Bike Sharing Demand 데이터 분석

In [64]:

```
from pandas import Series, DataFrame
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats
import missingno as msno # 결측치 확인 패키지
from datetime import datetime
from sklearn.model_selection import cross_val_score #교차검증
%matplotlib inline
```

Train 데이터 입력

In [65]:

```
train = pd.read_csv("train.csv", parse_dates=["datetime"])
print(train.shape)
train.head()
```

```
(10886, 12)
```

Out[65]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspee
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0

Test 데이터 입력

In [66]:

```
test = pd.read_csv("test.csv", parse_dates=["datetime"])
print(test.shape)
test.head()
```

```
(6493, 9)
```

Out[66]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspee
0	2011-01-20 00:00:00	1	0	1	1	10.66	11.365	56	26.0027
1	2011-01-20 01:00:00	1	0	1	1	10.66	13.635	56	0.0000
2	2011-01-20 02:00:00	1	0	1	1	10.66	13.635	56	0.0000
3	2011-01-20 03:00:00	1	0	1	1	10.66	12.880	56	11.0014
4	2011-01-20 04:00:00	1	0	1	1	10.66	12.880	56	11.0014

변수

In [67]:

```
train.columns
```

Out[67]:

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

datetime - hourly date + timestamp
season - 1 = spring, 2 = summer, 3 = fall, 4 = winter
holiday - 공휴일
workingday - 공휴일이 아닌날
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 - 실제온도
atemp - 체감온도
humidity - 흡도
windspeed - 풍속
casual - 렌탈시 등록되 있지 않은 사용자 수
registered - 렌탈시 등록되어 있는 사용자 수
count - 총 렌탈 회수

데이터 타입

In [68]:

```
train.dtypes
```

Out[68]:

```
datetime      datetime64[ns]
season                 int64
holiday                int64
workingday             int64
weather                int64
temp                 float64
atemp                float64
humidity               int64
windspeed            float64
casual                 int64
registered             int64
count                  int64
dtype: object
```

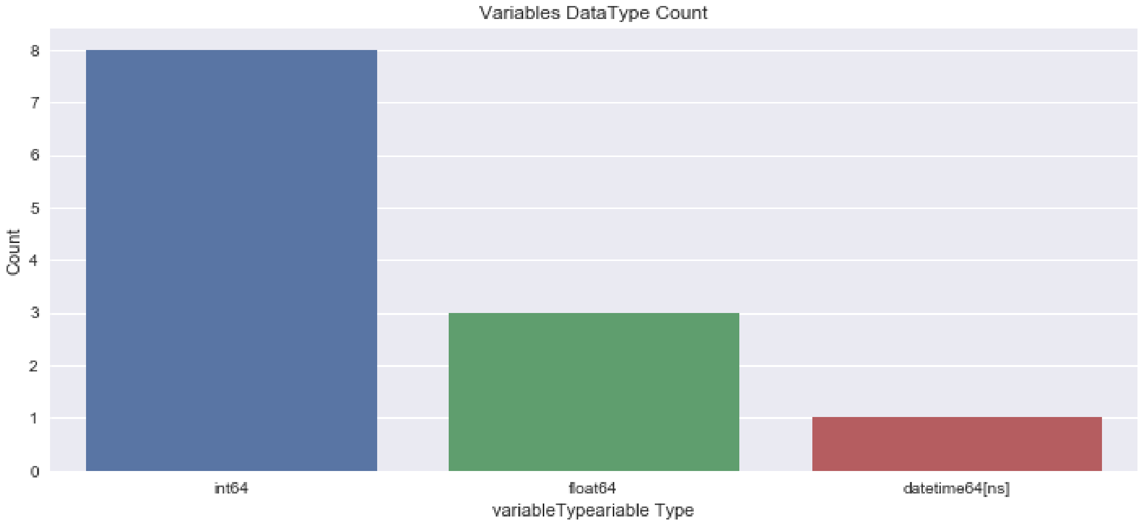
변수별 데이터타입 개수

In [69]:

```
dataTypeDf = pd.DataFrame(train.dtypes.value_counts()).reset_index().rename(columns={"index":"variableType",0:"count"})
fig,ax = plt.subplots()
fig.set_size_inches(12,5)
sns.barplot(data=dataTypeDf,x="variableType",y="count",ax=ax)
ax.set(xlabel='variableTypeariable Type', ylabel='Count',title="Variables DataType Count")
```

Out[69]:

```
[<matplotlib.text.Text at 0x13aa5e70>,
 <matplotlib.text.Text at 0x13acc5f0>,
 <matplotlib.text.Text at 0x13abafd0>]
```



데이터 요약

In [70]:

```
train.describe()
```

Out[70]:

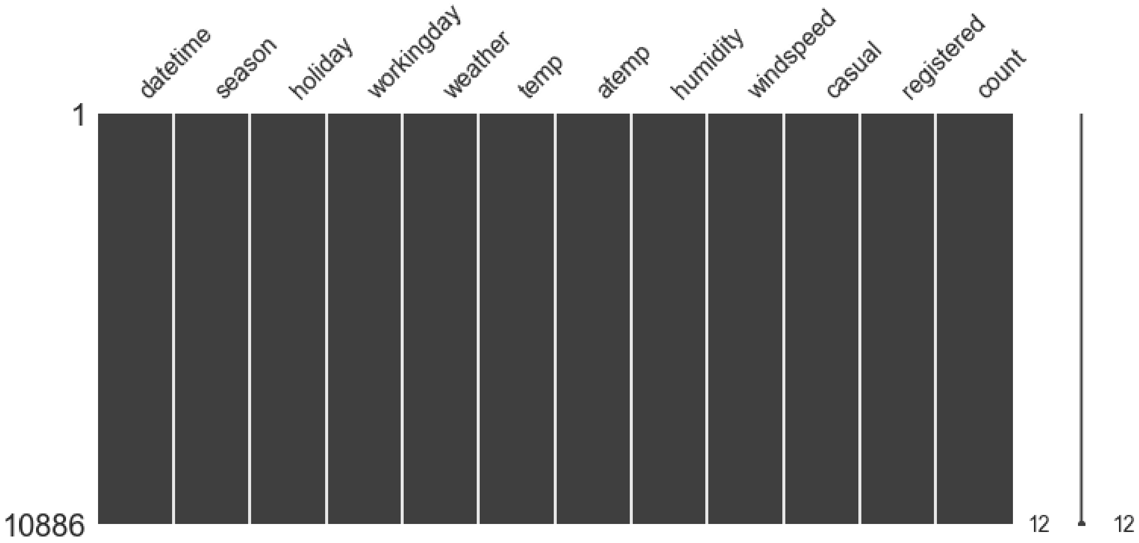
	season	holiday	workingday	weather	temp	
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.4746
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.7600
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455

데이터 분석

데이터 결측치

In [71]:

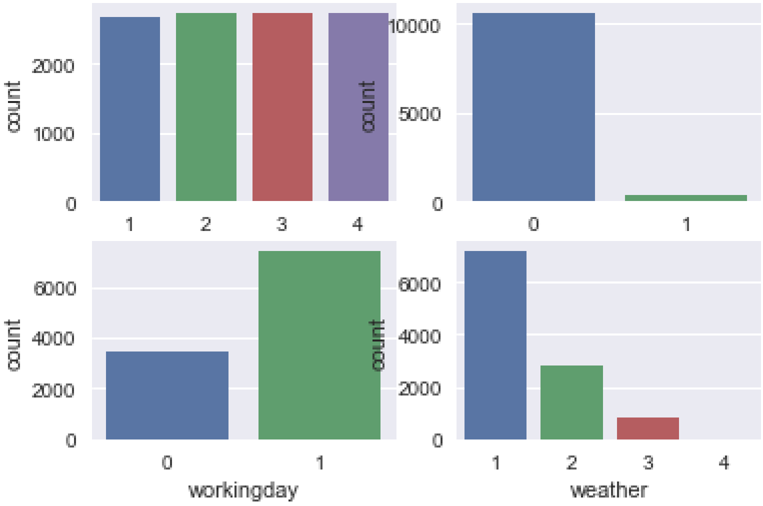
```
msno.matrix(train,figsize=(12,5))
```



명목형 변수에 대한 일변량 분석(Univariate analysis)

In [72]:

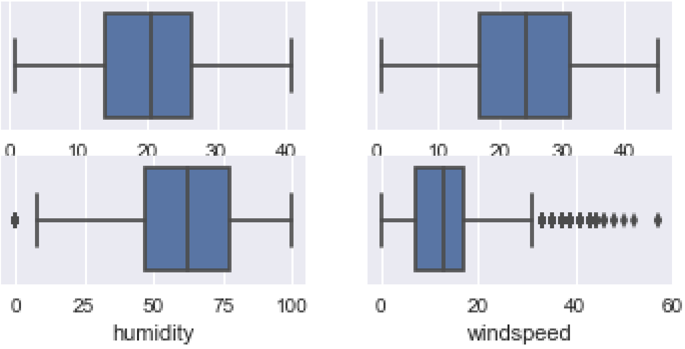
```
cat_names=['season', 'holiday', 'workingday', 'weather']

i=0
for name in cat_names:
    i=i+1
    plt.subplot(2,2,i)
    sns.countplot(name,data=train)
```



연속형 변수에 대한 일변량 분석(Univariate analysis)

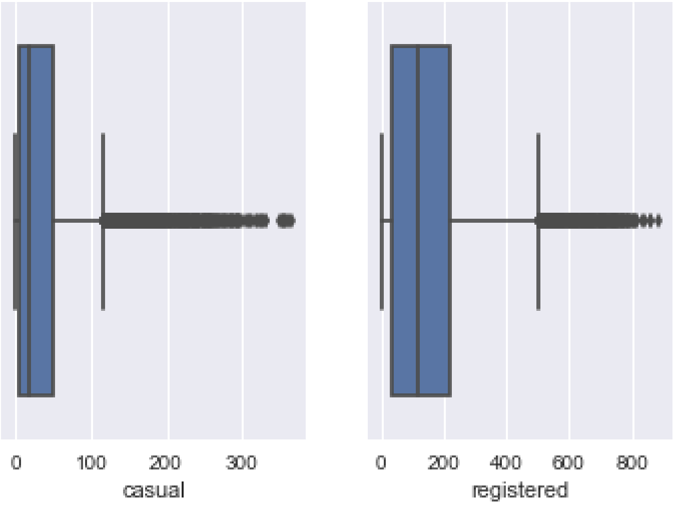
In [73]:

```
cont_names=['temp','atemp','humidity','windspeed']

i=0
for name in cont_names:
    i=i+1
    plt.subplot(3,2,i)
    sns.boxplot(name,data=train)
```



In [74]:

```
cont_names=['casual','registered']

i=0
for name in cont_names:
    i=i+1
    plt.subplot(1,2,i)
    sns.boxplot(name,data=train)
```



Datetime 변수를 파생변수를 생성시킨다

In [75]:

```
train['datetime'].head()
```

Out[75]:

```
0   2011-01-01 00:00:00
1   2011-01-01 01:00:00
2   2011-01-01 02:00:00
3   2011-01-01 03:00:00
4   2011-01-01 04:00:00
Name: datetime, dtype: datetime64[ns]
```

In [76]:

```
train['dat_year'] = train["datetime"].dt.year
train['dat_month'] = train["datetime"].dt.month
train['dat_day'] = train["datetime"].dt.day
train['dat_hour'] = train["datetime"].dt.hour
train['dat_minute'] = train["datetime"].dt.minute
train['dat_second'] = train["datetime"].dt.second
train['dat_dayofweek'] = train["datetime"].dt.dayofweek
#train[["datetime","dat_year",'dat_month','dat_day','dat_hour','dat_minute','dat_second','dat_da
yofweek']]
```

In [77]:

```
test['dat_year'] = test["datetime"].dt.year
test['dat_month'] = test["datetime"].dt.month
test['dat_day'] = test["datetime"].dt.day
test['dat_hour'] = test["datetime"].dt.hour
test['dat_minute'] = test["datetime"].dt.minute
test['dat_second'] = test["datetime"].dt.second
test['dat_dayofweek'] = test["datetime"].dt.dayofweek
```

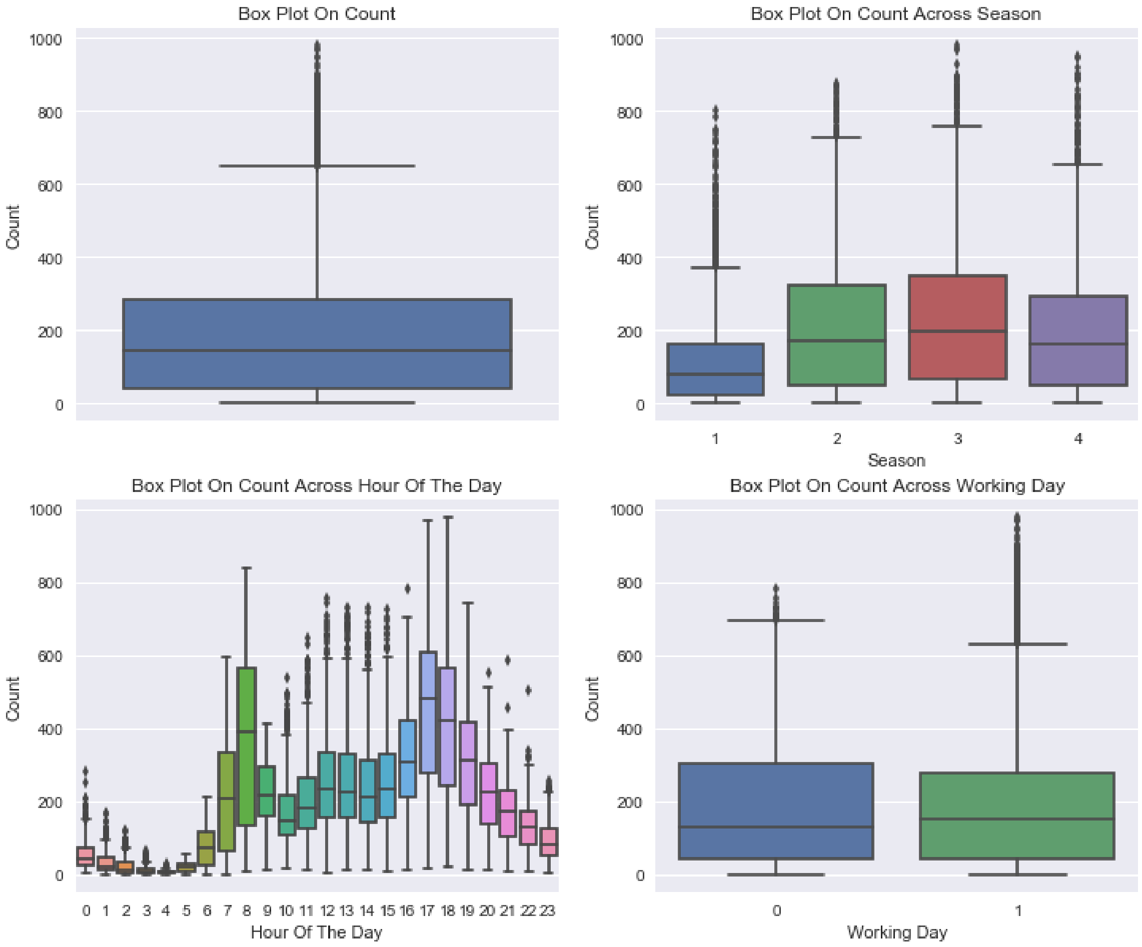
Outlier 분석

In [78]:

```
fig, axes = plt.subplots(nrows=2,ncols=2)
fig.set_size_inches(12, 10)
sns.boxplot(data=train,y="count",orient="v",ax=axes[0][0])
sns.boxplot(data=train,y="count",x="season",orient="v",ax=axes[0][1])
sns.boxplot(data=train,y="count",x="dat_hour",orient="v",ax=axes[1][0])
sns.boxplot(data=train,y="count",x="workingday",orient="v",ax=axes[1][1])

axes[0][0].set(ylabel='Count',title="Box Plot On Count")
axes[0][1].set(xlabel='Season', ylabel='Count',title="Box Plot On Count Across Season")
axes[1][0].set(xlabel='Hour Of The Day', ylabel='Count',title="Box Plot On Count Across Hour Of
 The Day")
axes[1][1].set(xlabel='Working Day', ylabel='Count',title="Box Plot On Count Across Working Day"
)
```

Out[78]:

```
[<matplotlib.text.Text at 0x3371cf10>,
 <matplotlib.text.Text at 0x33715430>,
 <matplotlib.text.Text at 0x33715fb0>]
```



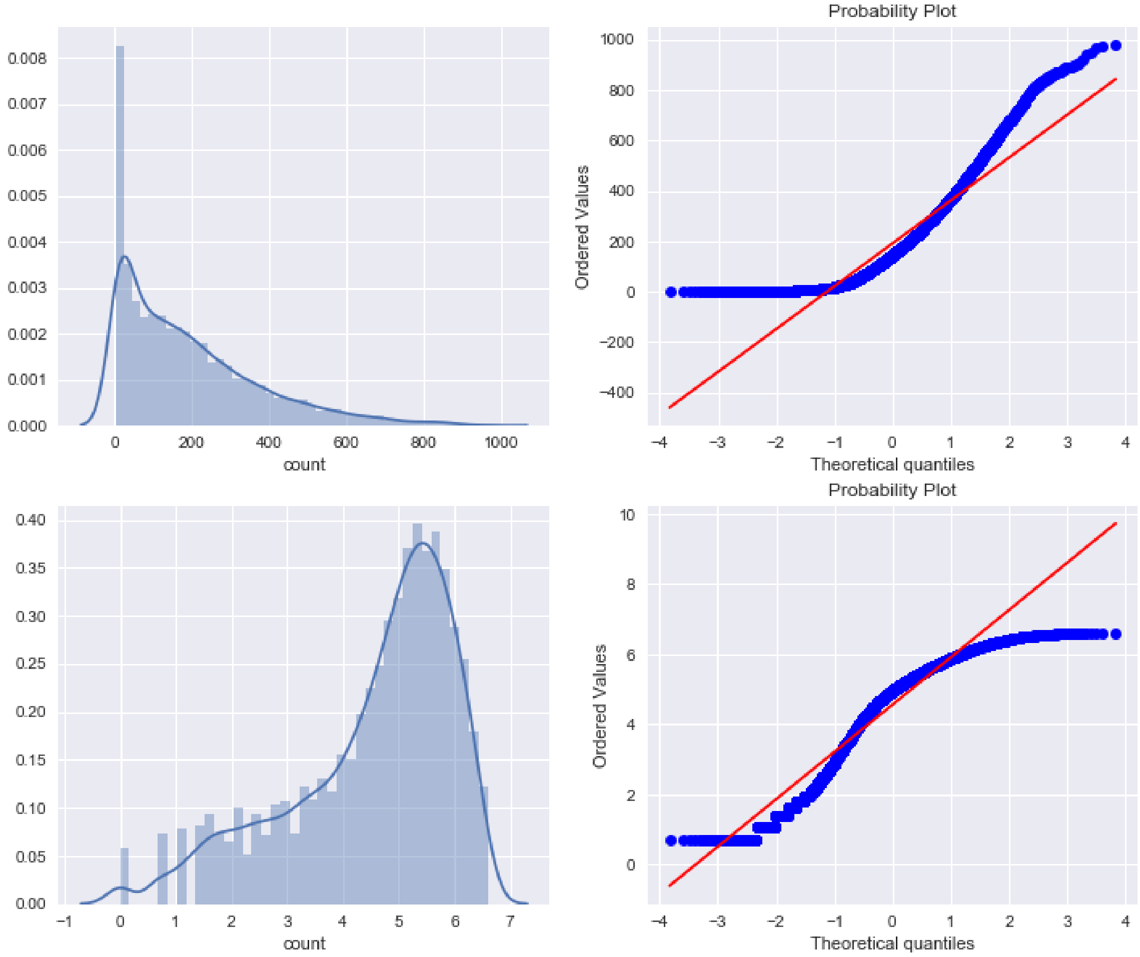
log를 이용한 count 데이터 분포 확인

In [115]:

```
fig,axes = plt.subplots(ncols=2,nrows=2)
fig.set_size_inches(12, 10)
sns.distplot(train["count"],ax=axes[0][0])
stats.probplot(train["count"], dist='norm', fit=True, plot=axes[0][1])
sns.distplot(np.log(dailyDataWithoutOutliers["count"]),ax=axes[1][0])
stats.probplot(np.log1p(dailyDataWithoutOutliers["count"]), dist='norm', fit=True, plot=axes[1][
1])
```

Out[115]:

```
((array([-3.82819677, -3.60401975, -3.48099008, ...,  3.48099008,
          3.60401975,  3.82819677]),
  array([ 0.69314718,  0.69314718,  0.69314718, ...,  6.5971457 ,
          6.59850903,  6.5998705 ])),
 (1.3486990121229785, 4.5624238680878078, 0.95811767809096116))
```



season,weather, dat_dayofweek 변수의 파생변수를 생성시킨다

In [79]:

```
train['season1'] = train['season' ] == 1
train['season2'] = train['season' ] == 2
train['season3'] = train['season' ] == 3
train['season4'] = train['season' ] == 4
#train[['season','season1','season2','season3','season4']]
```

In [80]:

```
train['weather1'] = train['weather' ] == 1
train['weather2'] = train['weather' ] == 2
train['weather3'] = train['weather' ] == 3
train['weather4'] = train['weather' ] == 4
#train[['weather','weather1','weather2','weather3','weather4']]
```

In [81]:

```
train['dat_dayofweek1'] = train['dat_dayofweek' ] == 0
train['dat_dayofweek2'] = train['dat_dayofweek' ] == 1
train['dat_dayofweek3'] = train['dat_dayofweek' ] == 2
train['dat_dayofweek4'] = train['dat_dayofweek' ] == 3
train['dat_dayofweek5'] = train['dat_dayofweek' ] == 4
train['dat_dayofweek6'] = train['dat_dayofweek' ] == 5
train['dat_dayofweek7'] = train['dat_dayofweek' ] == 6
```

In [82]:

```
test['season1'] = test['season' ] == 1
test['season2'] = test['season' ] == 2
test['season3'] = test['season' ] == 3
test['season4'] = test['season' ] == 4
```

In [83]:

```
test['weather1'] = test['weather' ] == 1
test['weather2'] = test['weather' ] == 2
test['weather3'] = train['weather' ] == 3
test['weather4'] = test['weather' ] == 4
#test[['weather','weather1','weather2','weather3','weather4']]
```

In [84]:

```
test['dat_dayofweek1'] = test['dat_dayofweek' ] == 0
test['dat_dayofweek2'] = test['dat_dayofweek' ] == 1
test['dat_dayofweek3'] = test['dat_dayofweek' ] == 2
test['dat_dayofweek4'] = test['dat_dayofweek' ] == 3
test['dat_dayofweek5'] = test['dat_dayofweek' ] == 4
test['dat_dayofweek6'] = test['dat_dayofweek' ] == 5
test['dat_dayofweek7'] = test['dat_dayofweek' ] == 6
```

파생변수 생성 (holiday + workingday)

In [85]:

```
test['hwday'] = test['holiday'] + test['workingday']
```

In [86]:

```
train['hwday'] = train['holiday'] + train['workingday']
#train[['hwday','holiday','workingday']]
```

전체 변수 재확인

In [87]:

```
train.columns
```

Out[87]:

```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count',
       'dat_year', 'dat_month', 'dat_day', 'dat_hour', 'dat_minute',
       'dat_second', 'dat_dayofweek', 'season1', 'season2', 'season3',
       'season4', 'weather1', 'weather2', 'weather3', 'weather4',
       'dat_dayofweek1', 'dat_dayofweek2', 'dat_dayofweek3', 'dat_dayofweek4',
       'dat_dayofweek5', 'dat_dayofweek6', 'dat_dayofweek7', 'hwday'],
      dtype='object')
```

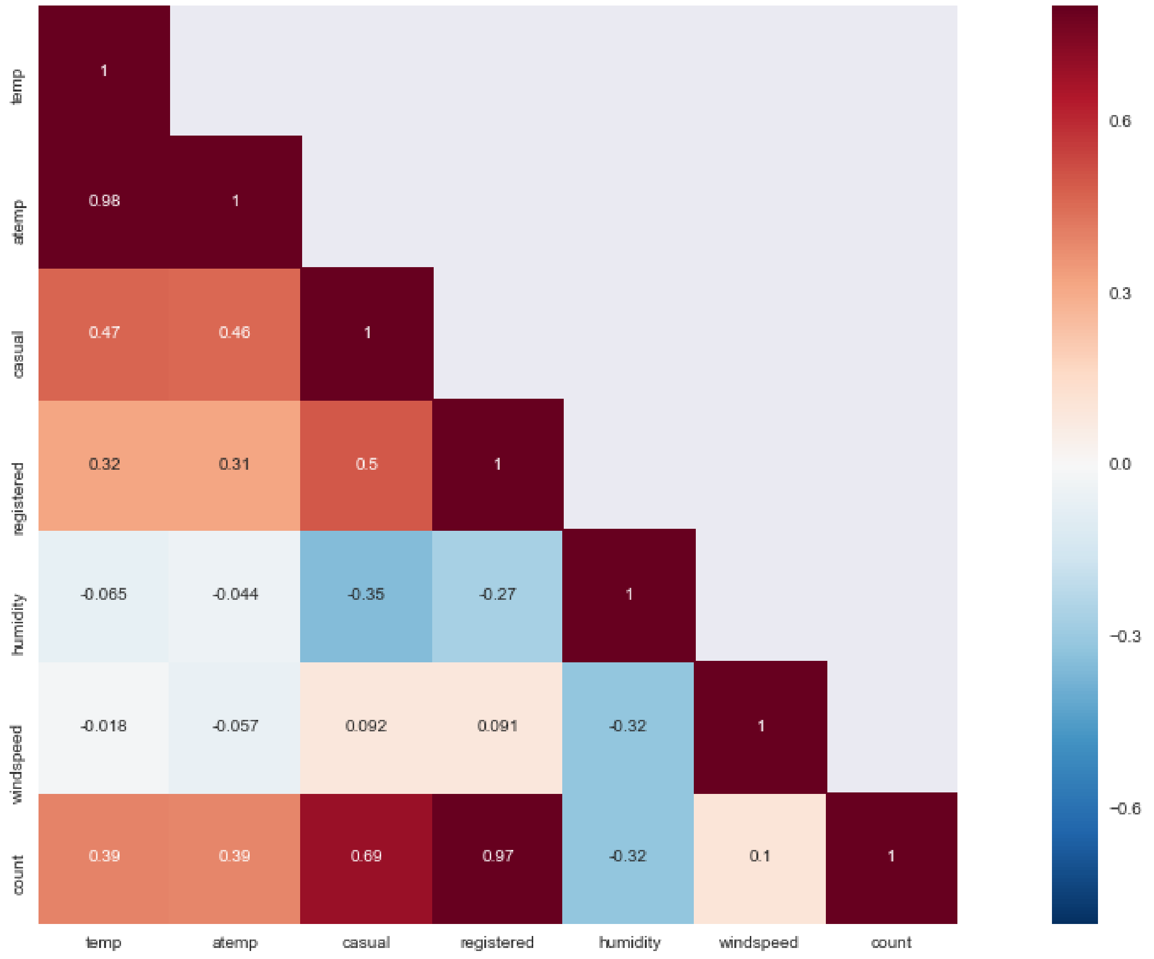
상관관계분석

In [88]:

```
corrMatt = train[["temp","atemp","casual","registered","humidity","windspeed","count"]].corr()
mask = np.array(corrMatt)
mask[np.tril_indices_from(mask)] = False
fig,ax= plt.subplots()
fig.set_size_inches(20,10)
sns.heatmap(corrMatt, mask=mask,vmax=.8, square=True,annot=True)
```

Out[88]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x149c58d0>
```



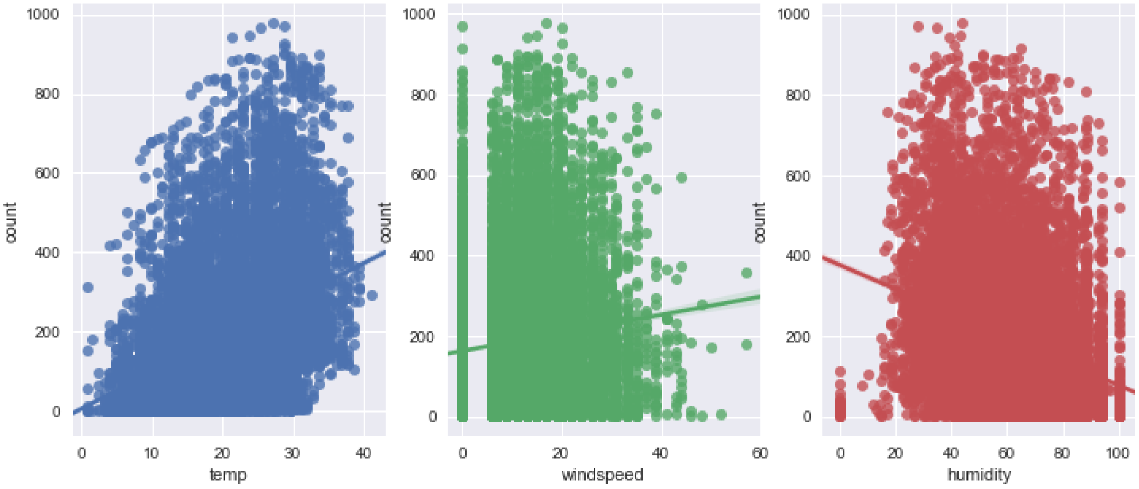
Regression plot을 이용한 변수간 관계 확인

In [89]:

```
fig,(ax1,ax2,ax3) = plt.subplots(ncols=3)
fig.set_size_inches(12, 5)
sns.regplot(x="temp", y="count", data=train,ax=ax1)
sns.regplot(x="windspeed", y="count", data=train,ax=ax2)
sns.regplot(x="humidity", y="count", data=train,ax=ax3)
```

Out[89]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xd067fb0>
```

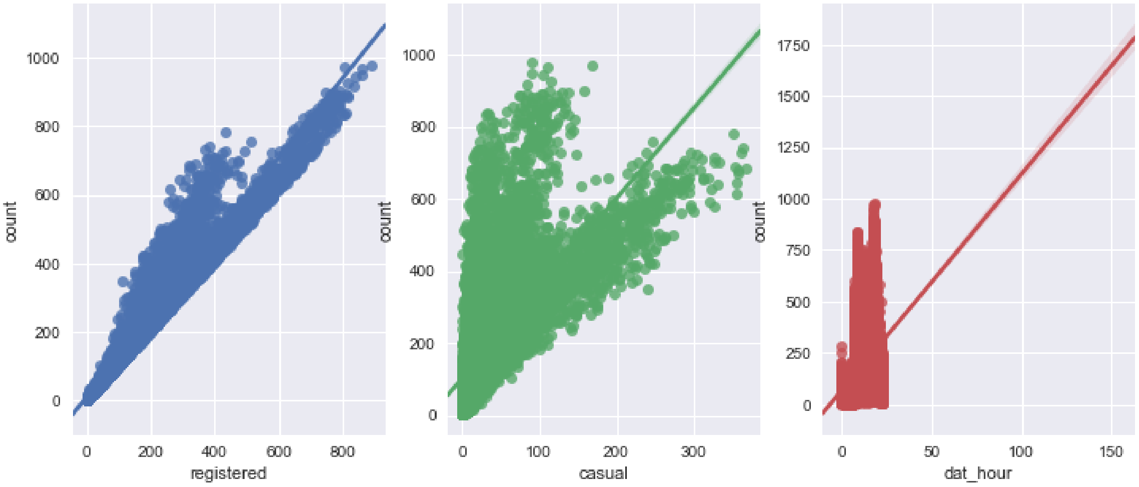


In [90]:

```
fig,(ax1,ax2,ax3) = plt.subplots(ncols=3)
fig.set_size_inches(12, 5)
sns.regplot(x="registered", y="count", data=train,ax=ax1)
sns.regplot(x="casual", y="count", data=train,ax=ax2)
sns.regplot(x="dat_hour", y="count", data=train,ax=ax3)
```

Out[90]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xd10e0f0>
```

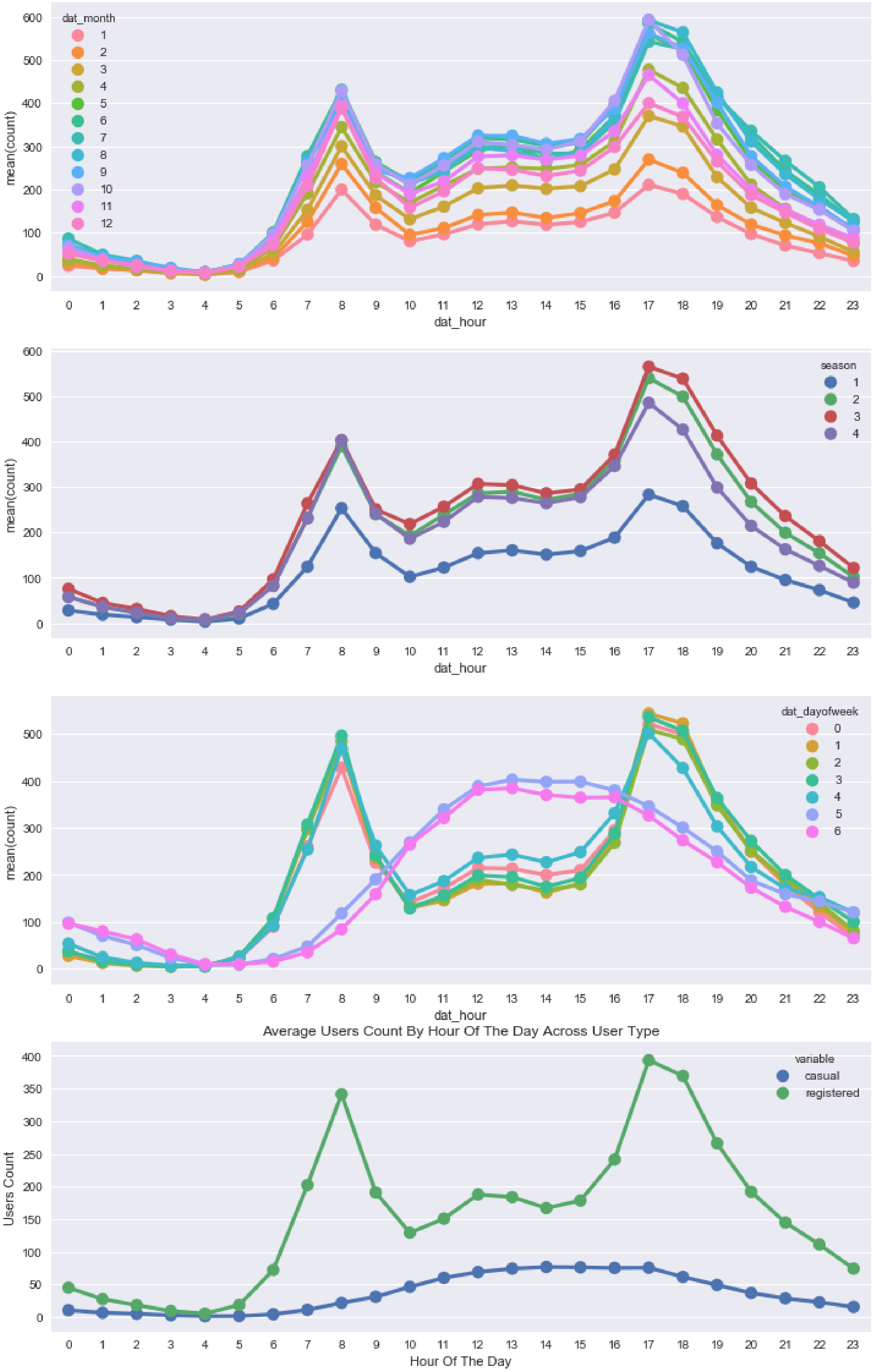


count,hour 변수 와 다른 변수(Month,Season,Weekday,Usertype)관계

In [91]:

```
figure, (ax1,ax2,ax3,ax4) = plt.subplots(nrows=4, ncols=1)
figure.set_size_inches(12,20)

sns.pointplot(data=train, x="dat_hour", y="count", hue="dat_month", ci=None, ax=ax1)
sns.pointplot(data=train, x="dat_hour", y="count", hue="season", ci=None, ax=ax2)
sns.pointplot(data=train, x="dat_hour", y="count", hue="dat_dayofweek", ci=None, ax=ax3)

data1 = pd.melt(train[["dat_hour","casual","registered"]], id_vars=['dat_hour'], value_vars=['ca
sual', 'registered'])
data2 = pd.DataFrame(data1.groupby(["dat_hour","variable"],sort=True)["value"].mean()).reset_ind
ex()
sns.pointplot(x="dat_hour", y="value",hue="variable",hue_order=["casual","registered"], data=dat
a2,ax=ax4)
ax4.set(xlabel='Hour Of The Day', ylabel='Users Count',title="Average Users Count By Hour Of The
 Day Across User Type",label='big')
```

```
Out[91]:

[<matplotlib.text.Text at 0x14573950>,
 <matplotlib.text.Text at 0x14a12e50>,
 <matplotlib.text.Text at 0x145717f0>,
 None]
```


dat_month
1
2
3
4
5
6
7
8
9
10
11
12
mean(count)
0
100
200
300
400
500
600
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
dat_hour
season
1
2
3
4
mean(count)
0
100
200
300
400
500
600
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
dat_hour
dat_dayofweek
0
1
2
3
4
5
6
mean(count)
0
100
200
300
400
500
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
dat_hour
Average Users Count By Hour Of The Day Across User Type
variable
casual
registered
Users Count
0
50
100
150
200
250
300
350
400
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 Of The Day

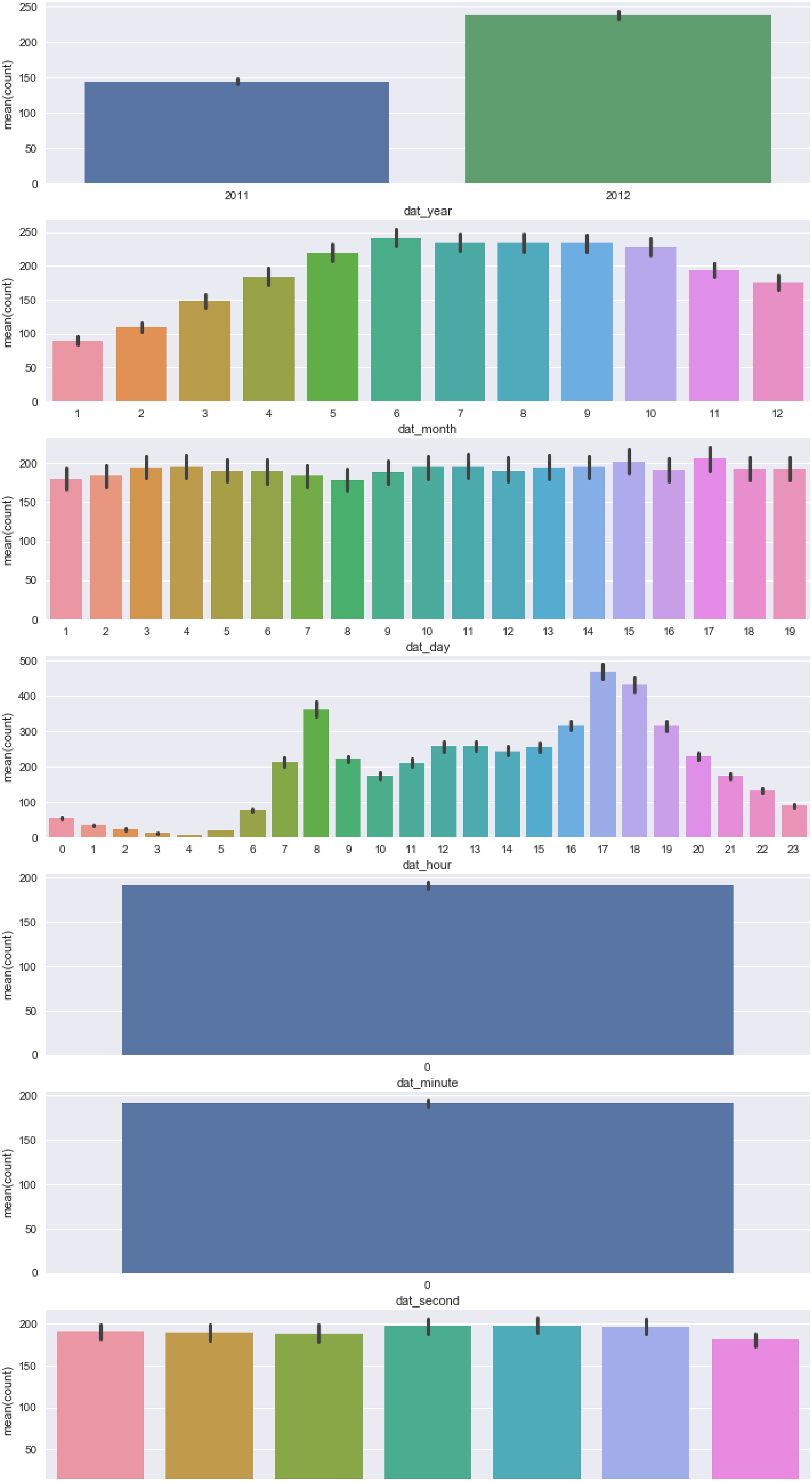
렌탈은 주로 등록된 사람이 많이 사용하고, Workingday에 아침, 저녁 출퇴근 시간에 많이 사용된다.

날짜데이터 와 count 관계

In [92]:

```
figure, (ax1,ax2,ax3,ax4,ax5,ax6,ax7) = plt.subplots(nrows=7, ncols=1)
figure.set_size_inches(12,24)

sns.barplot(data=train, x="dat_year", y="count", ax=ax1)
sns.barplot(data=train, x="dat_month", y="count", ax=ax2)
sns.barplot(data=train, x="dat_day", y="count", ax=ax3)
sns.barplot(data=train, x="dat_hour", y="count", ax=ax4)
sns.barplot(data=train, x="dat_minute", y="count", ax=ax5)
sns.barplot(data=train, x="dat_second", y="count", ax=ax6)
sns.barplot(data=train, x="dat_dayofweek", y="count", ax=ax7)
```

```
Out[92]:

<matplotlib.axes._subplots.AxesSubplot at 0x145478f0>
```


mean(count)
0
50
100
150
200
250
2011
2012
dat_year
mean(count)
0
50
100
150
200
250
1
2
3
4
5
6
7
8
9
10
11
12
dat_month
mean(count)
0
50
100
150
200
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
dat_day
mean(count)
0
100
200
300
400
500
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
dat_hour
mean(count)
0
50
100
150
200
0
dat_minute
mean(count)
0
50
100
150
200
0
dat_second
mean(count)
50
100
150
200

분, 초에 대한 데이터는 무의미 한 것으로 보인다.

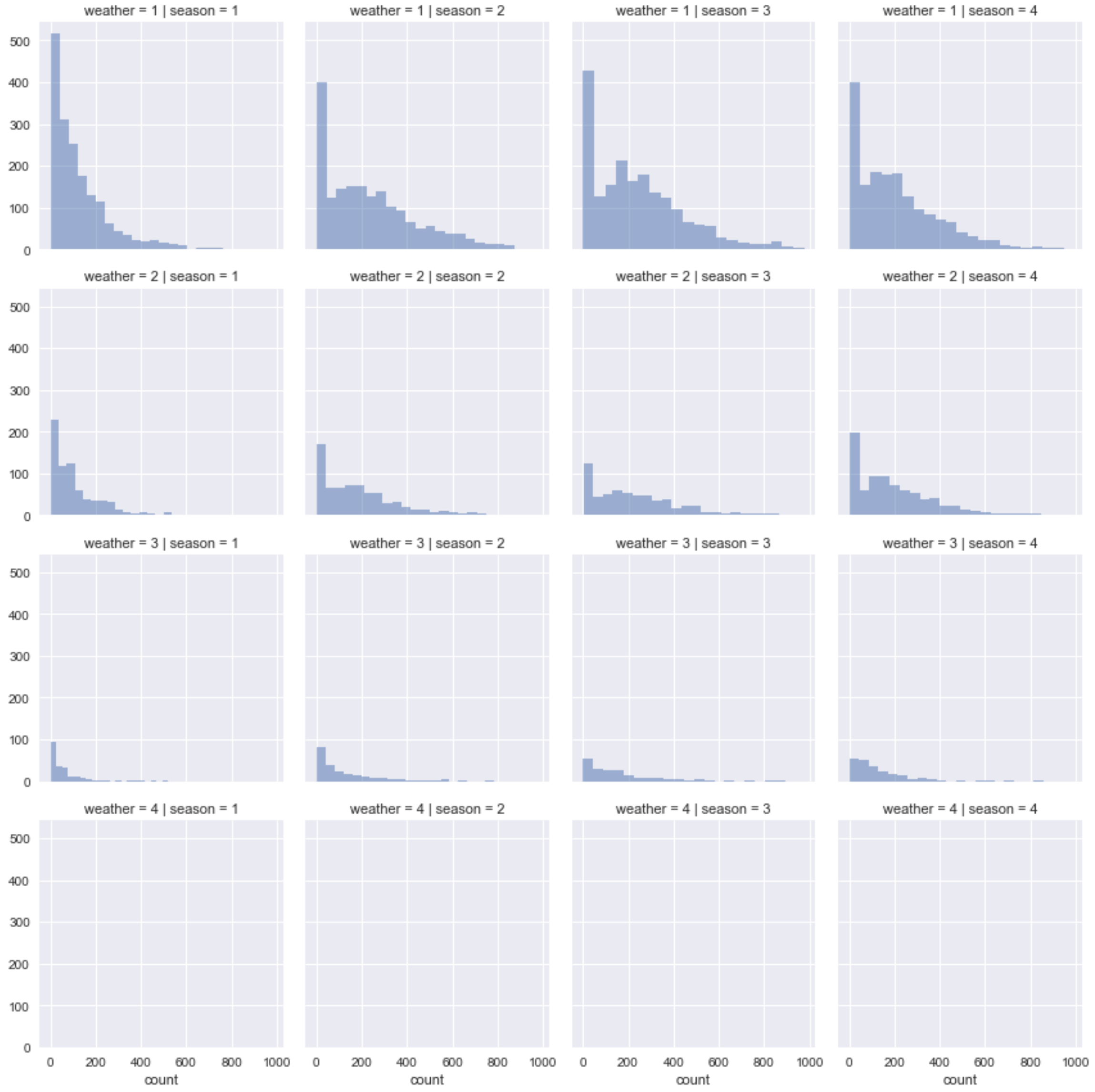
weather 와 season 별 count 관계

In [93]:

```
grid = sns.FacetGrid(train, col='season', row='weather')
grid.map(plt.hist, 'count', alpha=0.5, bins=20)
```

Out[93]:

```
<seaborn.axisgrid.FacetGrid at 0xd04a170>
```



weather3 와 season3 & weather4 와 season4 는 count에 영향이 적은 것 같다.

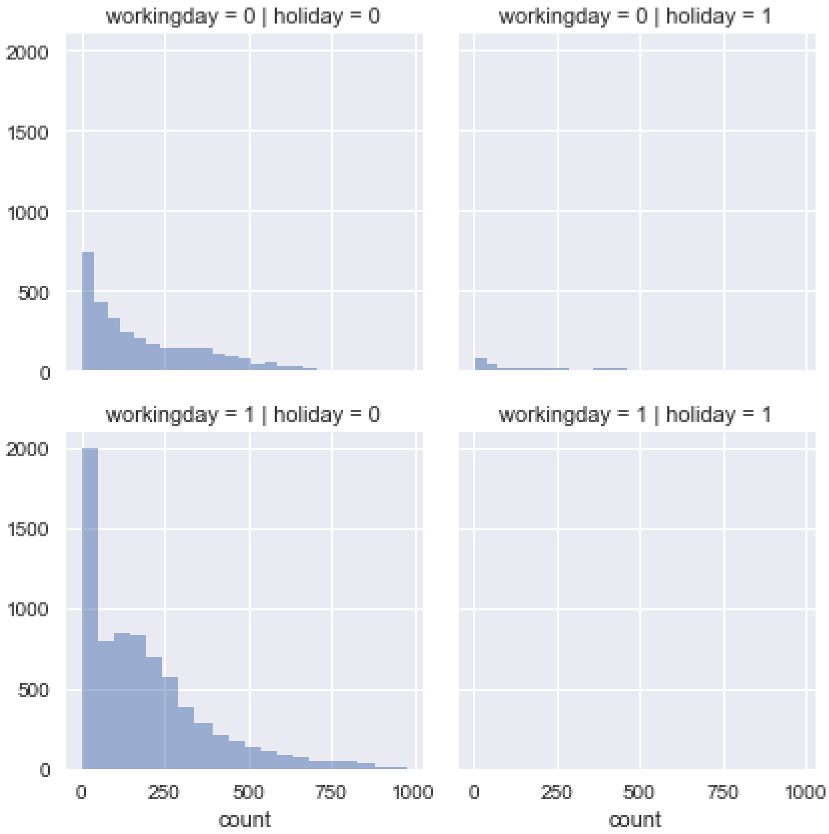
holiday 와 workingday 별 count

In [94]:

```
grid = sns.FacetGrid(train, col='holiday', row='workingday')
grid.map(plt.hist, 'count', alpha=0.5, bins=20)
```

Out[94]:

```
<seaborn.axisgrid.FacetGrid at 0x3529e7b0>
```



workingday 변수가 count와 관계가 있다.

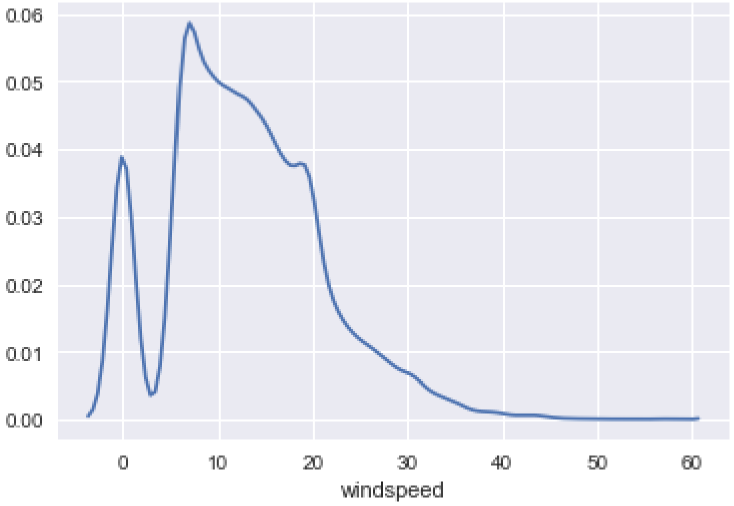
windspeed 값의 상당수가 0으로 나타난다.

In [95]:

```
plt.subplots()
sns.distplot(train["windspeed"], hist=False)
```

Out[95]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x35a205f0>
```



변수선택

In [96]:

```
feature_names = ['workingday','temp','atemp','humidity',
                 'dat_year','dat_hour',
                 'dat_dayofweek1','dat_dayofweek2',
                 'dat_dayofweek3','dat_dayofweek4',
                 'dat_dayofweek5','dat_dayofweek6',
                 'dat_dayofweek7','season','weather']
feature_names
```

Out[96]:

```
['workingday',
 'temp',
 'atemp',
 'humidity',
 'dat_year',
 'dat_hour',
 'dat_dayofweek1',
 'dat_dayofweek2',
 'dat_dayofweek3',
 'dat_dayofweek4',
 'dat_dayofweek5',
 'dat_dayofweek6',
 'dat_dayofweek7',
 'season',
 'weather']
```

In [97]:

```
x_train = train[feature_names]
print(x_train.shape)
x_train.head()
```

```
(10886, 15)
```

Out[97]:

	workingday	temp	atemp	humidity	dat_year	dat_hour	dat_dayofweek1	dat_day
0	0	9.84	14.395	81	2011	0	False	False
1	0	9.02	13.635	80	2011	1	False	False
2	0	9.02	13.635	80	2011	2	False	False
3	0	9.84	14.395	75	2011	3	False	False
4	0	9.84	14.395	75	2011	4	False	False

In [98]:

```
x_test = test[feature_names]
print(x_test.shape)
x_test.head()
```

```
(6493, 15)
```

Out[98]:

	workingday	temp	atemp	humidity	dat_year	dat_hour	dat_dayofweek1	dat_day
0	1	10.66	11.365	56	2011	0	False	False
1	1	10.66	13.635	56	2011	1	False	False
2	1	10.66	13.635	56	2011	2	False	False
3	1	10.66	12.880	56	2011	3	False	False
4	1	10.66	12.880	56	2011	4	False	False

예측할 변수 count 선정 (test 데이터의 y_test 을 예측한다)

In [99]:

```
label_name = "count"
y_train = train[label_name]
print(y_train.shape)
y_train.head()
```

```
(10886,)
```

Out[99]:

```
0    16
1    40
2    32
3    13
4     1
Name: count, dtype: int64
```

모델생성

DecisionTreeRegressor : Regression 모델

In [100]:

```
from sklearn.tree import DecisionTreeRegressor

model = DecisionTreeRegressor(random_state=37)
model
```

Out[100]:

```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_impurity_split=1e-07,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, presort=False, random_state=37,
           splitter='best')
```

> random_state는 random number generator을 의미한다

RandomForest : Regression 모델

In [101]:

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators=500,n_jobs=-1)
model
```

Out[101]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_split=1e-07, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           n_estimators=500, n_jobs=-1, oob_score=False, random_state=None,
           verbose=0, warm_start=False)
```

fit 와 predict 작업을 core 개수 만큼 할당 하도록 n_jobs=-1로 설정한다

교차검증 MSE방식

In [102]:

```
score1 = cross_val_score(model,x_train,y_train,cv=20,
                         scoring="neg_mean_absolute_error").mean()
print(score1)
```

```
-32.2200376172
```

모델을 만든후, 주어진 데이터를 이용하여 모델을 평가한다
해당 데이터를 20조각으로 나누어 오차평균을 구하고 이를 통해 모델을 평가한다

생성한 모델을 선택한 변수의 데이터에 학습시킨다

In [103]:

```
model.fit(x_train,y_train)
```

Out[103]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_split=1e-07, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           n_estimators=500, n_jobs=-1, oob_score=False, random_state=None,
           verbose=0, warm_start=False)
```

학습시킨 모델로 predict 한다

In [104]:

```
predictions=model.predict(x_test)
print(predictions.shape)
predictions[0:10]
```

```
(6493,)
```

Out[104]:

```
array([  11.466     ,    4.222     ,    3.286     ,    3.934     ,
          3.406     ,    7.006     ,   40.72      ,  106.336     ,
        240.566     ,  129.64466667])
```

교차검증 rmsle 방식

Score = Root Mean Squared Logarithmic Error, RMSLE.

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i+1)-\log(a_i+1))^2}$$

In [105]:

```
import numpy as np
from sklearn.metrics import make_scorer

def rmsle(predict, actual):
    predict = np.array(predict)
    actual = np.array(actual)
    log_predict = np.log(predict + 1)
    log_actual = np.log(actual + 1)
    difference = log_predict - log_actual
    square_difference = difference ** 2
    mean_square_difference = square_difference.mean()
    score = np.sqrt(mean_square_difference)
    return score

rmsle_score = make_scorer(rmsle)
rmsle_score
```

Out[105]:

```
make_scorer(rmsle)
```

Log 씌우기

In [106]:

```
y_train = np.log(y_train + 1)
print(y_train.shape)
y_train.head()
```

```
(10886,)
```

Out[106]:

```
0    2.833213
1    3.713572
2    3.496508
3    2.639057
4    0.693147
Name: count, dtype: float64
```

DecisionTreeRegressor : Regression 모델

In [107]:

```
from sklearn.tree import DecisionTreeRegressor

model = DecisionTreeRegressor(random_state=37)
model
```

Out[107]:

```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_impurity_split=1e-07,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, presort=False, random_state=37,
           splitter='best')
```

RandomForest : Regression 모델

In [108]:

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators=500,n_jobs=-1)
model
```

Out[108]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_split=1e-07, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           n_estimators=500, n_jobs=-1, oob_score=False, random_state=None,
           verbose=0, warm_start=False)
```

xgboost 모델

In [109]:

```
import xgboost as xgb
from xgboost.sklearn import XGBRegressor

model = xgb.XGBRegressor(max_depth=4,min_child_weight=6,gamma=0.4,colsample_bytree=0.6,subsample
=0.6,n_estimators=600)
model
```

Out[109]:

```
XGBRegressor(base_score=0.5, colsample_bylevel=1, colsample_bytree=0.6,
       gamma=0.4, learning_rate=0.1, max_delta_step=0, max_depth=4,
       min_child_weight=6, missing=None, n_estimators=600, nthread=-1,
       objective='reg:linear', reg_alpha=0, reg_lambda=1,
       scale_pos_weight=1, seed=0, silent=True, subsample=0.6)
```

교차검증

In [110]:

```
score2 = cross_val_score(model, x_train, y_train, cv=20, scoring=rmsle_score).mean()
print(score2)
```

```
0.0831943818229
```

생성한 모델을 선택한 변수의 데이터에 학습시킨다