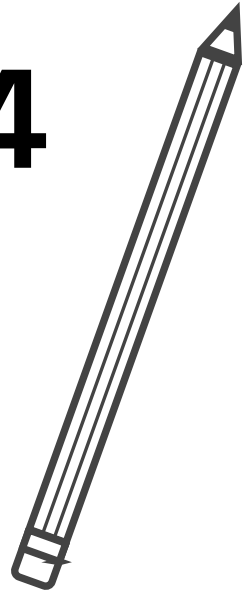
Linear Regression

Regression-team-4



장지혜 정다은

데이터 살펴보기 (전처리: 피처설정/피처정규화)

```
1 bike.shape
```

```
(8760, 14)
```

```
1 bike.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8760 entries, 0 to 8759
Data columns (total 14 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Date                       8760 non-null   object
 1   Rented Bike Count          8760 non-null   int64
 2   Hour                       8760 non-null   int64
 3   Temperature(℃)             8760 non-null   float64
 4   Humidity(%)                8760 non-null   int64
 5   Wind speed (m/s)           8760 non-null   float64
 6   Visibility (10m)           8760 non-null   int64
 7   Dew point Temperature(℃)   8760 non-null   float64
 8   Solar Radiation (MJ/m2)    8760 non-null   float64
 9   Rainfall(mm)               8760 non-null   float64
 10  Snowfall (cm)              8760 non-null   float64
 11  Seasons                    8760 non-null   object
 12  Holiday                    8760 non-null   object
 13  Functioning Day            8760 non-null   object
dtypes: float64(6), int64(4), object(4)
memory usage: 958.2+ KB
```

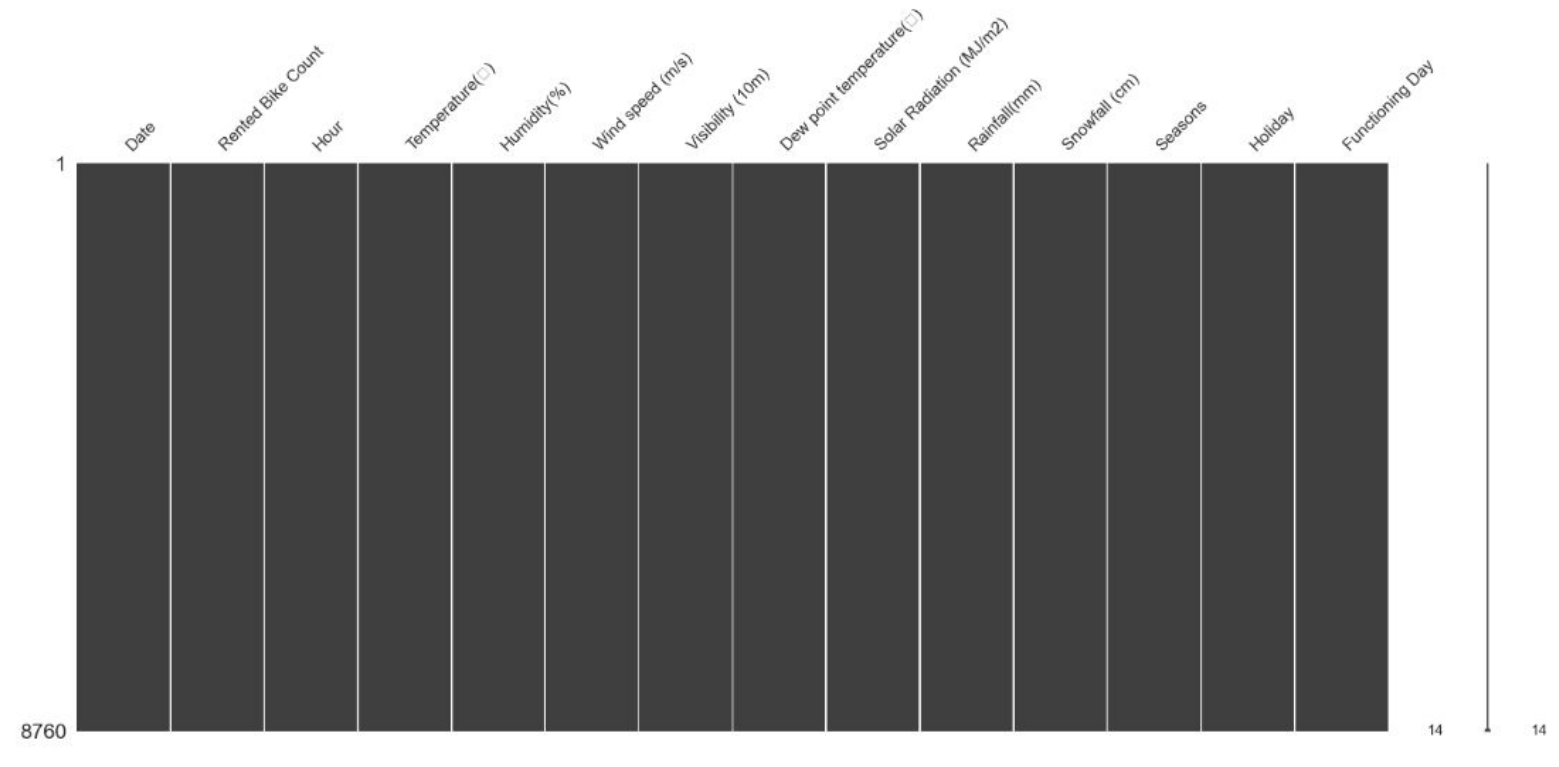
Dataset: 'SeoulBikeData.csv'

Linear Regression Model-Bike Sharing Demand

- Date : 날짜 일/월/연도 (01/12/2017~30/11/2018)
- Rented Bike Count : 자전거 대여 횟수
- Hour : 시간(1시간 간격)
- Temperature(℃) : 기온
- Humidiy(%) : 습도
- Wind speed(m/s) : 풍속
- Visibility(10m) : 가시거리
- Dew point Temperature(℃) : 이슬점
- Solar Radiation (NJ/m2) : 태양 복사량
- Rainfall(mm) : 강우량
- Snowfall (cm) : 강설량
- Seasons: 계절 (Winter, Spring, Summer, Atumn)
- Holiday: 휴일 (Holiday, No_Holiday)
- Functioning Day : 운영일 (Yes, No)

결측치 확인

```
1 msno.matrix(bike);
```



대략적인 통계와 데이터 확인

```
1 bike["Functioning Day"].value_counts()
```

```
Yes    8465
No      295
Name: Functioning Day, dtype: int64
```

```
1 bike["Holiday"].value_counts()
```

```
No Holiday    8328
Holiday        432
Name: Holiday, dtype: int64
```

```
1 bike["Date"].value_counts()
```

```
05/03/2018    24
13/02/2018    24
13/05/2018    24
07/10/2018    24
31/05/2018    24
              ..
23/12/2017    24
26/05/2018    24
03/02/2018    24
31/01/2018    24
13/10/2018    24
Name: Date, Length: 365, dtype: int64
```

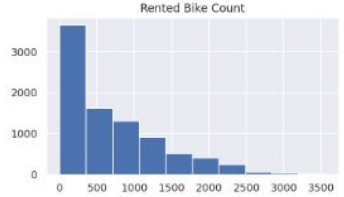
```
1 bike["Seasons"].value_counts()
```

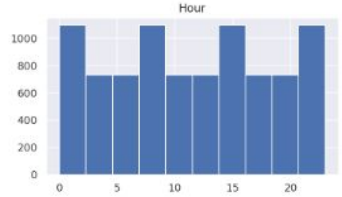
```
Summer    2208
Spring    2208
Autumn    2184
Winter    2160
Name: Seasons, dtype: int64
```

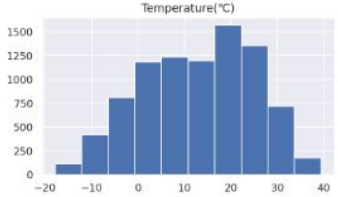
```
1 bike.describe()
```

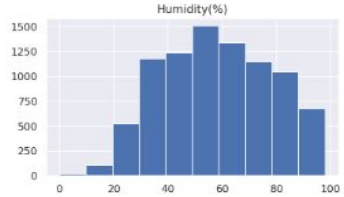
	Rented Bike Count	Hour	Temperature(℃)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point Temperature(℃)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)
count	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000
mean	704.602055	11.500000	12.882922	58.226256	1.724909	1436.825799	4.073813	0.569111	0.148687	0.075068
std	644.997468	6.922582	11.944825	20.362413	1.036300	608.298712	13.060369	0.868746	1.128193	0.436746
min	0.000000	0.000000	-17.800000	0.000000	0.000000	27.000000	-30.600000	0.000000	0.000000	0.000000
25%	191.000000	5.750000	3.500000	42.000000	0.900000	940.000000	-4.700000	0.000000	0.000000	0.000000
50%	504.500000	11.500000	13.700000	57.000000	1.500000	1698.000000	5.100000	0.010000	0.000000	0.000000
75%	1065.250000	17.250000	22.500000	74.000000	2.300000	2000.000000	14.800000	0.930000	0.000000	0.000000
max	3556.000000	23.000000	39.400000	98.000000	7.400000	2000.000000	27.200000	3.520000	35.000000	8.800000

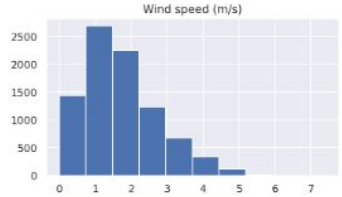
대략적인 통계와 데이터 확인

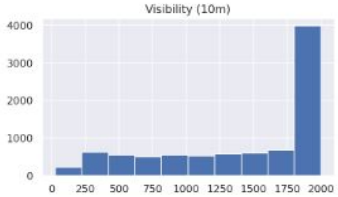


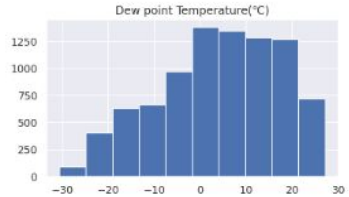


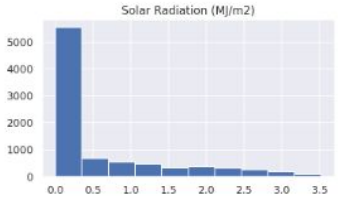


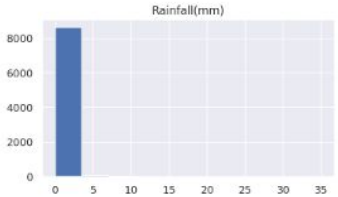


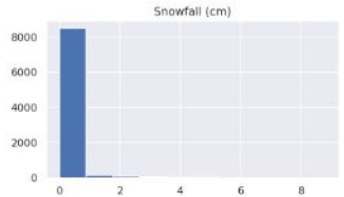






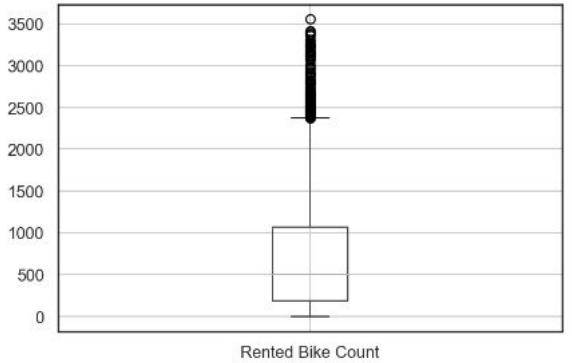






자전거 대여 횟수 상자그림

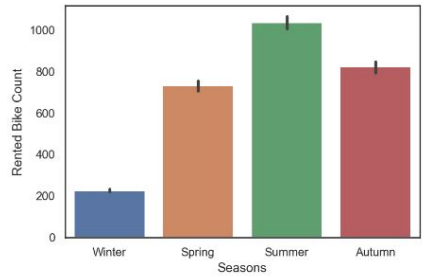
```
1  bike.boxplot(column=["Rented Bike Count"]);
```



Feature Drop (해당 피쳐 원핫인코딩 후 세분화해서 넣었던 모델 파일x)

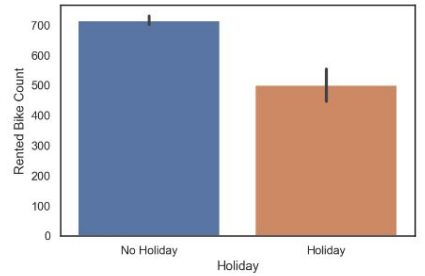
계절별 자전거 렌트 횟수

```
1 sns.barplot('Seasons', 'Rented Bike Count', data=bike);
```



휴일 자전거 렌트 횟수

```
1 sns.barplot('Holiday', 'Rented Bike Count', data=bike);
```



OLS Regression Results

Dep. Variable:	Rented Bike Count	**R-squared:**	0.472
Model:	OLS	**Adj. R-squared:**	0.472
Method:	Least Squares	**F-statistic:**	695.7
Date:	Thu, 25 Mar 2021	**Prob (F-statistic):**	0.00
Time:	04:33:50	**Log-Likelihood:**	-53014.
No. Observations:	7008	**AIC:**	1.060e+05
Df Residuals:	6998	**BIC:**	1.061e+05
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	374.3436	6.456	8.058	0.000	283.276	465.411
Hour	28.1547	0.871	32.338	0.000	26.448	29.861
Temperature(℃)	24.9196	0.601	41.434	0.000	23.741	26.099
Humidity(%)	-6.0454	0.370	-16.345	0.000	-6.770	-5.320
Wind speed (m/s)	1.8840	5.910	0.319	0.750	-9.701	13.469
Visibility (10m)	0.0227	0.011	2.004	0.045	0.000	0.045
Rainfall(mm)	-55.6524	4.905	-11.347	0.000	-65.267	-46.038
Snowfall (cm)	11.3190	13.300	0.851	0.395	-14.754	37.392
Seasons	70.6908	6.451	10.958	0.000	58.045	83.337
Holiday	-164.3242	25.376	-6.476	0.000	-214.068	-114.580

Omnibus:	874.761	**Durbin-Watson:**	2.012
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1869.468

Feature Drop 이유 :

Temperature, Rainfall, Snowfall 등과 같이 계절적 요소의 영향을 받는 다른 Feature들이 있어 Seasons는 feature에서 제거

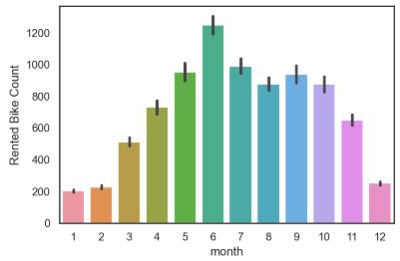
(범주형 데이터 변수화한 후 비교한 자료 누락)

Holiday : 휴일이 아닌경우에도 꾸준한 대여횟수, 기상요소에 집중하기 위해 해당 feature 제거

Feature Drop

월별 자전거 렌트 횟수

```
sns.barplot('month', 'Rented Bike Count', data=bike);
```



OLS Regression Results

Dep. Variable:	Rented Bike Count	R-squared (uncentered):	0.759
Model:	OLS	Adj. R-squared (uncentered):	0.759
Method:	Least Squares	F-statistic:	2003.
Date:	Tue, 23 Mar 2021	Prob (F-statistic):	0.00
Time:	09:01:55	Log-Likelihood:	-53014.
No. Observations:	7008	AIC:	1.060e+05
Df Residuals:	6997	BIC:	1.061e+05
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Hour	28.2070	0.872	32.358	0.000	26.498	29.916
Temperature(℃)	24.8005	0.610	40.682	0.000	23.605	25.995
Humidity(%)	-6.0262	0.370	-16.280	0.000	-6.752	-5.301
Wind speed (m/s)	1.4161	5.923	0.239	0.811	-10.195	13.027
Visibility (10m)	0.0232	0.011	2.043	0.041	0.001	0.045
Rainfall(mm)	-55.7445	4.905	-11.365	0.000	-65.360	-46.129
Snowfall (cm)	13.6234	13.439	1.014	0.311	-12.722	39.968
Seasons	75.7524	7.740	9.787	0.000	60.579	90.925
Holiday	-161.3491	25.494	-6.329	0.000	-211.325	-111.373
year	0.1855	0.023	8.056	0.000	0.140	0.231
month	-2.4326	2.050	-1.187	0.235	-6.452	1.586

Omnibus:	873.023	Durbin-Watson:	2.012
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1859.871
Skew:	0.764	Prob(JB):	0.00
Kurtosis:	5.009	Cond. No.	1.14e+04

①

일/월/연도 정보를 가지고 있는 'Date' 를 'year', 'month'로 나눠 더미변수화하여 비교.

'Season' 또한 사계로 나눠 R-squared 및 상관계수 등을 관찰하였으나

데이터 특성상 이용시간대에 집중하는 것이 좋겠다는 판단하에 'Hour'를 나누기로 함

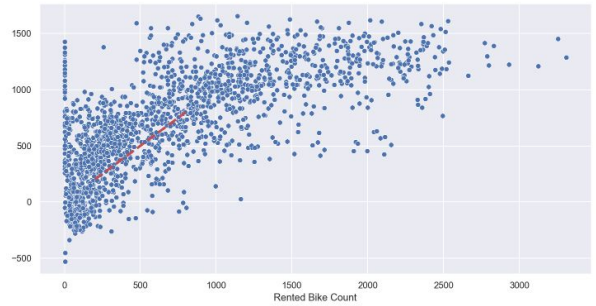
②

Feature Drop 이유 :
범주형 데이터를 더미변수화하여 각 피쳐별로 비교하였으나 'Hour'에 초점을 맞추는 것이 중요하다고 판단하여 방향을 수정함
데이터 전처리 과정에서 시간내 완수하지 못함
(Issue 차후계획 부분에서 설명)

각각의 피쳐를 넣고 빼는 과정에서 높은 수치의 R-squared가 나왔으며, 일별 데이터에 시간까지 포함해 가중치가 붙어 높게 측정 된 것으로 추측

시간과 관련된 feature를 빼게 된 이유
(시간만 그룹별로 추출하고 나면 drop하려고 남겨둠)

```
sns.scatterplot(y_test,predictions);
plt.plot([200,800], [200, 800], 'r', ls='dashed', lw=3);
```




Correlation of Bike Sharing with Features
Rented Bike Count
Temperature(℃)
Humidity(%)
Wind speed (m/s)
Visibility (10m)
Dew point Temperature(℃)
Solar Radiation (MJ/m2)
Rainfall(mm)
Snowfall (cm)
1 0.54 -0.2 0.12 0.2 0.38 0.26 -0.12 -0.14
0.54 1 0.16 -0.036 0.035 0.91 0.35 0.05 -0.22
-0.2 0.16 1 -0.34 -0.54 0.54 -0.46 0.24 0.11
0.12 -0.036 -0.34 1 0.17 -0.18 0.33 -0.02 -0.0036
0.2 0.035 -0.54 0.17 1 -0.18 0.15 -0.17 -0.12
0.38 0.91 0.54 -0.18 -0.18 1 0.094 0.13 -0.15
0.26 0.35 -0.46 0.33 0.15 0.094 1 -0.074 -0.072
-0.12 0.05 0.24 -0.02 -0.17 0.13 -0.074 1 0.0085
-0.14 -0.22 0.11 -0.0036 -0.12 -0.15 -0.072 0.0085 1
Rented Bike Count
Temperature(℃)
Humidity(%)
Wind speed (m/s)
Visibility (10m)
Dew point Temperature(℃)
Solar Radiation (MJ/m2)
Rainfall(mm)
Snowfall (cm)
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

예측하기

1-1. 피처설정

```
bike.drop(['Date','Hour', 'Functioning Day','Seasons', 'Holiday'], axis=1, inplace=True)
#'Holiday' 원핫 인코딩
# bike["Seasons"] Drop 이유 : 계절보다 일별, 시간대 데이터의 영향을 우선/다중공선성 우려
bike
```

	Rented Bike Count	Temperature(℃)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point Temperature(℃)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)
0	254	-5.2	37	2.2	2000	-17.6	0.0	0.0	0.0
1	204	-5.5	38	0.8	2000	-17.6	0.0	0.0	0.0
2	173	-6.0	39	1.0	2000	-17.7	0.0	0.0	0.0
3	107	-6.2	40	0.9	2000	-17.6	0.0	0.0	0.0
4	78	-6.0	36	2.3	2000	-18.6	0.0	0.0	0.0

회귀분석을 위한 학습, 테스트 데이터셋 분리

```
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt

x = bike.drop('Rented Bike Count', axis=1)
y = bike['Rented Bike Count']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2,
                                                    random_state=13)
np.unique(y_train, return_counts=True)
```

```
(array([   0,    2,    3, ..., 3404, 3418, 3556], dtype=int64),
 array([247,   1,   1, ...,   1,   1,   1], dtype=int64))
```

Feature : 기상적 요인에 집중

각 Feature 마다 독립성을 유지하기 힘든 요소들이지만..

Scaling 전

OLS Regression Results

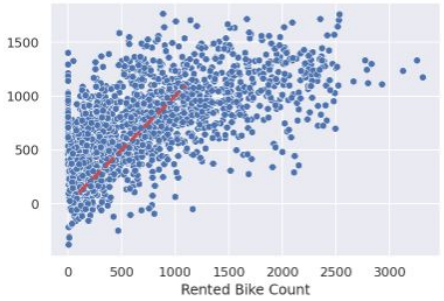
Dep. Variable:	Rented Bike Count	R-squared (uncentered):	0.721
Model:	OLS	Adj. R-squared (uncentered):	0.721
Method:	Least Squares	F-statistic:	2264.
Date:	Thu, 25 Mar 2021	Prob (F-statistic):	0.00
Time:	00:40:06	Log-Likelihood:	-53523.
No. Observations:	7008	AIC:	1.071e+05
Df Residuals:	7000	BIC:	1.071e+05
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Temperature(℃)	66.7930	1.836	36.377	0.000	63.194	70.392
Humidity(%)	-1.2779	0.218	-5.858	0.000	-1.706	-0.850
Wind speed (m/s)	54.7244	6.316	8.664	0.000	42.343	67.106
Visibility (10m)	0.0214	0.012	1.853	0.064	-0.001	0.044
Dew point Temperature(℃)	-33.5347	1.701	-19.710	0.000	-36.870	-30.199
Solar Radiation (MJ/m2)	-124.2501	9.726	-12.774	0.000	-143.317	-105.183
Rainfall(mm)	-51.4735	5.252	-9.801	0.000	-61.769	-41.178
Snowfall (cm)	32.3509	14.350	2.254	0.024	4.221	60.480

Omnibus:	820.855	Durbin-Watson:	1.984
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1402.065
Skew:	0.802	Prob(JB):	3.51e-305
Kurtosis:	4.494	Cond. No.	3.75e+03

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 3.75e+03. This might indicate that there are strong multicollinearity or other numerical problems.

```
sns.scatterplot(y_test, predictions);
plt.plot([100, 1100], [100,1100], 'r', ls='dashed', lw=3);
```



회귀 분석 계수 학습& 학습된 계수 출력

```
reg =linear_model.LinearRegression()
model =reg.fit(x_train, y_train)

# 학습된 계수출력
print(reg.coef_)
```

```
[ 3.21055896e+01 -1.18922161e+01  5.15831265e+01 -1.21011059e-02
  3.87775676e+00 -1.19571603e+02 -4.56901454e+01  4.09148008e+01]
```

Linear Regression

```
from sklearn.linear_model import LinearRegression

reg = LinearRegression()
reg.fit(x_train, y_train)
```

```
LinearRegression()
```

```
import numpy as np
from sklearn.metrics import mean_squared_error

pred_tr = reg.predict(x_train)
pred_test = reg.predict(x_test)
rmse_tr = (np.sqrt(mean_squared_error(y_train, pred_tr)))
rmse_test = (np.sqrt(mean_squared_error(y_test, pred_test)))

print('RMSE of Train Data: ', rmse_tr)
print('RMSE of Test Data: ', rmse_test)
```

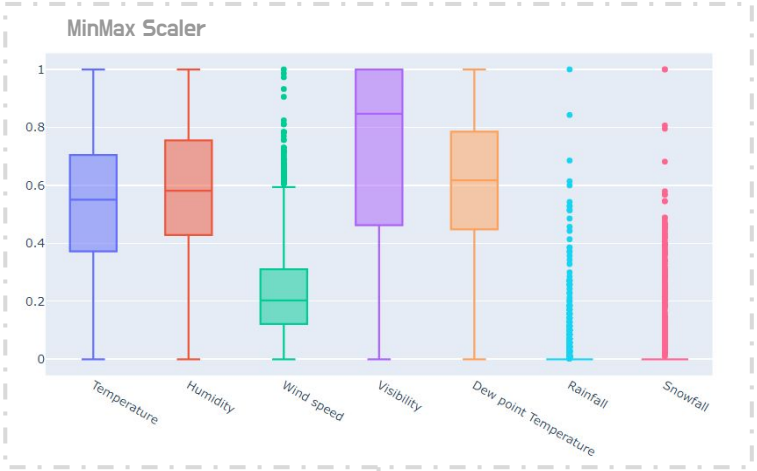
```
RMSE of Train Data:  499.60007850123118
RMSE of Test Data:  504.12044149319263
```

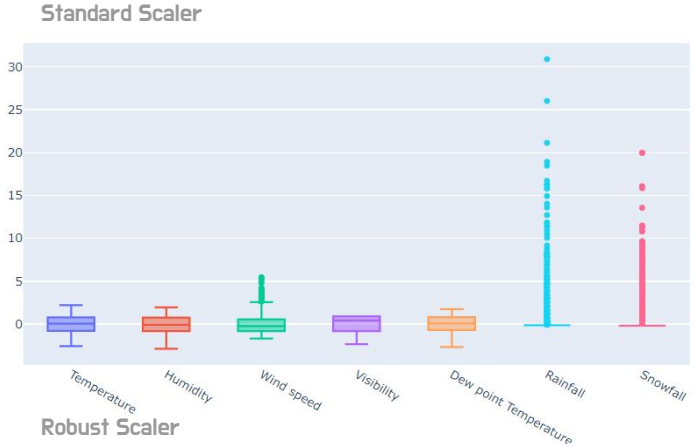
예측 모델 평가하기

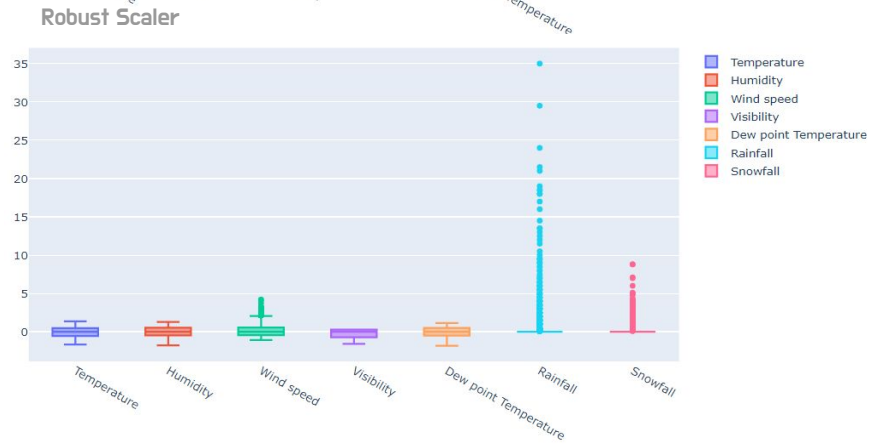
```
#reg = LinearRegression()
#reg.fit(x_train, y_train)

print(reg.score(x_train, y_train))
print(reg.score(x_test, y_test))
```

```
0.3954586543529801
0.40631299385389665
```

MinMax, Standard, Robust Scaler 비교







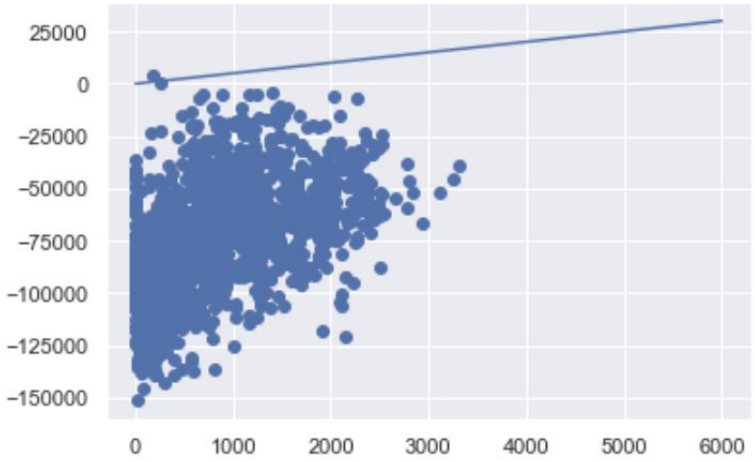

```
1 X_mss_pd.describe()
```

	Temperature(℃)	Humidity(%)	Wind_speed_(m/s)	Visibility_(10m)	Dew_point_Temperature(℃)	Solar_Radiation_(MJ/m2)	Rainfall(mm)	Snowfall_(cm)
count	7008.000000	7008.000000	7008.000000	7008.000000	7008.000000	7008.000000	7008.000000	7008.000000
mean	0.535811	0.592876	0.233260	0.715183	0.598653	0.163285	0.004230	0.008302
std	0.207689	0.207670	0.140851	0.308028	0.224735	0.247875	0.033534	0.049440
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.372378	0.428571	0.121622	0.465788	0.446367	0.000000	0.000000	0.000000
50%	0.548951	0.581633	0.202703	0.846427	0.614187	0.002841	0.000000	0.000000
75%	0.701049	0.755102	0.310811	1.000000	0.783737	0.269886	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

LinearRegresstion 학습 후 RMSE 및 성능확인 (MinMax Scaler 적용 후)

```
from sklearn.metrics import mean_squared_error
pred_tr = reg.predict(X_mms)
pred_test = reg.predict(X_test)

rmse_tr = (np.sqrt(mean_squared_error(y_train,pred_tr)))
rmse_test =(np.sqrt(mean_squared_error(y_test,pred_test)))

print('RMSE of Train Data : ', rmse_tr)
print('RMSE of Test Data : ', rmse_test)
```

```
RMSE of Train Data :  499.6000785012318
RMSE of Test Data :  82558.86585736295
```



교차검증

```
1 from sklearn.model_selection import cross_val_score
2
3 scores = cross_val_score(reg,X_mms, y_train, scoring = 'neg_mean_squared_error', cv=10)
4
5 X_mms_rmse_scores = np.sqrt(-scores)
```

```
1 def display_scores(scores):
2     print("점수", scores)
3     print("평균", scores.mean())
4     print("표준 편차", scores.std())
5
6 display_scores(X_mms_rmse_scores)
```

```
점수 [485.36958491 454.16506775 509.70103542 521.46491431 521.88801338
 489.1854826  511.46770102 508.48828526 494.20104967 504.43732279]
평균 500.03684571141764
표준 편차 19.29415674861269
```

Decision Tree Regressor

```
from sklearn.tree import DecisionTreeRegressor

tree_reg = DecisionTreeRegressor(random_state=13)
tree_reg.fit(X_mms, y_train)
```

```
DecisionTreeRegressor(random_state=13)
```

결과

```
predictions = tree_reg.predict(X_mms)
tree_mse = mean_squared_error(y_train, predictions)
tree_rmse = np.sqrt(tree_mse)
tree_rmse
```

```
15.980666651332308
```

교차 검증

```
scores = cross_val_score(tree_reg,X_mms, y_train, scoring = 'neg_mean_squared_error', cv=10)
tree_X_mms_rmse_scores = np.sqrt(-scores)
```

결과

```
display_scores(tree_X_mms_rmse_scores)
```

```
점수 [544.04053669 533.37709773 548.86348144 582.40936429 552.78695564
 603.84786844 578.28015956 553.89883756 529.56278732 551.64394184]
평균 557.8711030524211
표준 편차 22.105797070839213
```

RandomForest Regressor

```
from sklearn.ensemble import RandomForestRegressor

forest_reg = RandomForestRegressor(n_estimators=100, random_state=13)
forest_reg.fit(X_mms, y_train)
```

```
RandomForestRegressor(random_state=13)
```

```
predictions = forest_reg.predict(X_mms)
forest_mse = mean_squared_error(y_train, predictions)
forest_rmse = np.sqrt(forest_mse)
forest_rmse
```

```
152.18416850297916
```

```
forest_scores = cross_val_score(forest_reg, X_mms, y_train,
                                scoring="neg_mean_squared_error", cv=10)
forest_rmse_scores = np.sqrt(-forest_scores)
display_scores(forest_rmse_scores)
```

```
점수 [401.28531368 387.25897671 413.33910753 413.11061638 410.54642749
 423.10204938 402.63623709 390.56128251 407.44229973 399.07390417]
평균 404.8356214687136
표준 편차 10.378775643742424
```

ISSUE (추후계획)

- **Label or Feature 로그 스케일링**

Rented Bike Count	Hour	Temperature(캘)	Humidity(%)
254	0	-5.2	37
204	1	-5.5	38
173	2	-6.0	39
107	3	-6.2	40
78	4	-6.0	36
...	...	...	...

Temperature(℃)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point Temperature(℃)
-5.2	37	2.2	2000	-17.6
-5.5	38	0.8	2000	-17.6
-6.0	39	1.0	2000	-17.7
-6.2	40	0.9	2000	-17.6
-6.0	36	2.3	2000	-18.6
...	...	...	...	...

- **다중 공선성 확인**
  - ridge : 다중 공선성 방지 모델
  - lasso : 작은 값의 파라미터를 0으로

만들어 변수를 모델에서 삭제하고 단순하게 만듦

다중 공선성 확인: 회귀 분석 예측 성능을 높이기 위한 방법

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

#피처마다의 VIF 계수를 출력
vif = pd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(x.values, i)  for i in range(x.shape[1])]
vif["features"] = x.columns
vif.round(1)
```

	VIF Factor	features
0	29.1	Temperature(℃)
1	5.1	Humidity(%)
2	4.5	Wind speed (m/s)
3	9.1	Visibility (10m)
4	15.2	Dew point Temperature(℃)
5	2.8	Solar Radiation (MJ/m2)
6	1.1	Rainfall(mm)
7	1.1	Snowfall (cm)

- **robust 스케일러 적용 후 학습 (이상치)**

ISSUE (추후계획) : 시간데이터 (구간별)

주제 선정 및 프로젝트의 의의 :
- 특정 피쳐가 공유자전거의 수요에 끼치는 영향 파악
- 약 2년간의 공유 자전거 사용자의 스타일 파악
- 어떤 피쳐가 가장 많은 영향을 끼치는지 살펴보고 공유자전거 수요에 알맞게 배차할 수 있는 비즈니스 모델 제시

-자전거를 언제 가장 많이 빌릴까
-자전거를 언제 탈까 (출퇴근 사용)

각각의 변수가 어떻게 발생하고 영향을 끼치는지
※ 시간대가 매우 중요
어느 시간대에 수요가 많아서 대여대수를 늘려야하는지 예측하는 모델

0시~23시 시간을
특정 시간에 맞게
분할하여 Feature 로 생성

```python
def hour(x):
    if x >=0 and x <=5:
        x=1
    return x
bike["time_1"] = bike['Hour'].apply(hour)
bike["time_1"][bike["time_1"] != 1] = 0
bike["time_1"]
```

```python
bike['time_1'].head(50)
```

```
0     1
1     1
2     1
3     1
4     1
5     1
6     0
7     0
8     0
9     0
10    0
11    0
12    0
13    0
14    0
15    0
16    0
17    0
18    0
19    0
20    0
21    0
22    0
23    0
24    1
25    1
26    1
27    1
28    1
29    1
```

	Rented Bike Count	Hour	Temperature(℃)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point Temperature(℃)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	TIME_1	TIME_2	TIME_3	TIME_4
											0시~5시	6~11시	12시~17시	18시~23시
0	254	0	-5.2	37	2.2	2000	-17.6	0.0	0.0	0.0				
1	204	1	-5.5	38	0.8	2000	-17.6	0.0	0.0	0.0				
2	173	2	-6.0	39	1.0	2000	-17.7	0.0	0.0	0.0				
3	107	3	-6.2	40	0.9	2000	-17.6	0.0	0.0	0.0				
4	78	4	-6.0	36	2.3	2000	-18.6	0.0	0.0	0.0				
...	...	...	...	...	...	...	...	...	...	...				
8755	1003	19	4.2	34	2.6	1894	-10.3	0.0	0.0	0.0				
8756	764	20	3.4	37	2.3	2000	-9.9	0.0	0.0	0.0				
8757	694	21	2.6	39	0.3	1968	-9.9	0.0	0.0	0.0				
8758	712	22	2.1	41	1.0	1859	-9.8	0.0	0.0	0.0				
8759	584	23	1.9	43	1.3	1909	-9.3	0.0	0.0	0.0				

'Hour' Feature Drop

마찬가지로 모든 컬럼의
값을 일 단위,
4등분한 '평균값'으로
묶어줘야함 (데이터 전처리)

당연한 이야기지만
데이터를
어떤 관점에서
접근하느냐에 따라
방법도
달라져야겠죠...

Regression-repo-4

감사합니다!

Thank You :)

장지혜 정다은

Flow (발표자료 및 데이터 분석시 참고)

1. 데이터 살펴보기 -전처리 (데이터 파악: 피처설정/피처정규화)
2. 예측(원핫인코딩: 단위 맞추기),
   테스트 데이터셋 분리, 회귀 분석계수학습(회귀 모델 학습)
3. 평가 : 어떤 피처가 가장 영향력이 강한 피처일까
   피처들의 상관관계 분석(히트맵)/다중공선성 확인
4. 시각화 (분석결과 시각화하기)

ISSUE : 날짜, 시간 정보가 들어있지만 각각 나눠져 있고 TYPE이 다름
카테고리화 해서 일정 시간대 분포를 보려고 하는데 잘 안됨 ㅠㅠ 왓
어제 강사님이 알려주신 Facebook Prophet는 활용이 ::: 안될 듯(시계열 데이터-트렌드)