**입니다. 이를 활용하고, 수집한 데이터에서 사분위수로 스루패스 성공률의 기준을 나누어 **상위 66% 이상일 경우, 스루패스를 많이 성공시켰다, 33~66% 구간을 중간정도 성공 했다, 33% 이하의 구간을 적게 성공시켰다라고 분류하였습니다.**
- **패스 성공률이 높다의 기준: FC 온라인에서 패스 성공률의 평균은 90%입니다. 하지만 수집된 데이터를 기준으로 박스플롯을 그려본 결과, 90%인 경우, 상위 75%의 구간이기 때문에, 90% 이상을 패스 성공률이 높다라고 설정하였습니다.**

1. **도전적인 패스 시도 당 승률의 차이가 있을까?**



도전적인 패스 시도 당 승률을 시각화한 차트입니다. low(16% 미만), median (16% ~ 28%), high(28% 초과)로 승률의 차이가 순차적으로 나타남을 보입니다. 확실히 도전적인 패스를 많이 하는 사람들의 승률이 높게 나타남을 볼 수 있습니다. 이를 통계적으로 검증하기 위해 ANOVA를 사용하여 집단 간의 차이가 유의미한 지 알아보았습니다.

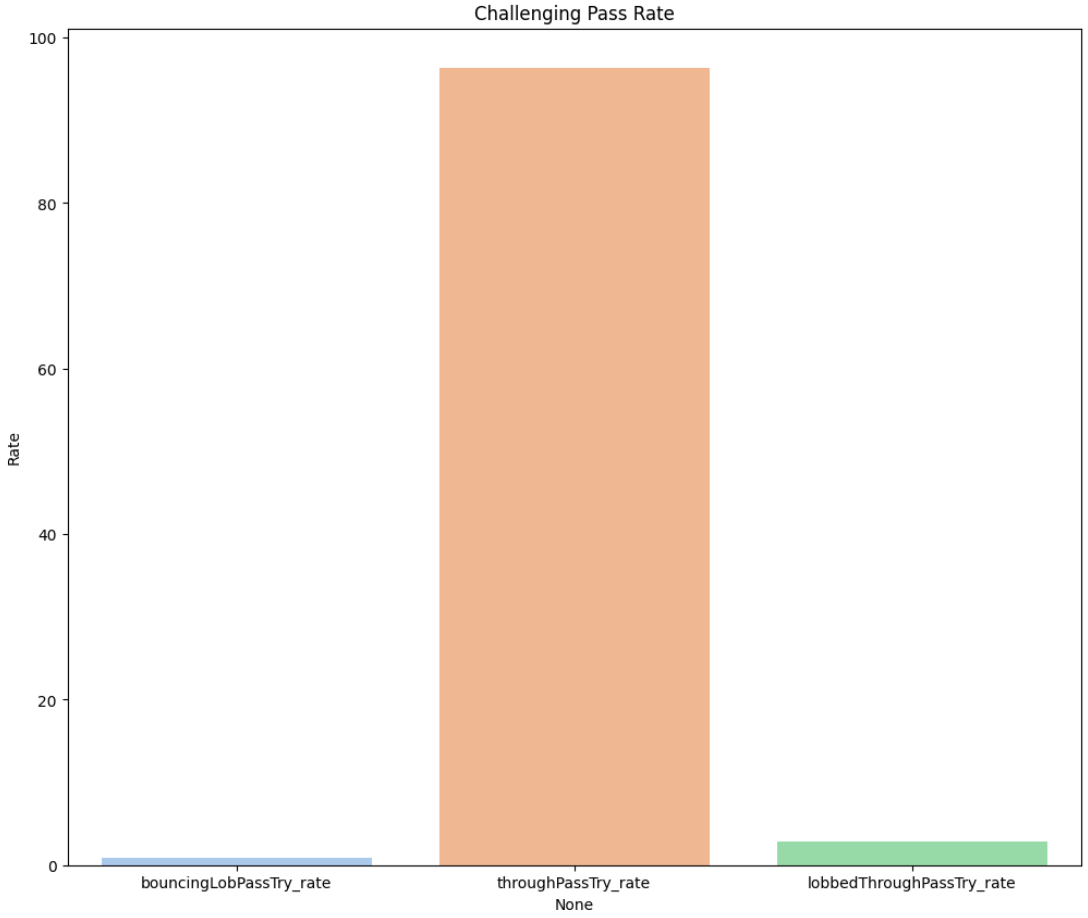
```
# 승률의 차이가 통계적으로 유의미한 지 검증
# p값이 7.827560e-48이므로 평균 도전적인 패스 시도 수에 따라서 승률의 차이
# ANOVA 모델 생성
```

```
model = ols('win_rate ~ C(pass_challenged_cat)', data=pass_tria
# ANOVA 테이블 계산
anova_table = sm.stats.anova_lm(model, typ=2)
# 결과 출력
print(anova_table)
```

	sum_sq	df	F	PR(>F)
C(pass_challenged_cat)	30.388138	2.0	109.971318	7.827560e-48
Residual	1090.113356	7890.0	NaN	NaN

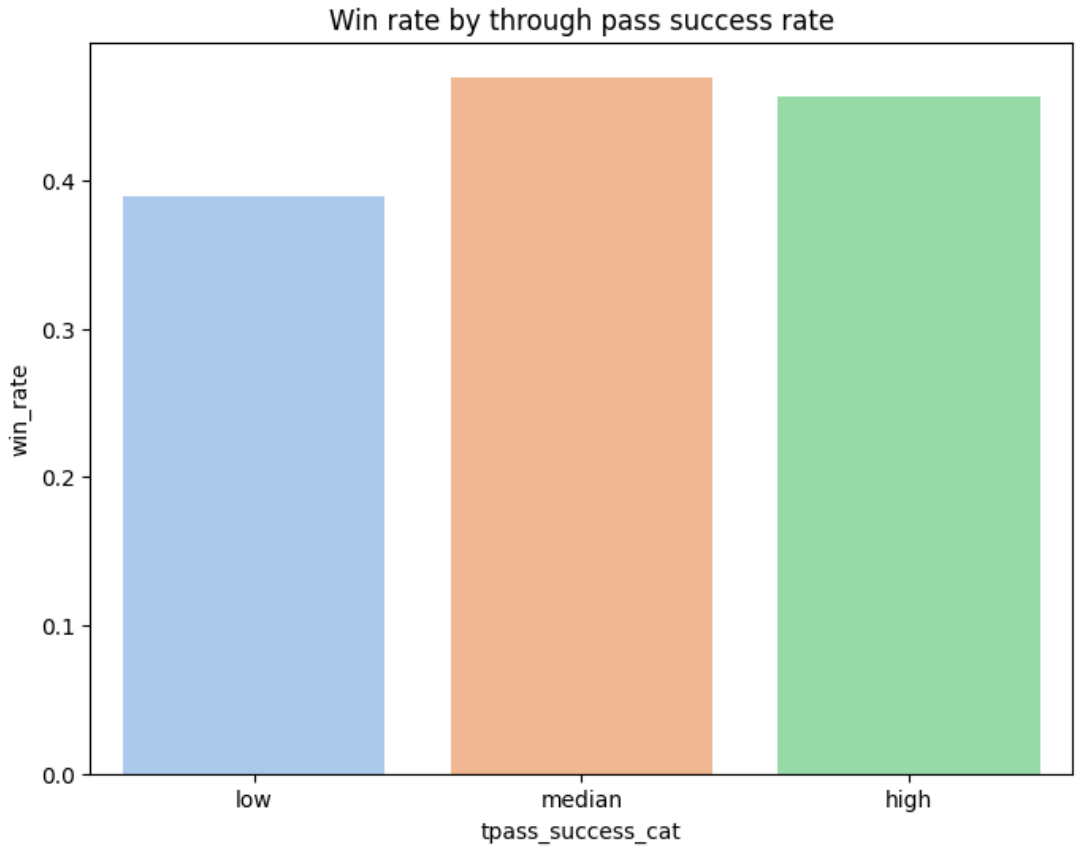
그 결과, p값이 7.827560e-48이므로, 평균 도전적인 패스 시도 비율에 따라서 승률의 차이가 난다는 것은 통계적으로 유의미한 것임을 알 수 있었습니다.

2. **유저들은 어떤 도전적인 패스를 선호할까?**



FC 온라인 유저들은 도전적인 패스로 스루패스를 선호하고 있음을 알 수 있었습니다. 스루패스의 경우 도전적인 패스 중에 95%의 비중으로 가장 많이 사용되고 있었습니다. 그렇다면 스루패스 성공률에 따라서 승률도 다르게 나타나는 것일지 궁금하였습니다.

3. **스루패스 성공률에 따라서 승률이 다르게 나타날까?**



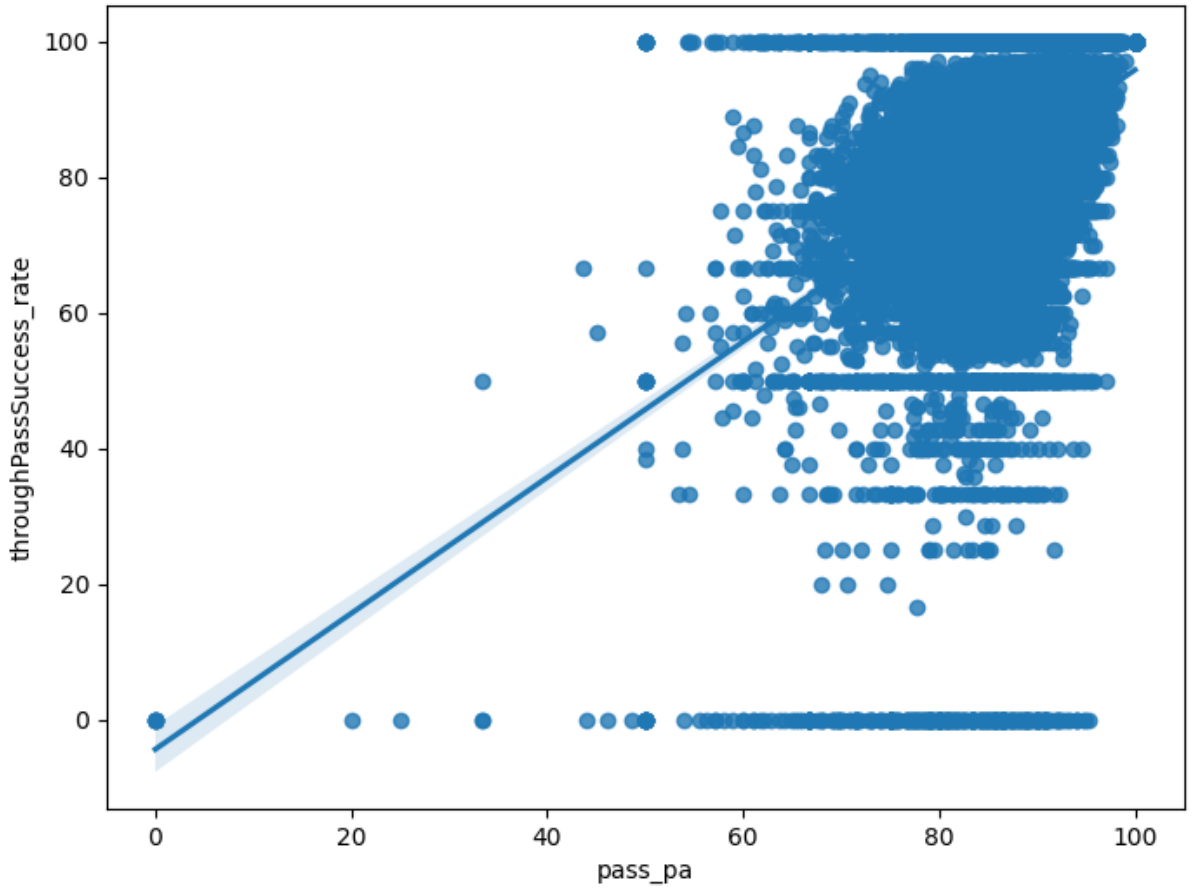
low-high-median순으로 승률이 높아지고 있음을 알 수 있었습니다. (low 약 38%, high 약 46%, median 약 47%,의 승률) 도전적인 패스에서 스루패스가 가장 많이 선호되는 패스지만, 스루패스의 성공율에 따라서 평균적인 승률이 높아진다고 볼 순 없었습니다. 그러나, 승률을 높이기 위해서는 스루패스의 성공률을 85% 언저리에 위치 시켜야 승률이 높아진다는 것을 확인할 수 있었습니다.

집단 간 차이가 유의미한지 통계적으로 증명하기 위해서 ANOVA 분석을 활용하였습니다.

```
# 스루패스 성공률에 따라 승률의 차이가 유의미하다고 볼 수 있다.
model = ols('win_rate ~ C(tpass_success_cat)', data=tpass_succe
# ANOVA 테이블 계산
anova_table = sm.stats.anova_lm(model, typ=2)
# 결과 출력
print(anova_table)
```

	sum_sq	df	F	PR(>F)
C(tpass_success_cat)	8.848411	2.0	31.400967	2.610486e-14
Residual	1111.653082	7890.0	NaN	NaN

p값이 2.610486e-14로 나타난 것을 보아 통계적으로 유의미함을 알 수 있었습니다. 즉, 승률을 높이기 위해서는 스루패스를 유저 평균 이상(85% 이상)으로 성공시켜한다는 것을 알 수 있었습니다. 그렇다면 스루패스 성공률이 좋은 선수는 누구일까요?



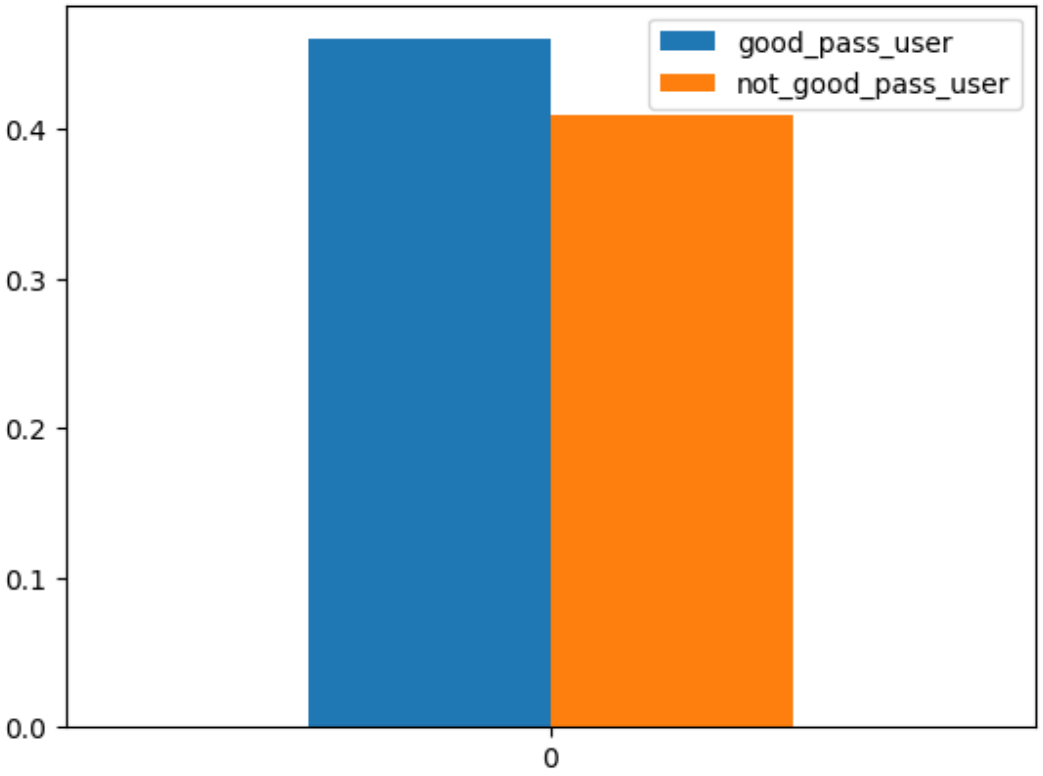
이 그래프는 패스 성공률와 스루패스 성공률을 나타낸 차트입니다. 차트를 보면 패스 성공률이 높아질 때, 스루패스 성공률도 높아짐을 알 수 있습니다. 따라서, 패스 성공률이 높은 선수들은 스루패스 성공률도 높을 것이라고 짐작할 수 있습니다. 따라서, player 데이터 프레임에서 패스 성공률이 좋은 선수들을 찾아보았습니다. 패스 성공률이 좋은 선수들을 찾아보기 위해서 10게임 이상 플레이된 선수들을 표본으로 잡고, 스루패스의 경우 주로 미드필더들이 주로 공격수들에게 골을 넣기 위해서 제공하므로, 수비수와 골키퍼의 경우는 필터링합니다. 또한, **'패스 성공률이 높다'**의 기준은 FC 온라인 통계 데이터를 기준으로 90% 이상이 평균이지만, 수집된 데이터에서는 성공률 90%가 상위 75% 구간이기 때문에 90% 이상을 **'패스 성공률이 높다'**라고 설정하였습니다.

	className	name	passSuccessRate	assist
0	JNM (Journeyman)	야리 리트마넨	0.988889	0.100000
1	24 TOTS (24 Team Of The	세루 기라시	0.980000	0.600000

	Season)			
2	BOE21 (Best of Europe 21)	그라니트 자카	0.978992	0.000000
3	24 TOTS (24 Team Of The Season)	A. 바스토니	0.978682	0.090909
4	ICON TM(제한) (ICON The Moment Bound)	존 반스	0.975855	0.125000
…	…	…	…	…
366	ICON TM (ICON The Moment)	호나우두	0.900340	0.000000
367	22 TOTS (22 Team Of The Season)	주드 벨링엄	0.900276	0.148148
368	UT (Unexpected Transfer)	레반도프스키	0.900243	0.300000
369	24 TOTY-N (24 Team Of The Nominated)	루카 모드리치	0.900221	0.274194
370	ICON (ICON)	디디에 드로그바	0.900194	0.384236

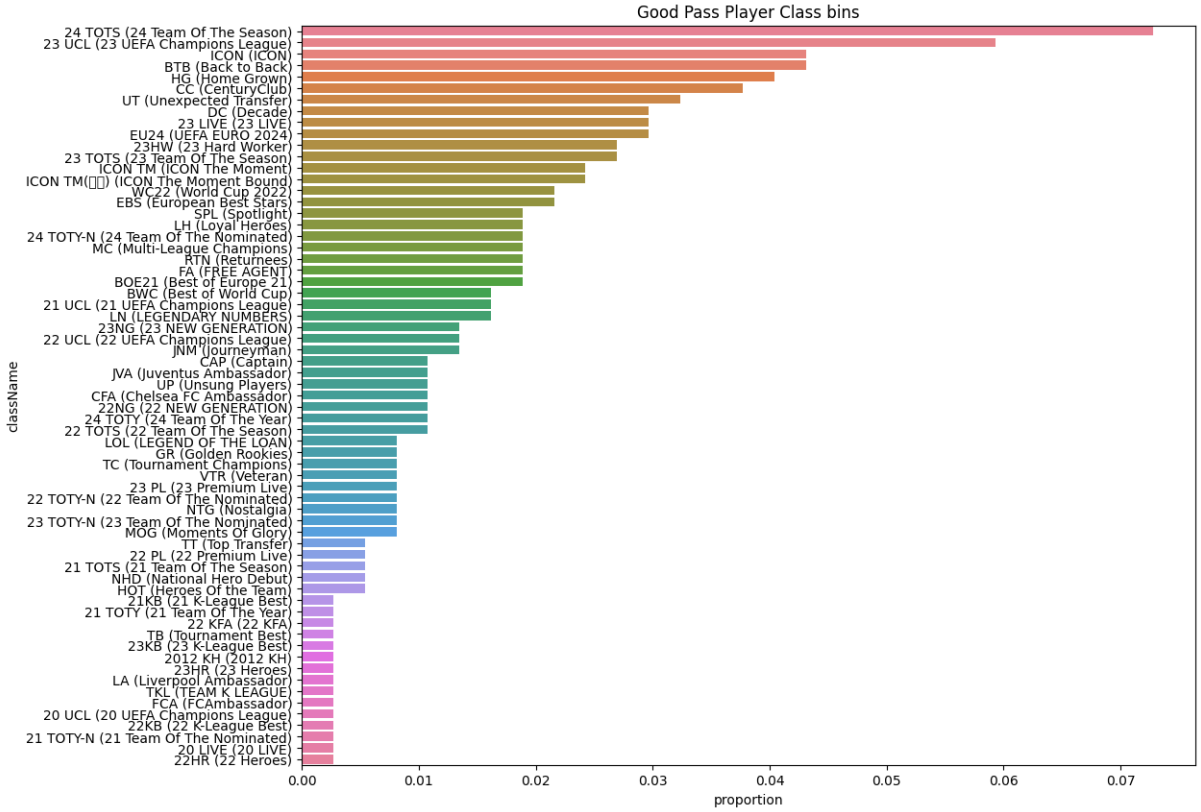
그 결과, 다음과 같은 선수들을 도출할 수 있었습니다.

4. **패스 성공률이 높은 선수를 사용할 때 승률이 더 높을까?**



good_pass_user의 경우 패스 성공률이 높은 선수들을 사용한 유저들의 집단이고, not_good_pass_user은 그 이외의 선수들을 사용한 유저들의 집단입니다. 패스 성공률이 좋은 선수를 사용한 유저들의 경우 승률이 46%, 그 이외의 선수들을 사용한 유저들의 집단의 경우 평균 승률은 40%로 나타납니다. 따라서, 패스 성공률이 좋은 유저를 사용할 경우, 6%의 승률이 더 높음을 알 수 있습니다.

5. **패스가 좋은 선수들은 어떤 클래스에 가장 많이 속해있을까?**



패스가 좋은 선수들은 24 TOTS, 23 UCL, ICON, BTB.. 순으로 나타나는 것을 볼 수 있습니다. 특히, 24TOTS, 23UCL, ICON, BTB의 경우, 패스가 좋은 선수들이 전체 선수에서 각각 7%, 6%, 4%, 4%에 속해있습니다. 따라서, 해당 표에 나타난 클래스의 선수들을 사용한다면 승리할 확률이 높일 수 있을 것입니다.

3.4.1.1 결론 및 전략제시

플레이 메이커를 영입하라

패스 성공률이 높은 선수를 사용하는 유저라면 상대편보다 승률을 6%정도 더 우위를 가질 수 있습니다. 따라서, 좋은 플레이 메이커를 영입하여 패스의 질을 높여보는 것이 좋습니다.

안전한 패스를 피해라

스루패스, 로빙 스루 패스를 적극적으로 활용해야 합니다. 스루패스, 로빙 스루 패스를 적극적으로 활용하고 도전적인 패스를 80% 이상 성공시킨 사람이 승리할 확률이 낮게 성공시킨 집단보다 8~9% 높게 나타납니다.

24TOTS, 23 UCL, ICON, BTB 선수를 구매하라

패스가 좋은 선수들은 24TOTS, 23UCL, ICON, BTB에 각각 7%, 6%, 4%, 4% 속해있습니다. 따라서, 해당 클래스에 있는 선수들을 사용한다면 승리할 확률을 높여줄 것입니다.

3.4.2 팀 가치가 높을수록 승률이 높을까?

가설 배경

팀 가치는 게임 내의 선수 18명의 가치를 합친 것을 의미합니다. 팀 가치가 높다면 그만큼 강력한 선수들이 팀 내에서 존재한다는 것임을 유추할 수 있습니다. 좋은 선수들로 구성된 팀이라면 게임 내에서 상대를 만났을 때 승리할 확률이 더 높다는 것이 증명된다면 승률 1%를 올리기 위해서 과금을 어느 정도 해야할 지 근거를 제시해 줄 수 있을 것이며, 그에 따른 게임사의 과금 전략을 제시해 볼 수 있을 것입니다.

가설의 중요성

게임사에게 과금 전략을 세우는 데 중요한 인사이트를 제공합니다. 승률을 높이기 위한 과금 패턴을 이해함으로써, 게임사는 다양한 과금 옵션을 제공할 수 있습니다. 예를 들어, 특정 금액의 과금을 통해 팀 가치를 단기간에 크게 향상시킬 수 있는 프로모션이나 이벤트를 기획할 수 있습니다. 이는 플레이어들의 과금 유도를 통해 게임사의 수익을 극대화하는 데 기여할 것입니다.

배경 지식

- 평균적으로 15만원 정도 과금했을 때. 1조 BP를 얻을 수 있습니다. 출처: FC온라인 커뮤니티

데이터 분석

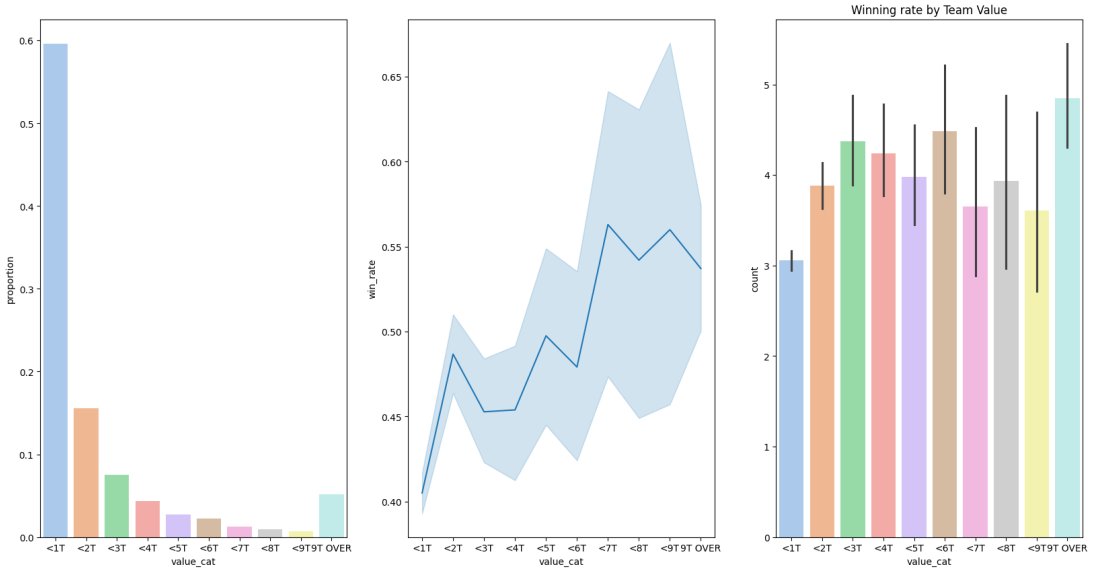
해당 데이터를 분석하기 위해서는 구단 가치 카테고리를 만듭니다.

```
# 전체 유저를 대상으로 데이터 프레임 형성
f1 = fifa_money.groupby(['nickname'])[['team_value','real_money
f1 = pd.merge(fifa_temp, f1, on=['nickname'])

# 구단 가치 카테고리 만들기
def value_cat(x):
    if x<1000000000000:
        return '<1T'
    elif x>=1000000000000 and x<2000000000000:
        return '<2T'
    elif x>=2000000000000 and x<3000000000000:
        return '<3T'
    elif x>=3000000000000 and x<4000000000000:
        return '<4T'
    elif x>=4000000000000 and x<5000000000000:
        return '<5T'
    elif x>=5000000000000 and x<6000000000000:
        return '<6T'
    elif x>=6000000000000 and x<7000000000000:
        return '<7T'
    elif x>=7000000000000 and x<8000000000000:
```

```
        return '<8T'
    elif x>=8000000000000 and x<9000000000000:
        return '<9T'
    else:
        return '9T OVER'

f1['value_cat'] = f1['team_value'].apply(value_cat)
f1.groupby(['value_cat'])['win_rate'].mean()
order = ['<1T','<2T','<3T','<4T','<5T','<6T','<7T','<8T','<9T',
f1['value_cat'] = pd.Categorical(f1['value_cat'], categories=or
```

구단 가치 카테고리는 1조를 간격으로 카테고리 화 되었으며, 9조가 넘는 구단일 경우 9T OVER 카테고리에 할당합니다.

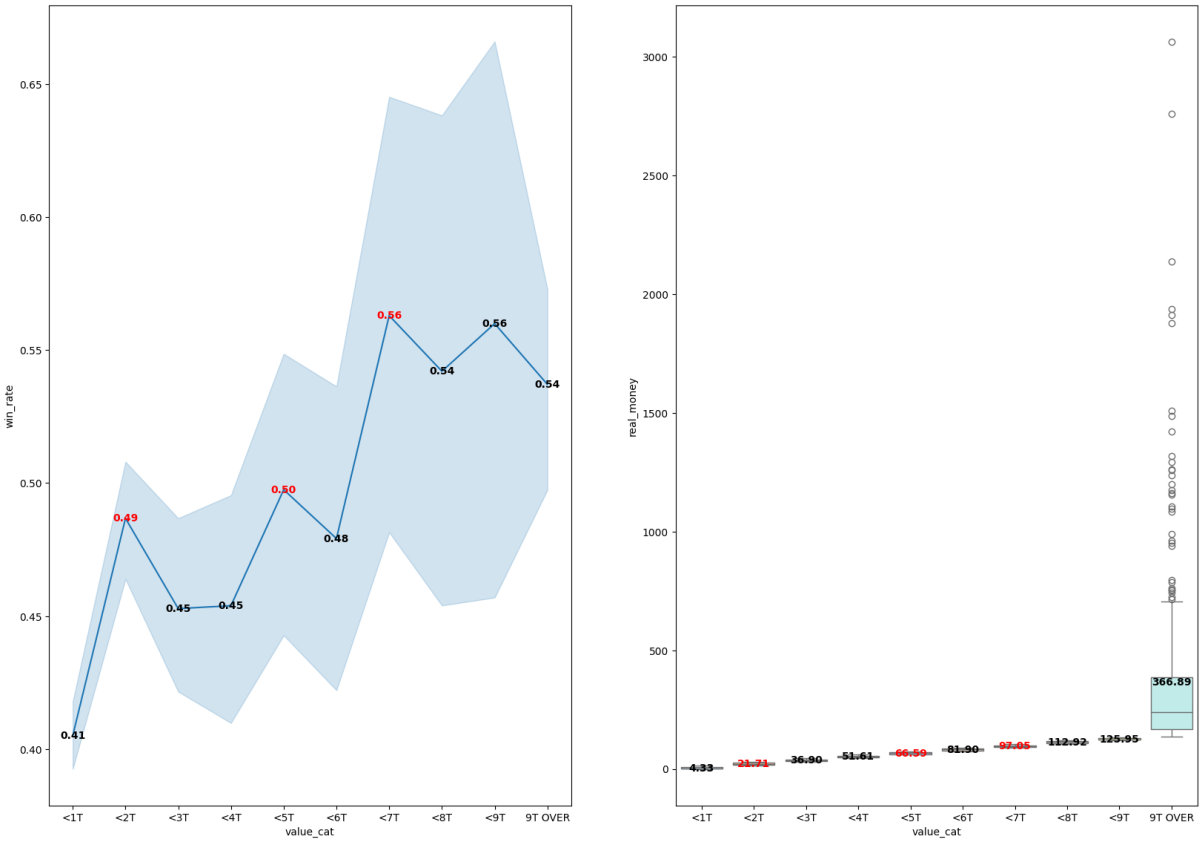
1. **유저가 속한 구단 가치 빈도 막대 그래프/구단 가치에 따른 승률/구단 가치에 따른 평균 게임 플레이 횟수**



구단 가치 빈도 막대그래프를 확인해본 결과 1조 이하 구단이 가장 많으며, 9조 이하 구단까지 빈도 수가 하락하다가, 9조 이상 구단에서 상승하는 경향을 볼 수 있었습니다. 뿐만 아니라, 구단 가치에 따른 승률 선 그래프를 확인해본 결과, 구단 가치에 따라 승률이 상승하고 있는 추세를 볼 수 있습니다. 구단 가치에 따른 평균 게임 플레이 횟수는 전반적인 추세를 보이진 않지만, 9조 이상의 구단 가치를 지닌 유저들의 경우, 약 5게임 정도(약 1시간) 하루에 플레이함을 볼 수 있었습니다.

	sum_sq	df	F	PR(>F)
C(value_cat)	13.39441	9.0	10.729345	1.059947e-16
Residual	891.48934	6427.0	NaN	NaN

구단 가치에 따라 승률의 차이가 유의미한 것을 파악하기 위해 ANOVA 분석을 시행했습니다. 그 결과 p값이 1.059947e-16으로 구단 가치에 따라 승률의 차이가 유의미함을 알 수 있었습니다.



위의 그래프는 구단 가치에 따라 승률이 상승하는 것을 나타내는 그래프와 구단 가치를 현금화(1조 당 15만원)하여 나타낸 박스 플롯입니다. 해당 그래프들을 종합적으로 판단하였을 때, 1조 이하의 구단은 승률이 41% 에서 2조 구단은 49%, 5조 구단은 50%, 7조 구단은 56%로 상승하고, 7조 이후의 구단에서는 50%를 상회하는 확률로 나타나고 있는 것을 볼 수 있습니다. 즉, 1조 이하 구단에서 2조 이하 구단으로 가면 승률이 약 8% 오른다고 볼 수 있고, 2조 이하 구단에서 5조 이하 구단으로 가면 승률이 약 1% 오른다고 볼 수 있으며, 5조 이하 구단에서 7조 이하 구단으로 가면 승률이 약 6% 오른다고 볼 수 있습니다. 이를 1조의 구단가치 당 15만원의 현금 가치를 지닌다는 점을 알 때, 즉, 1조 이하 구단에서는 승률을 1% 올리기 위해 18500원(15만원/8)을 과금해야 하며, 2조 이하 구단에서는 승률을 1% 올리기 위해 45만원을 과금해야 하고, 5조 이하 구단에서는 승률을 1% 올리기 위해 5만원을 과금해야한다는 것을 알 수 있습니다.

3.4.2.1 결론

1조 이하 구단들을 상대로 과금 전략을 펼쳐라!

1조 구단에 위치한 유저들은 표본 중에 가장 많은 비율을 차지하며 평균 승률이 매우 낮습니다.(41%) 그러나, 이 유저들이 2조 구단으로 탈바꿈하는 경우 승률이 가장 급격하게 상승하게 되며 게임 운영에 있어 중요한 유저들이기 때문에 많이 패배하는 유저들을 대상으로 전력강화 패키지(15만원)을 출시하여 한정으로 출시하거나 혹은 실속강화패키지 (18500원)을 출시하여 이들이 과금할 수 있도록 유도해야 합니다.

부자 구단도 대우하라!

9조 이상 구단에 위치한 유저들은 현금으로 예상되는 구단 가치가 약 360만원이(구단 현금가치 추정 편차도 크다) 넘으며, 데이터 상에 하루 평균 5게임 정도 하는 충성도가 높은 유저들이기 때문에 이 유저들에게는 가격대가 높은 패키지를 팔거나, VIP 제도 등을 적용해서 이탈을 방지하도록 하는 멤버십/프리미엄 과금 전략을 사용합니다.

임대 선수를 이용하기

24TOTS, 23UCL, ICON 시즌 등 패스 성공률이 높은 선수들을 단기간 임대할 수 있는 서비스를 제공하고.임대 기간과 선수의 등급에 따라 다양한 가격대를 설정하여 1조 이하 구단 가치를 지닌 사람들에게 좋은 선수를 이용할 수 있도록 제시하고 이후 선수를 직접 사용할 수 있도록 과금을 유도해야 합니다.

4. 머신러닝 모델링

4.1.0 라이브러리

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split,cross_val_
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_ma
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from scipy import stats
```

```
from sklearn.model_selection import GridSearchCV
import optuna
import xgboost as xgb
import lightgbm as lgb
from sklearn.svm import SVC
```

4.1.1. 모델 학습을 위한 추가 전처리

종속변수인 플레이어의 매치 결과(matchDetail.matchResult)와 상관 관계가 높은 평점(matchDetail.averageRating)과 화면 상 표시 골(shoot.goalTotalDisplay)에 대해 학습 전에 추가적으로 제거를 진행하였습니다. 평점은 플레이어의 실력, 다른 지표와 상관없이 승리 무조건 높게 나오는 경향이 있으며, 화면 상 표시 골은 일반적으로 실제로 유저가 득점한 골의 개수가 표시되지만, 기권 등으로 인한 몰수패인 경우 승리한 유저의 화면에 무조건 3득점을 한 것으로 표기됩니다. 즉, 이러한 이유를 근거로 모델 학습에 방해를 주는 컬럼으로 판단하였습니다.

```
# 평점 및 화면 상 표시 골 컬럼 삭제

mcdf = pd.read_csv('match_without_json.csv')
mcdf=mcdf.drop(['shoot.goalTotalDisplay','matchDetail.average
Rating'],axis=1)
```

4.1.2. Feature 재정의

모델 학습을 위해 데이터셋의 피처들을 카테고리형과 수치형으로 구분하였습니다. 정의된 수치형 피처들은 주로 플레이어의 게임 플레이 통계와 관련된 다양한 측정값을 포함합니다. 이 과정에서 카테고리형 피처를 제외한 나머지 모든 피처를 수치형 피처로 간주하여, 이후 전처리 및 모델 학습에 활용될 수 있도록 준비하였습니다.

```
# 카테고리형 및 수치형 피처 구분

categorical_features = ['matchDetail.controller_gamepad', 'ma
tchDetail.controller_keyboard']
numeric_features = [col for col in mcdf.columns[3:] if col no
t in categorical_features]
```

4.1.3. 이상치 처리

모델의 성능을 저해할 수 있는 이상치(outliers)에 대해 다음과 같은 처리를 진행하였습니다. 우선, 모든 수치형 변수(numeric_features)에 대해 z-점수(z-score)를 계산하였으며, 그 절대값이 3을 초과하는 경우를 이상치로 정의하였습니다. 이러한 이상치는 데이터 분포의 정상적인 범위를 벗어나는 값으로, 모델 학습에 방해가 될 수 있기 때문에 처리 대상이 되었습니다. z-점수가 3을 초과하는 값을 해당 변수의 중앙값(median)으로 대체하였습니다. 중앙값은 극단값의 영향을 받지 않으므로 이상치 처리 대체 방법으로 선정하였고고, 이를 통해 데이터의 분포를 안정화하고 모델 학습의 신뢰성을 높이고자 합니다.

```
## 이상치 처리

z_scores = np.abs(stats.zscore(mcdf[numeric_features]))
threshold = 3
for col in numeric_features:
    median = mcdf[col].median()
    mcdf.loc[np.abs(stats.zscore(mcdf[col])) > threshold, co
l] = median
```

4.2.1 데이터 분할

모델 학습을 위해 데이터를 학습용과 테스트용으로 분할하였습니다. 이를 통해 모델의 성능을 검증하고 일반화 능력을 평가할 수 있습니다.

먼저, 종속 변수와 독립 변수를 분리하였습니다:

```
X = mcdf.drop('matchDetail.matchResult', axis=1)
y = mcdf['matchDetail.matchResult']
X_train, X_test, y_train, y_test = train_test_split(X, y, tes
t_size=0.3, random_state=42)
```

- **X**: 독립 변수로, `matchDetail.matchResult`를 제외한 모든 피처를 포함합니다.
- **y**: 종속 변수로, `matchDetail.matchResult`를 포함합니다.

이후, 데이터를 학습용과 테스트용으로 분할하였습니다:

- **X_train, y_train**: 전체 데이터의 70%를 차지하는 학습용 데이터입니다.
- **X_test, y_test**: 전체 데이터의 30%를 차지하는 테스트용 데이터입니다.

`test_size=0.3`은 데이터의 30%를 테스트 세트로 사용함을 의미하며, `random_state=42`는 데이터 분할의 재현성을 위해 난수 생성 시드를 설정한 것입니다.

4.2.2 하이퍼 파라미터 튜닝

모델의 성능을 최적화하기 위해 **Optuna**를 사용하여 하이퍼 파라미터 튜닝을 수행하였습니다. Optuna는 효율적인 하이퍼 파라미터 최적화를 위해 설계된 오픈소스 라이브러리로, Bayesian Optimization을 기반으로 한 샘플링 기법을 사용합니다.

Optuna의 주요 원리와 장점은 다음과 같습니다:

- **효율적인 탐색**: Bayesian Optimization 기법을 사용하여 하이퍼 파라미터 공간을 효율적으로 탐색합니다. 이를 통해 적은 수의 시도만으로도 최적의 하이퍼 파라미터를 찾을 수 있습니다.
- **Pruning 기능**: 중간 평가를 통해 성능이 낮은 시도를 조기에 종료(pruning)하여 계산 자원을 절약합니다.
- **유연성**: 다양한 목적 함수와 제약 조건을 지원하여 다양한 모델과 문제에 적용 가능합니다.

4.2.3 Logistic Regression

로지스틱 회귀(Logistic Regression)는 이진 분류 문제를 해결하기 위해 널리 사용되는 선형 모델입니다. 이 모델은 입력 변수들의 선형 결합을 사용하여 특정 사건이 발생할 확률을 예측합니다.

```
def objective(trial):
    param = {
        'classifier__C': trial.suggest_loguniform('classifier__
        'classifier__solver': trial.suggest_categorical('classi
        'classifier__penalty': trial.suggest_categorical('class
    }

    # 파이프라인 생성
    preprocessor = ColumnTransformer(
        transformers=[
            ('num', StandardScaler(), numeric_features),
            ('cat', OneHotEncoder(), categorical_features)
        ])
```

```
    pipeline_lg = Pipeline(steps=[
        ('preprocessor', preprocessor),
        ('classifier', LogisticRegression(max_iter=1000))
    ])

    pipeline_lg.set_params(**param)
    return cross_val_score(pipeline_lg, X_train, y_train, cv=5,

study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=10)

# 최적 파라미터 설정 및 학습
best_params = study.best_trial.params

# 파이프라인 생성 및 학습
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numeric_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])

pipeline_lg = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(max_iter=1000))
])

pipeline_lg.set_params(**best_params)
pipeline_lg.fit(X_train, y_train)

# 예측 및 평가
y_pred = pipeline_lg.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

4.2.4 Random Forest

랜덤 포레스트(Random Forest)는 앙상블 학습 방법의 하나로, 여러 개의 의사결정 트리를 사용하여 분류 또는 회귀 문제를 해결합니다. 각 트리는 독립적으로 학습되고, 최종 예측은 모든 트리의 예측을 평균 내거나 다수결 투표를 통해 결정됩니다.

```
def objective(trial):
    param = {
        'n_estimators': trial.suggest_int('n_estimators', 100,
        'max_depth': trial.suggest_int('max_depth', 3, 30),
        'min_samples_split': trial.suggest_int('min_samples_spl
        'min_samples_leaf': trial.suggest_int('min_samples_leaf
        'max_features': trial.suggest_categorical('max_features
    }

    pipeline = Pipeline(steps=[
        ('preprocessor', preprocessor),
        ('classifier', RandomForestClassifier(**param, random_s
    ])

    scores = cross_val_score(pipeline, X_train, y_train, cv=5,
    return scores.mean()

study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=10)

# 최적의 하이퍼파라미터를 사용하여 최종 모델 학습
best_params = study.best_trial.params

pipeline_rf = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(**best_params, random
])

pipeline_rf.fit(X_train, y_train)

# 모델 예측 및 성능 평가
y_pred = pipeline_rf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

4.2.5 Xgboost

XGBoost는 극도로 효율적이고 유연한 부스팅 알고리즘으로, 주로 회귀 및 분류 문제를 해결하는 데 사용됩니다. 여기서 부스팅(Boosting)은 약한 학습자(주로 결정 트리)를 결합하여 강력한 학습자를 만드는 앙상블 학습 방법이다. XGBoost는 속도와 성능을 크게 향상시킨 버전의 Gradient Boosting 알고리즘이다.

```
def objective(trial):
    param = {
        'verbosity': 0,
        'objective': 'multi:softprob',
        'num_class': 3,
        'eval_metric': 'mlogloss',
        'eta': trial.suggest_loguniform('eta', 0.01, 0.3),
        'max_depth': trial.suggest_int('max_depth', 3, 9),
        'min_child_weight': trial.suggest_int('min_child_weight
        'gamma': trial.suggest_loguniform('gamma', 0.1, 1.0),
        'subsample': trial.suggest_float('subsample', 0.5, 1.0)
        'colsample_bytree': trial.suggest_float('colsample_bytr
        'lambda': trial.suggest_loguniform('lambda', 1e-3, 10.0
        'alpha': trial.suggest_loguniform('alpha', 1e-3, 10.0),
    }

    pipeline = Pipeline(steps=[
        ('preprocessor', preprocessor),
        ('classifier', xgb.XGBClassifier(**param, use_label_enc
    ])

    scores = cross_val_score(pipeline, X_train, y_train, cv=5,
    return scores.mean()

study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=30)

# 최적의 하이퍼파라미터를 사용하여 최종 모델 학습
best_params = study.best_trial.params
```

```
best_params['verbosity'] = 0
best_params['objective'] = 'multi:softprob'
best_params['num_class'] = 3
best_params['eval_metric'] = 'mlogloss'

pipeline_xgb = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', xgb.XGBClassifier(**best_params, use_label_e
])

pipeline_xgb.fit(X_train, y_train)

# 평가
y_pred = pipeline.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

4.2.6 LightGBM

LightGBM은 Microsoft에서 개발한 고성능 Gradient Boosting Framework입니다. 주로 대규모 데이터셋과 고차원 데이터에 대해 매우 빠르고 효율적인 학습을 제공하며, LightGBM은 주로 분류, 회귀, 순위 예측 등의 작업에 사용됩니다.

```
def objective(trial):
    param = {
        'objective': 'multiclass',
        'num_class': 3,
        'metric': 'multi_logloss',
        'learning_rate': trial.suggest_loguniform('learning_rat
        'max_depth': trial.suggest_int('max_depth', 3, 9),
        'num_leaves': trial.suggest_int('num_leaves', 20, 150),
        'feature_fraction': trial.suggest_float('feature_fracti
        'bagging_fraction': trial.suggest_float('bagging_fracti
        'bagging_freq': trial.suggest_int('bagging_freq', 1, 7)
        'lambda_l1': trial.suggest_loguniform('lambda_l1', 1e-3
        'lambda_l2': trial.suggest_loguniform('lambda_l2', 1e-3
```

```
        'min_child_samples': trial.suggest_int('min_child_sampl
        'min_child_weight': trial.suggest_loguniform('min_child
    }

    pipeline_lgb = Pipeline(steps=[
        ('preprocessor', preprocessor),
        ('classifier', lgb.LGBMClassifier(**param))
    ])

    pipeline_lgb.fit(X_train, y_train)
    preds = pipeline_lgb.predict(X_test)
    accuracy = accuracy_score(y_test, preds)
    return accuracy

study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=30)

best_params = study.best_trial.params
best_params['objective'] = 'multiclass'
best_params['num_class'] = 3
best_params['metric'] = 'multi_logloss'

pipeline_lgb = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', lgb.LGBMClassifier(**best_params))
])

pipeline_lgb.fit(X_train, y_train)

# 평가
preds = pipeline_lgb.predict(X_test)
accuracy = accuracy_score(y_test, preds)
conf_matrix = confusion_matrix(y_test, preds)
class_report = classification_report(y_test, preds)
```

4.2.7 성능 평가

각 모델에 대해 Accuracy, Precision, Recall, F1-score 지표를 통해 성능을 비교하였습니다.

Model	Accuracy	Precision	Recall	F1-score
Logistic Regression	**0.7313**	0.70 0.57 0.76	0.84 0.01 0.82	0.77 0.02 0.79
Random Forest	0.7357	0.73 1.00 0.75	0.83 0.00 0.86	0.77 0.01 0.80
XgBoost	0.7379	0.73 0.43 0.74	0.82 0.01 0.86	0.77 0.01 0.79
LightGBM	**0.7384**	0.72 0.14 0.75	0.83 0.00 0.85	0.77 0.00 0.80

- 로지스틱 회귀 모델은 0 클래스에 대한 성능이 매우 낮습니다. 이는 모델이 0 클래스를 거의 예측하지 못함을 나타냅니다. -1 클래스와 1 클래스에 대해서는 비교적 균형 잡힌 성능을 보여줍니다.
- 랜덤 포레스트 모델도 0 클래스에 대한 성능이 매우 낮습니다. 이는 모델이 0 클래스를 거의 예측하지 못함을 나타냅니다. -1 클래스와 1 클래스에 대해서는 비교적 좋은 성능을 보입니다.
- XGBoost 모델은 1 클래스에 대한 성능이 매우 낮습니다. 이는 모델이 1 클래스를 거의 예측하지 못함을 나타냅니다. 0 클래스와 2 클래스에 대해서는 비교적 좋은 성능을 보입니다. (xgboost는 class 0부터 시작)
- LightGBM 모델도 1 클래스에 대한 성능이 매우 낮습니다. 이는 모델이 1 클래스를 거의 예측하지 못함을 나타냅니다. 0 클래스와 2 클래스에 대해서는 비교적 좋은 성능을 보입니다.

모델 공통점

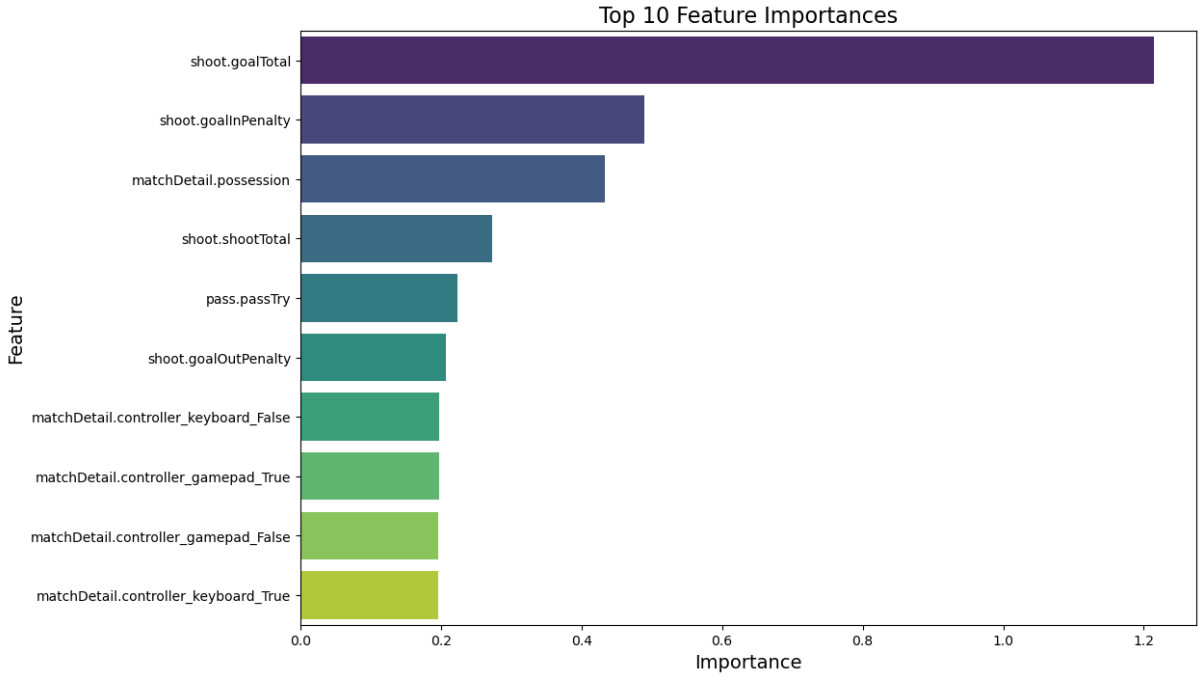
- **비슷한 정확도**: 모든 모델의 정확도가 비슷하게 나타납니다. 이는 모델들이 데이터셋에서 비슷한 패턴을 학습하고 있으며, 클래스 불균형으로 인한 한계가 정확도에 반영되었기 때문일 가능성이 있습니다. 정확도는 전체 데이터 중 올바르게 예측된 샘플의 비율을 나타내지만, 클래스 불균형 문제를 반영하지 못하기 때문에 각 클래스별 세부 성능 지표를 함께 고려해야 합니다.

- **무승부에 대한 낮은 성능**: 무승부에 대한 Precision과 Recall이 매우 낮습니다. 이는 데이터셋에서 무승부가 잘 예측되지 않는다는 것을 의미합니다. 이러한 성능 저하는 데이터의 불균형, 클래스의 특성, 또는 모델의 한계에서 기인할 수 있습니다.

모델들의 성능을 종합적으로 평가할 때, LightGBM과 XGBoost가 전반적으로 우수한 성능을 보여줍니다.

4.2.8 Feature Importance

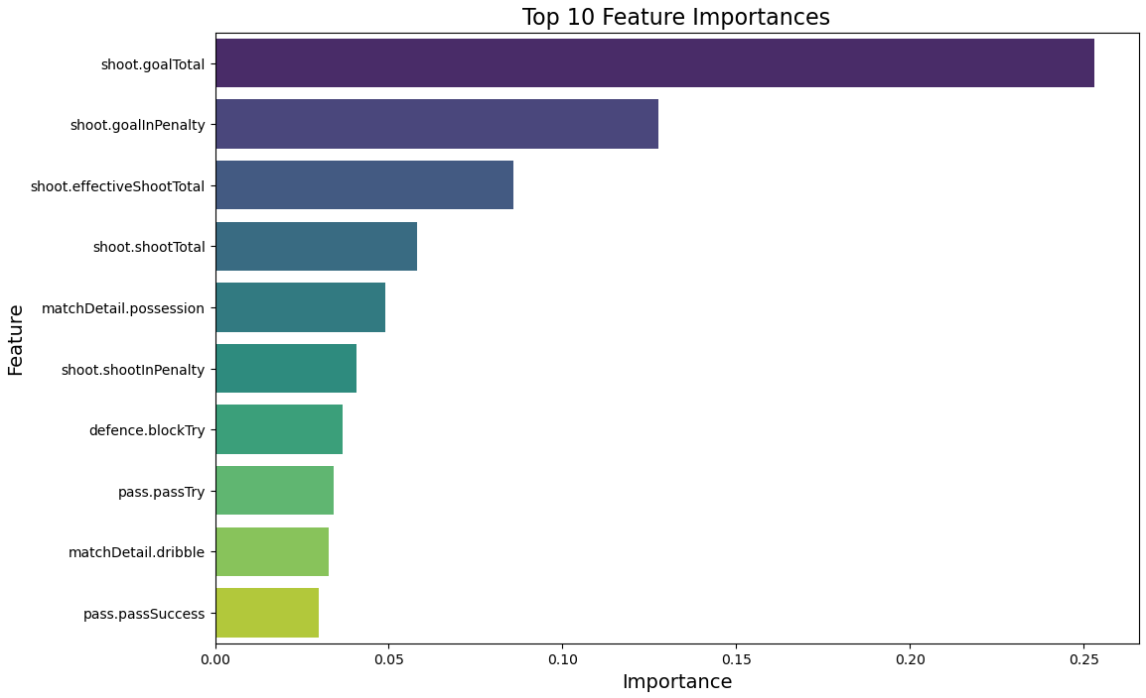
- **Logistic Regression**




```
Top 10 Features by Importance:

shoot.goalTotal                          1.213966
shoot.goalInPenalty                      0.488905
matchDetail.possession                   0.433146
shoot.shootTotal                         0.272574
pass.passTry                             0.222915
shoot.goalOutPenalty                     0.207316
matchDetail.controller_keyboard_False    0.197463
matchDetail.controller_gamepad_True      0.197463
```

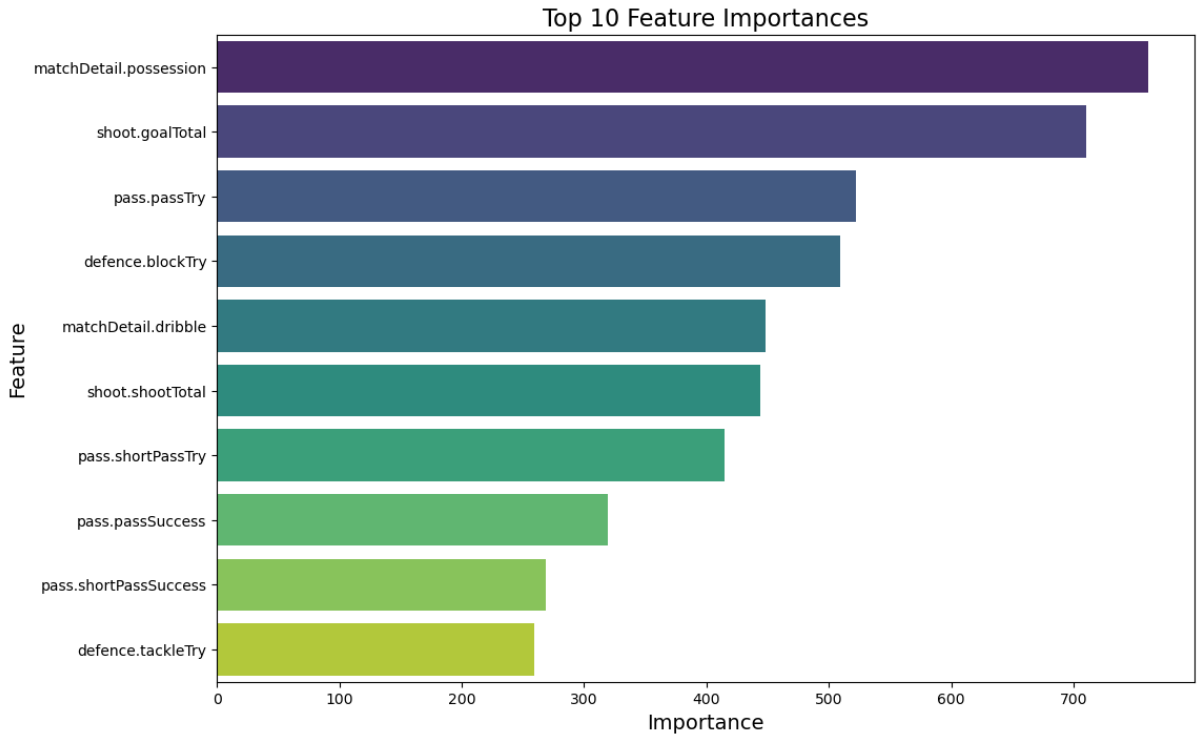
```
matchDetail.controller_gamepad_False     0.196675
matchDetail.controller_keyboard_True     0.196675
```

- **Random Forest**



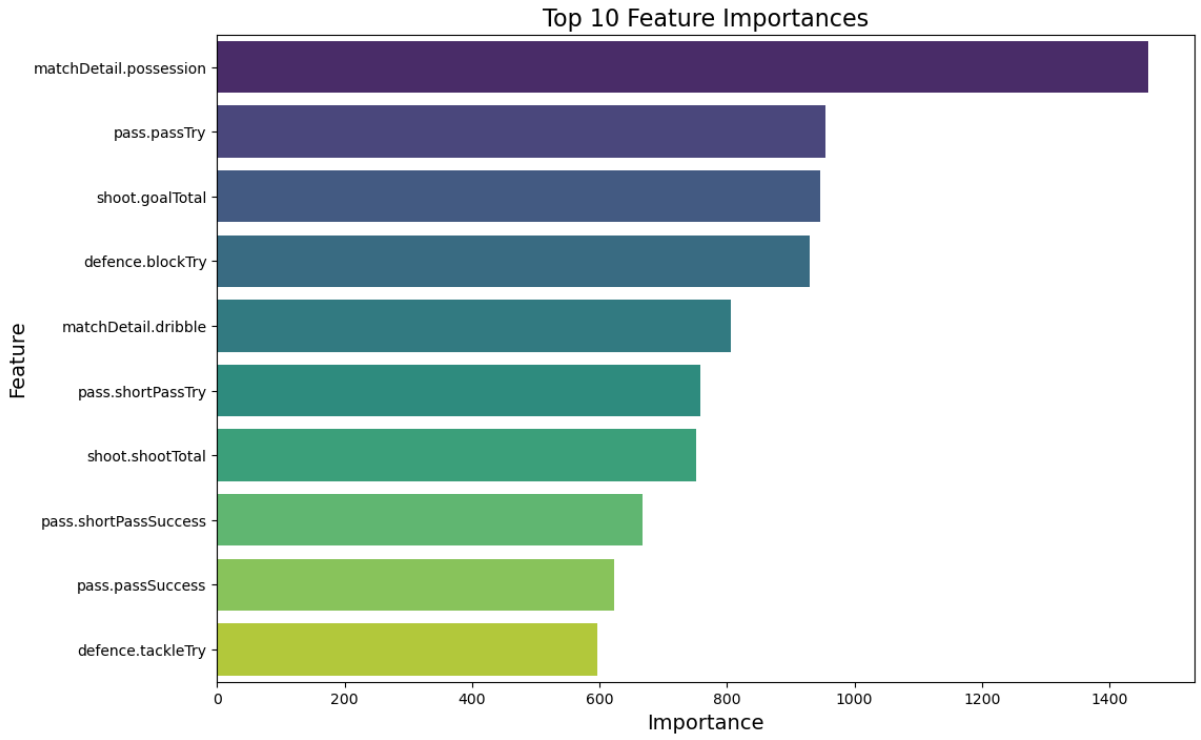

```
Top 10 Features by Importance:

shoot.goalTotal              0.253116
shoot.goalInPenalty          0.127670
shoot.effectiveShootTotal    0.085905
shoot.shootTotal             0.058197
matchDetail.possession       0.049063
shoot.shootInPenalty         0.040631
defence.blockTry             0.036846
pass.passTry                 0.034051
matchDetail.dribble          0.032649
pass.passSuccess             0.029926
```

- Xgboost




```
Top 10 Features by Importance:

matchDetail.possession    752
shoot.goalTotal           713
pass.passTry              523
defence.blockTry          518
matchDetail.dribble       467
shoot.shootTotal          462
pass.shortPassTry         420
pass.passSuccess          324
pass.shortPassSuccess     273
defence.tackleTry         266
```

- LightGBM




```
Top 10 Features by Importance:

matchDetail.possession    1461
pass.passTry               954
shoot.goalTotal            947
defence.blockTry           930
matchDetail.dribble        806
pass.shortPassTry          758
shoot.shootTotal           752
pass.shortPassSuccess      667
pass.passSuccess           623
defence.tackleTry          597
```

공통 중요 요인

- 총 득점 수(goalTotal)

플레이어가 득점할 수록 매치에서 승리할 확률이 높음

- 매치 점유율(possesion)

플레이어가 공을 점유한 시간이 길수록 매치에서 승리할 확률이 높음

특정 중요 요인

- 패널티 존 득점(goalInPenalty)

LR, RF Model

- 패스 시도 수(pass.try)

XGB, LGBM Model