

Dacon Basic 참가후기

따릉이 대여량 예측하기

최종수정일 : 2022-01-21

Link : <https://dacon.io/competitions/official/235837/codeshare/3724?page=1&dtype=recent>

INDEX

- 1 대회 소개 및 문제 정의
- 2 데이터 소개 및 EDA
- 3 데이터 전처리 및 변수 선택
- 4 모델 학습 과정 및 개선
- 5 마무리



대회 소개 및 문제 정의

➤ 대회 소개 (<https://dacon.io/competitions/official/235837/overview/description>)



따릉이 대여량 예측 경진대회
데이콘 베이직 Basic | 정형 | NMAE

🏆 상금 : 참가시 최소 50 XP, 특별상 데이콘 후드

📅 2021.11.01 ~ 2021.11.12 17:59 [+ Google Calendar](#)

👤 603명 📅 마감

참여중

➤ 문제 소개

훈련 데이터
2018, 2019, 2020
4~6월 일간 따릉이 대여량

➡
예측

평가 데이터
2021년
4~6월 일간 따릉이 대여량

데이터 소개



Train Data : 2018.4~6, 2019.4~6, 2020.4~6 기상 상황과 일간 대여량 데이터

| | date_time | wind_direction | sky_condition | precipitation_form | wind_speed | humidity | low_temp | high_temp | Precipitation_Probability | number_of_rentals |
|---|------------|----------------|---------------|--------------------|------------|----------|----------|-----------|---------------------------|-------------------|
| 0 | 2018-04-01 | 207.500 | 4.000 | 0.000 | 3.050 | 75.000 | 12.600 | 21.000 | 30.000 | 22994 |
| 1 | 2018-04-02 | 208.317 | 2.950 | 0.000 | 3.278 | 69.833 | 12.812 | 19.000 | 19.500 | 28139 |
| 2 | 2018-04-03 | 213.516 | 2.911 | 0.000 | 2.690 | 74.879 | 10.312 | 15.316 | 19.113 | 26817 |
| 3 | 2018-04-04 | 143.836 | 3.692 | 0.425 | 3.138 | 71.849 | 8.312 | 12.368 | 43.493 | 26034 |
| 4 | 2018-04-05 | 95.905 | 4.000 | 0.723 | 3.186 | 73.784 | 5.875 | 10.421 | 63.378 | 2833 |



Test Data : 2021.4~6 기상 상황에 대한 데이터

| | date_time | wind_direction | sky_condition | precipitation_form | wind_speed | humidity | low_temp | high_temp | Precipitation_Probability |
|---|------------|----------------|---------------|--------------------|------------|----------|----------|-----------|---------------------------|
| 0 | 2021-04-01 | 108.833 | 3.000 | 0.000 | 2.900 | 28.333 | 11.800 | 20.667 | 18.333 |
| 1 | 2021-04-02 | 116.717 | 3.850 | 0.000 | 2.662 | 46.417 | 12.000 | 19.000 | 28.500 |
| 2 | 2021-04-03 | 82.669 | 4.000 | 0.565 | 2.165 | 77.258 | 8.875 | 16.368 | 52.847 |
| 3 | 2021-04-04 | 44.123 | 3.466 | 0.466 | 3.747 | 63.288 | 6.250 | 17.368 | 37.671 |
| 4 | 2021-04-05 | 147.791 | 1.500 | 0.000 | 1.560 | 48.176 | 7.188 | 18.684 | 4.459 |

변수 의미 소개



Train Data

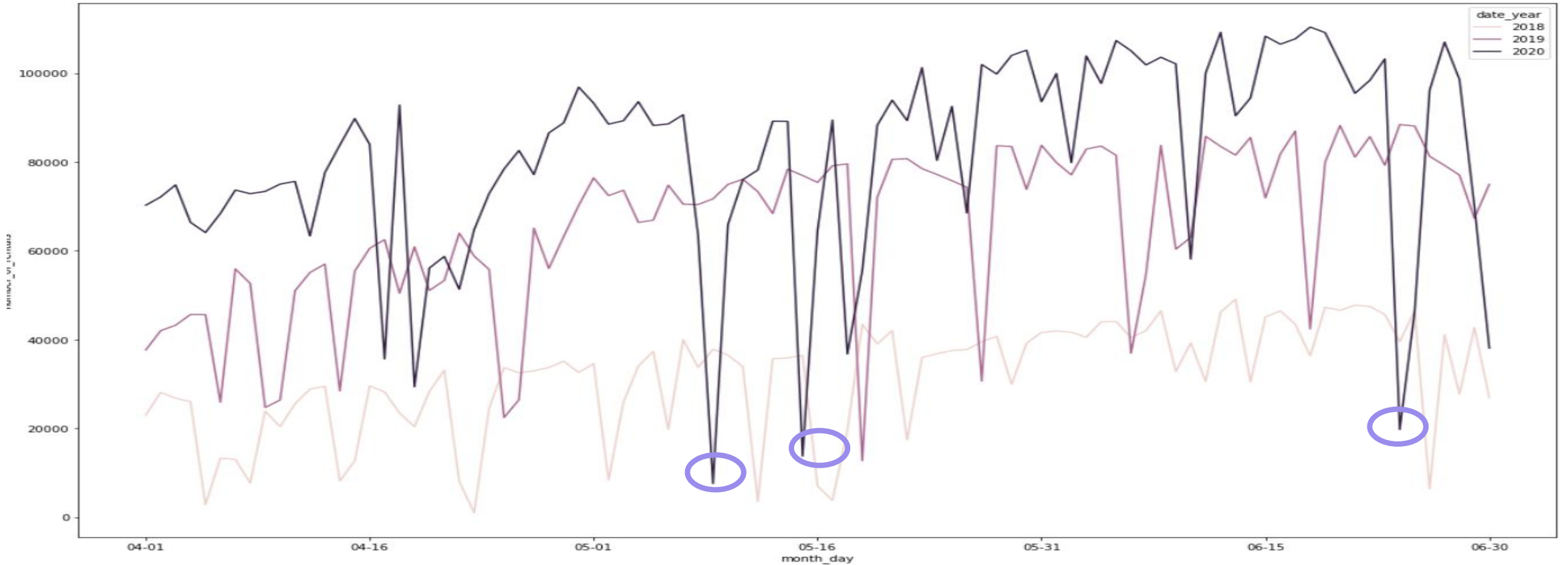
| | date_time | wind_direction | sky_condition | precipitation_form | wind_speed | humidity | low_temp | high_temp | Precipitation_Probability | number_of_rentals |
|---|------------|----------------|---------------|--------------------|------------|----------|----------|-----------|---------------------------|-------------------|
| 0 | 2018-04-01 | 207.500 | 4.000 | 0.000 | 3.050 | 75.000 | 12.600 | 21.000 | 30.000 | 22994 |
| 1 | 2018-04-02 | 208.317 | 2.950 | 0.000 | 3.278 | 69.833 | 12.812 | 19.000 | 19.500 | 28139 |
| 2 | 2018-04-03 | 213.516 | 2.911 | 0.000 | 2.690 | 74.879 | 10.312 | 15.316 | 19.113 | 26817 |
| 3 | 2018-04-04 | 143.836 | 3.692 | 0.425 | 3.138 | 71.849 | 8.312 | 12.368 | 43.493 | 26034 |
| 4 | 2018-04-05 | 95.905 | 4.000 | 0.723 | 3.186 | 73.784 | 5.875 | 10.421 | 63.378 | 2833 |

| 변수명 | 의미 | 변수명 | 의미 |
|--------------------|----------------------------------|---------------------------|----------|
| date_time | 날짜 | humidity | 습도(%) |
| wind_direction | 풍향 | low_temp | 최저온도(°C) |
| sky_condition | 평균 하늘 상태 - 맑으면 1, 구름3, 흐리면 4 | high_temp | 최고온도(°C) |
| precipitation_form | 일 평균 강우상태 - 맑으면 0, 비 1, 소나기 4 | precipitation_probability | 강우확률(%) |
| wind_speed | 일 평균 풍속 | number_of_rentals | 대여량 |

Train Data EDA (1)



연도별 따릉이 대여량의 변화 확인



- 대체로 시간이 지날수록 대여량 증가
- 그러나 일부 대여량이 매우 낮은 값은?

Train Data EDA (2)



2020년에서 대여량이 매우 낮은 시점 확인

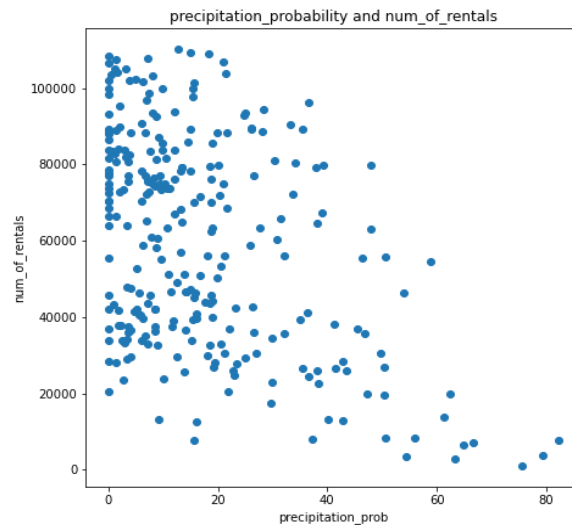
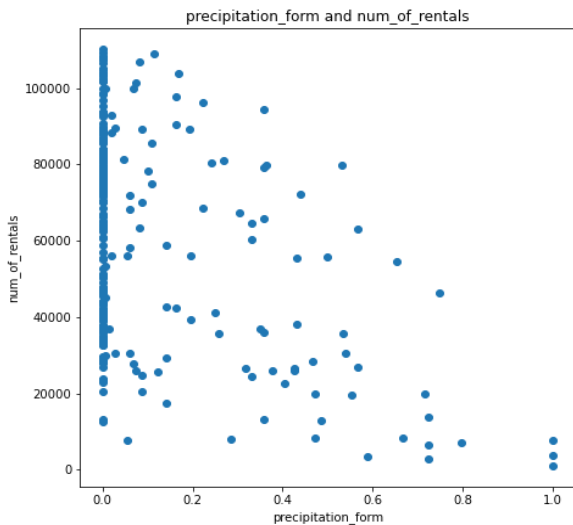
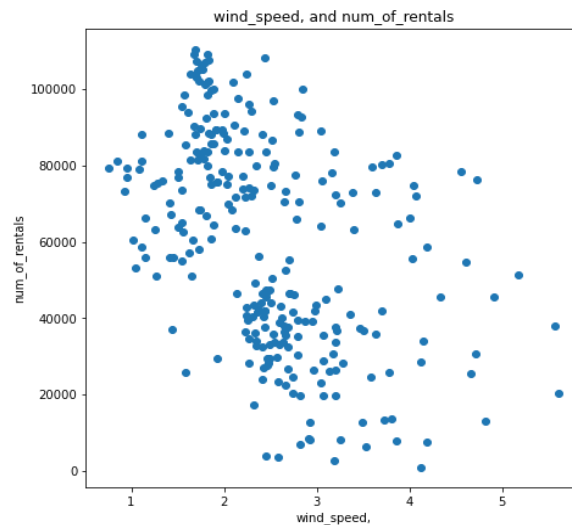
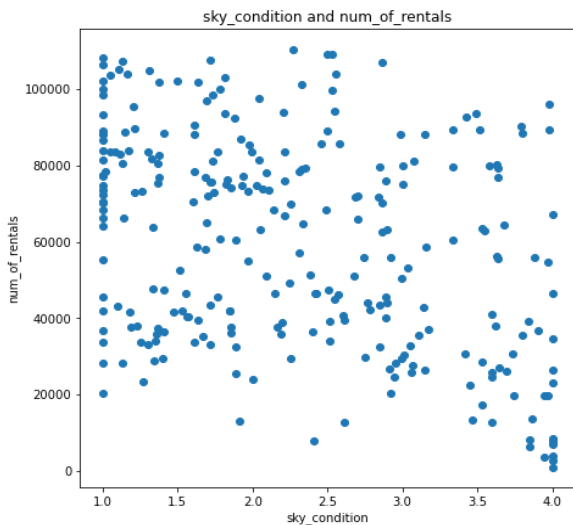
| | date_time | wind_direction | sky_condition | precipitation_form | wind_speed | humidity | low_temp | high_temp | Precipitation_Probability | number_of_rentals |
|-----|------------|----------------|---------------|--------------------|------------|----------|----------|-----------|---------------------------|-------------------|
| 220 | 2020-05-09 | 144.142 | 4.000 | 1.000 | 4.192 | 80.034 | 13.938 | 21.158 | 82.162 | 7600 |
| 226 | 2020-05-15 | 140.966 | 3.865 | 0.723 | 3.804 | 69.122 | 15.875 | 22.263 | 61.216 | 13782 |
| 266 | 2020-06-24 | 124.797 | 3.973 | 0.716 | 3.203 | 76.182 | 21.375 | 26.421 | 62.500 | 19756 |
| 200 | 2020-04-19 | 108.432 | 2.993 | 0.142 | 1.930 | 57.939 | 8.625 | 15.526 | 25.068 | 29375 |
| 198 | 2020-04-17 | 153.196 | 3.797 | 0.534 | 3.066 | 60.068 | 9.625 | 17.632 | 46.892 | 35656 |
| 229 | 2020-05-18 | 154.453 | 3.905 | 0.351 | 3.511 | 75.473 | 12.688 | 20.211 | 45.554 | 36761 |
| 272 | 2020-06-30 | 120.797 | 3.622 | 0.432 | 5.574 | 77.061 | 19.125 | 26.053 | 41.284 | 38086 |
| 267 | 2020-06-25 | 143.777 | 4.000 | 0.750 | 2.447 | 84.527 | 20.688 | 27.316 | 54.054 | 46415 |
| 203 | 2020-04-22 | 294.034 | 2.385 | 0.000 | 5.176 | 42.500 | 3.938 | 12.000 | 13.851 | 51337 |
| 230 | 2020-05-19 | 188.635 | 3.635 | 0.432 | 4.036 | 76.622 | 11.125 | 18.789 | 46.284 | 55556 |

대여량이 매우 낮아졌을 때에는 비가 많이 온 날씨가거나 기온이 예년에 비해 매우 낮다.

Train Data EDA (3)



날씨와 자전거 대여량의 관계 파악을 위한 시각화



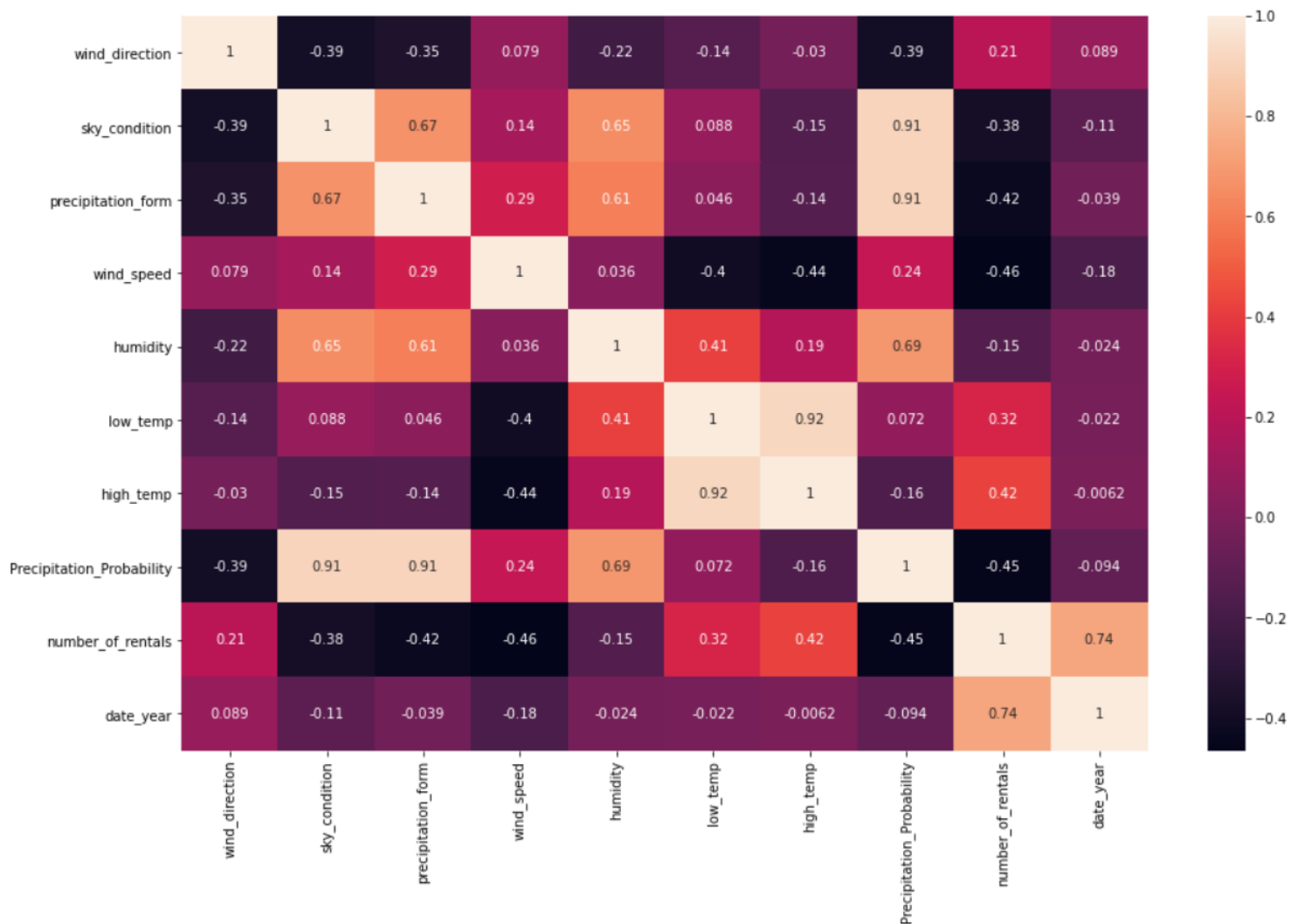
- 하늘 상태, 풍속, 강우 형태, 강우 확률
4가지 기후 변수와 자전거 대여량의 시각화 진행
- 전체적으로 풍속 및 강우의 값이 클 수록, 즉 기후 상태가 좋지 않다고 판단될 수록 자전거 이용량이 낮아지는 경향성을 보이고 있다.
- 또한 대여량이 매우 낮은 값들 (2만 미만)에서는 하늘 상태도 맑지는 않다는 것을 알 수 있다.

즉, 날씨가 나빠질수록 대여량은 급감한다고 추론할 수 있고 이에 대한 경우를 별도로 관리해줘야 함을 알 수 있다.

Train Data EDA (4)



변수들간의 상관관계 확인



- 풍향, 습도와 대여량의 관계는 적다.
 - 강우량 등에는 어느정도 영향을 받는다.
 - 온도에 대해서도 영향을 고려해야한다.
-
- 습도는 다른 방식으로 사용할 예정이다.
 - 풍향은 변수 사용에서 제외한다.
 - 강우확률보다는 강우형태로 실제로 비가 어느 정도 내렸는지를 표현하는게 좋아 강우확률 또한 사용변수에서 제외한다.

추가 변수 생성 (1)



날짜에 대한 처리 - 주말 변수의 추가

```
def is_weekend(t):  
    if t.weekday() >= 5:  
        return 1  
    else:  
        return 0
```

```
train_df['is_weekend'] = train_df.date_time.apply(lambda t : is_weekend(t))  
test_df['is_weekend'] = test_df.date_time.apply(lambda t : is_weekend(t))
```

```
train_df['number_of_rentals'].groupby(train_df.is_weekend).agg('median')
```

```
is_weekend  
0    60925  
1    64330  
Name: number_of_rentals, dtype: int64
```

중간값 기준
평일과, 주말에는 대여량의 차이가 보인다.

해당 변수를 사용한다.

추가 변수 생성 (2)



기후에 대한 고려 - 불쾌지수

단순히 덥다가 아니라, 짜증 등의 기후적인 요인 발생으로 탑승량 감소가 있을 것이기에 불쾌지수 반영

```
# 불쾌지수 공식의 활용
def get_discomfort(humid, min_t, max_t):
    # 전체적인 탑승의 경향성을 반영하기 위해 출퇴근 시간의 사용량이 많음에도 불구하고, 평균온도로 고려합니다.
    temp = (min_t + max_t) / 2
    humid = humid / 100

    discomfort = 1.8 * temp - 0.558 * (1 - humid) * (1.8 * temp - 26) + 32
    return discomfort
```



기후에 대한 고려 - 추위와 일교차에 대한 고려

```
# 추운 정도 반영
train_df['cold_measure'] = train_df['low_temp'] / train_df['wind_speed']
test_df['cold_measure'] = test_df['low_temp'] / test_df['wind_speed']

# 일교차 반영
# 일교차를 반영하면, 체감온도를 반영할 수 있다.
train_df['temp_diff'] = train_df['high_temp'] - train_df['low_temp']
test_df['temp_diff'] = test_df['high_temp'] - test_df['low_temp']
```

추운 정도에 대한 반영을 위해
최저온도를 풍속으로 나눈다.
온도가 낮고, 바람이 세게 불 수록
더 춥게 느껴질것이다.

즉 추운 정도는 값이 적을수록 더 강도가 세다.

추가 변수 생성 (3)



자전거를 타기 어려운 정도에 대한 고려

풍속이 세고, 날씨가 좋지 않을 경우 자전거를 타기 어려울 것이다.

```
train_df['hardship'] = train_df['sky_condition'] * train_df['wind_speed']  
test_df['hardship'] = test_df['sky_condition'] * test_df['wind_speed']
```



최종 변수 선택 결과

| | sky_condition | precipitation_form | wind_speed | low_temp | high_temp | number_of_rentals | date_year | is_weekend | discomfort | hardship | cold_measure | temp_diff |
|---|---------------|--------------------|------------|----------|-----------|-------------------|-----------|------------|------------|-----------|--------------|-----------|
| 0 | 4.000 | 0.000 | 3.050 | 12.600 | 21.000 | 22994 | 2018 | 1 | 61 | 12.200000 | 4.131148 | 8.400 |
| 1 | 2.950 | 0.000 | 3.278 | 12.812 | 19.000 | 28139 | 2018 | 0 | 60 | 9.670100 | 3.908481 | 6.188 |
| 2 | 2.911 | 0.000 | 2.690 | 10.312 | 15.316 | 26817 | 2018 | 0 | 55 | 7.830590 | 3.833457 | 5.004 |
| 3 | 3.692 | 0.425 | 3.138 | 8.312 | 12.368 | 26034 | 2018 | 0 | 51 | 11.585496 | 2.648821 | 4.056 |
| 4 | 4.000 | 0.723 | 3.186 | 5.875 | 10.421 | 2833 | 2018 | 0 | 48 | 12.744000 | 1.844005 | 4.546 |

모델학습 과정 - 평가지표 및 모델선택



평가지표 : NMAE

$$NMAE = \frac{1}{n} \sum_i^m \frac{|true_i - predict_i|}{true_i}$$

```
# 평가지표, 이 대회에서는 NMAE를 사용하게 됩니다.  
def get_nmae(pred, y):  
    nmae = np.mean(abs(pred-y)/y)  
    return nmae
```



학습 계획

선형회귀
Baseline

XGBoost

단 시간의 흐름에 따른 변화와 기후에 따른 변수 등 다양한 상황이 존재하여 모두 반영하기 위해
Train set을 별도로 나눠 Valid하는 절차는 거치지 않고, 바로 평가 데이터를 통해 정확도를 확인하였다.

모델학습 과정 - 선형회귀의 시도



선형회귀

stat_models는 상수항 추가를 꼭 이렇게 처리해줘야합니다.

```
X0 = sm.add_constant(X)
model = sm.OLS(y, X0)
result = model.fit()
print(result.summary())
```

OLS Regression Results

| | | | |
|-------------------|-------------------|---------------------|-----------|
| Dep. Variable: | number_of_rentals | R-squared: | 0.854 |
| Model: | OLS | Adj. R-squared: | 0.848 |
| Method: | Least Squares | F-statistic: | 153.2 |
| Date: | Sun, 14 Nov 2021 | Prob (F-statistic): | 2.48e-103 |
| Time: | 13:17:06 | Log-Likelihood: | -2916.4 |
| No. Observations: | 273 | AIC: | 5855. |
| Df Residuals: | 262 | BIC: | 5895. |
| Df Model: | 10 | | |
| Covariance Type: | nonrobust | | |

선형회귀로 예측시, 기후적인 부분들의 변수들 (sky_condition, wind_speed)나 Discomfort 등의 변수들이 유의미하지 않다고 나온다.

이는 기후적인 요인이 좋지 않은 날의 예측에는 선형 모델이 매우 나뻐를 알 수 있다. (훈련 데이터 NMAE 0.308)

| | coef | std err | t | P> t | [0.025 | 0.975] |
|--------------------|------------|----------|---------|-------|-----------|-----------|
| const | -4.834e+07 | 1.73e+06 | -27.875 | 0.000 | -5.18e+07 | -4.49e+07 |
| sky_condition | 2999.9943 | 2366.580 | 1.268 | 0.206 | -1659.943 | 7659.932 |
| precipitation_form | -3.568e+04 | 4919.535 | -7.253 | 0.000 | -4.54e+04 | -2.6e+04 |
| wind_speed | 2901.4154 | 2481.878 | 1.169 | 0.243 | -1985.551 | 7788.382 |
| low_temp | 870.1221 | 748.064 | 1.163 | 0.246 | -602.861 | 2343.105 |
| high_temp | 1464.5967 | 685.278 | 2.137 | 0.034 | 115.242 | 2813.951 |
| date_year | 2.396e+04 | 862.968 | 27.768 | 0.000 | 2.23e+04 | 2.57e+04 |
| is_weekend | -1583.7664 | 1474.539 | -1.074 | 0.284 | -4487.221 | 1319.689 |
| discomfort | -586.3170 | 985.433 | -0.595 | 0.552 | -2526.694 | 1354.060 |
| hardship | -1819.3935 | 886.977 | -2.051 | 0.041 | -3565.905 | -72.882 |
| cold_measure | 719.9213 | 363.877 | 1.978 | 0.049 | 3.426 | 1436.416 |
| temp_diff | 594.4745 | 277.390 | 2.143 | 0.033 | 48.277 | 1140.672 |

| | | | |
|----------------|--------|-------------------|----------|
| Omnibus: | 44.907 | Durbin-Watson: | 1.636 |
| Prob(Omnibus): | 0.000 | Jarque-Bera (JB): | 106.216 |
| Skew: | -0.783 | Prob(JB): | 8.62e-24 |
| Kurtosis: | 5.624 | Cond. No. | 1.82e+17 |

Notes:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
- [2] The smallest eigenvalue is 3.36e-26. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.

```
reg = LinearRegression()
reg.fit(X, y)
```

```
# X를 다시 학습시켜 훈련 데이터 전체의 예측 정도는 확인해보자.
X_pred = reg.predict(X)
get_nmae(X_pred, y)
```

0.30812151147228606

모델학습 과정 - XGBoost의 적용

➤ XGBoost 적용(GridSearchCV를 활용한 최적 Parameter 찾기)

```
reg_cv = XGBRegressor()
xgb_params = {'max_depth' : [4, 5, 6, 7],
              'n_estimators' : [5, 10, 20, 50, 100, 200],
              'random_state' : [42]}
grid = GridSearchCV(estimator = reg_cv, param_grid = xgb_params, scoring = 'neg_mean_absolute_error')

grid.fit(X, y)
best_param = grid.best_params_
print(best_param)

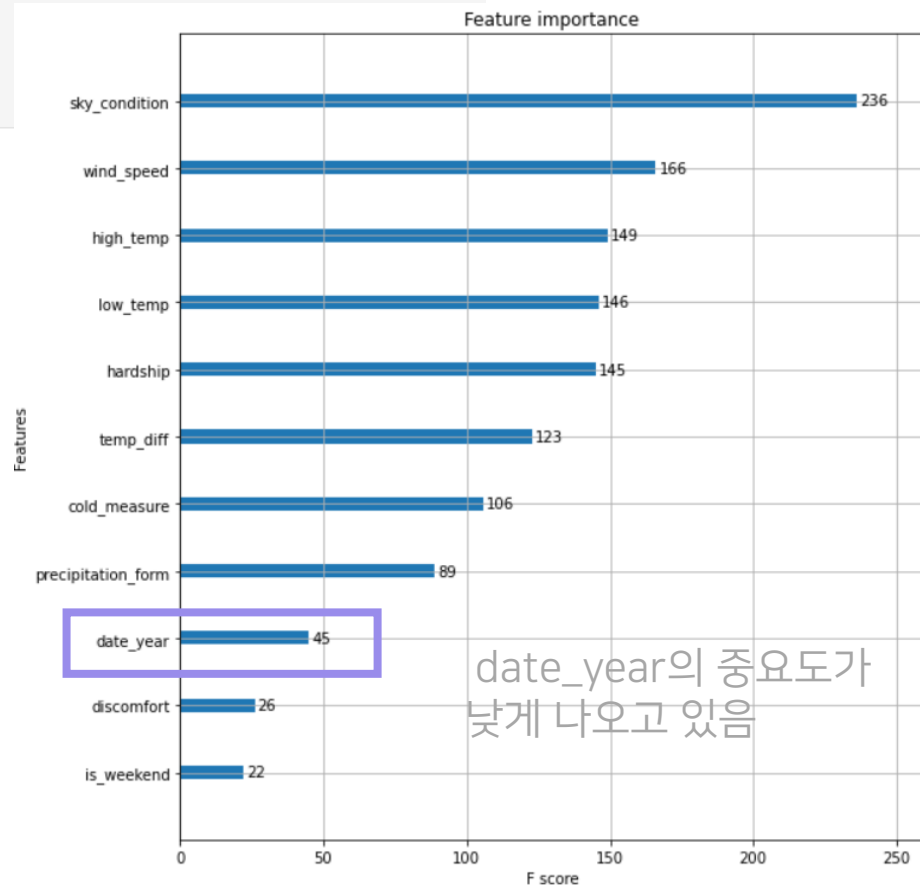
{'max_depth': 4, 'n_estimators': 50, 'random_state': 42}
```

실제 훈련에서는 전체 기간에 대한 반영 필요로
Estimator 개수를 100개로 증가.

```
reg = XGBRegressor(max_depth = 4, n_estimators = 100)
reg.fit(X, y)
pred = reg.predict(test_df)

# X값을 토대로 다시 정확도 확인
X_pred = reg.predict(X)
get_nmae(X_pred, y)
```

0.005514551818855783

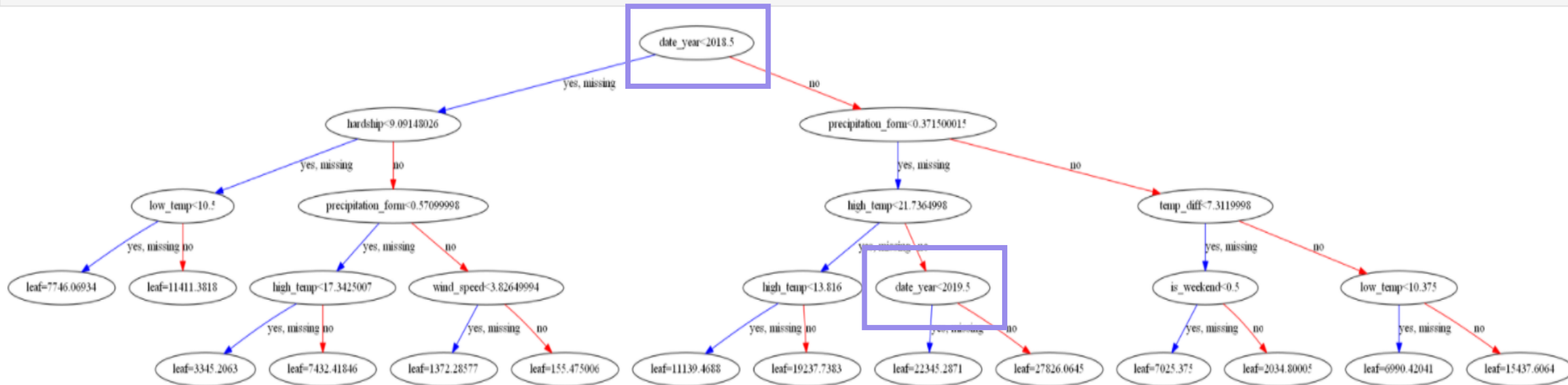


모델개선 과정 - XGBoost에서 일어난 일



XGBoost의 시각화

```
plot_tree(reg)
fig = plt.gcf()
fig.set_size_inches(35, 25)
```



학습 흐름을 보았을 때, date_year로 2020년과 2021년의 유의미한 차이를 반영하기 어려워 보인다.

모델개선 과정 - 연도에 따른 변화분 반영



상승분 고려

```
reg_2018 = sum(train_df.loc[train_df['date_year'] == 2018, 'number_of_rentals'].values)
reg_2019 = sum(train_df.loc[train_df['date_year'] == 2019, 'number_of_rentals'].values)
reg_2020 = sum(train_df.loc[train_df['date_year'] == 2020, 'number_of_rentals'].values)
print(reg_2019/reg_2018)
print(reg_2020/reg_2019)
```

```
2.095622727544442
1.2358394161314505
```



최종 학습 모델

```
# 최종 모델이다. 예측된 값에 일괄적으로 1.2배를 곱해서 상승분 반영으로 마무리한다.
reg = XGBRegressor(max_depth = 4, n_estimators = 100)
reg.fit(X, y)
pred = 1.2 * reg.predict(test_df)
# get_result(pred)
```



2020 ~ 2021 상승률이
전년과 동률이라고 가정하였습니다.

이에 따라, pred에 1.2배를 곱합니다.

마무리



최종 결과 (Private 1위!)

| # | 팀 | 팀 멤버 | 최종점수 |
|---|--------|---|---------|
| 1 | 다람이도토리 |  | 0.25052 |
| 1 | 다람이도토리 |  | 0.25052 |



개선방향

- 날씨가 좋지 않은 날에 대한 기준 성립 후 1.2배 미부여
- 월별/시기별 연도별 상승분에 대한 상세 반영
- 시계열 모델이나 딥러닝 모델의 적용 시도
- Train/Test를 나눌 수 있는 효율적인 방법에 대한 고려