Bike Kaggle 데이터 분석

Data Fields

- datetime - hourly date + timestamp (시간 데이터 + timestamp)
- season - 1 = spring, 2 = summer, 3 = fall, 4 = winter (봄[1], 여름[2], 가을[3], 겨울[4])
- holiday - whether the day is considered a holiday (휴일인지 아닌지)
- workingday - whether the day is neither a weekend nor holiday(일하는 날인지 아닌지)
- weather - 날씨에 따른 값
  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 (바람 속도)
- casual - number of non-registered user rentals initiated (등록되지 않은 사용자의 빌린 자전거 대수)
- egistered - number of registered user rentals initiated (등록된 사용자의 빌린 자전거 대수)
- count - number of total rentals (자전거 빌린 총 대수 - 시간별)

In [19]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

01 데이터 준비

In [20]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [21]:

```
## train 데이터 셋 , test 데이터 셋
## train 은 학습을 위한 데이터 셋
## test 은 예측을 위한 데이터 셋(평가)
## parse_dates : datetime 컬럼을 시간형으로 불러올 수 있음
train = pd.read_csv("bike/train.csv", parse_dates=['datetime'])
test = pd.read_csv("bike/test.csv", parse_dates=['datetime'])
sub = pd.read_csv("bike/sampleSubmission.csv")
```

In [22]:

```
print(train.shape)   # : 행과 열 갯수 확인
print(test.shape)
print()
print(train.columns)
print(test.columns)
```

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

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

02 유의한 파생 변수 생성

In [23]:

```
new_tr = train.copy()
new_test = test.copy()
```

In [24]:

```
## 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
```

In [25]:

```
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']   ]
```

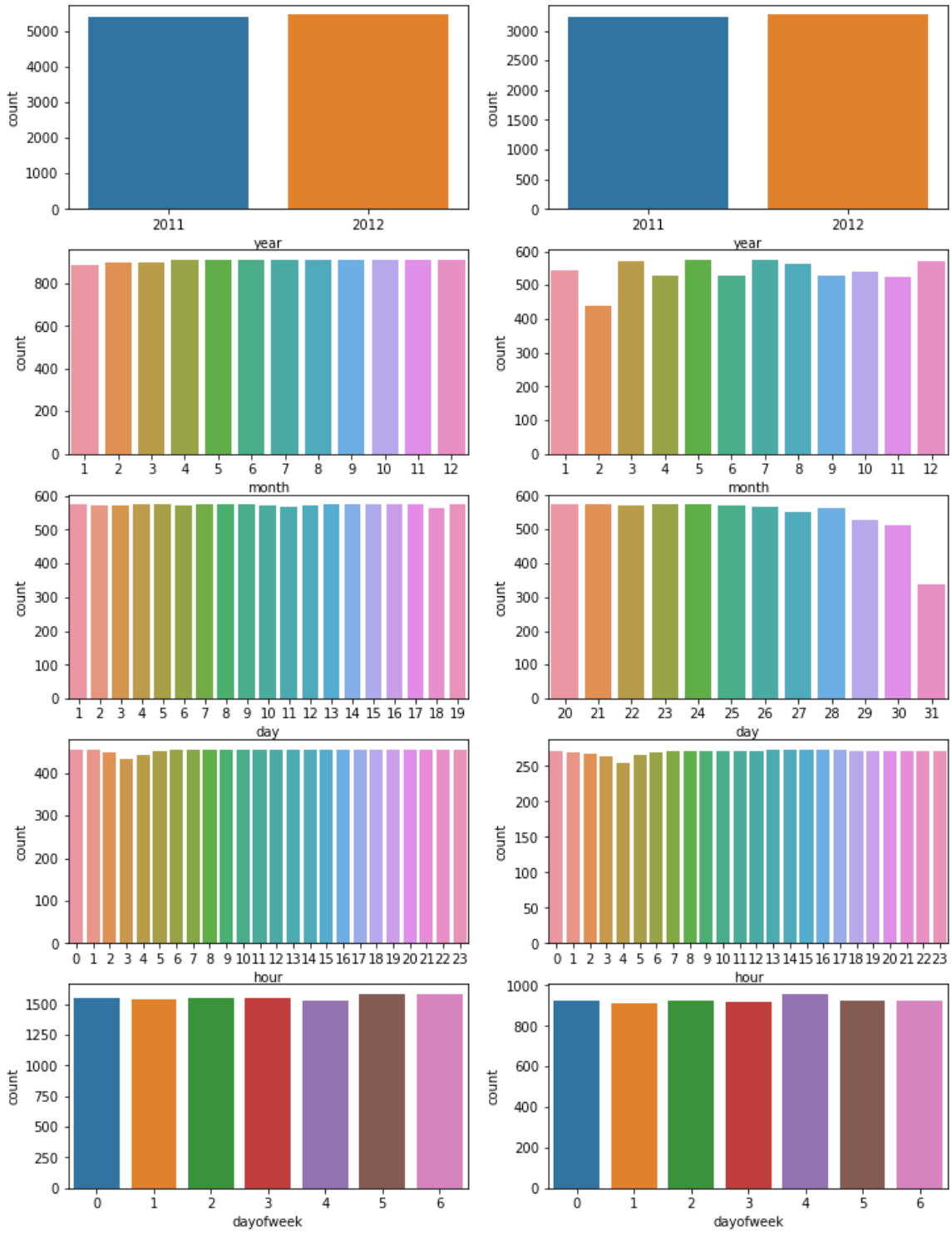
Out[25]:

	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

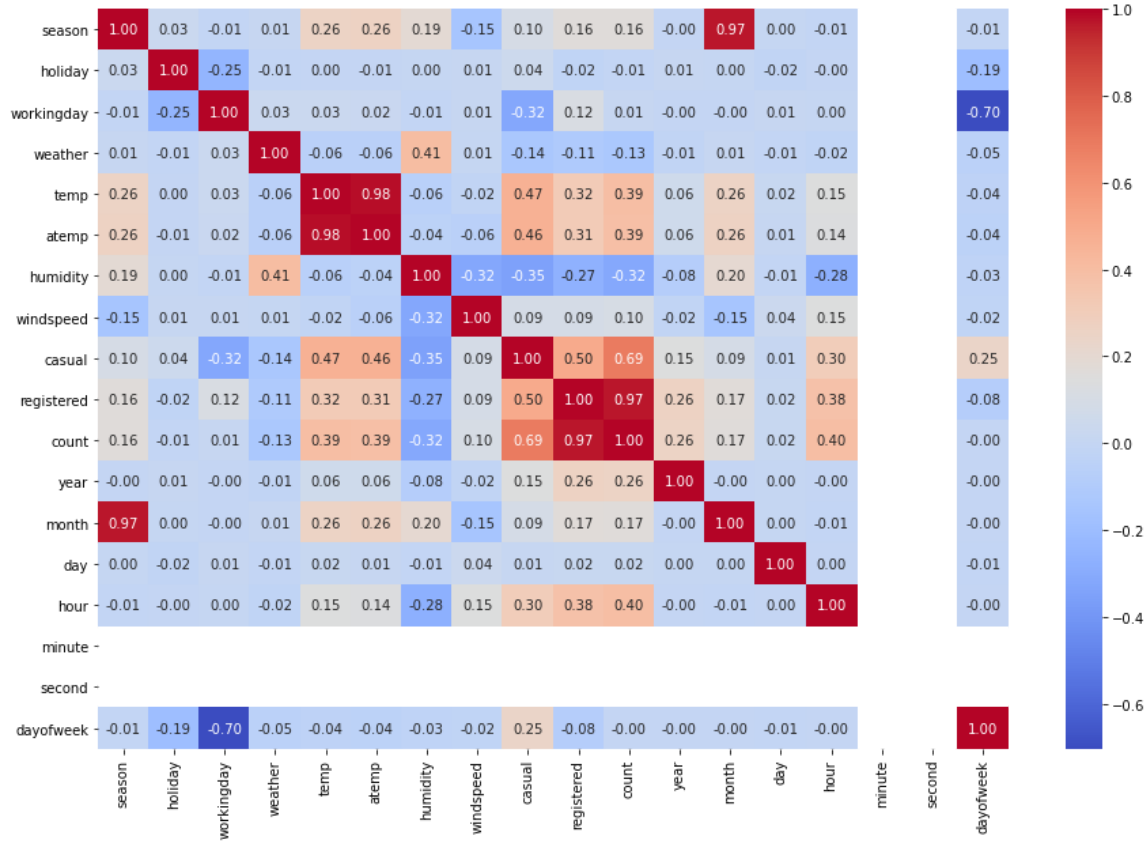
In [26]:

```
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(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(name, data = new_test)

plt.show()
```


count
year
2011
2012
month
day
hour
dayofweek

In [27]:

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



In [28]:

```
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 [29]:

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

In [30]:

```
label_name = 'log_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[30]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	year	hour	d
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	

03 모델 생성

In [31]:

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

In [33]:

```
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
import lightgbm as lgb
import numpy as np
import time
```

In [34]:

```
### Model 01. RandomForest
### Model 02. Xgboost
### Model 03. LightGBM Model 1
### Model 04. LightGBM Model 2

model_list = ["RandomForestRegressor",  "xgb_basic", "lightgbm-model1", "lightgbm-model2"]
model_score = []
model_time = []
```

In [35]:

```
now_time = time.time()

model_RF = RandomForestRegressor(random_state=30)
model_RF.fit(X_train, y_train)
score = cross_val_score(model_RF, X_train, y_train, cv=5, scoring="neg_mean_squared_error")
m_score = np.abs(score.mean())  # 절대값
model_score.append(m_score)

pro_time = time.time() - now_time
model_time.append(pro_time)

print(pro_time)  # 걸린 시간
print("RandomForestRegressor Score : {}".format(m_score))  # 점수
```

```
17.299612045288086
RandomForestRegressor Score : 0.2325269392461268
```

In [36]:

```
data_dmatrix = xgb.DMatrix(data=X_train,label=y_train)
```

- learning_rate: 0~1사이의 값. 과적합을 방지하기 위한 단계 크기
- max_depth: 각각의 나무 **모델의 최대 깊이**
- subsample: 각 나무마다 사용하는 샘플 퍼센트, 낮은 값은 underfitting(과소적합)을 야기할 수 있음.
- colsample_bytree: 각 나무마다 사용하는 feature 퍼센트. High value can lead to overfitting.
- n_estimators: 트리의 수(우리가 모델을 생성할)
- loss function(손실함수)결정.
- objective(목적함수)
  - reg:linear for regression problems(회귀 문제),
  - reg:logistic for classification problems with only decision(분류 문제),
  - binary:logistic for classification problems with probability.
- alpha : L1 규제에 대한 항

In [37]:

```
now_time = time.time()

xg_reg = xgb.XGBRegressor(objective ='reg:linear',
            colsample_bytree = 0.3, # 각나무마다 사용하는 feature 비율
            learning_rate = 0.1,
            max_depth = 3,
            alpha = 0.1,
            n_estimators = 100)  # n_estimators=100

xg_reg.fit(X_train, y_train)
score = cross_val_score(xg_reg, X_train, y_train, cv=5, scoring="neg_mean_squared_error")
m_score = np.abs(score.mean())  # 절대값
model_score.append(m_score)

pro_time = time.time() - now_time
model_time.append(pro_time)

print(pro_time)  # 걸린 시간
print("Xgboost Score : {}".format(m_score))  # 점수
```

```
[00:08:07] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.2.0/s
rc/objective/regression_obj.cu:174: reg:linear is now deprecated in favor of reg:s
quarederror.
[00:08:07] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.2.0/s
rc/objective/regression_obj.cu:174: reg:linear is now deprecated in favor of reg:s
quarederror.
[00:08:07] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.2.0/s
rc/objective/regression_obj.cu:174: reg:linear is now deprecated in favor of reg:s
quarederror.
[00:08:07] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.2.0/s
rc/objective/regression_obj.cu:174: reg:linear is now deprecated in favor of reg:s
quarederror.
[00:08:07] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.2.0/s
rc/objective/regression_obj.cu:174: reg:linear is now deprecated in favor of reg:s
quarederror.
[00:08:07] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.2.0/s
rc/objective/regression_obj.cu:174: reg:linear is now deprecated in favor of reg:s
quarederror.
[00:08:07] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.2.0/s
rc/objective/regression_obj.cu:174: reg:linear is now deprecated in favor of reg:s
quarederror.
[00:08:07] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.2.0/s
rc/objective/regression_obj.cu:174: reg:linear is now deprecated in favor of reg:s
quarederror.
[00:08:07] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.2.0/s
rc/objective/regression_obj.cu:174: reg:linear is now deprecated in favor of reg:s
quarederror.
[00:08:07] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.2.0/s
rc/objective/regression_obj.cu:174: reg:linear is now deprecated in favor of reg:s
quarederror.
[00:08:07] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.2.0/s
rc/objective/regression_obj.cu:174: reg:linear is now deprecated in favor of reg:s
quarederror.
[00:08:07] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.2.0/s
rc/objective/regression_obj.cu:174: reg:linear is now deprecated in favor of reg:s
quarederror.
0.7440195083618164
Xgboost Score : 0.3994248678485513
```

LightGBM Model

In [38]:

```
import lightgbm as lgb
```

In [39]:

```
now_time = time.time()

m_lgbm1 = lgb.LGBMRegressor()
m_lgbm1.fit(X_train, y_train)
score = cross_val_score(m_lgbm1, X_train, y_train,
                        cv=5, scoring="neg_mean_squared_error")

m_score = np.abs(score.mean())  # 절대값
model_score.append(m_score)

pro_time = time.time() - now_time
model_time.append(pro_time)

print(pro_time)  # 걸린 시간
print("LightGBM Score : {}".format(m_score))  # 점수
```

```
0.8918182849884033
LightGBM Score : 0.1835259588783745
```

LightGBM Model2

In [40]:

```
hyperparameters = {'boosting_type': 'gbdt',
                   'colsample_bytree': 0.7250136792694301,
                   'is_unbalance': False,
                   'learning_rate': 0.013227664889528229,
                   'min_child_samples': 20,
                   'num_leaves': 56,
                   'reg_alpha': 0.7543896477745794,
                   'reg_lambda': 0.07152751159655985,
                   'subsample_for_bin': 240000,
                   'subsample': 0.5233384321711397,
                   'n_estimators': 1093}
```

In [41]:

```
now_time = time.time()

m_lgbm2 = lgb.LGBMRegressor(**hyperparameters)
m_lgbm2.fit(X_train, y_train)
score = cross_val_score(m_lgbm2, X_train, y_train,
                        cv=5, scoring="neg_mean_squared_error")

m_score = np.abs(score.mean())  # 절대값
model_score.append(m_score)

pro_time = time.time() - now_time
model_time.append(pro_time)

print(pro_time)  # 걸린 시간
print("LightGBM Score : {}".format(m_score))  # 점수
```

```
13.264843702316284
LightGBM Score : 0.1731408336725748
```

In [42]:

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

Out[42]:

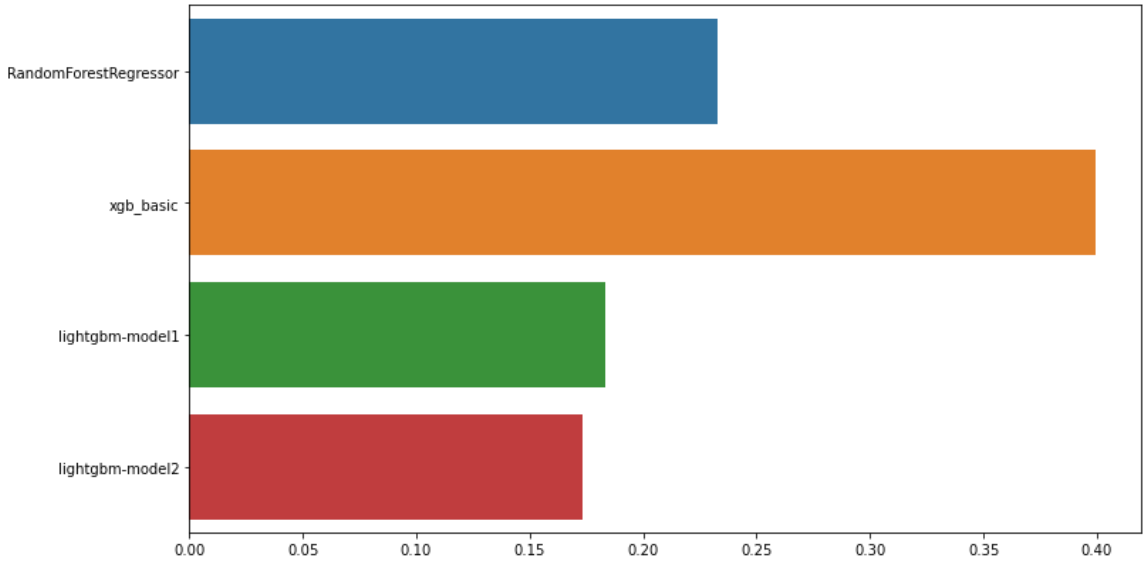
	model_name	score	time
0	RandomForestRegressor	0.232527	17.299612
1	xgb_basic	0.399425	0.744020
2	lightgbm-model1	0.183526	0.891818
3	lightgbm-model2	0.173141	13.264844

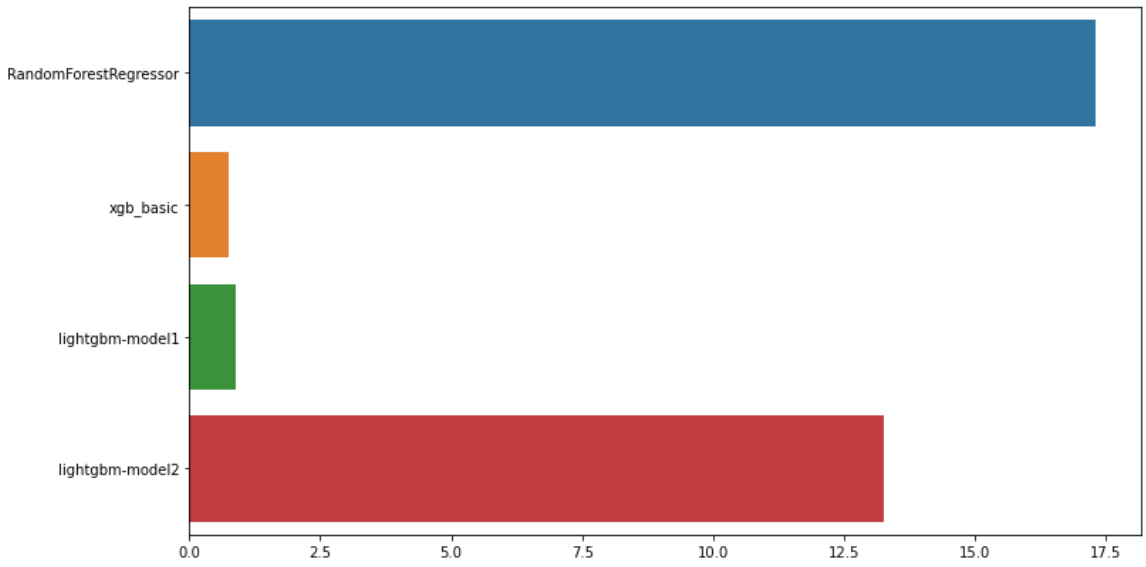
In [43]:

```
plt.figure(figsize=(12,15))  ##전체 그래프 크기 지정

plt.subplot(2,1,1)
sns.barplot(x=model_score , y=model_list, data=dat)

plt.subplot(2,1,2)
sns.barplot(x=model_time , y=model_list, data=dat)
```

Out[43]:

<matplotlib.axes._subplots.AxesSubplot at 0x192a903dee0>





04 최종 모델

In [45]:

```
# 최종 모델 선택 및 제출
m_lgbm2 = lgb.LGBMRegressor(**hyperparameters)
m_lgbm2.fit(X_train, y_train)

pred = m_lgbm2.predict(X_test)
sub = pd.read_csv("bike/sampleSubmission.csv")
sub['count'] = np.expm1(pred)
sub.to_csv("sub_v04_lgbm_winadd.csv", index=False)
```

05 모델 합치기

In [46]:

```
# 최종 모델 선택 및 제출
model_RF = RandomForestRegressor(random_state=30)
model_RF.fit(X_train, y_train)
pred1 = model_RF.predict(X_test)

# 최종 모델 선택 및 제출
m_lgbm2 = lgb.LGBMRegressor(**hyperparameters)
m_lgbm2.fit(X_train, y_train)
pred2 = m_lgbm2.predict(X_test)

sub['count'] = np.expm1(pred1) * 0.2 + np.expm1(pred2) * 0.8
sub.to_csv("sub_v07_lgbm_rf_add.csv", index=False)
```

06 casual, registered를 예측후, 실행

In [47]:

```
new_tr.columns
```

Out[47]:

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

In [48]:

```
new_tr = new_tr.drop(['log_count'], axis=1)
for col in ['casual', 'registered', 'count']:
    new_tr['%s_log' % col] = np.log1p(new_tr[col])
```

In [49]:

```
new_tr.columns
```

Out[49]:

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

In [50]:

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

In [51]:

```
label_name = 'casual_log'                # 렌탈 대수 (종속변수)
X_train = new_tr[feature_names]
y_train = new_tr[label_name]         # 렌탈 대수 변수 값 선택
X_test = new_test[feature_names]     # 테스트 데이터의 변수 선택
```

In [52]:

```
# 최종 모델 선택 및 제출
m_lgbm2 = lgb.LGBMRegressor(**hyperparameters)
m_lgbm2.fit(X_train, y_train)

pred1 = m_lgbm2.predict(X_test)
```

In [53]:

```
label_name = 'registered_log'                # 렌탈 대수 (종속변수)
X_train = new_tr[feature_names]
y_train = new_tr[label_name]         # 렌탈 대수 변수 값 선택
X_test = new_test[feature_names]     # 테스트 데이터의 변수 선택
```

In [54]:

```
# 최종 모델 선택 및 제출
m_lgbm2 = lgb.LGBMRegressor(**hyperparameters)
m_lgbm2.fit(X_train, y_train)

pred2 = m_lgbm2.predict(X_test)
```

In [56]:

```
sub = pd.read_csv("bike/sampleSubmission.csv")
sub['count'] = np.expm1(pred1) + np.expm1(pred2)
sub.to_csv("sub_v08_lgbm3.csv", index=False)
```

0.37136(88)

07. 두개의 다른 모델 합치기

In [57]:

```
# 최종 모델 선택 및 제출
model_RF = RandomForestRegressor(random_state=30)
model_RF.fit(X_train, y_train)
pred_rf = model_RF.predict(X_test)

# 6번 모델
lgbm_c_r = np.expm1(pred1) + np.expm1(pred2)
sub['count'] = np.expm1(pred_rf) * 0.2 +  lgbm_c_r * 0.8
sub.to_csv("sub_v09_lgbm_rf_add.csv", index=False)
```

0.37198

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)

In [ ]:

```

```