yulu-business-case-1

March 23, 2024

1 Business Case: Yulu - Hypothesis Testing

2 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.

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

#1. **Define Problem Statement and perform Exploratory Data Analysis**

Loading The Dataset

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

Let's check the first 5 data

[3]:
```
df.head()
```

[3]:

	datetime	season	holiday	workingday	weather	temp	atemp	\
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	

```
   humidity  windspeed  casual  registered  count
0        81        0.0       3          13     16
1        80        0.0       8          32     40
2        80        0.0       5          27     32
3        75        0.0       3          10     13
4        75        0.0       0           1      1
```

Let's check the full data

[4]:
```
df
```

[4]:
```
                 datetime  season  holiday  workingday  weather   temp  \
0     2011-01-01 00:00:00       1        0           0        1   9.84
1     2011-01-01 01:00:00       1        0           0        1   9.02
2     2011-01-01 02:00:00       1        0           0        1   9.02
3     2011-01-01 03:00:00       1        0           0        1   9.84
4     2011-01-01 04:00:00       1        0           0        1   9.84
…                     …     …       …           …       …     …
10881 2012-12-19 19:00:00       4        0           1        1  15.58
10882 2012-12-19 20:00:00       4        0           1        1  14.76
10883 2012-12-19 21:00:00       4        0           1        1  13.94
10884 2012-12-19 22:00:00       4        0           1        1  13.94
10885 2012-12-19 23:00:00       4        0           1        1  13.12

        atemp  humidity  windspeed  casual  registered  count
0      14.395        81     0.0000       3          13     16
1      13.635        80     0.0000       8          32     40
2      13.635        80     0.0000       5          27     32
3      14.395        75     0.0000       3          10     13
4      14.395        75     0.0000       0           1      1
…       …       …         …       …          …     …
10881  19.695        50    26.0027       7         329    336
10882  17.425        57    15.0013      10         231    241
10883  15.910        61    15.0013       4         164    168
10884  17.425        61     6.0032      12         117    129
10885  16.665        66     8.9981       4          84     88

[10886 rows x 12 columns]
```

Exploring the data

[5]:
```
df.shape
```

[5]:
```
(10886, 12)
```

[6]:
```
df.ndim
```

[6]: 2

[7]:
```
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
```

[8]:
```
df.describe()
```

[8]:

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

	atemp	humidity	windspeed	casual	registered
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
mean	23.655084	61.886460	12.799395	36.021955	155.552177
std	8.474601	19.245033	8.164537	49.960477	151.039033
min	0.760000	0.000000	0.000000	0.000000	0.000000
25%	16.665000	47.000000	7.001500	4.000000	36.000000
50%	24.240000	62.000000	12.998000	17.000000	118.000000
75%	31.060000	77.000000	16.997900	49.000000	222.000000
max	45.455000	100.000000	56.996900	367.000000	886.000000

	count

```
count  10886.000000
mean     191.574132
std      181.144454
min        1.000000
25%       42.000000
50%      145.000000
75%      284.000000
max      977.000000
```

Datatype of following attributes needs to change to proper data type

1. **datetime** - to datetime
2. **season** - to categorical
3. **holiday** - to categorical
4. **workingday** - to categorical
5. **weather** - to categorical

[9]:
```
df['datetime'] = pd.to_datetime(df['datetime'])

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

[10]:
```
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
```

Missing Value Detection:

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

[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
```

[11]:
```

```

There are zero null values across the entire dataset.

[12]:
```
print(df['datetime'].min(), df['datetime'].max())
```

```
2011-01-01 00:00:00 2012-12-19 23:00:00
```

[13]:
```
# number of unique values in each categorical columns
df[cat_cols].melt().groupby(['variable','value'])[['value']].count()
```

[13]:

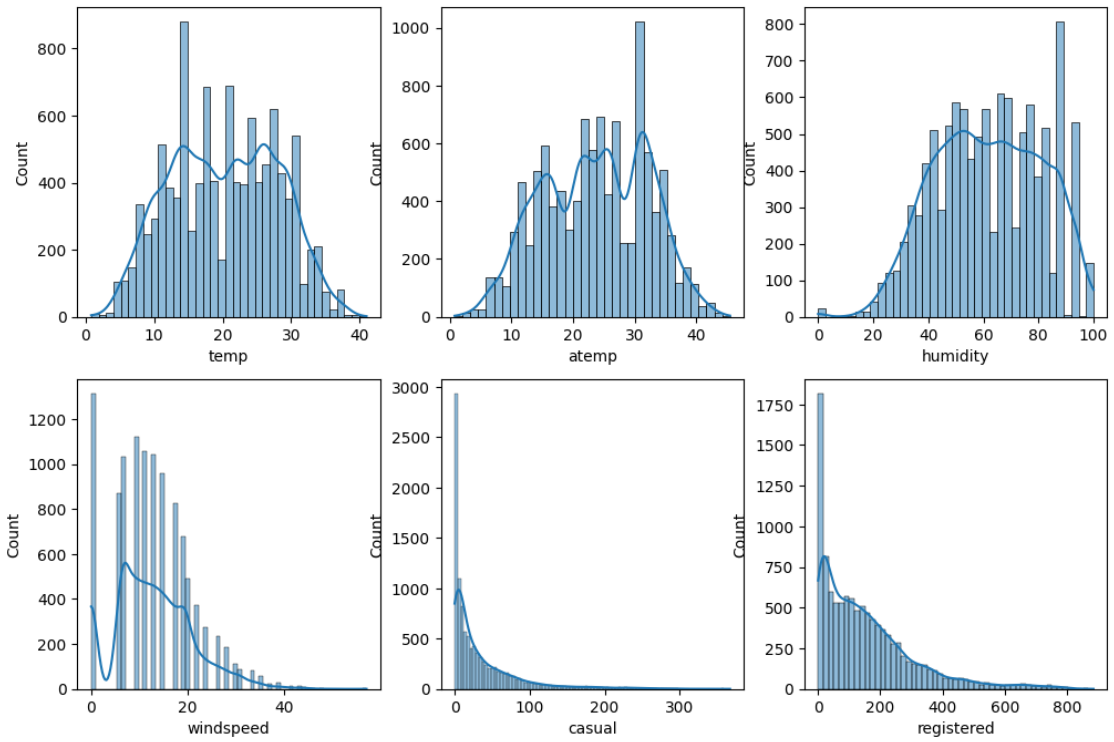
variable	value	value
holiday	0	10575
	1	311
season	1	2686
	2	2733
	3	2733
	4	2734
weather	1	7192
	2	2834
	3	859
	4	1
workingday	0	3474
	1	7412

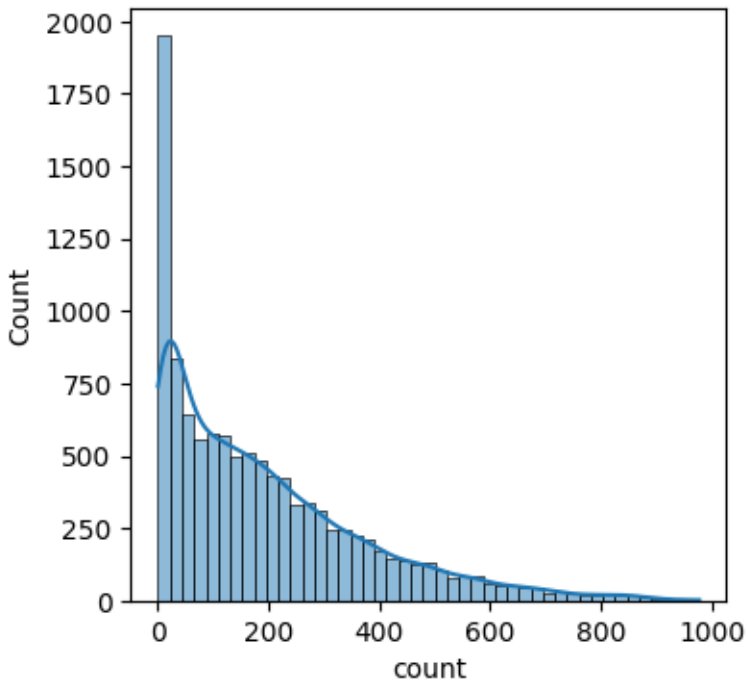
3 Distribution plots of all the continuous variable(s) barplots/countplots of all the categorical variables

3.1 Univariate Analysis

Histplot

[14]:
```
num_cols = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered',␣
 ↪'count']
fig,axis = plt.subplots(nrows= 2, ncols = 3, figsize = (12,8))
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()
plt.figure(figsize=(4, 4))
sns.histplot(df[num_cols[-1]], kde = True)
plt.show()
```

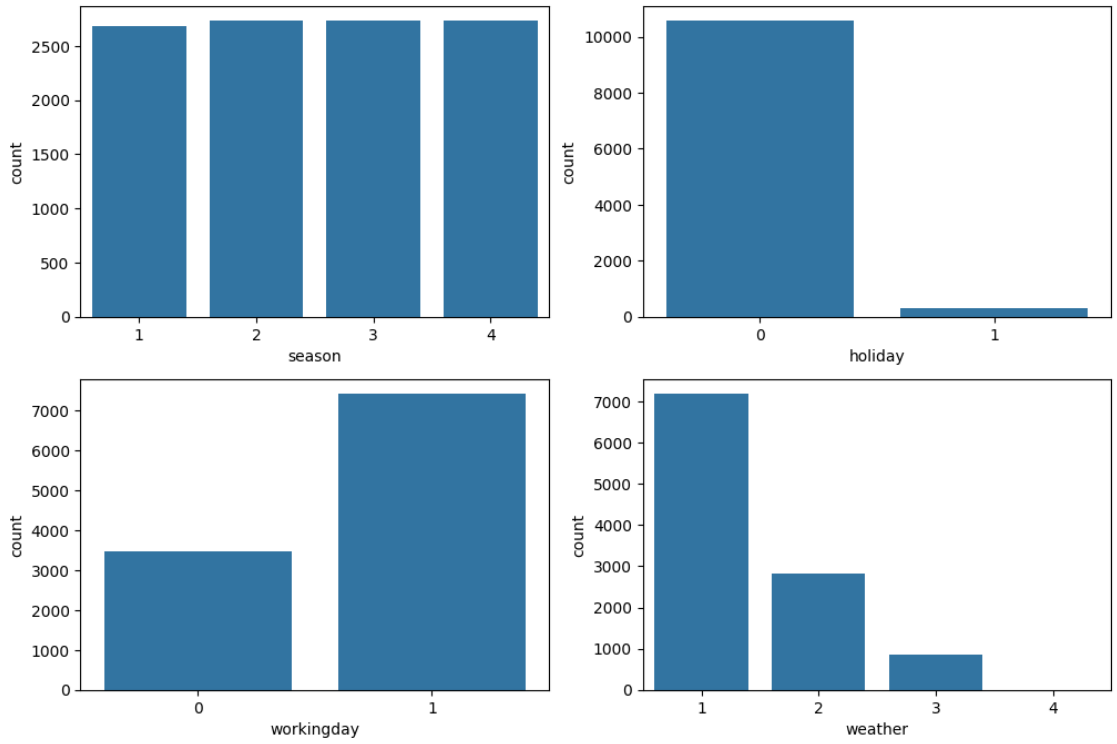




1. **casual**, **registered** and **count** somewhat looks like Log Normal Distribution
2. **temp**, **atemp** and **humidity** looks like they follows the Normal Distribution
3. **windspeed** follows the binomial distribution

Count Plot

[15]:
```
cat_cols= ['season','holiday','workingday','weather']
fig,axis = plt.subplots(nrows= 2, ncols = 2, figsize = (12,8))
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()
```

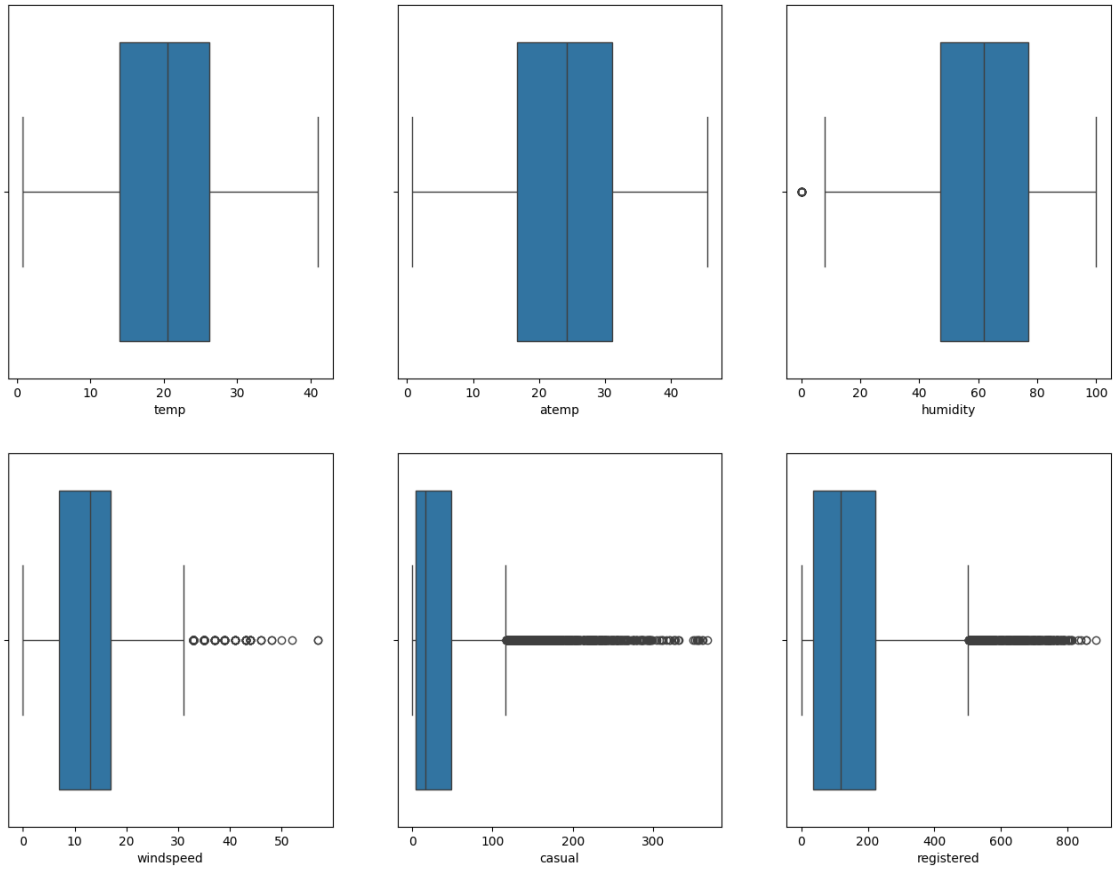


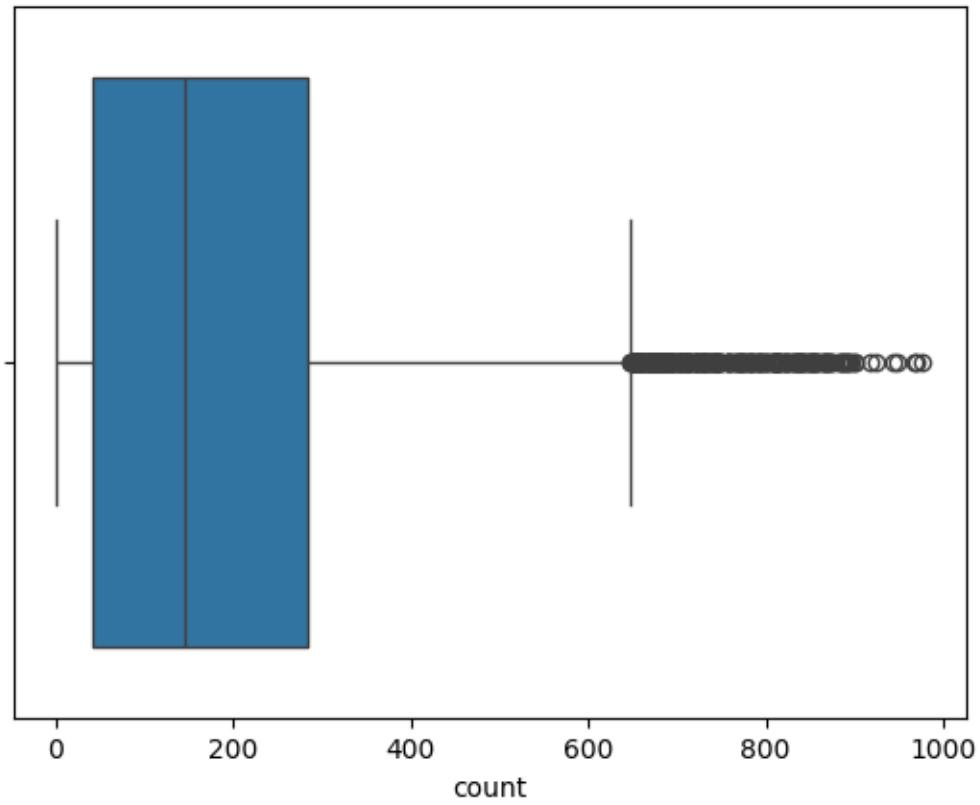
Data looks common as it should be like equal number of days in each season, more working days and weather is mostly Clear, Few clouds and partly cloudy.

Boxplot

Using box plots to visualize the distributions and identify the outliers

[16]:
```
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.boxplot(data = df, x = num_cols[index], ax = axis[row, col])
    index += 1
plt.show()
sns.boxplot(x = df[num_cols[-1]])
plt.figure(figsize=(4, 4))
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




```
<Figure size 400x400 with 0 Axes>
```

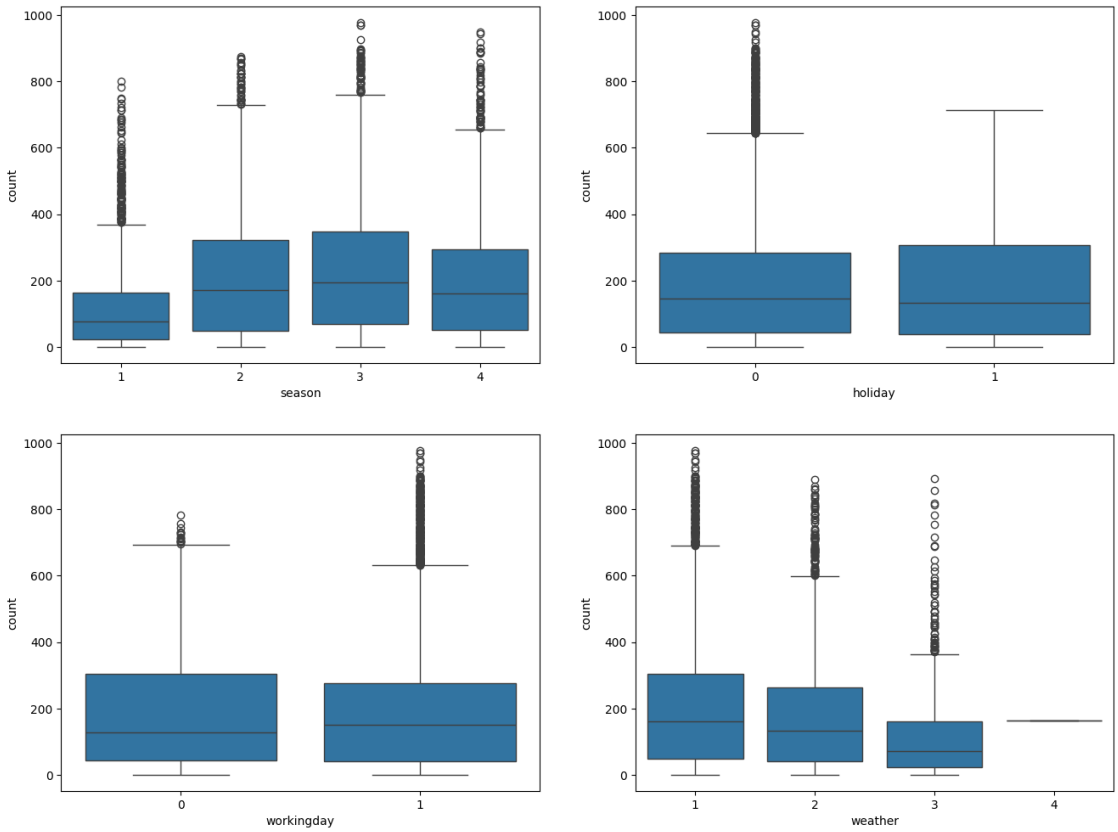
Looks like humidity, casual, registered and count have outliers in the data.

4 Relationships between important variables such as workday and count, season and count, weather and count.

5 Bivariate Analysis

Boxplot

[17]:
```
cat_cols= ['season','holiday','workingday','weather']
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()
```

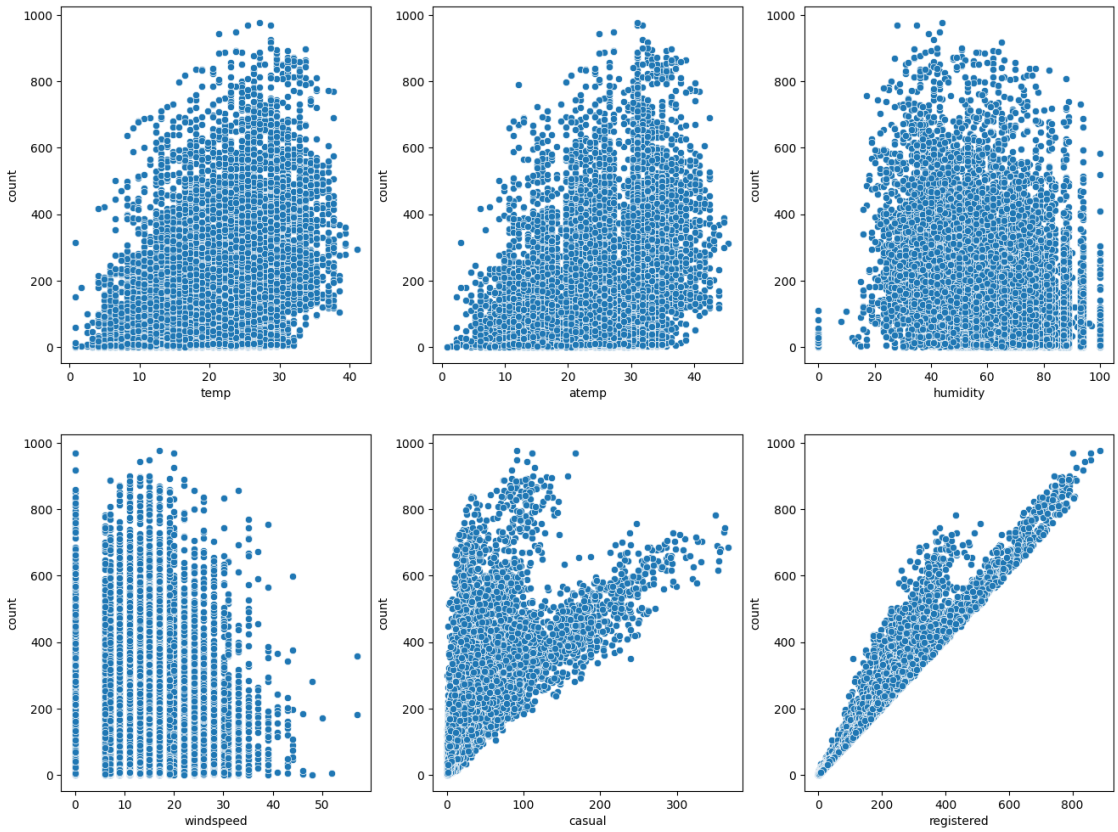


- In **summer** and **fall** seasons more bikes are rented as compared to other seasons.
- Whenever its a holiday more bikes are rented.
- It is also clear from the workingday also that whenever day is holiday or weekend, slightly more bikes were rented.
- Whenever there is **rain**, **thunderstorm**,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

Scatterplot

To identify a possible relationship between changes observed in different sets of variables.

[18]:
```
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.scatterplot(data = df, x = num_cols[index], y = "count", ax = axis[row,␣
 ↪col])
    index += 1
plt.show()
```



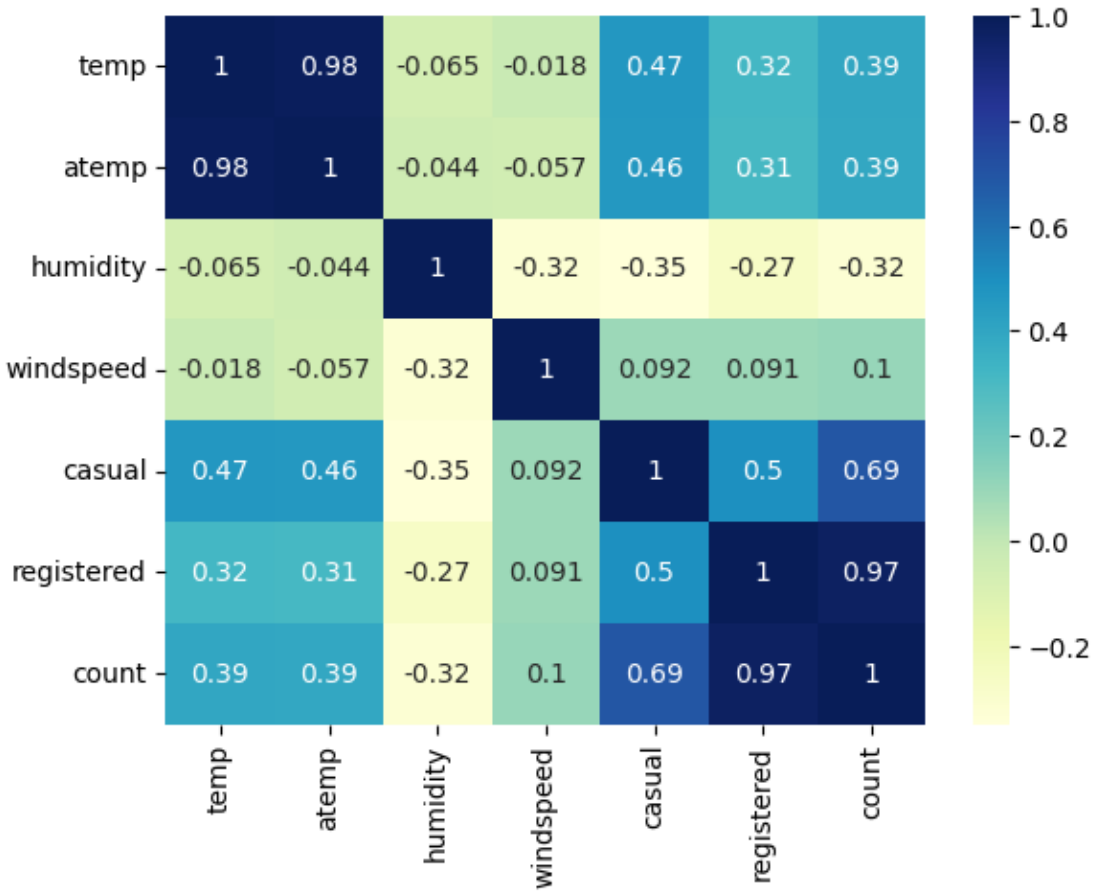
- Whenever the humidity is less than 20, number of bikes rented is very very low.
- Whenever the temperature is less than 10, number of bikes rented is less.
- Whenever the windspeed is greater than 35, number of bikes rented is less.

Heatmap

Understanding the correlation between count and numerical variables.

[19]:
```
df.corr()['count']
sns.heatmap(df.corr(), annot= True, cmap= "YlGnBu")
plt.show()
```

```
<ipython-input-19-c3b270b3075b>: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 the value of numeric_only
to silence this warning.
  df.corr()['count']
<ipython-input-19-c3b270b3075b>: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 the value of numeric_only
to silence this warning.
  sns.heatmap(df.corr(), annot= True, cmap= "YlGnBu")
```



Variables are positively correlated here, where one increases and the other increases too. In contrast, some are negatively correlated also where the high values of one variable go with the low values of another variable. corr= +1 indicates perfect positive correlation.

6 2.Hypothesis Testing

7 2- Sample T-Test

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

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.

[20]:
```
data_group1 = df[df['workingday']==0]['count'].values
data_group2 = df[df['workingday']==1]['count'].values
```

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

```
30171.346098942427 34040.69710674686
```

[20]:
```
1.0
```

Here, the ratio is 34040.70 / 30171.35 which is less than 4:1

[21]:
```
statistic, p_value = stats.ttest_ind(a=data_group1, b=data_group2,␣
 ↪equal_var=True)
print("statistic: ", statistic)
print("p-value: ", p_value)
```

```
statistic:  -1.2096277376026694
p-value:  0.22644804226361348
```

[22]:
```
if p_value < 0.05:
    print("Reject H0")
    print("Atleast one group have different mean")
else:
    print("Fail to reject H0")
    print("All groups have same mean")
```

```
Fail to reject H0
All groups have same mean
```

Since pvalue is greater than 0.05 so we cannot 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.

8 ANOVA

ANNOVA to check if No. of cycles rented is similar or different in different: 1. weather 2. season

ANOVA assumptions

ANOVA compares the means of different groups and shows you if there are any statistical differences between the means. ANOVA is classified as an omnibus test statistic. This means that it can't tell you which specific groups were statistically significantly different from each other, only that at least two of the groups were.

ANOVA relies on three main assumptions that must be met for the test results to be valid.

Normality

The first assumption is that the groups each fall into what is called a normal distribution. This means that the groups should have a bell-curve distribution with few or no outliers.

Homogeneity of variance

Also known as homoscedasticity, this means that the variances between each group are the same.

Independence

The final assumption is that each value is independent from each other. This means, for example, that unlike a conjoint analysis the same person shouldn't be measured multiple times.

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.

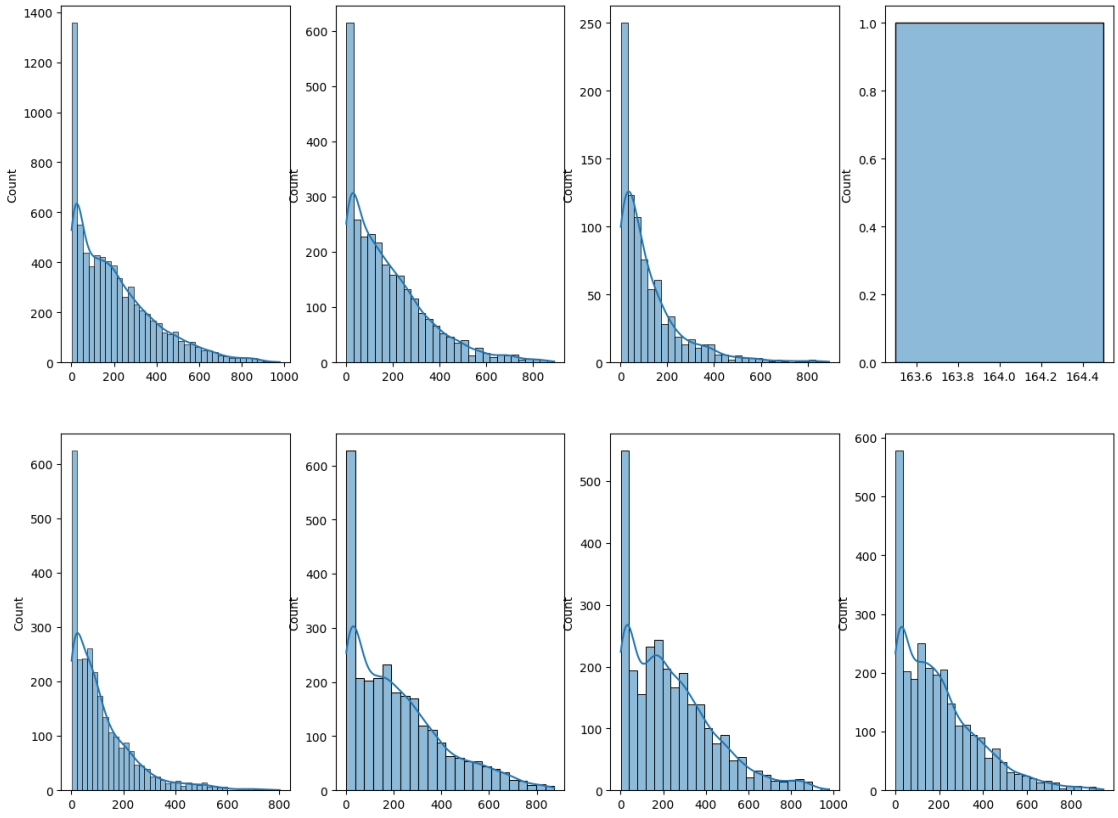
Significance level (alpha): 0.05

[23]:
```
# defining the data groups for the ANOVA
from statsmodels.graphics.gofplots import qqplot
gp1 = df[df['weather']==1]['count'].values
gp2 = df[df['weather']==2]['count'].values
gp3 = df[df['weather']==3]['count'].values
gp4 = df[df['weather']==4]['count'].values

gp5 = df[df['season']==1]['count'].values
gp6 = df[df['season']==2]['count'].values
gp7 = df[df['season']==3]['count'].values
gp8 = df[df['season']==4]['count'].values
groups=[gp1,gp2,gp3,gp4,gp5,gp6,gp7,gp8]
```

[24]:
```
fig,axis = plt.subplots(nrows= 2, ncols = 4, figsize = (16,12))
index = 0
for row in range(2):
  for col in range(4):
    sns.histplot(groups[index], ax = axis[row, col], kde = True)
    index += 1
plt.show
```

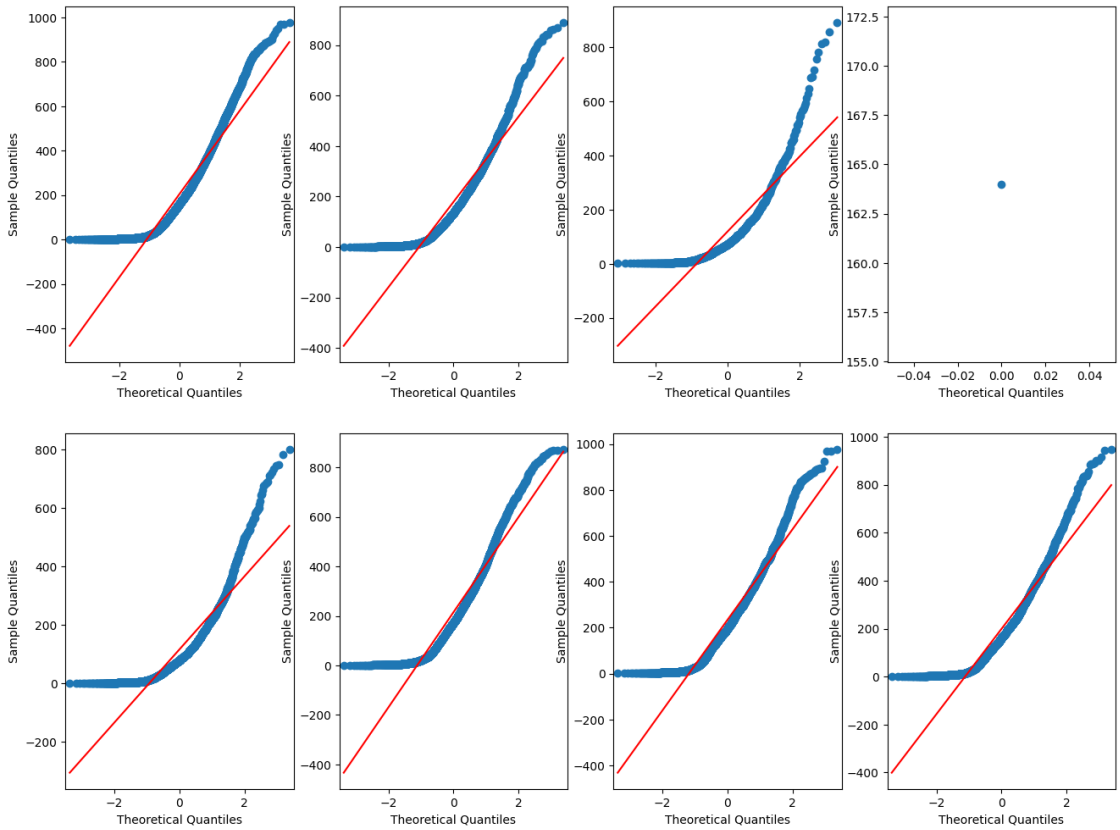
[24]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



As one group above in the graph does not reflect as gaussian, So above data also fails the normality test.

[25]:
```
fig,axis = plt.subplots(nrows= 2, ncols = 4, figsize = (16,12))
index = 0
for row in range(2):
  for col in range(4):
    qqplot(groups[index], line = "s", ax = axis[row, col])
    index += 1
plt.show
```

[25]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



A good way to assess the normality of a data would be to use a Q-Q plot, which gives us a graphical visualization of normality. But, here above Q-Q plot also fails to follow the normality test.

Equal variance: Levene's Test

Null Hypothesis: Variances is similar in different weather and season.

Alternate Hypothesis: Variances is not similar in different weather and season.

Significance level (alpha): 0.05

[26]:
```
levene_stat, p_value = stats.levene(gp1,gp2,gp3,gp4,gp5,gp6,gp7,gp8)
print(p_value)
```

```
3.463531888897594e-148
```

[27]:
```
if p_value < 0.05:
 print("Reject the Null hypothesis.Variances are not equal")
else:
 print("Fail to Reject the Null hypothesis.Variances are equal")
```

```
Reject the Null hypothesis.Variances are not equal
```

There are two tests that you can run that are applicable when the assumption of homogeneity of variances has been violated: (1) Welch or (2) Brown and Forsythe test.

Alternatively, you could run a Kruskal-Wallis H Test. For most situations it has been shown that the Welch test is best.

[28]:
```
#assumptions of ANOVA don't hold, we need Kruskal Wallis
kruskal_stat, p_value = stats.kruskal(gp1,gp2,gp3,gp4,gp5,gp6,gp7,gp8)
print("p_value===",p_value)
if p_value<0.05:
  print("Since p-value is less than 0.05, we reject the null hypothesis")
```

```
p_value=== 4.614440933900297e-191
Since p-value is less than 0.05, we reject the null hypothesis
```

9 Chi-square test

Chi-square test to check if Weather is dependent on the season.

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

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

Significance level (alpha): 0.05

[29]:
```
data_table = pd.crosstab(df['season'], df['weather'])
print("Observed values:")
data_table
```

```
Observed values:
```

[29]:

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

[30]:
```
val = stats.chi2_contingency(data_table)
print(val)
```

```
Chi2ContingencyResult(statistic=49.158655596893624,
pvalue=1.549925073686492e-07, dof=9, expected_freq=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]]))
```

[31]:
```
expected_values = val[3]
print(expected_values)
```

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

```
[[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]]
degrees of freedom:  9
```

[32]:
```
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}")
```

```
chi-square test statistic:  44.09441248632364
critical value: 16.918977604620448
p-value: 1.3560001579371317e-06
```

[33]:
```
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")
```

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

10 Insights

1. In summer and fall seasons more bikes are rented as compared to other seasons.
2. Whenever its a holiday more bikes are rented.
3. It is also clear from the workingday also that whenever day is holiday or weekend, slightly more bikes were rented.
4. Whenever there is rain, thunderstorm, snow or fog, there were less bikes were rented.
5. Whenever the humidity is less than 20, number of bikes rented is very very low.
6. Whenever the temperature is less than 10, number of bikes rented is less.
7. Whenever the windspeed is greater than 35, number of bikes rented is less.

8. A 2-sample T-test on working and non-working days with respect to count,implies that the mean population count of both categories are the same.
9. By performing a Chi2 test on season and weather (categorical variables), we can infer that there is an impact on weather dependent on season.

11 Recommendations

1. 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.
2. With a significance level of 0.05, workingday has no effect on the number of bikes being rented.
3. In very low humid days, company should have less bikes in the stock to be rented.
4. Whenever temperature is less than 10 or in very cold days, company should have less bikes.
5. Whenever the windspeed is greater than 35 or in thunderstorms, company should have less bikes in stock to be rented.