

# 대신러닝 프로젝트

따름이와 짹짱이

최해정 김다나 문병찬 손원준 이지수

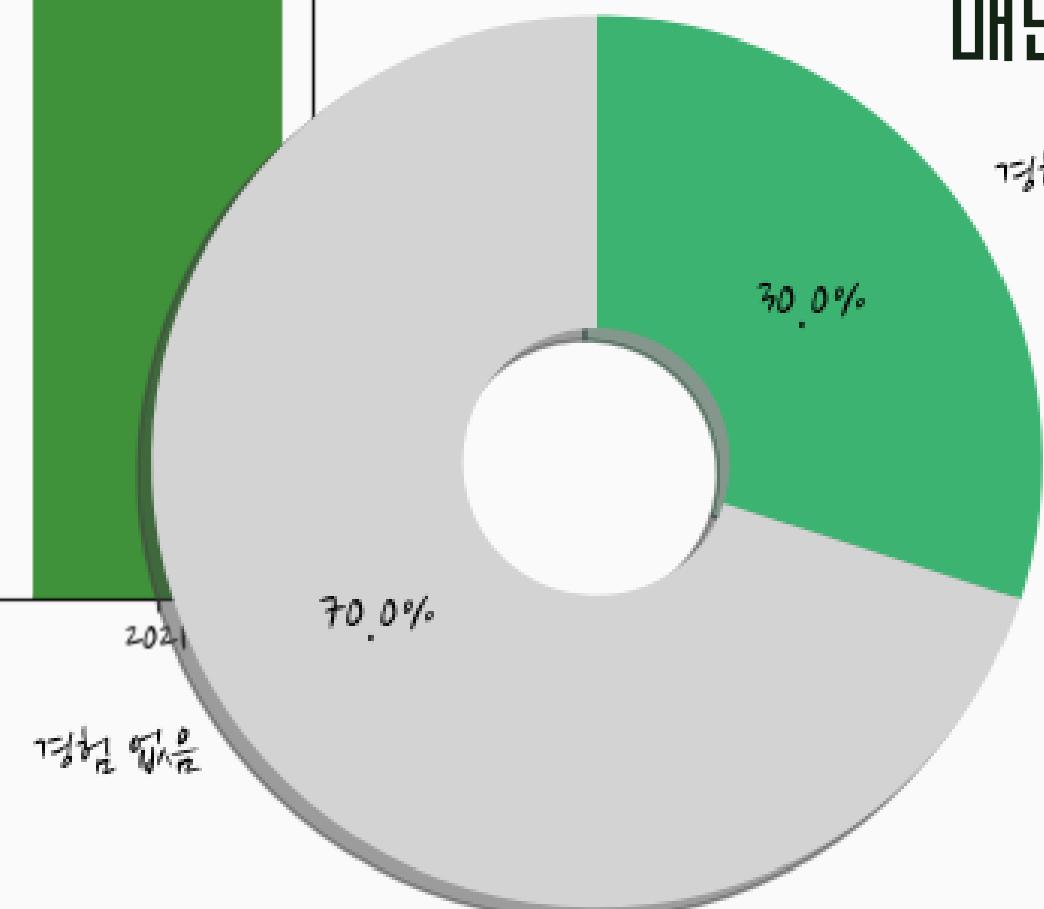
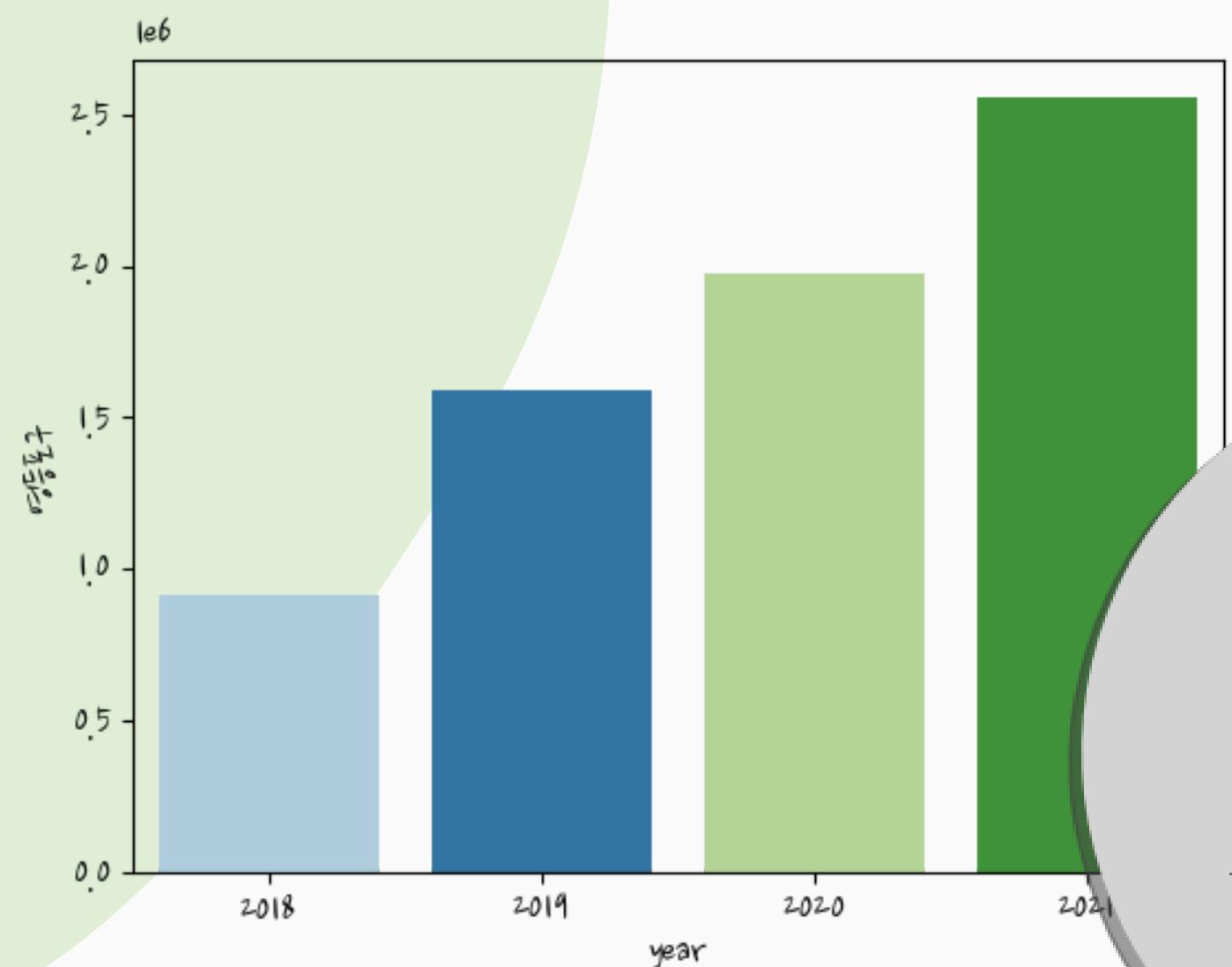




## ● 따릉이 대여량 수요예측

- 프로젝트 선정 배경
- Data Collection
- Data Preprocessing
- Model
- Predict
- Review

# 프로젝트 선정 배경



**서울시 공유 정책 만족도 1위**

올해 4월 25일 기준 누적 이용 건수 1억건

서울시민 3명 중 1명은 사용 경험

매년 증가하는 이용량



# 프로젝트 선정 배경

망가지고 방치되어 점검되지 않은 자전거들

## 100억 적자?

2017	2018	2019	2020
-42억원	-67억원	-89억원	-100억원

편성예산

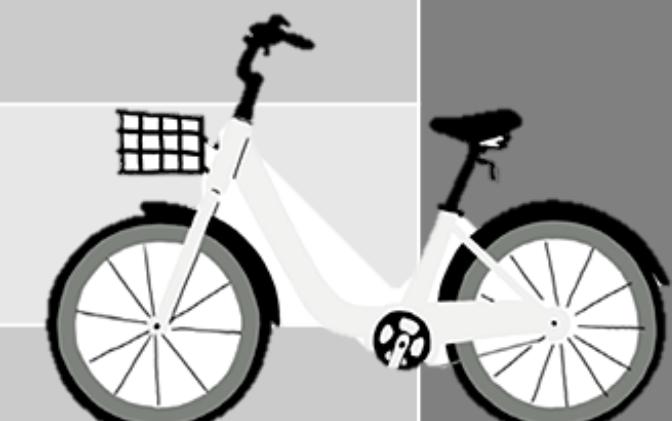
326억원 → 323억원

정확한 대여량 예측으로 예산 확립에 도움



# Data Collection

공공자전거					
	2018	2019	2020	2021	2022
일별	공공자전거 이용정보(일별)_2018.01_05.csv 공공자전거 이용정보(일별)_2018.06.csv 공공자전거 이용정보(일별)_2018.07_12.csv 서울특별시_공공자전거 일별 대여건수_(2018_2019.03).csv	공공자전거 이용정보(일별)_2019.01_05.csv 공공자전거 이용정보(일별)_2019.06.csv 공공자전거 이용정보(일별)_2019.07.csv 공공자전거 이용정보(일별)_2019.08.csv 공공자전거 이용정보(일별)_2019.09.csv 공공자전거 이용정보(일별)_2019.10.csv 공공자전거 이용정보(일별)_2019.11.csv 공공자전거 이용정보(일별)_2019.12.csv 서울특별시_공공자전거 일별 대여건수_20190601_20191130.xlsx	공공자전거 이용정보(일별)_2020.01_05.csv 공공자전거 이용정보(일별)_2020.06.csv 공공자전거 이용정보(일별)_2020.07_12.csv	공공자전거 이용정보(일별)_2021.01.csv 공공자전거 이용정보(일별)_2021.02.csv 공공자전거 이용정보(일별)_2021.03.csv 공공자전거 이용정보(일별)_2021.04.csv 공공자전거 이용정보(일별)_2021.05.csv 공공자전거 이용정보(일별)_2021.06_재산출.csv 공공자전거 이용정보(일별)_2021.07.csv 공공자전거 이용정보(일별)_2021.08.csv 공공자전거 이용정보(일별)_2021.09.csv 공공자전거 이용정보(일별)_2021.10.csv 공공자전거 이용정보(일별)_2021.11.csv 공공자전거 이용정보(일별)_2021.12.csv 서울특별시_공공자전거 일별 대여건수_21.02.01_21.06.30.csv	공공자전거 이용정보(일별)_22.06.csv 공공자전거 이용정보(일별)_22.05.csv 공공자전거 이용정보(일별)_22.04.csv 공공자전거 이용정보(일별)_22.03.csv 공공자전거 이용정보(일별)_22.02.csv 공공자전거 이용정보(일별)_22.01.csv
시간별	공공자전거 이용정보(시간대별)_2018년.zip	공공자전거 이용정보(시간대별)_2019년.zip	공공자전거 이용정보(시간대별)_2020년.zip	공공자전거 이용정보(시간대별)_2021년.zip	
위치	공공자전거 대여소 정보(22.06월 기준).csv 공공자전거 대여소 정보(21.01.31 기준).csv 공공자전거 대여소 정보(21.06월 기준).xlsx 공공자전거 대여소 정보(21.12월 기준).xlsx 대여소위치.tsv( 외부 깃허브 참조)[자치구별 대여건수 결측값 처리 참조] 상단 대여소위치.tsv( 외부 깃허브 출처) 따릉이 대여소(과거 자료 참조) -> # 카카오api를 사용한 위경도에서 주소 추출 <a href="https://github.com/vuski/SeoulBikeStationLocation">https://github.com/vuski/SeoulBikeStationLocation</a>				
날씨	황사.csv OBS_부유분진_DD_20221208065758.csv (황사-미세먼지) 날씨정보 수집 (기상청_지상(종관, ASOS) 일자료 조회서비스) <a href="https://www.data.go.kr/data/15059093/openapi.do">https://www.data.go.kr/data/15059093/openapi.do</a>				
기타	2018-2021년_년도_자치구별_주민등록인구.csv 2018-2021년_년도_자치구별_추계인구.csv 서울생활인구 파일 출처 <a href="https://data.seoul.go.kr/dataList/OA-15439/S/1/datasetView.do">https://data.seoul.go.kr/dataList/OA-15439/S/1/datasetView.do</a> 2018-2022_영등포_교통이슈_리스트.csv 공휴일 수집(한국천문연구원_특일 정보) <a href="https://www.data.go.kr/data/15012690/openapi.do">https://www.data.go.kr/data/15012690/openapi.do</a>				
출처	서울열린데이터광장 <a href="https://data.seoul.go.kr/dataList/datasetTotalList.do">https://data.seoul.go.kr/dataList/datasetTotalList.do</a> 서울시 교통정보 시스템 공지사항 데이터 (웹 크롤링) <a href="https://topis.seoul.go.kr/notice/openNoticeList.do">https://topis.seoul.go.kr/notice/openNoticeList.do</a>				



# Data Collection

year	month	day	기온	기압	풍속	운량	일강수량	최대순간 풍속풍향	최대풍속풍향	평균풍속	최다풍향	일최 심적 설	일최 심신 적설	강 수 여 부	일 일 미세먼지 농도( $\mu\text{g}/\text{m}^3$ )	휴일여부	요 일	계 절	영등포구

일시

year  
month  
day  
요일  
계절  
휴일여부

날씨

기온  
풍속  
운량  
일강수량  
평균풍속  
최다풍향

TARGET

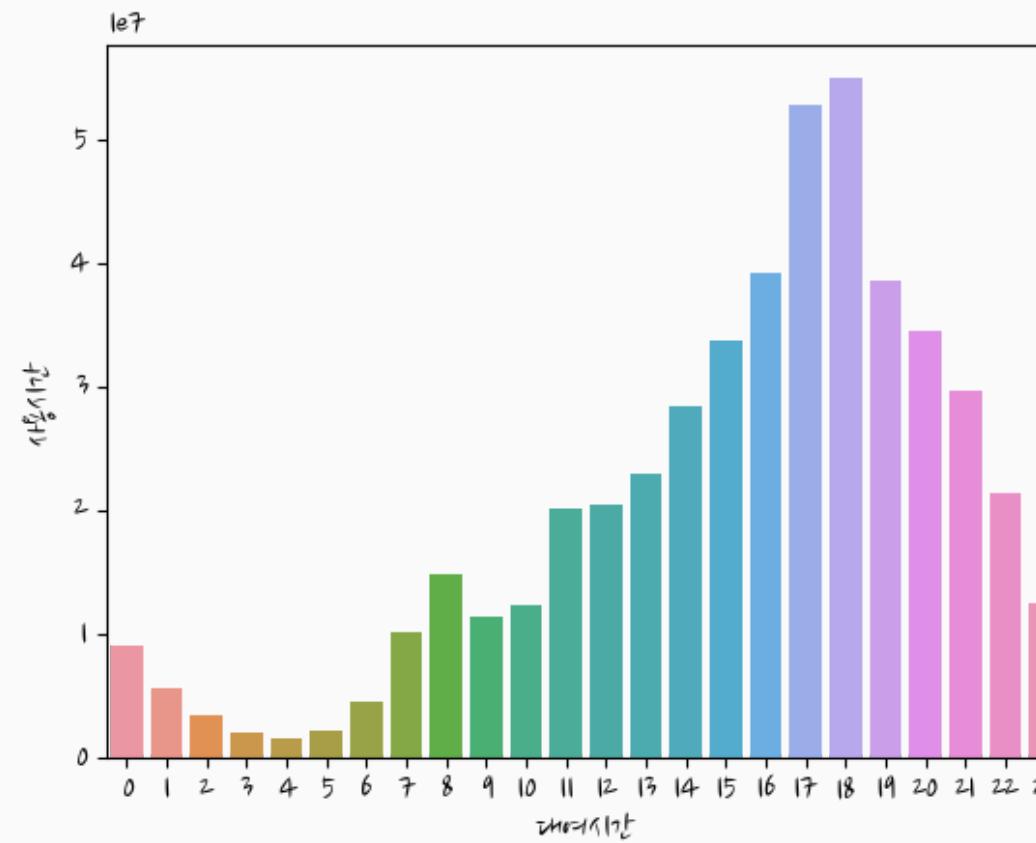
최대순간풍속풍향  
최대풍속풍향  
일최심적설  
일최심신적설  
강수여부  
일 미세먼지 농도( $\mu\text{g}/\text{m}^3$ )

영등포구

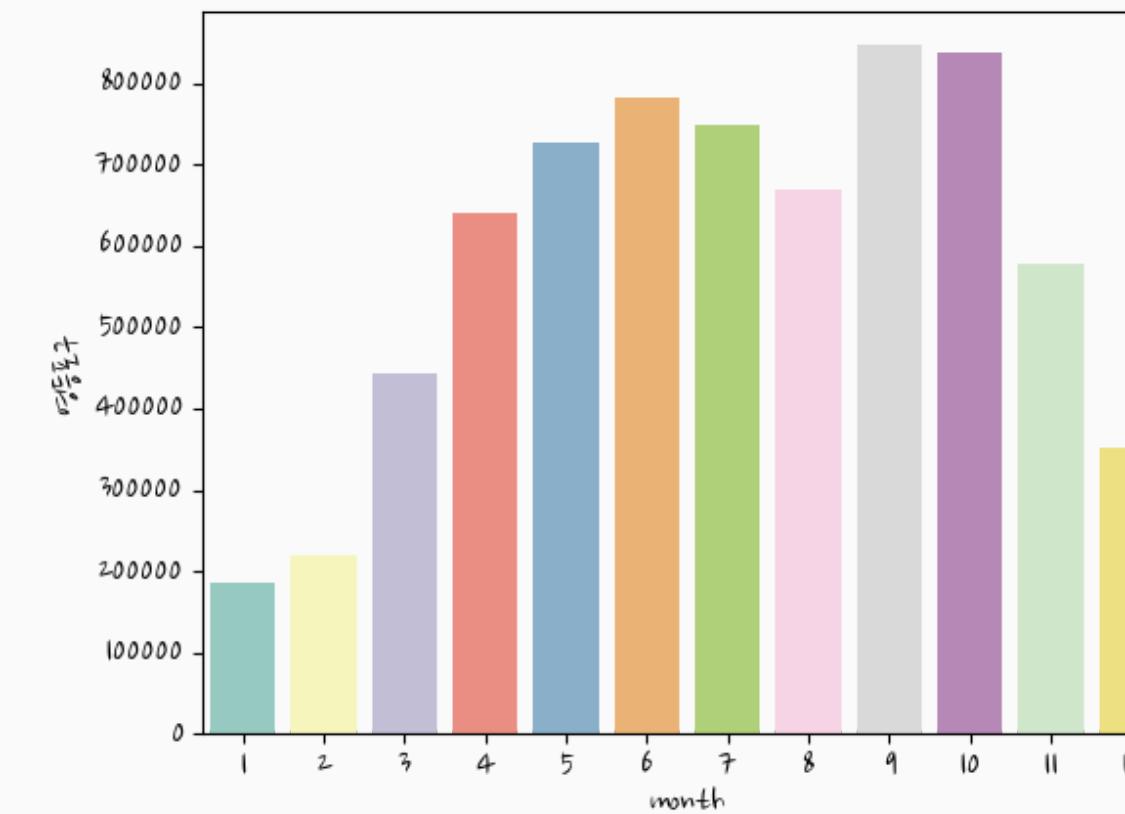


# 데이터 특징

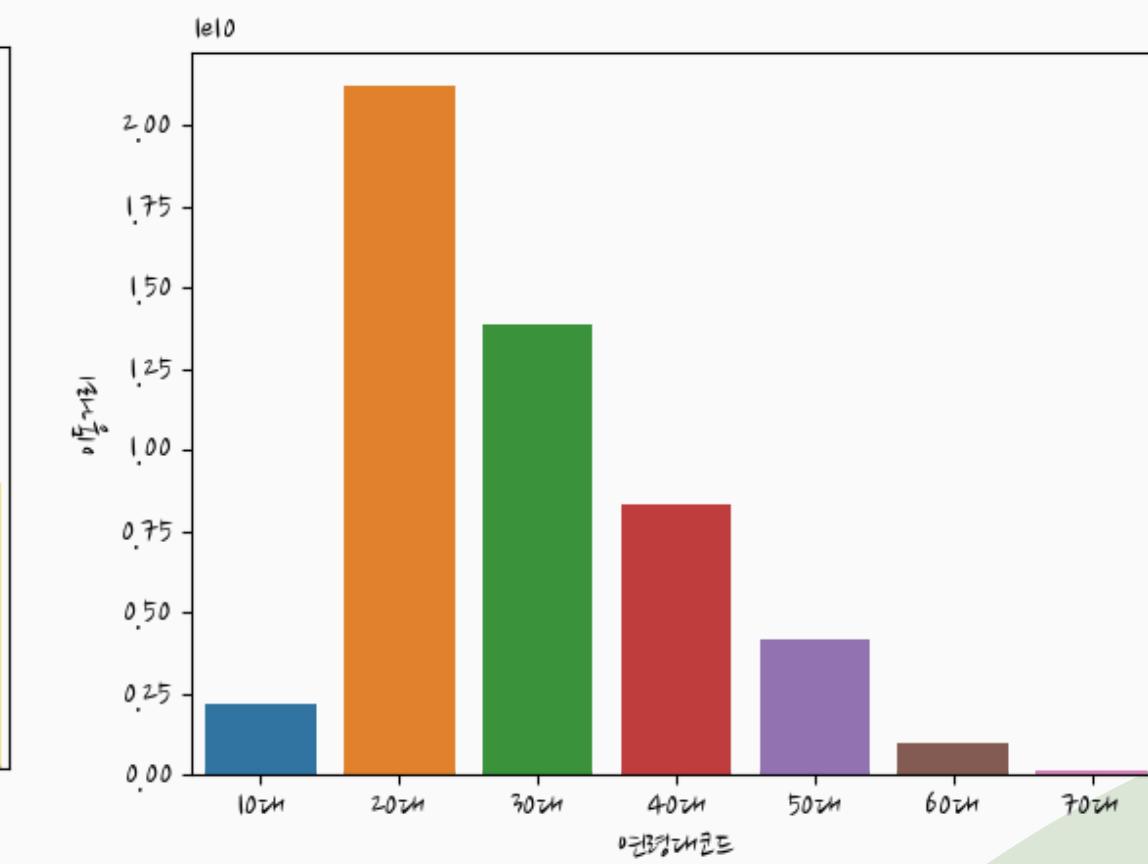
시간별 이용시간



월별 이용량

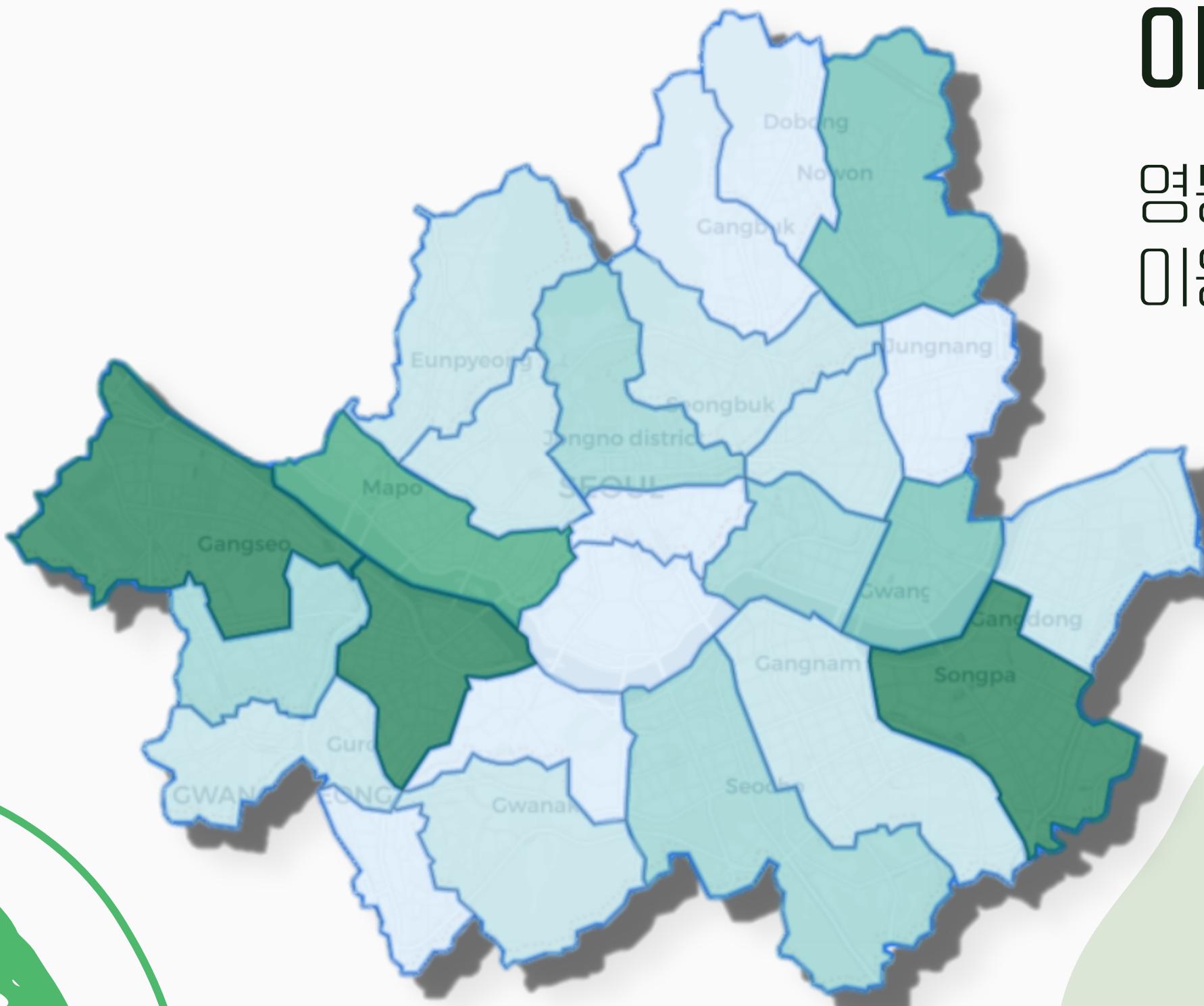


연령대별 이용량



17~18시 사용량이 많음  
봄, 가을 이용량이 많음  
20대가 가장 많이 이용함

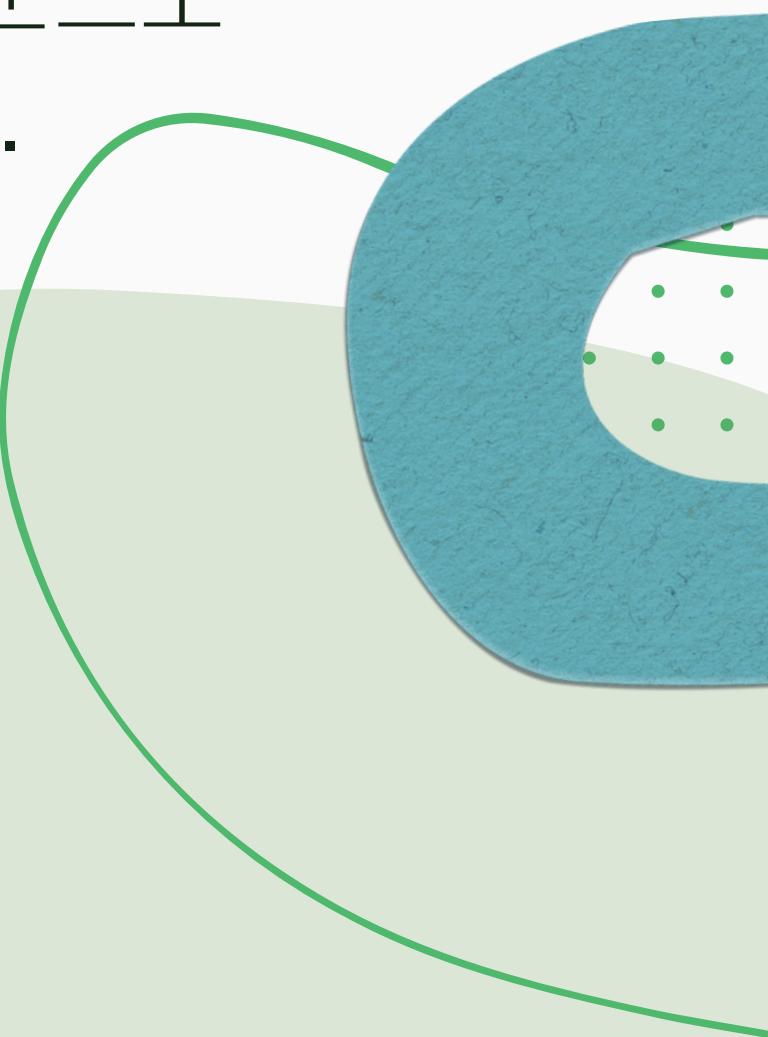
# 데이터 특징



## 이용량 1위 영등포구

영등포구, 강서구, 송파구 순으로  
이용량이 가장 많이 나타남.

자치구	대여량
0 영등포구	7031847
1 강서구	6689153
2 송파구	6654142
3 마포구	5570756
4 광진구	4520200
5 노원구	4413184



# Model

분류		회귀	
DecisionTreeClassifier	GridSearchCV	LinearRegression	Ridge, Lasso, GridSearchCV
RandomForestClassifier	VotingClassifier	RandomForestRegressor	GridSearchCV
LogisticRegression	KNeighborsClassifier	GradientBoostingRegressor	LightGBM
LGBMClassifier		XGBoostingRegressor	



# 분류 Data Preprocessing

시간대구분	운동량	탄소량	이동거리	사용시간	연령대코드
17	68.21	0.61	2650.0	26.0	20대
10	52.07	0.61	2630.0	23.0	40대
8	19.08	0.15	660.0	3.0	40대

Target

LabelEncoding

연령대코드	value_counts()
20대	210211
30대	166898
40대	101646
50대	47826
60대	10665
10대	9006
70대	5382

연령대코드	value_counts()
1	210211
2	166898
3	101646
4	47826
5	10665
0	9006
6	5382

# Model DecisionTreeClassifier

```
dt_clf = DecisionTreeClassifier(random_state=32)
dt_clf.fit(x_train, y_train)
dt_pred = dt_clf.predict(x_test)
accuracy_score(y_test, dt_pred)
```

0.37339001332402766

0.373

## GridSearchCV

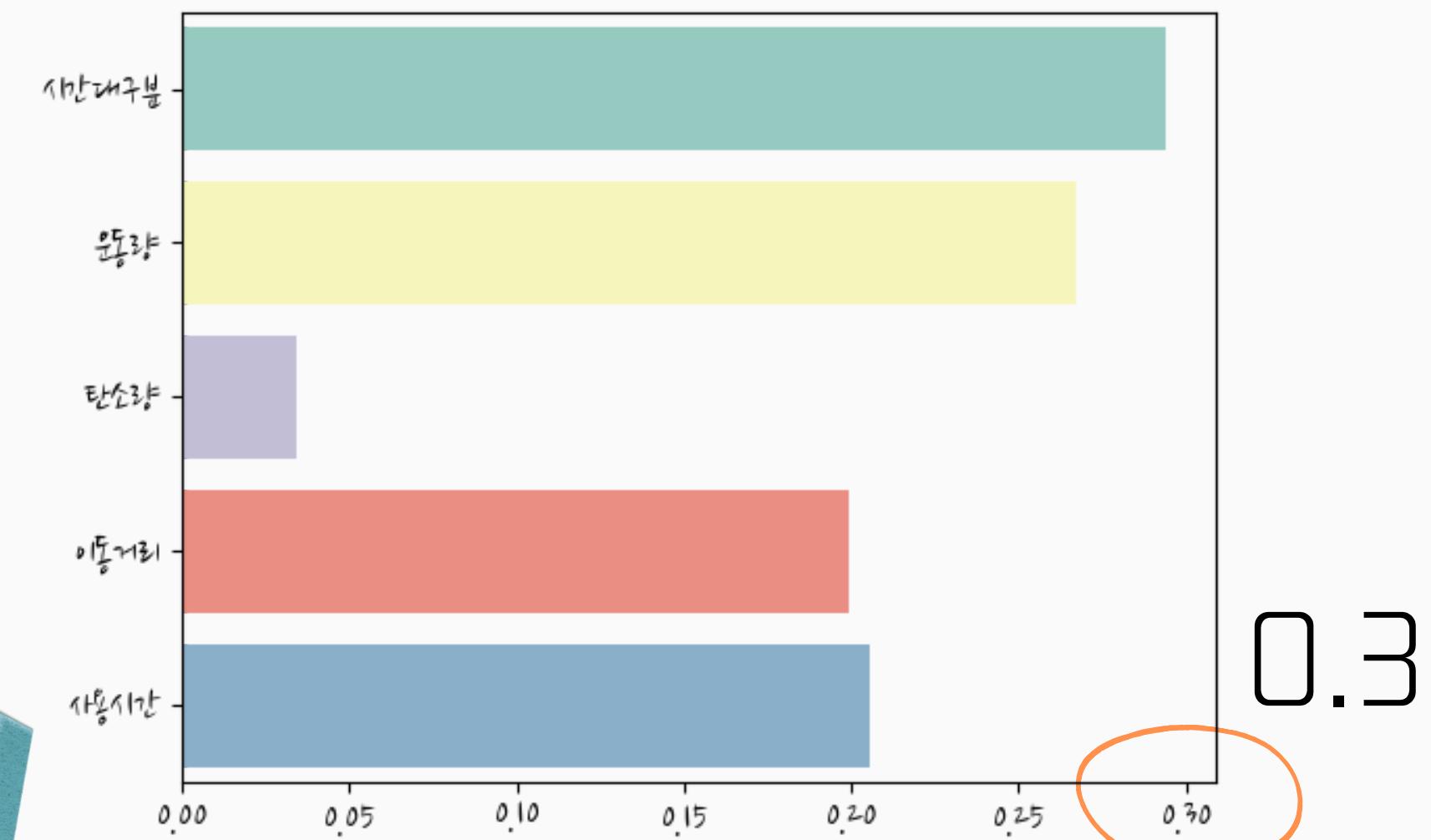
```
dt_clf = DecisionTreeClassifier(random_state=32,
                                max_depth=10,
                                min_samples_split=5,
                                min_samples_leaf=2)

dt_clf.fit(x_train, y_train)
dt_pred = dt_clf.predict(x_test)
accuracy_score(y_test, dt_pred)
```

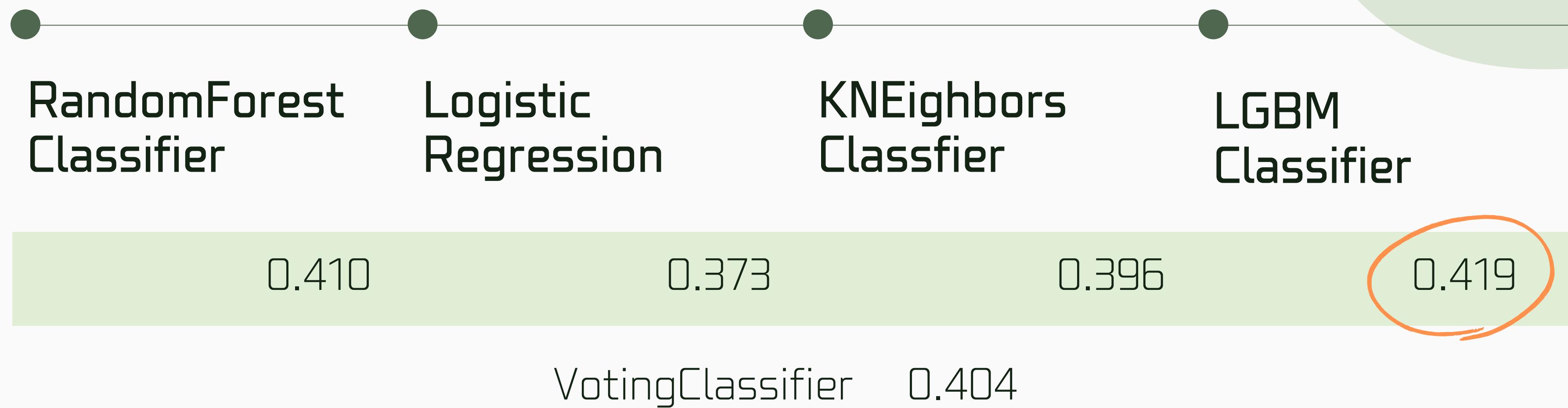
0.4018871173874029

0.401

feature importances



# Model



# 회귀 Data Preprocessing

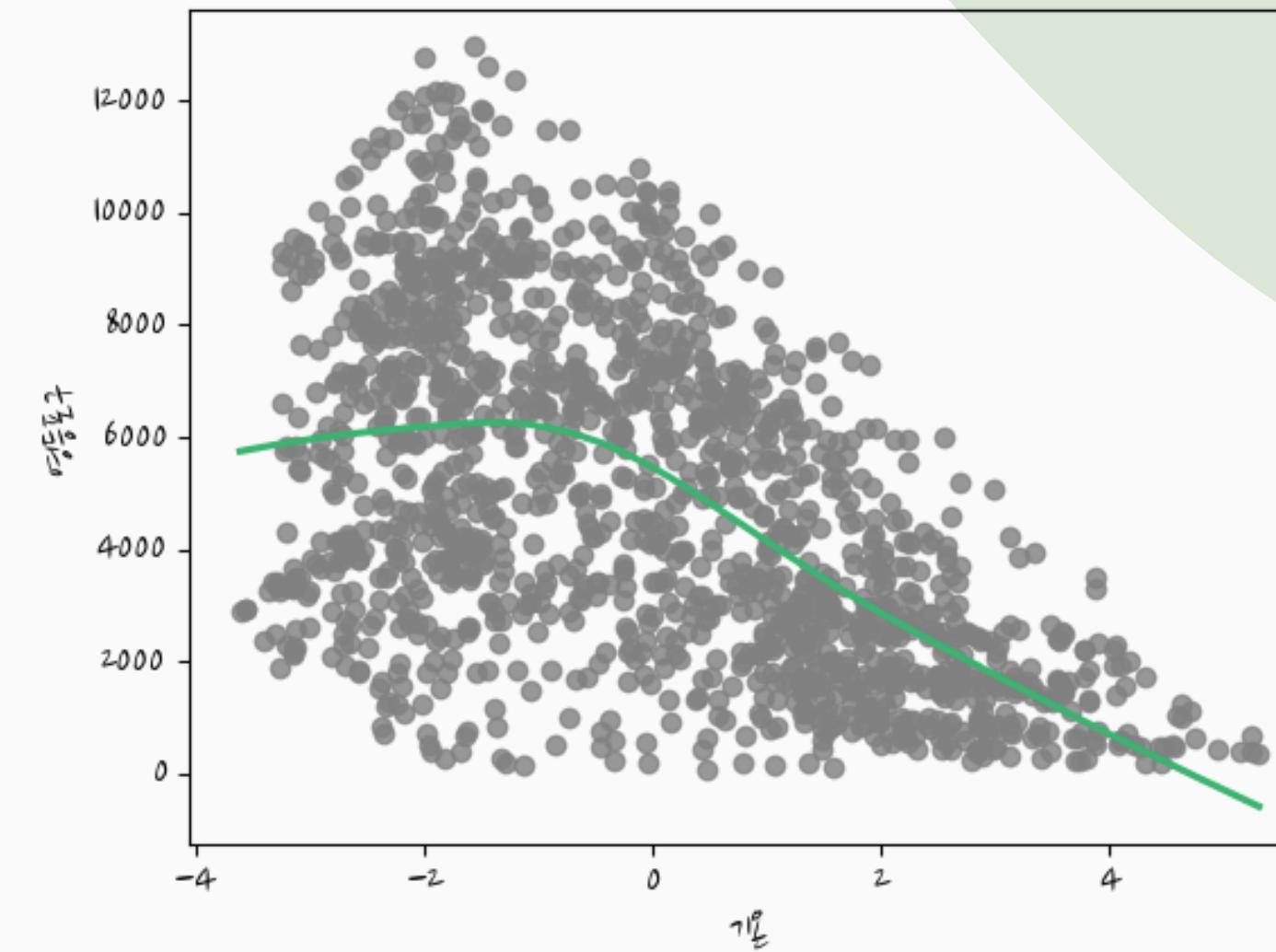
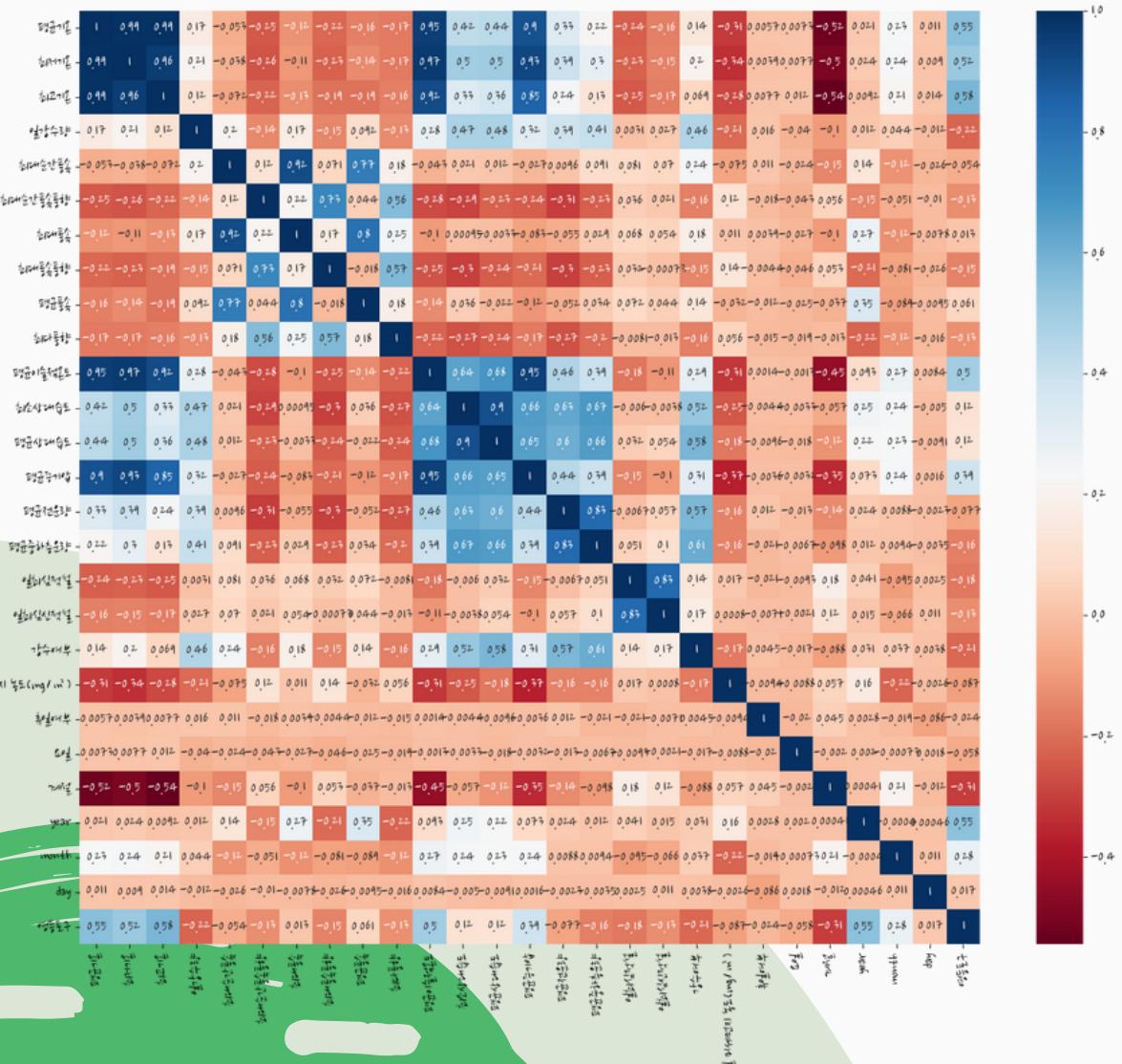
평균 기온	최저 기온	최고 기온	일강 수량	최대 간증 속	최대 순간 풍향	최대 풍속	최대 풍 속 풍향	평균 풍 속	최다 풍향	...	일 최심신적 설	강수여부	일 미세먼지 농도 ( $\mu\text{g}/\text{m}^3$ )	휴일여부	요일	계절	year	month	day	영등포구
13.6	6.9	19.1	86.9	13.4	180	7.3	230	4.2	270	...	0.0	1.0	0.0	0	3	3	2020	11	19	2553.0
23.3	19.9	27.5	0.0	9.7	180	5.6	200	2.7	200	...	0.0	0.0	19.0	0	4	3	2020	9	4	10969.0
24.6	19.2	30.9	0.0	7.8	290	4.4	290	2.0	290	...	0.0	0.0	48.0	1	5	2	2020	6	6	9061.0

Target

영등포구의  
하루 대여량 예측

# 회귀 Data Preprocessing

## 선형관계 확인

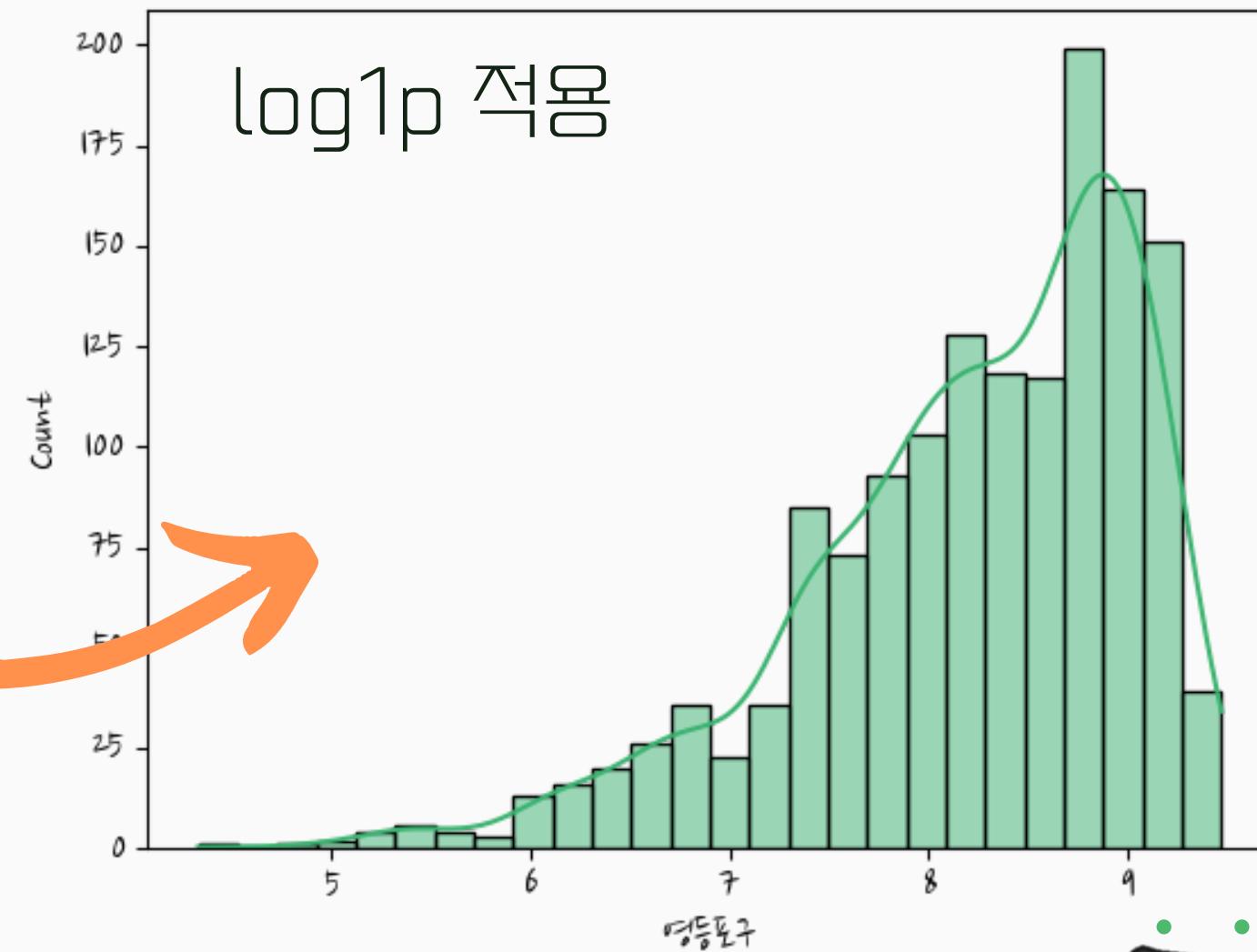
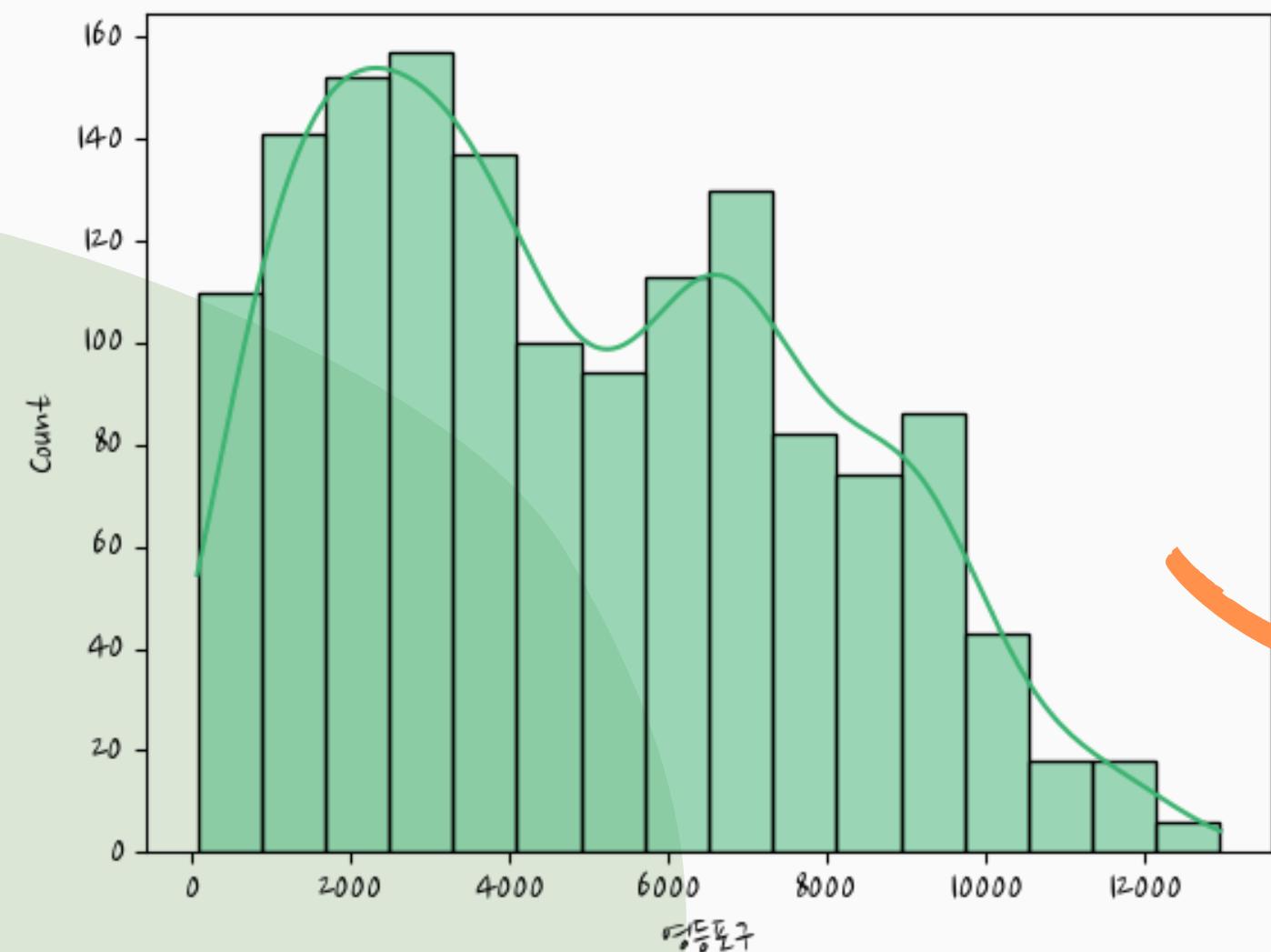


기온과 -0.58의 상관계수

약한 음의 선형관계

# 회귀 Data Preprocessing

## 정규분포 확인

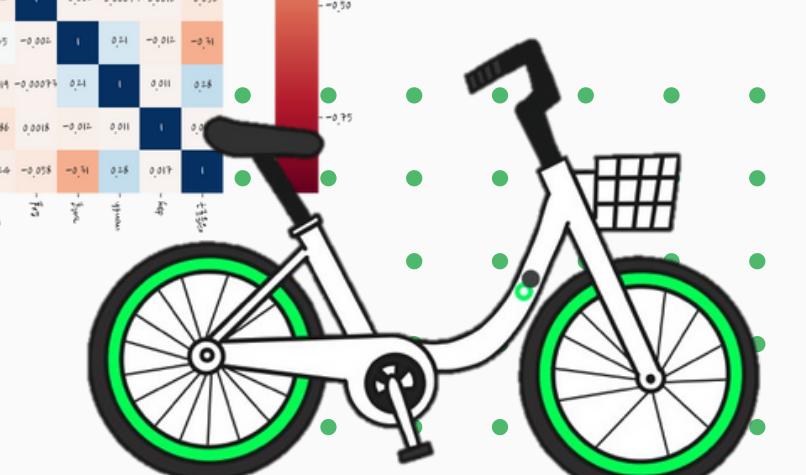
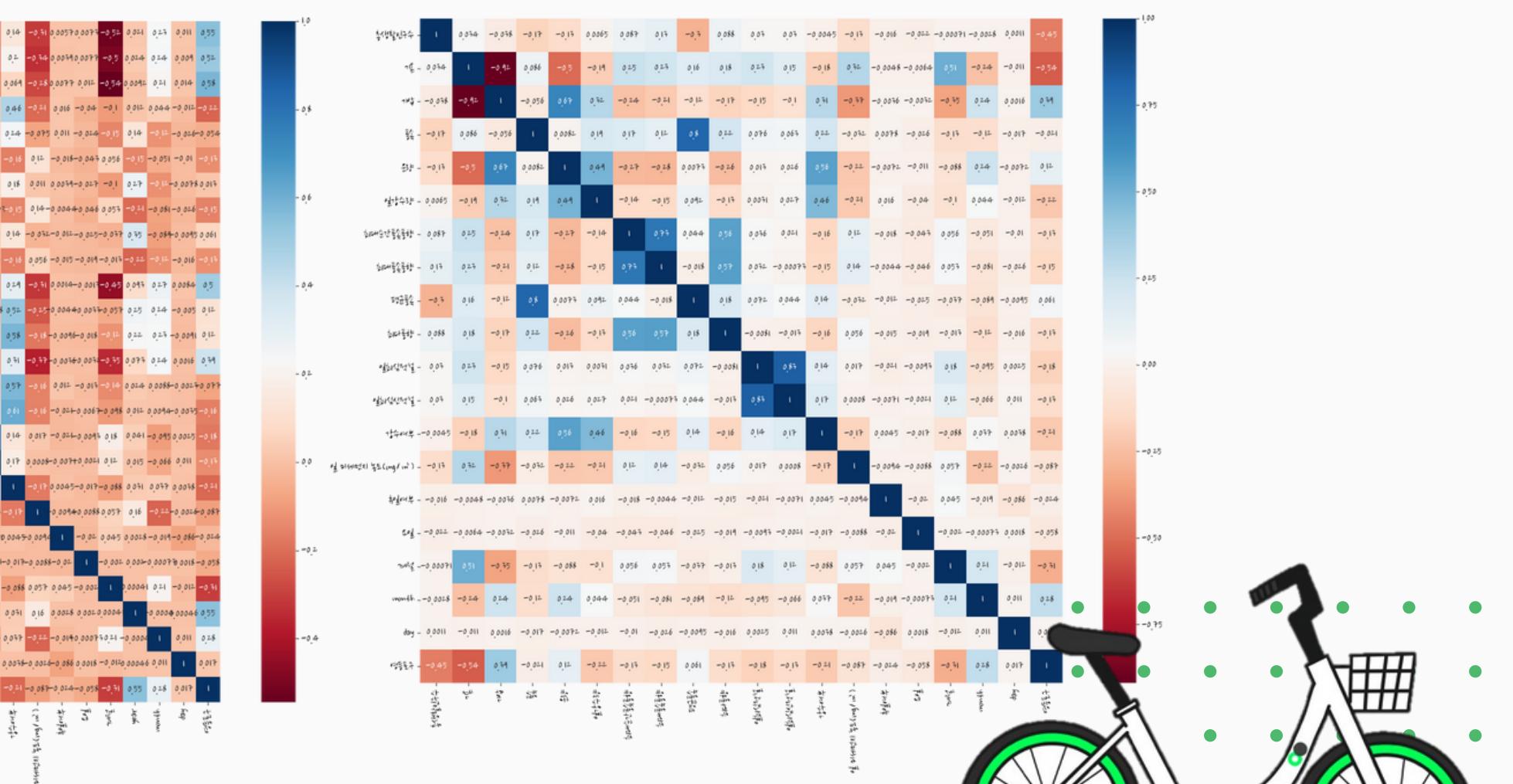


# 회귀 Data Preprocessing

corr		
평균기온	최고기온	0.989298
최고기온	평균기온	0.989298
평균기온	최저기온	0.989262
최저기온	평균기온	0.989262
평균이슬점온도	최저기온	0.967202
최저기온	평균이슬점온도	0.967202
최고기온	최저기온	0.961977
최저기온	최고기온	0.961977
평균기온	평균이슬점온도	0.953476
평균이슬점온도	평균기온	0.953476
최대순간풍속	최대풍속	0.916619
최대풍속	최대순간풍속	0.916619
최소상대습도	평균상대습도	0.904746
평균상대습도	최소상대습도	0.904746
평균증기압	평균기온	0.898762
평균기온	평균증기압	0.898762
최고기온	평균증기압	0.851163
평균증기압	최고기온	0.851163
평균증하층운량	평균전운량	0.831119
평균전운량	평균증하층운량	0.831119

feature들 간의 상관관계 확인

PCA 적용



# 회귀 Data Preprocessing

## 왜곡도 확인

```
• • • •  
• features_index = dd.dtypes[dd.dtypes != 'object'].index  
• skew_features= dd[features_index].apply(lambda x : skew(x))  
• skew_features_top = skew_features[skew_features>1]  
• skew_features_top.sort_values(ascending=False)
```

일최심신적설	12.166799
일최심적설	8.847381
일강수량	5.001375
일 미세먼지 농도( $\mu\text{g}/\text{m}^3$ )	3.246978
풍속	1.201223
총생활인구수	1.172244

2 이상인 값에 log1p 적용

```
dd[skew_features_top.index]= np.log1p(dd[skew_features_top.index])
```

## 원애햄인코딩

```
dd = pd.get_dummies(test5)
```

```
dd.columns  
Index(['총생활인구수', '기온', '기압', '풍속', '운량', '일강수량', '최대순간풍속풍향', '최대풍속풍향', '평균풍속',  
'최다풍향', '일최심적설', '일최심신적설', '일 미세먼지 농도( $\mu\text{g}/\text{m}^3$ )', '영등포구', '강수여부_0.0',  
'강수여부_1.0', '휴일여부_0', '휴일여부_1', '요일_0', '요일_1', '요일_2', '요일_3', '요일_4',  
'요일_5', '요일_6', '계절_1', '계절_2', '계절_3', '계절_4', 'month_1', 'month_10',  
'month_11', 'month_12', 'month_2', 'month_3', 'month_4', 'month_5',  
'month_6', 'month_7', 'month_8', 'month_9', 'day_1', 'day_10', 'day_11',  
'day_12', 'day_13', 'day_14', 'day_15', 'day_16', 'day_17', 'day_18',  
'day_19', 'day_2', 'day_20', 'day_21', 'day_22', 'day_23', 'day_24',  
'day_25', 'day_26', 'day_27', 'day_28', 'day_29', 'day_3', 'day_30',  
'day_31', 'day_4', 'day_5', 'day_6', 'day_7', 'day_8', 'day_9'],  
dtype='object')
```

```
df.shape
```

(1461, 20)

```
dd.shape
```

(1461, 72)

수치로 계산되면  
안되는 값은  
원애햄인코딩 적용

# Model LinearRegression

전처리 전

```
lr_reg = LinearRegression()
lr_reg.fit(X_train, y_train)
pred = lr_reg.predict(X_test)
mse = mean_squared_error(y_test, pred)
rmse = np.sqrt(mse)
print(f'lr_reg MSE: {mse}, RMSE: {rmse}, R2: {r2_score(y_test, pred)}')

ridge_reg = Ridge()
ridge_reg.fit(X_train, y_train)
pred = ridge_reg.predict(X_test)
mse = mean_squared_error(y_test, pred)
rmse = np.sqrt(mse)
print(f'ridge_reg MSE: {mse}, RMSE: {rmse}, R2: {r2_score(y_test, pred)}')

lasso_reg = Lasso()
lasso_reg.fit(X_train, y_train)
pred = lasso_reg.predict(X_test)
mse = mean_squared_error(y_test, pred)
rmse = np.sqrt(mse)
print(f'lasso_reg MSE: {mse}, RMSE: {rmse}, R2: {r2_score(y_test, pred)}')
```

전처리 후

lr\_reg MSE: 0.163,  
RMSE: 0.40472  
R2: 0.76041

ridge\_reg MSE: 0.163  
RMSE: 0.40470  
R2: 0.76044

lasso\_reg MSE: 0.493  
RMSE: 0.702  
R2: 0.277

# Model LinearRegression

lr\_reg MSE: 0.163,  
RMSE: 0.40472  
R2: 0.76041

ridge\_reg MSE: 0.163  
RMSE: 0.40470  
R2: 0.76044

lasso\_reg MSE: 0.493  
RMSE: 0.702  
R2: 0.277

GridSearchCV 파라미터 적용

best\_params

'max\_depth' : 5  
'min\_samples\_split' : 2  
'min\_samples\_leaf' : 1

lr\_reg MSE: 0.163  
RMSE: 0.40472  
R2: 0.76041

ridge\_reg MSE: 0.164  
RMSE: 0.40498  
R2: 0.76010

lasso\_reg MSE: 0.168  
RMSE: 0.41  
R2: 0.753

lasso 향상

# Model RandomForestRegressor

GridSearchCV 파라미터 적용

`best_params`

'max\_depth' : 30  
'min\_samples\_split' : 2  
'min\_samples\_leaf' : 1

MSE: 0.1228  
RMSE: 0.3504  
R2: 0.8203

MSE: 0.1159  
RMSE: 0.3404  
R2: 0.8304

성능 향상

# Model GradientBoostingRegressor

```
gb_ref = GradientBoostingRegressor(random_state=32)
gb_ref.fit(x_train, y_train)
gb_pred = gb_ref.predict(x_test)

mse = mean_squared_error(y_test, pred)
rmse = np.sqrt(mse)
print('MSE:', mse)
print('RMSE: ', np.round(rmse,3))
print(f'R2: {r2_score(y_test, pred)}')
```

MSE: 0.12283794091253145

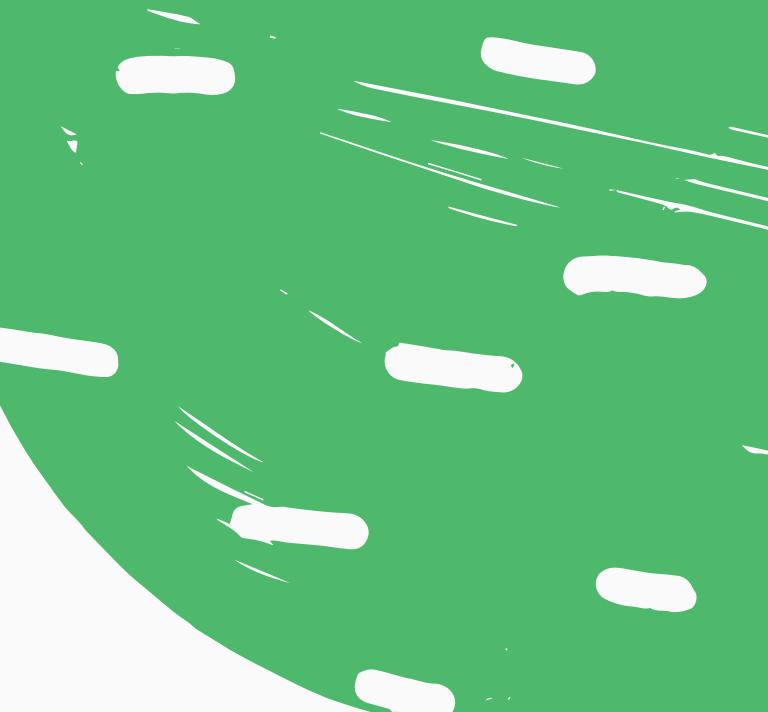
RMSE: 0.35

R2: 0.8203311327580028

MSE: 0.122

RMSE: 0.35

R2: 0.82



# Model LGBMRegressor

LGBM

test5.shape

(1461, 27)

10,000 개 이하

RMSE : 0.352

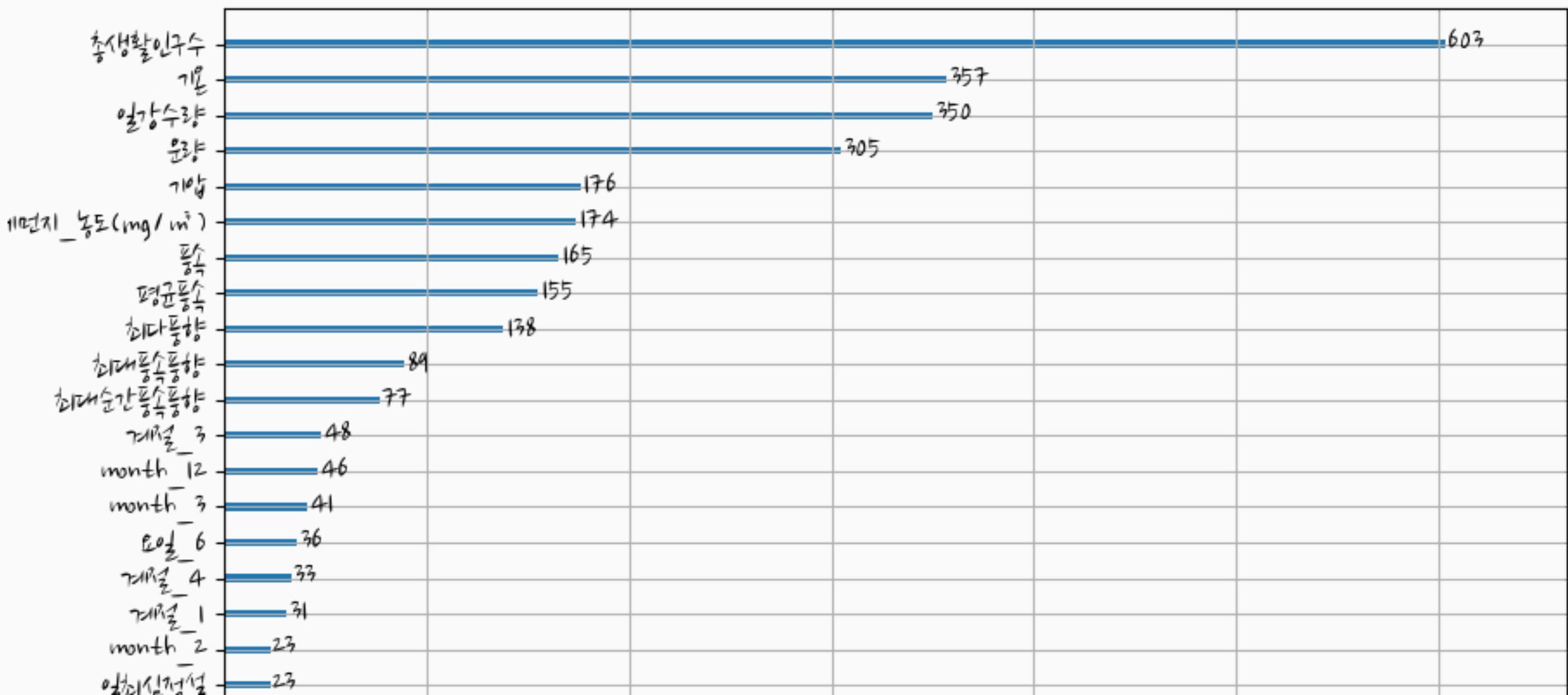
R2 : 0.819

과적합

feature importance

총생활 인구수, 기온, 일강수량

Feature importance



# Model XGBoostingRegressor

학습용 8 : 테스트용 2

최종 학습용 9 : 검증용 1

MSE: 0.122

RMSE: 0.35

R2: 0.82

[0]	train-rmse:6.78144	eval-rmse:6.78472
[1]	train-rmse:6.44733	eval-rmse:6.45758
[2]	train-rmse:6.12977	eval-rmse:6.14542
[3]	train-rmse:5.82805	eval-rmse:5.84722
[4]	train-rmse:5.54135	eval-rmse:5.56566
[5]	train-rmse:5.26899	eval-rmse:5.29960
[6]	train-rmse:5.01015	eval-rmse:5.04497
[7]	train-rmse:4.76424	eval-rmse:4.80204
[8]	train-rmse:4.53060	eval-rmse:4.57229
[9]	train-rmse:4.30866	eval-rmse:4.35620
[10]	train-rmse:4.09765	eval-rmse:4.15464
[11]	train-rmse:3.89729	eval-rmse:3.95726
[12]	train-rmse:3.70684	eval-rmse:3.77078
[13]	train-rmse:3.52599	eval-rmse:3.59499
[14]	train-rmse:3.35397	eval-rmse:3.42928
[15]	train-rmse:3.19067	eval-rmse:3.27119
[16]	train-rmse:3.03563	eval-rmse:3.12106
[17]	train-rmse:2.88815	eval-rmse:2.97646
[18]	train-rmse:2.74811	eval-rmse:2.83929
[19]	train-rmse:2.61504	eval-rmse:2.71130
[20]	train-rmse:2.48850	eval-rmse:2.58774
[21]	train-rmse:2.36840	eval-rmse:2.47125
[22]	train-rmse:2.25419	eval-rmse:2.36068
[23]	train-rmse:2.14573	eval-rmse:2.25597
[24]	train-rmse:2.04273	eval-rmse:2.15661

# Model

Linear  
Regression

MSE: 0.163  
RMSE: 0.404  
R2: 0.76

RandomForest  
Regressor

MSE: 0.115  
RMSE: 0.34  
R2: 0.83

Gradient  
Boosting  
Regressor

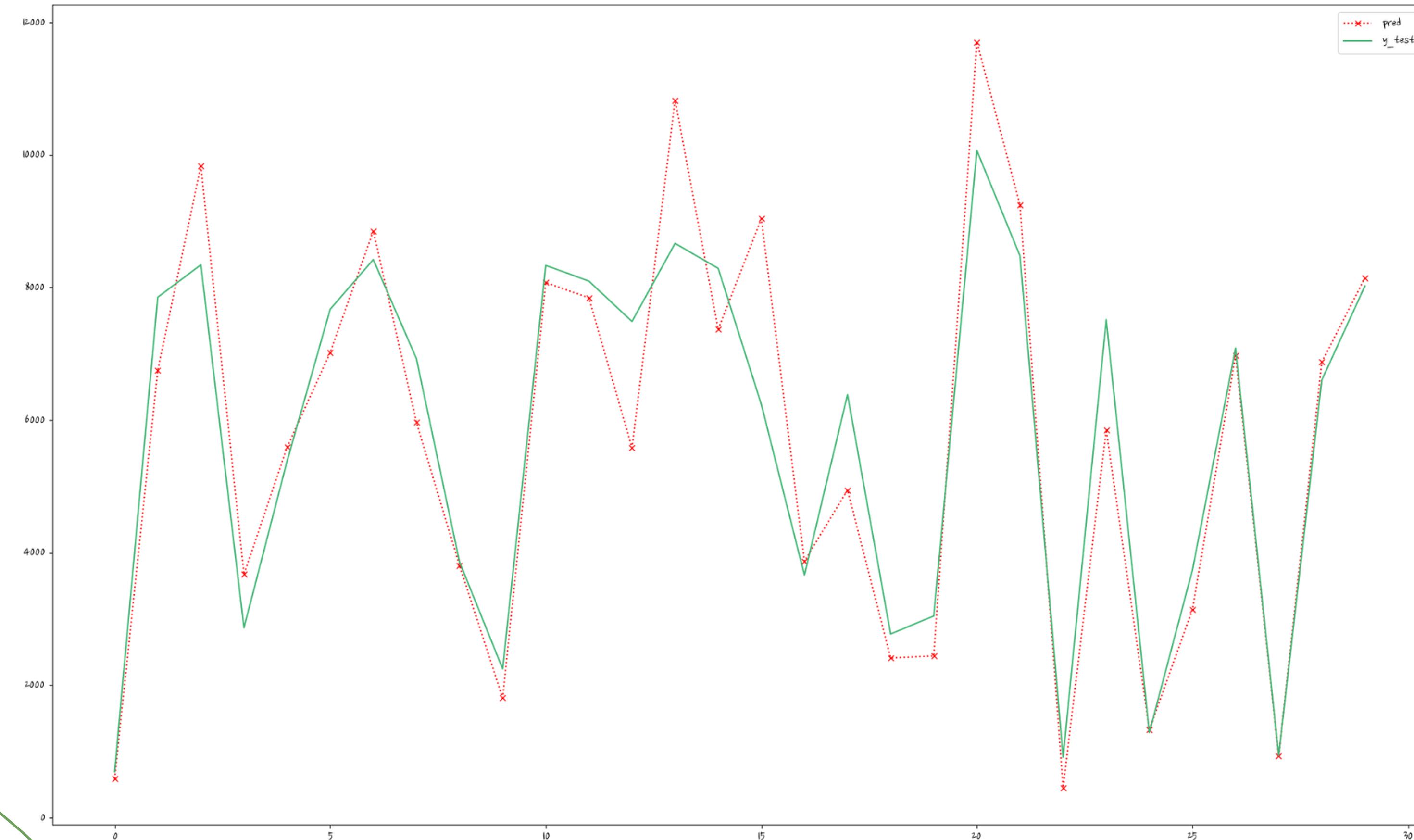
MSE: 0.118  
RMSE: 0.361  
R2: 0.808

XGBoosting  
Regressor

MSE: 0.122  
RMSE: 0.349  
R2: 0.821

# RandomForest Regressor

MSE: 0.115  
RMSE: 0.34  
R2: 0.83



# 실제 예측 결과 2022년 데이터 추가

## RandomForestRegressor

```
rf_clf_pred_22 = rf_clf.predict(X_22)
mse = mean_squared_error(y_ans, rf_clf_pred_22)
rmse = np.sqrt(mse)
print()
print('### 랜덤포레스트 예측모델 결과 ###')
print(f'MSE: {mse}, RMSE: {rmse}, R2: {r2_score(y_ans, rf_clf_pred_22)}')

### 랜덤포레스트 예측모델 결과 ###
MSE: 0.4679549961908833, RMSE: 0.684072361808956, R2: -0.18010575013513486
```

## LinearRegression Lasso

```
lasso_reg_pred_22 = lasso_reg.predict(X_22)
mse = mean_squared_error(y_ans, lasso_reg_pred_22)
rmse = np.sqrt(mse)
print()
print('###라쏘 예측모델 결과 ###')
print(f'MSE: {mse}, RMSE: {rmse}, R2: {r2_score(y_ans, lasso_reg_pred_22)}')

###라쏘 예측모델 결과 ###
MSE: 0.3204748439431654, RMSE: 0.5661049760805547, R2: 0.19181500531149276
```

MSE: 0.467

RMSE: 0.684

R2: -0.18

과적합



MSE: 0.324

RMSE: 0.566

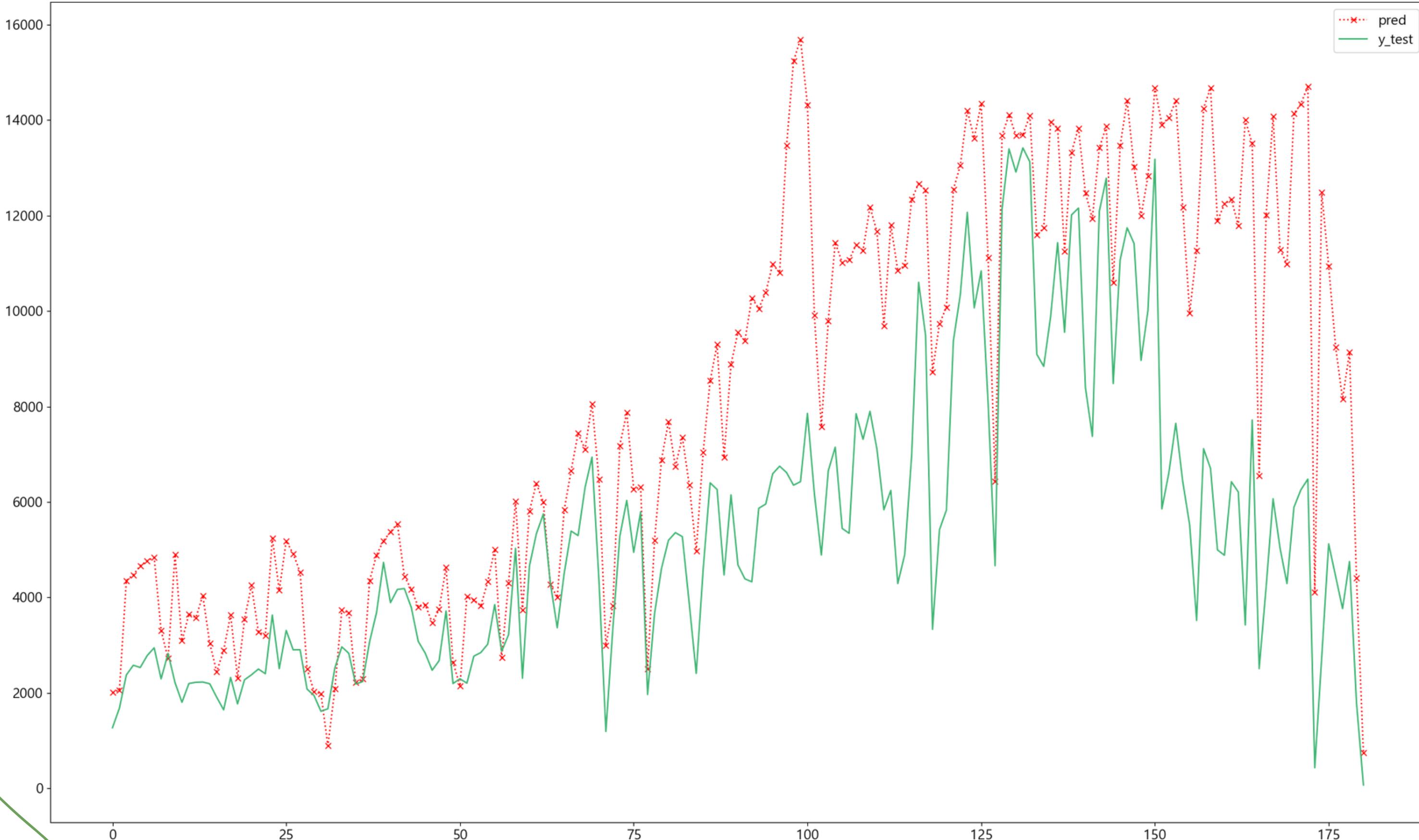
R2: 0.191



# LinearRegression Lasso

MSE: 0.324  
RMSE: 0.566  
R2: 0.191

Lasso 결과 (정답-예측값)



# Review

전처리 데이터 多  
상관관계가 극명한 feature 無

- 날씨 이외에 상관관계가 있는 컬럼을 찾지 못함
- 관련이 있을 것이라 예상한 컬럼과 연관이 없어 시간적 소비가 심하였음 (검색량, 시간별인구, 따릉이 거치대 수, 교통통제일 등)
- 어플 데이터나 대여소 데이터를 제대로 활용하지 못함
- 년도별 따릉이 운영 개수 증가세 반영이 2022년 실제 자료 비교에서 반영되지 않음
- 모든 구를 확인하지 못하여 자치구마다 다른 모델을 만들어야함.
- 실질적 예산확인 문제에 도움이 되지않음



상관관계가 낮은 이유

피할 수 없으면 즐겨라!