
K-107
BIG DATA
FIFA 빅데이터 분석을 통한
선수 WAGE 예측 프로그램 개발 및 활용 방안
FIFA
Photo by K-107

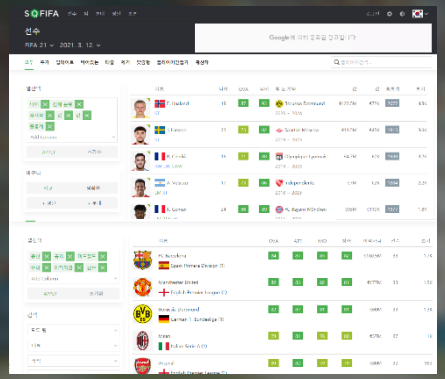
K-107
BIG DATA

목차

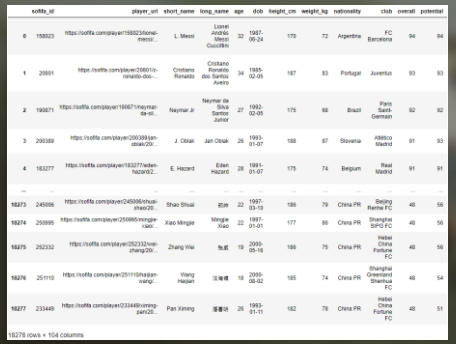
FIFA 빅데이터 분석을 통한
선수 WAGE 예측 프로그램 개발 및 활용 방안

FIFA 데이터 소개

K-107 BIG DATA



- ✓ sofifa.com에서 매주 업데이트되는 축구 선수 데이터.
- ✓ 실제 축구 선수들, 관계자들이 참고할 정도로 높은 데이터 정확도로 유명하다.



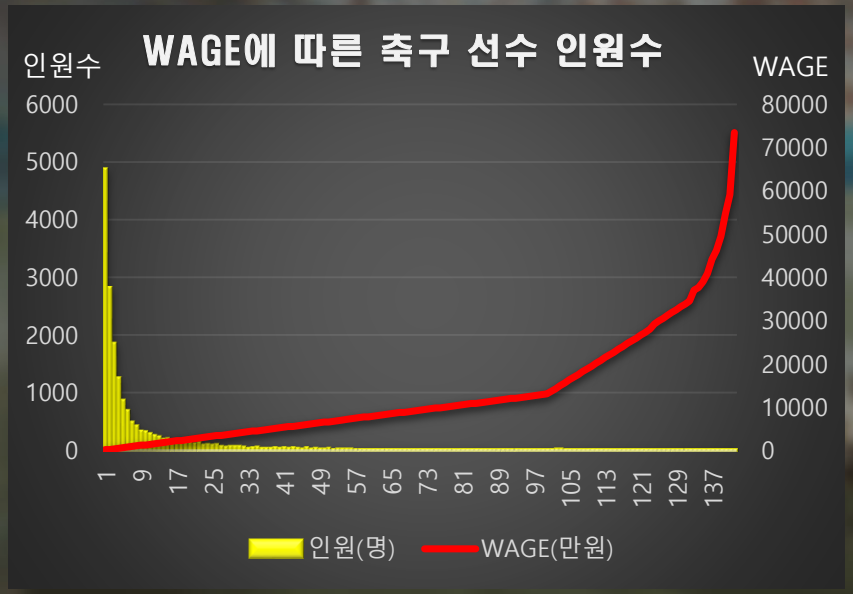
- ✓ Kaggle에서 sofifa 2019, 2020데이터를 다운받아 Python으로 분석을 실시.
- ✓ 약 1만8천명의 축구선수, 100여 가지 선수 정보를 나타내는 데이터이다.

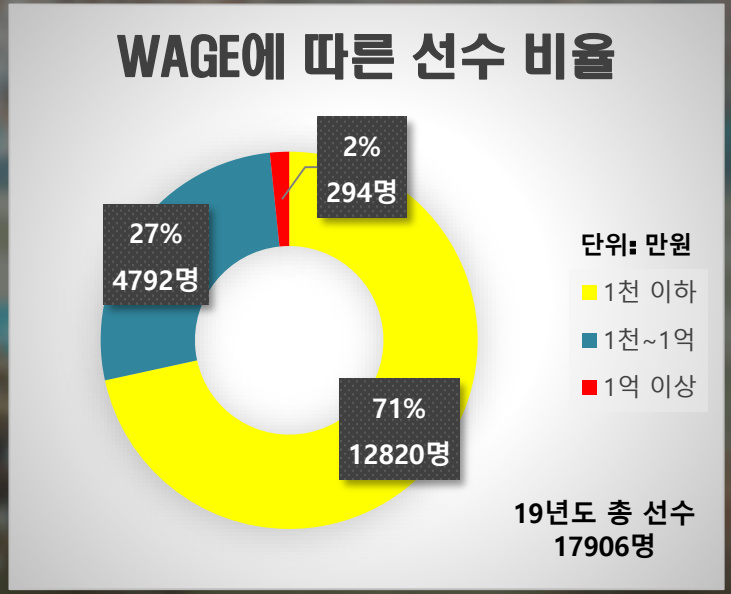
주요 정보 컬럼	
ID	선수 고유번호
Name	이름
Age	나이
Nationality	국적
Club	소속 팀
Position	포지션
Overall	현재 능력치
Potential	미래 능력치
Wage	주급
Value	예상 이적료
International Reputation	국제적 명성도
Reactions	반응 속도
Composure	침착성
Loaned From	임대 클럽
Contract Valid Until	계약 기간
. . . .	. . . .

선수 정보 중 WAGE를 예측

K-107 BIG DATA

WAGE 지급 방식의 편향성

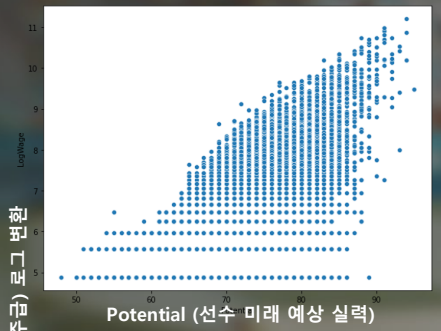


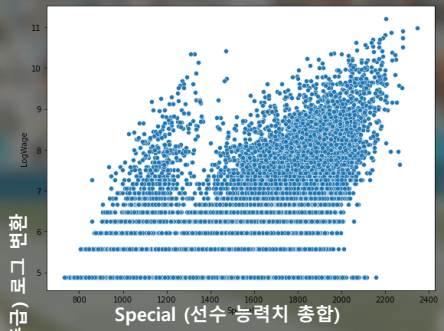


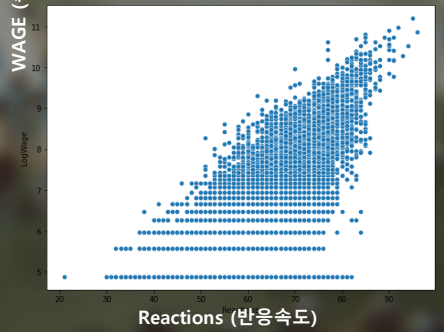
✓ 소수의 인원에게 WAGE가 쏠려 있다.

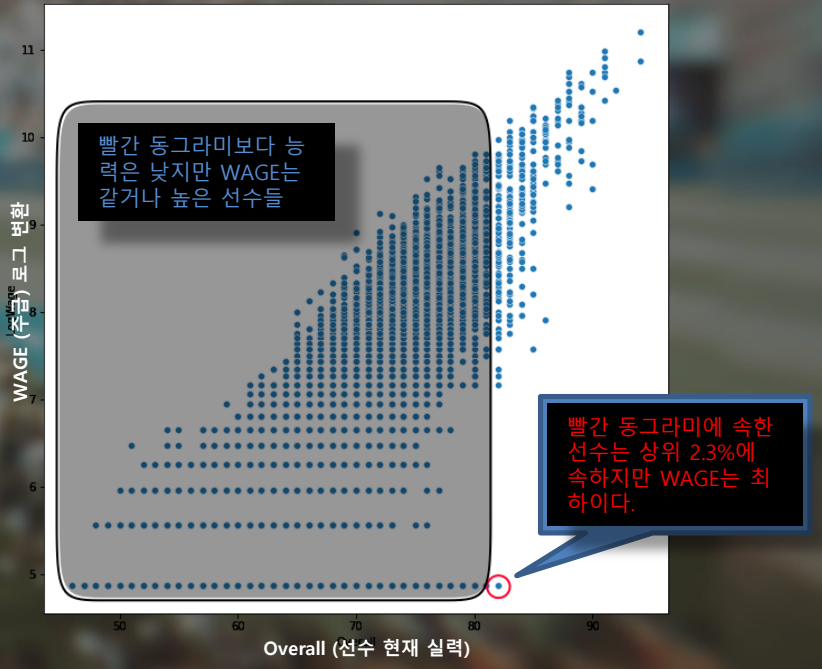
K-107 BIG DATA

WAGE 지급 방식의 불공정성





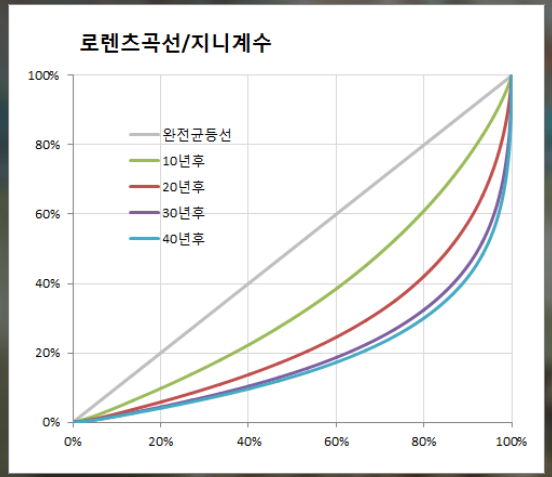


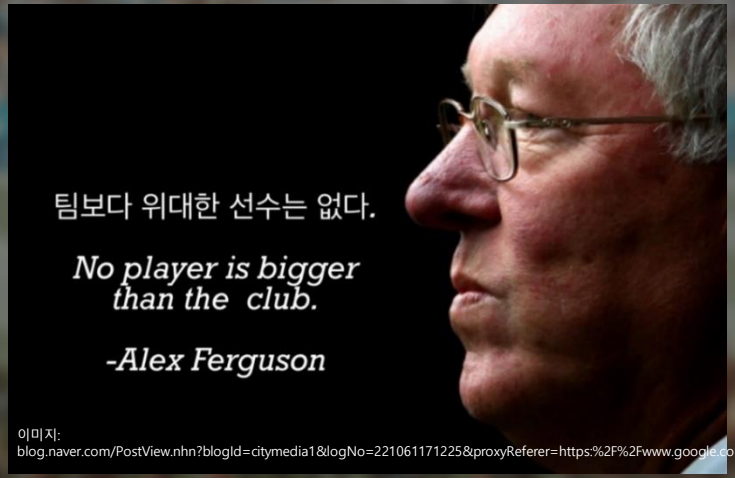


✓ 선수들이 자신의 능력에 맞는 WAGE를 받는다고 볼 수 없음.

K-107 BIG DATA

편향성과 불공정성을 해결하기 위한 아이디어





이미지:
blog.naver.com/PostView.nhn?blogId=citymedia1&logNo=221061171225&proxyReferer=https:%2F%2Fwww.google.com%2F

실물 경제로 단순화하여 모델링	
선수	시민
팀	사는 지역
WAGE	재산
선수의 이산형 정보들을 시민의 소유 물건으로 생각	
Real Face	자동차
International Reputation	TV
Position	옷
Loaned From	가방
Age	시계
. . . .	. . . .

1. 하위 WAGE 선수들의 다양한 능력 수치를 낮은 값으로 줄여야 함.
2. 상위 WAGE 선수들은 적은 능력 수치 증가에도 WAGE가 기하급수적으로 증가해야 함.
3. WAGE 지급 주체인 팀을 알아야 예측이 가능함.

아이디어
실현 과정

K-107
BIG DATA

WAGE 예측을 위한 과정

1. ‘팀WAGE표’ 제작

표를 통해 팀이 소속 선수에게 어떻게 WAGE를 지급하는지 수치화.
선수들을 능력보다 소속된 **팀**으로 우선 평가.

⬇

2. 공식 적용

가중치를 곱한 선수 능력치가
‘팀WAGE표’에 따라 조정되는 **공식** 제작.

K-107 BIG DATA

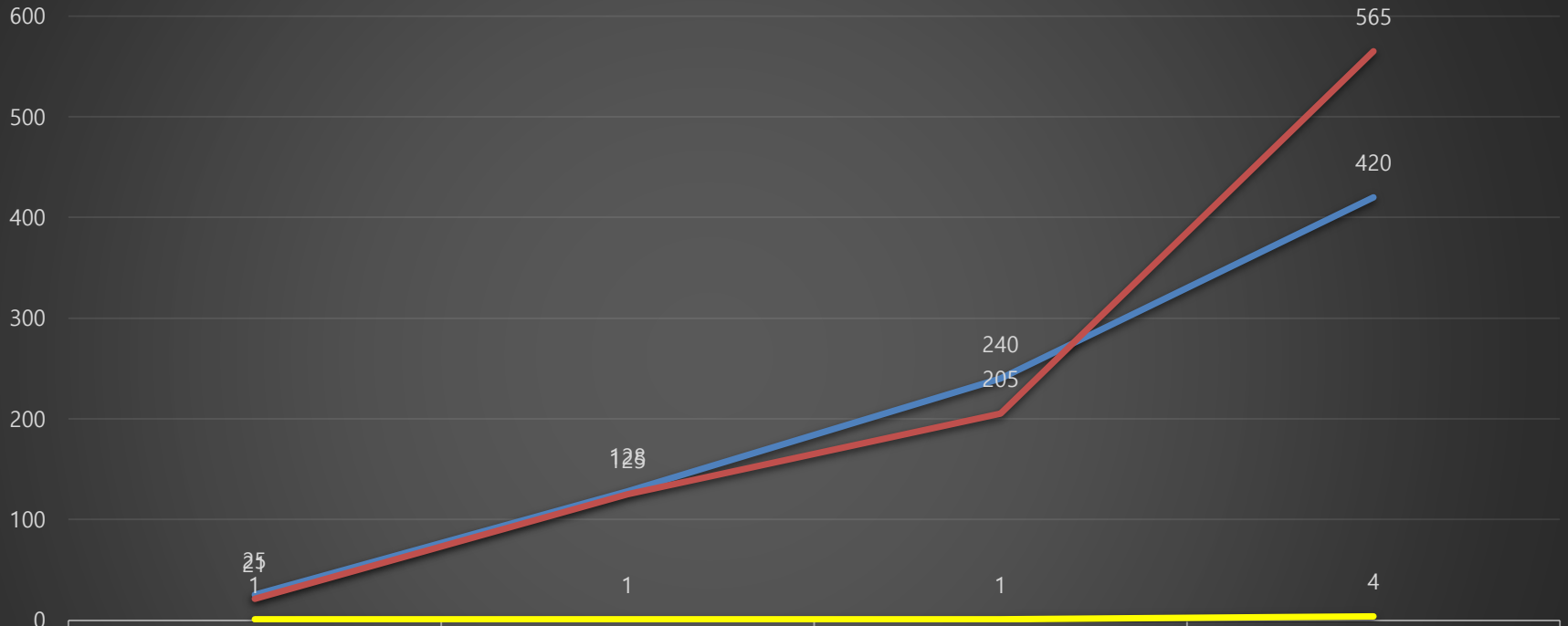
'팀WAGE표'

	ClubWageMean	ClubWeight25	ClubWeight50	ClubWeight75	ClubWeight100	ClubWageMin
	팀 WAGE 평균 (총액으로 생각)	전체 선수 최저 WAGE 대비 팀 내 하위 25%인 선수의 주급은 몇 배인가?	전체 선수 최저 WAGE 대비 팀 내 하위 50%인 선수의 주급은 몇 배인가?	전체 선수 최저 WAGE 대비 팀 내 하위 75%인 선수의 주급은 몇 배인가?	전체 선수 최저 WAGE 대비 팀 내 하위 100%인 선수의 주급은 몇 배인가?	팀 내 최소 WAGE
Real Madrid	20335	24.5	127.5	240	420	1170
FC Barcelona	17654	21	125	205	565	520
. . . .	. . . .	. . . .	. . . .	. . . .	. . . .	. . . .
KFC Uerdinge 05	325	1	2	3	6	130
CD Huachipato	245	1	1	2	7	130
FSV Zwickau	137	1	1	1	2	130
Cork City	130	1	1	1	1	130

팀

‘팀WAGE표’ 그래프

K-107 BIG DATA



	ClubWeight25	ClubWeight50	ClubWeight75	ClubWeight100
Real Madrid	25	128	240	420
FC Barcelona	21	125	205	565
최하위팀	1	1	1	4

Real Madrid FC Barcelona 최하위팀

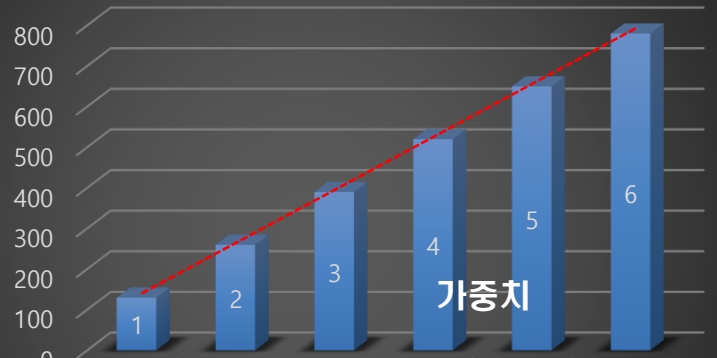
K-107 BIG DATA

선수에게 적용 가능한 가중치 5가지 (5 Weights)

Age, International Reputation, Position, Real Face, Loaned From

1. Age

실물 경제의 시계로 생각

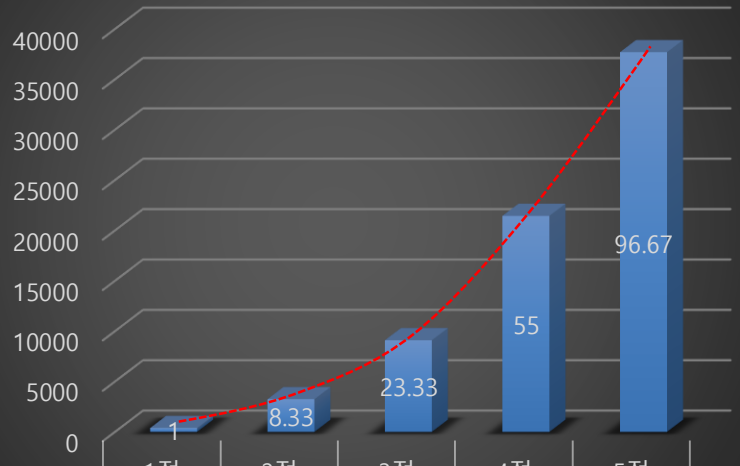


	16~19, 42~47세	20,21,40,41세	22~24세,37~39세	25,26,35,36세	27~29, 32~34세	30,31세
Wage 중앙값	130	260	390	520	650	780

만원

2. International Reputation

실물 경제의 TV로 생각



	1점	2점	3점	4점	5점
Wage 중앙값	390	3250	9100	21450	37700

만원

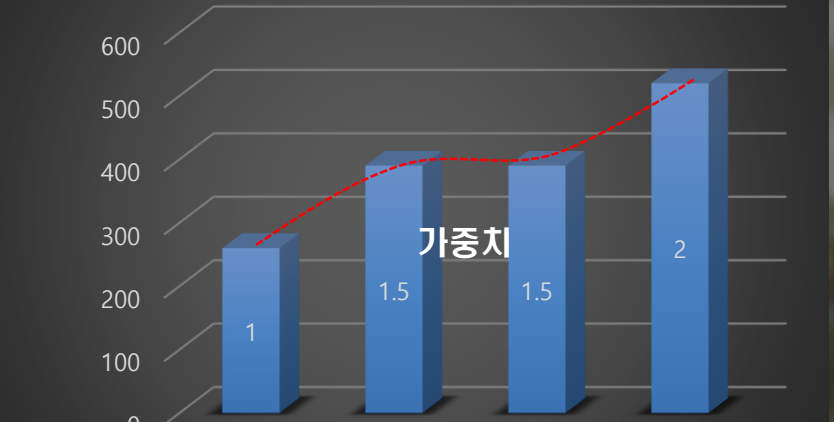
K-107 BIG DATA

선수에게 적용 가능한 가중치 5가지 (5 Weights)

Age, International Reputation, Position, Real Face, Loaned From

3.Position

실물 경제의 옷으로 생각

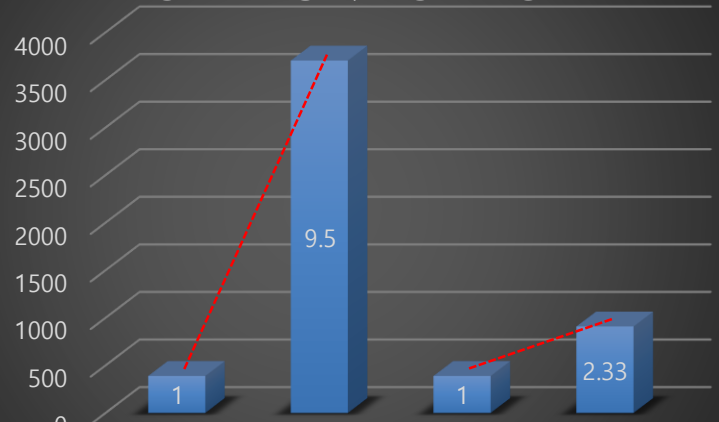


	GK	DF	MF	FW
Wage 중앙값	260	390	390	520

만원

4.Real Face, 5.Loaned From

실물 경제의 자동차, 가방으로 생각



	No	Yes	No	Yes
Wage 중앙값	390	3705	390	910

만원

K-107 BIG DATA

WAGE 예측을 위한 공식 1

팀에 따라 선수의 능력치를 조정하기 위해

$$PredictWage = \beta\left(\frac{\alpha - 1}{\alpha}\right)^2 + \frac{\gamma}{\alpha}$$

α: *ClubWeight100, 75, 50, 25 중 1개*

β: *선수의 능력치 중 1개*

γ: 전체 모든 β의 최솟값

⬇

실제 계산에서는

α: *ClubWeight*100,

β: 5 *Weights* * *ClubWageMin*

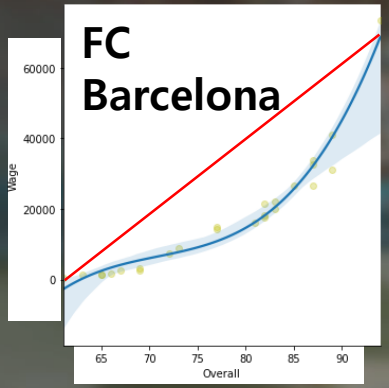
을 사용.

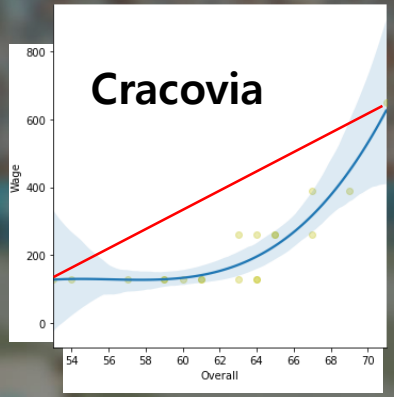
(PPT에선 이해를 위해 간략하게 표현. 실제 적용 공식은 약간 상이함.)

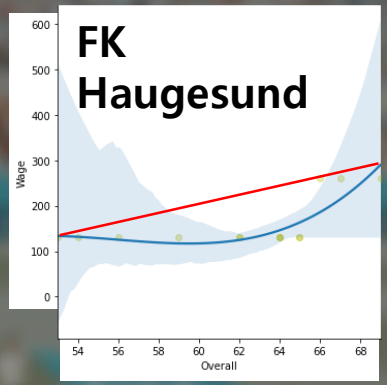
공식 계산

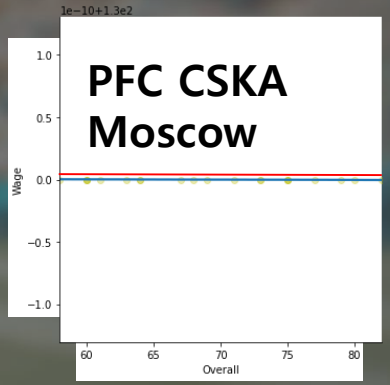
K-107 BIG DATA

공식 1에 대한 그래프를 통한 설명









$$PredictWage = \beta\left(\frac{565-1}{565}\right)^2 + \frac{\gamma}{565}$$

$$\| \quad 0.996$$

$$PredictWage = \beta\left(\frac{5-1}{5}\right)^2 + \frac{\gamma}{5}$$

$$\| \quad 0.64$$

$$PredictWage = \beta\left(\frac{2-1}{2}\right)^2 + \frac{\gamma}{2}$$

$$\| \quad 0.25$$

$$PredictWage = \beta\left(\frac{1-1}{1}\right)^2 + \frac{\gamma}{1}$$

$$\| \quad 0$$

- ✓ 팀 WAGE 그래프에 따라 β가 반영되는 것이 다르다.
- ✓ β값이 팀에 따라 조정되며 최소 β 인 γ 는 보장하는 공식이다.

K-107 BIG DATA

WAGE 예측을 위한 공식 2

공식 1에서 조정되지 못한 선수 중 하위 WAGE선수들을 재조정하기 위해

if (ClubWeight100=1 & 선수의 *PredictWage* <= 팀 내 하위 100%(전체)): *PredictWage* = -20

elif (ClubWeight75=1 & 선수의 *PredictWage* <= 팀 내 하위 75%): *PredictWage* = -20

elif (ClubWeight50=1 & 선수의 *PredictWage* <= 팀 내 하위 50%): *PredictWage* = -20

elif (ClubWeight25=1 & 선수의 *PredictWage* <= 팀 내 하위 25%): *PredictWage* = -20

else: pass

- ✓ *PredictWage*를 '팀WAGE표'에 맞게 다시 조정하는 과정이다.
- ✓ -20 수치는 그래프와 상관관계를 확인하여 나온 값이다.

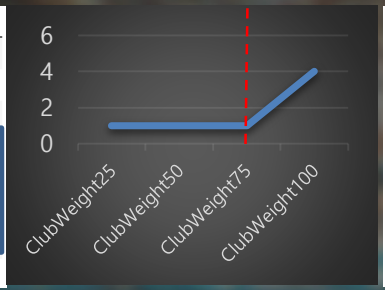
공식 계산

K-107 BIG DATA

공식 2에 대한 데이터 프레임을 통한 설명.

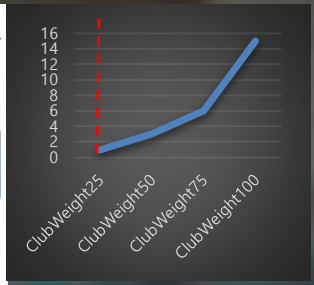
Cf. 전체 축구 선수 최저 LogWage: 48.751973

	LogWage	PredictLogWage
count	16.000000	16.000000
mean	50.729090	87.375000
std	4.435778	10.397916
min	48.751973	78.000000
25%	48.751973	83.500000
50%	48.751973	85.500000
75%	48.751973	91.000000
max	62.557500	122.000000



	LogWage	PredictLogWage
count	20.000000	20.000000
mean	50.879468	79.800000
std	3.914737	1.852452
min	48.751973	76.000000
25%	48.751973	78.000000
50%	48.751973	80.000000
75%	50.475281	81.000000
max	59.687076	83.000000

	LogWage	PredictLogWage
count	18.000000	18.000000
mean	53.415718	89.944444
std	5.237938	3.826721
min	48.751973	82.000000
25%	48.751973	87.000000
50%	52.198589	91.000000
75%	55.645204	93.000000
max	62.557500	94.000000



	Club	Name	LogWage	PredictLogWage
10512	Wellington Phoenix	C. Elliot	48.751973	78
11943	Wellington Phoenix	L. Cacace	48.751973	78
13420	Wellington Phoenix	S. Singh	48.751973	78
11711	Wellington Phoenix	B. Waine	48.751973	79
11730	Wellington Phoenix	A. Rufer	48.751973	85
12888	Wellington Phoenix	D. Fox	48.751973	85
13217	Wellington Phoenix	F. Kurto	48.751973	85
13[illegible]	Wellington Phoenix	R. Lowry	48.751973	85
11[illegible]	Wellington Phoenix	L. Fenton	48.751973	86
13[illegible]	Wellington Phoenix	A. Durante	48.751973	86
13763	Wellington Phoenix	T. Doyle	48.751973	86
5770	Wellington Phoenix	R. Krishna	62.557500	91
11249	Wellington Phoenix	D. Williams	48.751973	91
11333	Wellington Phoenix	N. Burns	48.751973	91
7931	Wellington Phoenix	M. Kopczyński	59.687076	92
9126	Wellington Phoenix	S. Taylor	55.645204	122

피해자

-20

	Club	Name	LogWage	PredictLogWage
13425	Incheon United FC	Kim Jin Ya	48.751973	76
11350	Incheon United FC	Lim Eun Soo	48.751973	78
12762	Incheon United FC	Jeong Dong Yun	48.751973	78
11870	Incheon United FC	Kim Dong Min	48.751973	78
13527	Incheon United FC	Jung San	48.751973	78
10[illegible]	Incheon United FC	Kim Jung Ho	48.751973	78
10[illegible]	Incheon United FC	Lee Jeong Bin	48.751973	78
11[illegible]	Incheon United FC	Lee Jin Hyung	48.751973	79
13587	Incheon United FC	Yun Sang Ho	48.751973	80
8560	Incheon United FC	Kwak Hae Seong	55.645204	80
8551	Incheon United FC	Kim Yong Hwan	55.645204	80
11152	Incheon United FC	Kim Dae Jung	48.751973	80
11636	Incheon United FC	Kang Ji Yong	48.751973	81
12296	Incheon United FC	K. Appiah-Kubi	48.751973	81
12994	Incheon United FC	Ko Seul Ki	48.751973	81
13122	Incheon United FC	Lee Yun Pyo	48.751973	81
9099	Incheon United FC	Kim Dong Suk	55.645204	82
12236	Incheon United FC	Choi Jong Hoan	48.751973	82
7104	Incheon United FC	G. Bunoza	59.687076	82
7622	Incheon United FC	Nam Joon Jae	59.687076	83

피해자

-20

	Club	Name	LogWage	PredictLogWage
12722	1. FC Kaiserslautern	J. Scholz	48.751973	82
11143	1. FC Kaiserslautern	F. Botiseriu	48.751973	82
12237	1. FC Kaiserslautern	C. Sickinger	48.751973	87
12043	1. FC Kaiserslautern	L. Gottwalt	48.751973	87
11577	1. FC Kaiserslautern	T. Bergmann	48.751973	87
10855	1. FC Kaiserslautern	D. Schad	48.751973	87
14299	1. FC Kaiserslautern	C. Kühlwetter	48.751973	90
13332	1. FC Kaiserslautern	O. Özdemir	48.751973	90
14261	1. FC Kaiserslautern	W. Hesl	48.751973	90
6686	1. FC Kaiserslautern	L. Spalvis	62.557500	92
7812	1. FC Kaiserslautern	K. Kraus	59.687076	92
5830	1. FC Kaiserslautern	M. Albæk	62.557500	93
9715	1. FC Kaiserslautern	F. Dick	55.645204	93
9639	1. FC Kaiserslautern	J. Löhmannsröben	55.645204	93
10055	1. FC Kaiserslautern	C. Hemlein	55.645204	93
10390	1. FC Kaiserslautern	A. Hainault	55.645204	93
8312	1. FC Kaiserslautern	J. Biada	55.645204	94
7774	1. FC Kaiserslautern	T. Thiele	59.687076	94

-20

K-107 BIG DATA

WAGE 예측을 위한 공식 3

공식 2 계산에서 무고하게 조정된 피해자 선수들을 위한 공식

공식 2에서 계산된 *PredictWage*를 선수의 *Total*(선수 능력 통합 파생변수)과 *Value*를 이용해 계산.

*PredictWage1 = PredictWage – (전체 선수 Total 최댓값-선수 Total)*7*
*– (전체 선수 Value 최댓값- 선수 Value)*10*

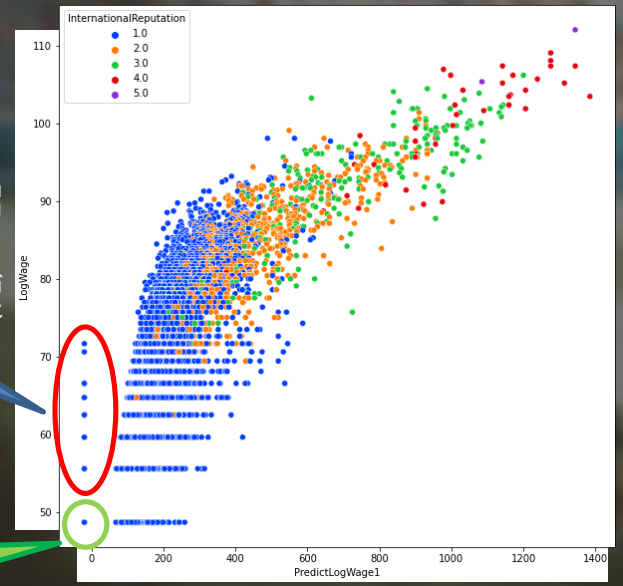
- ✓ *PredictWage1*을 선수 능력을 약간 반영하여 재조정하는 과정이다.
- ✓ 7, 10 수치는 그래프와 상관관계를 확인하여 나온 값이다.

공식 계산

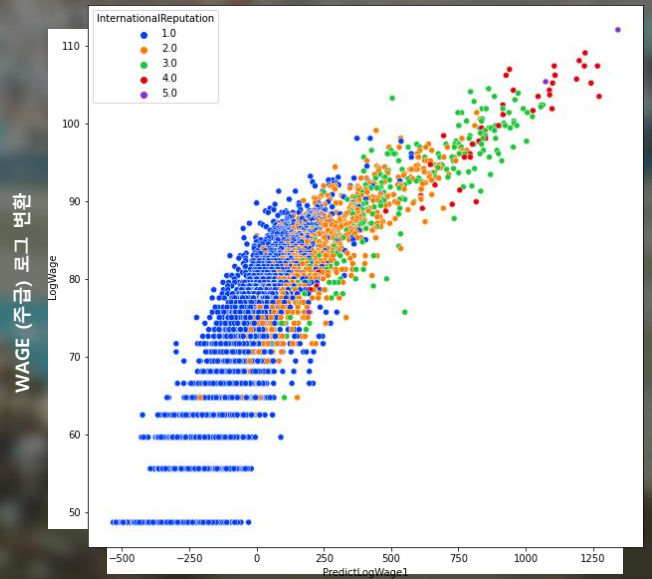
K-107 BIG DATA

공식 3에 대한 산점도를 통한 설명.

공식 3 적용 전



공식 3 적용 후



✓ 공식 2 적용 후 생긴 비정상적인 산점도를 공식 3으로 해소

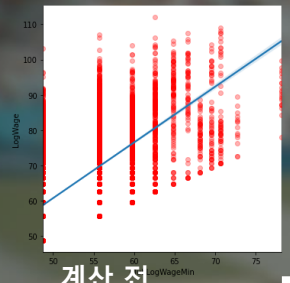
공식 계산

K-107 BIG DATA

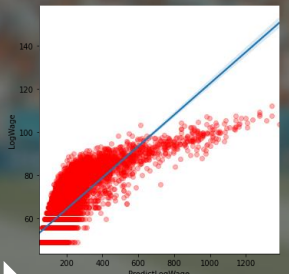
‘팀WAGE표’+ 선수의 가중치 5개 + 공식1,2,3 을 이용한 실제 ClubWageMin에서 PredictWage1으로의 변화

ClubLogWageMin의 결정계수

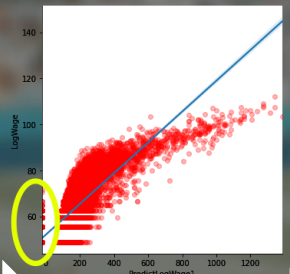
```
XGBRegressor(n_estimators=500)
로그 변환된 MSE:    92.973
로그 변환된 RMSE:   9.642
결정계수 R2:        0.37
```



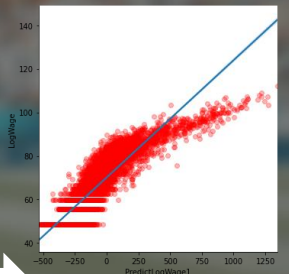
계산 전 ClubWageMin



공식 1 계산 후



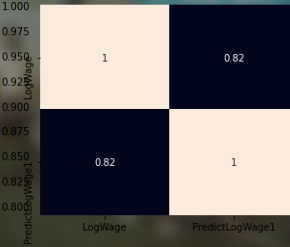
공식 2 적용 후

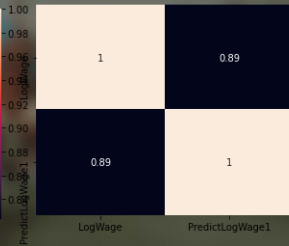


공식 3 계산 후

PredictedLogWage1의 결정계수

```
XGBRegressor(n_estimators=500)
로그 변환된 MSE:    20.863
로그 변환된 RMSE:   4.568
결정계수 R2:        0.859
```

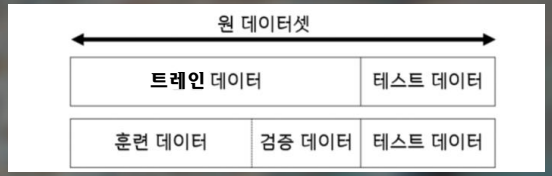




✓ *모든 공식의 계산이 완료된 PredictWage1 컬럼 하나만으로도 xgb 85.9%의 예측률을 보임.*

트레인&테스트 데이터 사전 분리

- ✓ 19년도 데이터의 80%를 트레인 데이터, 20%를 테스트 데이터로 결정.
- ✓ 먼저 테스트 데이터를 분리하지 않으면 독립변수와 종속변수의 관계가 반영된 그래프, 평균 등으로 테스트 데이터의 답이 나타나버림.
- ✓ 간단한 처리(NaN값 제거, 단위 변경 등)만 해준 뒤 바로 분리하고 테스트 데이터는 피클로 저장.
- ✓ 트레인 데이터만을 가지고 아아디어 적용 후, 트레인 데이터의 80%를 훈련 데이터, 20%를 검증 데이터로 분리하여 모델을 훈련, 검증하고 난 뒤 피클로 저장했던 테스트 데이터로 예측을 진행.



최종 선정 컬럼 13가지

원본, 범주화, 파생변수로 구분		공식 적용 내용에 따른 구분	
원본 5	InternationalReputation, RealFace, LoanedFrom, Overall, LogValue	AgeWeight, InternationalReputation, RealFace, LoanedFrom, PositionClassWeight	가중치 5
범주화 2	AgeWeight, PositionClassWeight	Overall, LogValue	선수 능력 2
파생변수 6	ClubLogWageMean, ClubWeight100, ClubWeight75, ClubWeight50, ClubWeight25, PredictLogWage1	ClubLogWageMean, ClubWeight100, ClubWeight75, ClubWeight50, ClubWeight25'	팀WAGE표 5
		PredictLogWage1	공식 계산 1

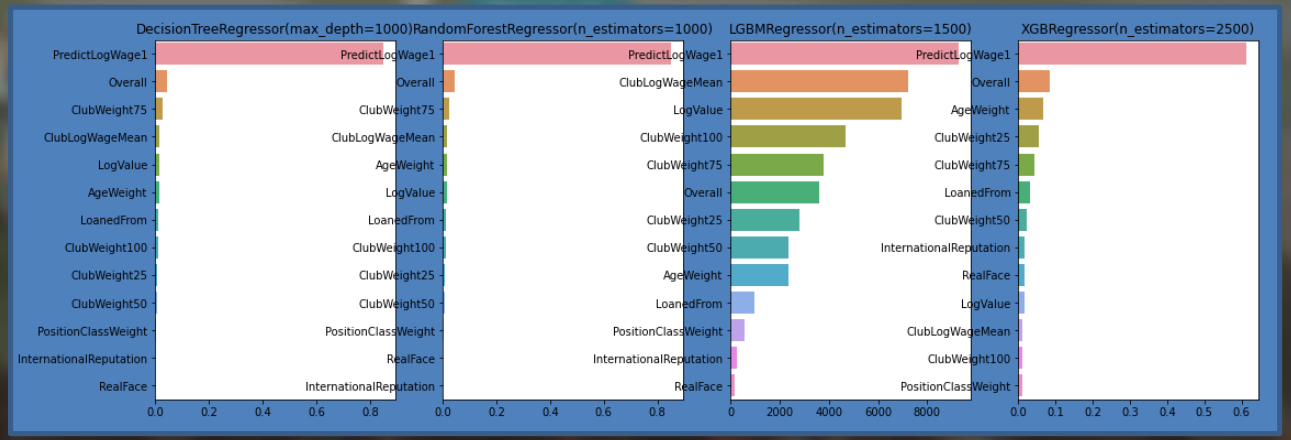
K-107 BIG DATA

검증 데이터 결과

- ✓ Log 변환한 값으로 테스트함.
- ✓ 트레인 데이터의 80%를 훈련 데이터로, 20%를 검증 데이터로 나눠 테스트한 결과, R^2가 LGBM : 96.5%, XGB : 96.6% 예측을 보였음.
- ✓ RMSE와 변수중요도를 확인한 결과가 비슷하여 신뢰가능한 검증된 모델임을 확인.

```
RandomForestRegressor(n_estimators=1000)
로그 변환된 MSE:    7.237
로그 변환된 RMSE:   2.69
결정계수 R2:        0.951
==============================
LGBMRegressor(n_estimators=1500)
로그 변환된 MSE:    5.146
로그 변환된 RMSE:   2.268
결정계수 R2:        0.965
==============================
[15:25:13] WARNING: src/objective/regress
XGBRegressor(n_estimators=2500)
로그 변환된 MSE:    5.01
로그 변환된 RMSE:   2.238
결정계수 R2:        0.966
```

DecisionTree, RandomForest, LGBM, XGB에서 변수 중요도

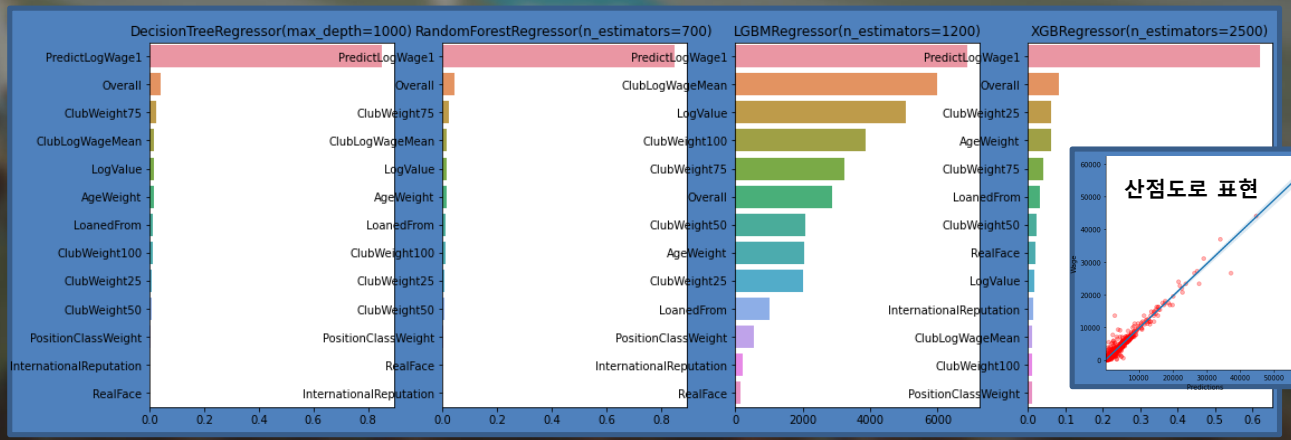


테스트 데이터 결과

- ✓ 트레인 데이터에서 만든 '팀WAGE표'를 테스트 데이터의 팀에 맞춰 그대로 붙여줌.
- ✓ 테스트 데이터는 트레인 데이터에서와 똑같은 가중치&수치 대입과 계산 과정을 거침.
- ✓ 트레인 데이터로 훈련시킨 모델에서 예측한 테스트 데이터의 LogWage는 LGBM : 95.3%, XGB : 95.3% 예측을 보였음.
- ✓ RMSE와 변수중요도를 확인한 결과가 비슷하여 신뢰 가능한 결과로 최종 결론.

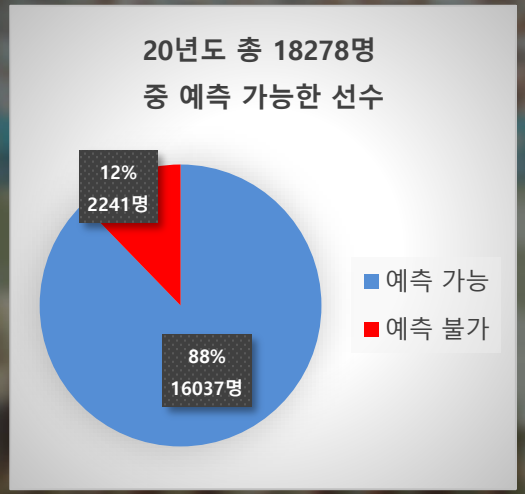
```
RandomForestRegressor(n_estimators=700)
로그 변환된 MSE:   10.225
로그 변환된 RMSE:  3.198
결정계수 R2:       0.933
==============================
LGBMRegressor(n_estimators=1200)
로그 변환된 MSE:   7.101
로그 변환된 RMSE:  2.665
결정계수 R2:       0.953
==============================
[15:36:05] WARNING: src/objective/regres
XGBRegressor(n_estimators=2500)
로그 변환된 MSE:   7.078
로그 변환된 RMSE:  2.66
결정계수 R2:       0.953
```

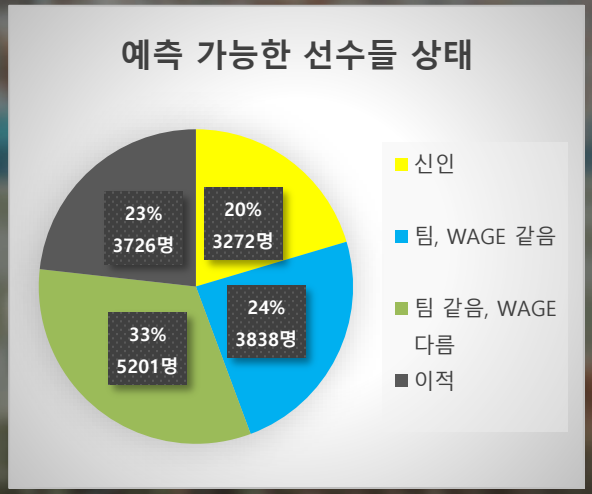
DecisionTree, RandomForest, LGBM, XGB에서 변수 중요도



20년도 WAGE 예측

19년도 '팀WAGE표'와 20년도 선수 데이터를 이용하여 20년도 WAGE를 예측.





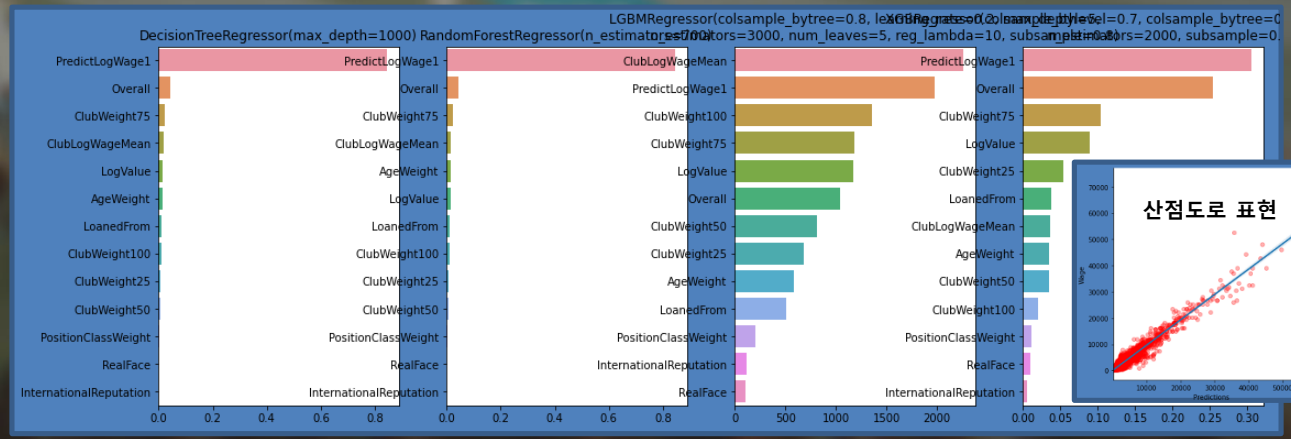
✓ 20년에 새로 데뷔한 신인도 예측 가능, 다만 19년도에 없던 팀에 소속된 선수는 예측 불가.

20년도 WAGE 예측 결과

- ✓ 19년도 '팀WAGE표'를 20년도에 그대로 붙여주고 19년도에서 이용한 공식과 수치를 그대로 적용.
- ✓ 19년도 전체 데이터로 훈련시킨 모델을 통한 20년도 전체 데이터의 LogWage 예측은 LGBM : 93%, XGB : 92.9% 을 보였음.
- ✓ 능력 수치 변동됐지만(무조건 나이는 한 살 더 먹음) 팀과 WAGE가 그대로 유지된(계약에 의해) 선수들이 무려 3833명 이기 때문에 예측률이 떨어진 것으로 판단.
- ✓ RMSE와 변수중요도를 확인한 결과가 비슷하여 신뢰 가능한 결과로 최종 결론.

```
RandomForestRegressor(n_estimators=700)
로그 변환된 MSE:   12.202
로그 변환된 RMSE:  3.493
결정계수 R2:       0.921
==============================
LGBMRegressor(colsample_bytree=0.6, lear
              n_estimators=3000, num_lea
로그 변환된 MSE:   10.874
로그 변환된 RMSE:  3.298
결정계수 R2:       0.93
==============================
[11:06:41] WARNING: src/objective/regres
XGBRegressor(colsample_bylevel=0.7, cols
             n_estimators=2000, subsampl
로그 변환된 MSE:   10.965
로그 변환된 RMSE:  3.311
결정계수 R2:       0.929
```

DecisionTree, RandomForest, LGBM, XGB에서 변수 중요도



예측 활용

K-107 BIG DATA

선수 이적 & 재계약 시 WAGE 예측 프로그램 개발

✓ 훈련된 인공지능을 이용하여 **선수 이름**과 **이적 or 재계약 희망 팀**을 입력하면 예측 WAGE가 나옴

✓ 신인도 예측이 가능하며 '팀WAGE표'가 있는 팀이면 예측이 가능.

19년도 데이터로 19년도 가상 이적 예측

```
transfer('E. Hazard', "Real Madrid")
```

	Name	CurrentClub	Wage	TransferClub	PredictWage	Difference
0	E. Hazard	Chelsea	44200	Real Madrid	59122	14922

19년도 ' 구단WAGE표'와 20년도 선수 데이터로 20년도 가상 이적 예측

```
transfer('H. Son', "Manchester United")
```

	Name	LastClub	Wage19	CurrentClub	TransferClub	PredictWage20	Difference19
0	H. Son	Tottenham Hotspur	16250.0	Tottenham Hotspur	Manchester United	29143	12893.0

이미지:
https://www.voakorea.com/korea/korea-life/2542642

선수와 팀 간 WAGE 협상용으로 활용 가능

✓ 선수는 프로그램을 이용하여 이적 **or** 재계약 할 팀에서 **자신이 받을 적정 WAGE**를 미리 알고 협상에 임하여 불이익을 피할 수 있다.

✓ 팀은 프로그램을 이용하여 **오버페이를 방지**할 수 있다.

이미지:
https://www.cbssports.com/soccer/news/former-barcelona-president-josep-maria-bartomeu-arrested-as-police-raid-camp-nou-per-reports/

소감

K-107 BIG DATA

1. FIFA데이터와 같은 소득&재산 등 돈 관련 데이터 분석에 자신이 생김.
2. 능력을 높이는 것도 중요하지만 능력을 인정해 주는 곳에 있어야 한다.
3. 인공지능만 믿을 것이 아니라 분석가가 우선적으로 계산해야 한다.
4. 인공지능 알고리즘 공부를 더 많이 해야겠다.
5. '팀WAGE표'의 세분화와 공식들의 개선, 선수 계약기간까지 반영 한다면 R^2가 더 오를 수 있을 것이다.
6. 웹서비스나 어플로 프로그램을 구현하여 상용화하겠다.

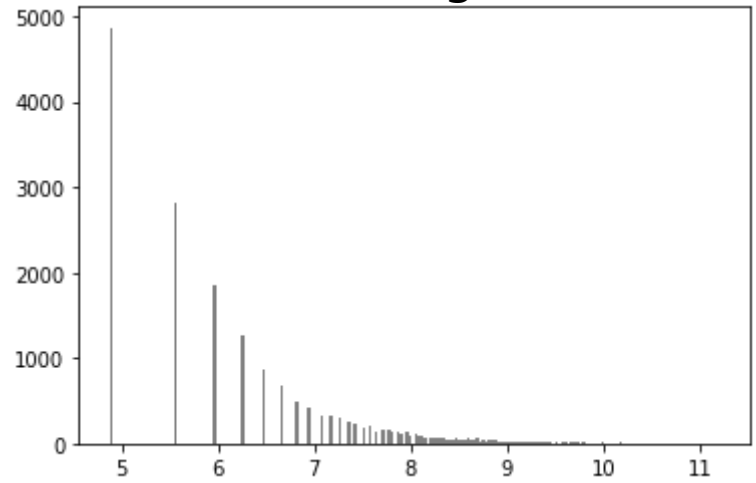
첨부 자료들

K-107

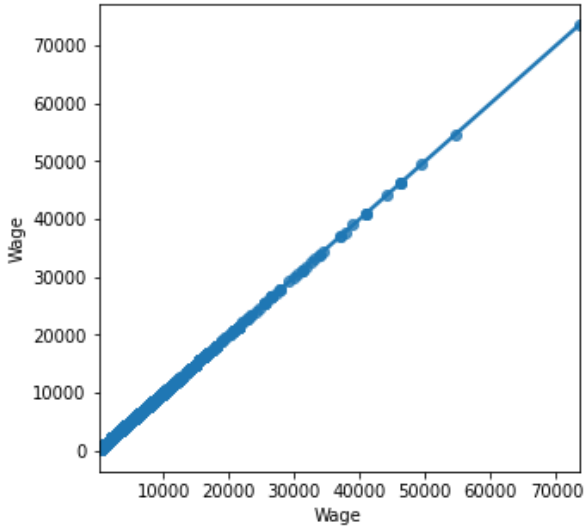
최소 Wage부터 정렬하여 해당 Wage를 받는 선수 숫자

Wage	130	260	390	520	650	780	910	1040	1170	1300	1430	1560	1690	1820	1950	2080	2210	2340	2470	2600	2730	2860	2990	3120
Counts	3866	2234	1508	987	716	551	392	335	264	252	238	200	188	147	149	108	136	124	109	115	94	103	64	83

로그 변환된 Wage 분포



이러면 100%예측이다.



첨부 자료들

K-107

훈련 데이터에서 Total 컬럼 만들기.

평균과 표준편차 알아보기

```
print('Reactions 평균: ',df.Reactions.mean())
print('Reactions 표준편차: ',df.Reactions.std())
```

```
Reactions 평균:  61.84375872661268
Reactions 표준편차:  8.99182501655843
```

```
print('Composure 평균: ',df.Composure.mean())
print('Composure 표준편차: ',df.Composure.std())
```

```
Composure 평균:  58.66699246020664
Composure 표준편차:  11.364099005853578
```

```
print('Potential 평균: ',df.Potential.mean())
print('Potential 표준편차: ',df.Potential.std())
```

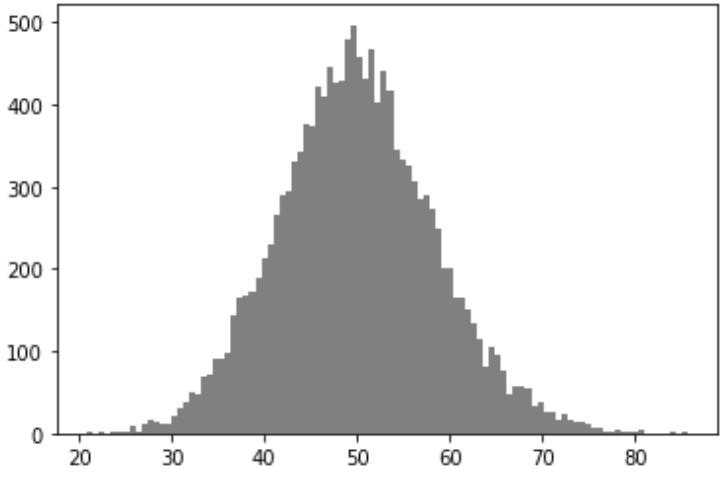
```
Potential 평균:  71.3573722423904
Potential 표준편차:  6.127127303729245
```

표준점수로 변환

```
df.Reactions = (df.Reactions-df.Reactions.mean())/df.Reactions.std()*10 +50
df.Composure = (df.Composure-df.Composure.mean())/df.Composure.std()*10 +50
df.Potential = (df.Potential-df.Potential.mean())/df.Potential.std()*10 +50
```

가중평균으로 Total 컬럼 생성

```
df['Total'] = (df.Reactions*48 + df.Composure*38 +
               df.Potential*35)/(48+38+35)
```



첨부 자료들

K-107

클럽 LogWage 곡선 제작

훈련 데이터에서 ClubWeight컬럼들 만들기

- 25, 50, 75, max로 계산
- 클럽의 가장 높은 LogWage를 전체의 최저 LogWage대비 몇 배인지 계산하여
- 클럽 LogWage 곡선으로 적용한다.

이 가중치는 다른 가중치를 더욱 배가 시켜준다. 예를 들어 강남에서 사는 사람에게 자동차가 있다면 그 자동차는

타지역보다 비싼 자동차일테니 다른 지역의 자동차들보다 강남가중치를 주는 것이 마땅하다는 생각에서 시작했다.

!! 쉽게 새로운 컬럼 만드는 방법!!

- df['Multiple'] = 1

```
ClubWeight100 = pd.DataFrame(np.zeros(df.shape[0]),columns=['ClubWeight100'])
ClubWeight75 = pd.DataFrame(np.zeros(df.shape[0]),columns=['ClubWeight75'])
ClubWeight50 = pd.DataFrame(np.zeros(df.shape[0]),columns=['ClubWeight50'])
ClubWeight25 = pd.DataFrame(np.zeros(df.shape[0]),columns=['ClubWeight25'])

df = pd.concat([df,ClubWeight100,ClubWeight75,ClubWeight50,ClubWeight25], axis=1)
```

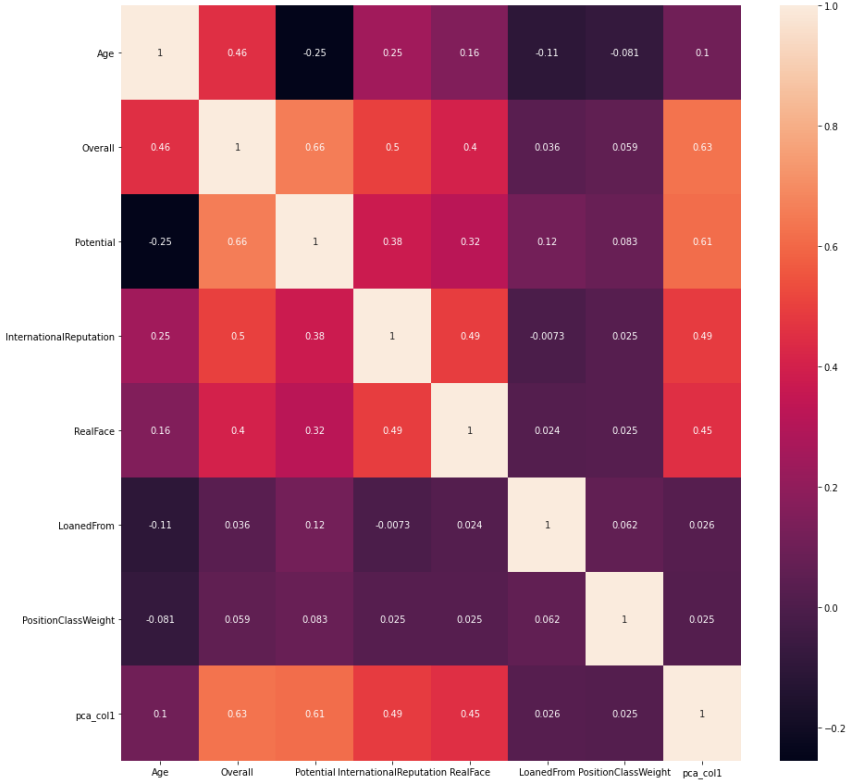
```
for i in range(df.shape[0]):
    df.ClubWeight100[i] = df[df.Club==df.Club[i]].LogWage.describe()[7]/df.LogWage.min()
    df.ClubWeight75[i] = df[df.Club==df.Club[i]].LogWage.describe()[6]/df.LogWage.min()
    df.ClubWeight50[i] = df[df.Club==df.Club[i]].LogWage.describe()[5]/df.LogWage.min()
    df.ClubWeight25[i] = df[df.Club==df.Club[i]].LogWage.describe()[4]/df.LogWage.min()
```

첨부 자료들

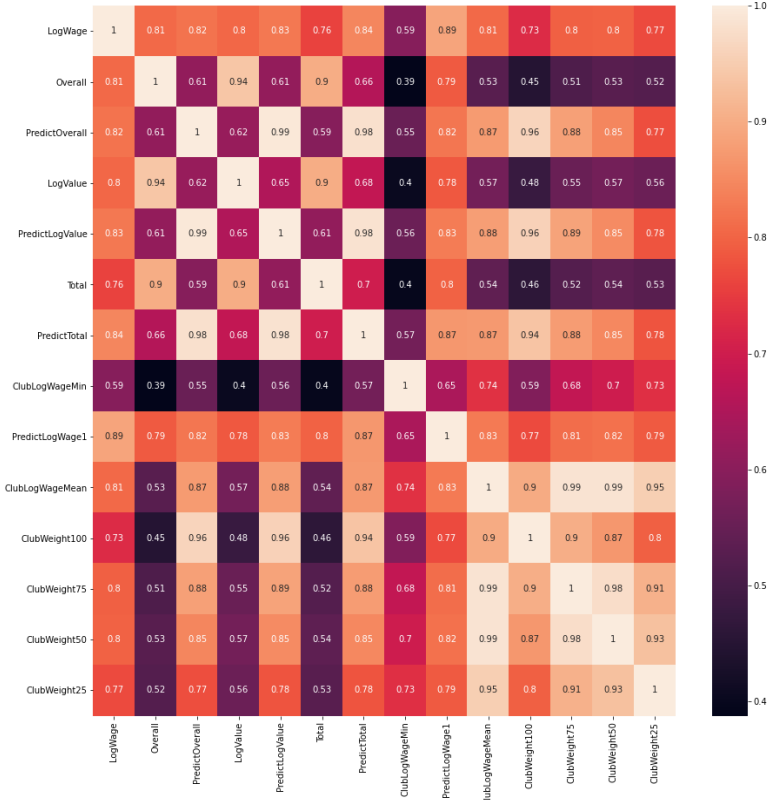
훈련 데이터에서 컬럼들의 상관계수

K-107

이산형 데이터 컬럼들의 상관계수



연속형 데이터 컬럼들의 상관계수

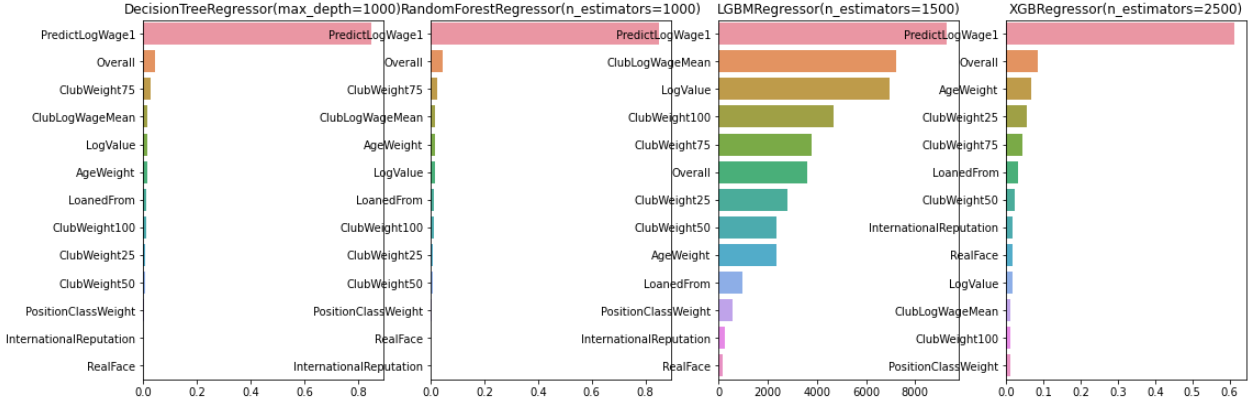


첨부 자료들

K-107

```
==============================
KNeighborsRegressor(n_neighbors=3)
로그 변환된 MSE:   17.178
로그 변환된 RMSE:  4.145
결정계수 R2:       0.884
==============================
LinearRegression()
로그 변환된 MSE:   16.678
로그 변환된 RMSE:  4.084
결정계수 R2:       0.887
==============================
Ridge(alpha=0.05)
로그 변환된 MSE:   7.658
로그 변환된 RMSE:  2.767
결정계수 R2:       0.948
==============================
RandomForestRegressor(n_estimators=1000)
로그 변환된 MSE:   7.237
로그 변환된 RMSE:  2.69
결정계수 R2:       0.951
==============================
LGBMRegressor(n_estimators=1500)
로그 변환된 MSE:   5.146
로그 변환된 RMSE:  2.268
결정계수 R2:       0.965
==============================
[15:25:13] WARNING: src/objective/regress
XGBRegressor(n_estimators=2500)
로그 변환된 MSE:   5.01
로그 변환된 RMSE:  2.238
결정계수 R2:       0.966
```

훈련 데이터와 검증 데이터에서의
R^2와 변수 중요도



첨부 자료들

K-107

테스트 데이터에 가중치 대입

훈련데이터에서 계산한 수치(평균, 중앙값, 표준편차 등등)들을 그대로 테스트 데이터에 적용해야한다. 테스트 데이터에서는 아무런 값의 계산이 있어서는 안된다.

Age 가중치

```
# Age 변환

df_test['AgeWeight'] = df_test['Age']

def change_value_from(x):
    out = ''
    if x in [30,31]: out = 1.375
    elif x in [29,32]: out = 1.333
    elif x in [28,33]: out = 1.333
    elif x in [27,34]: out = 1.333
    elif x in [26,35]: out = 1.229
    elif x in [25,36]: out = 1.229
    elif x in [24,37]: out = 1.229
    elif x in [23,38]: out = 1.229
    elif x in [22,39]: out = 1.229
    elif x in [21,40]: out = 1.146
    elif x in [20,41]: out = 1.146
    elif x in [19,42]: out = 1
    elif x in [18,43,44,45]: out = 1
    elif x in [16,17,46,47]: out = 1
    return float(out)

df_test.AgeWeight = df_test.Age.apply(lambda x: change_value_from(x))
df_test.AgeWeight
```

```
10       1.333
5718     1.375
11780    1.229
8245     1.229
9673     1.229
         ...
1794     1.229
2982     1.333
2190     1.375
6670     1.375
11536    1.333
```

InternationalReputation 가중치

```
#InternationalReputation변환
def change(x):
    cat = ''
    if x == 1: cat = 1
    elif x == 2: cat = 1.36
    elif x == 3: cat = 1.54
    elif x == 4: cat = 1.68
    elif x == 5: cat = 1.78
    return float(cat)

df_test['InternationalReputationWeight'] = df_test.InternationalReputation. \
                                           apply(lambda x: change(x))
df_test.InternationalReputationWeight
```

```
10       1.68
5718     1.00
11780    1.00
8245     1.00
9673     1.00
         ...
1794     1.00
2982     1.00
2190     1.00
6670     1.00
11536    1.00
```

RealFace 가중치

```
def change(x):
    cat = ''
    if x == 0: cat = 1
    elif x ==1: cat = 1.38
    return cat

df_test['RealFaceWeight'] = df_test.RealFace.apply(lambda x: change(x))
df_test.RealFaceWeight
```

```
10       1.38
5718     1.00
11780    1.00
8245     1.00
9673     1.00
         ...
1794     1.00
2982     1.00
2190     1.00
6670     1.00
11536    1.00
```

첨부 자료들

K-107

Total 컬럼 생성

```
df_test.Reactions = (df_test.Reactions-61.84375872661268)/ ₩
                    8.99182501655843*10 +50
df_test.Composure = (df_test.Composure-58.66699246020664)/ ₩
                    11.364099005853578*10 +50
df_test.Potential = (df_test.Potential-71.3573722423904)/ ₩
                    6.127127303729245*10 +50

# 가중평균

df_test['Total'] = (df_test.Reactions*48 + df_test.Composure*38 +
                    df_test.Potential*35)/(48+38+35)
```

테스트 데이터에 가중치 대입과 Total 컬럼 생성

LoanedFrom 가중치

```
def change(x):
    cat = ''
    if x == 0: cat = 1
    elif x ==1: cat = 1.15
    return cat

df_test['LoanedFromWeight'] = df_test.LoanedFrom.apply(lambda x: change(x))
df_test.LoanedFromWeight
```

```
10       1.0
5718     1.0
11780    1.0
8245     1.0
9673     1.0
        ...
1794     1.0
2982     1.0
2190     1.0
6670     1.0
11536    1.0
```

PositionClass 가중치

```
def change(x):
    cat = ''
    if x == 'GoalKeeper': cat = 1
    elif x == 'Defender': cat = 1.07
    elif x == 'Medfielder': cat = 1.07
    elif x == 'Foward': cat = 1.13
    return cat

df_test['PositionClassWeight'] = df_test.PositionClass.apply(lambda x: change(x))
df_test.PositionClassWeight
```

```
10       1.13
5718     1.00
11780    1.00
8245     1.00
9673     1.07
        ...
1794     1.13
2982     1.13
2190     1.07
6670     1.00
11536    1.07
```

첨부 자료들

K-107

훈련데이터의 클럽 관련 컬럼들을 테스트데이터에 삽입

```
df_train = pd.read_pickle('./dataset/나혼자FIFA_훈련데이터최종.pkl')
df_train
```

```
df_train1 = df_train[['Club','ClubWeight100','ClubWeight75','ClubWeight50',
                      'ClubWeight25','ClubLogWageMean', 'ClubLogWageMedian',
                      'ClubLogWageMin']]
df_train1
```

	Club	ClubWeight100	ClubWeight75	ClubWeight50	ClubWeight25	ClubLogWageMean	ClubLogWageMedian	ClubLogWageMin
0	FC Barcelona	2.298240	2.090291	1.988824	1.622995	91.552604	96.959097	62.557500
1	Real Madrid	2.237408	2.122623	1.991895	1.654479	93.004187	97.108826	70.656134
2	Real Madrid	2.237408	2.122623	1.991895	1.654479	93.004187	97.108826	70.656134
3	Real Madrid	2.237408	2.122623	1.991895	1.654479	93.004187	97.108826	70.656134
4	Real Madrid	2.237408	2.122623	1.991895	1.654479	93.004187	97.108826	70.656134
...	...	...	...	...	...	...	...	...
14319	Atlético Huila	1.141394	1.000000	1.000000	1.000000	49.613627	48.751973	48.751973
14320	Chamois Niortais Football Club	1.224301	1.141394	1.000000	1.000000	51.929775	48.751973	48.751973
14321	New York Red Bulls	1.425160	1.366217	1.141394	1.000000	57.836026	55.645204	48.751973
14322	Lech Poznań	1.449298	1.283179	1.224301	1.000000	58.365376	59.687076	48.751973
14323	Excelsior	1.470892	1.224301	1.141394	1.000000	55.645760	55.645204	48.751973

14324 rows × 8 columns

```
df_join = pd.merge(df_test, df_train1,on='Club')
df_join
```

	Wage	Name	Age	Nationality	Overall	Potential	Club
0	26650	R. Lewandowski	29	Poland	90	80.426376	FC Bayern München
1	26650	R. Lewandowski	29	Poland	90	80.426376	FC Bayern München
2	26650	R. Lewandowski	29	Poland	90	80.426376	FC Bayern München
3	26650	R. Lewandowski	29	Poland	90	80.426376	FC Bayern München
4	26650	R. Lewandowski	29	Poland	90	80.426376	FC Bayern München
...	...	...	...	...	...	...	...
76750	130	Rayco	21	Spain	62	49.416738	CD Lugo
76751	130	Rayco	21	Spain	62	49.416738	CD Lugo
76752	130	Rayco	21	Spain	62	49.416738	CD Lugo
76753	130	Rayco	21	Spain	62	49.416738	CD Lugo
76754	130	Rayco	21	Spain	62	49.416738	CD Lugo

76755 rows × 38 columns

첨부 자료들

K-107

```
df_join = df_join.drop_duplicates()
df_join
```

	Wage	Name	Age	Nationality	Overall	Potential	Club	Value	Special	Prefer
0	26650	R. Lewandowski	29	Poland	90	80.426376	FC Bayern München	10010000.0	2152	
20	650	P. Will	19	Germany	61	59.209255	FC Bayern München	68250.0	1627	
40	650	L. Mai	18	Germany	64	70.633858	FC Bayern München	110500.0	1402	
60	11050	K. Coman	22	France	83	75.530117	FC Bayern München	4420000.0	1918	
80	650	F. Evina	17	Germany	63	65.737600	FC Bayern München	94250.0	1645	
...	...	...	...	...	...	...	...	...	...	
76634	260	Matheus Jesus	21	Brazil	65	57.577169	Portimonense SC	107250.0	1769	
76657	130	Pepê	20	Brazil	62	55.945083	Portimonense SC	74750.0	1604	
76680	650	Wilson Manafá	23	Portugal	70	55.945083	Portimonense SC	260000.0	1802	
76703	1560	S. Nakajima	23	Japan	78	70.633858	Portimonense SC	1820000.0	1951	
76726	130	Rayco	21	Spain	62	49.416738	CD Lugo	61750.0	1649	

3582 rows × 38 columns

```
df_test = df_join.sort_values(by='Wage', ascending=False)
df_test = df_test.reset_index()
df_test.drop('index',axis=1,inplace=True)
df_test[['Club', 'Name','ClubWeight100', 'ClubWeight75', 'ClubWeight50',
    'ClubWeight25', 'ClubLogWageMean', 'ClubLogWageMin']]
```

	Club	Name	ClubWeight100	ClubWeight75	ClubWeight50	ClubWeight25	ClubLogWageMean	ClubLogWageMin
0	FC Barcelona	L. Suárez	2.298240	2.090291	1.988824	1.622995	91.552604	62.557500
1	FC Barcelona	Coutinho	2.298240	2.090291	1.988824	1.622995	91.552604	62.557500
2	Real Madrid	Casemiro	2.237408	2.122623	1.991895	1.654479	93.004187	70.656134
3	FC Barcelona	Piqué	2.298240	2.090291	1.988824	1.622995	91.552604	62.557500
4	Manchester United	P. Pogba	2.139041	2.012069	1.962605	1.708364	90.809410	69.479371
...	...	...	...	...	...	...	...	...
3577	AS Béziers	J. Saki	1.283179	1.000000	1.000000	1.000000	50.566985	48.751973
3578	AS Béziers	S. Beusnard	1.283179	1.000000	1.000000	1.000000	50.566985	48.751973
3579	AS Béziers	Cidinho	1.283179	1.000000	1.000000	1.000000	50.566985	48.751973
3580	SKN St. Pölten	R. Ljubicic	1.366217	1.283179	1.224301	1.141394	59.612481	48.751973
3581	CD Lugo	Rayco	1.425160	1.328871	1.283179	1.141394	60.928494	48.751973

3582 rows × 8 columns

첨부 자료들

K-107

테스트 데이터에서 PredictLogWage1컬럼 생성

```
df_test['PredictLogWage'] = 1
```

```
for i in range(df_test.shape[0]):
    df_test.PredictLogWage[i] = 2*(df_test.ClubWeight100[i]-1)+ ₩
    (df_test.AgeWeight[i]*df_test.InternationalReputationWeight[i]+ ₩
    df_test.RealFaceWeight[i]*df_test.PositionClassWeight[i]*df_test.LoanedFromWeight[i]+ ₩
    df_test.ClubWeight75[i]*df_test.ClubLogWageMin[i])+48.75197323201151
```

```
df_test['PredictLogWage1'] = df_test.PredictLogWage
```

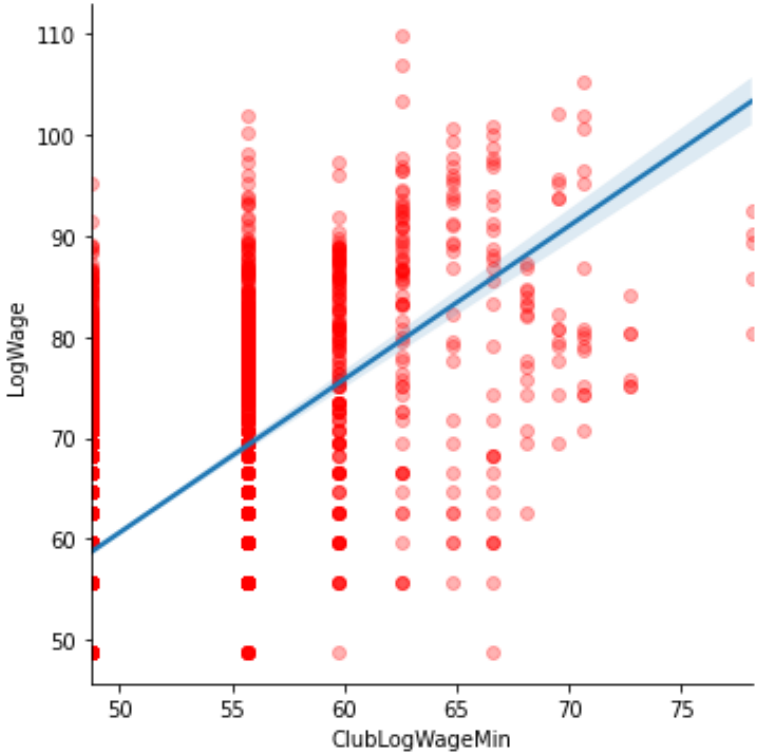
```
# 선수 한 명씩 추출
for i in range(df_test.shape[0]):


    # 클럽의 최소LogWage와 최대LogWage의 배수가 1이고
    if df_test.ClubWeight100[i] == 1:
        # 이 선수의 능력치가 클럽의 하위 100%에 속하면 능력치를 0으로 만든다.
        if df_test.PredictLogWage[i]<=df_test[df_test.Club==df_test.Club[i]].PredictLogWage.describe()[7]:
            df_test.PredictLogWage1[i] = -20


    # 클럽의 최소LogWage와 75% LogWage의 배수가 1이고
    if df_test.ClubWeight75[i] == 1:
        # 이 선수의 능력치가 클럽의 하위 75%에 속하면 능력치를 0으로 만든다.
        if df_test.PredictLogWage[i]<=df_test[df_test.Club==df_test.Club[i]].PredictLogWage.describe()[6]:
            df_test.PredictLogWage1[i] = -20


    # 클럽의 최소LogWage와 50% LogWage의 배수가 1이고
    if df_test.ClubWeight50[i] == 1:
        # 이 선수의 능력치가 클럽의 하위 50%에 속하면 능력치를 0으로 만든다.
        if df_test.PredictLogWage[i]<=df_test[df_test.Club==df_test.Club[i]].PredictLogWage.describe()[5]:
            df_test.PredictLogWage1[i] = -20


    # 클럽의 최소LogWage와 25% LogWage의 배수가 1이고
    if df_test.ClubWeight25[i] == 1:
        # 이 선수의 능력치가 클럽의 하위 25%에 속하면 능력치를 0으로 만든다.
        if df_test.PredictLogWage[i]<=df_test[df_test.Club==df_test.Club[i]].PredictLogWage.describe()[4]:
            df_test.PredictLogWage1[i] = -20
```

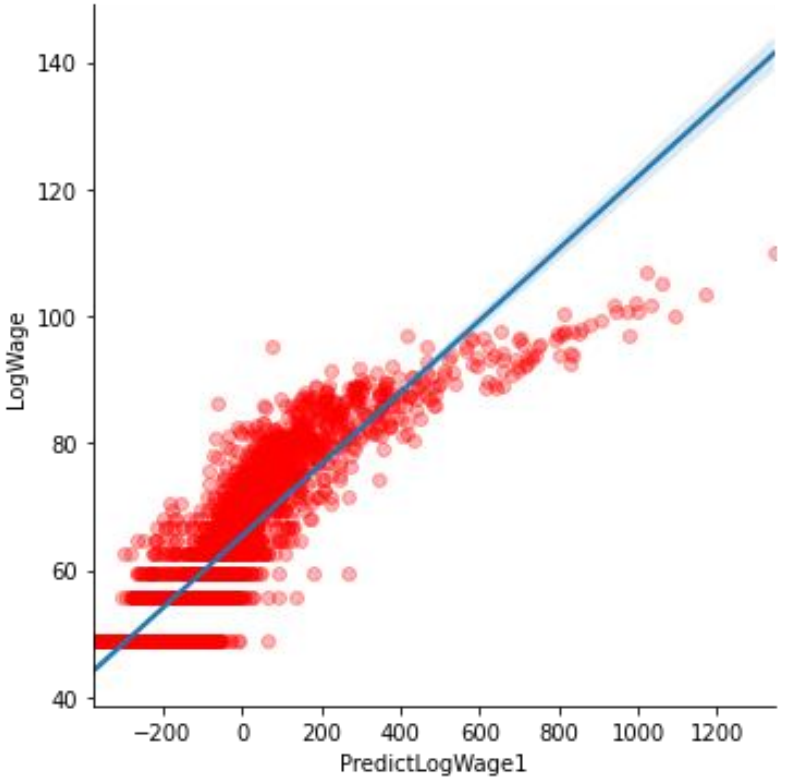
```
for i in range(df_test.shape[0]):
    df_test.PredictLogWage1[i]=df_test.PredictLogWage1[i]-(df_test.Total.max()-df_test.Total[i])*7 ₩
                               -(df_test.LogValue.max()-df_test.LogValue[i])*10
```

첨부 자료들

K-107

테스트 데이터에 가중치 적용과 공식 계산 후 그래프 모습



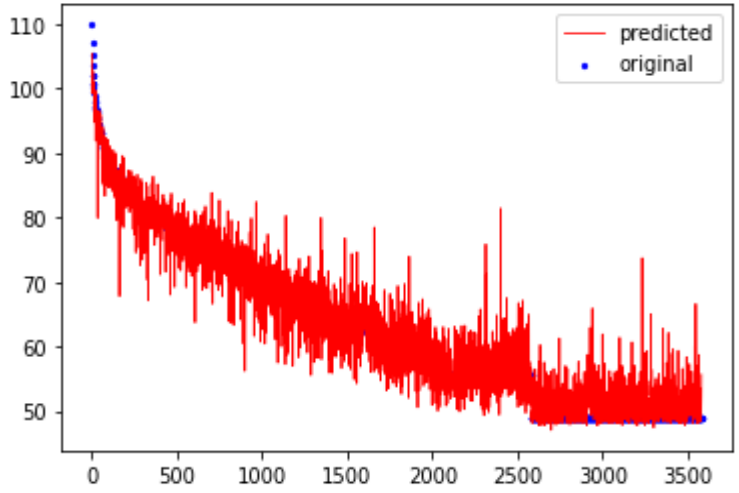


첨부 자료들

K-107

테스트 데이터 예측 결과

```
K-fold CV average score: 0.94
r2 :  0.9194785254069993
MSE: 12.25
RMSE: 3.50
```

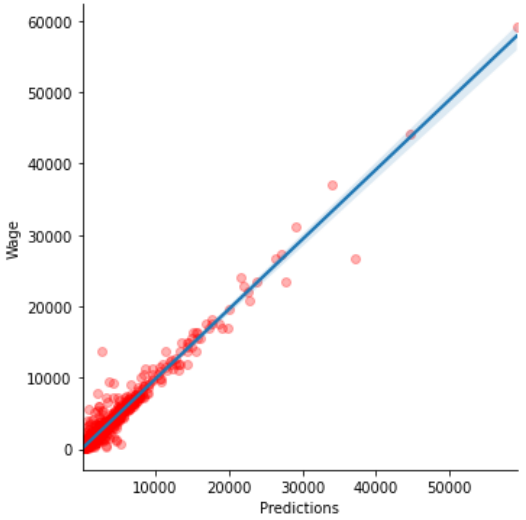


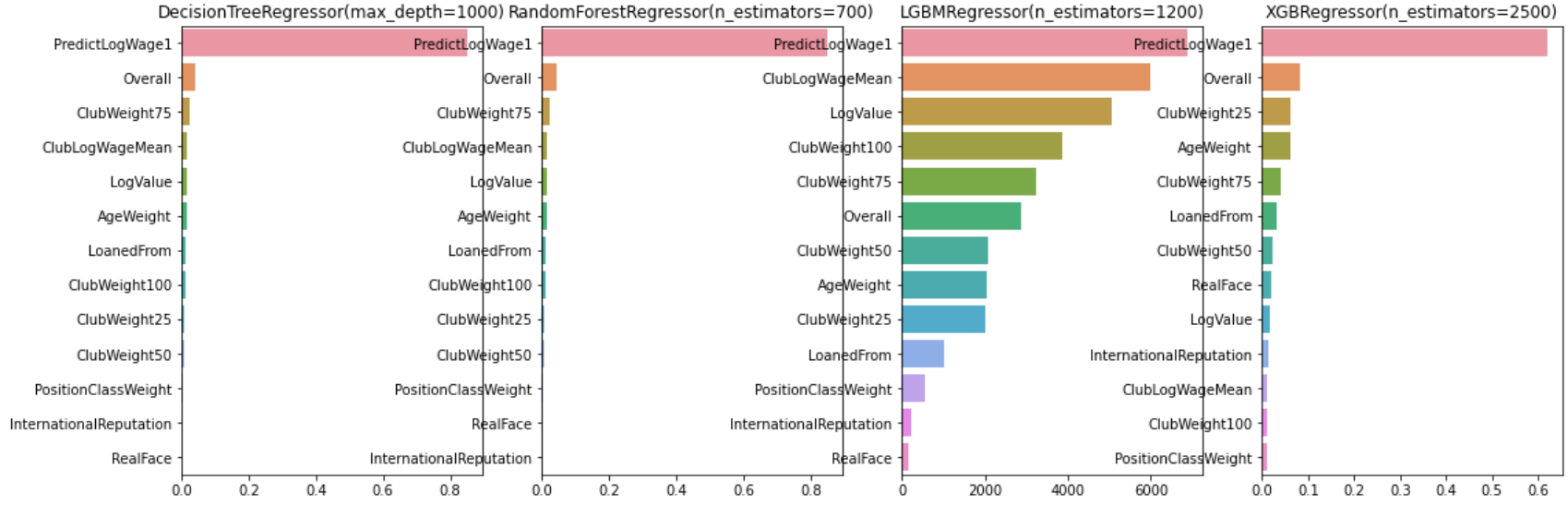

```
==================================
KNeighborsRegressor(n_neighbors=3)
로그 변환된 MSE:   23.979
로그 변환된 RMSE:  4.897
결정계수 R2:       0.842
==================================
LinearRegression()
로그 변환된 MSE:   18.826
로그 변환된 RMSE:  4.339
결정계수 R2:       0.876
==================================
Ridge(alpha=10)
로그 변환된 MSE:   10.997
로그 변환된 RMSE:  3.316
결정계수 R2:       0.928
==================================
RandomForestRegressor(n_estimators=700)
로그 변환된 MSE:   10.225
로그 변환된 RMSE:  3.198
결정계수 R2:       0.933
==================================
LGBMRegressor(n_estimators=1200)
로그 변환된 MSE:   7.101
로그 변환된 RMSE:  2.665
결정계수 R2:       0.953
==================================
[15:36:05] WARNING: src/objective/regres
XGBRegressor(n_estimators=2500)
로그 변환된 MSE:   7.078
로그 변환된 RMSE:  2.66
결정계수 R2:       0.953
```

첨부 자료들

K-107

테스트 데이터 예측 결과의 변수 중요도와 Wage로 변환 한 뒤 산점도 표시





첨부 자료들

K-107

19년도 테스트 데이터에서 Wage 상위 12명과 하위 15명에서의 1대1 예측 결과

Nationality	Uruguay	Brazil	Brazil	Spain	France	Poland	Spain	Brazil	France	Spain	Spain	Belgium
Club	FC Barcelona	FC Barcelona	Real Madrid	FC Barcelona	Manchester United	FC Bayern München	Real Madrid	Manchester City	Arsenal	Real Madrid	Chelsea	Manchester City
Name	L. Suárez	Coutinho	Casemiro	Piqué	P. Pogba	R. Lewandowski	Lucas Vázquez	Fernandinho	A. Lacazette	Nacho Fernández	Azpilicueta	V. Kompany
Wage	59150	44200	37049	31199	27299	26649	26649	24050	23399	23399	22750	22099
Predictions	59125	44693	33973	29166	27181	37208	26407	21530	27759	23691	21975	22603

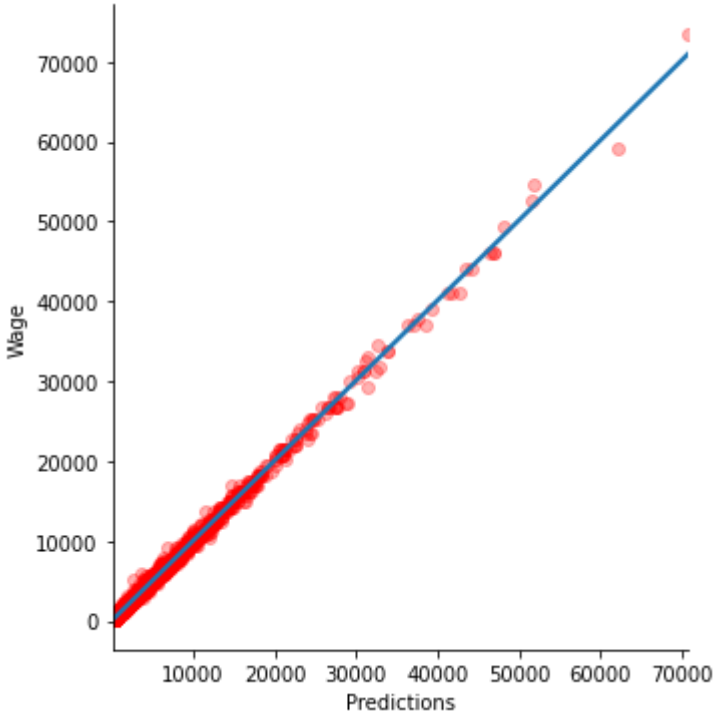
Chile	Chile	New Zealand	Germany	Korea Republic	France	Austria	Gambia	France	France	France	France	Brazil	Austria	Spain
CD Antofagasta	CD Antofagasta	FC Carl Zeiss Jena	FC Carl Zeiss Jena	Incheon United FC	Clermont Foot 63	FK Austria Wien	AS Béziers	AS Béziers	AS Béziers	AS Béziers	AS Béziers	AS Béziers	SKN St. Pölten	CD Lugo
C. Gutiérrez	P. Jerez	L. Rogerson	F. Brügmann	Lee Tae Hee	L. Rajot	N. Hahn	A. Kanté	M. Irigoyemborde	V. Viot	J. Saki	S. Beusnard	Cidinho	R. Ljubicic	Rayco
129	129	129	129	129	129	129	129	129	129	129	129	129	129	129
129	146	131	136	155	215	126	130	136	123	116	128	147	120	196

첨부 자료들

K-107

19년도 전체 데이터로 공식 적용하여 훈련시킨 모델의 정확도

```
==============================
[LightGBM] [Warning] Unknown parameter: colsample_bylevel
LGBMRegressor(colsample_bylevel=0.6, colsample_bytree=0.6, learning_rate=0.2,
              max_depth=5, n_estimators=3000, num_leaves=5, reg_lambda=10,
              subsample=0.8)
훈련데이터의 로그 변환된 MSE:   2.734
훈련데이터의 로그 변환된 RMSE:  1.654
훈련데이터의 결정계수 R2:       0.982
==============================
[08:54:56] WARNING: src/objective/regression_obj.cu:152: reg:linear is now dep
XGBRegressor(colsample_bylevel=0.7, colsample_bytree=0.7, max_depth=5,
             n_estimators=2000, subsample=0.8)
훈련데이터의 로그 변환된 MSE:   0.709
훈련데이터의 로그 변환된 RMSE:  0.842
훈련데이터의 정계수 R2:       0.995
```



첨부 자료들

K-107

선수가 대략적으로 주전으로 뛸 가능성이 있는 팀을 보여줌.

1. 이적하려는 선수의 포지션과 이적하려는 구단의 같은 포지션 선수들을 Overall과 Potential로 비교하여 구단 내 1위가 된다면 즉시 주전 전력감으로 생각.
2. 1위는 아니라고 해도 구단 내 같은 포지션에서 능력치가 최소인 선수보다 높다면 엔트리에 들어갈 수 있다고 생각.
3. 각각을 판단하는 함수를 정의.

```
def entry33(player):
    name = df[df.Name==player]
    position = name.PositionClass2.values[0]
    clubs = df[df.PositionClass2.values==position].Club.unique()
    teams = []
    for club in clubs:
        if name.Overall.values>df[df.Club==club][df[df.Club==club].PositionClass2==position].Overall.min():
            if name.Potential.values>df[df.Club==club][df[df.Club==club].PositionClass2==position].Potential.min():
                teams.append(club)
    df_teams = pd.DataFrame(teams , columns=['Club'])
    df_join = df[['Club','ClubWageMean']]
    df_entry33 = pd.merge(df_teams, df_join,on='Club')
    df_entry33 = df_entry33.drop_duplicates()
    df_entry33 = df_entry33.sort_values(by='ClubWageMean', ascending=False)
    df_entry33 = df_entry33.reset_index()
    df_entry33.drop('index',axis=1,inplace=True)
    print('엔트리 포함 가능한 팀의 수: ', len(df_entry33))
    return df_entry33.T
```

```
def best11(player):
    name = df[df.Name==player]
    position = name.PositionClass2.values[0]
    clubs = df[df.PositionClass2.values==position].Club.unique()
    teams = []
    for club in clubs:
        if name.Overall.values>=df[df.Club==club][df[df.Club==club].PositionClass2==position].Overall.max():
            if name.Potential.values>=df[df.Club==club][df[df.Club==club].PositionClass2==position].Potential.max():
                teams.append(club)
    df_teams = pd.DataFrame(teams , columns=['Club'])
    df_join = df[['Club','ClubWageMean']]
    df_best11 = pd.merge(df_teams, df_join,on='Club')
    df_best11 = df_best11.drop_duplicates()
    df_best11 = df_best11.sort_values(by='ClubWageMean', ascending=False)
    df_best11 = df_best11.reset_index()
    df_best11.drop('index',axis=1,inplace=True)
    print('주전 가능한 팀의 수: ', len(df_best11))
    return df_best11.T
```

첨부 자료들

K-107

19년도 주급WAGE표를 20년도 선수 정보에 붙여준 뒤
19년도 전체 훈련데이터로 학습한 모델로 20년도 선수의 WAGE를 예측

```
================================
LGBMRegressor(colsample_bytree=0.8, learning_rate=0.2, max_depth=5,
              n_estimators=3000, num_leaves=5, reg_lambda=10, subsample=0.8)
로그 변환된 MSE:   10.895
로그 변환된 RMSE:  3.301
결정계수 R2:       0.929
================================
[08:57:59] WARNING: src/objective/regression_obj.cu:152: reg:linear is now d
XGBRegressor(colsample_bylevel=0.7, colsample_bytree=0.7, max_depth=5,
             n_estimators=2000, subsample=0.8)
로그 변환된 MSE:   10.965
로그 변환된 RMSE:  3.311
결정계수 R2:       0.929
```