Kaggle 대회

- 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 (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	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 (시간대 별 자전거 빌린 대수)

01 데이터 준비

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
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[2]:

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

02 유의한 파생 변수 생성

In [3]:

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

In [4]:

```
## 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 [5]:

```
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']   ]
```

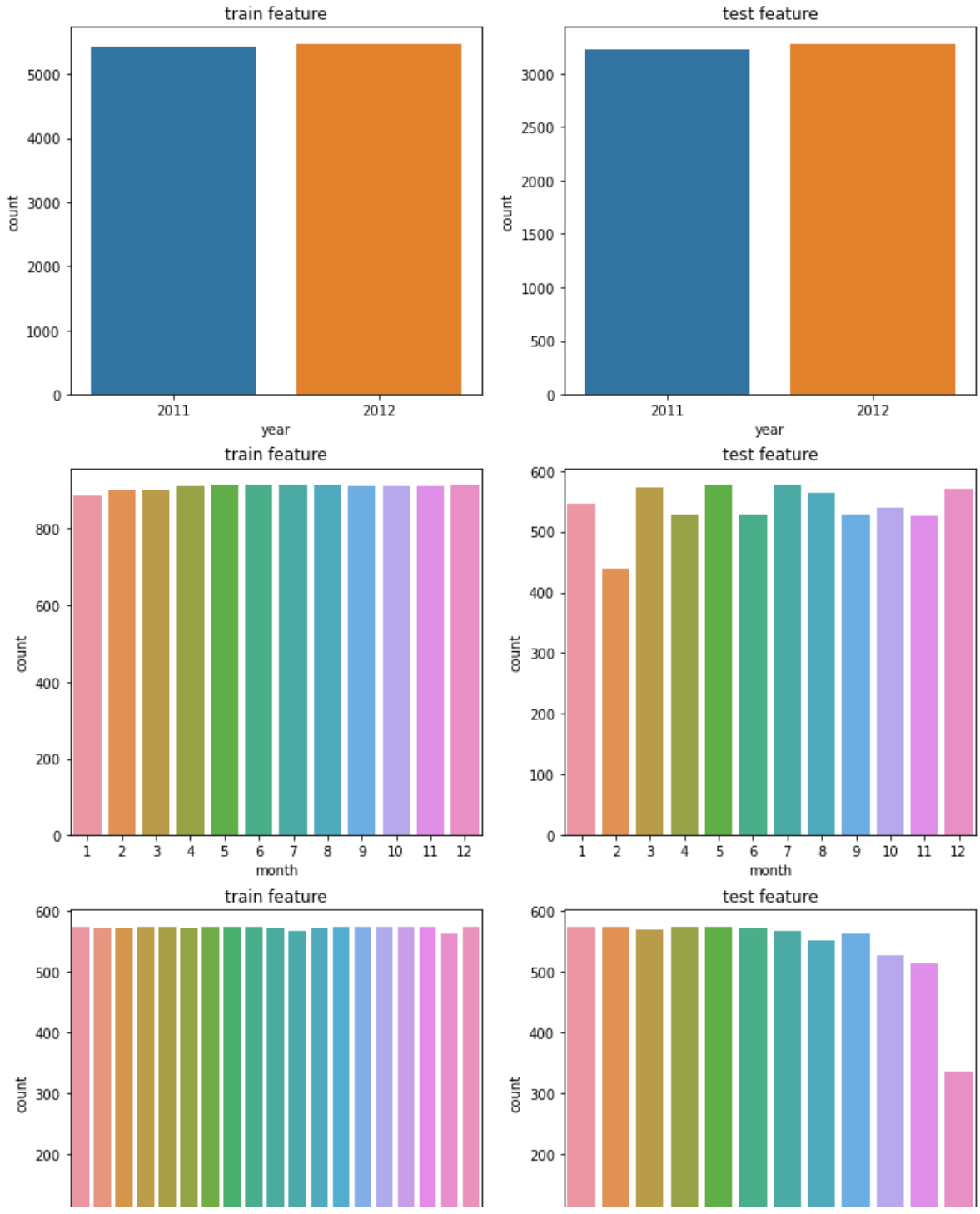
Out[5]:

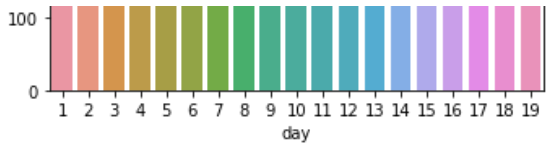
	datetime	year	month	day	hour	dayofweek
0	2011-01-20 00:00:00	2011	1	20	0	3
1	2011-01-20 01:00:00	2011	1	20	1	3
2	2011-01-20 02:00:00	2011	1	20	2	3
3	2011-01-20 03:00:00	2011	1	20	3	3
4	2011-01-20 04:00:00	2011	1	20	4	3
...	...	...	...	...	...	...
6488	2012-12-31 19:00:00	2012	12	31	19	0
6489	2012-12-31 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

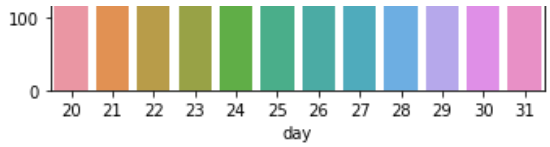
6493 rows × 6 columns

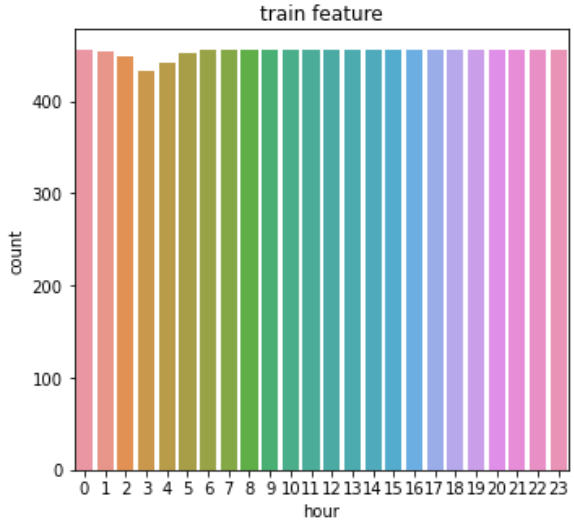
In [7]:

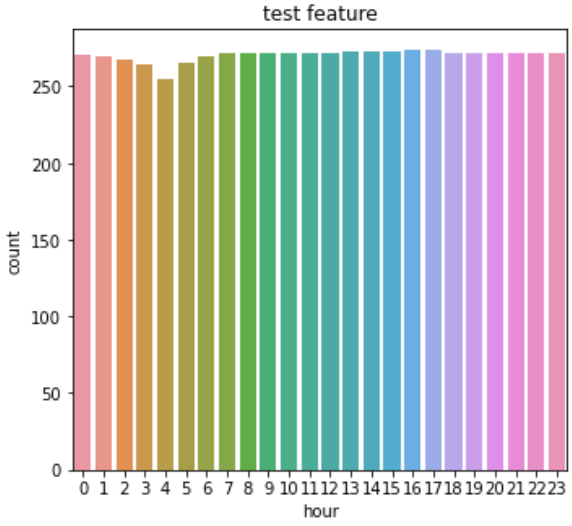
```
col_names = ['year','month','day','hour','dayofweek']
i = 0
plt.figure(figsize=(12,35))  ##전체 그래프 크기 지정

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)
  plt.title("train feature")

  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.title("test feature")
plt.show()
```

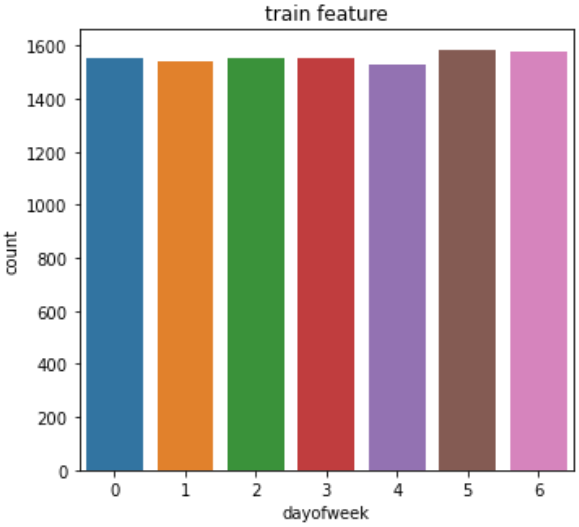


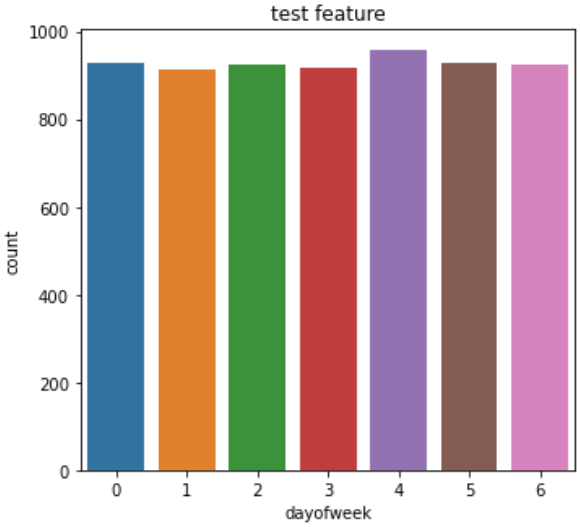

100
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
day


100
0
20 21 22 23 24 25 26 27 28 29 30 31
day


train feature
count
400
300
200
100
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
hour

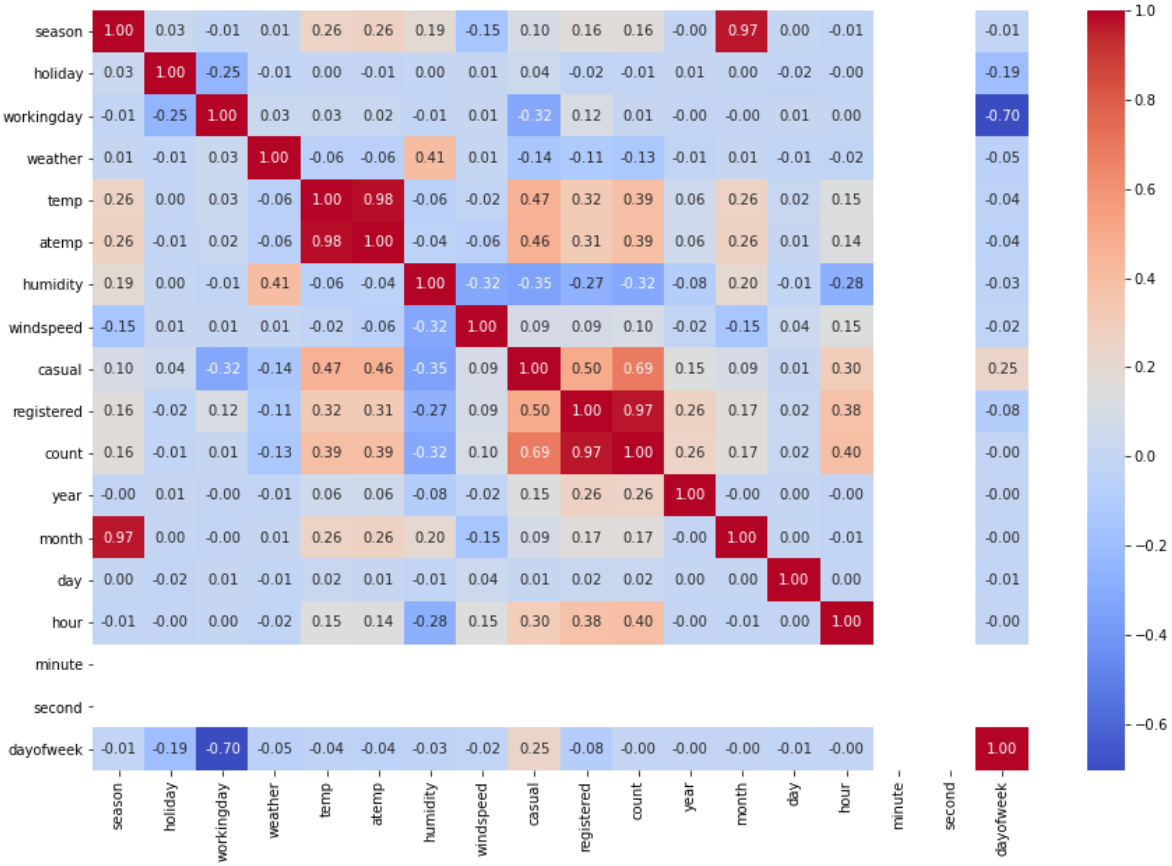

test feature
count
250
200
150
100
50
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
hour


train feature
count
1600
1400
1200
1000
800
600
400
200
0
0 1 2 3 4 5 6
dayofweek


test feature
count
1000
800
600
400
200
0
0 1 2 3 4 5 6
dayofweek

In [8]:

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



In [17]:

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

In [18]:

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

In [19]:

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

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

03 모델 생성

In [61]:

```
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
import lightgbm as lgb
import time
```

In [81]:

```
def model_val(model_name, model_obj):
    now_time = time.time()

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

    pro_time = time.time() - now_time

    print("수행 속도 : {0:.3f}(s)".format(p_time) )  # 걸린 시간
    print("{} Score : {}".format(model_name, mse_score))  # 점수

    return m_score, pro_time
```

In [95]:

```
### 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"]
exe_model = []
model_score = []
model_time = []
```

In [96]:

```
m_name = model_list[0]

# 점수 및 걸린 시간
if m_name not in exe_model:
    model_RF = RandomForestRegressor(random_state=30)
    mse_score, p_time = model_val(m_name, model_RF)

    exe_model.append(m_name)        # 실행된 모델
    model_score.append(mse_score)   # 실행된 모델 점수
    model_time.append(p_time)       # 실행된 모델 걸린 시간
else:
    print(f"{m_name} 이미 실행 완료")
```

```
MSE 평균 :  0.154720099967246
수행 속도 : 0.834(s)
RandomForestRegressor Score : 0.11563736237766957
```

파라미터

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 낮은 값은 underfitting(과소적합)을 야기할 수 있음.	기본값 : 1
colsample_bytree	각 나무마다 사용하는 **feature 비율**. High value can lead to overfitting.	기본값 : 1
reg_alpha(or alpha)	L1 규제에 대한 항 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

In [97]:

```
m_name = model_list[1]

# 점수 및 걸린 시간
if m_name not in exe_model:
    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 = 1000)  # n_estimators=100

    mse_score, p_time = model_val(m_name, xg_reg)

    exe_model.append(m_name)        # 실행된 모델
    model_score.append(mse_score)   # 실행된 모델 점수
    model_time.append(p_time)       # 실행된 모델 걸린 시간
else:
    print(f"{m_name} 이미 실행 완료")
```

```
[02:17:18] 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.
[02:17:20] 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.
[02:17:20] 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.
[02:17:21] 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.
[02:17:22] 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.
[02:17:23] 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 평균 :  0.11607022466516914
수행 속도 : 5.915(s)
xgb_basic Score : 0.154720099967246
```

LightGBM Model

In [98]:

```
import lightgbm as lgb
```

In [99]:

```
m_name = model_list[2]

# 점수 및 걸린 시간
if m_name not in exe_model:
    m_lgbm1 = lgb.LGBMRegressor()

    mse_score, p_time = model_val(m_name, m_lgbm1)

    exe_model.append(m_name)        # 실행된 모델
    model_score.append(mse_score)   # 실행된 모델 점수
    model_time.append(p_time)       # 실행된 모델 걸린 시간
else:
    print(f"{m_name} 이미 실행 완료")
```

```
MSE 평균 :  0.11563736237766957
수행 속도 : 6.543(s)
lightgbm-model1 Score : 0.11607022466516914
```

LightGBM Model2

In [100]:

```
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': 2000}
```

In [101]:

```
m_name = model_list[3]

# 점수 및 걸린 시간
if m_name not in exe_model:
    m_lgbm2 = lgb.LGBMRegressor(**hyperparameters)

    mse_score, p_time = model_val(m_name, m_lgbm2)

    exe_model.append(m_name)        # 실행된 모델
    model_score.append(mse_score)   # 실행된 모델 점수
    model_time.append(p_time)       # 실행된 모델 걸린 시간
else:
    print(f"{m_name} 이미 실행 완료")
```

```
MSE 평균 :  0.11673046439315801
수행 속도 : 0.875(s)
lightgbm-model2 Score : 0.11563736237766957
```

In [103]:

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

Out[103]:

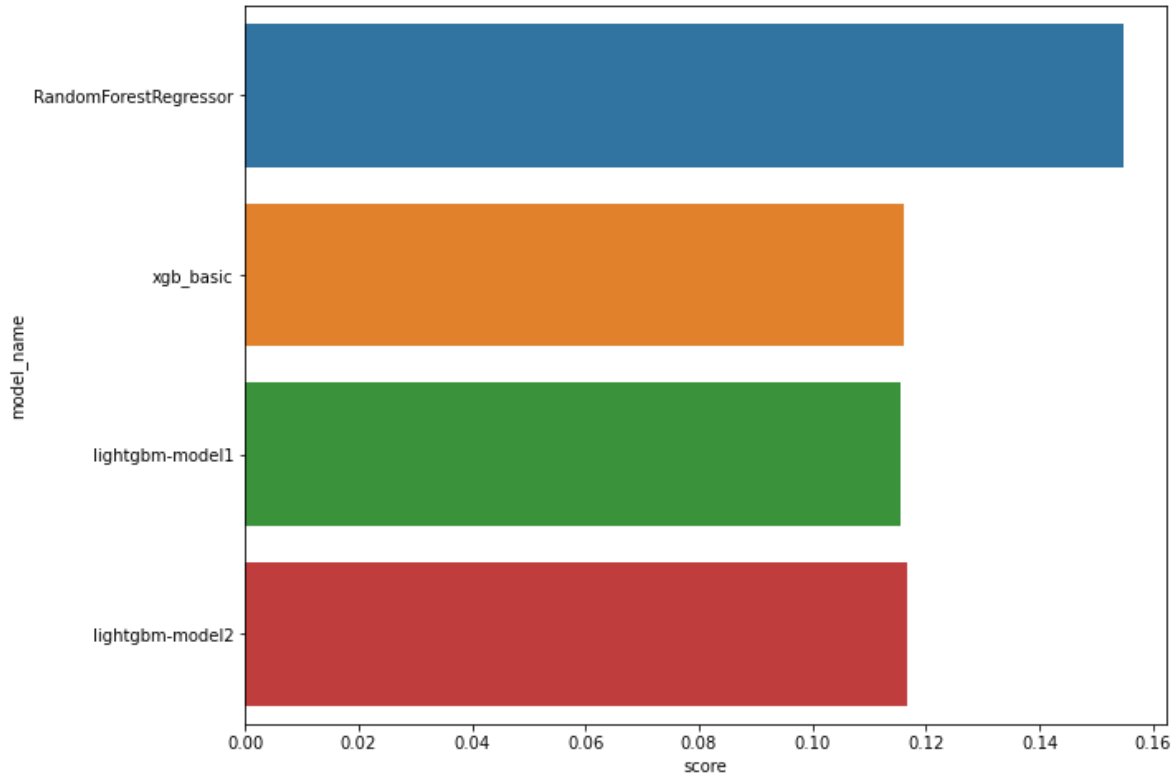
	model_name	score	time
0	RandomForestRegressor	0.154720	5.914816
1	xgb_basic	0.116070	6.542641
2	lightgbm-model1	0.115637	0.874806
3	lightgbm-model2	0.116730	19.341477

In [105]:

```
plt.figure(figsize=(10,8))  ##전체 그래프 크기 지정
sns.barplot(x="score" , y="model_name", data=dat)
```

Out[105]:

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



04 최종 모델

In [108]:

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

pred = m_lgbm2.predict(X_test_l)
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 [110]:

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

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

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 [111]:

```
new_tr.columns
```

Out[111]:

```
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 [112]:

```
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 [113]:

```
new_tr.columns
```

Out[113]:

```
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 [114]:

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

In [115]:

```
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 [117]:

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

pred1 = m_lgbm2.predict(X_test_l)
```

In [118]:

```
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 [119]:

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

pred2 = m_lgbm2.predict(X_test_l)
```

In [120]:

```
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 [121]:

```
# 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_l)

# 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

08. 휴일에 일정한 가중치를 곱하기

In [122]:

```
new_test.columns
```

Out[122]:

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

In [123]:

```
pd.options.display.max_rows = 200
sub["holiday"] = new_test["holiday"]
sub.columns
```

Out[123]:

```
Index(['datetime', 'count', 'holiday'], dtype='object')
```

크리스마스

In [124]:

```
sub.iloc[6332:6345,]
```

Out[124]:

	datetime	count	holiday
6332	2012-12-25 07:00:00	39.636393	1
6333	2012-12-25 08:00:00	124.382777	1
6334	2012-12-25 09:00:00	127.291005	1
6335	2012-12-25 10:00:00	133.831408	1
6336	2012-12-25 11:00:00	155.038302	1
6337	2012-12-25 12:00:00	176.982356	1
6338	2012-12-25 13:00:00	190.880083	1
6339	2012-12-25 14:00:00	198.470642	1
6340	2012-12-25 15:00:00	182.206283	1
6341	2012-12-25 16:00:00	212.430477	1
6342	2012-12-25 17:00:00	232.030846	1
6343	2012-12-25 18:00:00	213.038263	1
6344	2012-12-25 19:00:00	168.734747	1

In [125]:

```
sub['count'] = np.expm1(pred1) + np.expm1(pred2)
```

In [126]:

```
# count 값을 0.5를 곱함 - 크리스마스
sub.iloc[3065:3087,1] = sub.iloc[3065:3087,1] * 0.5
sub.iloc[6332:6345,1] = sub.iloc[6332:6345,1] * 0.5
```

실습 - 추수감사절, 미국 현충일, 크리스마스 이브해보기

In [127]:

```
# 추수 감사절
sub.iloc[2771:2794,1] = sub.iloc[2771:2794,1] * 0.5
sub.iloc[5992:6015,1] = sub.iloc[5992:6015,1] * 0.5

# 미국 현충일
sub.iloc[1258:1269,1] = sub.iloc[1258:1269,1] * 0.5
sub.iloc[4492:4515,1] = sub.iloc[4492:4515,1] * 0.5

# 크리스마스 이브
sub.iloc[3041:3063,1] = sub.iloc[3041:3063,1] * 0.5
sub.iloc[6308:6330,1] = sub.iloc[6308:6330,1] * 0.5
```

In [128]:

```
sub.drop("holiday", 1, inplace=True)
sub.to_csv("sub_v10_lgbm_rf_add.csv", index=False)
```

0.36183 (15등)

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)