감귤나무 착과량 예측

PRESENTATION

TABLE OF CONTENTS

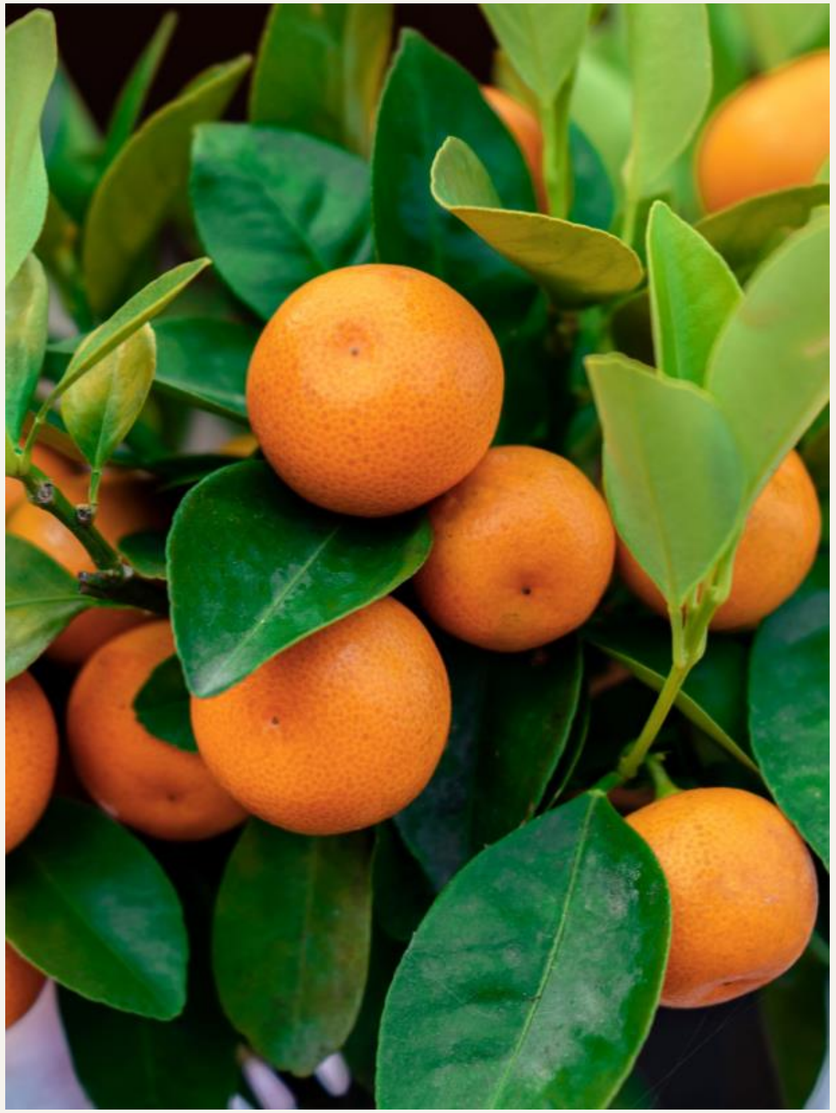
갈귤나무 착과량 예측

1. 제공된 수고, 수관폭, 엽록소, 새순 등의 변수들을 사용하여 착과량을 예측해야함.

2. 사용기법은 자유임.

3. CSV 파일을 불러온다던가, PANDAS, NUMPY 등을 불러오는 과정은
   모두들 다 했을 것이니 본 발표에서 다루지 않겠음.

4. 다른 팀들도 다들 잘 알테니 (생략)...



OUR FOCUS

1. 성능이 비교적 떨어질지라도 기본기를 탄탄하게 사용하자.

2. 포기하지 말자.

3. 배운 내용을 최대한 활용하자.

파생 변수 생성 근거

```
In [53]: corr_matrix
```

Out[53]:

	착과량 (int)	수고(m)	수관폭 1(min)	수관폭 2(max)	수관폭평균	2022-09-01 새순	2022-09-02 새순	2022-09-03 새순	2022-09-04 새순	2022-09-05 새순	...	2022-11-25 엽록소	2022-11-26 엽록소	2022-11-27 엽록소	2022-11-28 엽록소
착과량 (int)	1.000000	0.004526	0.008675	0.031372	0.021068	-0.981033	-0.980868	-0.980791	-0.980969	-0.980717	...	0.011146	0.011484	0.011809	0.012024

1. Y값과 주어진 X값들의 상간관계를 확인 해본 결과, 새순 데이터를 제외하고는 미미한 상관성을 보임.

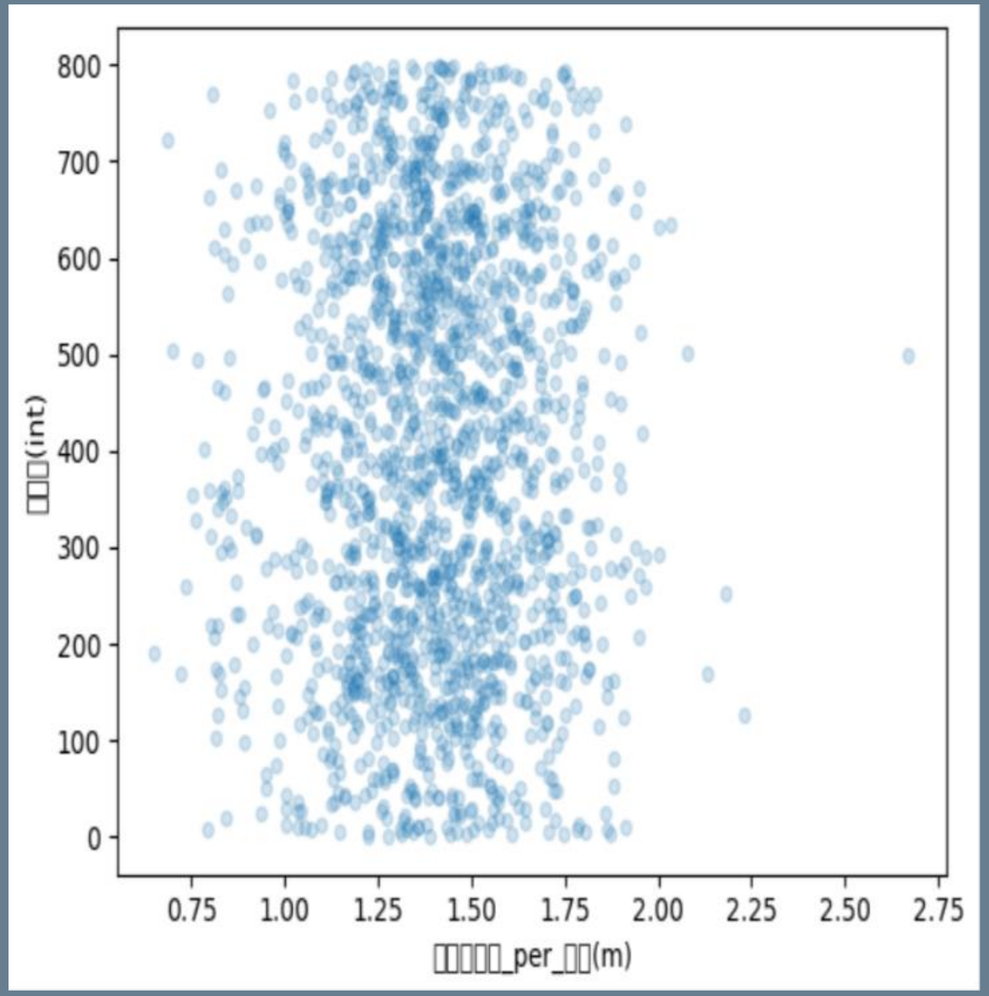
2. 각 변수간의 상관성은 있을 수 있으나, 근본적으로 Y값과 큰 상관성없는 X들은 크게 활용가치가 없다고 판단함.

3. 그래도 새순을 재외한 다른 변수들로 파생변수를 만들어 보려 했으나, 역시나 실패하였음.

3. 따라서 최종적으로 새순 데이터만을 이용하여 파생변수를 만들기로 의사결정.

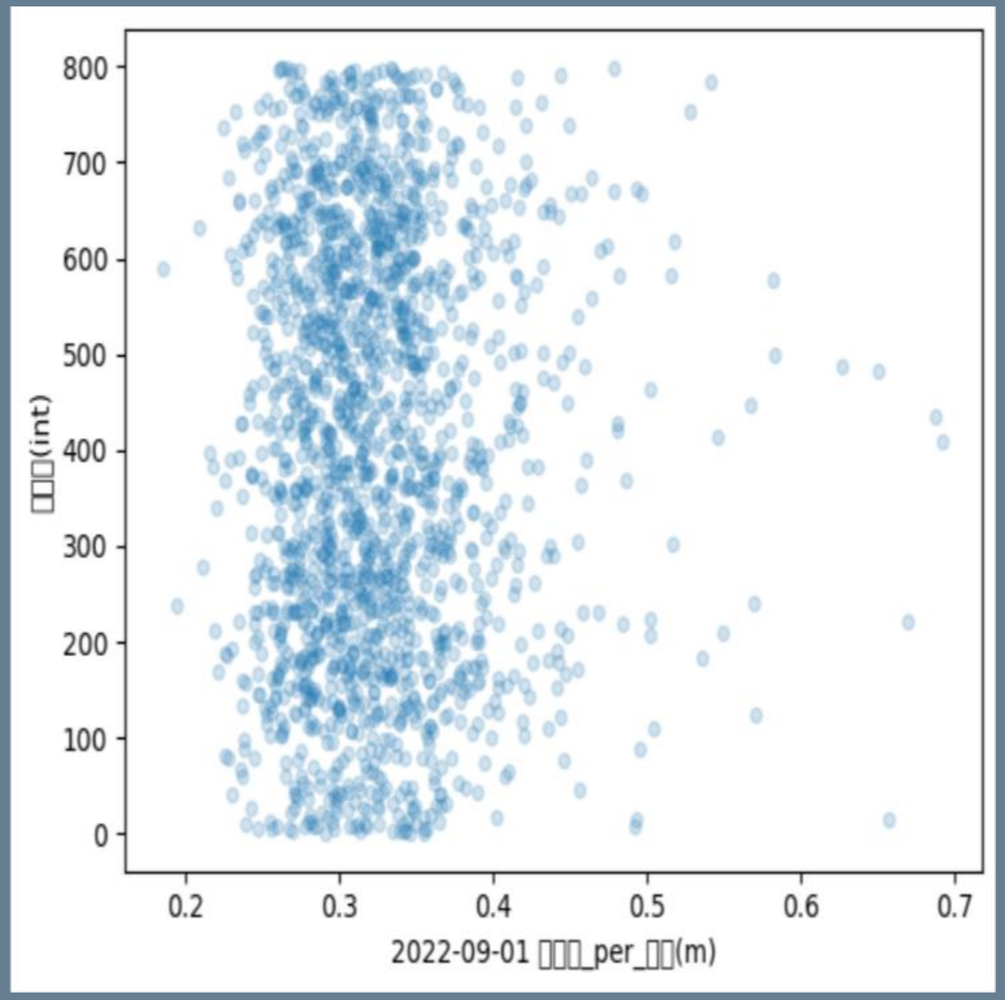
'수관폭 평균 per 수고(m)' 와 '착과량(int)'의 상관성

- '수고'와 '수관폭 평균' 변수들 간의 상관성이 비교적 높아 이들을 하나의 비율로 나타내어 착과량과 대응해보았지만 상관성이 보이지 않았음.
- 신장과 몸무게 비율인 BMI와 같이, 나무의 건강성을 판단하려는 1차원적 시선으로 접근하였음.
- BECAUSE,
  건장한 나무가 열매를 많이 달 수 있을 것이라 생각하였음.



파생 변수 생성 까지의 시도

'2022-09-01 엽록소_per_수고(m)' 과 '착과량(int)'의 상관성



- 나무의 키가 크면, 햇빛을 많이 받아,
  엽록소가 많이 생길 것이고,
  엽록소가 많으면 열매가 많이 열릴 것이라는 생각으로
  나름 논리적으로 접근함.

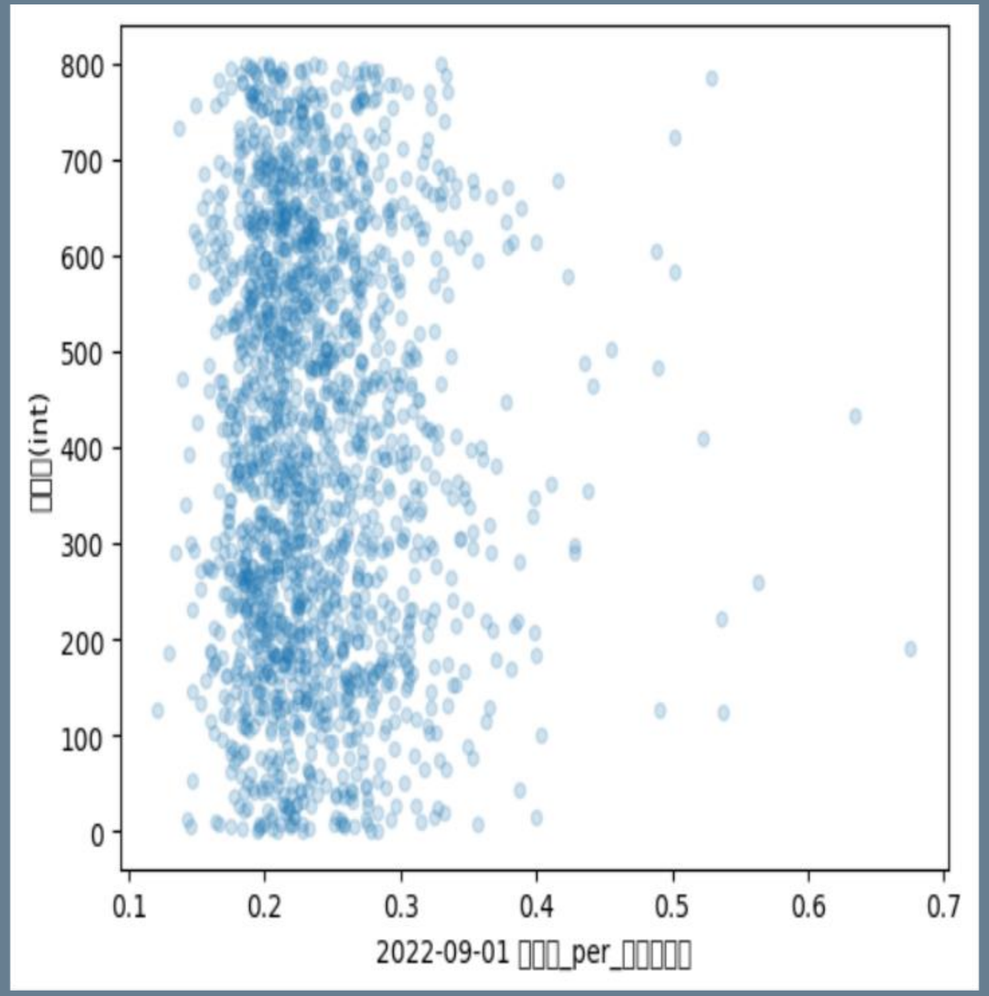
- 하지만 앞에서도 보았듯이,
  엽록소 데이터는 착과량에 미미한 영향을 줄 뿐이고,
  이 파생변수의 특정 부분에 Y값들이 몰려있어,
  수고와 엽록소의 비율은 각 나무별로 차이가 크지 않음.

- 따라서 엽록소 변수를 제외하기로 결정함.

파생 변수 생성 까지의 시도

'2022-09-01 엽록소_per_수관폭평균' 과 '착과량(int)'의 상관성

- 이 때도 수관폭평균을 잘 활용하면 상관성을 높일 수 있을 것이라 생각하였지만, 역시나 실패함.
- 엽록소 변수 자체가 큰 의미가 없으며, 엽록수를 이용하여 파생 변수를 만들 시, 기존의 변수를 오히려 어지럽히는 경향을 보임.
- 따라서 엽록소 변수를 제외하기로 결정함.



최종 파생 변수 생성

새순 평균, 새순 중간값, 새순 표준편차, 새순 최대값, 새순 차이

```
In [27]: leaf=train_set.iloc[:,6:95]
         leaf_mean=leaf.mean(axis='columns')

In [28]: leaf=train_set.iloc[:,6:95]
         leaf_min = leaf.min(axis='columns')

In [29]: leaf=train_set.iloc[:,6:95]
         leaf_max = leaf.max(axis='columns')

In [30]: leaf=train_set.iloc[:,6:95]
         leaf_std = leaf.std(axis='columns')

In [60]: leaf=train_set.iloc[:,6:95]
         leaf_median = leaf.median(axis='columns')

In [61]: leaf=train_set.iloc[:,6:95]
         leaf_gap = train_set.iloc[:,94] - train_set.iloc[:,6]

In [62]: train_set["새순 평균"] = leaf_mean

In [63]: train_set["새순 표준편차"] = leaf_std

In [64]: train_set["새순 중간값"] = leaf_median

In [65]: train_set["새순 최소값"] = leaf_min

In [66]: train_set["새순 최대값"] = leaf_max

In [67]: train_set["새순 차이"] = leaf_gap
```

최종 파생 변수 생성

새순 평균, 새순 중간값, 새순 표준편차, 새순 최대값, 새순 차이

새순 평균	새순 표준편차	새순 중간값	새순 최소값	새순 최대값	새순 차이
-0.975934	-0.979356	-0.971924	0.766651	-0.981033	0.966134

- 완성된 각각의 파생 변수와 착과량(int)간의 상관관계가 매우 높은 것으로 확인됨.
- 총 6개의 파생 변수들 중 비교적 상관성이 떨어지는 '새순 최소값'을 제외하고,
  남은 5개의 것들로 모델을 구동하기로 결정하였음.

다중 선형 회귀 모델

- 정해진 5개의
  최종 파생 변수들을
  각각의 X값으로 해서
  다중 선형 회귀 모델을 구성함.

- 구글링을 적극 활용하여
  비교적 간단하면서,
  성능은 잘 챙기는 모델을
  만드려고함.

- 다른 모델들에 비해
  파라미터 수정이 거의 필요없고
  우리가 알고 있는
  문법과 능력으로 어느정도
  구동의 틀이 잡혀졌기 때문에
  이 모델을 선택함.

- 또한 데이터의 행이 크게 많지 않아서,
  과적합을 우려하여 LGBM 방식을
  활용하지 않았음.

- RANDOM-FOREST는 못하겠었음.ㅠㅠ

In [63]:
```
from sklearn.model_selection import train_test_split
x = train[['새순 평균','새순 최대값', '새순 중간값', '새순 표준편차', '새순 차이']]
y = train[['착과량(int)']]
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.8, test_size=0.2, random_state = 7547)
```

모델 선택과 훈련

In [64]:
```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=5, include_bias = False)
poly.fit(x_train)
train_poly = poly.transform(x_train)
poly.get_feature_names()
```

Out[64]:
```
['x0',
 'x1',
 'x2',
 'x3',
 'x4',
 'x0^2',
 'x0 x1',
 'x0 x2',
 'x0 x3',
 'x0 x4',
 'x1^2',
 'x1 x2',
 'x1 x3',
 'x1 x4',
 'x2^2',
 'x2 x3',
 'x2 x4',
 'x3^2',
 'x3 x4'
```

In [65]:
```
from sklearn.linear_model import LinearRegression
mlr = LinearRegression()
mlr.fit(x_train, y_train)
```

Out[65]: LinearRegression()

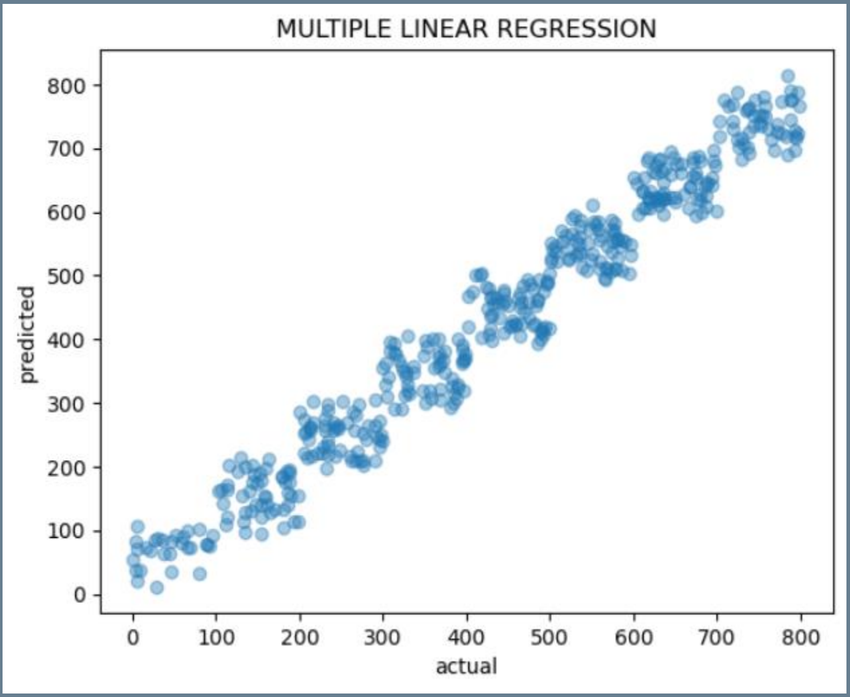
다중 선형 회귀 모델

```
In [108]: print(mlr.score(x_train, y_train))
          0.9638096524501238

In [109]: print(mlr.score(x_test, y_test))
          0.9613761586717904
```

– 평가계수 (= 결정계수) 확인결과 –

TRAIN, TEST에 관해 모두 0.96이상이 나왔으며, 차이는 약 0.002 정도가 남.



– 예측도 시각화 –

실제값과 모델의 예측값간의 상관성을 시각화 하였으며,

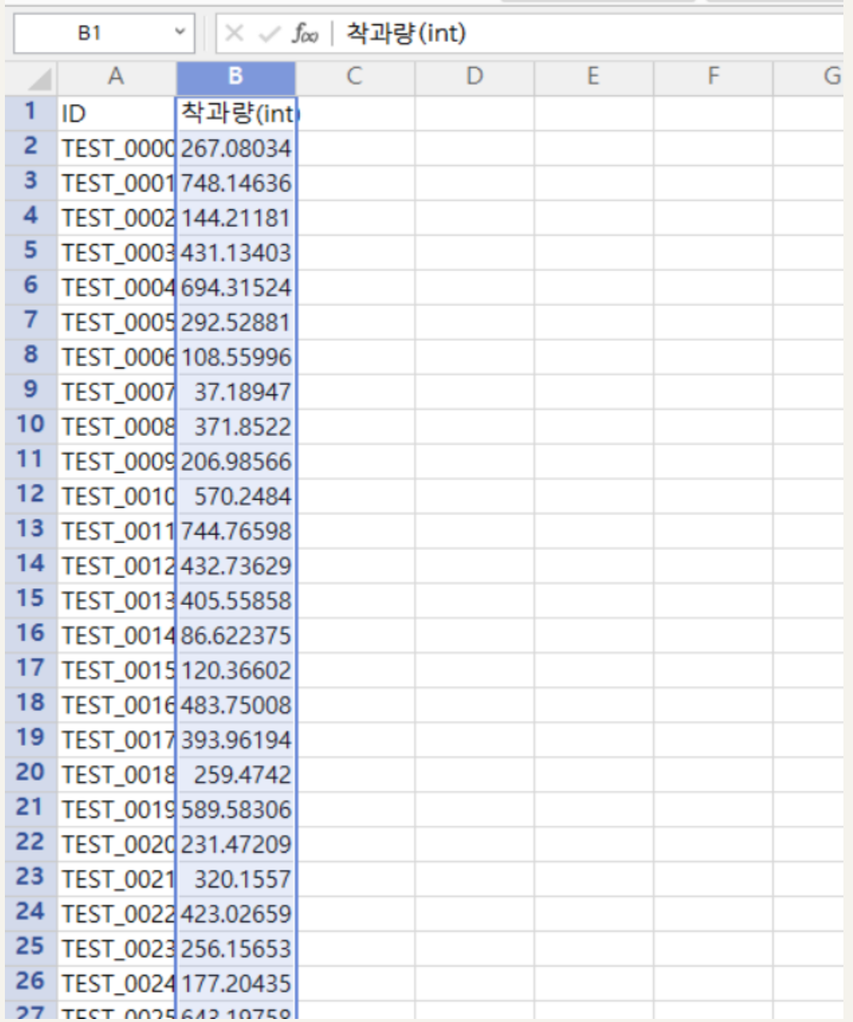
준수한 상관관계를 보임.

모델 예측 결과

노력의 결실. 우리들의 황금기. 모두들 화이팅!!

- 모델 완성후 결과는 이런식으로 도출되었음.

- 다들 귀한 겨울 방학기간에 고생이 다들 많아요...
  하지만 포기하지않고 끝까지가는 우리들의 멋진 자세를
  언젠간 보상받을 것입니다. 끝까지 가봅시다.!!



B1 | 착과량(int)

	A	B	C	D	E	F	G
1	ID	착과량(int)					
2	TEST_0000	267.08034					
3	TEST_0001	748.14636					
4	TEST_0002	144.21181					
5	TEST_0003	431.13403					
6	TEST_0004	694.31524					
7	TEST_0005	292.52881					
8	TEST_0006	108.55996					
9	TEST_0007	37.18947					
10	TEST_0008	371.8522					
11	TEST_0009	206.98566					
12	TEST_0010	570.2484					
13	TEST_0011	744.76598					
14	TEST_0012	432.73629					
15	TEST_0013	405.55858					
16	TEST_0014	86.622375					
17	TEST_0015	120.36602					
18	TEST_0016	483.75008					
19	TEST_0017	393.96194					
20	TEST_0018	259.4742					
21	TEST_0019	589.58306					
22	TEST_0020	231.47209					
23	TEST_0021	320.1557					
24	TEST_0022	423.02659					
25	TEST_0023	256.15653					
26	TEST_0024	177.20435					
27	TEST_0025	643.19759					

THANK YOU

각종 질문은 인스타 DM을 통해
정중히 부탁드리십니다.
저희는 기계가 아닙니다.ㅜㅜ

INSTAGRAM_ID
s00nuu_som
jangchun.lee.10