

# 서울월드컵경기장 K리그 관객 수 예측

2023315251 심건우



# CONTENTS

01

프로세스 소개

02

데이터 수집

03

데이터 전처리

04

데이터 학습

05

모델 평가

06

예측



# 프로세스 소개

## 요약

- 1) 회귀를 통해 관객수 예측
- 2) 데이터프레임에 수치형 데이터와 Object 데이터 존재
- 3) Object 데이터를 원-핫 인코딩으로 0,1로 변환시키면 번거로운 점이 많아짐을 확인
- 4) Object 데이터를 Word2Vec모델로 벡터화 한 후 수치형 데이터와 함께 회귀모델에 투입

# 데이터 수집

데이터

## 서울시설공단\_서울월드컵경기장 주경기장 사용자 통계\_20221231

- 데이터 출처/제공기관: 공공 데이터포털/서울시설공단

- 기간: 2015.02.17 ~ 2022.11.26

- 구성: 인덱스, 경기(행사)일시, 구분, 경기(행사), 주최, 관람인원, 수익금(원), 사용일수(일)

```
[71] import pandas as pd

data_location = "C:/Users/SKW/Downloads/Seoul_worldcup_crowd_num.xlsx"
seoul_gym_data = pd.read_excel(data_location)
seoul_gym_data.tail()
```

```
[176]
```

```
...
```

```
[177] seoul_gym_data.shape
```

```
[177]
```

```
... (280, 9)
```

```
[178] seoul_gym_data.info()
```

```
[178]
```

```
... <class 'pandas.core.frame.DataFrame'>
RangeIndex: 280 entries, 0 to 279
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   연번                   280 non-null   int64
1   경기(행사) 일시       280 non-null   datetime64[ns]
2   구 분                  280 non-null   object
3   경기(행사) 종류       280 non-null   object
4   경기(행사) 내용       280 non-null   object
5   주 최                  280 non-null   object
6   관람인원(명)          279 non-null   float64
7   수입금(원)            279 non-null   float64
8   사용일수(일)          280 non-null   int64
dtypes: datetime64[ns](1), float64(2), int64(2), object(4)
memory usage: 19.8+ KB
```

- 컬럼 개수 : 9개

(연번, 경기일시, 구분, 경기종류, 경기내용, 주최, 관람인원, 수입금, 사용일수)

- 행 개수 : 280개

- 관람인원과 수입금 데이터에서 각 한 개의 결측치가 존재

# 데이터 전처리

## 순서

- 1) datetime을 년, 월, 시간 그리고 요일로 분리
- 2) 관람인원을 사용일수로 나눈 '1일 관람인원'이라는 새로운 컬럼 표기
- 3) 경기(행사)내용에서 상대팀을 추출하여 '상대팀'이라는 새로운 컬럼 표기
- 4) 필요없는 컬럼 삭제

datetime을 년, 월, 시간 그리고 요일로 분리  
관람인원을 사용일수로 나눈 '1일 관람인원'이라는 새로운 컬럼 표기



```
import datetime
```

[179]

```
seoul_gym_data["경기요일"] = seoul_gym_data["경기(행사) 일시"].dt.dayofweek  
seoul_gym_data["시작시간"] = seoul_gym_data["경기(행사) 일시"].dt.hour  
seoul_gym_data["년도"] = seoul_gym_data["경기(행사) 일시"].dt.year  
seoul_gym_data["월"] = seoul_gym_data["경기(행사) 일시"].dt.month
```

[180]



```
seoul_gym_data["1일 관람인원"] = seoul_gym_data["관람인원(명)"] / seoul_gym_data["사용일수(일)"]
```

[181]

## 경기(행사)내용에서 상대팀을 추출하여 '상대팀'이라는 새로운 컬럼 표기

```
def extract_opponent(info):
    if info.endswith("):"):
        return info.split('(')[-1].split(')')[0][2:]
    else:
        vs_team = info.split('vs')[1].strip()
        return vs_team.split(' ')[0] if ' ' in vs_team else vs_team

[182]

only_kleague = seoul_gym_data[(seoul_gym_data["경기(행사) 종류"] == "K리그 클래식") | (seoul_gym_data["경기(행사) 종류"] == "K리그1")].copy()

[183]

only_kleague["상대팀"] = only_kleague["경기(행사) 내용"].apply(extract_opponent)
only_kleague

[184]
...

only_kleague["상대팀"].unique()

[185]
...
array(['전북', '제주', '대전', '성남', '전남', '울산', '인천', '부산', '수원', '광주', '포항',
      '상주', '수원FC', '수원삼성', '강원', '대구', '경남', '김천'], dtype=object)

only_kleague["상대팀"] = only_kleague["상대팀"].replace('수원FC', '수원엠펙')

[186]

only_kleague["상대팀"].unique()

[187]
...
array(['전북', '제주', '대전', '성남', '전남', '울산', '인천', '부산', '수원', '광주', '포항',
      '상주', '수원엠펙', '수원삼성', '강원', '대구', '경남', '김천'], dtype=object)
```



## Chapter 03-04

### 필요없는 컬럼 삭제(필요한 컬럼 추출) / 데이터 확인

1.

| 연번  | 경기(행사) 일시 | 구 분                 | 경기(행사) 종류 | 경기(행사) 내용 | 주 최               | 관람인원(명) | 수입금(원)  | 사용일수(일)    | 경기요일 | 시작시간 | 년도  | 월    | 1일 관람인원 | 상대팀     |     |
|-----|-----------|---------------------|-----------|-----------|-------------------|---------|---------|------------|------|------|-----|------|---------|---------|-----|
| 2   | 3         | 2015-03-14 14:00:00 | K리그       | K리그 클래식   | K리그 클래식(vs전북)     | GS스포츠   | 32516.0 | 53033020.0 | 1    | 5    | 14  | 2015 | 3       | 32516.0 | 전북  |
| 5   | 6         | 2015-04-04 14:00:00 | K리그       | K리그 클래식   | K리그 클래식-4R(vs제주)  | GS스포츠   | 22155.0 | 44383420.0 | 1    | 5    | 14  | 2015 | 4       | 22155.0 | 제주  |
| 6   | 7         | 2015-04-15 19:30:00 | K리그       | K리그 클래식   | K리그 클래식-6R(vs대전)  | GS스포츠   | 7186.0  | 32039255.0 | 1    | 2    | 19  | 2015 | 4       | 7186.0  | 대전  |
| 10  | 11        | 2015-05-02 14:00:00 | K리그       | K리그 클래식   | K리그 클래식-9R(vs성남)  | GS스포츠   | 18441.0 | 41630820.0 | 1    | 5    | 14  | 2015 | 5       | 18441.0 | 성남  |
| 12  | 13        | 2015-05-16 14:00:00 | K리그       | K리그 클래식   | K리그 클래식-11R(vs전남) | GS스포츠   | 17819.0 | 41393340.0 | 1    | 5    | 14  | 2015 | 5       | 17819.0 | 전남  |
| ... | ...       | ...                 | ...       | ...       | ...               | ...     | ...     | ...        | ...  | ...  | ... | ...  | ...     | ...     | ... |

2.

```

> only_kleague.info()
[188]
... <class 'pandas.core.frame.DataFrame'>
Index: 146 entries, 2 to 277
Data columns (total 15 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   연번         146 non-null    int64
1   경기(행사) 일시 146 non-null    datetime64[ns]
2   구 분        146 non-null    object
3   경기(행사) 종류 146 non-null    object
4   경기(행사) 내용 146 non-null    object
5   주 최        146 non-null    object
6   관람인원(명)  146 non-null    float64
7   수입금(원)    146 non-null    float64
8   사용일수(일)  146 non-null    int64
9   경기요일      146 non-null    int32
10  시작시간      146 non-null    int32
11  년도          146 non-null    int32
12  월            146 non-null    int32
13  1일 관람인원  146 non-null    float64
14  상대팀        146 non-null    object
dtypes: datetime64[ns](1), float64(3), int32(4), int64(2), object(5)
memory usage: 16.0+ KB

```

3.

```

> selectet_columns = ["경기요일", "년도", "월", "시작시간", "1일 관람인원", "상대팀"]
[189]

refined_data = only_kleague[selectet_columns]
[190]

refined_data.reset_index(drop=True, inplace=True)
refined_data.tail()
[191]
...

```

|     | 경기요일 | 년도   | 월  | 시작시간 | 1일 관람인원 | 상대팀  |
|-----|------|------|----|------|---------|------|
| 141 | 5    | 2022 | 9  | 19   | 10674.0 | 수원옌씨 |
| 142 | 1    | 2022 | 9  | 19   | 5588.0  | 강원   |
| 143 | 5    | 2022 | 10 | 14   | 8111.0  | 대구   |
| 144 | 2    | 2022 | 10 | 19   | 4572.0  | 김천   |
| 145 | 6    | 2022 | 10 | 19   | 7746.0  | 성남   |

**수치형 데이터:**  
경기요일,  
년도,  
월,  
시작시간,  
1일 관람인원

**Object 데이터:**  
상대팀

# 데이터 학습

## 순서

1) 상대팀 데이터를 Word2Vec모델로 벡터화하여 컬럼으로 추가

(하나의 고유값을 50개의 벡터로 변환)

2) 선형회귀모델을 통해 학습

(독립변수: 경기요일, 년도, 월, 시작시간, 상대팀 데이터 대한 벡터)

(종속변수: 1일 방문인원)

## Chapter 04-01

# 상대팀 데이터를 Word2Vec모델로 벡터화하여 컬럼으로 추가

```
# '상대팀' 컬럼의 고유한 값에 대한 리스트
teams = refined_data['상대팀'].unique().tolist()

# Word2Vec 모델 학습
model = Word2Vec([teams], vector_size=50, window=1, min_count=1, workers=4)
```

```
# 각 '상대팀'에 대한 벡터를 데이터프레임에 추가
for i in range(model.vector_size):
    refined_data[f'vector_{i}'] = refined_data['상대팀'].apply(lambda x: model.wv[x][i])

fin_data = refined_data.copy()
```

```
fin_data.tail()
```

```
...
```

|     | 경기요일 | 년도   | 월  | 시작시간 | 1일 관람인원 | 상대팀  | vector_0  | vector_1  | vector_2  | vector_3  | ... | vector_40 | vector_41 | vector_42 | vector_43 | vector_44 | vector_45 | vector_46 | vector_47 | vector_48 | vector_49 |
|-----|------|------|----|------|---------|------|-----------|-----------|-----------|-----------|-----|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| 141 | 5    | 2022 | 9  | 19   | 10674.0 | 수원옌씨 | -0.000543 | -0.017663 | -0.017235 | 0.005600  | ... | -0.005403 | 0.000889  | -0.007075 | -0.000839 | -0.001417 | 0.001646  | 0.016390  | -0.011473 | -0.003319 | 0.011143  |
| 142 | 1    | 2022 | 9  | 19   | 5588.0  | 강원   | -0.017357 | -0.002894 | 0.018959  | -0.015099 | ... | 0.004116  | -0.008007 | -0.016483 | 0.012556  | -0.003898 | -0.001332 | -0.003543 | -0.009071 | 0.008123  | -0.008540 |
| 143 | 5    | 2022 | 10 | 14   | 8111.0  | 대구   | -0.019163 | 0.017883  | 0.008326  | 0.018474  | ... | -0.004798 | 0.007257  | -0.000213 | -0.002401 | -0.002105 | -0.003345 | 0.001212  | 0.008331  | -0.008504 | -0.007671 |
| 144 | 2    | 2022 | 10 | 19   | 4572.0  | 김천   | -0.001074 | 0.000469  | 0.010211  | 0.018027  | ... | -0.019206 | 0.010016  | -0.017530 | -0.008786 | -0.000061 | -0.000593 | -0.015322 | 0.019231  | 0.009967  | 0.018464  |
| 145 | 6    | 2022 | 10 | 19   | 7746.0  | 성남   | 0.000189  | 0.006155  | -0.013625 | -0.002751 | ... | -0.011188 | 0.003461  | -0.001795 | 0.013587  | 0.007947  | 0.009059  | 0.002869  | -0.005400 | -0.008734 | -0.002064 |

5 rows × 56 columns

```
# 독립변수와 종속변수 설정
X = fin_data.drop(['1일 관람인원', '상대팀'], axis=1)
y = fin_data['1일 관람인원']

# 데이터 분할
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 선형회귀 모델 학습
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

데이터를 학습데이터에 80%, 평가데이터에 20% 할당

# 모델 평가

```
[198] # 선형회귀 모델 평가
train_score = regressor.score(X_train, y_train)
test_score = regressor.score(X_test, y_test)
print(f"Training score: {train_score}")
print(f"Test score: {test_score}")

... Training score: 0.5982967605342142
Test score: 0.3635517557869852
```

- 학습 스코어와 평가 스코어가 차이가 있어  
정확도가 떨어짐.

- **랜덤포레스트회귀 모델**을 사용. 결과적으로  
MSE는 높으나, 결정계수가 **0.6**보다 높아  
독립변수들의 종속변수 설명력이 유의하므로 최종 채택.



```
[203] from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

[204] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

[205] # 랜덤 포레스트 회귀 모델 생성 및 학습
regressor = RandomForestRegressor(n_estimators=100, random_state=42)
regressor.fit(X_train, y_train)

... RandomForestRegressor
RandomForestRegressor(random_state=42)

# 테스트 데이터에 대한 예측 수행
y_pred = regressor.predict(X_test)

# 평가 지표 계산: 여기서는 평균 제곱 오차(MSE)를 사용합니다.
mse = mean_squared_error(y_test, y_pred)
print(f"평균 제곱 오차(MSE): {mse}")

# 추가적으로, 결정 계수(R^2)도 계산할 수 있습니다.
r2_score = regressor.score(X_test, y_test)
print(f"결정 계수(R^2): {r2_score}")

[206] ... 평균 제곱 오차(MSE): 29633942.11983547
결정 계수(R^2): 0.603991377813131
```

## 예측

- 2024년 3월 경기 일정 데이터를 랜덤포레스트 모델에 투입
- 학습데이터와 같은 형태를 맞춰주고 투입하여 예측결과 수집

```
[235] pred_data = pd.read_excel("C:/Users/SKW/Desktop/march_seoul.xlsx")
pred_data
```

| 경기(행사) 일시    | 상대팀 |
|--------------|-----|
| 0 2024-03-02 | 광주  |
| 1 2024-03-10 | 인천  |
| 2 2024-03-16 | 제주  |
| 3 2024-03-31 | 강원  |

```
[236] pred_data["경기요일"] = pred_data["경기(행사) 일시"].dt.dayofweek
pred_data["년도"] = pred_data["경기(행사) 일시"].dt.year
pred_data["월"] = pred_data["경기(행사) 일시"].dt.month
pred_data["시작시간"] = pred_data["경기(행사) 일시"].dt.hour
```

```
[237] pred_data
```

| 경기(행사) 일시    | 상대팀 | 경기요일 | 년도   | 월 | 시작시간 |
|--------------|-----|------|------|---|------|
| 0 2024-03-02 | 광주  | 5    | 2024 | 3 | 0    |
| 1 2024-03-10 | 인천  | 6    | 2024 | 3 | 0    |
| 2 2024-03-16 | 제주  | 5    | 2024 | 3 | 0    |
| 3 2024-03-31 | 강원  | 6    | 2024 | 3 | 0    |

```
[238] # '상대팀' 컬럼의 고유한 값에 대한 리스트
teams2 = pred_data['상대팀'].unique().tolist()

# Word2Vec 모델 학습
model2 = Word2Vec([teams2], vector_size=50, window=1, min_count=1, workers=4)
```

```
[239] # 각 '상대팀'에 대한 벡터를 데이터프레임에 추가
for i in range(model2.vector_size):
    pred_data[f'vector_{i}'] = pred_data['상대팀'].apply(lambda x: model.wv[x][i])

fin_pred_data = pred_data.copy()
```

```
[240] fin_pred_data.drop(["경기(행사) 일시", "상대팀"], axis=1, inplace=True)
```

```
[241] fin_pred_data
```

| 경기요일 | 년도 | 월    | 시작시간 | vector_0 | vector_1  | vector_2  | vector_3  | vector_4  | vector_5  | ...       |     |
|------|----|------|------|----------|-----------|-----------|-----------|-----------|-----------|-----------|-----|
| 0    | 5  | 2024 | 3    | 0        | -0.017455 | 0.004260  | -0.001747 | -0.018638 | -0.018856 | -0.002821 | ... |
| 1    | 6  | 2024 | 3    | 0        | 0.008553  | 0.000152  | -0.019169 | -0.019331 | -0.012296 | -0.000257 | ... |
| 2    | 5  | 2024 | 3    | 0        | -0.017235 | 0.007329  | 0.010384  | 0.011484  | 0.014931  | -0.012337 | ... |
| 3    | 6  | 2024 | 3    | 0        | -0.017357 | -0.002894 | 0.018959  | -0.015099 | -0.010716 | 0.018633  | ... |

4 rows × 54 columns

# 예측결과

```
▶ # 모델을 사용하여 예측 수행
prediction = regressor.predict(fin_pred_data)

# 예측 결과 출력
print("예측된 1일 방문인원:", prediction)

[245]
... 예측된 1일 방문인원: [7965.36 9715.53 7189.03 5641.24]
```

- 해당 일정의 방문인원 예측결과가 차례대로 나오는 것을 확인

**감사합니다.**

