




자전거 수요량 예측

신승환

대한상공회의소 서울기술교육센터



자전거 수요량 예측

- Bike Sharing Demand
<https://www.kaggle.com/c/bike-sharing-demand>
- 데이터 (count 대여량 정보)를 바탕으로 자전거 대여량 예측
- 도구 : 파이썬

Train 데이터

	A	B	C	D	E	F	G	H	I	J	K	L
1	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
2	2011-01-01 0:00	1	0	0	1	9.84	14.395	81	0	3	13	16
3	2011-01-01 1:00	1	0	0	1	9.02	13.635	80	0	8	32	40
4	2011-01-01 2:00	1	0	0	1	9.02	13.635	80	0	5	27	32
5	2011-01-01 3:00	1	0	0	1	9.84	14.395	75	0	3	10	13
6	2011-01-01 4:00	1	0	0	1	9.84	14.395	75	0	0	1	1
7	2011-01-01 5:00	1	0	0	2	9.84	12.88	75	6.0032	0	1	1
8	2011-01-01 6:00	1	0	0	1	9.02	13.635	80	0	2	0	2
9	2011-01-01 7:00	1	0	0	1	8.2	12.88	86	0	1	2	3
10	2011-01-01 8:00	1	0	0	1	9.84	14.395	75	0	1	7	8
11	2011-01-01 9:00	1	0	0	1	13.12	17.425	76	0	8	6	14
12	2011-01-01 10:00	1	0	0	1	15.58	19.695	76	16.9979	12	24	36
13	2011-01-01 11:00	1	0	0	1	14.76	16.665	81	19.0012	26	30	56
14	2011-01-01 12:00	1	0	0	1	17.22	21.21	77	19.0012	29	55	84
15	2011-01-01 13:00	1	0	0	2	18.86	22.725	72	19.9995	47	47	94
16	2011-01-01 14:00	1	0	0	2	18.86	22.725	72	19.0012	35	71	106
17	2011-01-01 15:00	1	0	0	2	18.04	21.97	77	19.9995	40	70	110
18	2011-01-01 16:00	1	0	0	2	17.22	21.21	82	19.9995	41	52	93

- 자전거 공유 시스템인 워싱턴 DC의 Capital Bikeshare 프로그램 데이터
- (2011년 01월~2012년 12월) 2년 동안 자료

Train 데이터

datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
2011-01-01 0:00	1	0	0	1	9.84	14.395	81	0	3	13	16

- **Datetime** : 날짜 시간
- **Season 계절** : 1 = 봄, 2 = 여름, 3 = 가을, 4 = 겨울
- **Holiday 휴일** : 휴일로 간주되는지 여부
- **Workingday 근무일** : 주말이나 휴일이 아닌지 여부
- **Weather 날씨** : 1 맑음, 흐림 흐림, 부분 흐림, 부분 흐림
2 안개 + 흐림, 안개 + 깨진 구름, 안개 + 약간 구름, 안개
3 밝은 눈, 밝은 비 + 뇌우 + 흩어져있는 구름, 밝은 비 + 흩어져있는 구름
4 폭우 + 얼음 팔레트 + 뇌우 + 안개, 눈 + 안개
- **temp** : 섭씨 온도
- **atemp** : 섭씨 기온
- **humidity** : 상대 습도
- **windspeed** : 풍속
- **Casual** : 등록되지 않은 사용자 대여 수
- **registered** : 등록된 사용자 대여 수
- **count** : 총 대여 수

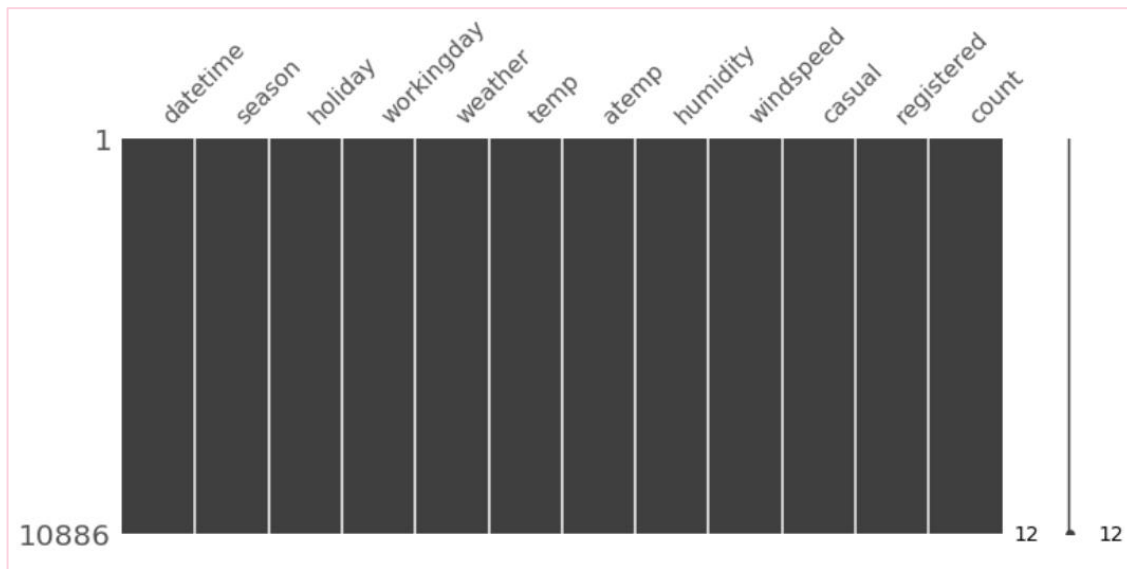
데이터

	A	B	C	D	E	F	G	H	I	J	K	L
1	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
2	2011-01-01 0:00	1	0	0	1	9.84	14.395	81	0	3	13	16
3	2011-01-01 1:00	1	0	0	1	9.02	13.635	80	0	8	32	40
4	2011-01-01 2:00	1	0	0	1	9.02	13.635	80	0	5	27	32
5	2011-01-01 3:00	1	0	0	1	9.84	14.395	75	0	3	10	13

- Train.csv ->
- 2011.01.11. ~ 2012.12.19. 데이터
- 각 월의 1~20일 데이터만 있음

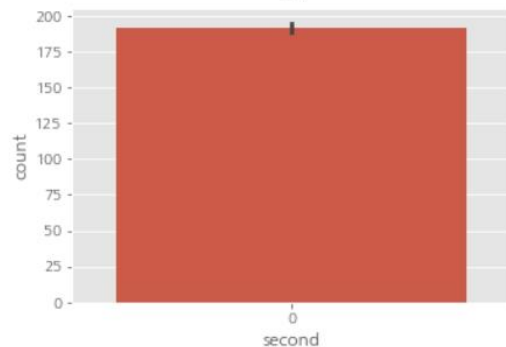
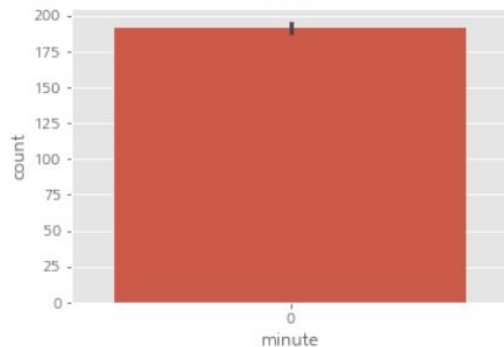
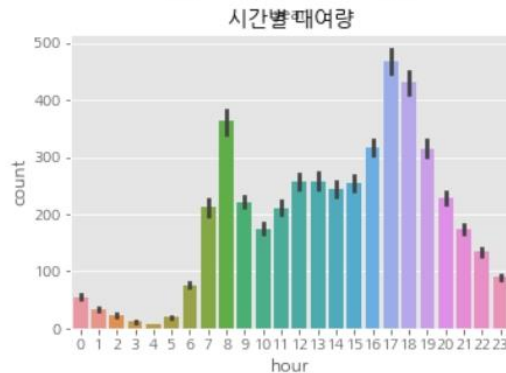
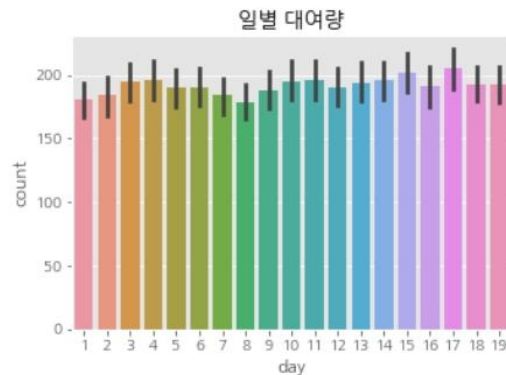
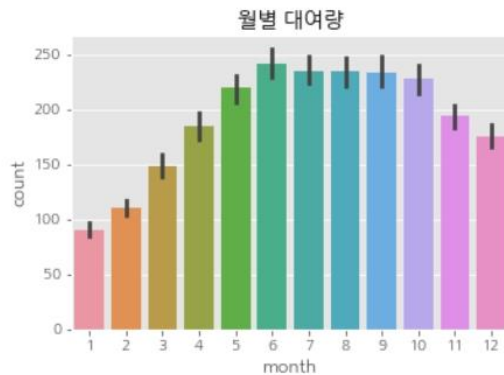
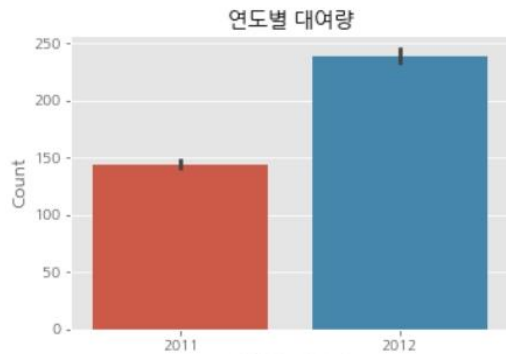
	A	B	C	D	E	F	G	H	I	J	K	L
1	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed			
2	2011-01-20 0:00	1	0	1	1	10.66	11.365	56	26.0027			
3	2011-01-20 1:00	1	0	1	1	10.66	13.635	56	0			
4	2011-01-20 2:00	1	0	1	1	10.66	13.635	56	0			
5	2011-01-20 3:00	1	0	1	1	10.66	12.88	56	11.0014			

- Test.csv ->
- 2011.01.01. ~ 2012.12.31. 데이터
- 20~30일 데이터 / count(대여수) 정보 없음

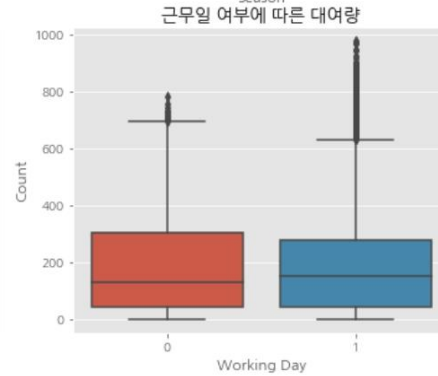
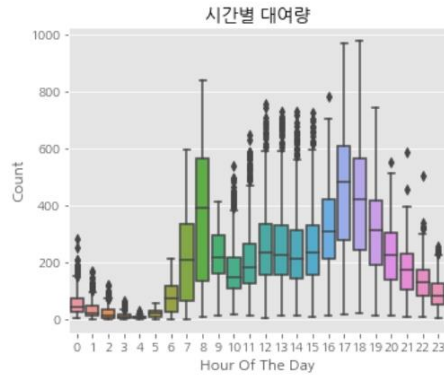
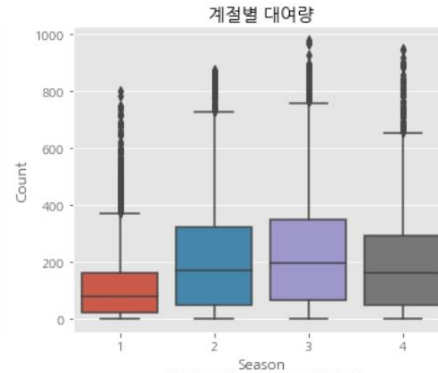
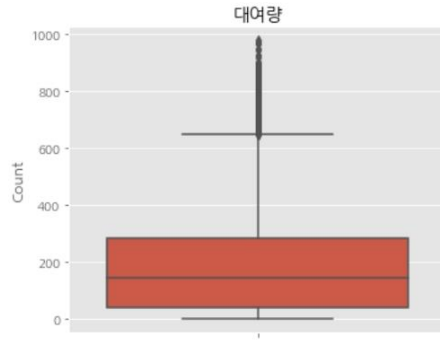


```
# missingno는 NaN 데이터들에 대해 시각화를 해줌  
# Nan = Not-a-Number로 숫자가 아니라는 뜻이다.  
# 즉 값이 없다는 뜻이다.
```

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

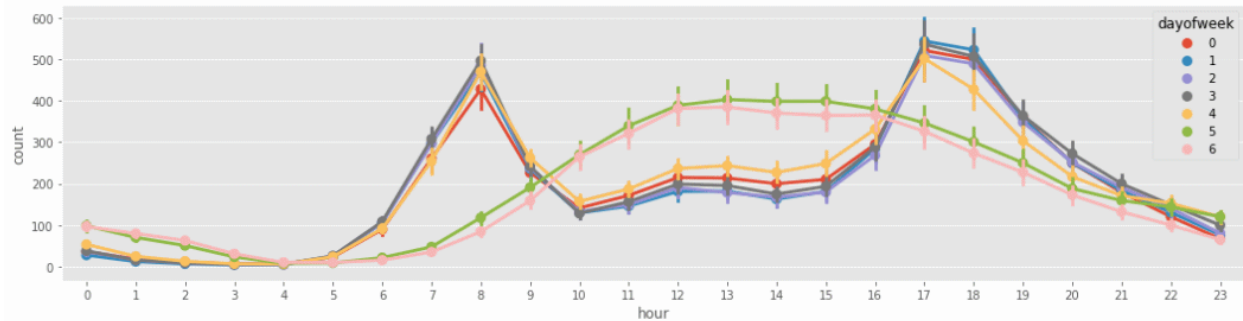
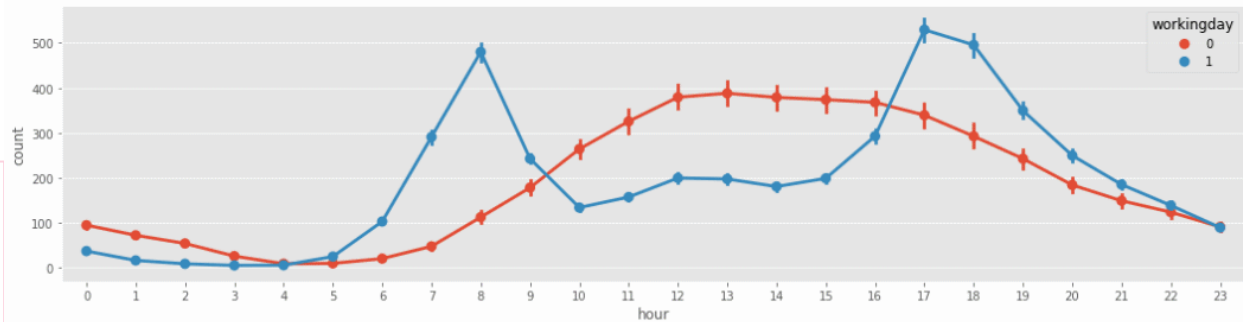
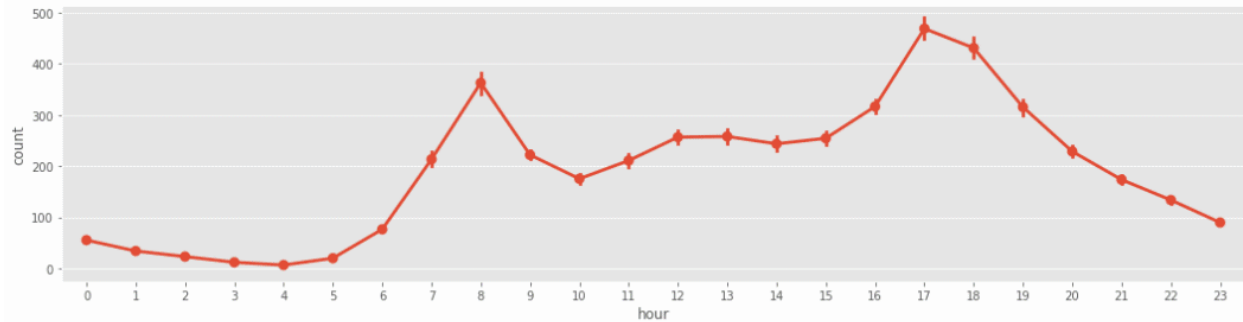


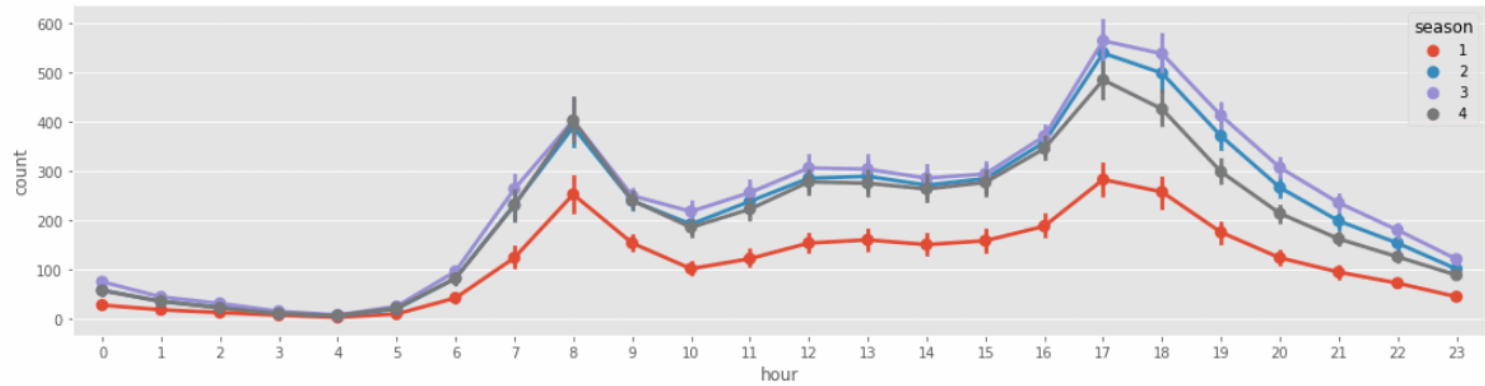
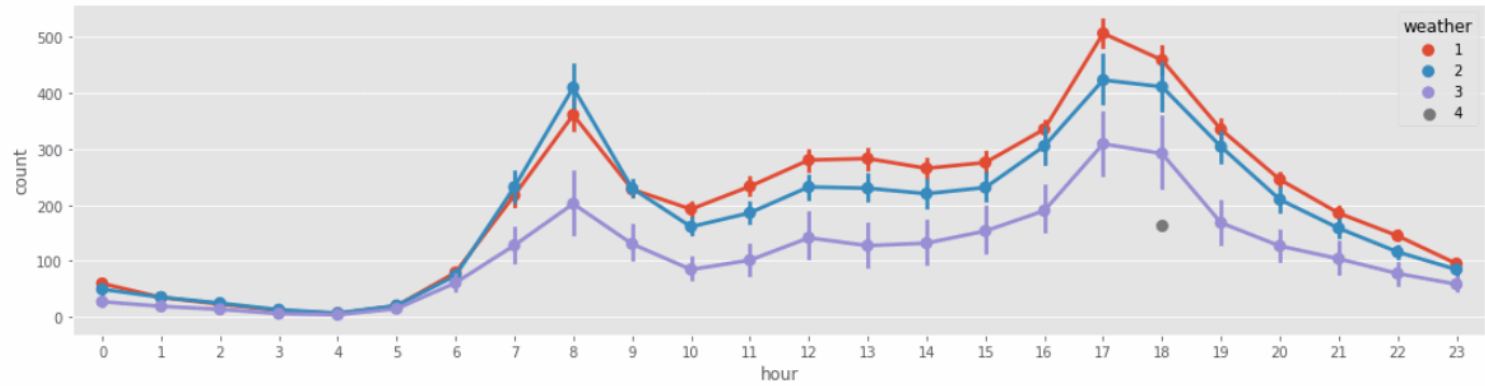
- 연도별 대여량은 2011년 보다 2012년에 더 많다.
- 월별 대여량은 6월에 가장 많고 7~10월도 대여량이 많다. 그리고 1월에 가장 적다.
- 일별대여량 정보는 train 데이터에는 1일부터 19일까지만 있고, 예측하고자 하는 test 데이터에는 20~30(31)까지 정보가 존재한다. 그래서 이 데이터는 feature로 사용하면 안 된다.
- 시간 대 대여량을 보면 출퇴근 시간에 대여량이 많은 것 같다. 하지만 주말과 나누어 볼 필요가 있을 것 같다.
- 분, 초는 다 0이기 때문에 의미가 없다. feature로 사용하지 않는다.



- 계절별 대여량은 가을에 가장 많음. 여름 겨울 봄 순서임.
- 근무일이 아닌 휴일(0)에 대여량이 더 많음

- 휴일(0)에는 점심(12~15)에 대여량이 많다
토요일, 일요일에도 동일하게 나타난다.
- 날씨가 좋을때(0) 대여량이 많다



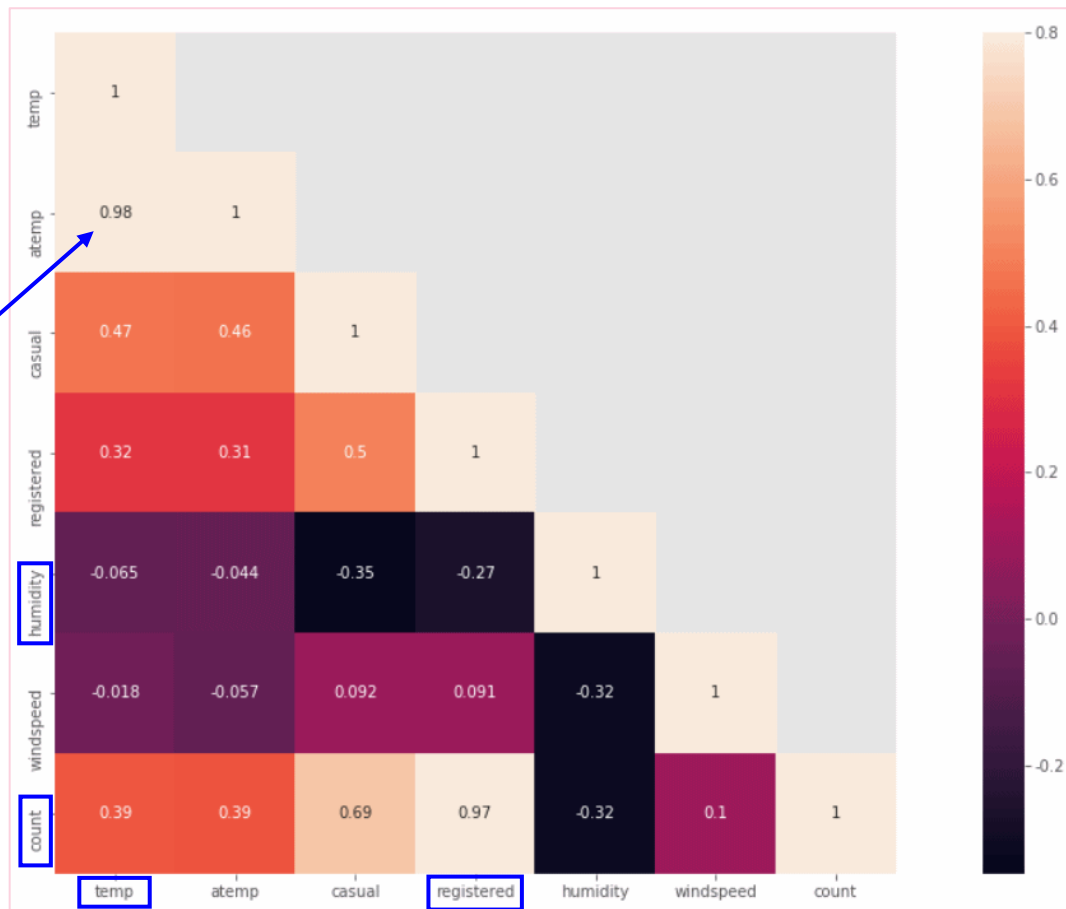


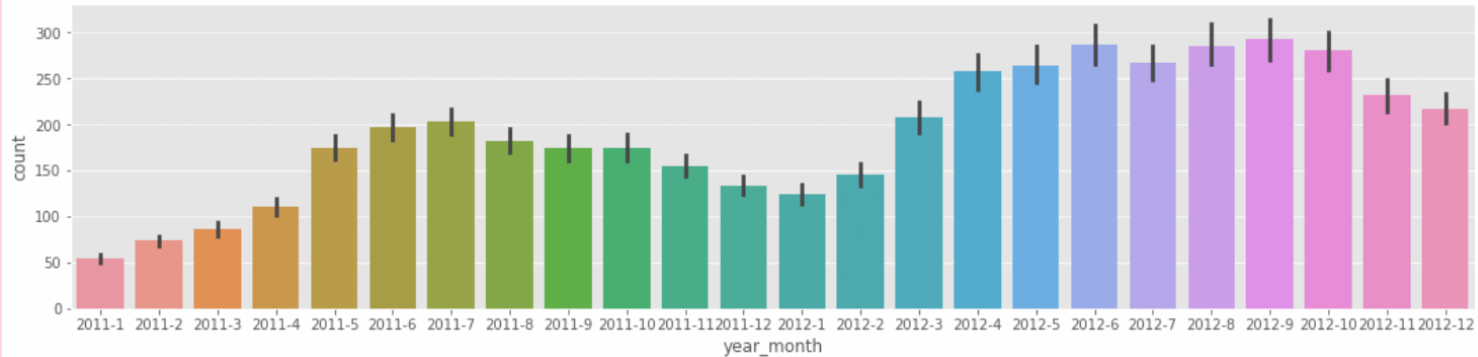
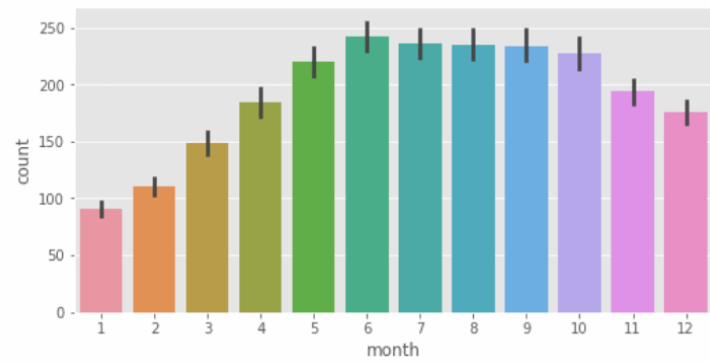
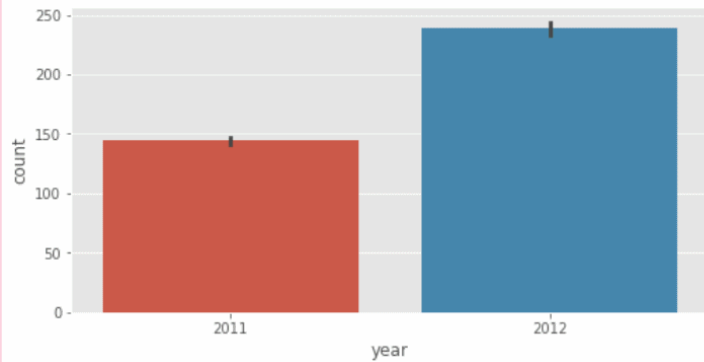
- 날씨가 좋을때(1) 대여량이 많다.
- 가을(3)에 대여량이 많다

Multicollinearity (다중공선성)

- 회귀 분석에서 사용된 모형의 일부 예측 변수가 다른 예측 변수와 상관 정도가 높아, 데이터 분석 시 부정적인 영향을 미치는 현상을 말함.
- 다중공선성이란 입력변수들 간의 상관관계가 존재하여 회귀 계수의 분산을 크게 하기 때문에, 회귀 분석 시 추정 회귀 계수를 믿을 수 없게 되는 문제가 발생하는 것을 말함.
- 다중 회귀 모형에서 회귀 계수란 독립 변수의 변화에 따른 종속 변수의 변화량을 나타내기 때문에, 설명 변수들 사이에 유의한 상관관계가 존재하는 경우 한 설명변수를 다른 설명변수와의 함수 관계로 표시할 수 있다.
- 이러한 경우 회귀 계수의 분산이 증가하며, 회귀 계수 추정치가 불안하고 해석하기 어려워진다.

- 온도와 습도는 거의 연관관계가 없다.
- 대여량과 가장 연관이 높은 건 registered로 등록된 대여자이다.
- 그러나 registered, casual 정보는 train 데이터에만 있고 test 데이터에는 이 값이 없으므로 feature로 사용하지 않는다.
- atemp(체감온도)와 temp(온도)는 0.98로 상관관계가 높지만, 너무 상관관계가 높기 때문에 온도와 체감온도는 같은 데이터로 보여지고 feature로 사용하기에 적합하지 않을 수 있다.
- 모델 구축 과정에서 atemp와 temp 중 하나는 데이터에 다중공선성을 나타낼 것이기 때문에 feature로 사용하지 않는다.





- 2011년보다 2012년의 대여량이 더 많다.
- 겨울보다는 여름에 대여량이 많다.
- 2011년과 2012년의 연_월별 데이터를 이어보면 전체적으로 증가하는 추세이다.

랜덤포레스트로 풍속 예측

- 랜덤포레스트로 풍속을 예측해서 값을 넣어줌
- train 데이터 : 풍속 값이 있는 데이터
- test 데이터 : 풍속 값이 0인 데이터

```
# 풍속 값을 예측하는 메서드 정의
def predict_windspeed(data):
    # 풍속이 0인것과 아닌 것을 나누어 준다.
    dataWind0 = data.loc[data['windspeed'] == 0]
    dataWindNot0 = data.loc[data['windspeed'] != 0]

    # 풍속을 예측할 feature를 선택한다.
    wCol = ["season", "weather", "humidity", "month", "temp", "year", "atemp"]

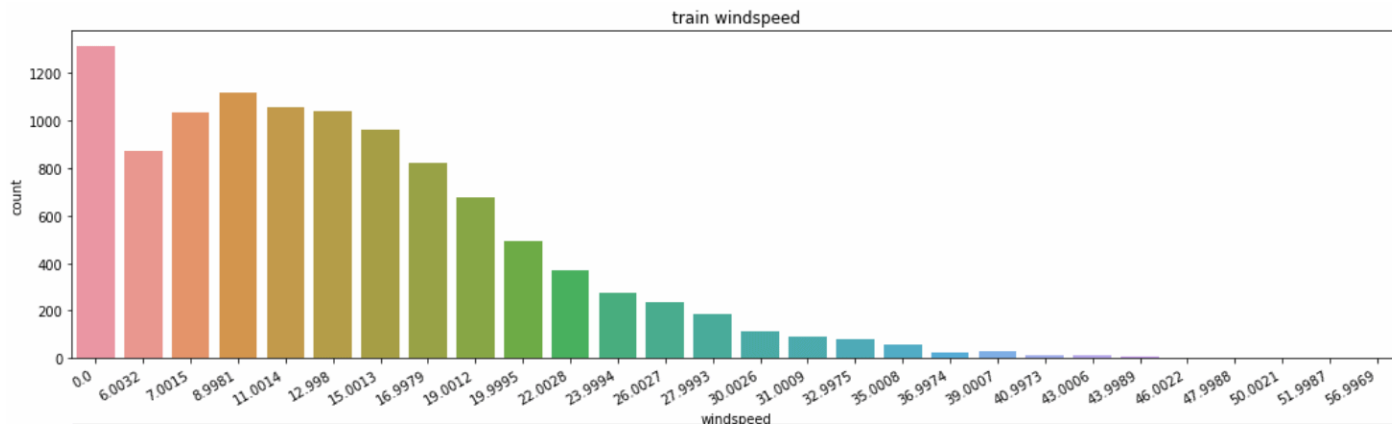
    # 풍속이 0이 아닌 데이터들의 타입을 스트링으로 바꿔준다.
    dataWindNot0["windspeed"] = dataWindNot0["windspeed"].astype("str")

    # 랜덤포레스트 분류기를 사용한다.
    rfModel_wind = RandomForestClassifier()

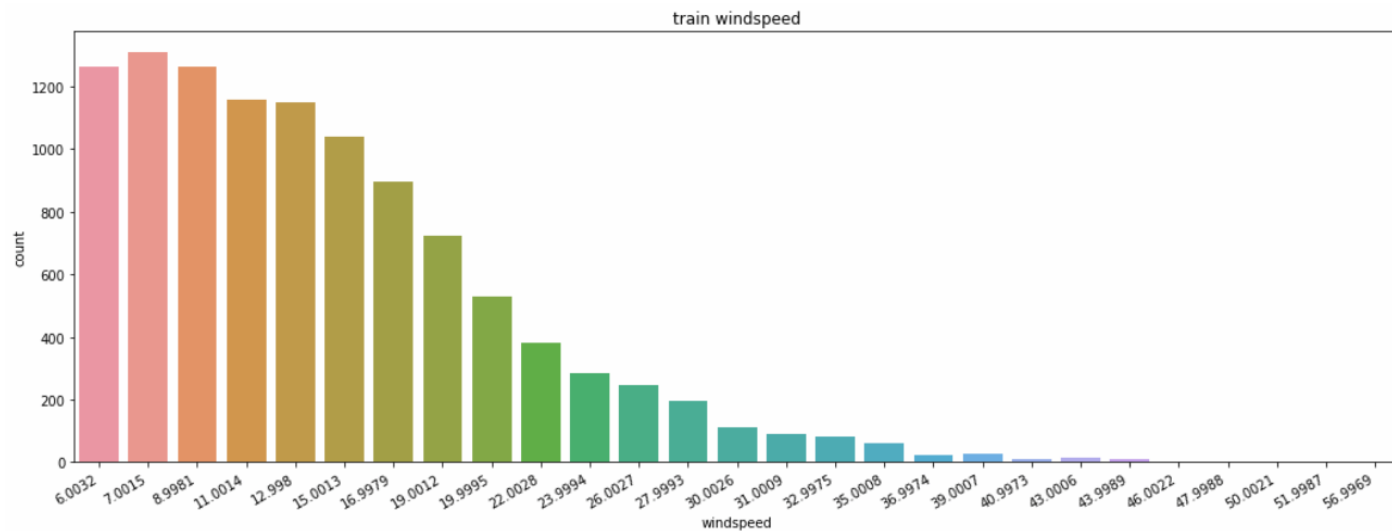
    # wCol에 있는 피쳐의 값을 바탕으로 풍속을 학습시킨다.
    rfModel_wind.fit(dataWindNot0[wCol], dataWindNot0["windspeed"])

    # 학습한 값을 바탕으로 풍속이 0으로 기록 된 데이터의 풍속을 예측한다.
    wind0Values = rfModel_wind.predict(X=dataWind0[wCol])
```

■ Before



■ After



Feature

	season	weather	temp	atemp	humidity	windspeed	year	hour	dayofweek	holiday	workingday
0	1	2	9.84	12.880	75	6.0032	2011	5	5	0	0
1	1	1	15.58	19.695	76	16.9979	2011	10	5	0	0
2	1	1	14.76	16.665	81	19.0012	2011	11	5	0	0
3	1	1	17.22	21.210	77	19.0012	2011	12	5	0	0
4	1	2	18.86	22.725	72	19.9995	2011	13	5	0	0


```
forest = forest.fit(train_data_features, train[ 'label' ])
```

- [각 리뷰마다 피쳐 벡터화 한 train_data_features]와 [sentiment]을 넣어서 [forest] 모델에 학습을 시킴 .

```
result = forest.predict(test_data)
```

- [각 리뷰마다 피쳐 벡터화 한 test_data_features] 넣고, 학습이 된 forest 모델로 sentiment를 예측을 함.

train_data

	season	weather	temp	atemp	humidity	windspeed
0	1	2	9.84	12.880	75	6.0032
1	1	1	15.58	19.695	76	16.9979
2	1	1	14.76	16.665	81	19.0012
3	1	1	17.22	21.210	77	19.0012
4	1	2	18.86	22.725	72	19.9995

학습 (fit)

forest

예측
(predict)

test_data

	season	weather	temp	atemp	humidity	windspeed
0	1	2	9.84	12.880	75	6.0032
1	1	1	15.58	19.695	76	16.9979
2	1	1	14.76	16.665	81	19.0012
3	1	1	17.22	21.210	77	19.0012
4	1	2	18.86	22.725	72	19.9995

결과

test[label]

	sentiment
0	1
1	1
2	0
3	0
4	1

train[label]

	sentiment
0	1
1	1
2	0
3	0
4	1

- LinerRegression

```
lModel = LinearRegression().  
y_train_log = np.log1p(y_train)  
lModel.fit(X_train, y_train_log) # 모델을 학습시킨다  
preds = lModel.predict(X_train) # 예측.
```

- Ridge

```
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 ) # 모델을 학습시킴.  
preds = grid_ridge_m.predict(X_train) # 예측.
```

- Lasso

```
lasso_m_ = Lasso()  
alpha = 1/np.array([0.1, 1, 2, 3, 4, 10, 30,100,200,300,400,800,900,1000])  
lasso_params_ = { 'max_iter':[3000], 'alpha':alpha}  
grid_lasso_m = GridSearchCV( lasso_m_, lasso_params_, scoring = rmsle_scorer, cv=5)  
y_train_log = np.log1p(y_train)  
grid_lasso_m.fit( X_train , y_train_log ) # 모델을 학습시킴.  
preds = grid_lasso_m.predict(X_train) # 예측.
```

- GradientBoostRegressor

```
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) # 모델을 학습시킴.  
preds = gbm.predict(X_train) # 예측
```

랜덤포레스트

```
from sklearn.ensemble import RandomForestRegressor
max_depth_list = []
model = RandomForestRegressor(n_estimators=100,
                              n_jobs=-1,
                              random_state=0)

# 모델 학습
model.fit(X_train, y_train)

# 예측
predictions = model.predict(X_test)
```

파일 홈 삽입 페이지 레이아웃 수식 데이터 검토					
A1	:	X ✓ fx	datetime		
	A	B	C	D	
1	datetime	count			
2	2011-01-20 0:00	12.51			
3	2011-01-20 1:00	5.05			
4	2011-01-20 2:00	4.35			
5	2011-01-20 3:00	3.54			
6	2011-01-20 4:00	3.24			
7	2011-01-20 5:00	6.27			
8	2011-01-20 6:00	38.61			
9	2011-01-20 7:00	104.71			
10	2011-01-20 8:00	233.47			
11	2011-01-20 9:00	135.22			
12	2011-01-20 10:00	63.1			
13	2011-01-20 11:00	63.54			
14	2011-01-20 12:00	90.12			
15	2011-01-20 13:00	75.42			
16	2011-01-20 14:00	86.93			
17	2011-01-20 15:00	86.95			
18	2011-01-20 16:00	102.63			
19	2011-01-20 17:00	205.56			
20	2011-01-20 18:00	179.83			
21	2011-01-20 19:00	99.76			
22	2011-01-20 20:00	71.29			
23	2011-01-20 21:00	50.21			
24	2011-01-20 22:00	46.29			



Thank You

감사합니다