FIFA 선수 이적료 예측

김민찬, 고준석, 윤수아, 정재헌

중간보고서

데이터 전처리

- 연속형

1. age : 나이
2. stat_overall : 선수의 현재 능력치 입니다.
3. stat_potential : 선수가 경험 및 노력을 통해 발전할 수 있는 정도입니다.
4. value(예측값) : FIFA가 선정한 선수의 이적 시장 가격 (단위 : 유로) 입니다

- 이산/범주형

1. continent : 선수들의 국적이 포함되어 있는 대륙입니다
2. contract_until(전처리 필요) : 선수의 계약기간이 언제까지인지 나타내어 줍니다
3. position : 선수가 선호하는 포지션입니다. ex) 공격수, 수비수 등
4. prefer_foot : 선수가 선호하는 발입니다. ex) 오른발
5. reputation : 선수가 유명한 정도입니다. ex) 높은 수치일 수록 유명한 선수
6. stat_skill_moves : 선수의 개인기 능력치 입니다.

-정리에서 제외한 변수

1. id : 선수 고유의 아이디
2. name : 이름

데이터 전처리

```
train.describe()
```

	id	age	reputation	stat_overall	stat_potential	stat_skill_moves	value
count	8932.000000	8932.000000	8932.000000	8932.000000	8932.000000	8932.000000	8.932000e+03
mean	7966.775750	25.209136	1.130878	67.091133	71.997201	2.401702	2.778673e+06
std	4844.428521	4.635515	0.423792	6.854910	5.988147	0.776048	5.840982e+06
min	0.000000	16.000000	1.000000	47.000000	48.000000	1.000000	1.000000e+04
25%	3751.750000	21.000000	1.000000	63.000000	68.000000	2.000000	3.750000e+05
50%	7696.500000	25.000000	1.000000	67.000000	72.000000	2.000000	8.250000e+05
75%	12082.250000	28.000000	1.000000	72.000000	76.000000	3.000000	2.600000e+06
max	16948.000000	40.000000	5.000000	94.000000	94.000000	5.000000	1.105000e+08

```
train.isnull().sum()
```

```
id                0
name              0
age               0
continent         0
contract_until    0
position          0
prefer_foot       0
reputation        0
stat_overall      0
stat_potential    0
stat_skill_moves  0
value             0
dtype: int64
```

- 데이터에 있는 null값과 수치값을 살펴봄.

데이터 전처리

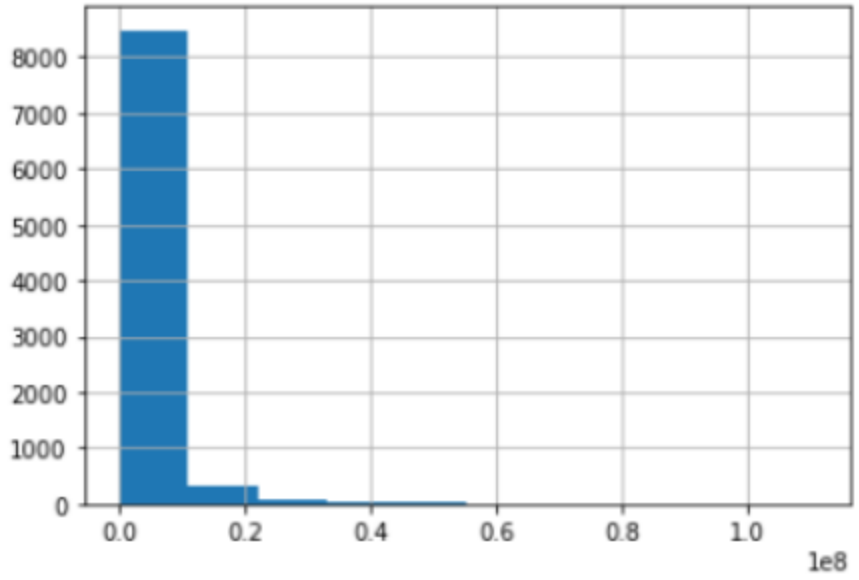
```
class ArrayChanger:
    def __init__(self):
        pass

    def replace_value(self, year, arr):
        for i in range(len(arr)):
            if str(year) in arr[i]:
                arr[i] = str(year)
        return arr

    def replace_all(self, arr):
        print("Replaced array for 2018: ", self.replace_value(2018, arr))
        print("Replaced array for 2019: ", self.replace_value(2019, arr))
        print("Replaced array for 2020: ", self.replace_value(2020, arr))

ac = ArrayChanger()
ac.replace_all(train['contract_until'])
ac.replace_all(test['contract_until'])
```

- 수치형과 범주형이 섞여있는 contract_until 변수를 수치형으로 변환

데이터 전처리

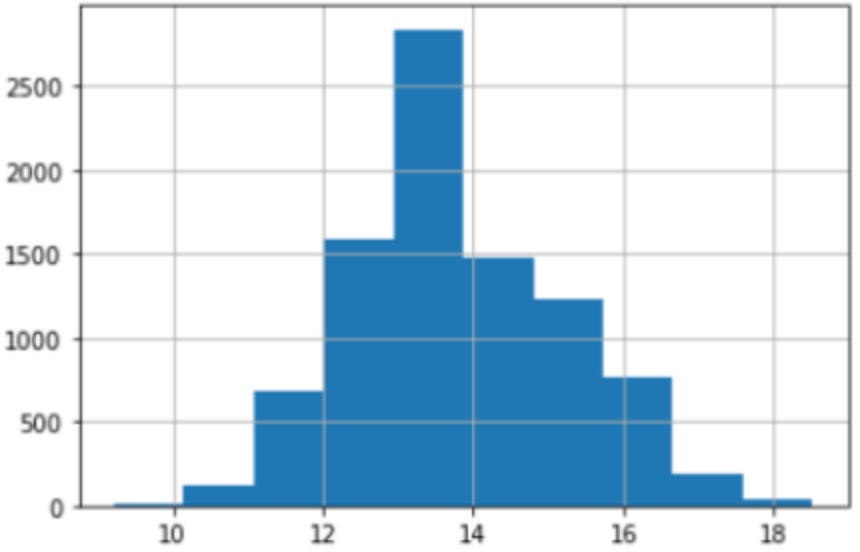
```
train["value"].hist()
```

```
<AxesSubplot:>
```




```
train["value_log"] = np.log(train["value"]+1)
train["value_log"].hist()
```

```
<AxesSubplot:>
```



- 왜도가 있는 value값을 log값을 취해 정규분포형태로 변환

데이터 전처리

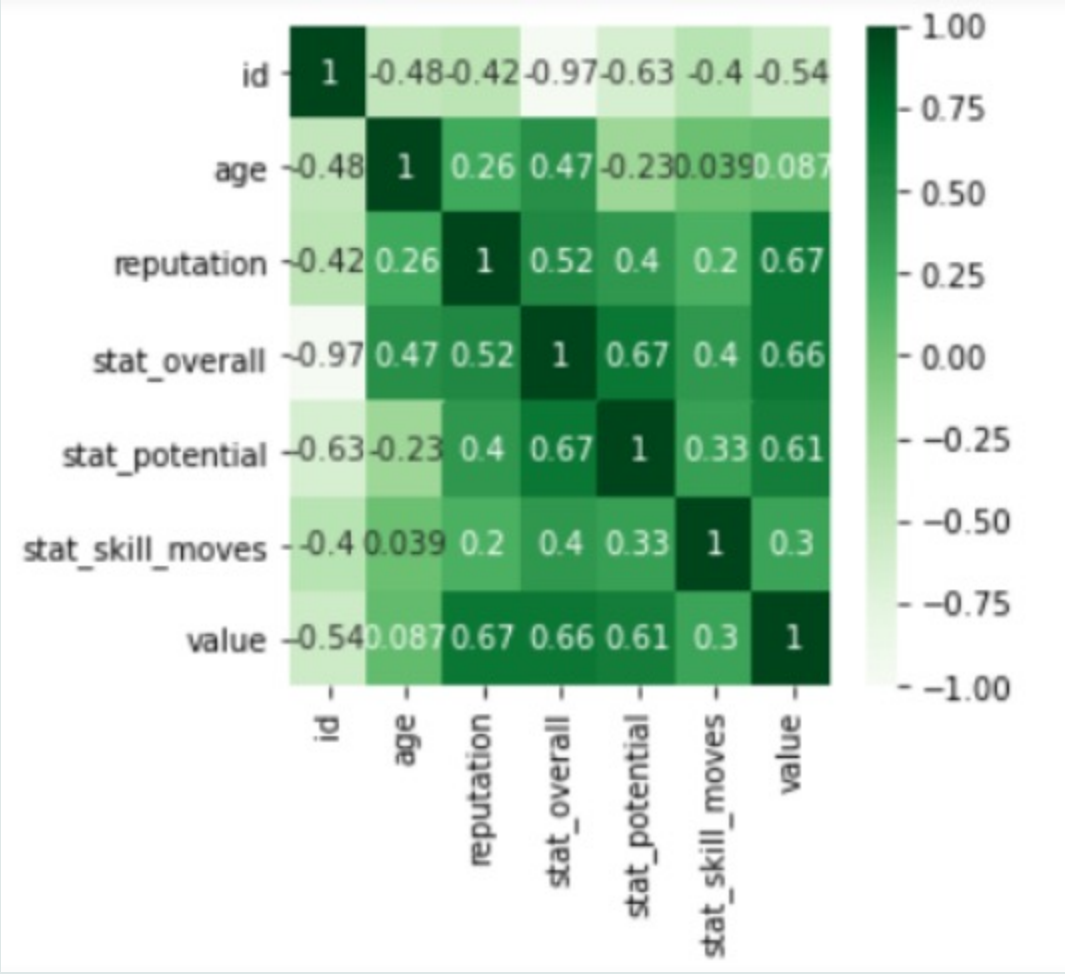
```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer

# 수치형 / 범주형 변수 선언
num_var = X_train.select_dtypes(include=np.number).columns.tolist()
cat_var = X_train.select_dtypes(exclude=np.number).columns.tolist()

# 수치형 변수 전처리를 위한 파이프라인 설정
num_pipeline = Pipeline([
        ('std_scaler', StandardScaler())
    ])

# 범주형 변수 전처리를 위한 파이프라인 설정
cat_pipeline = Pipeline([
        ('encoder', OneHotEncoder(sparse=False, drop='if_binary'))
    ])

# 수치형 / 범주형 변수에 대한 자동 전처리를 위한 파이프라인 설정

full_pipeline = ColumnTransformer([
        ("num", num_pipeline, num_var),
        ("cat", cat_pipeline, cat_var)])

X_train_pretained = full_pipeline.fit_transform(X_train)
```

-파이프라인 이용

- StandardScaler를 이용해 변수들의 정규화 진행

-범주형 변수는 OneHotEncoder를 이용해 수치형변수로 변환

데이터 전처리



- 상관관계 살펴보기

1. correlation이 높은 3가지 등 correlation을 본 후 insight를 도출해서 분석을 진행

2. 범주형 변수를 분석함에 있어 막대그래프 등을 이용해서 분석을 진행

3. age 변수에 대한 전처리를 다시 진행해야할 것으로 보임

파생변수 생성

1. spd*re 변수

- 축구 선수의 포지션은 그들의 역할과 통계에 영향을 미침
- 예를 들어, 공격수와 수비수는 서로 다른 통계를 갖고 있기 때문에, 축구 선수의 포지션에 따라 통계 파생 변수를 만들 수 있음
- 각 포지션의 통계의 평균값을 계산하여 파생 변수를 만들 수 있음
- 예를 들어, 다음 코드는 선수 포지션별로 stat_potential 및 value의 평균값을 계산하여 파생 변수를 생성

```
position_stat = train.groupby('position')[['stat_potential']].mean()
train = pd.merge(train, position_stat, on='position', how='left', suffixes=('', '_me
an'))
```

```
train['stat_potential_dif']= train['stat_potential']-train['stat_potential_mean']
```

```
train['stat_potential_dif^2']=train['stat_potential_dif'].pow(2)
```

-각 선수의 stat_potential을 각 포지션 별 stat_potential의 평균값에서 뺌 (position에서 위상을 표현)

-음수 값을 가지기 때문에 제곱

파생변수 생성

1. spd*re 변수

```
train['spd*re']=train['stat_potential_dif^2']*train['reputation']
```

- 자신이 속한 position에서 위상을 표현해내는 stat_potential_dif^2 변수와 자신의 위상을 표현해내는데 긴밀한 관련이 있는 reputation 변수 둘을 *로 연결해줌

<결과>

- 선수 가치와 상관관계를 분석했을 때 0.803132라는 수치로서 긴밀한 관계에 있다고 도출이 됨

파생변수 생성

2.sod*re 변수

- spd*re와 동일한 방식으로 진행

- 각 포지션의 통계의 평균값을 계산하여 파생 변수를 만들 수 있음

- 예를 들어, 다음 코드는 선수 포지션별로 stat_overall 및 value의 평균값을 계산하여 파생 변수를 생성

```
position_stat = train.groupby('position')[['stat_overall']].mean()
train = pd.merge(train, position_stat, on='position', how='left', suffixes=('', '_me
an'))
```

```
train['stat_overall_dif']= train['stat_overall']-train['stat_overall_mean']
```

```
train['sod*re']=train['stat_overall_dif^2']*train['reputation']
```

- 각 선수의 stat_overall을 각 포지션 별 stat_overall의 평균값에서 뺌 (자신이 속한 position에서 실력을 표현해내기 위해)

- 음수 값을 가지기 때문에 제곱

파생변수 생성

2.sod*re 변수

```
train['sod*re']=train['stat_overall_dif^2']*train['reputation']
```

- 자신이 속한 position에서 실력을 표현해내는 stat_overall_dif^2 변수와 자신의 실력을 표현해내는데 긴밀한 관련이 있는 reputation 변수 둘을 *로 연결

<결과>

- 선수 가치와 상관관계를 분석했을 때 0.791491라는 수치로서 긴밀한 관계에 있다고 도출이 됨

파생변수 생성

3. age_group 변수

- 선수의 경력 단계를 나타내는 연령 그룹 변수를 만들 수 있음
- 예를 들어, 다음 기준에 따라 연령 그룹을 만들 수 있습니다
- 20세 미만, 20-25세, 26-30세, 30세 이상.
- 아래 코드에서 bins는 각 그룹의 경계값을 나타내는 리스트이고, labels는 각 그룹에 해당하는 이름을 나타내는 리스트
- pd.cut 함수를 사용하여 age 변수를 bins로 구분하고, labels로 라벨링하여 age_group 변수에 저장
- 아래 코드는 연령 그룹을 나타내는 파생 변수를 만드는 예시

```
train['age_group'] = pd.cut(train['age'],
                            bins=[0, 20, 25, 30, 100],
                            labels=['under 20', '20-25', '26-30', 'over 30'])
```

```
age_group_level_map = {
    'under 20' :    0.4,
    '20-25'    :    0.3,
    '26-30'       :    0.2,
    'over 30'        :    0.1,
    }

train['age_group_level'] = train['age_group'].map(age_group_level_map)
```

- ordinal encoding을 통해 각 그룹의 수치를 조정할 수 있음

- 우선 임의의 수치로 설정

파생변수 생성

4. contract_duration 변수

```
import pandas as pd
from datetime import datetime

# 'contract_until' 컬럼의 값을 문자열로 변환하고 연도만 추출하여 'contract_year' 컬럼
생성
train['contract_year'] = train['contract_until'].str[-4:]

# 'contract_year' 컬럼의 값이 연도만 있는 경우, 월과 일을 1로 설정
for i, year in train[train['contract_until'].str.len() == 4]['contract_year'].iterit
ems():
    train.at[i, 'contract_until'] = datetime.strptime(year + '-01-01', '%Y-%m-%d')

# 'contract_until' 컬럼의 값을 datetime 형태로 변환
train['contract_until'] = pd.to_datetime(train['contract_until'])

# 'contract_until' 컬럼에서 년, 월, 일 정보 추출
train['contract_year'] = train['contract_until'].dt.year
train['contract_month'] = train['contract_until'].dt.month
train['contract_day'] = train['contract_until'].dt.day
```

변수설명

- 선수의 계약 기간을 나타내는 파생 변수를 만들 수 있음
- 이 변수는 현재 연도에서 계약 종료 연도를 뺌으로써 계산
- 여기서 2018은 현재 연도로 설정한 값
  - 따라서 계약이 2018년에 끝나는 경우, 계약 기간 파생 변수의 값은 0
  - 계약이 2017년에 끝나는 경우, 값은 -1
  - 계약이 2019년에 끝나는 경우, 값은 1

코드설명

-계약기간 변수를 datetime으로 변환

파생변수 생성

5. 계약 기간만을 이용한 파생변수들

```
import pandas as pd

# 계약 기간을 문자열로 변환
train['contract_until'] = train['contract_until'].astype(str)

# 계약 기간이 '2023'으로 끝나는 경우를 2023-01-01로 바꿔줌
train['contract_until'] = train['contract_until'].apply(lambda x: x + ', 01' if len
(x) == 4 else x)

# 문자열을 datetime 형태로 변환
train['contract_until'] = pd.to_datetime(train['contract_until'], format='%Y-%m-%d')

# 계약 기간과 관련된 파생 변수 생성
today = pd.to_datetime('today')
train['contract_length'] = (train['contract_until'] - today).dt.days
train['contract_length_years'] = train['contract_length'] / 365
train['contract_length_months'] = train['contract_length'] / 30
train['is_long_contract'] = (train['contract_length_years'] >= 3).astype(int)
train['is_short_contract'] = (train['contract_length_years'] <= 1).astype(int)
```

변수설명

- 예를 들어, 선수의 계약 기간이 길수록 그 선수가 팀에서 오랫동안 활약할 가능성이 높아질 것
- 이런 가정에 따라, 계약 기간을 이용해 다음과 같은 파생 변수들을 생성 가능

1. "contract_length": 계약 기간을 나타내는 변수
2. "contract_length_years": 계약 기간을 연 단위로 표시한 변수
3. "contract_length_months": 계약 기간을 월 단위로 표시한 변수
4. "is_long_contract": 계약 기간이 3년 이상인 경우 1, 그렇지 않은 경우 0으로 표시한 변수
5. "is_short_contract": 계약 기간이 1년 이하인 경우 1, 그렇지 않은 경우 0으로 표시한 변수

Thank You