Kaggle 입문하기 - 데이터 분석 입문

학습 내용

- 캐글에 대해 이해하기
- 기본 모델과 정규화를 적용해 보기

- URL : https://www.kaggle.com/ (https://www.kaggle.com/)
- Competitions 선택하면 다양한 대회 확인 가능.
- 대회 주제 : Bike Sharing Demand
- https://www.kaggle.com/c/bike-sharing-demand (https://www.kaggle.com/c/bike-sharing-demand)

Data Fields

필드명	설명
datetime	hourly date + timestamp
season	1 = spring, 2 = summer, 3 = fall, 4 = winter
holiday	whether the day is considered a holiday
workingday	whether the day is neither a weekend nor holiday
weather	1: Clear, Few clouds, Partly cloudy, Partly cloudy 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
temp	temperature in Celsius (온도)
atemp	"feels like" temperature in Celsius (체감온도)
humidity	relative humidity (습도)
windspeed	wind speed (바람속도)
casual	number of non-registered user rentals initiated (비가입자 사용유저)
registered	number of registered user rentals initiated (가입자 사용유저)
count	number of total rentals (전체 렌탈 대수)

In [1]:

```
import pandas as pd
```

In [2]:

```
train = pd.read_csv("bike/train.csv", parse_dates=['datetime'])
test = pd.read_csv("bike/test.csv", parse_dates=['datetime'])
```

입력 데이터 선택

In [3]:

```
f_names = ['temp', 'atemp']
X_tr_all = train[f_names]         # 학습용 데이터의 변수 선택
last_X_test = test[f_names]       # 테스트 데이터의 변수 선택
```

출력 데이터 선택

In [4]:

```
y_tr_all = train['count']
```

데이터 나누기

In [5]:

```
from sklearn.model_selection import train_test_split
```

In [6]:

```
X_train, X_test, y_train, y_test = train_test_split(X_tr_all,
                                                    y_tr_all,
                                                    test_size=0.3,
                                                    random_state=77)
```

모델 만들기 및 제출

모델 만들기 및 예측 순서

- 모델을 생성한다. model = 모델명()
- 모델을 학습한다. model.fit(입력값, 출력값)
- 모델을 이용하여 예측 model.predict(입력값)

In [7]:

```
from sklearn.linear_model import LinearRegression
```

In [8]:

```
model = LinearRegression()
model.fit(X_train, y_train)
# 정확도 확인
print("학습용 세트 정확도: {:.3f}".format(model.score(X_train, y_train)))
print("테스트 세트 정확도: {:.3f}".format(model.score(X_test, y_test)))

model.predict(X_test)         # 예측(새로운 데이터로)
```

```
학습용 세트 정확도: 0.159
테스트 세트 정확도: 0.146
```

Out[8]:

```
array([235.46986679, 151.05560946, 218.26182702, ..., 133.09294136,
       151.05560946,  82.34013525])
```

In [9]:

```
print( model.coef_ )          # 모델(선형회귀의 계수)
print( model.intercept_)      # 모델(선형 회귀의 교차점)
```

```
[8.18286924 0.99950771]
3.8812023741951975
```

학습된 모델로 예측 후, 이값으로 제출하기

In [10]:

```
sub = pd.read_csv("bike/sampleSubmission.csv")
sub.head()
```

Out[10]:

	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 [11]:

```
pred = model.predict(last_X_test)   # 예측
sub['count'] = pred
sub
```

Out[11]:

	datetime	count
0	2011-01-20 00:00:00	102.469994
1	2011-01-20 01:00:00	104.738876
2	2011-01-20 02:00:00	104.738876
3	2011-01-20 03:00:00	103.984248
4	2011-01-20 04:00:00	103.984248
...	...	...
6488	2012-12-31 19:00:00	103.984248
6489	2012-12-31 20:00:00	103.984248
6490	2012-12-31 21:00:00	103.984248
6491	2012-12-31 22:00:00	104.738876
6492	2012-12-31 23:00:00	104.738876

6493 rows × 2 columns

제출하기

In [12]:

```
# 처음 만는 제출용 csv 파일, 행번호를 없애기
sub.to_csv("firstsubmission.csv", index=False)
```

여러개의 변수를 사용하기

In [13]:

```
train.columns
```

Out[13]:

```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count'],
      dtype='object')
```

In [14]:

```
f_names = ['season', 'holiday', 'workingday', 'weather', 'temp', 'atemp', 'humidity', 'windspeed']
X_tr_all = train[f_names]         # 학습용 데이터의 변수 선택
last_X_test = test[f_names]       # 테스트 데이터의 변수 선택

y_tr_all = train['count']
```

데이터 나누기

In [15]:

```
from sklearn.model_selection import train_test_split
```

In [16]:

```
X_train, X_test, y_train, y_test = train_test_split(X_tr_all,
                                                    y_tr_all,
                                                    test_size=0.3,
                                                    random_state=77)
```

In [17]:

```
model = LinearRegression()
model.fit(X_train, y_train)
# 정확도 확인
print("학습용 세트 정확도: {:.3f}".format(model.score(X_train, y_train)))
print("테스트 세트 정확도: {:.3f}".format(model.score(X_test, y_test)))
```

```
학습용 세트 정확도: 0.262
테스트 세트 정확도: 0.257
```

MinMaxScaler

In [18]:

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import PolynomialFeatures
```

In [19]:

```
f_names = ['season', 'holiday', 'workingday', 'weather', 'temp', 'atemp', 'humidity', 'windspeed']
X_tr_all = train[f_names]         # 학습용 데이터의 변수 선택
```

In [20]:

```
scaler = MinMaxScaler().fit(X_tr_all)
nor_X_tr_all = scaler.transform(X_tr_all)
last_X_test = test[f_names]       # 테스트 데이터의 변수 선택

y_tr_all = train['count']
```

In [21]:

```
X_train, X_test, y_train, y_test = train_test_split(nor_X_tr_all,
                                                    y_tr_all,
                                                    test_size=0.3,
                                                    random_state=77)
```

In [22]:

```
model = LinearRegression()
model.fit(X_train, y_train)
# 정확도 확인
print("학습용 세트 정확도: {:.3f}".format(model.score(X_train, y_train)))
print("테스트 세트 정확도: {:.3f}".format(model.score(X_test, y_test)))
```

```
학습용 세트 정확도: 0.262
테스트 세트 정확도: 0.257
```

Ridge, Lasso

In [23]:

```
from sklearn.linear_model import Ridge, Lasso
```

In [24]:

```
model = Ridge()
model.fit(X_train, y_train)
# 정확도 확인
print("학습용 세트 정확도: {:.3f}".format(model.score(X_train, y_train)))
print("테스트 세트 정확도: {:.3f}".format(model.score(X_test, y_test)))
```

```
학습용 세트 정확도: 0.262
테스트 세트 정확도: 0.257
```

In [25]:

```
model = Lasso()
model.fit(X_train, y_train)
# 정확도 확인
print("학습용 세트 정확도: {:.3f}".format(model.score(X_train, y_train)))
print("테스트 세트 정확도: {:.3f}".format(model.score(X_test, y_test)))
```

```
학습용 세트 정확도: 0.258
테스트 세트 정확도: 0.254
```

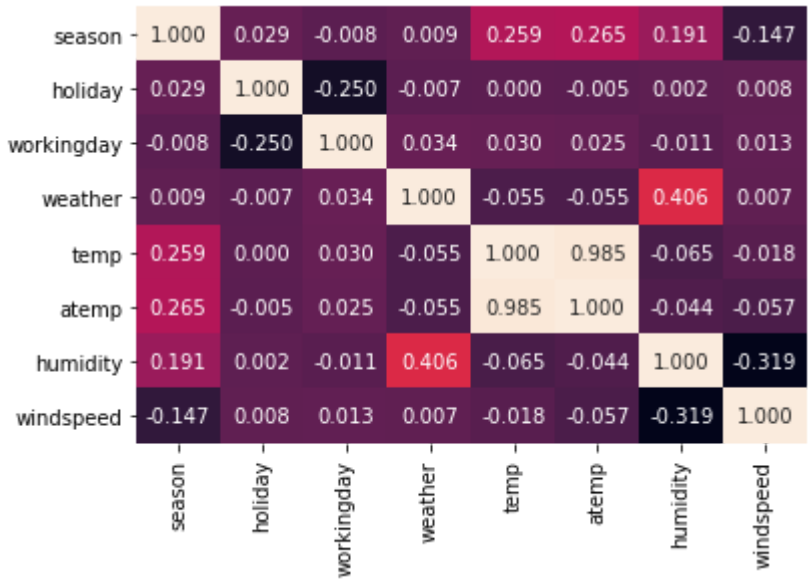
변수 생성

In [34]:

```
import seaborn as sns
sns.heatmap(X_tr_all.corr(), annot=True, fmt=".3f", cbar=False)
```

Out[34]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1cd53493f10>
```



In [35]:

```
print("원래 데이터 : ", X_tr_all.shape)

nor_X = MinMaxScaler().fit_transform(X_tr_all)  # 입력 데이터 정규화
ex_X = PolynomialFeatures(degree=2, include_bias=False).fit_transform(nor_X)  # 데이터 feature 추가

print("정규화, 추가 생성 : ", ex_X.shape, y_tr_all.shape)
print(type(X_tr_all), type(ex_X))
```

```
원래 데이터 :  (10886, 8)
정규화, 추가 생성 :  (10886, 44) (10886,)
<class 'pandas.core.frame.DataFrame'> <class 'numpy.ndarray'>
```

In [36]:

```
X_train, X_test, y_train, y_test = train_test_split(ex_X,
                                                    y_tr_all,
                                                    test_size=0.3,
                                                    random_state=77)
```

In [37]:

```
model_list = [LinearRegression(), Ridge(), Lasso()]
```

In [38]:

```
for model in model_list:
    model.fit(X_train, y_train)

    print("모델 : ", model)
    # 정확도 확인
    print("학습용 세트 정확도: {:.3f}".format(model.score(X_train, y_train)))
    print("테스트 세트 정확도: {:.3f}".format(model.score(X_test, y_test)))
```

```
모델 :  LinearRegression()
학습용 세트 정확도: 0.307
테스트 세트 정확도: 0.306
모델 :  Ridge()
학습용 세트 정확도: 0.302
테스트 세트 정확도: 0.304
모델 :  Lasso()
학습용 세트 정확도: 0.264
테스트 세트 정확도: 0.260
```

In [39]:

```
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[39]:

```
LinearRegression()
```

In [40]:

```
scaler = MinMaxScaler().fit(X_tr_all)
nor_X_test_all = scaler.transform(last_X_test)
ex_X = PolynomialFeatures(degree=2, include_bias=False).fit_transform(nor_X_test_all)
```

In [41]:

```
pred = model.predict(ex_X)   # 예측
sub['count'] = pred
sub.loc[sub['count'] < 0, 'count'] = 0
sub.head(3)
```

Out[41]:

	datetime	count
0	2011-01-20 00:00:00	120.228629
1	2011-01-20 01:00:00	117.465407
2	2011-01-20 02:00:00	117.465407

In [42]:

```
# 처음 만는 제출용 csv 파일, 행번호를 없애기
sub.to_csv("second_sub.csv", index=False)
```

1.37583

In [ ]:

```

```