## Kaggle 대회

- 대회 주제 : Bike Sharing Demand
- [https://www.kaggle.com/c/bike-sharing-demand (https://www.kaggle.com/c/bike-sharing-demand)](https://www.kaggle.com/c/bike-sharing-demand)

## 학습 내용

- 비선형 변환을 실습을 통해 알아본다.
- 다양한 모델의 교차 검증을 통해 비교해 본다.

In [1]:

```
from IPython.display import display, Image
```

## Data Fields

| 필드명 | 설명 |
|---|---|
| datetime | hourly date + timestamp |
| season | 1 = spring, 2 = summer, 3 = fall, 4 = winter<br>(봄[1], 여름[2], 가을[3], 겨울[4]) |
| holiday | whether the day is considered a holiday<br>(휴일인지 아닌지) |
| workingday | whether the day is neither a weekend nor holiday<br>(일하는 날인지 아닌지) |
| weather | 1: Clear, Few clouds, Partly cloudy, Partly cloudy<br>2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist<br>3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds<br>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 (시간대 별 자전거 빌린 대수) |

In [2]:

```
import pandas as pd
```

## 1-1 데이터 준비하기

In [114]:

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

train.shape, test.shape
```

Out[114]:

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

In [115]:

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

## 1-3 파생변수(더미변수) 생성

In [116]:

```
new_tr = train.copy()   # 데이터 백업
new_test = test.copy()
new_tr.columns
```

Out[116]:

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

In [117]:

```
## 더미변수, 파생변수 생성
new_tr['year'] = new_tr['datetime'].dt.year
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 # Monday=0, Sunday=6
```

In [118]:

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

Out[118]:

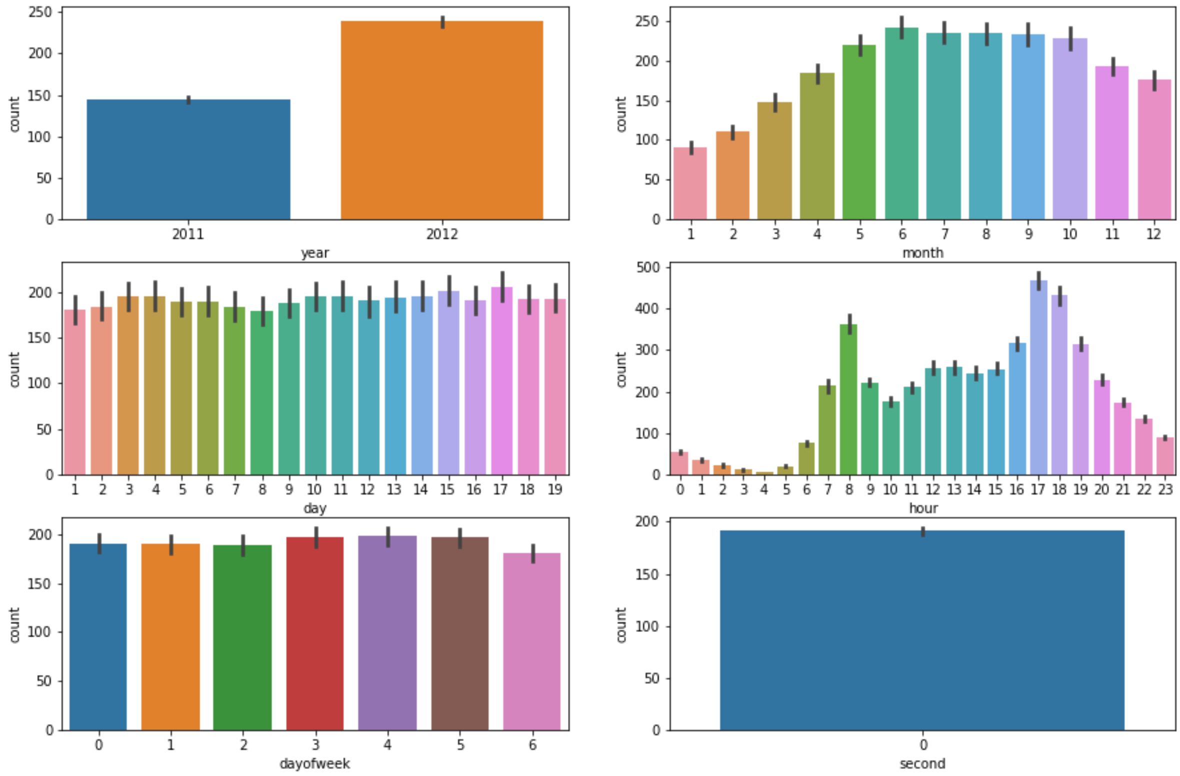
| | datetime | year | month | day | hour | dayofweek |
|---|---|---|---|---|---|---|
| **0** | 2011-01-20 00:00:00 | 2011 | 1 | 20 | 0 | 3 |
| **1** | 2011-01-20 01:00:00 | 2011 | 1 | 20 | 1 | 3 |
| **2** | 2011-01-20 02:00:00 | 2011 | 1 | 20 | 2 | 3 |
| **3** | 2011-01-20 03:00:00 | 2011 | 1 | 20 | 3 | 3 |
| **4** | 2011-01-20 04:00:00 | 2011 | 1 | 20 | 4 | 3 |
| **...** | ... | ... | ... | ... | ... | ... |
| **6488** | 2012-12-31 19:00:00 | 2012 | 12 | 31 | 19 | 0 |
| **6489** | 2012-12-31 20:00:00 | 2012 | 12 | 31 | 20 | 0 |
| **6490** | 2012-12-31 21:00:00 | 2012 | 12 | 31 | 21 | 0 |
| **6491** | 2012-12-31 22:00:00 | 2012 | 12 | 31 | 22 | 0 |
| **6492** | 2012-12-31 23:00:00 | 2012 | 12 | 31 | 23 | 0 |

6493 rows × 6 columns

## 변수 시각화

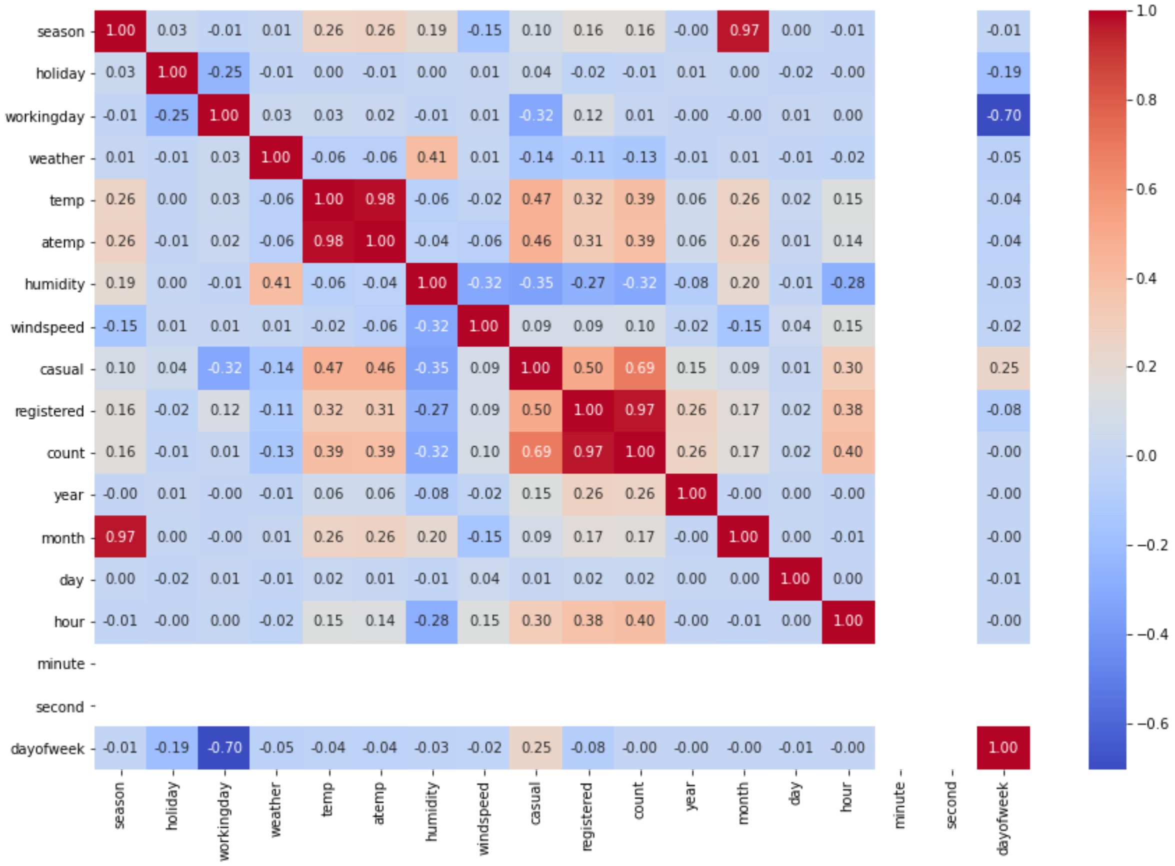
In [119]:

```
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 [120]:

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




In [121]:

```
print(new_tr.columns)
print(new_test.columns)
```

```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count',
       'year', 'month', 'day', 'hour', 'minute', 'second', 'dayofweek'],
      dtype='object')
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'year', 'month', 'day', 'hour',
       'minute', 'second', 'dayofweek'],
      dtype='object')
```

In [122]:

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

## 변수 선택 및 데이터 나누기

In [220]:

```
feature_names = [ 'season', 'holiday', 'workingday', 'weather',
                  'temp', 'atemp', 'humidity', 'windspeed',
                  "year", "hour", "dayofweek"]  # 공통 변수

X = new_tr[feature_names]     # 학습용 데이터 변수 선택
y = new_tr['count']        # 렌탈 대수 변수 값 선택

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=77,
                                                    test_size=0.2)
```

In [221]:

```
X_test_l = new_test[feature_names]    # 테스트 데이터의 변수 선택
```

### 1-4 모델 만들기 및 제출

In [222]:

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
```

In [223]:

```
model = LinearRegression()
model.fit(X_train, y_train)
pred_lr = model.predict(X_test_l)   # 예측(새로운 데이터로)
pred_lr
```

Out[223]:

```
array([-26.65558781, -23.5682814 , -15.75475438, ..., 212.69553441,
       230.79247316, 220.44695327])
```

### 의사 결정 트리 모델 만들기

In [224]:

```
model = DecisionTreeRegressor()  # 모델 객체 생성.
model.fit(X_train, y_train)
pred_tree = model.predict(X_test_l)   # 예측(새로운 데이터로)
pred_tree
```

Out[224]:

```
array([ 19.,   4.,   3., ..., 120., 102.,  46.])
```

In [225]:

```
seed = 37
model = RandomForestRegressor(n_jobs=-1, random_state=seed)  # 모델 객체 생성.
model.fit(X_train, y_train)           # 모델 학습(공부가 되었다.)
pred_rf = model.predict(X_test_l)     # 예측(새로운 데이터로)
pred_rf
```

Out[225]:

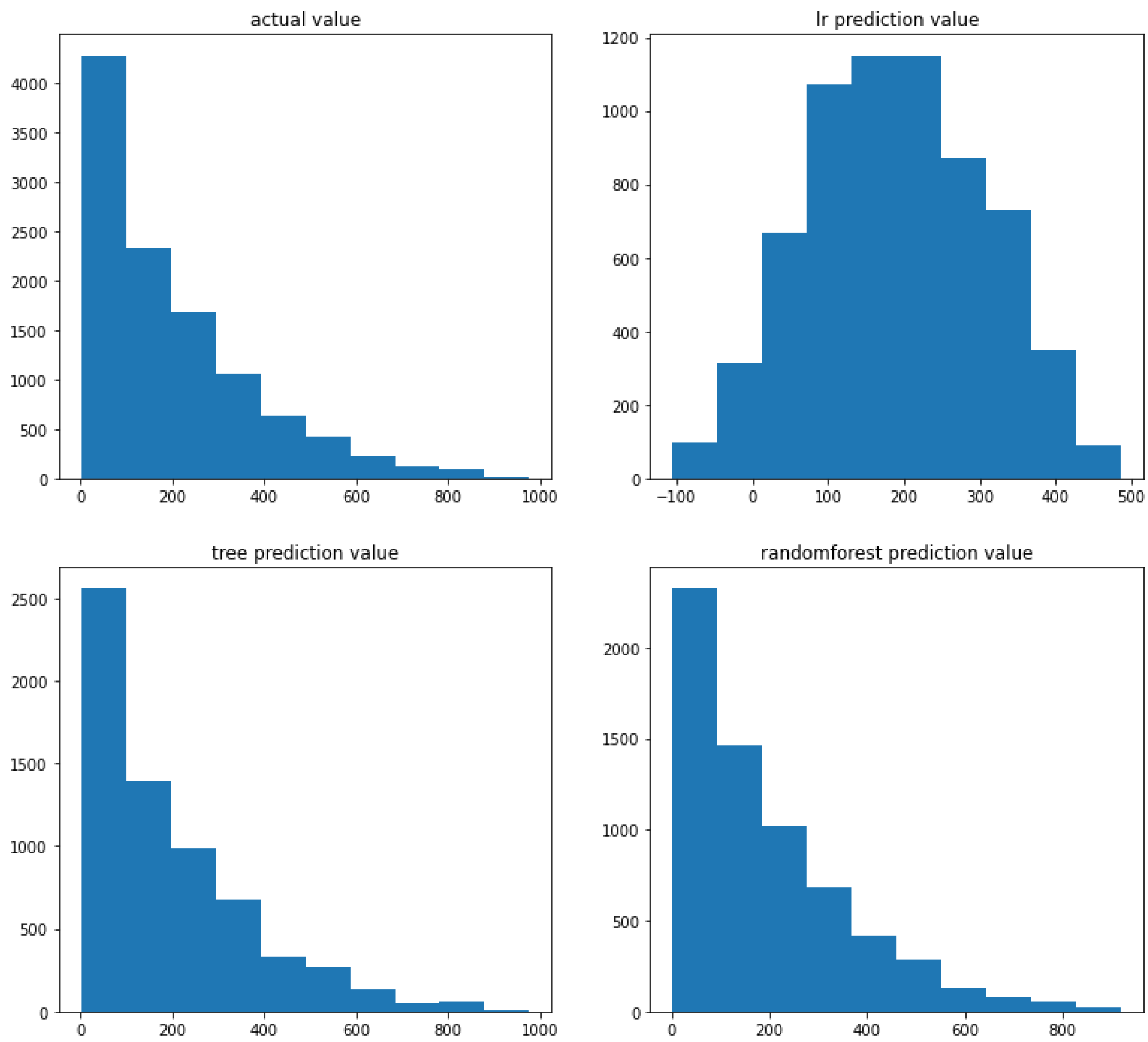
```
array([12.39      ,  4.85      ,  4.28      , ..., 97.375     ,
       99.41333333, 47.53      ])
```

In [226]:

```
plt.figure(figsize=(13,12))
plt.subplot(2,2,1)
plt.hist(new_tr['count'])  #  학습용 데이터 자전거 렌탈 대수
plt.title("actual value")

plt.subplot(2,2,2)
plt.hist(pred_lr)          #  선형 회귀로 활용한 예측한 대수
plt.title("lr prediction value")

plt.subplot(2,2,3)
plt.hist(pred_tree)        #  의사결정트리로 활용한 예측한 대수
plt.title("tree prediction value")

plt.subplot(2,2,4)
plt.hist(pred_rf)          #  랜덤포레스트를 활용한 예측한 대수
plt.title("randomforest prediction value")
```

Out[226]:

```
Text(0.5, 1.0, 'randomforest prediction value')
```




## 그렇다면 어떤 모델이 나은지 어떻게 판단할 수 있는가?

## 1-5 모델 평가 및 제출

- 데이터 나누는 방법으로 기본으로 train_test_split 함수가 있음.
- 교차검증 반복 함수 cross_val_score
  - cross-validation에 의해 점수를 평가한다.
- 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 [227]:

```
from sklearn.model_selection import cross_val_score
```

In [228]:

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
```

In [229]:

```
import numpy as np
```

In [231]:

```
def model_val(model_name):
    model = model_name
    model.fit(X_train, y_train)
    score = cross_val_score(model, X_test, y_test,
                     cv=5, scoring="neg_mean_squared_error")
    print("MSE :", score)
    m_score = np.abs(score.mean())  # 절대값
    print("MSE 평균(cv=5) : ", m_score)
    return m_score
```

In [232]:

```
model_list = ["LinearRegression",  "DecisionTreeRegressor",
              "KNeighborsRegressor", "RandomForestRegressor",
              "AdaBoostRegressor"]

model_score = []
```

## 선형회귀

In [233]:

```
model = LinearRegression()
score = model_val(model)
model_score.append(score)
```

```
MSE : [-22540.87373402 -19094.68032085 -20036.84934944 -20196.9630662
 -17713.50855232]
MSE 평균(cv=5) :  19916.575004567534
```

## 의사결정트리 decision tree, knn

In [234]:

```
model = DecisionTreeRegressor()
score = model_val(model)
model_score.append(score)
```

```
MSE : [-6002.26490826 -3807.30504587 -7884.62155963 -5713.82298851
 -6912.06896552]
MSE 평균(cv=5) :  6064.016693556892
```

In [235]:

```
model = KNeighborsRegressor()
score = model_val(model)
model_score.append(score)
```

```
MSE : [-19682.97458716 -17341.75192661 -17639.02293578 -16804.47521839
 -16124.67577011]
MSE 평균(cv=5) :  17518.580087609407
```

## 앙상블 RandomForest, Ada

In [236]:

```
model = RandomForestRegressor()
score = model_val(model)
model_score.append(score)
```

```
MSE : [-3999.89238424 -2837.97516268 -4429.42613691 -2935.73099296
 -2786.79624811]
MSE 평균(cv=5) :  3397.9641849816207
```

In [237]:

```
model = AdaBoostRegressor()
score = model_val(model)
model_score.append(score)
```

```
MSE : [-12128.7338982   -9981.58719735 -11694.52560442 -11668.47771995
  -9504.15457325]
MSE 평균(cv=5) :  10995.495798633263
```

In [238]:

```
import pandas as pd
dat1 = pd.DataFrame( {'model_name':model_list, 'score': model_score })
dat1
```

Out[238]:

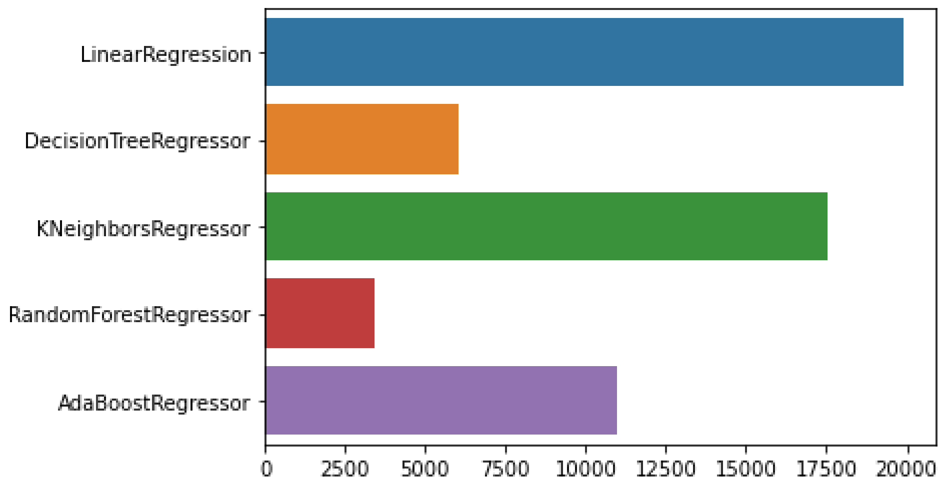
| | model_name | score |
|---|---|---|
| **0** | LinearRegression | 19916.575005 |
| **1** | DecisionTreeRegressor | 6064.016694 |
| **2** | KNeighborsRegressor | 17518.580088 |
| **3** | RandomForestRegressor | 3397.964185 |
| **4** | AdaBoostRegressor | 10995.495799 |

In [239]:

```
# 모델과 스코어 확인하기
sns.barplot(x=model_score , y=model_list, data=dat)
```

Out[239]:

<AxesSubplot:>




In [240]:

```
new_tr['log_count'] = np.log1p(new_tr['count'] )
```

In [251]:

```python
feature_names = [ 'season', 'holiday', 'workingday', 'weather',
                  'temp', 'atemp', 'humidity', 'windspeed',
                  "year", "hour", "dayofweek"]  # 공통 변수

X = new_tr[feature_names]     # 학습용 데이터 변수 선택
y = new_tr['log_count']        # 렌탈 대수 변수 값 선택

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=77,
                                                    test_size=0.2)
```

In [252]:

```python
model_name = ["LinearRegression",  "DecisionTreeRegressor",
              "KNeighborsRegressor", "RandomForestRegressor",  "AdaBoostRegressor"]

model_list = [LinearRegression(),  DecisionTreeRegressor(),
              KNeighborsRegressor(), RandomForestRegressor(),  AdaBoostRegressor()]

model_score = []

for idx, model_one in enumerate(model_list):
    score = model_val(model_one)
    print("Model Name :", model_name[idx])
    model_score.append(score)
    print()
```

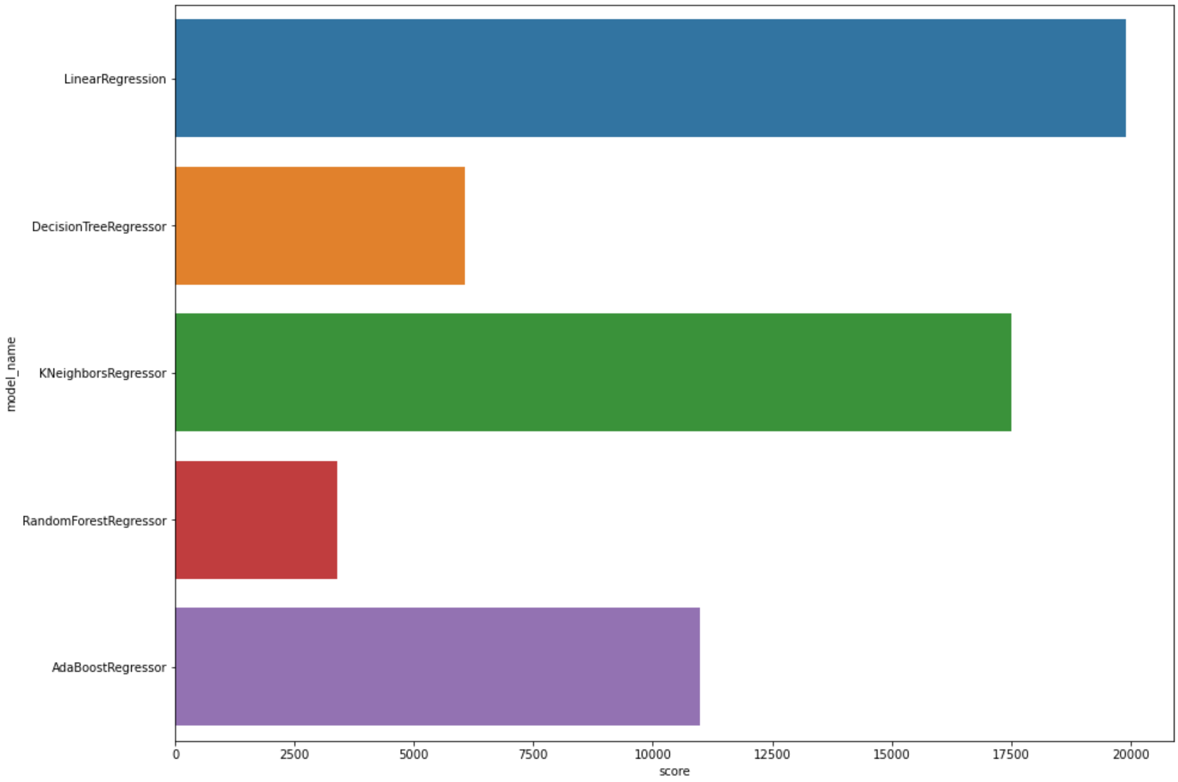
```
MSE : [-0.97170551 -1.11464491 -1.1644267  -1.05059175 -1.0600164 ]
MSE 평균(cv=5) :  1.0722770551415117
Model Name : LinearRegression

MSE : [-0.29831207 -0.32531271 -0.26158545 -0.28370707 -0.25677382]
MSE 평균(cv=5) :  0.28513822431278973
Model Name : DecisionTreeRegressor

MSE : [-0.7409194  -0.82228105 -0.88626807 -0.70829583 -0.91391292]
MSE 평균(cv=5) :  0.8143354543600811
Model Name : KNeighborsRegressor

MSE : [-0.17161291 -0.14505555 -0.15245613 -0.15230711 -0.14547804]
MSE 평균(cv=5) :  0.1533819463232813
Model Name : RandomForestRegressor

MSE : [-0.40948858 -0.37513755 -0.39424902 -0.39493443 -0.39940738]
MSE 평균(cv=5) :  0.39464339253984515
Model Name : AdaBoostRegressor
```

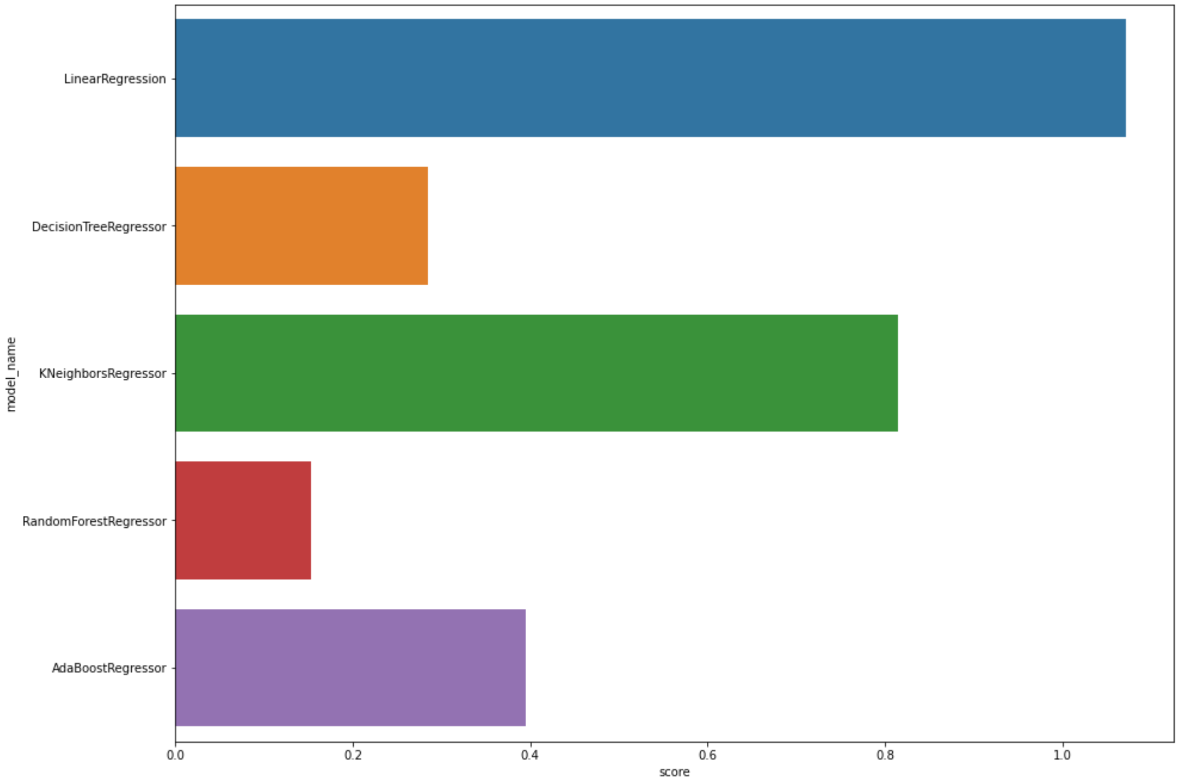
In [255]:

```
import pandas as pd
dat2 = pd.DataFrame( {'model_name':model_name, 'score': model_score })

# 모델과 스코어 확인하기
plt.figure(figsize=(15,25))
plt.subplot(2,1,1)
sns.barplot(x="score" , y="model_name", data=dat1)

plt.subplot(2,1,2)
sns.barplot(x="score" , y="model_name", data=dat2)
```

Out[255]:

```
<AxesSubplot:xlabel='score', ylabel='model_name'>
```




## 머신러닝 대표적 앙상블 중의 하나 XGBOOSTING 기법 사용해보기

In [256]:

```
import xgboost as xgb
```

In [257]:

```
# 기본 옵션 확인
xg_reg = xgb.XGBRegressor()
xg_reg
```

Out[257]:

```
XGBRegressor(base_score=None, booster=None, colsample_bylevel=None,
             colsample_bynode=None, colsample_bytree=None, gamma=None,
             gpu_id=None, importance_type='gain', interaction_constraints=None,
             learning_rate=None, max_delta_step=None, max_depth=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             random_state=None, reg_alpha=None, reg_lambda=None,
             scale_pos_weight=None, subsample=None, tree_method=None,
             validate_parameters=None, verbosity=None)
```

## 사이킷런 기반 파라미터 설명

- 기본값은 사이킷런 기반 기본값

| 파라미터명 | 설명 | 사이킷런 기본값(파이썬기반) |
| ---: | --- | ---: |
| learning_rate(or eta) | 0~1사이의 값. 과적합을 방지하기 위한 학습률 값 | 기본값 : 0.1(0.3) |
| n_estimators(or num_boost_rounds) | **트리의 수** | 기본값 100(10) |
| max_depth | 각각의 나무 **모델의 최대 깊이** | 기본값 3(6) |
| subsample | 각 나무마다 사용하는 데이터 샘플 비율<br>낮은 값은 underfitting(과소적합)을 야기할 수 있음. | 기본값 : 1 |
| colsample_bytree | 각 나무마다 사용하는 **feature 비율**.<br>High value can lead to overfitting. | 기본값 : 1 |
| reg_alpha(or alpha) | L1 규제에 대한 항<br>피처가 많을 수록 적용을 검토한다. | 기본값 : 0 |
| reg_lambda(or lambda) | L2 규제의 적용 값. 피처의 개수가 많을 경우 적용 검토 | 기본값 : 1 |
| scale_pos_weight | 불균형 데이터셋의 균형 유지 | 기본값 : 1 |

## 학습 태스크 파라미터

| 파라미터명 | 설명 | 사이킷런 기본값(파이썬기반) |
| ---: | --- | ---: |
| objective(목적함수) | **reg:linear** for regression problems(회귀 문제),<br>**reg:logistic** for classification problems with only decision(분류 문제),<br>**binary:logistic** for classification problems with probability.(이진 분류) | |

In [259]:

```
xg_reg = xgb.XGBRegressor(objective ='reg:linear',
            colsample_bytree = 0.3, # 각나무마다 사용하는 feature 비율
            learning_rate = 0.05,
            max_depth = 5,
            alpha = 0.1,
            n_estimators = 3000)

xg_reg
```

Out[259]:

```
XGBRegressor(alpha=0.1, base_score=None, booster=None, colsample_bylevel=None,
             colsample_bynode=None, colsample_bytree=0.3, gamma=None,
             gpu_id=None, importance_type='gain', interaction_constraints=None,
             learning_rate=0.05, max_delta_step=None, max_depth=5,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=3000, n_jobs=None, num_parallel_tree=None,
             objective='reg:linear', random_state=None, reg_alpha=None,
             reg_lambda=None, scale_pos_weight=None, subsample=None,
             tree_method=None, validate_parameters=None, verbosity=None)
```

## 학습

In [263]:

```
%%time

score = model_val(xg_reg)
```

```
[01:46:31] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/sr
c/objective/regression_obj.cu:171: reg:linear is now deprecated in favor of reg:squa
rederror.
[01:46:39] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/sr
c/objective/regression_obj.cu:171: reg:linear is now deprecated in favor of reg:squa
rederror.
[01:46:43] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/sr
c/objective/regression_obj.cu:171: reg:linear is now deprecated in favor of reg:squa
rederror.
[01:46:46] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/sr
c/objective/regression_obj.cu:171: reg:linear is now deprecated in favor of reg:squa
rederror.
[01:46:50] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/sr
c/objective/regression_obj.cu:171: reg:linear is now deprecated in favor of reg:squa
rederror.
[01:46:54] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/sr
c/objective/regression_obj.cu:171: reg:linear is now deprecated in favor of reg:squa
rederror.
MSE : [-0.1163149  -0.11471285 -0.13099835 -0.13975009 -0.12373197]
MSE 평균(cv=5) :  0.12510163295532067
Wall time: 27.2 s
```

In [264]:

```
dat2.loc[5] = ['xgboost', score]
```

In [265]:

```
dat2
```

Out[265]:

| | model_name | score |
|---|---|---|
| **0** | LinearRegression | 1.072277 |
| **1** | DecisionTreeRegressor | 0.285138 |
| **2** | KNeighborsRegressor | 0.814335 |
| **3** | RandomForestRegressor | 0.153382 |
| **4** | AdaBoostRegressor | 0.394643 |
| **5** | xgboost | 0.125102 |

In [266]:

```
# 최종 모델 선택 및 제출
model_last = xgb.XGBRegressor(objective ='reg:linear',
            colsample_bytree = 0.3, # 각나무마다 사용하는 feature 비율
            learning_rate = 0.05,
            max_depth = 5,
            alpha = 0.1,
            n_estimators = 3000)

model_last.fit(X_train, y_train)
pred = model_last.predict(X_test_l)

sub['count'] = np.expm1(pred)
sub.to_csv("submission_last.csv", index=False)
```

```
[01:47:07] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/sr
c/objective/regression_obj.cu:171: reg:linear is now deprecated in favor of reg:squa
rederror.
```

# REF

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

## History

- 2020-06 update
- 2021-10 update v12