


인공지능 - 변영철 교수님

BIKE SHARING DEMAND

인공지능

전산통계학과 2017108311

양준혁



CONTENTS



01 개요/필요성

02 데이터 라이브러리

03 데이터 소개

04 데이터 시각화

05 데이터 전처리

06 데이터 학습 및 테스트



01

개요

자전거 수요 예측

자전거 수요에 영향을 미치는 요인들을 알아보고 자전거 대여량을 예측하는 목표로함

필요성

최근 전 세계 많은 도시는 교통량 및 대기오염을 감축하기 위해 공유자전거 시스템을 도입하여 운영하고 있고, 서울시에서도 2015년부터 따릉이 공유자전거 서비스를 제공하고 있다. 공유자전거의 사용이 확산됨에 따라 대여소별로 자전거의 수요는 증가하고 있으나, 제한된 예산 하에서 대여소별로 수요를 관리하기 때문에 운영 및 관리상의 어려움이 존재한다. 현재 자전거 재배치를 통해 대여소별로 수요의 변동을 해결하려고 노력하고 있으나, 불확실한 미래의 사용자 수요를 정확히 예측하는 것이 보다 근본적





02

데이터 라이브러리



큰 규모의 다차원 배열과 수치 연산을 효율적으로 처리해주는 라이브러리



데이터를 분석하거나 통계처리해주는 라이브러리



데이터를 시각화해주는 라이브러리



데이터를 다양하게 시각화하는데 도움을 주는 라이브러리



다양한 머신러닝 알고리즘이 구현된 기계 학습 라이브러리



파이썬을 기반으로 과학적 컴퓨팅 영역의 여러 기본적인 작업을 위한 라이브러리

데이터 라이브러리

```
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
```

```
# 노트북 안에 그래프를 그리기 위해
%matplotlib inline
```

```
# 그래프에서 격자로 숫자 범위가 눈에 잘 띄도록 ggplot 스타일을 사용
plt.style.use('ggplot')
```

```
# 그래프에서 마이너스 폰트 깨지는 문제에 대한 대처
mpl.rcParams['axes.unicode_minus'] = False
```



03

데이터 소개


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

csv파일을 불러오고

(10886, 12)

```
# train.columns
# train.dtypes
train.info()
```

info 함수로 파일의 전반적 내용을 확인

```
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
datetime      10886 non-null datetime64[ns]
season        10886 non-null int64
holiday       10886 non-null int64
workingday    10886 non-null int64
weather       10886 non-null int64
temp          10886 non-null float64
atemp         10886 non-null float64
humidity      10886 non-null int64
windspeed     10886 non-null float64
casual        10886 non-null int64
registered    10886 non-null int64
count         10886 non-null int64
dtypes: datetime64[ns](1), float64(3), int64(8)
memory usage: 1020.6 KB
```

10886의 행과 12개의 columns로 이루어져 있다

데이터 소개

자전거 수요 예측

```
train.head()
```

| | datetime | season | holiday | workingday | weather | temp | atemp | humidity | windspeed | casual | registered | count |
|---|---------------------|--------|---------|------------|---------|------|--------|----------|-----------|--------|------------|-------|
| 0 | 2011-01-01 00:00:00 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 81 | 0.0 | 3 | 13 | 16 |
| 1 | 2011-01-01 01:00:00 | 1 | 0 | 0 | 1 | 9.02 | 13.635 | 80 | 0.0 | 8 | 32 | 40 |
| 2 | 2011-01-01 02:00:00 | 1 | 0 | 0 | 1 | 9.02 | 13.635 | 80 | 0.0 | 5 | 27 | 32 |
| 3 | 2011-01-01 03:00:00 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 75 | 0.0 | 3 | 10 | 13 |
| 4 | 2011-01-01 04:00:00 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 75 | 0.0 | 0 | 1 | 1 |

- datetime - 시간별 날짜 + 타임스탬프
- season - 1 = 봄, 2 = 여름, 3 = 가을, 4 = 겨울
- holiday - 해당 날짜가 휴일인지
- workingday - 주말이나 휴일이 아닌 경우
- weather
- 1: 맑음, 구름이 적음, 부분적으로 흐림, 부분적으로 흐림
 - 2: 안개 + 흐림, 안개 + 깨진 구름, 안개 + 적은 구름, 안개
 - 3: 가벼운 눈, 가벼운 비 + 뇌우 + 흩어진 구름, 가벼운 비 + 흩어진 구름
 - 4: 폭우 + 얼음 팔레트 + 뇌우 + 안개, 눈 + 안개

- temp - 섭씨 온도
- atemp - 섭씨로 표시된 "느낌" 온도
- humidity - 상대 습도
- windspeed - 풍속
- casual - 시작된 등록되지 않은 사용자 렌탈 수
- registered - 시작된 등록된 사용자 렌탈 수
- count - 총 대여 횟수

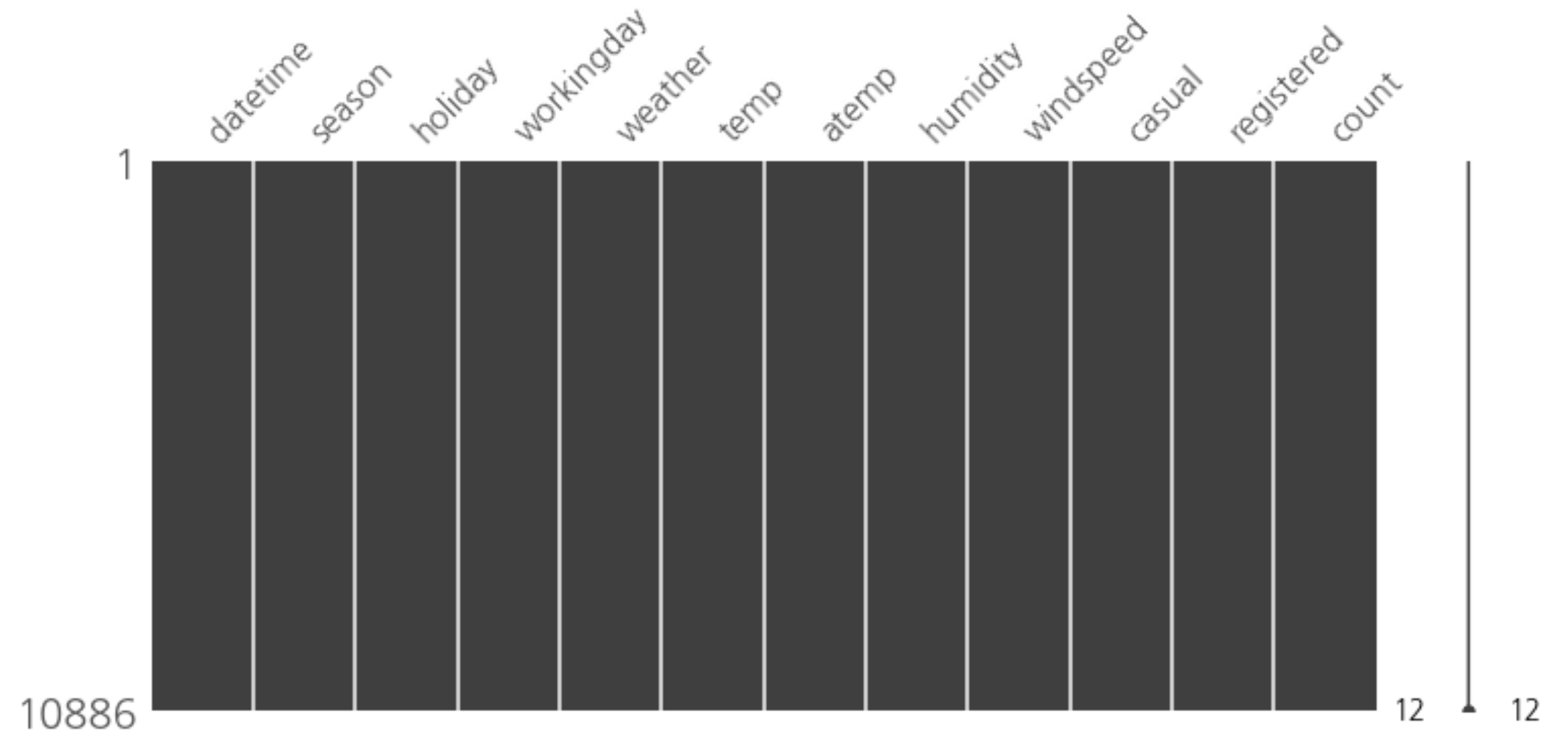
```
train.isnull().sum()
```

```
datetime    0
season      0
holiday     0
workingday  0
weather     0
temp       0
atemp      0
humidity    0
windspeed   0
casual      0
registered  0
count       0
dtype: int64
```

isnull()함수로
결측값 유무 확인
-> 없음

```
import missingno as msno

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



null값 유무 시각
화 Tool 이용
-> 없음



04

데이터 시각화

```
train["year"] = train["datetime"].dt.year
train["month"] = train["datetime"].dt.month
train["day"] = train["datetime"].dt.day
train["hour"] = train["datetime"].dt.hour
train["minute"] = train["datetime"].dt.minute
train["second"] = train["datetime"].dt.second
train.shape
```

(10886, 18)

```
train.head()
```

| | datetime | season | holiday | workingday | weather | temp | atemp | humidity | windspeed | casual | registered | count | year | month | day | hour | minute | second |
|---|---------------------|--------|---------|------------|---------|------|--------|----------|-----------|--------|------------|-------|------|-------|-----|------|--------|--------|
| 0 | 2011-01-01 00:00:00 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 81 | 0.0 | 3 | 13 | 16 | 2011 | 1 | 1 | 0 | 0 | 0 |
| 1 | 2011-01-01 01:00:00 | 1 | 0 | 0 | 1 | 9.02 | 13.635 | 80 | 0.0 | 8 | 32 | 40 | 2011 | 1 | 1 | 1 | 0 | 0 |
| 2 | 2011-01-01 02:00:00 | 1 | 0 | 0 | 1 | 9.02 | 13.635 | 80 | 0.0 | 5 | 27 | 32 | 2011 | 1 | 1 | 2 | 0 | 0 |
| 3 | 2011-01-01 03:00:00 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 75 | 0.0 | 3 | 10 | 13 | 2011 | 1 | 1 | 3 | 0 | 0 |
| 4 | 2011-01-01 04:00:00 | 1 | 0 | 0 | 1 | 9.84 | 14.395 | 75 | 0.0 | 0 | 1 | 1 | 2011 | 1 | 1 | 4 | 0 | 0 |

train 데이터에 시각화 하기 편하게 년 월 일 시간 분 초 columns 추가

```
figure, ((ax1,ax2,ax3), (ax4,ax5,ax6)) = plt.subplots(nrows=2, ncols=3)
figure.set_size_inches(18,8)
```

```
sns.barplot(data=train, x="year", y="count", ax=ax1)
sns.barplot(data=train, x="month", y="count", ax=ax2)
sns.barplot(data=train, x="day", y="count", ax=ax3)
sns.barplot(data=train, x="hour", y="count", ax=ax4)
sns.barplot(data=train, x="minute", y="count", ax=ax5)
sns.barplot(data=train, x="second", y="count", ax=ax6)
```

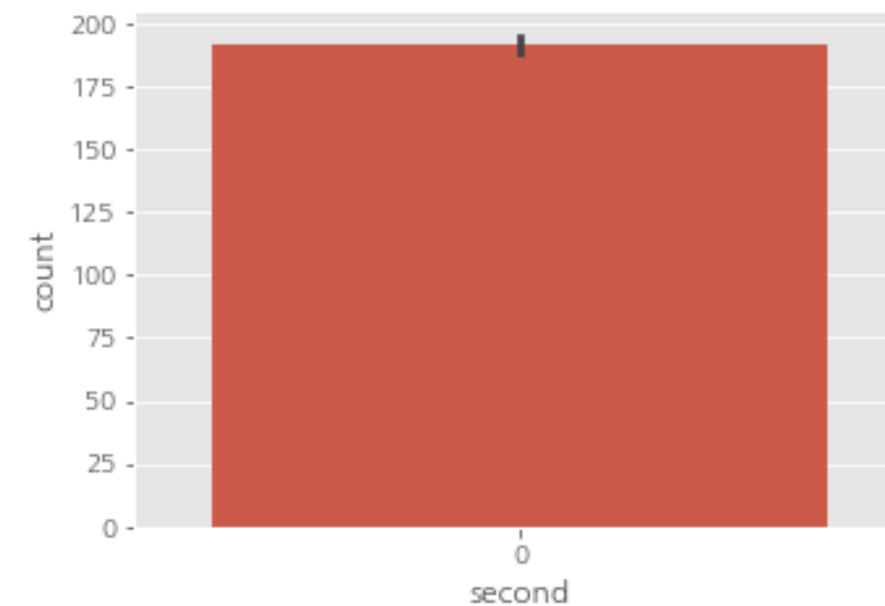
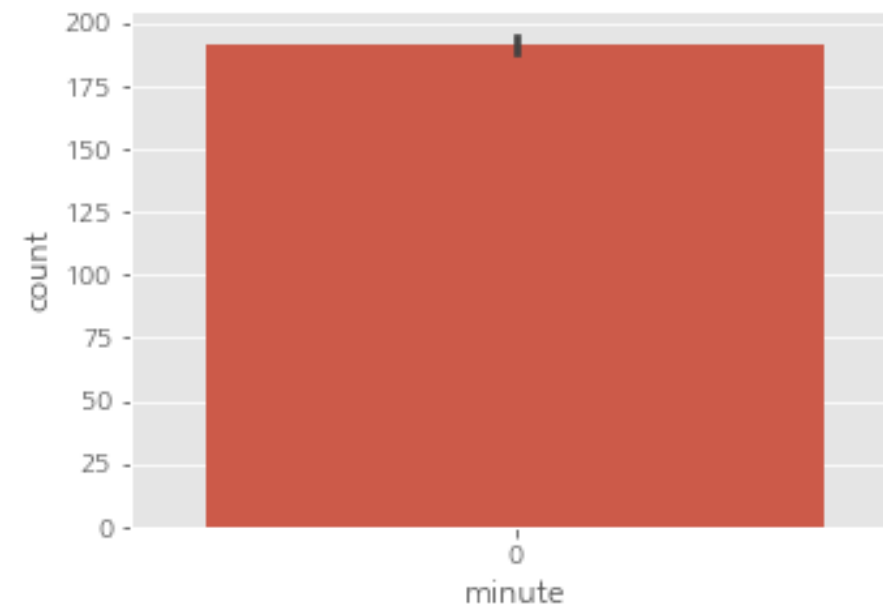
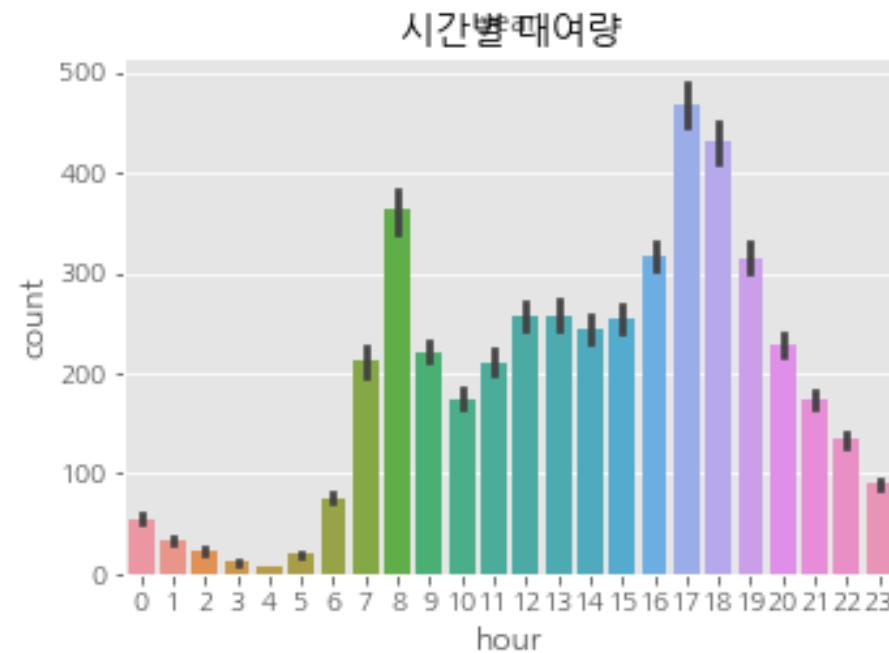
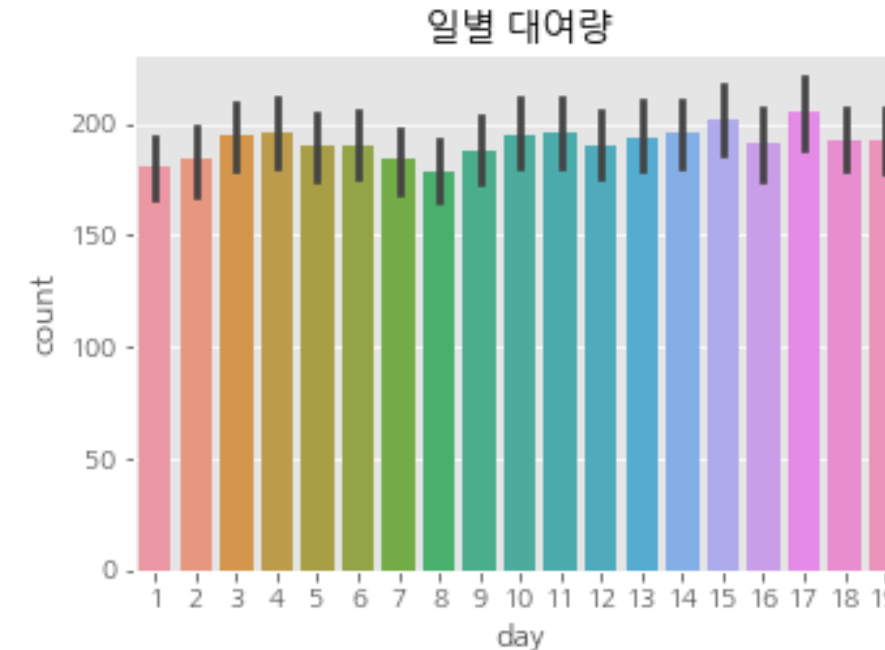
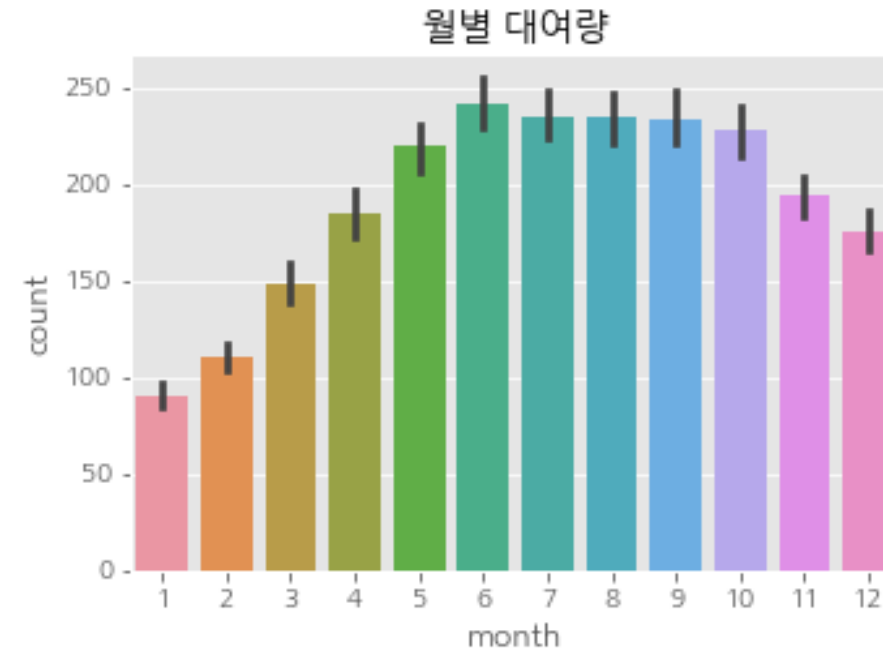
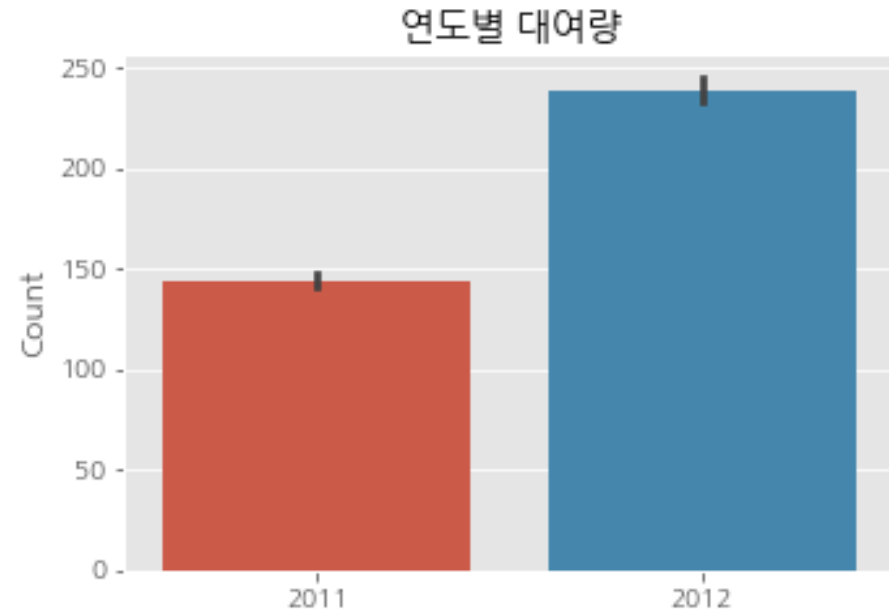
```
ax1.set(ylabel='Count',title="연도별 대여량")
ax2.set(xlabel='month',title="월별 대여량")
ax3.set(xlabel='day', title="일별 대여량")
ax4.set(xlabel='hour', title="시간별 대여량")
```

[Text(0.5,0,'hour'), Text(0.5,1,'시간별 대여량')]

연도별 대여량은 2011년 보다 2012년이 더 많다.

월별 대여량은 6월에 가장 많고 7~10월도 대여량이 많다. 그리고 1월에 가장 적다.

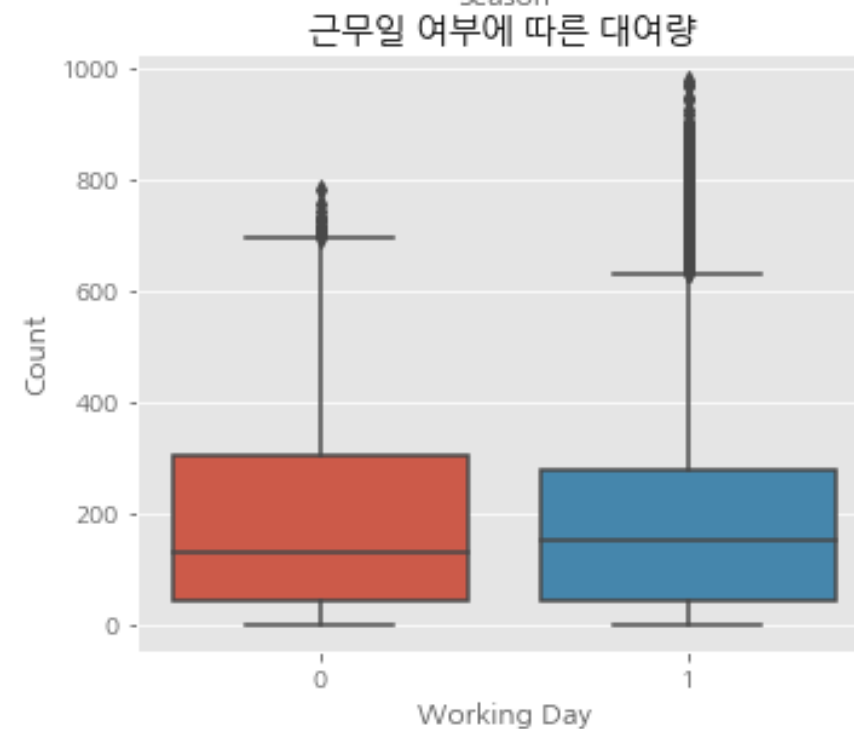
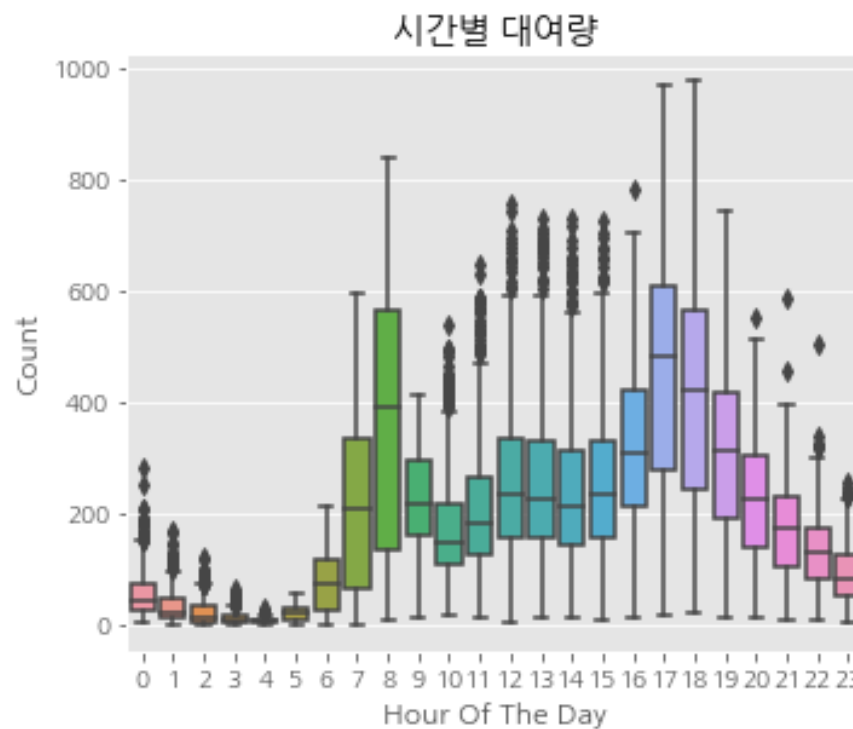
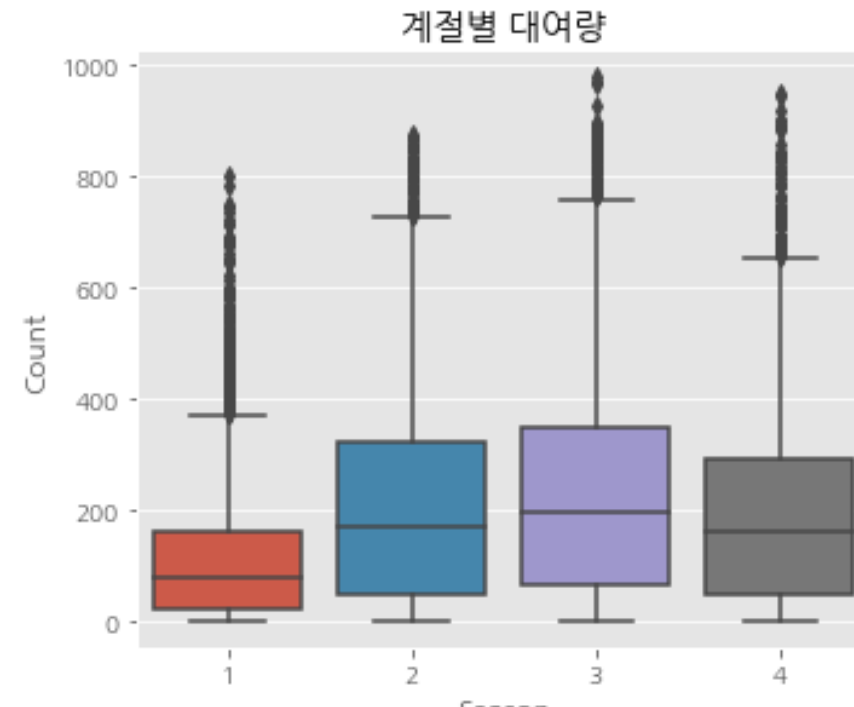
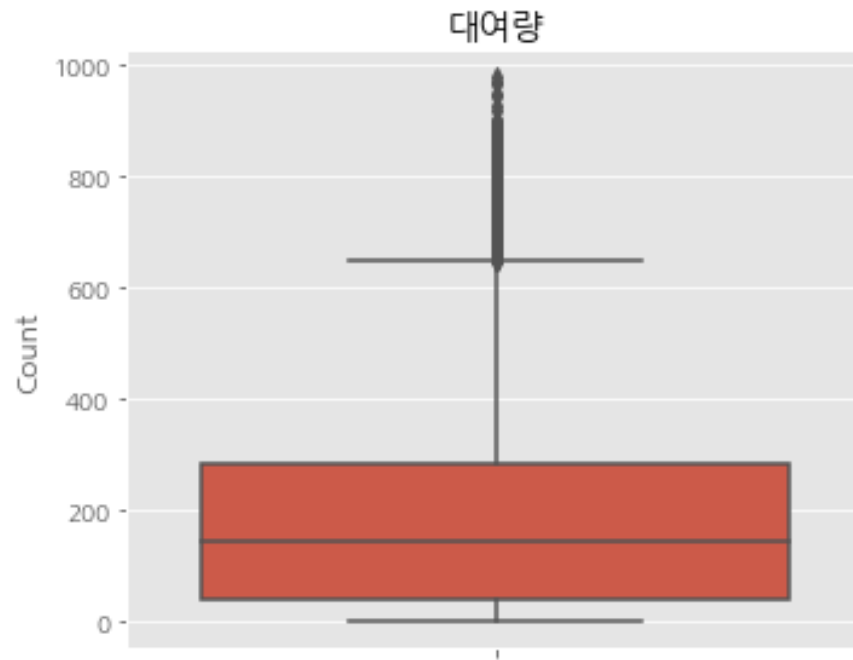
시간 대 대여량을 보면 출퇴근 시간에 대여량이 많은 것 같다. 하지만 주말과 나누어 볼 필요가 있을 것 같다. 분, 초도 다 0이기 때문에 의미가 없다.



```
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="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="대여량")
axes[0][1].set(xlabel='Season', ylabel='Count',title="계절별 대여량")
axes[1][0].set(xlabel='Hour Of The Day', ylabel='Count',title="시간별 대여량")
axes[1][1].set(xlabel='Working Day', ylabel='Count',title="근무일 여부에 따른 대여량")
```

[Text(0,0.5,'Count'), Text(0.5,0,'Working Day'), Text(0.5,1,'근무일 여부에 따른 대여량')]



boxplot
특정 시간에 몰려있고,
봄이 가장 적고,
시간별 대여는 위 그래프와 비슷
휴일의 경우가 조금 더 대여량 많음

```
train["dayofweek"] = train["datetime"].dt.dayofweek
train.shape
```

```
(10886, 19)
```

```
train["dayofweek"].value_counts()
```

```
5    1584
6    1579
3    1553
2    1551
0    1551
1    1539
4    1529
Name: dayofweek, dtype: int64
```

dayofweek
0-6:월-금

```
fig, (ax1, ax2, ax3, ax4, ax5) = plt.subplots(nrows=5)
fig.set_size_inches(18, 25)

sns.pointplot(data=train, x="hour", y="count", ax=ax1)

sns.pointplot(data=train, x="hour", y="count", hue="workingday", ax=ax2)

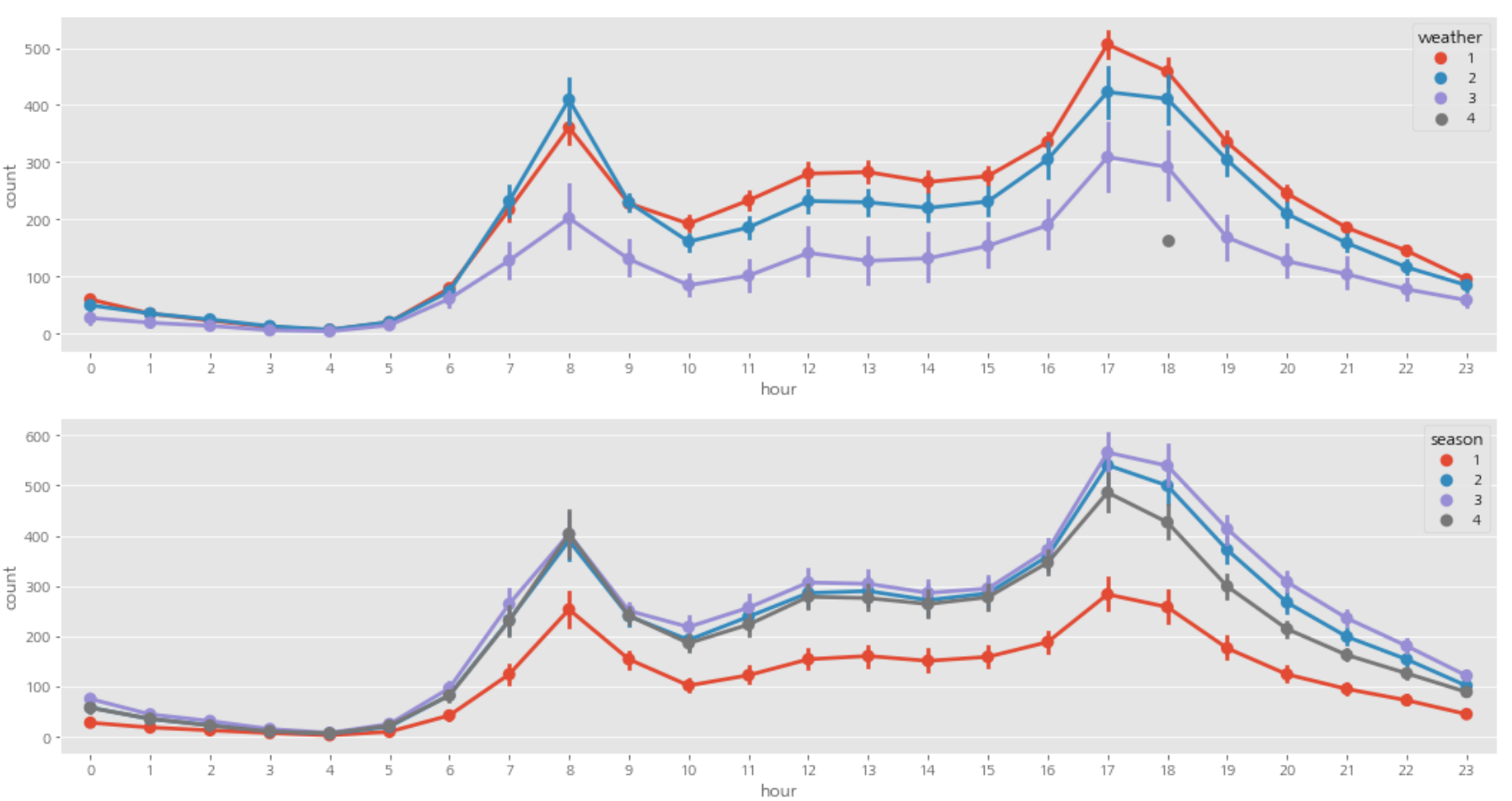
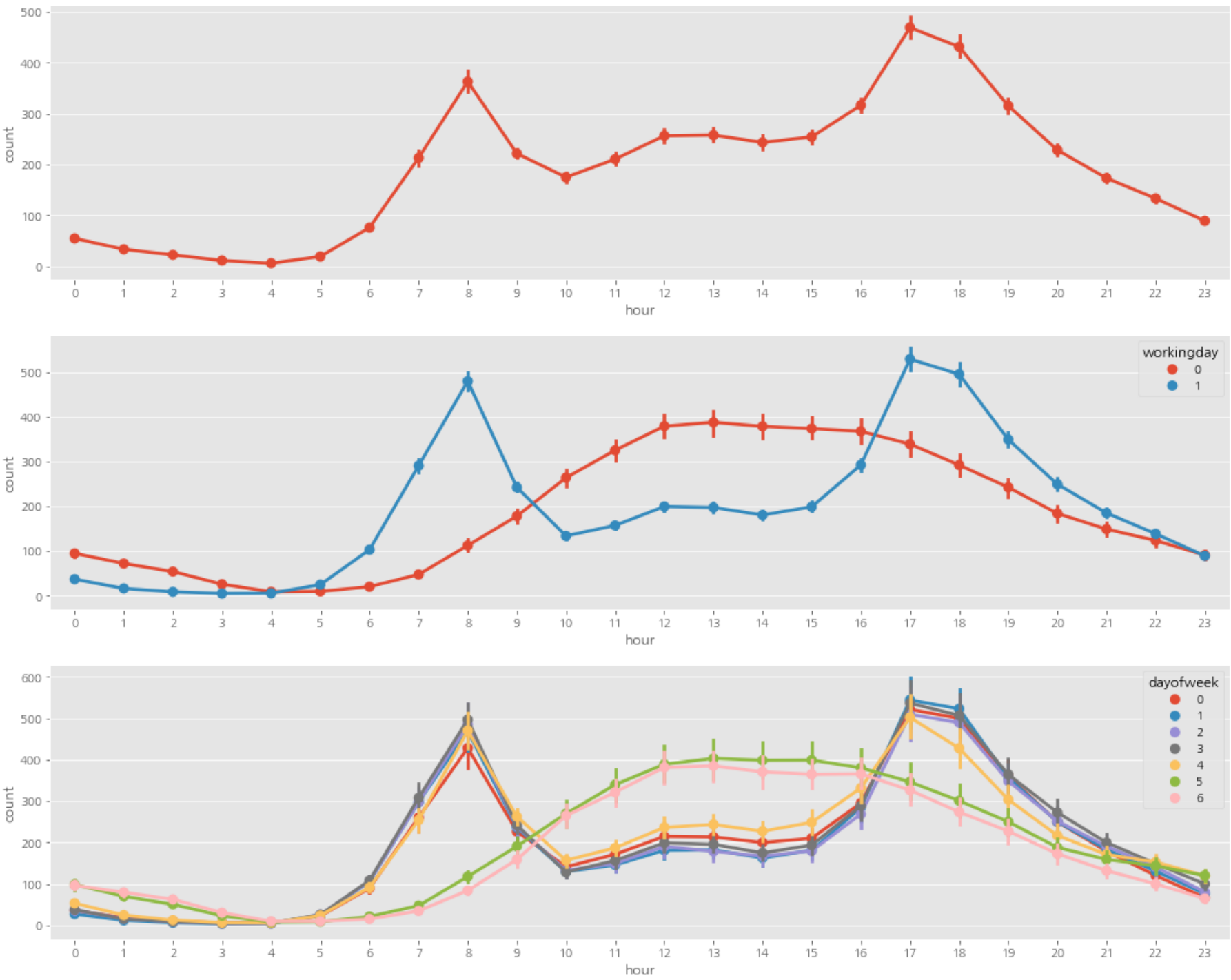
sns.pointplot(data=train, x="hour", y="count", hue="dayofweek", ax=ax3)

sns.pointplot(data=train, x="hour", y="count", hue="weather", ax=ax4)

sns.pointplot(data=train, x="hour", y="count", hue="season", ax=ax5)
```

pointplot으로 시각화

pointplot으로 시각화



출퇴근 시간에 명확히 더 대여하는것으로 보임
workingday가 아닌 휴일에는 점심시간대 많은 대여량
토요일과 일요일은 위 휴일 그래프와 비슷
weather 1234
season 1234 봄에 가장 적게 빌림 시간은 출퇴근 시간

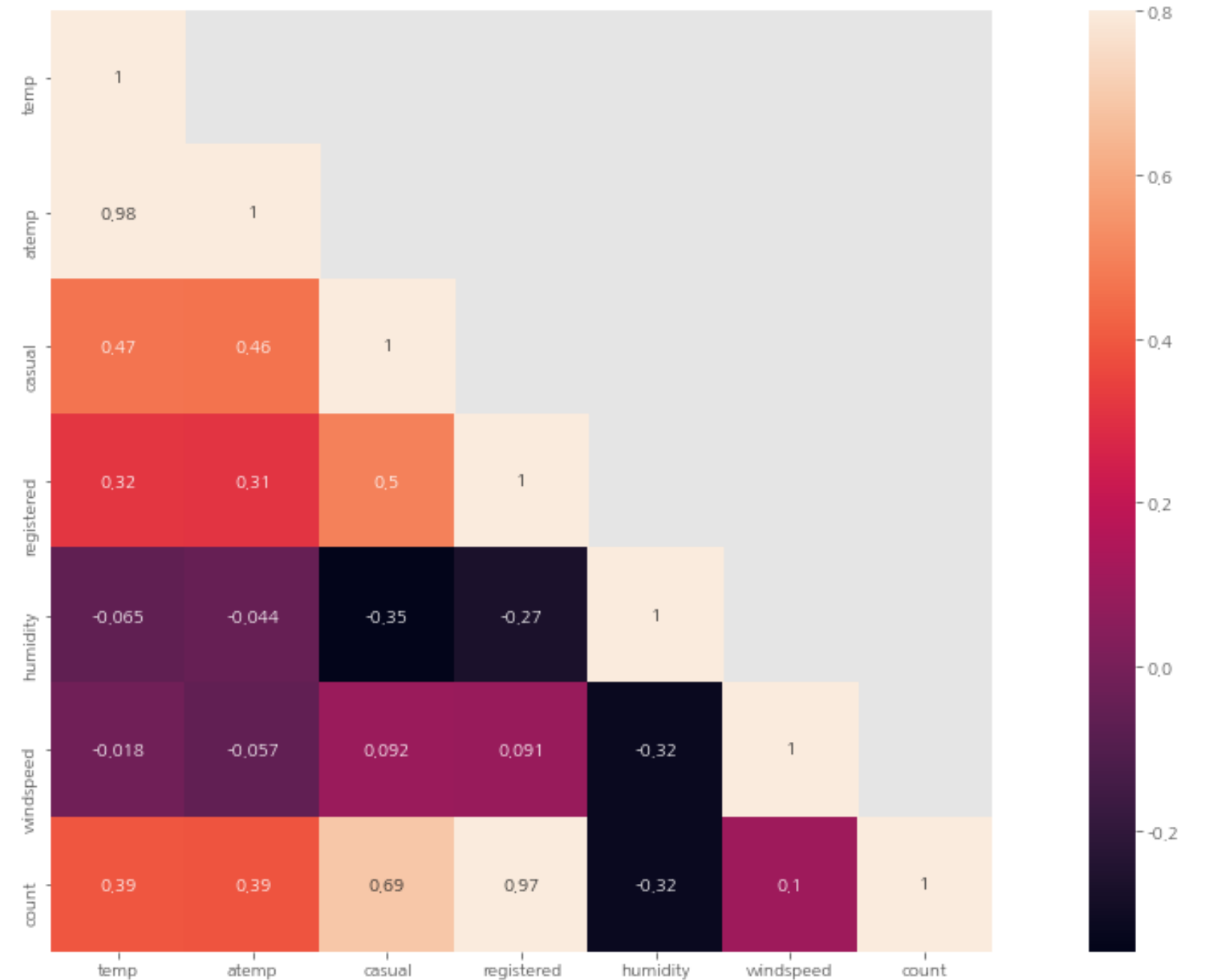
```
corrMatt = train[["temp", "atemp", "casual", "registered", "humidity", "windspeed", "count"]]
corrMatt = corrMatt.corr()
print(corrMatt)
```

```
mask = np.array(corrMatt)
mask[np.tril_indices_from(mask)] = False
```

| | temp | atemp | casual | registered | humidity | windspeed | # |
|------------|-----------|-----------|-----------|------------|-----------|-----------|---|
| temp | 1.000000 | 0.984948 | 0.467097 | 0.318571 | -0.064949 | -0.017852 | |
| atemp | 0.984948 | 1.000000 | 0.462067 | 0.314635 | -0.043536 | -0.057473 | |
| casual | 0.467097 | 0.462067 | 1.000000 | 0.497250 | -0.348187 | 0.092276 | |
| registered | 0.318571 | 0.314635 | 0.497250 | 1.000000 | -0.265458 | 0.091052 | |
| humidity | -0.064949 | -0.043536 | -0.348187 | -0.265458 | 1.000000 | -0.318607 | |
| windspeed | -0.017852 | -0.057473 | 0.092276 | 0.091052 | -0.318607 | 1.000000 | |
| count | 0.394454 | 0.389784 | 0.690414 | 0.970948 | -0.317371 | 0.101369 | |

| | count |
|------------|-----------|
| temp | 0.394454 |
| atemp | 0.389784 |
| casual | 0.690414 |
| registered | 0.970948 |
| humidity | -0.317371 |
| windspeed | 0.101369 |
| count | 1.000000 |

```
fig, ax = plt.subplots()
fig.set_size_inches(20,10)
sns.heatmap(corrMatt, mask=mask, vmax=.8, square=True, annot=True)
```



온도, 습도, 풍속은 거의 연관관계가 없다.

대여량과 가장 연관이 높은 건 registered 로 등록 된 대여자가 많지만, test 데이터에는 이 값이 없다. atemp와 temp는 0.98로 상관관계가 높지만 온도와 체감온도로 피쳐로 사용하기에 적합하지 않을 수 있다.

```
fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2)
fig.set_size_inches(18, 4)

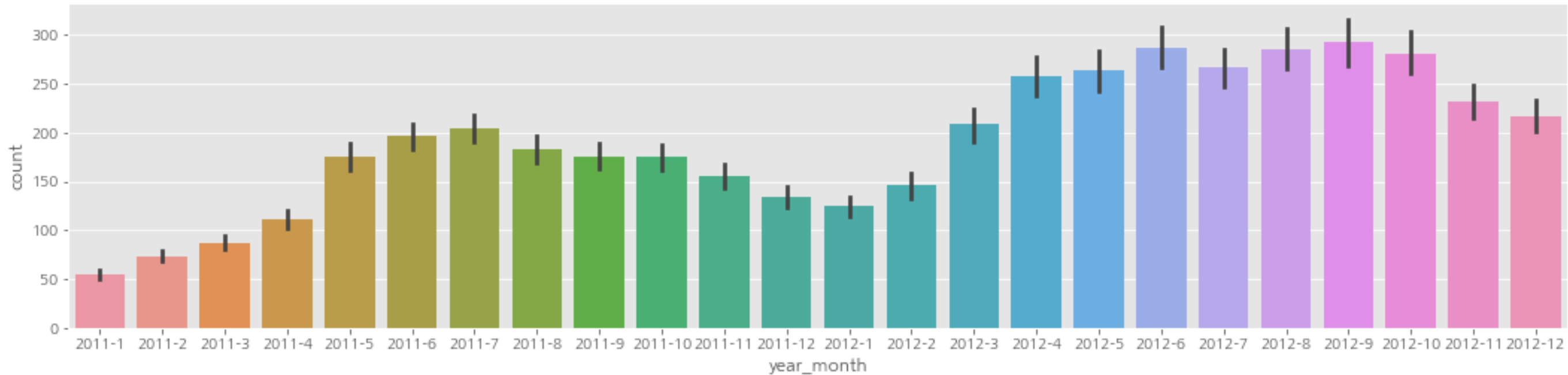
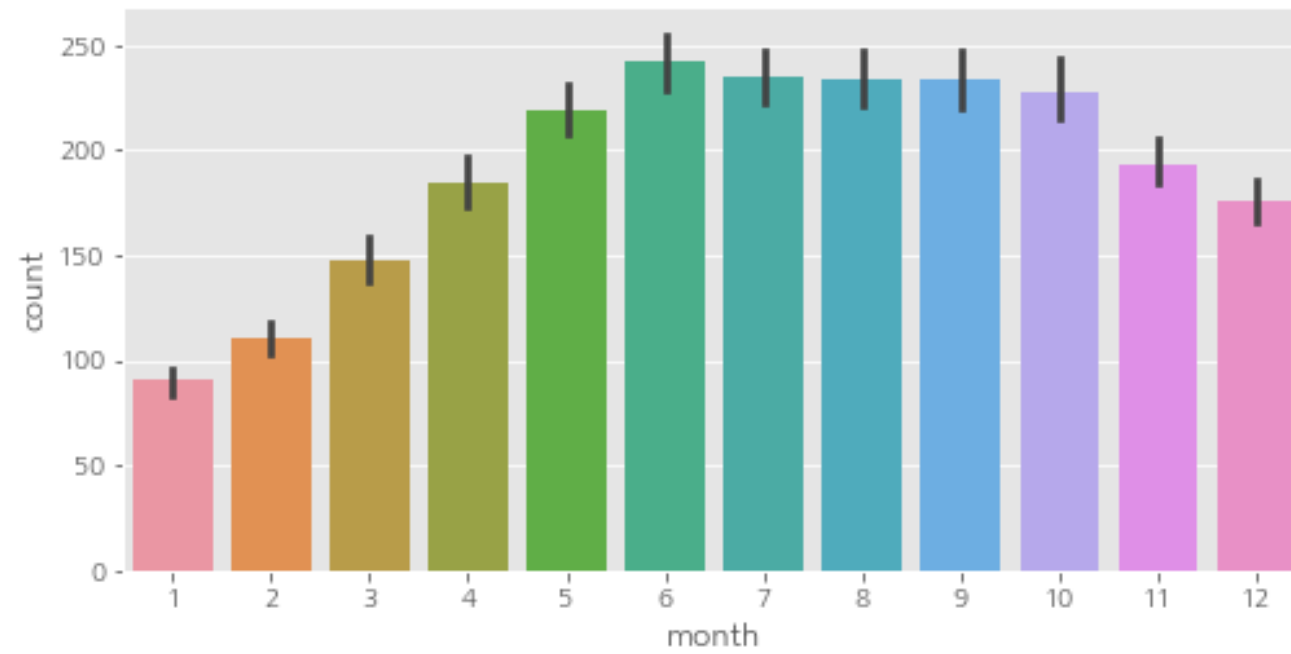
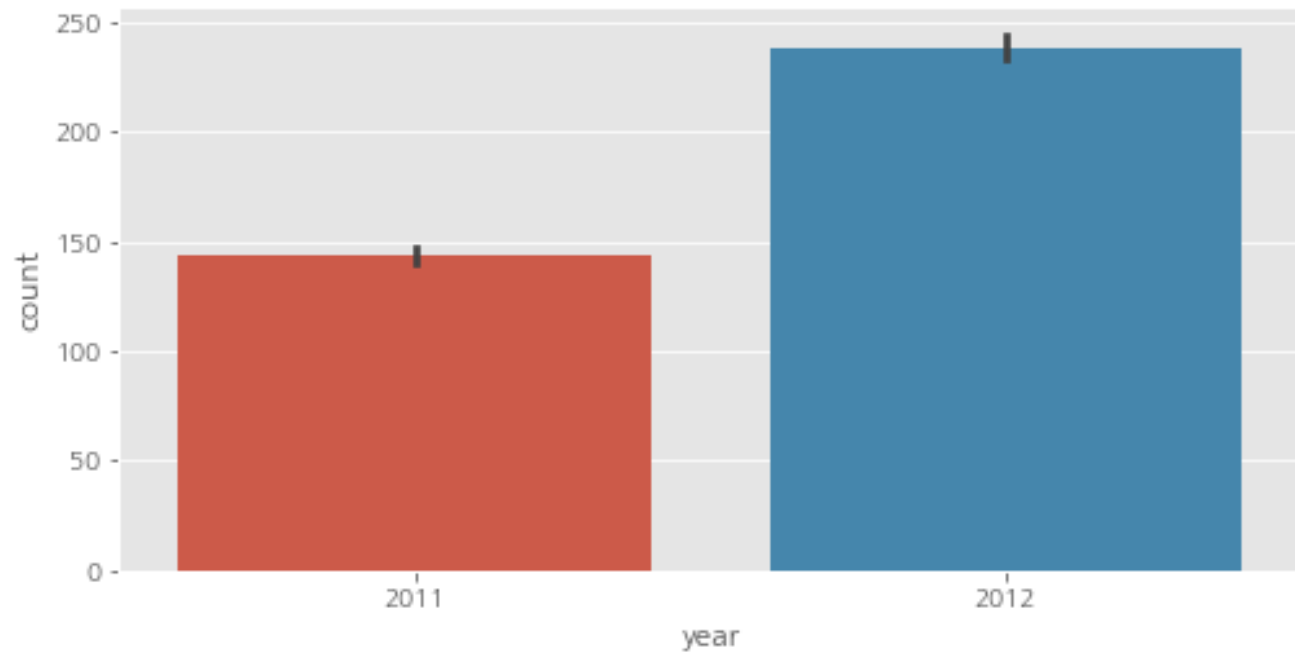
sns.barplot(data=train, x="year", y="count", ax=ax1)
sns.barplot(data=train, x="month", y="count", ax=ax2)

fig, ax3 = plt.subplots(nrows=1, ncols=1)
fig.set_size_inches(18, 4)

sns.barplot(data=train, x="year_month", y="count", ax=ax3)
```

2011년보다 2012년의 대여량이 더 많다.
겨울보다는 여름에 대여량이 많다.

2011년과 2012년의 월별 데이터를 이어보면 전체적으로 증가하는 추세이다.





05

데이터 전처리

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

(10886, 12)

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

(6493, 9)

```
train["year"] = train["datetime"].dt.year
train["month"] = train["datetime"].dt.month
train["hour"] = train["datetime"].dt.hour
train["dayofweek"] = train["datetime"].dt.dayofweek
train.shape
```

(10886, 16)

```
test["year"] = test["datetime"].dt.year
test["month"] = test["datetime"].dt.month
test["hour"] = test["datetime"].dt.hour
test["dayofweek"] = test["datetime"].dt.dayofweek
test.shape
```

(6493, 13)

사용할 피처 year, month, hour, dayofweek
추가

```
# 연속형 feature와 범주형 feature
# 범주형 feature의 type을 category로 변경 해 준다.
categorical_feature_names = ["season", "holiday", "workingday", "weather",
                             "dayofweek", "month", "year", "hour"]
```

```
for var in categorical_feature_names:
    train[var] = train[var].astype("category")
    test[var] = test[var].astype("category")
```

```
feature_names = ["season", "weather", "temp", "atemp", "humidity",
                 "year", "hour", "dayofweek", "holiday", "workingday"]
```

feature_names

```
['season',
 'weather',
 'temp',
 'atemp',
 'humidity',
 'year',
 'hour',
 'dayofweek',
 'holiday',
 'workingday']
```

선택된 피처

```
X_train = train[feature_names]

print(X_train.shape)
X_train.head()
```

(10886, 10)

| | season | weather | temp | atemp | humidity | year | hour | dayofweek | holiday | workingday |
|---|--------|---------|------|--------|----------|------|------|-----------|---------|------------|
| 0 | 1 | 1 | 9.84 | 14.395 | 81 | 2011 | 0 | 5 | 0 | 0 |
| 1 | 1 | 1 | 9.02 | 13.635 | 80 | 2011 | 1 | 5 | 0 | 0 |
| 2 | 1 | 1 | 9.02 | 13.635 | 80 | 2011 | 2 | 5 | 0 | 0 |
| 3 | 1 | 1 | 9.84 | 14.395 | 75 | 2011 | 3 | 5 | 0 | 0 |
| 4 | 1 | 1 | 9.84 | 14.395 | 75 | 2011 | 4 | 5 | 0 | 0 |

```
X_test = test[feature_names]

print(X_test.shape)
X_test.head()
```

(6493, 10)

| | season | weather | temp | atemp | humidity | year | hour | dayofweek | holiday | workingday |
|---|--------|---------|-------|--------|----------|------|------|-----------|---------|------------|
| 0 | 1 | 1 | 10.66 | 11.365 | 56 | 2011 | 0 | 3 | 0 | 1 |
| 1 | 1 | 1 | 10.66 | 13.635 | 56 | 2011 | 1 | 3 | 0 | 1 |
| 2 | 1 | 1 | 10.66 | 13.635 | 56 | 2011 | 2 | 3 | 0 | 1 |
| 3 | 1 | 1 | 10.66 | 12.880 | 56 | 2011 | 3 | 3 | 0 | 1 |
| 4 | 1 | 1 | 10.66 | 12.880 | 56 | 2011 | 4 | 3 | 0 | 1 |

```
label_name = "count"

y_train = train[label_name]

print(y_train.shape)
y_train.head()
```

(10886,)

| | |
|---|----|
| 0 | 16 |
| 1 | 40 |
| 2 | 32 |
| 3 | 13 |
| 4 | 1 |

Name: count, dtype: int64

새로운 데이터 프레임
X_train, X_test, y_train

```

from sklearn.metrics import make_scorer

def rmsle(predicted_values, actual_values, convertExp=True):

    if convertExp:
        predicted_values = np.exp(predicted_values),
        actual_values = np.exp(actual_values)

    # 넘파이로 배열 형태로 바꿔준다.
    predicted_values = np.array(predicted_values)
    actual_values = np.array(actual_values)

    # 예측값과 실제 값에 1을 더하고 로그를 씌워준다.
    # 값이 0일 수도 있어서 로그를 취했을 때 마이너스 무한대가 될 수도 있기 때문에 1을 더해 줌
    # 로그를 씌워주는 것은 정규분포로 만들어주기 위해
    log_predict = np.log(predicted_values + 1)
    log_actual = np.log(actual_values + 1)

    # 위에서 계산한 예측값에서 실제값을 빼주고 제곱을 해준다.
    difference = log_predict - log_actual
    difference = np.square(difference)

    # 평균을 낸다.
    mean_difference = difference.mean()

    # 다시 루트를 씌운다.
    score = np.sqrt(mean_difference)

    return score

```

- ④MSLE, ⑤RMSLE : MSE/RMSE에 로그(log) 적용

* MSLE 및 RMSLE는 '로그'의 특성으로 아웃라이어에 강건하며, (실제값/예측값) 상대적 비율을 계산하게 됨

$$MSLE = \frac{1}{n} \sum_{i=1}^n (\log(y_i + 1) - \log(\hat{y}_i + 1))^2$$

$$RMSLE = \sqrt{\frac{1}{n} \sum_{i=1}^n (\log(\hat{y}_i + 1) - \log(y_i + 1))^2}$$

rmsle값

값이 작을수록

-> 회귀 성능이 좋음

->예측을 잘했다는 의미



06

데이터 학습 및 테스트


```

from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.model_selection import GridSearchCV
from sklearn import metrics
import warnings
pd.options.mode.chained_assignment = None
warnings.filterwarnings("ignore", category=DeprecationWarning)

# 선형회귀 모델을 초기화
lModel = LinearRegression()

# 모델을 학습시킨다.
y_train_log = np.log1p(y_train)
lModel.fit(X_train, y_train_log)

# 예측하고 정확도를 평가한다.
preds = lModel.predict(X_train)
print ("RMSLE Value For Linear Regression: ",
        rmsle(np.exp(y_train_log), np.exp(preds), False))

```

RMSLE Value For Linear Regression: 0.9803697923313522

RMSLE = 0.9803

선형회귀 모델 Linear Regression Model

선형회귀 또는 최소제곱법은 가장 간단하고 오래된 회귀용 선형 알고리즘

선형회귀는 예측과 훈련 세트에 있는 타겟 y 사이의 평균제곱오차(MSE)를 최소화하는 파라미터 w 와 b 를 찾는다.

매개변수가 없는 것이 장점이지만, 모델의 복잡도를 제어할 수 없다는 단점이 있다.

```

ridge_m_ = Ridge()
ridge_params_ = { 'max_iter':[3000], 'alpha':[0.01, 0.1, 1, 2, 3, 4, 10, 30, 100, 200, 300, 400, 800, 900, 1000] }
rmsle_scorer = metrics.make_scorer(rmsle, greater_is_better=False)
grid_ridge_m = GridSearchCV( ridge_m_,
                             ridge_params_,
                             scoring = rmsle_scorer,
                             cv=5)

y_train_log = np.log1p(y_train)
grid_ridge_m.fit( X_train, y_train_log )
preds = grid_ridge_m.predict(X_train)
print (grid_ridge_m.best_params_)
print ("RMSLE Value For Ridge Regression: ", rmsle(np.exp(y_train_log), np.exp(preds), False))

df = pd.DataFrame(grid_ridge_m.cv_results_)
df.head()

```

```

{'alpha': 0.01, 'max_iter': 3000}
RMSLE Value For Ridge Regression: 0.9803697902780835

```

RMSLE = 0.9803

릿지 Regularization Model - Ridge

회귀를 위한 선형모델

가중치(w)의 모든 원소가 0에 가깝게
만들어 모든 피처가 주는 영향을 최소화
(기울기를 작게 만듦)

Regularization(정규화)는 오버피팅
(과대적합)이 되지 않도록 모델을 강제
로 제한한다는 의미

max_iter(반복 실행하는 최대 횟수)
는 3000을 넣어주었다.

```
from sklearn.ensemble import RandomForestRegressor
rfModel = RandomForestRegressor(n_estimators=100)

y_train_log = np.log1p(y_train)
rfModel.fit(X_train, y_train_log)

preds = rfModel.predict(X_train)
score = rmsle(np.exp(y_train_log), np.exp(preds), False)
print ("RMSLE Value For Random Forest: ", score)
```

RMSLE Value For Random Forest: 0.1070944699920936

RMSLE = 0.1071

랜덤 포레스트
Random Forest

의사 결정 트리를 개선하여 나온 것

기능을 무작위로 선택하고 관찰

의사 결정트리를 여러 개 만들어 그 결과
를 평균화 한다.

```
from sklearn.ensemble import GradientBoostingRegressor
gbm = GradientBoostingRegressor(n_estimators=4000, alpha=0.01);

y_train_log = np.log1p(y_train)
gbm.fit(X_train, y_train_log)

preds = gbm.predict(X_train)
score = rmsle(np.exp(y_train_log), np.exp(preds), False)
print ("RMSLE Value For Gradient Boost: ", score)
```

RMSLE Value For Gradient Boost: 0.2135740372724937

RMSLE = 0.2135

Ensemble Model - Gradient Boost

여러개의 결정트리를 묶어 강력한 모델을 만드는 또 다른 앙상블 기법

회귀와 분류에 모두 사용할 수 있음

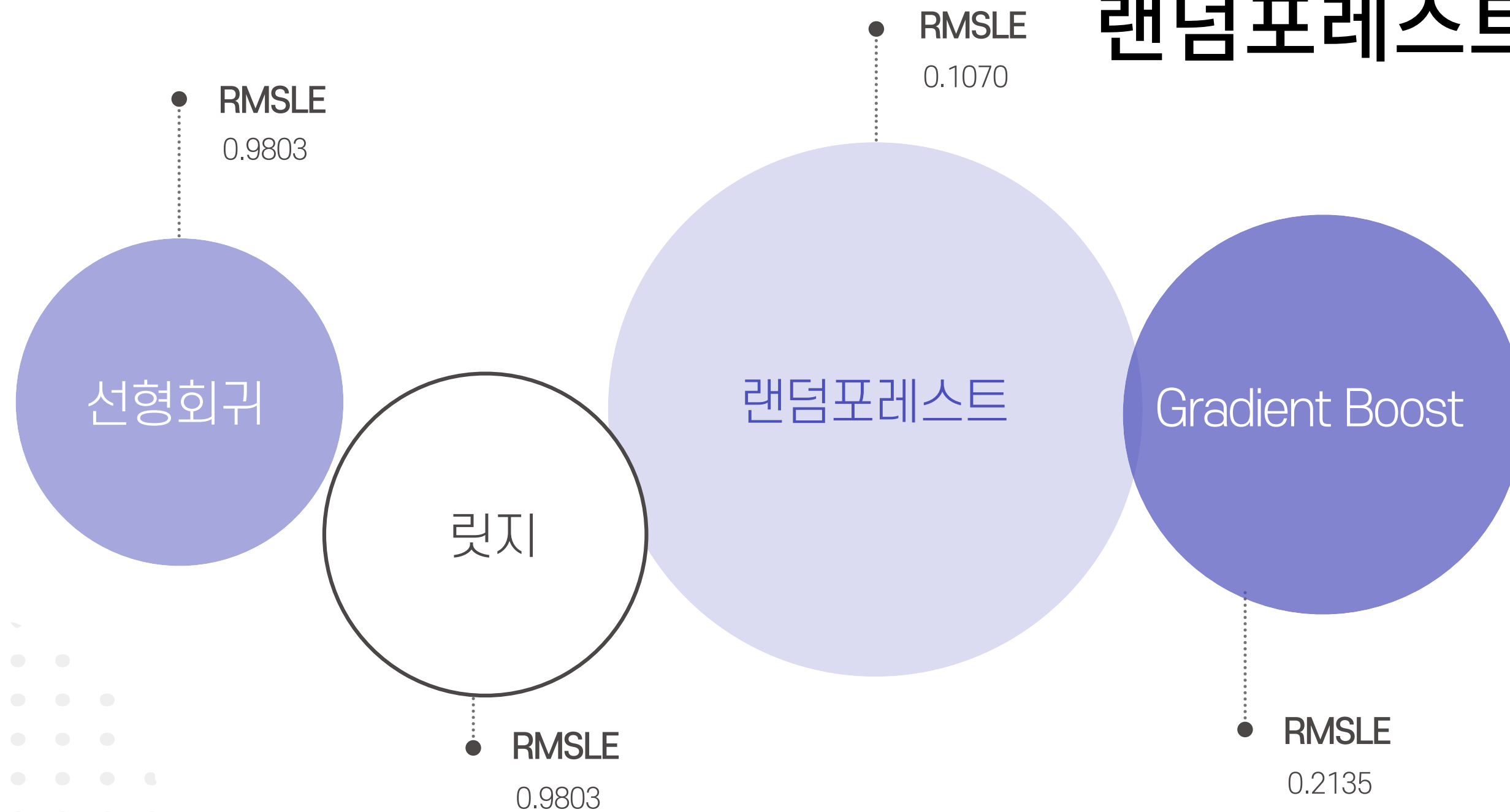
랜덤포레스트와 달리 이진 트리의 오차를 보완하는 방식으로 순차적으로 트리를 만든다.

무작위성이 없고 강력한 사전 가지치기가 사용 됨

1~5개의 깊이 얇은 트리를 사용하기 때문에 메모리를 적게 사용하고 예측이 빠름

결론 및 모델비교

랜덤포레스트가 가장 효율적





소감

최근 데이터 과학 분야의 관심성이 날로 증가하여, 자연스레 인공지능 분야에도 관심이 생겼고, 타학과라 머신러닝에 대한 어려운 부분이 있었지만 직접 코드를 분석하고 실행하는 과정을 통해 머신러닝 알고리즘을 공부할 수 있는 시간이었다.

데이터 분석의 중요성을 다시한번 깨닫고 다양한 분석 방법을 알아나가야겠다고 생각했다.



인공지능-변영철 교수님

THANK
YOU

제주대학교 전산통계학과

양준혁