여러가지 모델을 만들어보기

01 데이터 준비

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

In [2]:
```
train = pd.read_csv("../data/bike/train.csv", parse_dates=['datetime'])
test = pd.read_csv("../data/bike/test.csv", parse_dates=['datetime'])
sub = pd.read_csv("../data/bike/sampleSubmission.csv")
```

In [3]:
```
print(train.shape)   # : 행과 열 갯수 확인
print(test.shape)
```

```
(10886, 12)
(6493, 9)
```

In [4]:
```
import matplotlib.pyplot as plt ## seaborn 보다 고급 시각화 가능. but 코드 복잡
import seaborn as sns           ## seaborn은 matplotlib보다 간단하게 사용 가능
```

In [5]:
```
new_tr = train   # 데이터 백업
new_test = test
# train = new_tr  # 데이터 백업
# test = new_test
```

02 파생변수 생성

In [6]:
```
## 더미변수, 파생변수 생성
new_tr['year'] = new_tr['datetime'].dt.year
new_tr.head()
```

Out[6]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual
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 [7]:
```
new_tr['month'] = new_tr['datetime'].dt.month
new_tr['day'] = new_tr['datetime'].dt.day
new_tr['hour'] = new_tr['datetime'].dt.hour
new_tr['minute'] = new_tr['datetime'].dt.minute
new_tr['second'] = new_tr['datetime'].dt.second
```

```
new_tr['dayofweek'] = new_tr['datetime'].dt.dayofweek
new_tr[  ['datetime', 'year', 'month', 'day', 'hour', 'dayofweek']   ]
```

Out[7]:

	datetime	year	month	day	hour	dayofweek
0	2011-01-01 00:00:00	2011	1	1	0	5
1	2011-01-01 01:00:00	2011	1	1	1	5
2	2011-01-01 02:00:00	2011	1	1	2	5
3	2011-01-01 03:00:00	2011	1	1	3	5
4	2011-01-01 04:00:00	2011	1	1	4	5
...	...	...	...	...	...	...
10881	2012-12-19 19:00:00	2012	12	19	19	2
10882	2012-12-19 20:00:00	2012	12	19	20	2
10883	2012-12-19 21:00:00	2012	12	19	21	2
10884	2012-12-19 22:00:00	2012	12	19	22	2
10885	2012-12-19 23:00:00	2012	12	19	23	2

10886 rows × 6 columns

In [8]:
```
new_test['year'] = new_test['datetime'].dt.year
new_test['month'] = new_test['datetime'].dt.month
new_test['day'] = new_test['datetime'].dt.day
new_test['dayofweek'] = new_test['datetime'].dt.dayofweek
new_test['hour'] = new_test['datetime'].dt.hour
new_test['minute'] = new_test['datetime'].dt.minute
new_test['second'] = new_test['datetime'].dt.second
```

03 시각화 확인

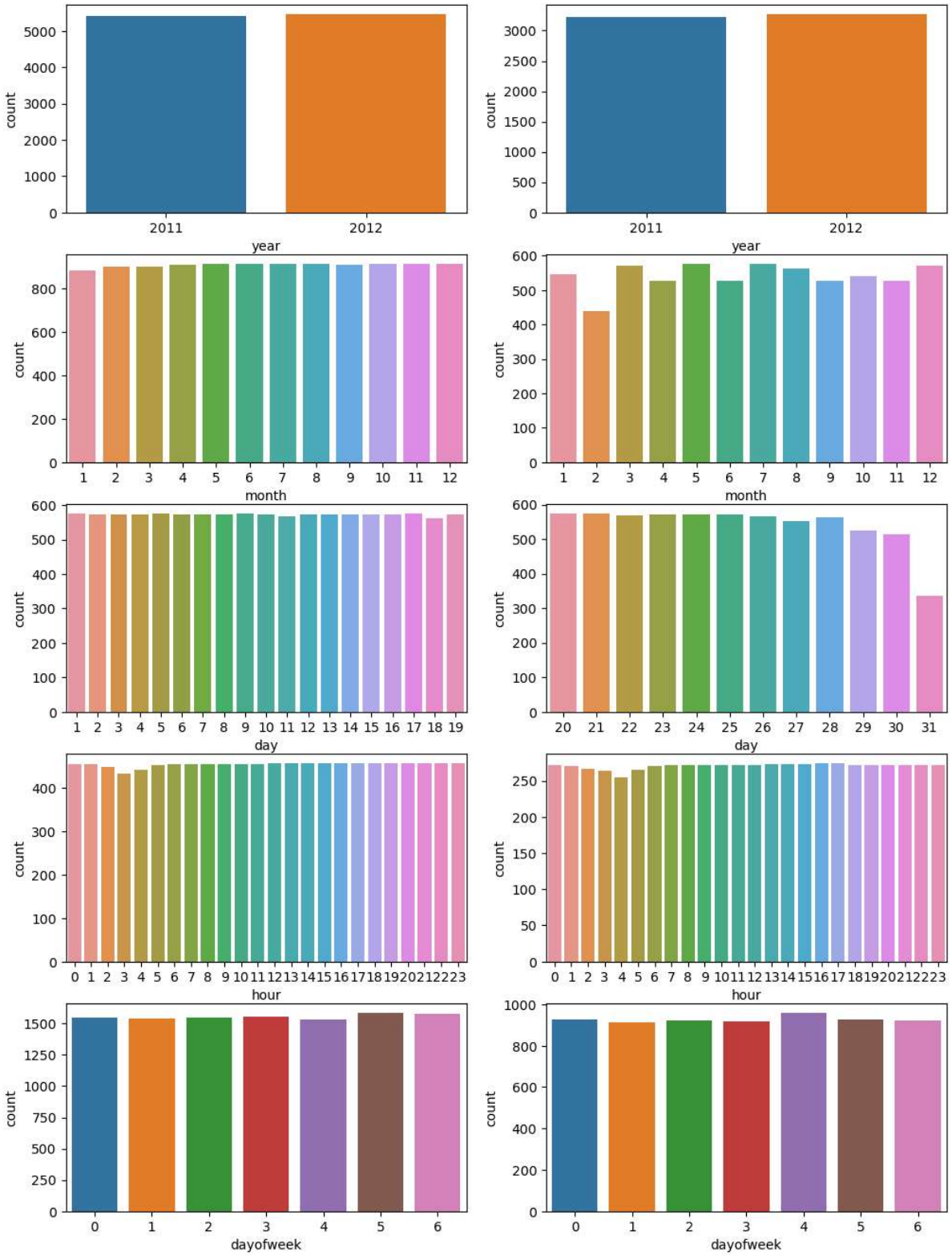
In [9]:
```
col_names = ['year','month','day','hour','dayofweek']
i = 0

plt.figure(figsize=(12,20))  ##전체 그래프 크기 지정

for name in col_names: ## 컬럼명으로 반복
  i = i+1
  plt.subplot(6,2,i)  ##2행2열, i = 1,2,3,4 (왼쪽 상단부터 시계방향으로 순번 지정)
  sns.countplot(x=name, data = new_tr)

  i = i+1
  plt.subplot(6,2,i)  ##2행2열, i = 1,2,3,4 (왼쪽 상단부터 시계방향으로 순번 지정)
  sns.countplot(x=name, data = new_test)

plt.show()
```

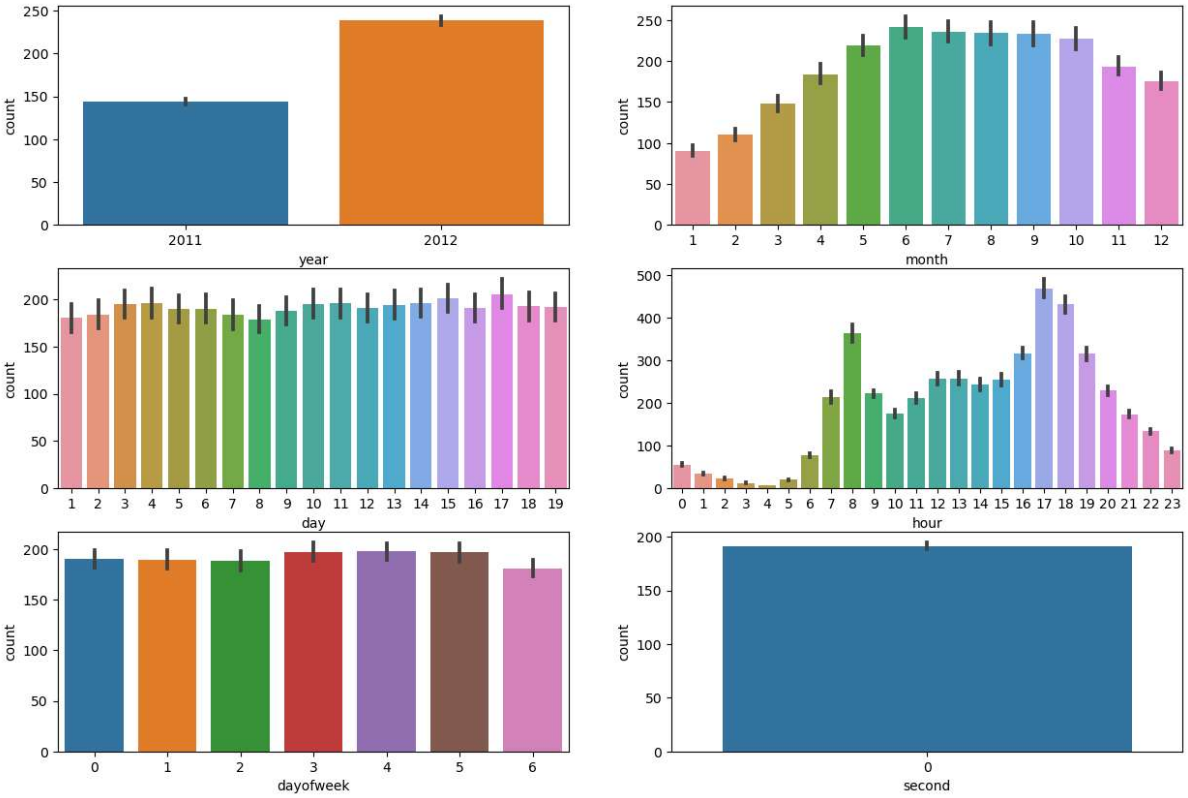


04 학습용 데이터 확인

In [10]:
```
datetime_names = ['year', 'month', 'day', 'hour', 'dayofweek', 'second']

i=0
plt.figure(figsize=(15,10))
for name in datetime_names:
    i = i + 1
    plt.subplot(3,2,i)
    sns.barplot(x=name, y='count', data=new_tr)

plt.show()
```



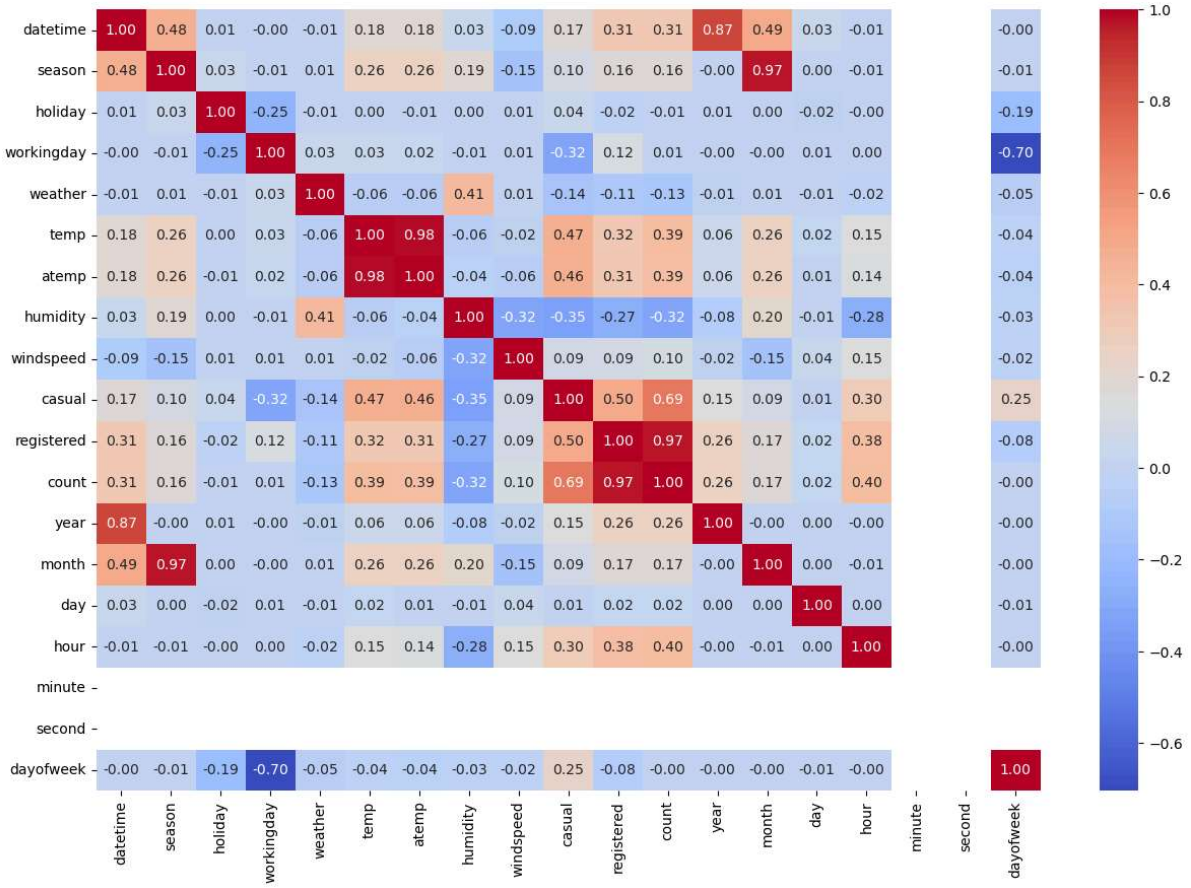
In [11]:
```
print(new_test.shape)
new_test[["datetime", "year", "month", "day", "hour", "minute", "second", "dayofweek
```

(6493, 16)

Out[11]:

	datetime	year	month	day	hour	minute	second	dayofweek
0	2011-01-20 00:00:00	2011	1	20	0	0	0	3
1	2011-01-20 01:00:00	2011	1	20	1	0	0	3
2	2011-01-20 02:00:00	2011	1	20	2	0	0	3
3	2011-01-20 03:00:00	2011	1	20	3	0	0	3
4	2011-01-20 04:00:00	2011	1	20	4	0	0	3

In [12]:
```
plt.figure(figsize=(15,10))
g = sns.heatmap(new_tr.corr(), annot=True, fmt=".2f", cmap="coolwarm")
```



05 feature 선택

In [13]:
```
feature_names = [ 'season', 'holiday', 'workingday', 'weather', 'temp',
 'atemp', 'humidity', 'windspeed', "year", "hour", "dayofweek"]  # 공통 변수
X_train = new_tr[feature_names]  # 학습용 데이터 변수 선택
print(X_train.head())
```

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	₩
0	1	0	0	1	9.84	14.395	81	0.0	
1	1	0	0	1	9.02	13.635	80	0.0	
2	1	0	0	1	9.02	13.635	80	0.0	
3	1	0	0	1	9.84	14.395	75	0.0	
4	1	0	0	1	9.84	14.395	75	0.0	

	year	hour	dayofweek
0	2011	0	5
1	2011	1	5
2	2011	2	5
3	2011	3	5
4	2011	4	5

In [14]:
```
label_name = 'count'                    # 렌탈 대수 (종속변수)
y_train = new_tr[label_name]            # 렌탈 대수 변수 값 선택
X_test = new_test[feature_names]        # 테스트 데이터의 변수 선택
X_test.head()                           # 테스트 데이터 선택된 내용 보기
```

Out[14]:

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

06 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)

선형회귀

In [15]:
```
from sklearn.linear_model import LinearRegression   # 선형회귀
```

In [16]:
```
seed = 37
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[16]:
```
▾ LinearRegression
LinearRegression()
```

In [17]:
```
predictions = model.predict(X_test)   # 예측(새로운 데이터로)
predictions
```

Out[17]:
```
array([-23.27179232, -20.84936197, -13.04580719, ..., 209.84495832,
       227.95174821, 217.86201958])
```

랜덤포레스트

In [18]:
```
from sklearn.ensemble import RandomForestRegressor   # 앙상블(의사결정트리 확장판)
```

In [19]:
```
seed = 37
model = RandomForestRegressor(n_jobs=-1, random_state=seed)  # 모델 객체 생성.
model
```

Out[19]:
```
▾ RandomForestRegressor
RandomForestRegressor(n_jobs=-1, random_state=37)
```

In [20]:
```
model.fit(X_train, y_train)           # 모델 학습(공부가 되었다.)
predictions = model.predict(X_test)   # 예측(새로운 데이터로)
predictions
```

Out[20]:
```
array([ 11.52      ,   4.58      ,   3.76      , ..., 102.1       ,
        99.91333333,  46.9       ])
```

In [21]:
```
# sub = pd.read_csv("sampleSubmission.csv")
sub['count'] = predictions
```

In [22]:
```
sub.head()
```

Out[22]:

	datetime	count
0	2011-01-20 00:00:00	11.52
1	2011-01-20 01:00:00	4.58
2	2011-01-20 02:00:00	3.76
3	2011-01-20 03:00:00	3.61
4	2011-01-20 04:00:00	2.97

In [23]:
```
# 처음 만는 제출용 csv 파일, 행번호를 없애기
sub.to_csv("firstsubmission.csv", index=False)
```

07 모델 평가

- 데이터 나누는 방법으로 기본으로 train_test_split 함수가 있음.
- 교차검증 반복 함수 cross_val_score
- cross_val_score(model, X, y, scoring=None, cv=None)
  model : 회귀 분석 모형
  X : 독립 변수 데이터
  y : 종속 변수 데이터
  scoring : 성능 검증에 사용할 함수 이름
  cv : 교차검증 생성기 객체 또는 숫자.
  None이면 KFold(3), 숫자 k이면 KFold(k)

In [24]:
```
from sklearn.model_selection import cross_val_score
```

In [25]:
```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import AdaBoostRegressor
```

선형회귀모델

In [32]:
```
model_linear = LinearRegression()
model_linear.fit(X_train, y_train)
score = cross_val_score(model_linear, X_train, y_train,
                        cv=5, scoring="neg_mean_squared_error")
print(score)
print("negative MSE 평균 :", score.mean())
```

```
[-10001.73269892 -14719.1205855  -13684.42876335 -33057.3894553
 -33971.64722717]
negative MSE 평균 : -21086.863746046292
```

의사결정트리

In [33]:
```
### 의사결정트리 decision tree, knn
```

In [34]:
```
model_decisionTree = DecisionTreeRegressor()
model_decisionTree.fit(X_train, y_train)
score = cross_val_score(model_decisionTree, X_train, y_train,
                        cv=5, scoring="neg_mean_squared_error")
print(score)
print("negative MSE 평균 :", score.mean())
```

```
[-8393.64841598 -5171.16123105 -8700.77273771 -8700.11391824
 -8638.2733119 ]
negative MSE 평균 : -7920.793922975105
```

KNN 모델

In [35]:
```
model_knn = KNeighborsRegressor()
model_knn.fit(X_train, y_train)
score = cross_val_score(model_knn, X_train, y_train,
                        cv=5, scoring="neg_mean_squared_error")
print(score)
print("negative MSE 평균 :", score.mean())
```

```
[-15451.48448118 -19539.02506201 -12562.05855765 -24291.4657051
 -28799.07480018]
negative MSE 평균 : -20128.621721223593
```

앙상블 모델(RandomForest)

In [36]:
```
model_RF = RandomForestRegressor()
model_RF.fit(X_train, y_train)
score = cross_val_score(model_RF, X_train, y_train,
                        cv=5, scoring="neg_mean_squared_error")
print(score)
print("negative MSE 평균 :", score.mean())
```

```
[-6976.28229286 -2798.97073096 -6354.29463881 -4678.830245
 -5827.829232  ]
negative MSE 평균 : -5327.241427926462
```

앙상블 모델(AdaBoostRegressor)

In [37]:
```
model_Ada = AdaBoostRegressor()
model_Ada.fit(X_train, y_train)
score = cross_val_score(model_Ada, X_train, y_train,
                        cv=5, scoring="neg_mean_squared_error")
print(score)
print("negative MSE 평균 :", score.mean())
```

```
[-18575.97877452 -11121.71801399 -14188.40102183 -15937.49003153
 -15974.69032062]
negative MSE 평균 : -15159.655632496597
```

In [38]:
```
import xgboost as xgb
```

In [39]:
```
data_dmatrix = xgb.DMatrix(data=X_train,label=y_train)
```

learning_rate: 학습률, 0~1사이의 값. 과적합을 방지하기 위한 단계 크기
max_depth: 각각의 나무 모델의 최대 깊이
subsample: 각 나무마다 사용하는 샘플 퍼센트, 낮은 값은 underfitting(과

소적합)을 야기할 수 있음.
colsample_bytree: 각 나무마다 사용하는 feature 퍼센트. High value can lead to overfitting.
n_estimators: 트리의 수(우리가 모델을 생성할)
objective:
   loss function(손실함수)결정.
   reg:linear   : for regression problems(회귀 문제),
   reg:logistic : for classification problems with only decision(분류 문제),
   binary:logistic for classification problems with probability.

In [40]:
```
# 기본 옵션 확인
xg_reg = xgb.XGBRegressor()
xg_reg
```

Out[40]:

XGBRegressor

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=
None,
             enable_categorical=False, eval_metric=None, feature_types=
None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=
None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
```

In [41]:
```
xg_reg = xgb.XGBRegressor(objective ='reg:linear',
                          colsample_bytree = 0.3,
                          learning_rate = 0.1,
                          max_depth = 3,
                          alpha = 0.1,
                          n_estimators = 100)  # n_estimators=100
xg_reg
```

Out[41]:

XGBRegressor

```
XGBRegressor(alpha=0.1, base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=0.3, device=None, early_stopping_rounds=N
one,
             enable_categorical=False, eval_metric=None, feature_types=
None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.1, max_bin=N
one,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=3, max_leaves=None,
```

In [43]:
```
xg_reg.fit(X_train,y_train)
score = cross_val_score(xg_reg, X_train, y_train,
                        cv=5, scoring="neg_mean_squared_error")
print(score)
print("negative MSE 평균 :", score.mean())
```

```
C:\Users\colab\AppData\Roaming\Python\Python311\site-packages\xgboost\core.py:160: UserWarning: [08:08:38] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-windows\src\objective\regression_obj.cu:209: reg:linear is now deprecated in favor of reg:squarederror.
  warnings.warn(smsg, UserWarning)
C:\Users\colab\AppData\Roaming\Python\Python311\site-packages\xgboost\core.py:160: UserWarning: [08:08:39] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-windows\src\objective\regression_obj.cu:209: reg:linear is now deprecated in favor of reg:squarederror.
  warnings.warn(smsg, UserWarning)
C:\Users\colab\AppData\Roaming\Python\Python311\site-packages\xgboost\core.py:160: UserWarning: [08:08:39] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-windows\src\objective\regression_obj.cu:209: reg:linear is now deprecated in favor of reg:squarederror.
  warnings.warn(smsg, UserWarning)
C:\Users\colab\AppData\Roaming\Python\Python311\site-packages\xgboost\core.py:160: UserWarning: [08:08:39] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-windows\src\objective\regression_obj.cu:209: reg:linear is now deprecated in favor of reg:squarederror.
  warnings.warn(smsg, UserWarning)
C:\Users\colab\AppData\Roaming\Python\Python311\site-packages\xgboost\core.py:160: UserWarning: [08:08:39] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-windows\src\objective\regression_obj.cu:209: reg:linear is now deprecated in favor of reg:squarederror.
  warnings.warn(smsg, UserWarning)
C:\Users\colab\AppData\Roaming\Python\Python311\site-packages\xgboost\core.py:160: UserWarning: [08:08:39] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-0b3782d1791676daf-1\xgboost\xgboost-ci-windows\src\objective\regression_obj.cu:209: reg:linear is now deprecated in favor of reg:squarederror.
  warnings.warn(smsg, UserWarning)
[ -5462.58038507  -6703.1085071   -5688.76188757 -17068.04195507
 -19588.38257509]
negative MSE 평균 : -10902.175061979642
```

최종 모델 선택

In [44]:
```
# 최종 모델 선택 및 제출
model_RF = RandomForestRegressor()
model_RF.fit(X_train, y_train)
predictions = model_RF.predict(X_test)
sub['count'] = predictions
sub.to_csv("submission_191013.csv", index=False)
```

REF

- cross_val_score:
  - https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html
- 모델 평가 scoring : https://scikit-learn.org/stable/modules/model_evaluation.html
- XGBOOST Documentation : https://xgboost.readthedocs.io/en/latest/index.html