Yulu - DSML - Hypothesis Testing - Case Study

by Pavan Kumaar

About Yulu

Yulu is India's leading micro-mobility service provider, which offers unique vehicles for the daily commute. Starting off as a mission to eliminate traffic congestion in India, Yulu provides the safest commute solution through a user-friendly mobile app to enable shared, solo and sustainable commuting.

Yulu zones are located at all the appropriate locations (including metro stations, bus stands, office spaces, residential areas, corporate offices, etc) to make those first and last miles smooth, affordable, and convenient!

Yulu has recently suffered considerable dips in its revenues. They have contracted a consulting company to understand the factors on which the demand for these shared electric cycles depends. Specifically, they want to understand the factors affecting the demand for these shared electric cycles in the Indian market.

How you can help here?

The company wants to know:

Which variables are significant in predicting the demand for shared electric cycles in the Indian market? How well those variables describe the electric cycle demands

Column Profiling:

- datetime: datetime
- season: season (1: spring, 2: summer, 3: fall, 4: winter)
- holiday: whether day is a holiday or not (extracted from [http://dchr.dc.gov/page/holiday-schedule (http://dchr.dc.gov/page/holiday-schedule)](http://dchr.dc.gov/page/holiday-schedule))
- workingday: if day is neither weekend nor holiday is 1, otherwise is 0.
- 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
- atemp: feeling temperature in Celsius
- humidity: humidity
- windspeed: wind speed
- casual: count of casual users
- registered: count of registered users
- count: count of total rental bikes including both casual and registered

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
import seaborn as sns
```

In [2]:
```
df = pd.read_csv("yulu_data.csv")
```

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   datetime    10886 non-null  object 
 1   season      10886 non-null  int64  
 2   holiday     10886 non-null  int64  
 3   workingday  10886 non-null  int64  
 4   weather     10886 non-null  int64  
 5   temp        10886 non-null  float64
 6   atemp       10886 non-null  float64
 7   humidity    10886 non-null  int64  
 8   windspeed   10886 non-null  float64
 9   casual      10886 non-null  int64  
 10  registered  10886 non-null  int64  
 11  count       10886 non-null  int64  
dtypes: float64(3), int64(8), object(1)
memory usage: 1020.7+ KB
```

Datatype of following attributes needs to changed to proper data type

- **datetime** - to datetime
- **season** - to categorical
- **holiday** - to categorical
- **workingday** - to categorical
- **weather** - to categorical

In [4]:
```
df['datetime'] = pd.to_datetime(df['datetime'])
cat_cols= ['season', 'holiday', 'workingday', 'weather']
for col in cat_cols:
    df[col] = df[col].astype('object')
```

In [5]:
```
# minimum datetime and maximum datetime
df['datetime'].min(), df['datetime'].max()
```

Out[5]:
```
(Timestamp('2011-01-01 00:00:00'), Timestamp('2012-12-19 23:00:00'))
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   datetime    10886 non-null  datetime64[ns]
 1   season      10886 non-null  object        
 2   holiday     10886 non-null  object        
 3   workingday  10886 non-null  object        
 4   weather     10886 non-null  object        
 5   temp        10886 non-null  float64       
 6   atemp       10886 non-null  float64       
 7   humidity    10886 non-null  int64         
 8   windspeed   10886 non-null  float64       
 9   casual      10886 non-null  int64         
 10  registered  10886 non-null  int64         
 11  count       10886 non-null  int64         
dtypes: datetime64[ns](1), float64(3), int64(4), object(4)
memory usage: 1020.7+ KB
```

In [7]:
```
df.head(10)
```

Out[7]:

	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	0.0000	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000	0	1	1
5	2011-01-01 05:00:00	1	0	0	2	9.84	12.880	75	6.0032	0	1	1
6	2011-01-01 06:00:00	1	0	0	1	9.02	13.635	80	0.0000	2	0	2
7	2011-01-01 07:00:00	1	0	0	1	8.20	12.880	86	0.0000	1	2	3
8	2011-01-01 08:00:00	1	0	0	1	9.84	14.395	75	0.0000	1	7	8
9	2011-01-01 09:00:00	1	0	0	1	13.12	17.425	76	0.0000	8	6	14

In [8]:
```
df.shape
```

Out[8]: (10886, 12)

In [9]:
```
df.describe()
```

Out[9]:

	temp	atemp	humidity	windspeed	casual	registered	count
count	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
mean	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191.574132
std	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.144454
min	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.000000
50%	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.000000
75%	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.000000
max	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.000000

In [10]:
```
df.isna().sum()
```

Out[10]:
```
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
```

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
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
```

In [12]:
```
df.duplicated().sum()
```

Out[12]:
```
0
```

In [13]:
```
df.nunique()
```

Out[13]:
```
datetime      10886
season            4
holiday           2
workingday        2
weather           4
temp             49
atemp            60
humidity         89
windspeed        28
casual          309
registered      731
count           822
dtype: int64
```

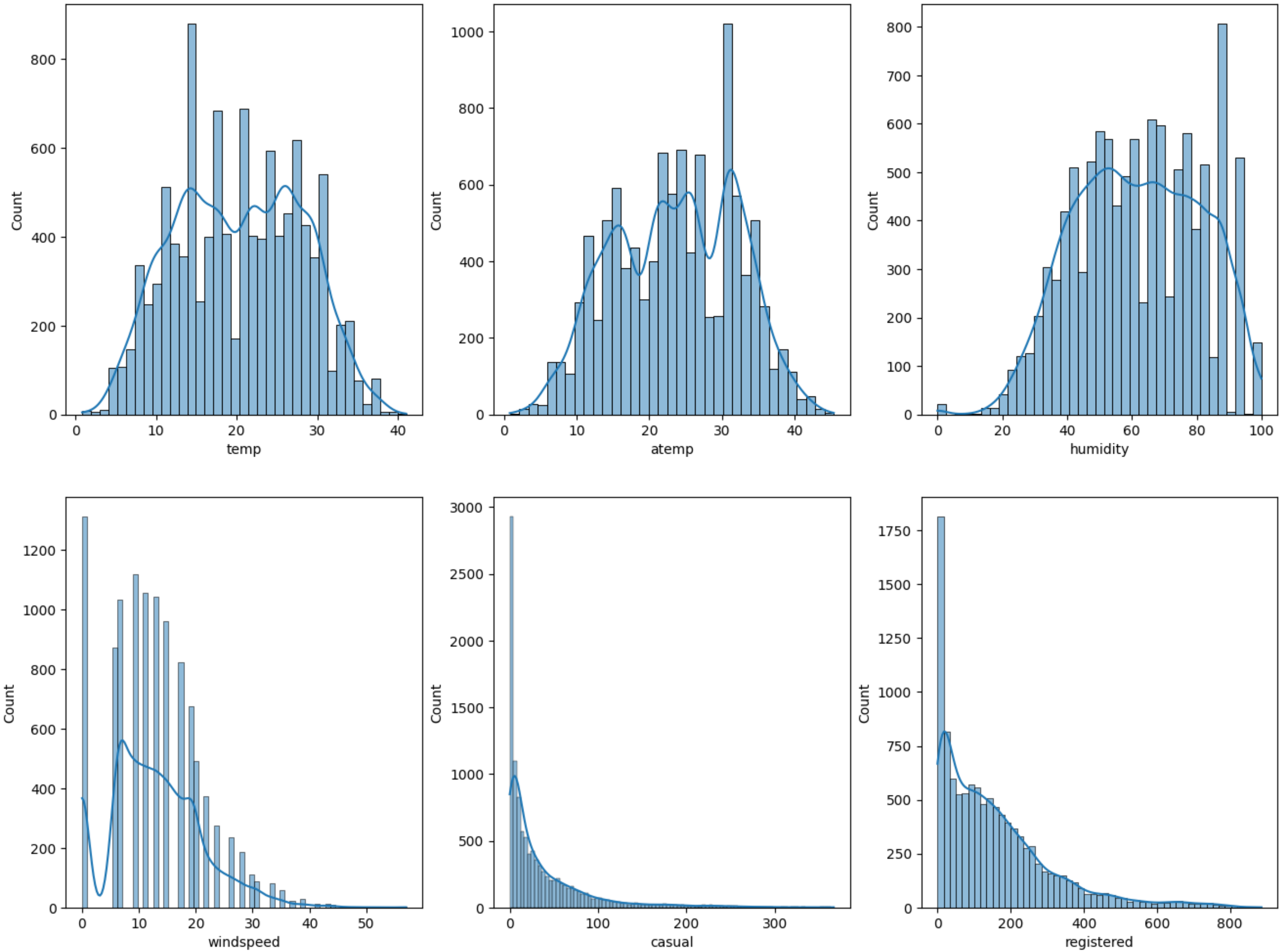
In [14]:
```
# number of unique values in each categorical columns
df[cat_cols].melt().groupby(['variable', 'value'])[['value']].count()
```

Out[14]:

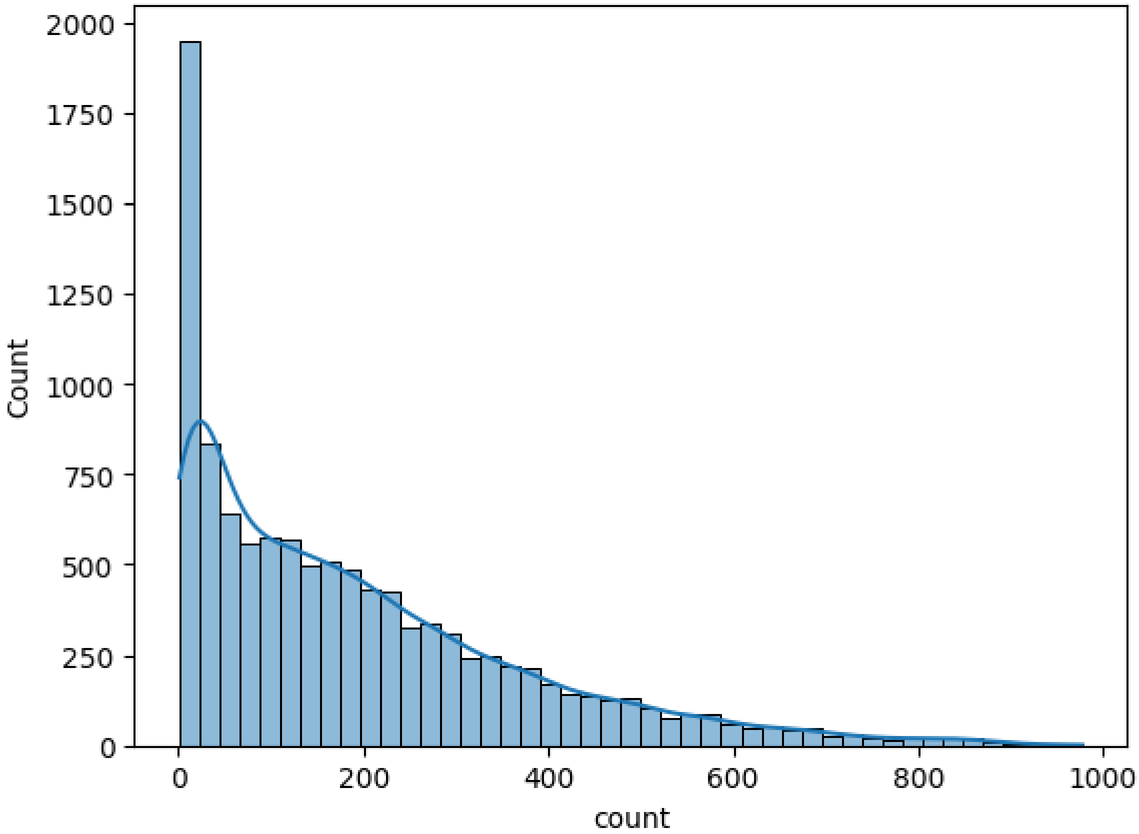
		value
variable	**value**	
holiday	**0**	10575
	1	311
season	**1**	2686
	2	2733
	3	2733
	4	2734
weather	**1**	7192
	2	2834
	3	859
	4	1
workingday	**0**	3474
	1	7412

Univariate Analysis

In [15]:
```
# understanding the distribution for numerical variables
num_cols = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered','count']

fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(16, 12))

index = 0
for row in range(2):
    for col in range(3):
        sns.histplot(df[num_cols[index]], ax=axis[row, col], kde=True)
        index += 1

plt.show()
sns.histplot(df[num_cols[-1]], kde=True)
plt.show()
```



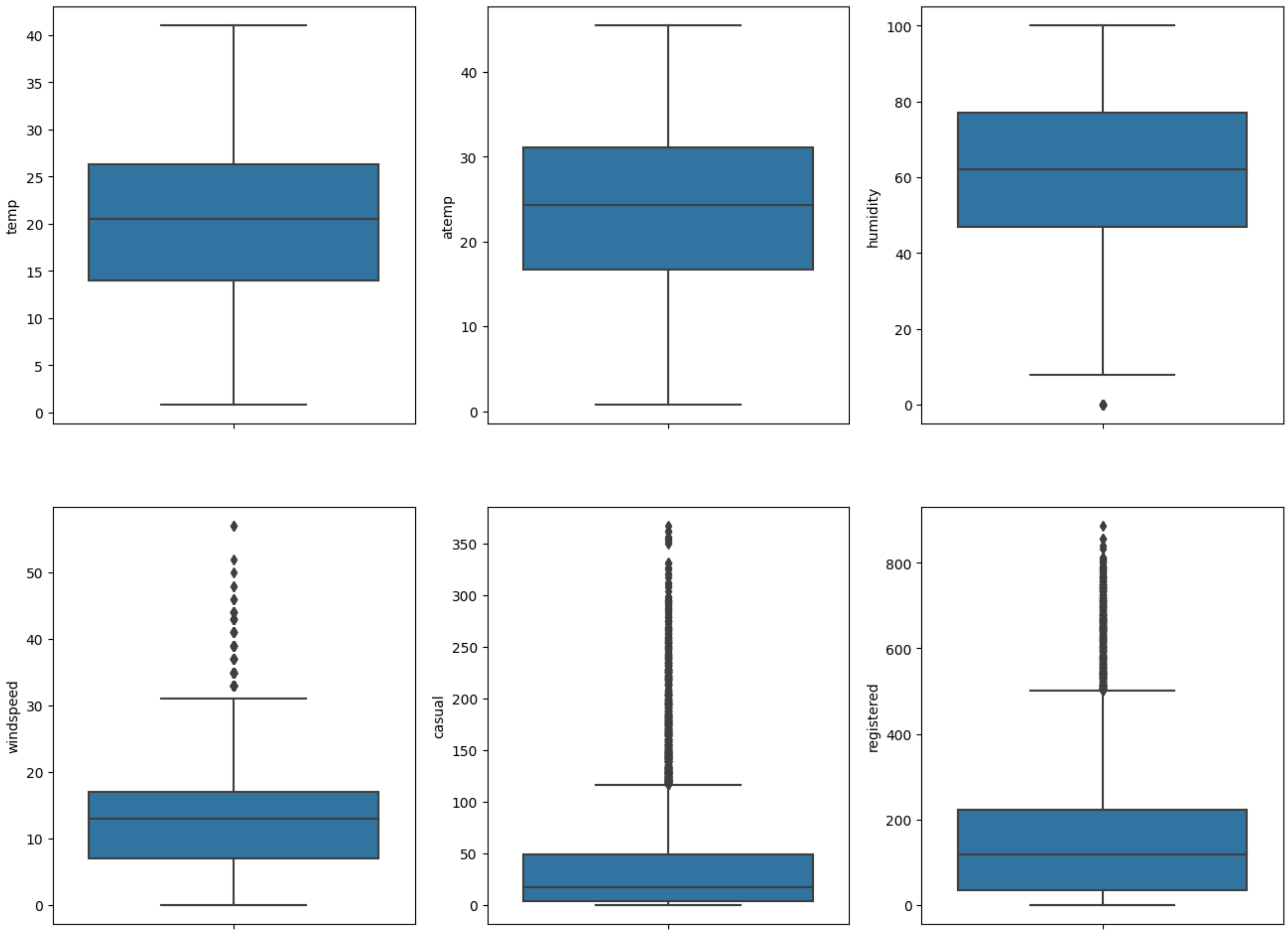
Count
temp
Count
atemp
Count
humidity
Count
windspeed
Count
casual
Count
registered

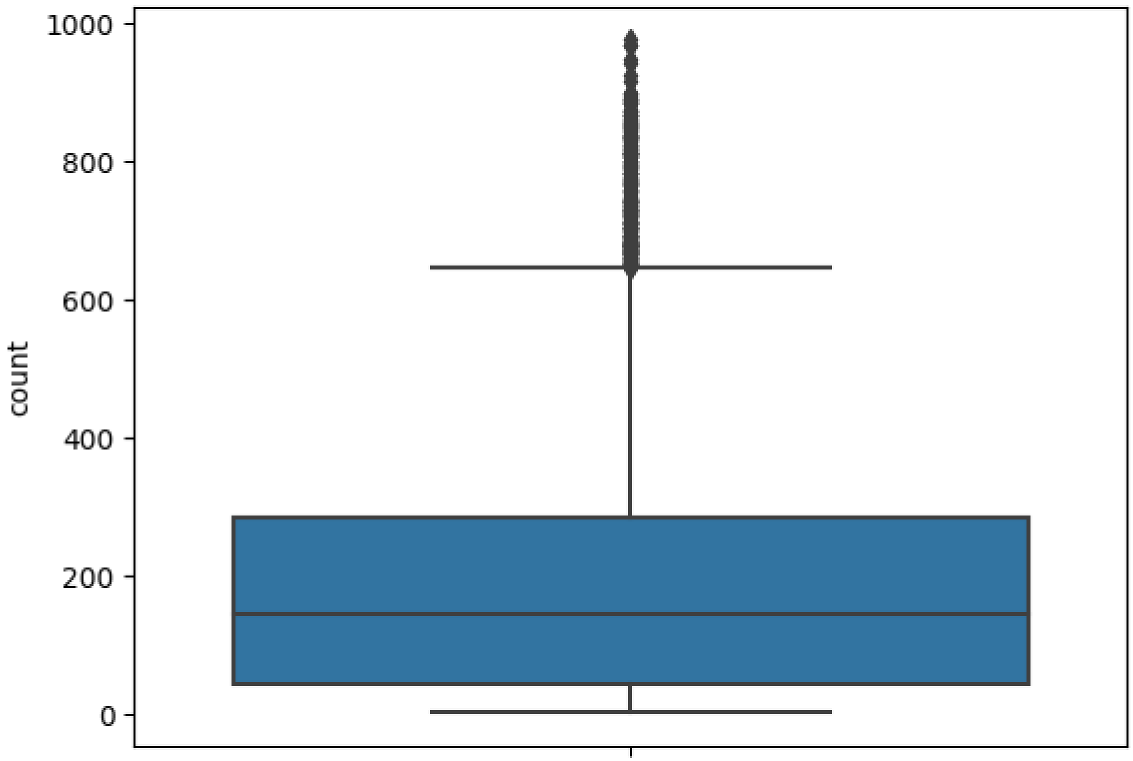


- casual, registered and count somewhat looks like Log Normal Distrinution
- temp, atemp and humidity looks like they follows the Normal Distribution
- windspeed follows the binomial distribution

In [16]:

```
# plotting box plots to detect outliers in the data
fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(16, 12))

index = 0
for row in range(2):
    for col in range(3):
        sns.boxplot(y=df[num_cols[index]], ax=axis[row, col])
        index += 1

plt.show()
sns.boxplot(y=df[num_cols[-1]])
plt.show()
```

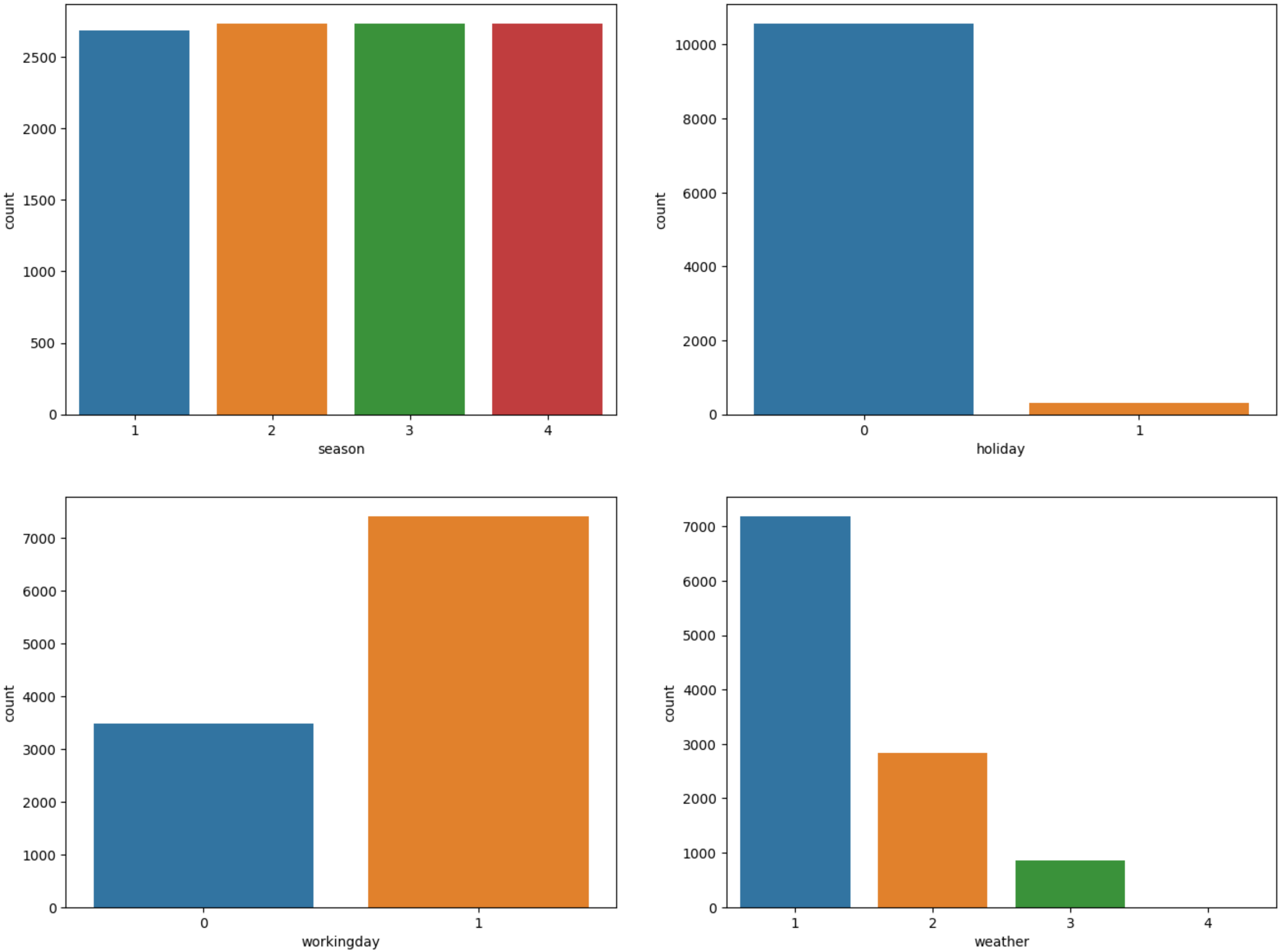

temp
0
5
10
15
20
25
30
35
40
atemp
0
10
20
30
40
humidity
0
20
40
60
80
100
windspeed
0
10
20
30
40
50
casual
0
50
100
150
200
250
300
350
registered
0
200
400
600
800



Looks like humidity, casual, registered and count have outliers in the data.

In [17]:

```
# countplot of each categorical column
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(16, 12))

index = 0
for row in range(2):
    for col in range(2):
        sns.countplot(data=df, x=cat_cols[index], ax=axis[row, col])
        index += 1

plt.show()
```



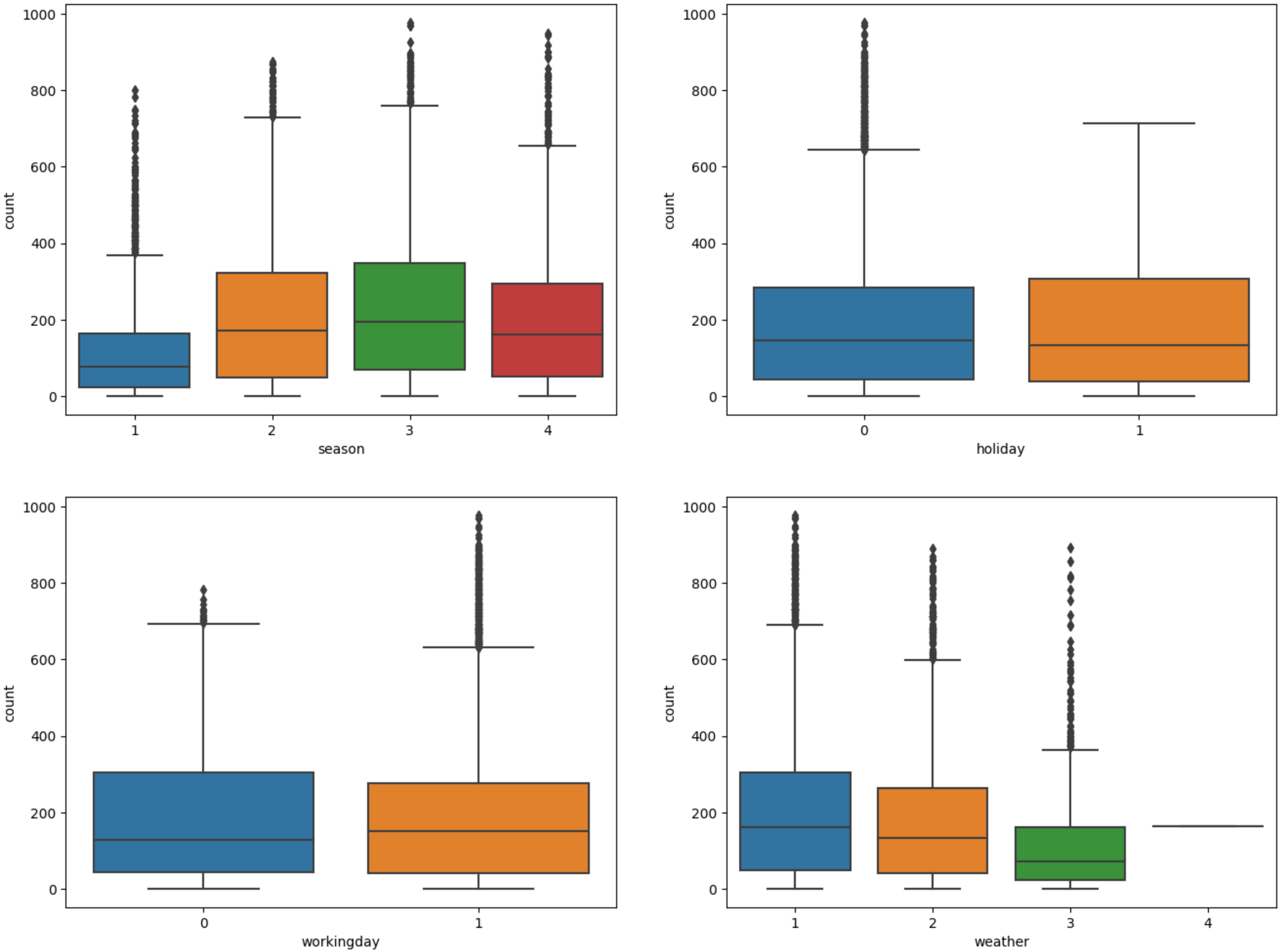
count
season
holiday
workingday
weather

Data looks common as it should be like equal number of days in each season, more working days and weather is mostly Clear, Few clouds, partly cloudy, partly cloudy.

Bi-variate Analysis

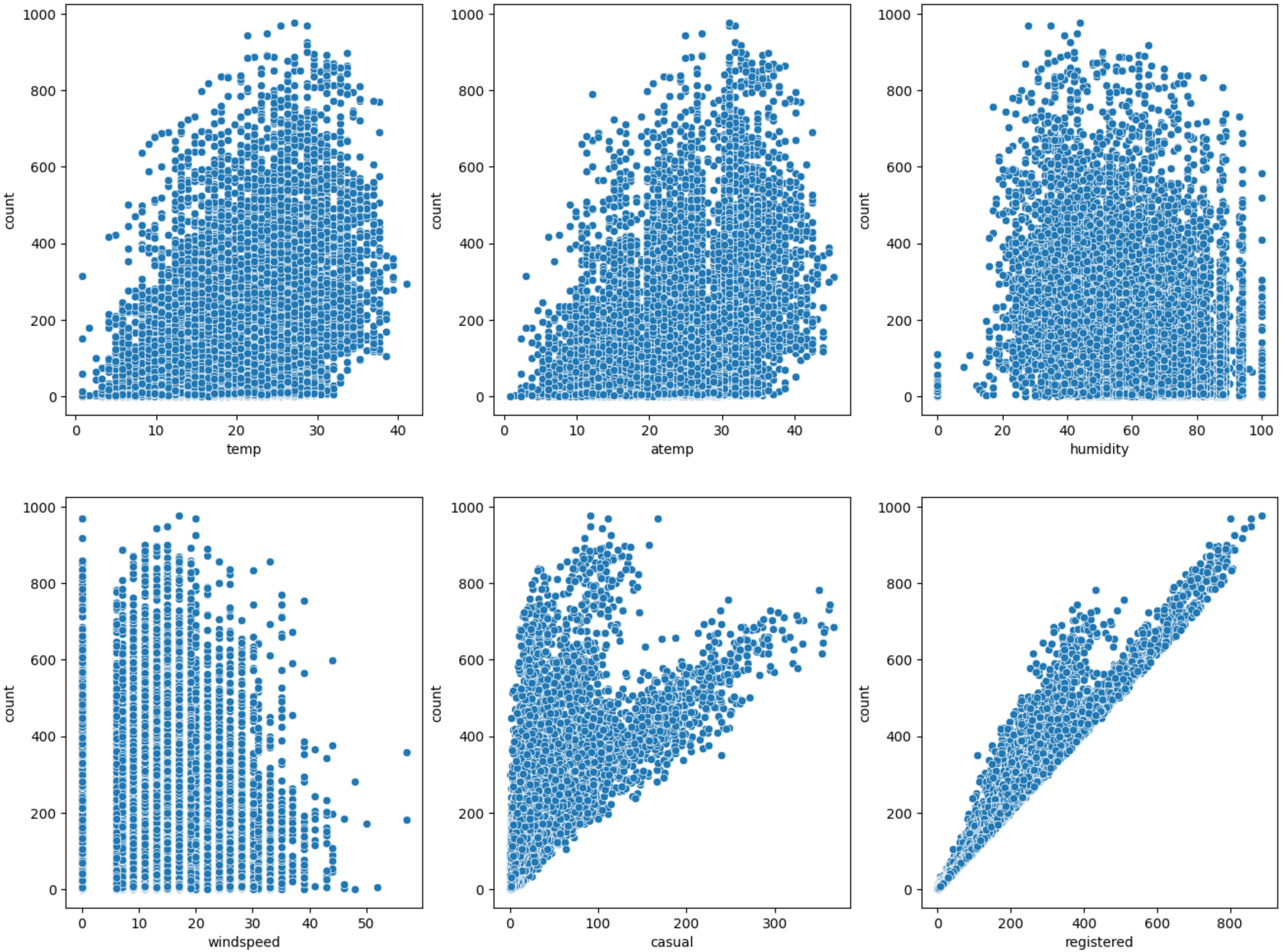
In [18]:

```
# plotting categorical variables againt count using boxplots
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(16, 12))

index = 0
for row in range(2):
    for col in range(2):
        sns.boxplot(data=df, x=cat_cols[index], y='count', ax=axis[row, col])
        index += 1

plt.show()
```


1000
800
600
400
200
0
count
1
2
3
4
season
holiday
0
1
workingday
weather

- In summer and fall seasons more bikes are rented as compared to other seasons.
- Whenever its a holiday more bikes are rented.
- It is also clear from the workingday also that whenever day is holiday or weekend, slightly more bikes were rented.
- Whenever there is rain, thunderstorm, snow or fog, there were less bikes were rented.

In [19]:
```
# plotting numerical variables againt count using scatterplot
fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(16, 12))

index = 0
for row in range(2):
    for col in range(3):
        sns.scatterplot(data=df, x=num_cols[index], y='count', ax=axis[row, col])
        index += 1

plt.show()
```


count
temp
atemp
humidity
windspeed
casual
registered

- Whenever the humidity is less than 20, number of bikes rented is very very low.
- Whenever the temperature is less than 10, number of bikes rented is less.
- Whenever the windspeed is greater than 35, number of bikes rented is less.

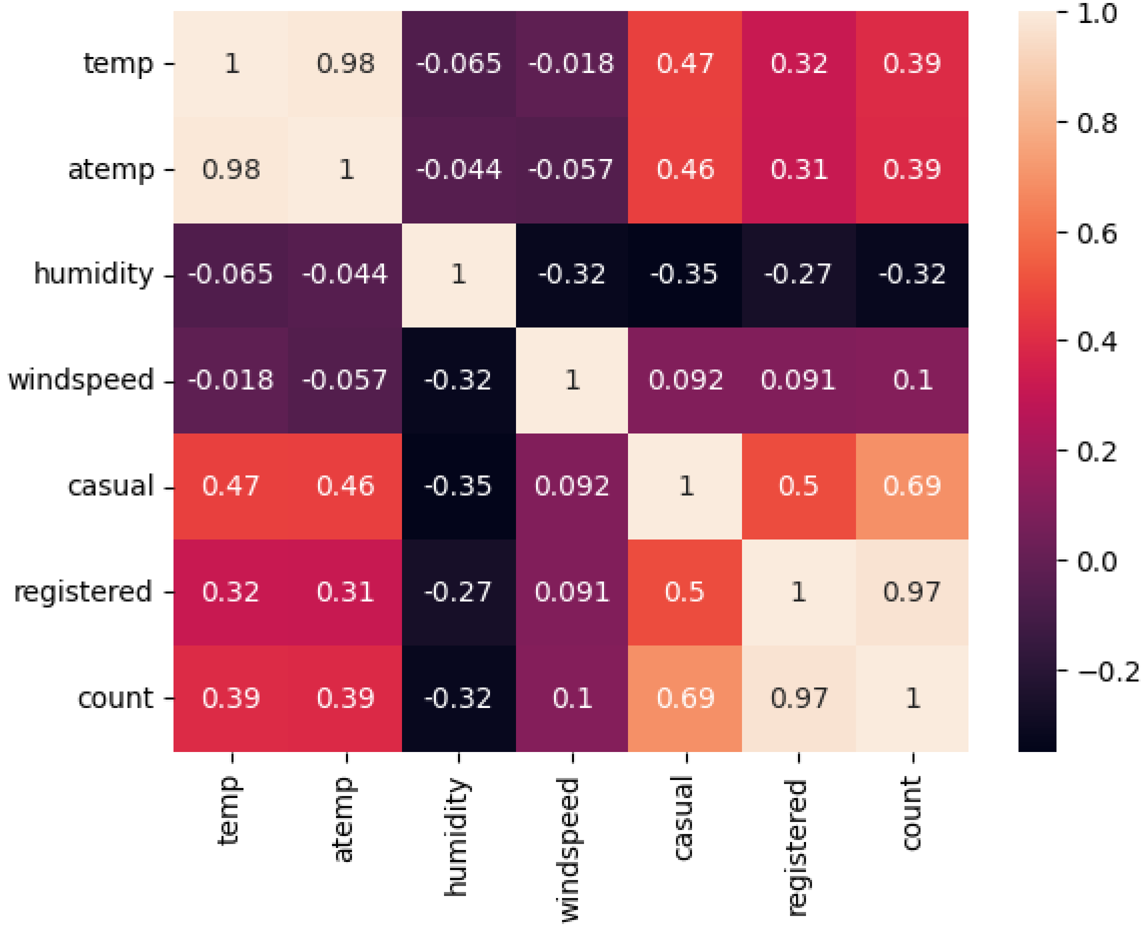
In [20]:

```
# understanding the correlation between count and numerical variables
df.corr()['count']
```

Out[20]:

```
temp          0.394454
atemp         0.389784
humidity     -0.317371
windspeed     0.101369
casual        0.690414
registered    0.970948
count         1.000000
Name: count, dtype: float64
```

In [21]:
```
sns.heatmap(df.corr(), annot=True)
plt.show()
```



Hypothesis Testing

Test-1

Null Hypothesis (H0): Holiday is independent of the season

Alternate Hypothesis (H1): Holiday is dependant of the season

Significance level (alpha): 0.05

We will use **chi-square test** to test hypyothesis defined above.

In [22]:
```
data_table = pd.crosstab(df['holiday'],df['season'])
print("Observed values:")
data_table
```

```
Observed values:
```

Out[22]:

season	1	2	3	4
holiday				
0	2615	2685	2637	2638
1	71	48	96	96

In [23]:
```
val = stats.chi2_contingency(data_table)
expected_values = val[3]
expected_values
```

Out[23]:
```
array([[2609.26419254, 2654.92145875, 2654.92145875, 2655.89288995],
       [  76.73580746,   78.07854125,   78.07854125,   78.10711005]])
```

In [24]:
```
nrows, ncols = 2, 4
dof = (nrows-1)*(ncols-1)
print("degrees of freedom: ", dof)
alpha = 0.05


chi_sqr = sum([(o-e)**2/e for o, e in zip(data_table.values, expected_values)])
chi_sqr_statistic = chi_sqr[0] + chi_sqr[1]
print("chi-square test statistic: ", chi_sqr_statistic)

critical_val = stats.chi2.ppf(q=1-alpha, df=dof)
print(f"critical value: {critical_val}")

p_val = 1-stats.chi2.cdf(x=chi_sqr_statistic, df=dof)
print(f"p-value: {p_val}")

if p_val <= alpha:
    print("\nSince p-value is less than the alpha 0.05, We reject the Null Hypothesis. Meaning that\
    Holiday is dependent on the season.")
else:
    print("Since p-value is greater than the alpha 0.05, We do not reject the Null Hypothesis. Meaning that\
    Holiday is independent on the season.")
```

```
degrees of freedom:  3
chi-square test statistic:  12.369405489593898
critical value: 7.814727903251179
p-value: 0.006219147034966399

Since p-value is less than the alpha 0.05, We reject the Null Hypothesis. Meaning that    Holiday is dependent on the season.
```

Test-2

Null Hypothesis: Holiday has effect on the number of cycles being rented.

Alternate Hypothesis: Holiday has no effect on the number of cycles being rented.

Significance level (alpha): 0.05

In [25]:
```
data_group1 = df[df['holiday']==0]['count'].values
data_group2 = df[df['holiday']==1]['count'].values

np.var(data_group1), np.var(data_group2)
```

Out[25]:
```
(32943.901106481346, 28233.99150132856)
```

Before conducting the two-sample T-Test we need to find if the given data groups have the same variance. If the ratio of the larger data groups to the small data group is less than 4:1 then we can consider that the given data groups have equal variance.

Here, the ratio is 32943.90 / 28233.99 which is less than 4:1

In [26]:
```
stats.ttest_ind(a=data_group1, b=data_group2, equal_var=True)
```

Out[26]:
```
Ttest_indResult(statistic=0.5626388963477119, pvalue=0.5736923883271103)
```

Since pvalue is greater than 0.05 so we **cannot reject** the Null hypothesis. We don't have the sufficient evidence to say that holiday has no effect on the number of cycles being rented.

Test-3

Null Hypothesis: Number of cycles rented is not similar on working days and holidays.

Alternate Hypothesis: Number of cycles rented is similar on working days and holidays.

Significance level (alpha): 0.05

Here, we will use the **ANOVA** to test the hypothess defined above

In [27]:
```
# defining the data groups for the ANOVA

gp1 = df[df['workingday']==0]['count'].values
gp2 = df[df['workingday']==1]['count'].values
gp3 = df[df['holiday']==0]['count'].values
gp4 = df[df['holiday']==1]['count'].values

# conduct the one-way anova
stats.f_oneway(gp1, gp2, gp3, gp4)
```

Out[27]:
```
F_onewayResult(statistic=0.5932337989075094, pvalue=0.6193690368227767)
```

Since p-value is greater than 0.05, we **cannot reject** the null hypothesis. This implies that Number of cycles rented is not similar in working days and holiday.

Insights

- In **summer** and **fall** seasons more bikes are rented as compared to other seasons.
- Whenever its a **holiday** more bikes are rented.
- It is also clear from the workingday also that whenever day is holiday or weekend, slightly more bikes were rented.
- Whenever there is **rain, thunderstorm, snow or fog**, there were less bikes were rented.

- Whenever the humidity is less than 20, number of bikes rented is very very low.
- Whenever the temperature is less than 10, number of bikes rented is less.
- Whenever the windspeed is greater than 35, number of bikes rented is less.

Recommendations

- In **summer** and **fall** seasons the company should have more bikes in stock to be rented. Because the demand in these seasons is higher as compared to other seasons.
- With a significance level of 0.05, holiday has effect on the number of bikes being rented.
- In very low humid days, company should have less bikes in the stock to be rented.
- Whenever temprature is less than 10 or in very cold days, company should have less bikes.
- Whenever the windspeed is greater than 35 or in thunderstorms, company should have less bikes in stock to be rented.

In [ ]:

```

```

```

```