[목차]

태스크 정의

워싱턴 D.C의 Bike Sharing 데이터를 활용해서 자전거 대여량을 예측한다.

모델 및 성능 평가 지표 선정

과제 실습을 통해 Data Engineer 과정을 복습하는 데 목적이 있으므로, 학습한 데이터 분석 방법 및 모델을 다양하게 적용한다.

- 결정 트리
- 랜덤포레스트
- kNN
- 선형 회귀
- ...

Bike Sharing 데이터의 수치형 Target 데이터(자전거 대여량)가 있으므로, 성능 평가 지표는 MSE로 한다.

데이터 탐색

데이터는 kaggle에서 구할 수 있다(https://www.kaggle.com/c/bike-sharing-demand/).

train 데이터와 test 데이터가 나눠져서 제공된다. test 데이터에는 Target 데이터가 빠져 있으며, Target 데이터를 채워서 kaggle에 제출하면 score를 확인할 수 있다.
약 2년 간의 1시간 단위 데이터가 train 데이터로 제공된다.

column	의미
datetime	시간. 연-월-일 시:분:초
season	계절. 봄(1), 여름(2), 가을(3), 겨울(4)
holiday	공휴일 여부. 공휴일(1), 공휴일 아님(0)
workingday	근무일 여부. 근무일(1), 근무일 아님(0)
weather	날씨. 깨끗한 날씨 or 약간의 구름(1), 구름 or 약간의 안개(2), 약간의 눈/비 or 천둥(3), 우박 or 많은 비(4)
temp	섭씨 온도
atemp	체감 섭씨 온도

column	의미
humidity	습도
windspeed	풍속
casual	비회원의 자전거 대여량
registered	회원의 자전거 대여량
count	총 자전거 대여량

데이터 전처리

시간 데이터 세분화

In [1]:
```
import pandas as pd

train = pd.read_csv('./data/bike_sharing_demand/train.csv', parse_dates=['datetime'])
train.head()
```

Out[1]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	r
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	

In [2]:
```
test = pd.read_csv('./data/bike_sharing_demand/test.csv', parse_dates=['datetime'])
test.head()
```

Out[2]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed
0	2011-01-20 00:00:00	1	0	1	1	10.66	11.365	56	26.0027
1	2011-01-20 01:00:00	1	0	1	1	10.66	13.635	56	0.0000
2	2011-01-20 02:00:00	1	0	1	1	10.66	13.635	56	0.0000
3	2011-01-20 03:00:00	1	0	1	1	10.66	12.880	56	11.0014

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed
4	2011-01-20 04:00:00	1	0	1	1	10.66	12.880	56	11.0014

In [3]:

```
# datetime 데이터 세분화
train['datetime-year'] = train['datetime'].dt.year
train["datetime-month"] = train["datetime"].dt.month
train["datetime-day"] = train["datetime"].dt.day
train['datetime-hour'] = train['datetime'].dt.hour
train["datetime-minute"] = train["datetime"].dt.minute
train["datetime-second"] = train["datetime"].dt.second
## 월(0), 화(1), 수(2), 목(3), 금(4), 토(5), 일(6)
train['datetime-dayofweek'] = train['datetime'].dt.dayofweek

train.head()
```

Out[3]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	r
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	

In [4]:

```
# teset 데이터에도 feature에 연도, 시간, 요일을 넣기 위해서 datetime 전처리
test['datetime-year'] = test['datetime'].dt.year
test["datetime-month"] = test["datetime"].dt.month
test["datetime-day"] = test["datetime"].dt.day
test['datetime-hour'] = test['datetime'].dt.hour
test["datetime-minute"] = test["datetime"].dt.minute
test["datetime-second"] = test["datetime"].dt.second
## 월(0), 화(1), 수(2), 목(3), 금(4), 토(5), 일(6)
test['datetime-dayofweek'] = test['datetime'].dt.dayofweek

test.head()
```

Out[4]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	datetime-year
0	2011-01-20 00:00:00	1	0	1	1	10.66	11.365	56	26.0027	2011

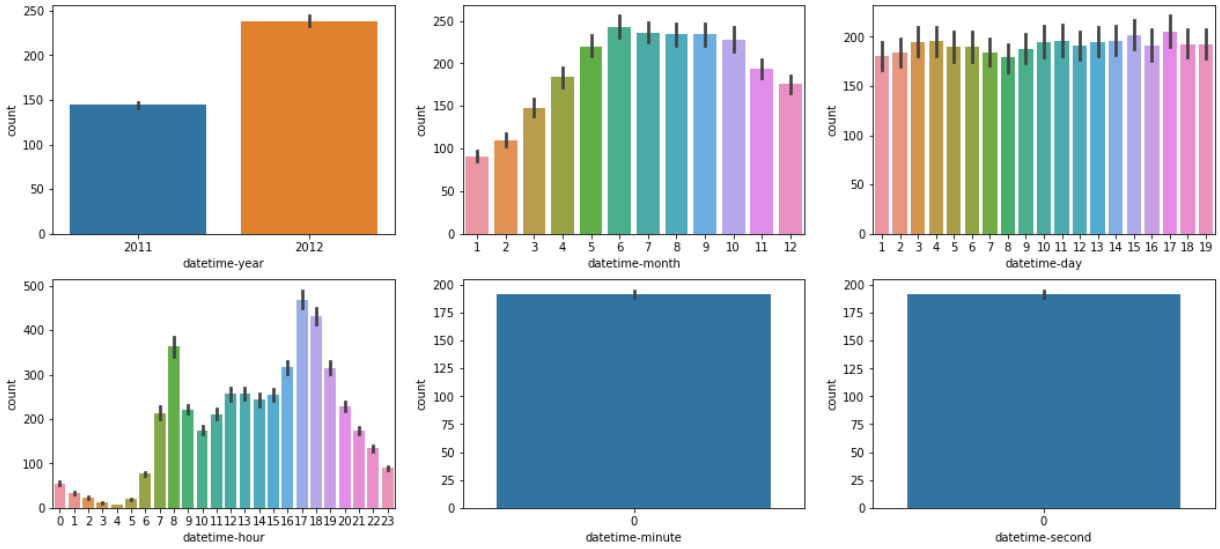
	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	datetime-year
1	2011-01-20 01:00:00	1	0	1	1	10.66	13.635	56	0.0000	2011
2	2011-01-20 02:00:00	1	0	1	1	10.66	13.635	56	0.0000	2011
3	2011-01-20 03:00:00	1	0	1	1	10.66	12.880	56	11.0014	2011
4	2011-01-20 04:00:00	1	0	1	1	10.66	12.880	56	11.0014	2011

feature 선정(랜덤포레스트, kNN, 선형 회귀)

데이터 요소 별 특성을 살펴보며 feature를 선정한다

In [5]:
```
%matplotlib inline

import seaborn as sns
import matplotlib.pyplot as plt

# 6개의 그래프를 2행 3열로 그리기
figure, ((ax1, ax2, ax3), (ax4, ax5, ax6)) = plt.subplots(nrows=2, ncols=3)

# 시각화의 전체 사이즈는 18x8로 설정
figure.set_size_inches(18, 8)

# seaborn의 barplot으로 subplots의 각 구역에
# 연, 월, 일, 시, 분, 초 별 자전거 대여량을 출력합니다.
sns.barplot(data=train, x="datetime-year", y="count", ax=ax1)
sns.barplot(data=train, x="datetime-month", y="count", ax=ax2)
sns.barplot(data=train, x="datetime-day", y="count", ax=ax3)
sns.barplot(data=train, x="datetime-hour", y="count", ax=ax4)
sns.barplot(data=train, x="datetime-minute", y="count", ax=ax5)
sns.barplot(data=train, x="datetime-second", y="count", ax=ax6)
```

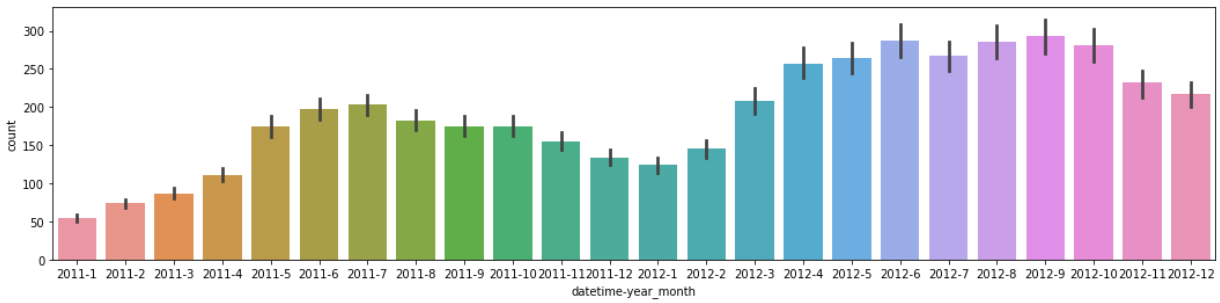
Out[5]: <AxesSubplot:xlabel='datetime-second', ylabel='count'>



- datetime-year : 2011년보다 2012년에 count가 높다. 연도에 따라 성장 중임을 추정할 수 있음
- datetime-month : 6월 ~ 10월 까지 대여량이 높으며, 12월 ~ 2월에는 대여량이 낮음을 확인
- datetime-day : 1일부터 19일 까지의 데이터만 존재함을 확인. test 데이터에 이후의 일자가 있음을 확인
- datetime-hour : 0시부터 5시까지는 count가 적음을 확인 할 수 있으며, 8시, 17시, 18시에 count가 높음을 확인
- datetime-minute, second : 시간 단위 데이터이므로 값이 0 밖에 없다

In [6]:
```
# datetime-year, datetime-month 데이터를 이어서 보기위한 전처리
train["datetime-year(str)"] = train["datetime-year"].astype('str')
train["datetime-month(str)"] = train["datetime-month"].astype('str')
train["datetime-year_month"] = train["datetime-year(str)"] + "-" + train["datetime-mo

figure, ax3 = plt.subplots(nrows=1, ncols=1)
figure.set_size_inches(18, 4)

sns.barplot(data=train, x="datetime-year_month", y="count", ax=ax3)
```

Out[6]: <AxesSubplot:xlabel='datetime-year_month', ylabel='count'>



- 2011.12 값은 2011.01 값과 차이가 많이 나지만, 다음 해 1월인 2012.01과는 큰 차이가 없다
- 따라서 month보다는 시간의 흐름에 따라 상승하는 경향이 강하다는 것을 알 수 있다
- year와 month를 합쳐서 feature로 선정하거나 아니면 month를 feature로 쓰지 않는 방법 등이 있다
- 이번 과제에서는 year만 feature로 선정하려고 한다

여기까지의 결과를 토대로 아래와 같은 feature를 선정하였다

In [7]:
```
# 모델 학습에 필요한 feature 선정
feature_columns = [
    "season",
    "holiday",
    "workingday",
    "weather",
    "temp",
    "atemp",
    "humidity",
    "windspeed",
    "datetime-year",
    "datetime-hour",
    "datetime-dayofweek"
]
```

In [8]:
```
# 모델 예측 대상 설정
target_column = 'count'
```

In [9]:
```
X_train = train[feature_columns]
X_train.head()
```

Out[9]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	datetime-year	datetime hou
0	1	0	0	1	9.84	14.395	81	0.0	2011	
1	1	0	0	1	9.02	13.635	80	0.0	2011	
2	1	0	0	1	9.02	13.635	80	0.0	2011	
3	1	0	0	1	9.84	14.395	75	0.0	2011	
4	1	0	0	1	9.84	14.395	75	0.0	2011	

In [10]:
```
X_test = test[feature_columns]
X_test.head()
```

Out[10]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	datetime-year	datetime hou
0	1	0	1	1	10.66	11.365	56	26.0027	2011	
1	1	0	1	1	10.66	13.635	56	0.0000	2011	
2	1	0	1	1	10.66	13.635	56	0.0000	2011	
3	1	0	1	1	10.66	12.880	56	11.0014	2011	
4	1	0	1	1	10.66	12.880	56	11.0014	2011	

In [11]:
```
y_train = train[target_column]
y_train.head()
```

Out[11]:
```
0    16
1    40
```

```
2    32
3    13
4     1
Name: count, dtype: int64
```

feature 선정(결정 트리 모델)

결정 트리 모델에서는 불쾌지수 feature를 하나 더 추가하였고, windspeed가 0인 값을 대상으로 보간법을 적용하였다.

In [12]:
```
# 불쾌지수
train["THI"] = ((9/5)*train["temp"])-(0.55*(1-(0.01*train["humidity"]))*((9/5)*train
test["THI"] = ((9/5)*test["temp"])-(0.55*(1-(0.01*test["humidity"]))*((9/5)*test["te
```

In [13]:
```
import numpy as np

# train.interpolate(method='linear', limit_direction='forward', axis=0)
train["windspeed_ipd"] = train["windspeed"]

train.loc[train["windspeed_ipd"] == 0, "windspeed_ipd"] = np.nan  #windspeed 0인값을
train['windspeed_ipd'].interpolate(method='linear', order=2, inplace=True)  #interpo

train.loc[train["windspeed_ipd"].isnull(), "windspeed_ipd"] = 10.0
```

In [14]:
```
# train.interpolate(method='linear', limit_direction='forward', axis=0)
test["windspeed_ipd"] = test["windspeed"]

test.loc[test["windspeed_ipd"] == 0, "windspeed_ipd"] = np.nan  #windspeed 0인값을 우
test['windspeed_ipd'].interpolate(method='linear', order=2, inplace=True)  #interpola

test.loc[test["windspeed_ipd"].isnull(), "windspeed_ipd"] = 10.0
```

In [15]:
```
feature_names = [
    'season',
    'holiday',
    'workingday',
    'weather',
    'temp',
    'atemp',
    'humidity',
    'datetime-year',
    'datetime-hour',
    'datetime-dayofweek',
    'windspeed_ipd',
    'THI'
]
```

In [16]:
```
X_train_dt = train[feature_names]
X_train_dt.head()
```

Out[16]:

	season	holiday	workingday	weather	temp	atemp	humidity	datetime-year	datetime-hour	datetime dayofwee
0	1	0	0	1	9.84	14.395	81	2011	0	
1	1	0	0	1	9.02	13.635	80	2011	1	

	season	holiday	workingday	weather	temp	atemp	humidity	datetime-year	datetime-hour	datetime dayofwee
2	1	0	0	1	9.02	13.635	80	2011	2	
3	1	0	0	1	9.84	14.395	75	2011	3	
4	1	0	0	1	9.84	14.395	75	2011	4	

In [17]:
```
X_test_dt = test[feature_names]
X_test_dt.head()
```

Out[17]:

	season	holiday	workingday	weather	temp	atemp	humidity	datetime-year	datetime-hour	datetime dayofwee
0	1	0	1	1	10.66	11.365	56	2011	0	
1	1	0	1	1	10.66	13.635	56	2011	1	
2	1	0	1	1	10.66	13.635	56	2011	2	
3	1	0	1	1	10.66	12.880	56	2011	3	
4	1	0	1	1	10.66	12.880	56	2011	4	

In [18]:
```
label_name = "count"
y_train_dt = np.log(train[label_name]+1)
y_train_dt.head()
```

Out[18]:
```
0    2.833213
1    3.713572
2    3.496508
3    2.639057
4    0.693147
Name: count, dtype: float64
```

학습 및 성능 평가

결정 트리

In [19]:
```
from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor(random_state= 37)  # random_state 값을 고정, 성능과는 치

model
```

Out[19]:
```
DecisionTreeRegressor(random_state=37)
```

In [20]:
```
model.fit(X_train_dt, y_train_dt)
```

Out[20]:
```
DecisionTreeRegressor(random_state=37)
```

In [21]:
```
predictions = model.predict(X_test_dt)
predictions[:5]
```

Out[21]:
```
array([2.56494936, 1.38629436, 0.69314718, 0.69314718, 0.69314718])
```

In [22]:
```
submit = pd.read_csv('./data/bike_sharing_demand/sampleSubmission.csv')
submit.shape
```

Out[22]:
```
(6493, 2)
```

In [23]:
```
submit["count"] = np.exp(predictions) - 1
submit.head()
```

Out[23]:

	datetime	count
0	2011-01-20 00:00:00	12.0
1	2011-01-20 01:00:00	3.0
2	2011-01-20 02:00:00	1.0
3	2011-01-20 03:00:00	1.0
4	2011-01-20 04:00:00	1.0

In [24]:
```
submit.to_csv("./data/bike_sharing_demand/decision-tree_0.csv", index = False)

pd.read_csv("./data/bike_sharing_demand/decision-tree_0.csv").head()
```

Out[24]:

	datetime	count
0	2011-01-20 00:00:00	12.0
1	2011-01-20 01:00:00	3.0
2	2011-01-20 02:00:00	1.0
3	2011-01-20 03:00:00	1.0
4	2011-01-20 04:00:00	1.0

kaggle 제출 결과

Submission and Description	Private Score	Public Score
decision-tree_0.csv just now by JaeYoung Jang decision-tree, default	0.52225	0.52225

랜덤포레스트

In [25]:
```
# 랜덤포레스트 모델 생성
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor()
model
```

Out[25]:
```
RandomForestRegressor()
```

In [26]:
```
# 랜덤포레스트 모델 교차 검증
```

```
# kaggle에서는 Bike Sharing Demand에 대해 RMSLE(Root Mean Squared Logarithmic Error) 방
# RMSLE 점수는 kaggle에 제출하면 얻을 수 있다.
# 과제에서는 모델 간의 대략적인 비교만을 위해 기본적인 scoring 공식을 사용하였다.
from sklearn.model_selection import cross_val_score

score = cross_val_score(model,
                        X_train,
                        y_train,
                        cv=20).mean()
score
```

Out[26]: 0.8767994499193291

In [27]:
```
model.fit(X_train, y_train)
```

Out[27]: RandomForestRegressor()

In [28]:
```
predictions = model.predict(X_test)
predictions
```

Out[28]: array([11.4 ,   4.71 ,   3.55 , ..., 105.16 , 104.495, 48.48])

In [29]:
```
# kaggle 제출용 template
submission = pd.read_csv('./data/bike_sharing_demand/sampleSubmission.csv')
submission.head()
```

Out[29]:

	datetime	count
0	2011-01-20 00:00:00	0
1	2011-01-20 01:00:00	0
2	2011-01-20 02:00:00	0
3	2011-01-20 03:00:00	0
4	2011-01-20 04:00:00	0

In [30]:
```
submission['count'] = predictions
submission.head()
```

Out[30]:

	datetime	count
0	2011-01-20 00:00:00	11.40
1	2011-01-20 01:00:00	4.71
2	2011-01-20 02:00:00	3.55
3	2011-01-20 03:00:00	3.77
4	2011-01-20 04:00:00	2.95

In [31]:
```
submission.to_csv('./data/bike_sharing_demand/random-forest_0.csv', index=False)
```

kaggle 제출 결과

RMSLE 공식으로 측정하므로 0에 가까울 수록 성능이 좋다

Submission and Description	Private Score	Public Score
random-forest_0.csv 2 minutes ago by JaeYoung Jang random-forest, default	0.42031	0.42031

kNN

In [32]:
```
# kNN 모델 생성
from sklearn.neighbors import KNeighborsRegressor

model = KNeighborsRegressor()
model
```

Out[32]:
```
KNeighborsRegressor()
```

In [33]:
```
# 랜덤포레스트 모델보다 점수가 낮아서, kaggle 결과도 안좋을 것으로 예상된다.
score =cross_val_score(model,
                       X_train,
                       y_train,
                       cv=20).mean()
score
```

Out[33]:
```
0.1412418300814157
```

In [34]:
```
model.fit(X_train, y_train)
predictions = model.predict(X_test)
predictions
```

Out[34]:
```
array([ 19. ,  20. ,  15.4, ...,  73.6, 105.6,  61.4])
```

In [35]:
```
# kaggle 제출용 template
submission = pd.read_csv('./data/bike_sharing_demand/sampleSubmission.csv')
submission['count'] = predictions
submission.head()
```

Out[35]:

	datetime	count
0	2011-01-20 00:00:00	19.0
1	2011-01-20 01:00:00	20.0
2	2011-01-20 02:00:00	15.4
3	2011-01-20 03:00:00	14.4
4	2011-01-20 04:00:00	14.2

In [36]:
```
submission.to_csv('./data/bike_sharing_demand/kNN_0.csv', index=False)
```

kaggle 제출 결과

RMSLE 공식으로 측정하므로 0에 가까울 수록 성능이 좋다

Submission and Description	Private Score	Public Score
kNN_0.csv a few seconds ago by JaeYoung Jang kNN, default	0.87218	0.87218

선형 회귀

In [37]:
```
# 선형 회귀 모델 생성
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model
```

Out[37]:
```
LinearRegression()
```

In [38]:
```
# kNN보다 더 점수가 낮다
score =cross_val_score(model,
                       X_train,
                       y_train,
                       cv=20).mean()
score
```

Out[38]:
```
0.12388279311614711
```

In [39]:
```
model.fit(X_train, y_train)
predictions = model.predict(X_test)
predictions
```

Out[39]:
```
array([-23.27179232, -20.84936197, -13.04580719, ..., 209.84495832,
       227.95174821, 217.86201958])
```

In [40]:
```
# kaggle 제출용 template
submission = pd.read_csv('./data/bike_sharing_demand/sampleSubmission.csv')
submission['count'] = predictions
submission.head()
```

Out[40]:

	datetime	count
0	2011-01-20 00:00:00	-23.271792
1	2011-01-20 01:00:00	-20.849362
2	2011-01-20 02:00:00	-13.045807
3	2011-01-20 03:00:00	-1.986454

	datetime	count
4	2011-01-20 04:00:00	5.817101

In [41]:
```
submission.to_csv('./data/bike_sharing_demand/linear-regression_0.csv', index=False)
```

kaggle 제출 결과

RMSLE 공식으로 측정하므로 0에 가까울 수록 성능이 좋다

Submission and Description	Private Score	Public Score
linear-regression_0.csv a minute ago by JaeYoung Jang linear-regression, default	Error	Error

음수로 예측해서 kaggle 점수 측정 오류가 났다

성능 향상을 위한 시도

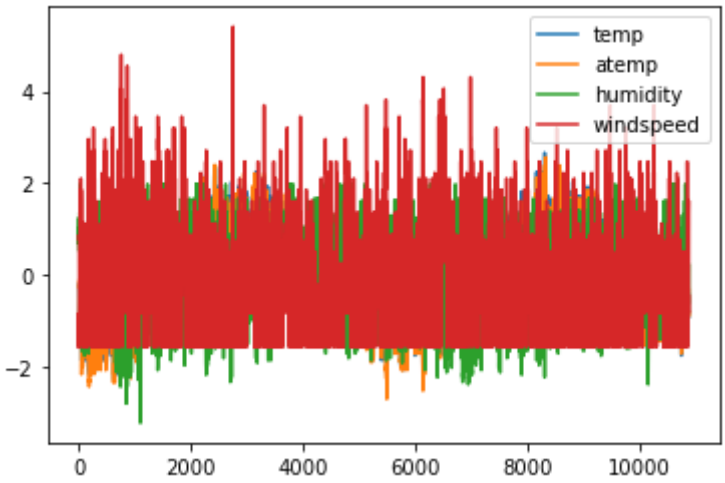
성능 향상을 위해 시도한 내용과 그에 따른 성능 측정 결과

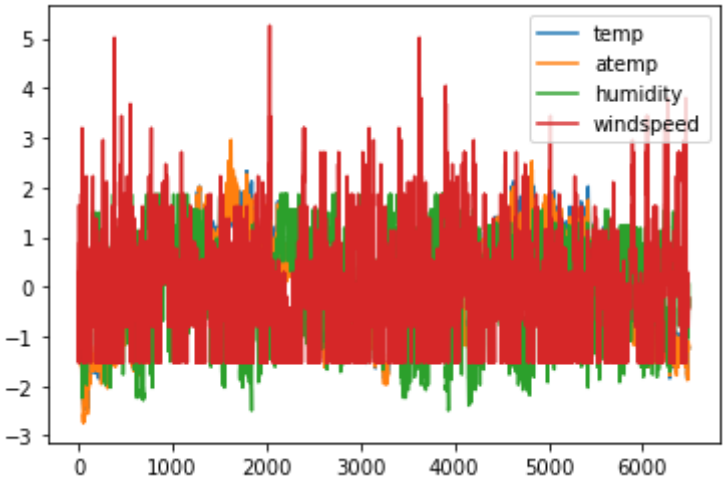
수치형 데이터 정규화

In [42]:
```
# Z점수 표준화 함수
def zscore_standize(arr):
    return (arr - arr.mean()) / (arr.std())
```

In [43]:
```
# 수치형 데이터 temp, atemp, humidity, windspeed 정규화 column 추가
X_train_standized = X_train.copy()
X_train_standized = X_train_standized.apply(zscore_standize)
X_test_standized = X_test.copy()
X_test_standized = X_test_standized.apply(zscore_standize)

# 정규화 확인
X_train_standized[['temp', 'atemp', 'humidity', 'windspeed']].plot()
X_test_standized[['temp', 'atemp', 'humidity', 'windspeed']].plot()
```

Out[43]: <AxesSubplot:>





랜덤포레스트

In [44]:

```
# 랜덤포레스트 모델 생성
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor()

# 랜덤포레스트 학습
model.fit(X_train_standized, y_train)
predictions = model.predict(X_test_standized)

# kaggle 제출용 template
submission = pd.read_csv('./data/bike_sharing_demand/sampleSubmission.csv')
submission['count'] = predictions

# kaggle 제출용 csv 저장
submission.to_csv('./data/bike_sharing_demand/random-forest_1.csv', index=False)
```

kaggle 제출 결과

Submission and Description	Private Score	Public Score
random-forest_1.csv a few seconds ago by JaeYoung Jang random-forest, standized	0.42338	0.42338

큰 차이가 없었다.

kNN

In [45]:
```
# kNN 모델 생성
from sklearn.neighbors import KNeighborsRegressor

model = KNeighborsRegressor()

# kNN 학습
model.fit(X_train_standized, y_train)
predictions = model.predict(X_test_standized)

# kaggle 제출용 template
submission = pd.read_csv('./data/bike_sharing_demand/sampleSubmission.csv')
submission['count'] = predictions

# kaggle 제출용 csv 저장
submission.to_csv('./data/bike_sharing_demand/kNN_1.csv', index=False)
```

kaggle 제출 결과

Submission and Description	Private Score	Public Score
kNN_1.csv just now by JaeYoung Jang kNN, standized	0.93893	0.93893

약간 나빠졌다.

선형 회귀

In [46]:
```
# 선형 회귀 모델 생성
from sklearn.linear_model import LinearRegression

model = LinearRegression()

# 선형 회귀 학습
model.fit(X_train_standized, y_train)
predictions = model.predict(X_test_standized)

# kaggle 제출용 template
submission = pd.read_csv('./data/bike_sharing_demand/sampleSubmission.csv')
submission['count'] = predictions

# kaggle 제출용 csv 저장
submission.to_csv('./data/bike_sharing_demand/linear-regression_1.csv', index=False)
```

kaggle 제출 결과

Submission and Description	Private Score	Public Score
linear-regression_1.csv just now by JaeYoung Jang linear-regression, standized	Error	Error

여전히 결과에 음수가 있어서 오류가 난다.

하이퍼 파라미터 조절

랜덤포레스트

In [47]:
```
# GridSearchCV를 통해 랜덤포레스트 하이퍼 파라미터 튜닝
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV

params = { 'n_estimators' : [10, 100],
           'max_depth' : [6, 8, 10, 12],
           'min_samples_leaf' : [8, 12, 18],
           'min_samples_split' : [8, 16, 20]
         }

model = RandomForestRegressor(n_jobs=-1)
grid_cv = GridSearchCV(model, param_grid = params, cv = 3, n_jobs = -1)
grid_cv.fit(X_train, y_train)

print('최적 하이퍼 파라미터: ', grid_cv.best_params_)
print('최고 예측 정확도: {:.4f}'.format(grid_cv.best_score_))
```

```
최적 하이퍼 파라미터:  {'max_depth': 10, 'min_samples_leaf': 8, 'min_samples_split': 16, 'n_estimators': 10}
최고 예측 정확도: 0.7164
```

In [48]:
```
# 최적 하이퍼 파라미터로 랜덤포레스트 학습
model = RandomForestRegressor(max_depth=12,
                              min_samples_leaf=8,
                              min_samples_split=16,
                              n_estimators=100,
                              n_jobs=-1)
model.fit(X_train, y_train)

predictions = model.predict(X_test)

# kaggle 제출용 template
submission = pd.read_csv('./data/bike_sharing_demand/sampleSubmission.csv')
submission['count'] = predictions

# kaggle 제출용 csv 저장
submission.to_csv('./data/bike_sharing_demand/random-forest_2.csv', index=False)
```

kaggle 제출 결과

Submission and Description	Private Score	Public Score
random-forest_2.csv just now by JaeYoung Jang random-forest, hyper parameter tuning	0.43414	0.43414

앞선 결과들과 비슷하다

kNN

In [49]:
```
from sklearn.neighbors import KNeighborsRegressor
```

```
from sklearn.model_selection import GridSearchCV

params = {
    'n_neighbors' : list(range(1,20)),
    'weights' : ["uniform", "distance"],
    'metric' : ['euclidean', 'manhattan', 'minkowski']
}

model = KNeighborsRegressor()

grid_cv = GridSearchCV(model, param_grid = params, cv = 3, n_jobs = -1)
grid_cv.fit(X_train, y_train)

print('최적 하이퍼 파라미터: ', grid_cv.best_params_)
print('최고 예측 정확도: {:.4f}'.format(grid_cv.best_score_))
```

```
최적 하이퍼 파라미터:  {'metric': 'manhattan', 'n_neighbors': 15, 'weights': 'distanc
e'}
최고 예측 정확도: 0.3278
```

In [50]:

```
# 최적 하이퍼 파라미터로 랜덤포레스트 학습
model = KNeighborsRegressor(metric='manhattan',
                            n_neighbors=15,
                            weights='distance')
model.fit(X_train, y_train)

predictions = model.predict(X_test)

# kaggle 제출용 template
submission = pd.read_csv('./data/bike_sharing_demand/sampleSubmission.csv')
submission['count'] = predictions

# kaggle 제출용 csv 저장
submission.to_csv('./data/bike_sharing_demand/kNN_2.csv', index=False)
```

kaggle 제출 결과

Submission and Description	Private Score	Public Score
kNN_2.csv just now by JaeYoung Jang kNN, hyper parameter tuning	0.93767	0.93767

첫 번째 결과보다는 안좋고, 두 번째랑은 비슷하다

Min-Max 스케일러 사용

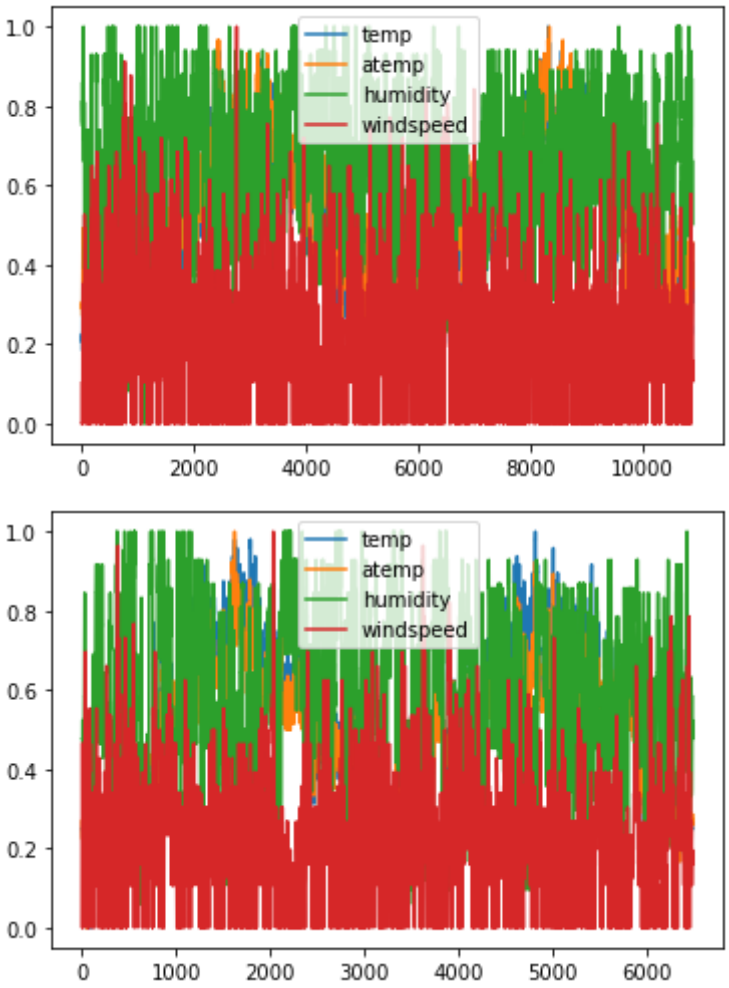
선형 회귀의 경우, 결과가 음수가 나왔기 때문에 이를 보정하기 위해 스케일을 0~1로 줄이는 Min-Max 스케일러를 사용해보기로 하였다.

In [51]:

```
# 최소최대 정규화
def minmax_normalize(arr):
    return (arr - arr.min()) / (arr.max() - arr.min())

# 수치형 데이터 temp, atemp, humidity, windspeed 정규화 column 추가
X_train_minstandized = X_train.copy()
X_train_standized = X_train_standized.apply(minmax_normalize)
X_test_standized = X_test.copy()
```

```
X_test_standized = X_test_standized.apply(minmax_normalize)

# 정규화 확인
X_train_standized[['temp', 'atemp', 'humidity', 'windspeed']].plot()
X_test_standized[['temp', 'atemp', 'humidity', 'windspeed']].plot()
```

Out[51]: <AxesSubplot:>



선형 회귀

In [52]:

```
# 선형 회귀 모델 생성
from sklearn.linear_model import LinearRegression

model = LinearRegression()

# 선형 회귀 학습
model.fit(X_train_standized, y_train)
predictions = model.predict(X_test_standized)

# kaggle 제출용 template
submission = pd.read_csv('./data/bike_sharing_demand/sampleSubmission.csv')
submission['count'] = predictions

# kaggle 제출용 csv 저장
submission.to_csv('./data/bike_sharing_demand/linear-regression_2.csv', index=False)
```

kaggle 제출 결과

Submission and Description	Private Score	Public Score
linear-regression_2.csv just now by JaeYoung Jang linear-regression, min-max scaler	Error	Error

여전히 결과에 음수가 있어서 오류가 난다.

최종 결과

성능이 가장 잘 되었던 경우 및 실험 결과

순서	모델	kaggle 제출 점수
1	결정 트리 기본	0.52225
2	랜덤포레스트 기본	0.42031
3	kNN 기본	0.87218
4	선형 회귀 기본	에러
5	랜덤포레스트 z-표준화	0.42338
6	kNN z-표준화	0.93893
7	선형 회귀 z-표준화	에러
8	랜덤포레스트 그리드 서치	0.43414
9	kNN 그리드 서치	0.93767
10	linear-regression min-max 정규화	에러

가장 좋은 결과는 랜덤포레스트로 아무 변화를 주지 않았을 때 얻을 수 있었다.

그리드 서치를 하면 그래도 조금은 나아질 것으로 기대했는데 나아지지 않았다. 기본적으로 scikit learn에서 제공하는 모델이 최적화가 잘 되어 있는 것으로 생각된다.

참고 자료

https://github.com/JasonKwak/python-study-ds/blob/main/06-bike-sharing-demand-0.43019.ipynb

https://injo.tistory.com/30

https://tobigs.gitbook.io/tobigs/data-analysis/python-knn