Business Case: Yulu - Hypothesis Testing

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))
- 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

Import the dataset and do usual exploratory data analysis steps like checking the structure & characteristics of the dataset

In [39]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import stats
from scipy.stats import chi2_contingency
from scipy.stats import chi2
from scipy.stats import ttest_ind
```

In [2]:
```
yulu_df = pd.read_csv('C://Users//dell//OneDrive//Desktop//Personal Doc//Yulu Dataset.csv')
yulu_df
```

Out[2]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	01-01-2011 00:00	1	0	0	1	9.84	14.395	81	0.0000	3	13	16
1	01-01-2011 01:00	1	0	0	1	9.02	13.635	80	0.0000	8	32	40
2	01-01-2011 02:00	1	0	0	1	9.02	13.635	80	0.0000	5	27	32
3	01-01-2011 03:00	1	0	0	1	9.84	14.395	75	0.0000	3	10	13
4	01-01-2011 04:00	1	0	0	1	9.84	14.395	75	0.0000	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...
10881	19-12-2012 19:00	4	0	1	1	15.58	19.695	50	26.0027	7	329	336
10882	19-12-2012 20:00	4	0	1	1	14.76	17.425	57	15.0013	10	231	241
10883	19-12-2012 21:00	4	0	1	1	13.94	15.910	61	15.0013	4	164	168
10884	19-12-2012 22:00	4	0	1	1	13.94	17.425	61	6.0032	12	117	129
10885	19-12-2012 23:00	4	0	1	1	13.12	16.665	66	8.9981	4	84	88

10886 rows × 12 columns

In [3]:
```
yulu_df.describe()
```

Out[3]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	register
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.0000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.799395	36.021955	155.5521
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.960477	151.0390
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.000000	0.000000	0.0000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.001500	4.000000	36.0000
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.998000	17.000000	118.0000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.997900	49.000000	222.0000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	56.996900	367.000000	886.0000

In [4]:
```
yulu_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
```

In [5]:
```
yulu_df.isna().sum()
```

Out[5]:
```
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 [6]:
```
# count of no. of rows and coloumns
yulu_df.shape
```

Out[6]:
```
(10886, 12)
```

In [7]:
```
yulu_df['weather'].unique()
```

Out[7]:
```
array([1, 2, 3, 4], dtype=int64)
```

In [8]:
```
yulu_df['weather'].value_counts()
```

Out[8]:
```
1    7192
2    2834
3     859
4       1
Name: weather, dtype: int64
```

In [9]:
```
# minimum datetime and maximum datetime
yulu_df['datetime'].min(), yulu_df['datetime'].max()
```

Out[9]:
```
('01-01-2011 00:00', '19-12-2012 23:00')
```

In [10]:
```
yulu_df['datetime'] = pd.to_datetime(yulu_df['datetime'])

cat_cols= ['season', 'holiday', 'workingday', 'weather']
for col in cat_cols:
    yulu_df[col] = yulu_df[col].astype('object')
```

In [11]:
```
yulu_df.iloc[:, 1:].describe(include='all')
```

Out[11]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
count	10886.0	10886.0	10886.0	10886.0	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000
unique	4.0	2.0	2.0	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	4.0	0.0	1.0	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	2734.0	10575.0	7412.0	7192.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	NaN	NaN	NaN	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191.57413
std	NaN	NaN	NaN	NaN	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.14445
min	NaN	NaN	NaN	NaN	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.00000
25%	NaN	NaN	NaN	NaN	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.00000
50%	NaN	NaN	NaN	NaN	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.00000
75%	NaN	NaN	NaN	NaN	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.00000
max	NaN	NaN	NaN	NaN	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.00000

In [12]:
```
# number of unique values in each categorical columns
yulu_df[cat_cols].melt().groupby(['variable', 'value'])[['value']].count()
```

Out[12]:

		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

In [13]:
```
# count of casual users
yulu_df['casual'].value_counts()
```

Out[13]:
```
0      986
1      667
2      487
3      438
4      354
      ... 
332      1
361      1
356      1
331      1
304      1
Name: casual, Length: 309, dtype: int64
```

In [14]:
```
# count of registered users
yulu_df['registered'].value_counts()
```

Out[14]:
```
3      195
4      190
5      177
6      155
2      150
      ... 
570      1
422      1
678      1
565      1
636      1
Name: registered, Length: 731, dtype: int64
```

In [15]:
```
yulu_df['workingday'].value_counts()
```

Out[15]:
```
1    7412
0    3474
Name: workingday, dtype: int64
```

What we have find in this dataset so far is-

- There is no missing values
- Count of registered users
- count of casual users
- number of unique values in each categorical columns
- minimum datetime and maximum datetime

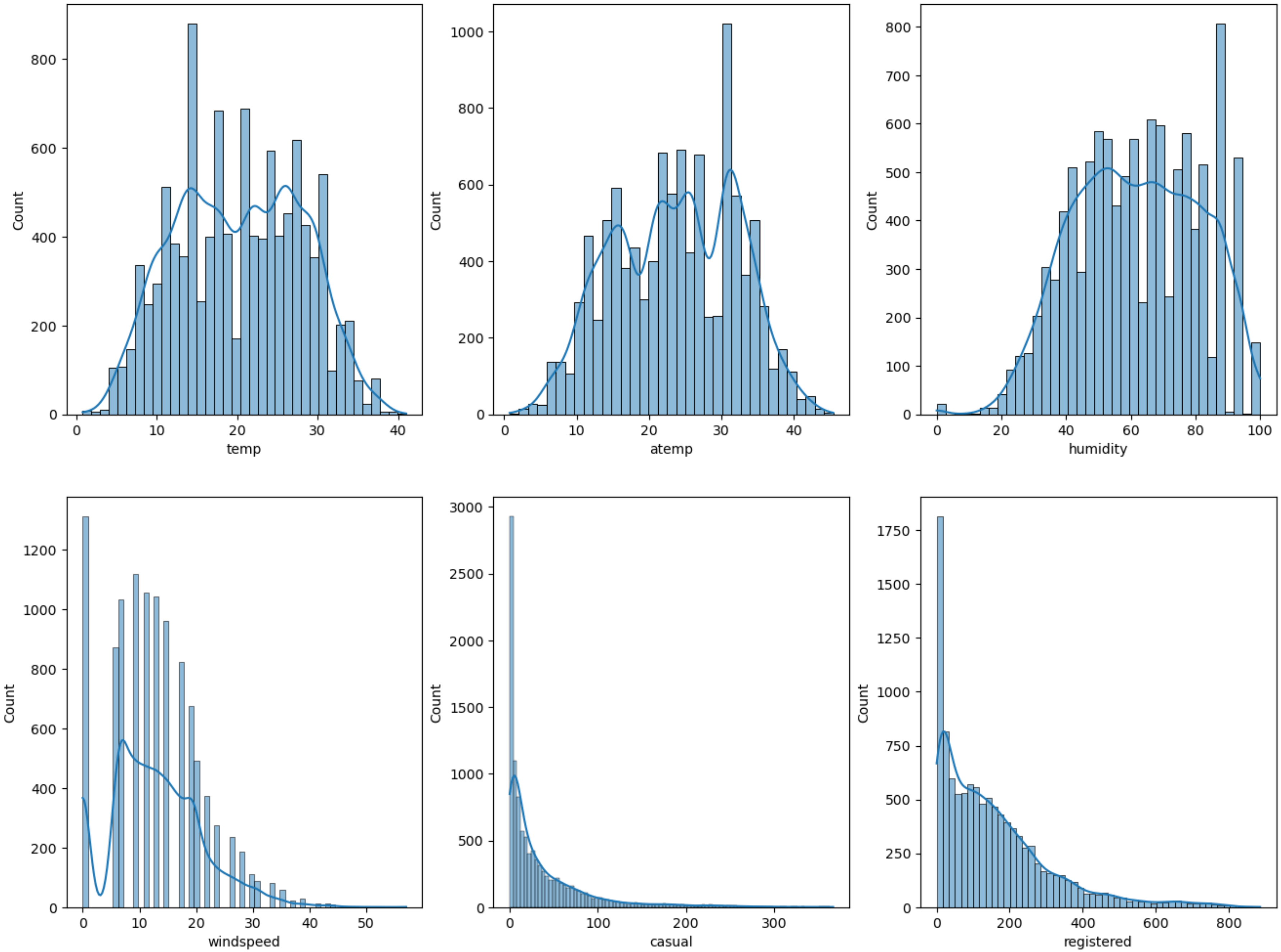
Univariate Analysis

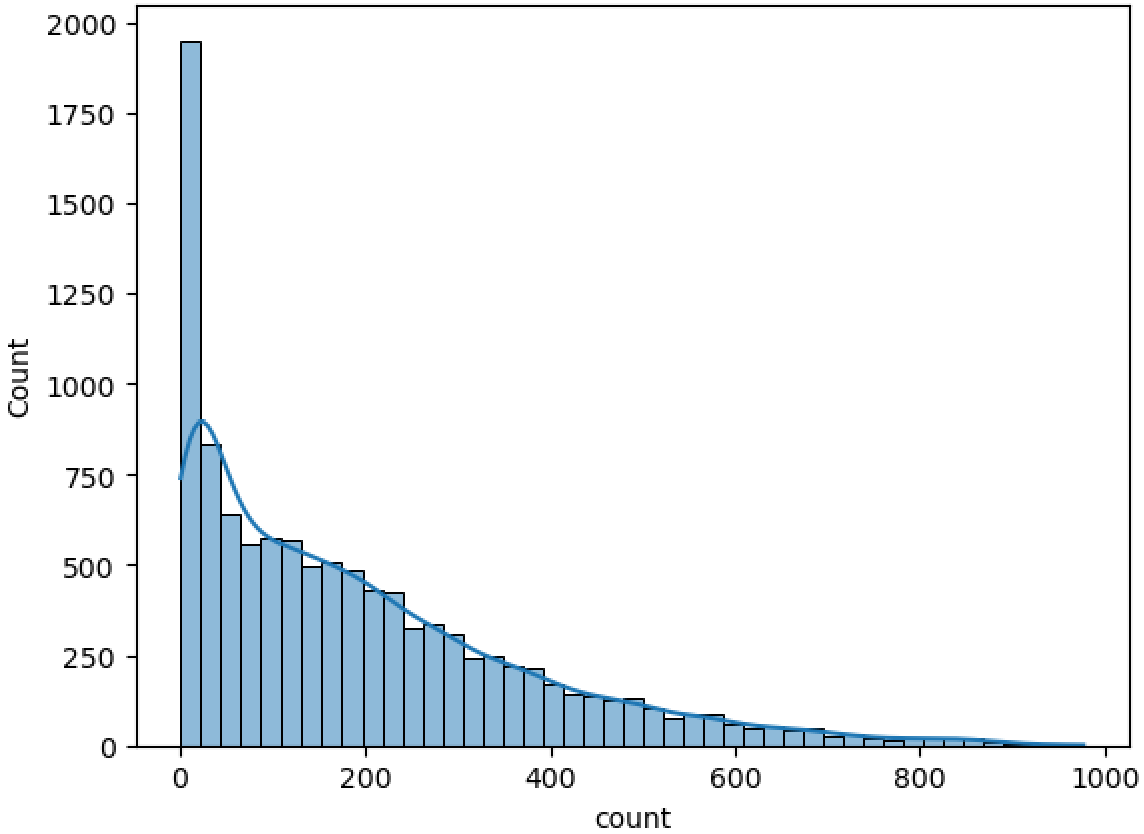
In [16]:
```
# understanding the distribution for numerical variables
num_cols = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered','count']

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(yulu_df[num_cols[index]], ax=axis[row, col], kde=True)
        index += 1

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


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



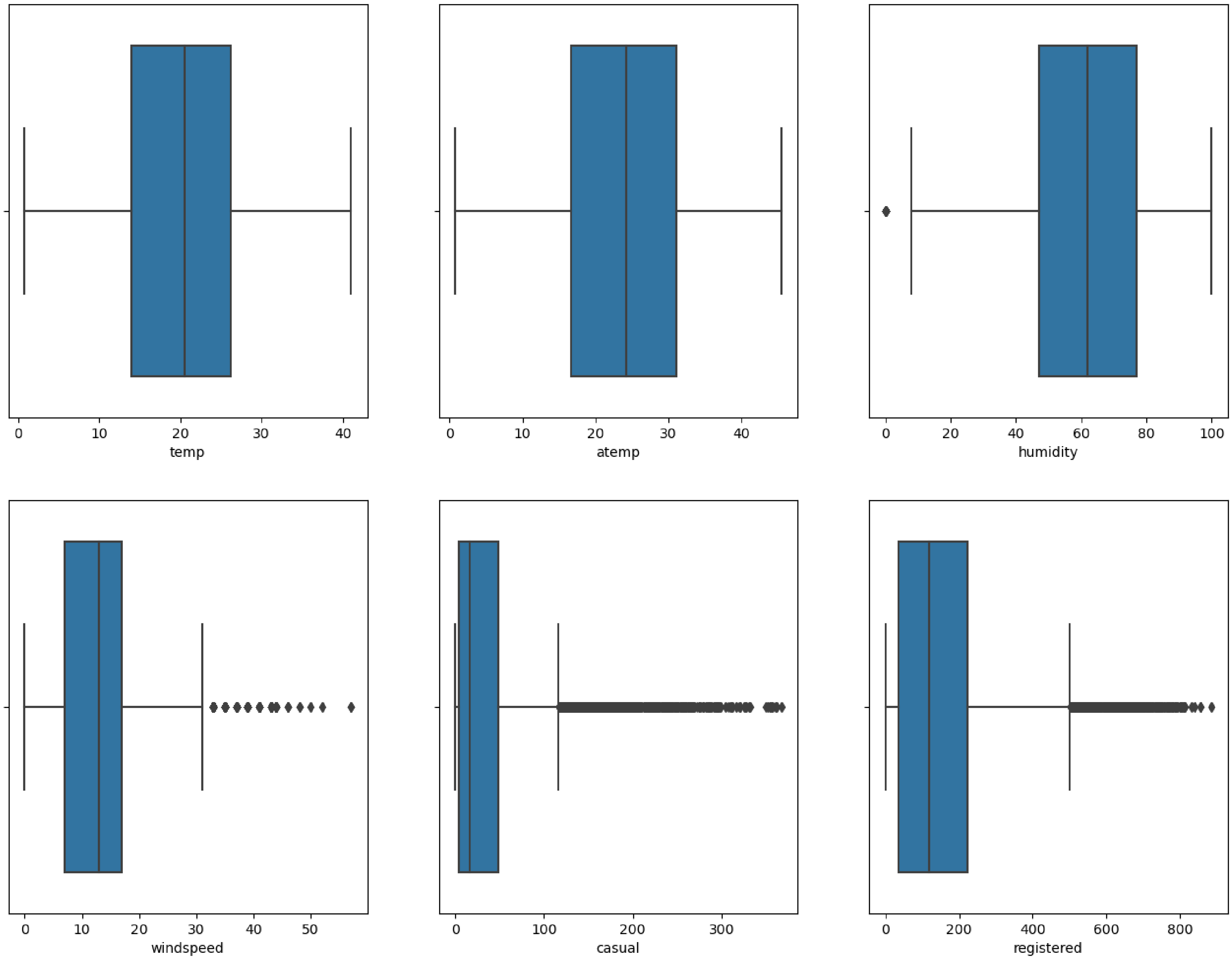
- casual, registered and count somewhat looks like Log Normal Distrinution
- temp, atemp and humidity looks like they follows the Normal Distribution
- windspeed follows the binomial distribution

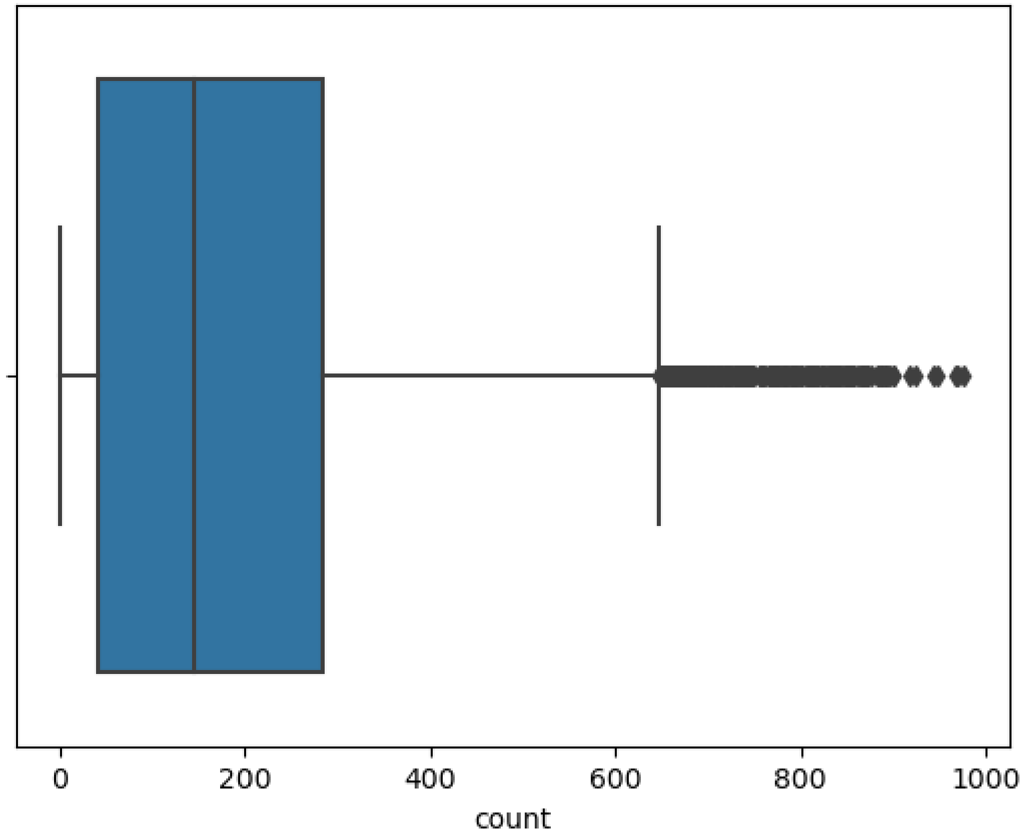
In [17]:

```python
# 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(x=yulu_df[num_cols[index]], ax=axis[row, col])
        index += 1

plt.show()
sns.boxplot(x=yulu_df[num_cols[-1]])
plt.show()
```


0 10 20 30 40
temp
0 10 20 30 40
atemp
0 20 40 60 80 100
humidity
0 10 20 30 40 50
windspeed
0 100 200 300
casual
0 200 400 600 800
registered



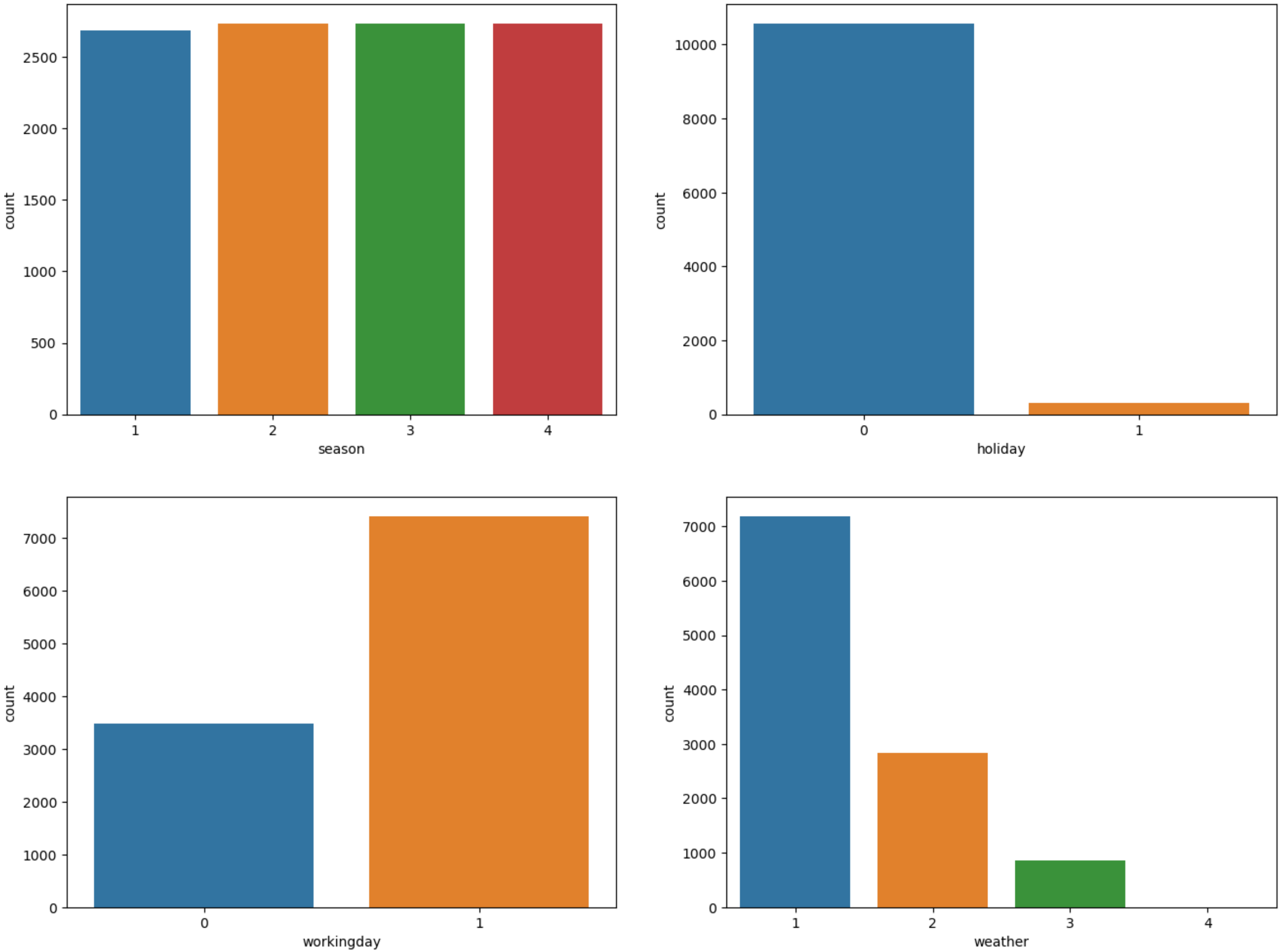
- Looks like humidity, casual, registered and count have outliers in the data.

In [18]:

```
# 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=yulu_df, x=cat_cols[index], ax=axis[row, col])
        index += 1

plt.show()
```



count
2500
2000
1500
1000
500
0
1
2
3
4
season
count
10000
8000
6000
4000
2000
0
0
1
holiday
count
7000
6000
5000
4000
3000
2000
1000
0
0
1
workingday
count
7000
6000
5000
4000
3000
2000
1000
0
1
2
3
4
weather

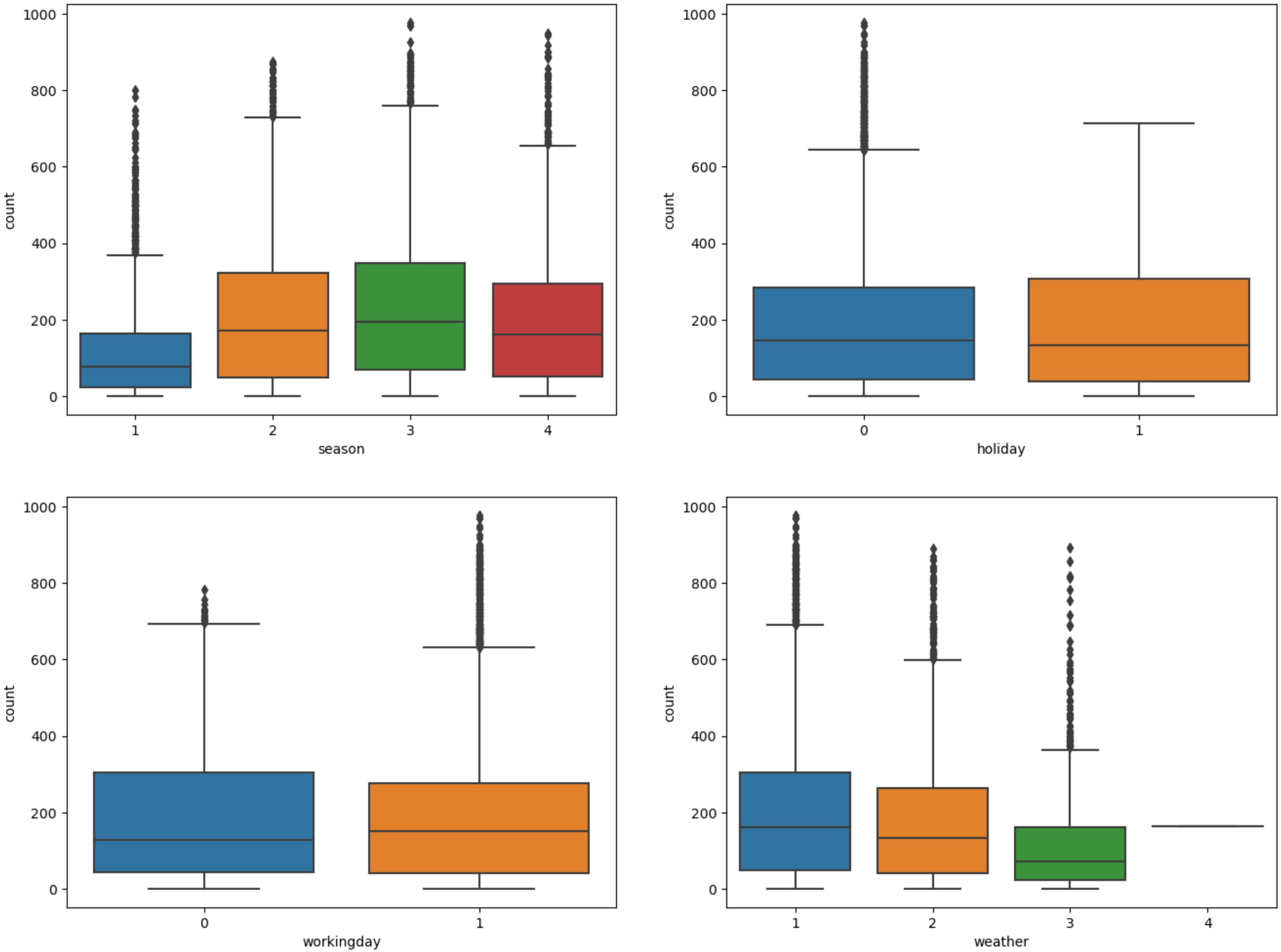
- Data looks common as it should be like equal number of days in each season, more working days and weather is mostly Clear, Few clouds, partly cloudy, partly cloudy.

Bi-variate Analysis

In [19]:
```
# 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=yulu_df, x=cat_cols[index], y='count', ax=axis[row, col])
        index += 1

plt.show()
```

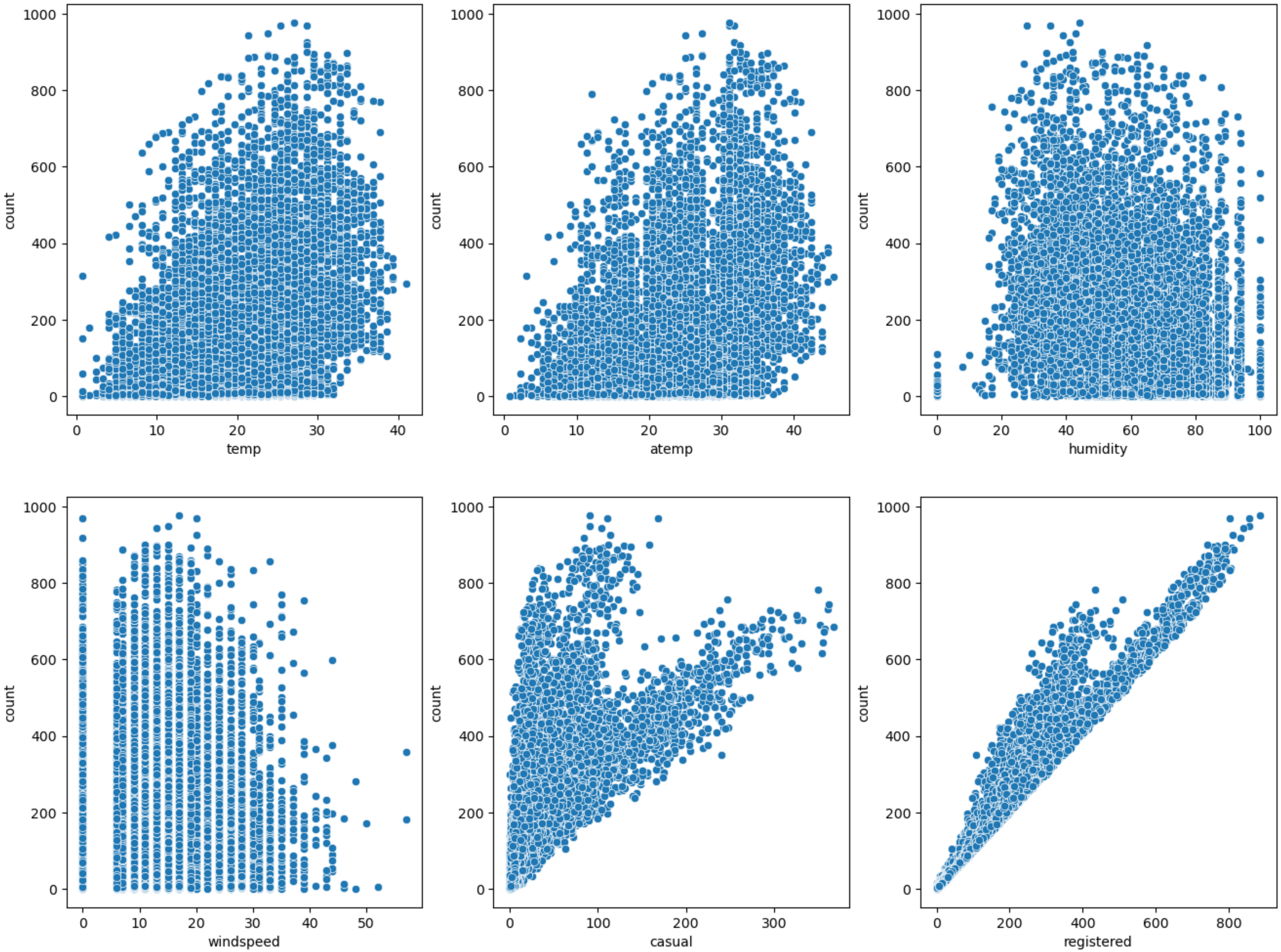

count
season
holiday
workingday
weather
0
200
400
600
800
1000
1
2
3
4
0
1

- 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 [21]:
```
# 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=yulu_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.

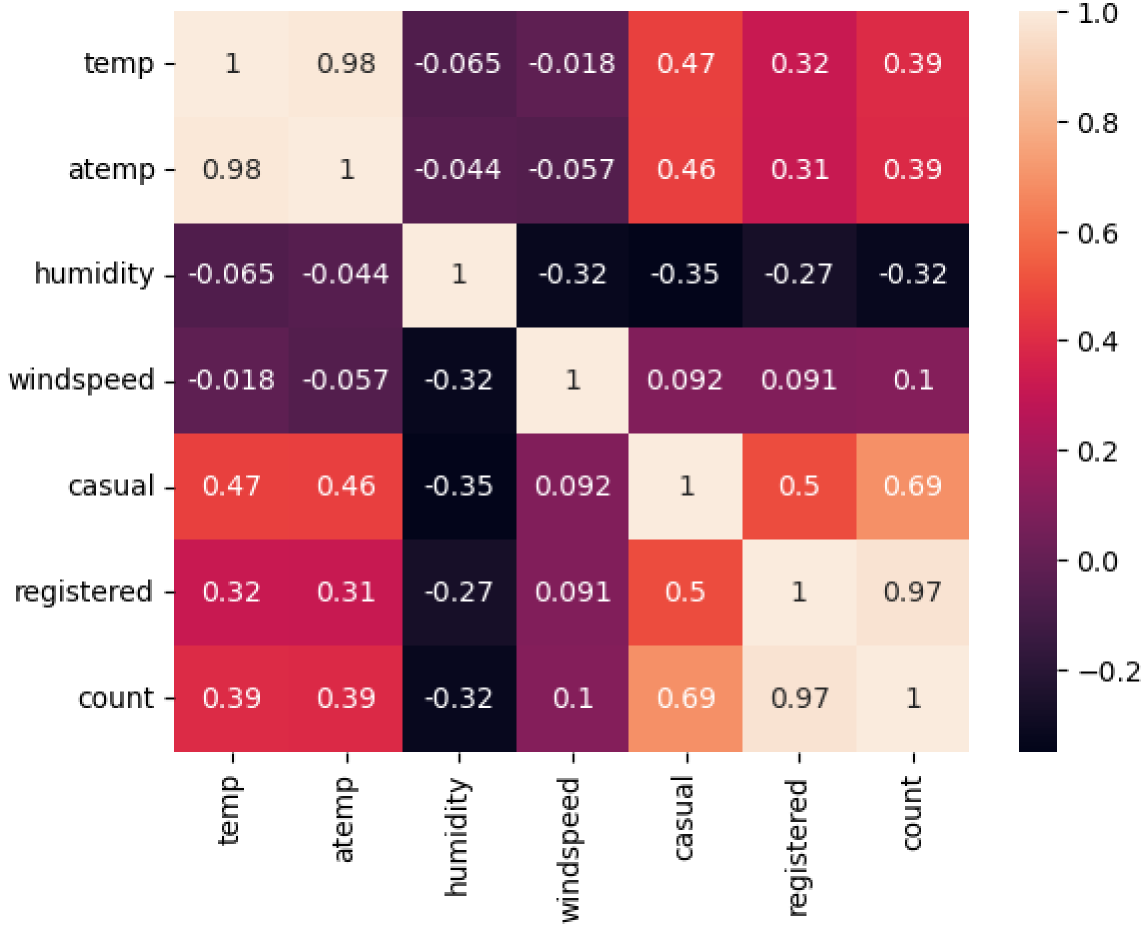
In [22]:
```python
# understanding the correlation between count and numerical variables
yulu_df.corr()['count']
```

```
C:\Users\dell\AppData\Local\Temp\ipykernel_5608\3672462325.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify th
e value of numeric_only to silence this warning.
  yulu_df.corr()['count']
```

Out[22]:
```
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 [23]:
```
sns.heatmap(yulu_df.corr(), annot=True)
plt.show()
```

```
C:\Users\dell\AppData\Local\Temp\ipykernel_5608\1497349677.py:1: FutureWarning: The default value of numeric_only in
DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify th
e value of numeric_only to silence this warning.
  sns.heatmap(yulu_df.corr(), annot=True)
```



Hypothesis Testing

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

Alternate Hypothesis (H1): Weather is not independent of the season

Significance level (alpha): 0.05

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

In [26]:
```
data_table = pd.crosstab(yulu_df['season'], yulu_df['weather'])
print("Observed values:")
data_table
```

```
Observed values:
```

Out[26]:

weather	1	2	3	4
season				
1	1759	715	211	1
2	1801	708	224	0
3	1930	604	199	0
4	1702	807	225	0

In [32]:
```
val = chi2_contingency(data_table)
expected_values = val[3]
expected_values
```

Out[32]:
```
array([[1.77454639e+03, 6.99258130e+02, 2.11948742e+02, 2.46738931e-01],
       [1.80559765e+03, 7.11493845e+02, 2.15657450e+02, 2.51056403e-01],
       [1.80559765e+03, 7.11493845e+02, 2.15657450e+02, 2.51056403e-01],
       [1.80625831e+03, 7.11754180e+02, 2.15736359e+02, 2.51148264e-01]])
```

In [36]:
```
nrows, ncols = 4, 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 = chi2.ppf(q=1-alpha, df=dof)
print(f"critical value: {critical_val}")

p_val = 1-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\
    Weather is dependent on the season.")
else:
    print("Since p-value is greater than the alpha 0.05, We do not reject the Null Hypothesis")
```

```
degrees of freedom:  9
chi-square test statistic:  44.09441248632364
critical value: 16.918977604620448
p-value: 1.3560001579371317e-06

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

Null Hypothesis: Working day has no effect on the number of cycles being rented.

Alternate Hypothesis: Working day has effect on the number of cycles being rented.

Significance level (alpha): 0.05

We will use the **2-Sample T-Test** to test the hypothess defined above

In [38]:
```
data_group1 = yulu_df[yulu_df['workingday']==0]['count'].values
data_group2 = yulu_df[yulu_df['workingday']==1]['count'].values

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

Out[38]:
```
(30171.346098942427, 34040.69710674686)
```

- 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 34040.70 / 30171.35 which is less than 4:1

In [41]:
```
ttest_ind(a=data_group1, b=data_group2, equal_var=True)
```

Out[41]:
```
Ttest_indResult(statistic=-1.2096277376026694, pvalue=0.22644804226361348)
```

Since pvalue is greater than 0.05 so we can not reject the Null hypothesis. We don't have the sufficient evidence to say that working day has effect on the number of cycles being rented.

Null Hypothesis: Number of cycles rented is similar in different weather and season.

Alternate Hypothesis: Number of cycles rented is not similar in different weather and season.

Significance level (alpha): 0.05

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

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

gp1 = yulu_df[yulu_df['weather']==1]['count'].values
gp2 = yulu_df[yulu_df['weather']==2]['count'].values
gp3 = yulu_df[yulu_df['weather']==3]['count'].values
gp4 = yulu_df[yulu_df['weather']==4]['count'].values

gp5 = yulu_df[yulu_df['season']==1]['count'].values
gp6 = yulu_df[yulu_df['season']==2]['count'].values
gp7 = yulu_df[yulu_df['season']==3]['count'].values
gp8 = yulu_df[yulu_df['season']==4]['count'].values

# conduct the one-way anova
f_oneway(gp1, gp2, gp3, gp4, gp5, gp6, gp7, gp8)
```

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, workingday has no 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.

Thank you