yulu-hypothesis-testing

March 1, 2024

1 Defining problem statement and analysizing basics metrics.

2 Business Problem

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.

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 ?

3 Importing dataset and libraries.

[4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import math
```

[5]:
```
df = pd.read_csv(r"bike_sharing.csv")
```

4 Analysing the basic metrics

[6]:
```
df.head()
```

[6]:
```
              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
```

[7]:
```
df.shape
```

[7]:
```
(10886, 12)
```

[8]:
```
print(f"Number of rows : {df.shape[0]}")
print(f"Number of columns : {df.shape[1]}")
```

```
Number of rows : 10886
Number of columns : 12
```

[9]:
```
df.columns
```

[9]:
```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count'],
      dtype='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  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
```

Changing datatype of below attributes : -

- datetime - to datetime
- season - to categorical

- holiday - to categorical
- workingday - to categorical
- weather - to categorical

[11]:
```
df['datetime'] = pd.to_datetime(df['datetime'])
cat_cols= ['season', 'holiday', 'workingday', 'weather']
for col in cat_cols:
  df[col] = df[col].astype('object')
```

[12]:
```
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
```

[13]:
```
df.isna().sum()
```

[13]:
```
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
```

[14]:
```
df.nunique()
```

[14]:
```
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
```

[15]:
```
def is__unique(i):
  print(df[i].unique())
```

[16]:
```
cols = ["season","holiday","workingday","weather"]

for ele in cols:
  print(ele)
  print(is__unique(ele))
  print("**************************")
```

```
season
[1 2 3 4]
None
**************************
holiday
[0 1]
None
**************************
workingday
[0 1]
None
**************************
weather
[1 2 3 4]
None
**************************
```

[17]:
```
def distribution(i):
  print(df[i].value_counts(normalize = True)*100)


columns = ["season","holiday","workingday","weather"]

```

```
for ele in columns:
  print(ele)
  print()
  print(distribution(ele))
  print("***************************")
```

```
season

4    25.114826
2    25.105640
3    25.105640
1    24.673893
Name: season, dtype: float64
None
***************************
holiday

0    97.14312
1     2.85688
Name: holiday, dtype: float64
None
***************************
workingday

1    68.087452
0    31.912548
Name: workingday, dtype: float64
None
***************************
weather

1    66.066507
2    26.033437
3     7.890869
4     0.009186
Name: weather, dtype: float64
None
***************************
```

[18]:

```
df.describe()
```

[18]:

```
              temp         atemp      humidity     windspeed        casual  \
count  10886.00000  10886.000000  10886.000000  10886.000000  10886.000000
mean      20.23086     23.655084     61.886460     12.799395     36.021955
std        7.79159      8.474601     19.245033      8.164537     49.960477
min        0.82000      0.760000      0.000000      0.000000      0.000000
25%       13.94000     16.665000     47.000000      7.001500      4.000000
```

```
50%       20.50000    24.240000    62.000000    12.998000    17.000000
75%       26.24000    31.060000    77.000000    16.997900    49.000000
max       41.00000    45.455000   100.000000    56.996900   367.000000

         registered         count
count  10886.000000  10886.000000
mean     155.552177    191.574132
std      151.039033    181.144454
min        0.000000      1.000000
25%       36.000000     42.000000
50%      118.000000    145.000000
75%      222.000000    284.000000
max      886.000000    977.000000
```

1. There are no missing values in the dataset.
2. Casual and registered attributes might have outliers because their mean and median are very far away to one another and the value of standard deviation is also high which tells us that there is high variance in the data of these attributes.

5 Minimum datetime and maximum datetime

[19]:
```
print(df['datetime'].min(), df['datetime'].max())
# number of unique values in each categorical columns
df[cat_cols].melt().groupby(['variable', 'value'])[['value']].count()
```

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

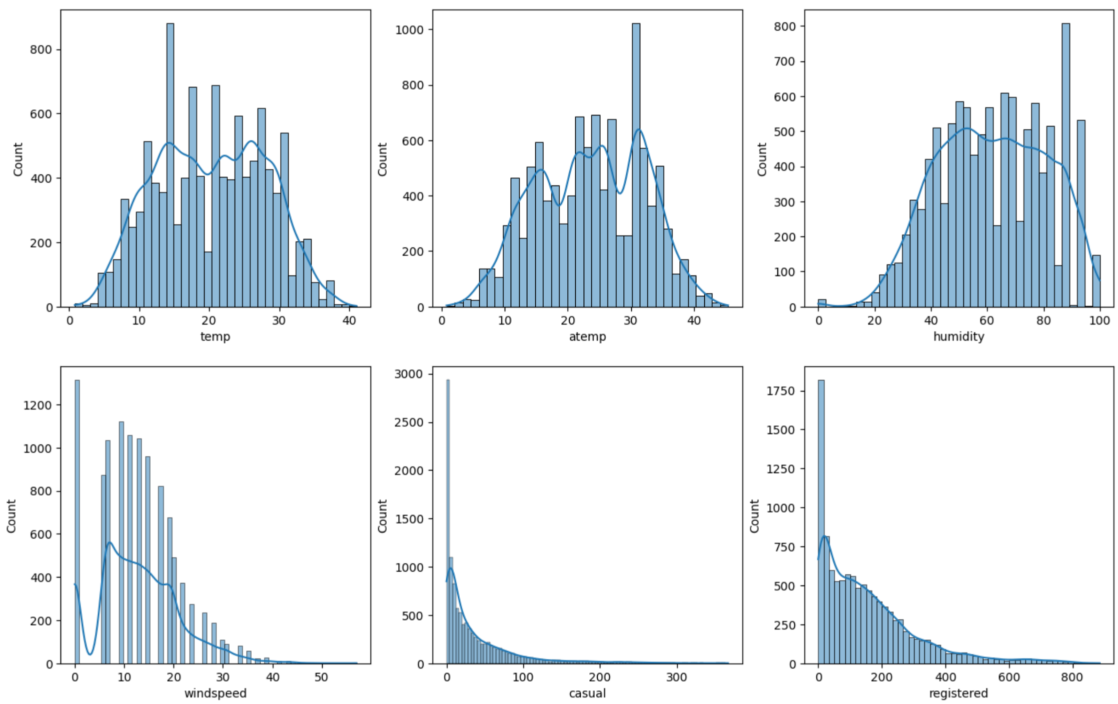
[19]:
```
                   value
variable   value
holiday    0       10575
           1         311
season     1        2686
           2        2733
           3        2733
           4        2734
weather    1        7192
           2        2834
           3         859
           4           1
workingday 0        3474
           1        7412
```

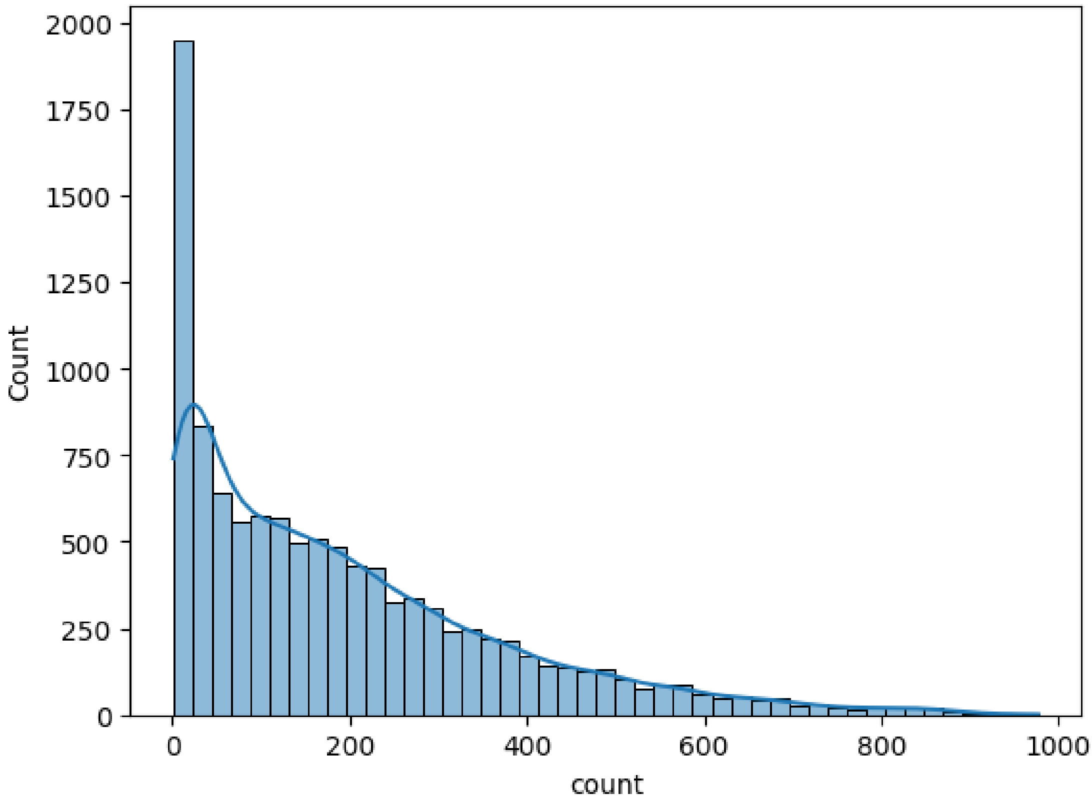
6 Univariate Analysis:

Try establishing a relation between the dependent and independent variable (Dependent "Count" & Independent: Workingday, Weather, Season etc)

[20]:

```
# 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, 10))
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()
```

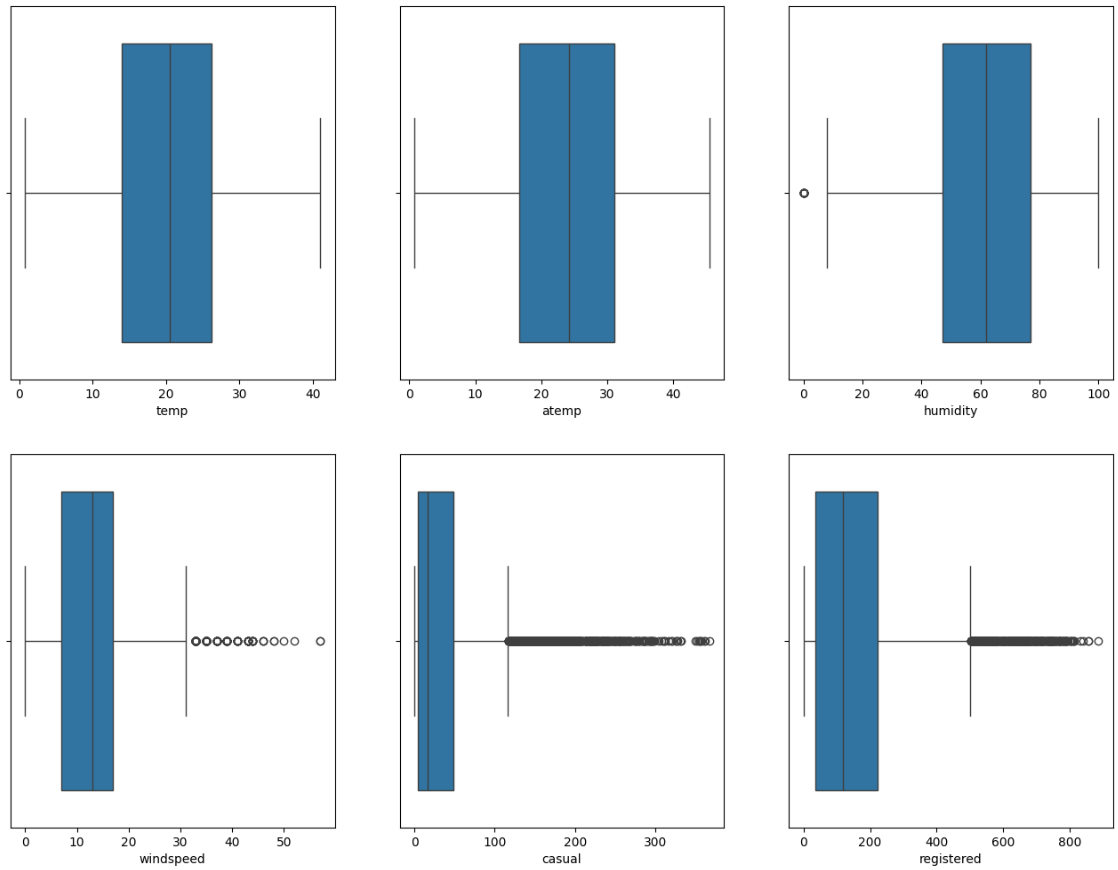


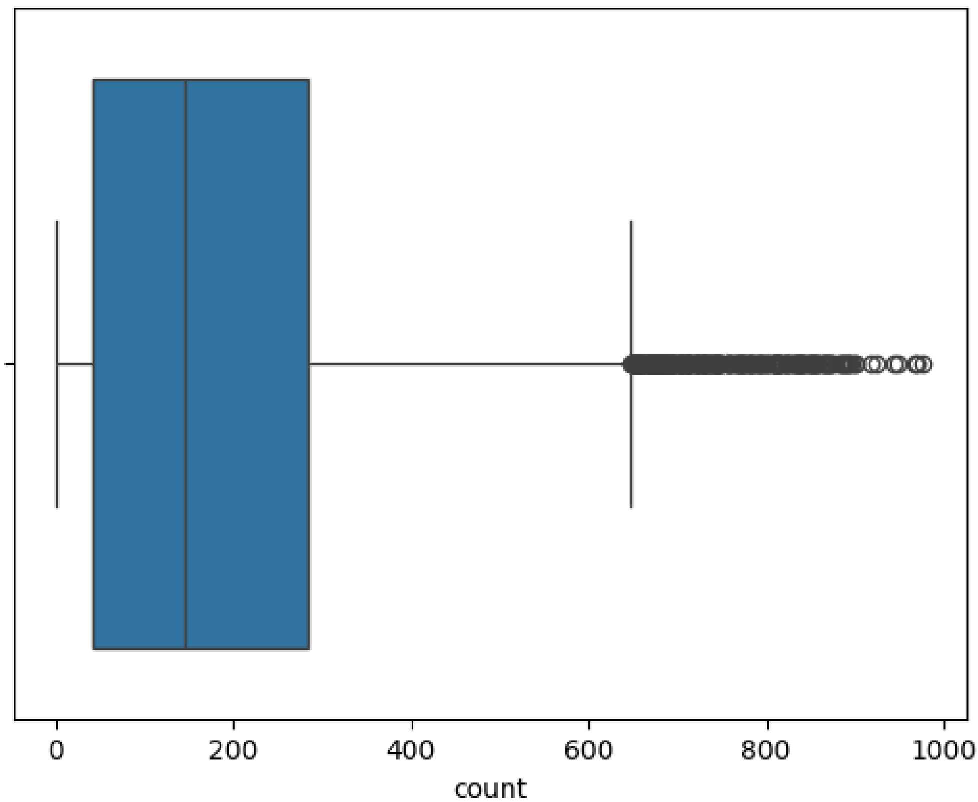


- Casual, registered and count somewhat looks like Log Normal Distribution.
- Temp, atemp and humidity looks like they follows the Normal Distribution.
- Windspeed follows the binomial distribution.

6.1 Plotting box plots to detect outliers in the data

[21]:
```
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=df[num_cols[index]], ax=axis[row, col])
    index += 1
plt.show()
sns.boxplot(x=df[num_cols[-1]])
plt.show()
```


temp
atemp
humidity
windspeed
casual
registered

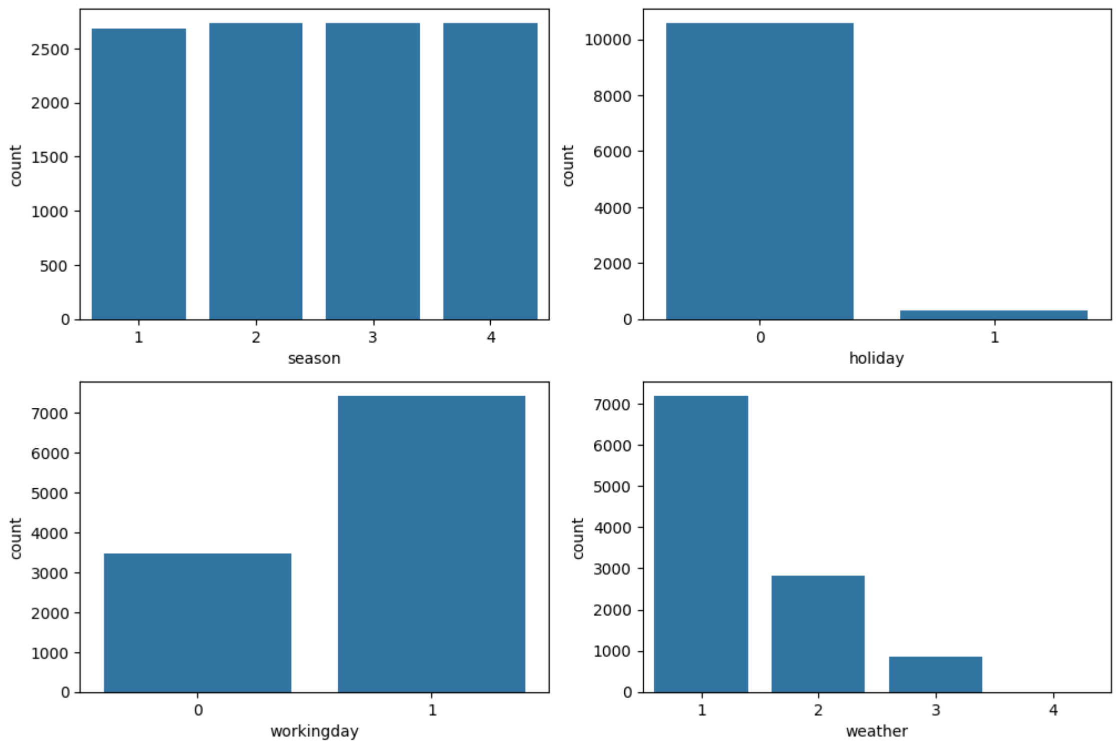


- Number of casual users and registered users keep changing based on different factors like weather, season. Hence a lot of outliers are seen in these two attributes.
- Windspeed changes as per change in weather. Rainy season has more windspeed as compared to summer. This might be the reason for outliers in windspeed data.

6.2 Countplot of each categorical column

[22]:
```
df.head()

df[df["workingday"] == 1]["count"].sum()

df[(df["workingday"] == 1) & (df["registered"])]["count"].sum()


df[df["workingday"] == 0]["count"].sum()

df[df["holiday"] == 1]["count"].sum()


df[df["holiday"] == 0]["count"].sum()
```

```
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()
```

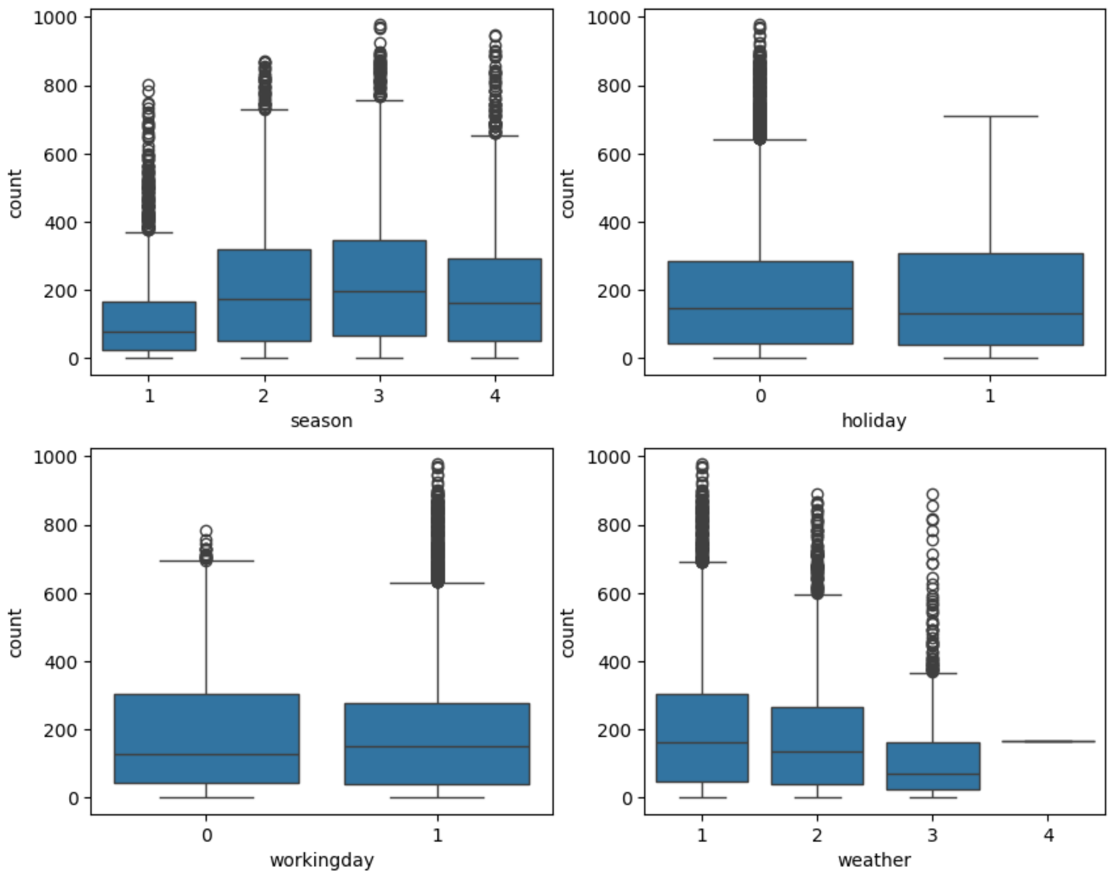


Data looks common as it should be like equal number of days in each season, more working days and weather is mostly Clear, Few clouds, partly cloudy, partly cloudy.

7 Bi-variate Analysis

7.1 Plotting categorical variables againt count using boxplots

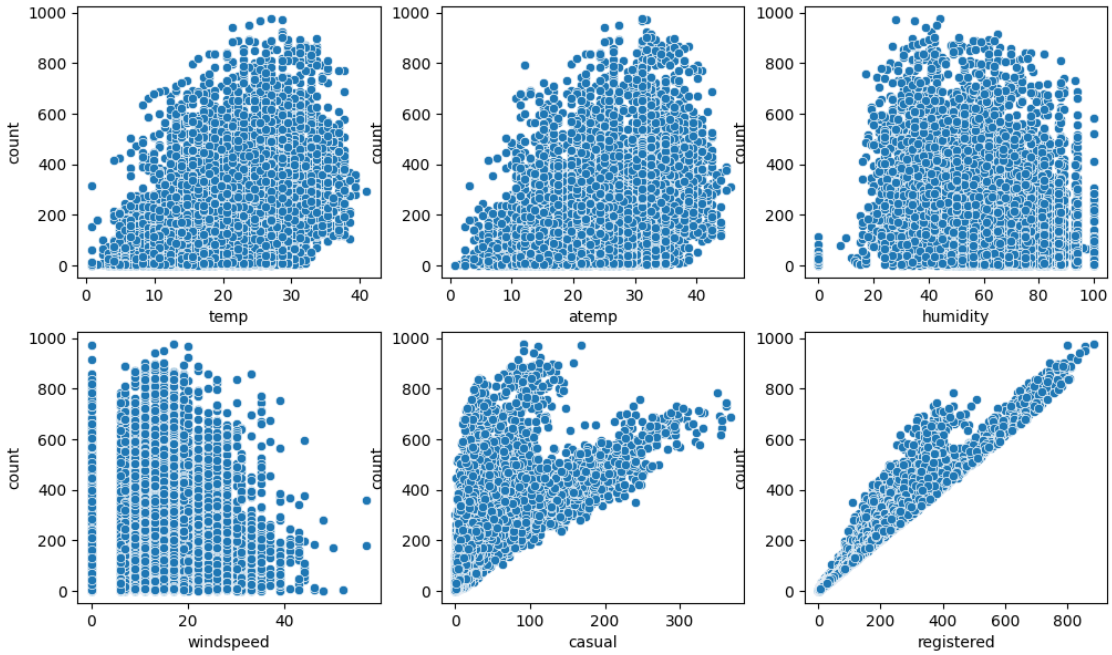
[23]:
```
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(10, 8))
index = 0
for row in range(2):
  for col in range(2):
    sns.boxplot(data=df, x=cat_cols[index], y='count', ax=axis[row,col])
    index += 1
plt.show()
```



- 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.

7.2 Plotting numerical variables againt count using scatterplot.

[24]:
```
fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(12, 7))
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()
```

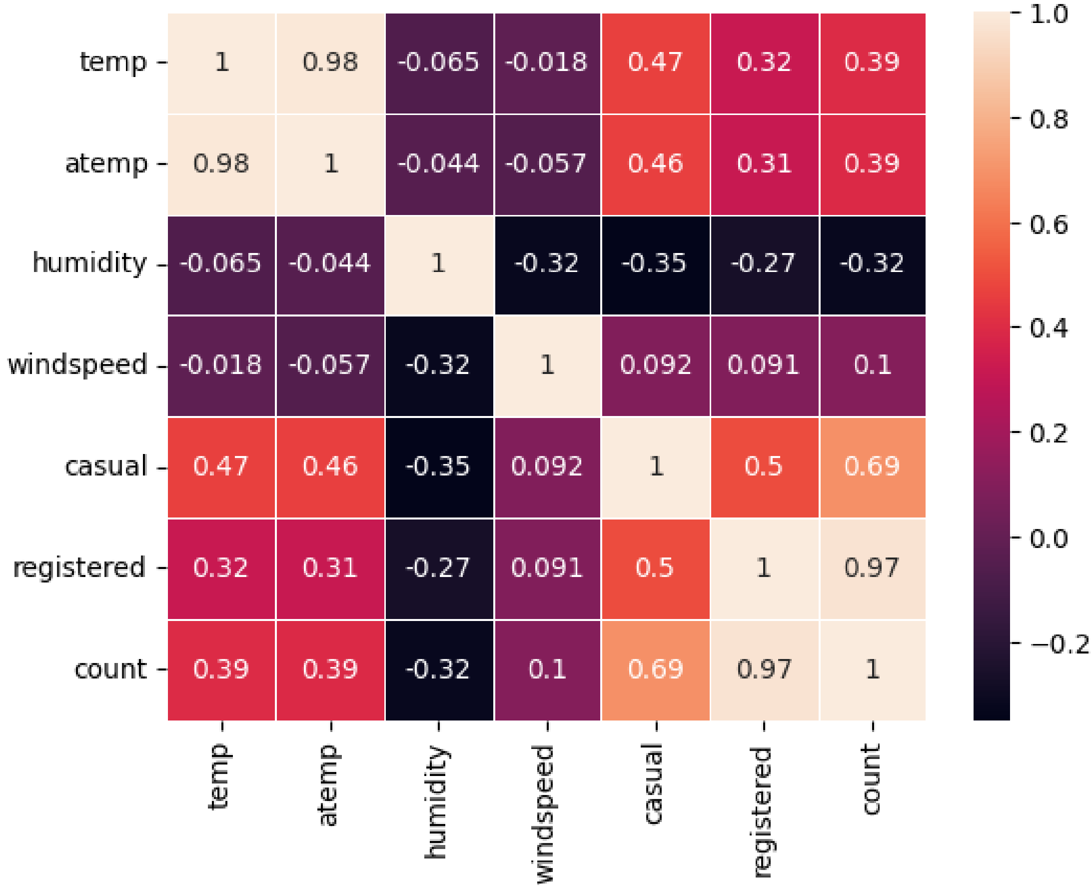


- Whenever the humidity is less than 20, number of bikes rented is very very low.
- Whenever the temperature is less than 10, number of bikes rented is less.
- Whenever the windspeed is greater than 35, number of bikes rented is less.

8 Understanding the correlation between count and numerical variables.

[25]:
```
#df.corr()['count']
sns.heatmap(df.corr(), annot=True, linewidth=.5)
plt.show()
```

```
<ipython-input-25-df3082b8665f>: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, linewidth=.5)
```



9 Hypothesis testing

10 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 dependent on the season
- Significance level (alpha): 0.05

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

val = stats.chi2_contingency(data_table)
print(val)
print()
print("**************************************************")
```

```python
Expected_values = val[3]
print(f'Expected_values : {val[3]}')
print()
print("***********************************************")


nrows, ncols = 4, 4
dof = (nrows-1)*(ncols-1)
print(f"Degrees of freedom: {dof}")
print()
print("***********************************************")
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(f"Chi-square test statistic: {chi_sqr_statistic}")
print()
print("***********************************************")


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

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


if p_val <= alpha:
  print("Since p-value is less than the alpha 0.05 we reject Null Hypothesis.␣
 ↪This indicates 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")
```

```
Observed values:
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]]))

***********************************************
```

```
Expected_values : [[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

***********************************************
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 Null Hypothesis. This
indicates weather is dependent on the season.
```

10.1 2- Sample T-Test to check if Working Day has an effect on the number of electric cycles rented :

- 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

[27]:
```
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
```

[27]:
```
1.0
```

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.

[28]:
```
stats.ttest_ind(a=data_group1, b=data_group2, equal_var=True)
```

[28]:
```
TtestResult(statistic=-1.2096277376026694, pvalue=0.22644804226361348,
df=10884.0)
```

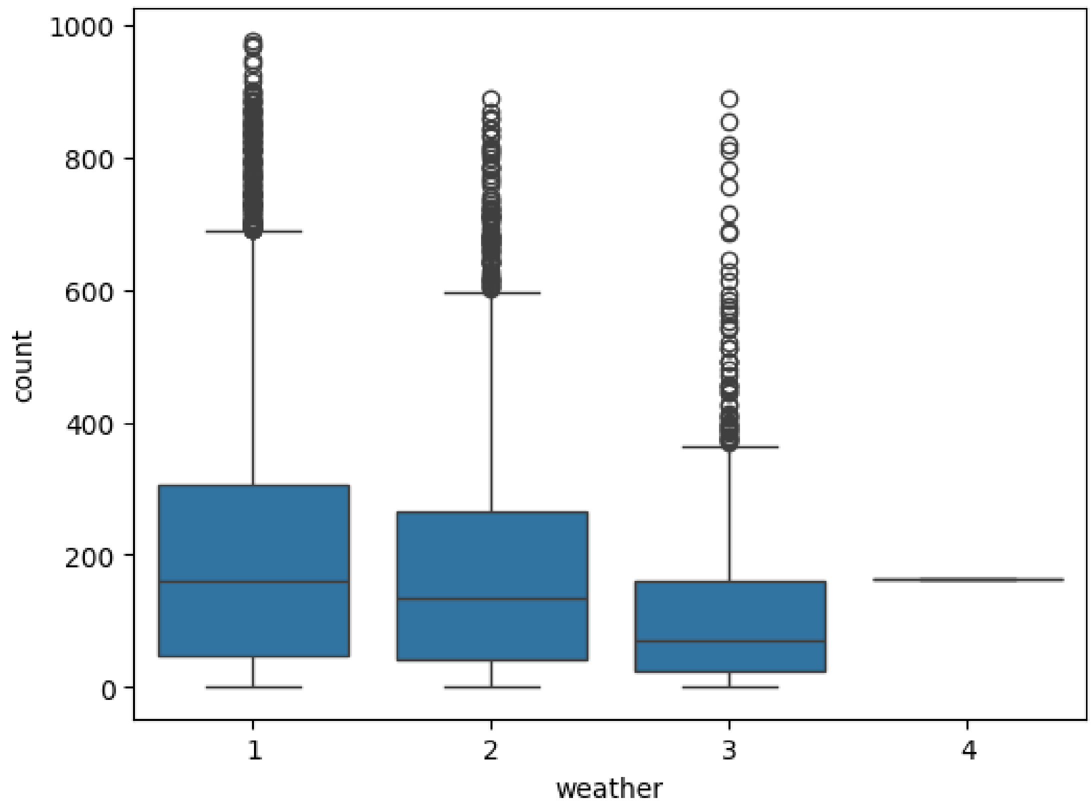
Since pvalue is greater than 0.05 so we cannot reject the Null hypothesis. We don't have the sufficient evidence to say that working day has effect on the number of cycles being rented.

10.2 ANNOVA to check if No. of cycles rented is similar or different in different weather and season.

- 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

10.2.1 Weather check

[29]:
```
df["weather"].unique()

df["weather"].value_counts()

sns.boxplot(x='weather', y='count', data=df)
plt.show()

count_g1 = df[df["weather"]==1]["count"]
count_g2 = df[df["weather"]==2]["count"]
count_g3 = df[df["weather"]==3]["count"]
count_g4 = df[df["weather"]==4]["count"]

a,b,c,d = [round(count_g1.mean(), 2),round(count_g2.mean(),2),round(count_g3.
 ↪mean(),2),round(count_g4.mean(),2)]

print(a, end= " ")
print(b, end= " ")
print(c, end= " ")
print(d)
```



205.24 178.96 118.85 164.0

[30]:
```
from scipy.stats import f_oneway, kruskal    # Numeric Vs categorical for many␣
↪categories

# H0: All weather's have same number of cycles rented.
# Ha: Atleast one or more weather conditions have different number of cycles␣
↪rented.

f_stats, p_value = f_oneway(count_g1,count_g2,count_g3,count_g4)
print(f"p_value : {p_value}")
print()

if p_value < 0.05:
    print("Reject H0")
    print("Different weathers have different number of cycles rented")
else:
    print("Fail to reject H0 or accept H0")
    print("All weather's have same number of cycles rented.")
```

p_value : 5.482069475935669e-42

```
Reject H0
Different weathers have different number of cycles rented
```

Since P-value is very less we reject the null hypothesis. Atleast one or more weather conditions have different number of cycles rented.

10.2.2 Season check

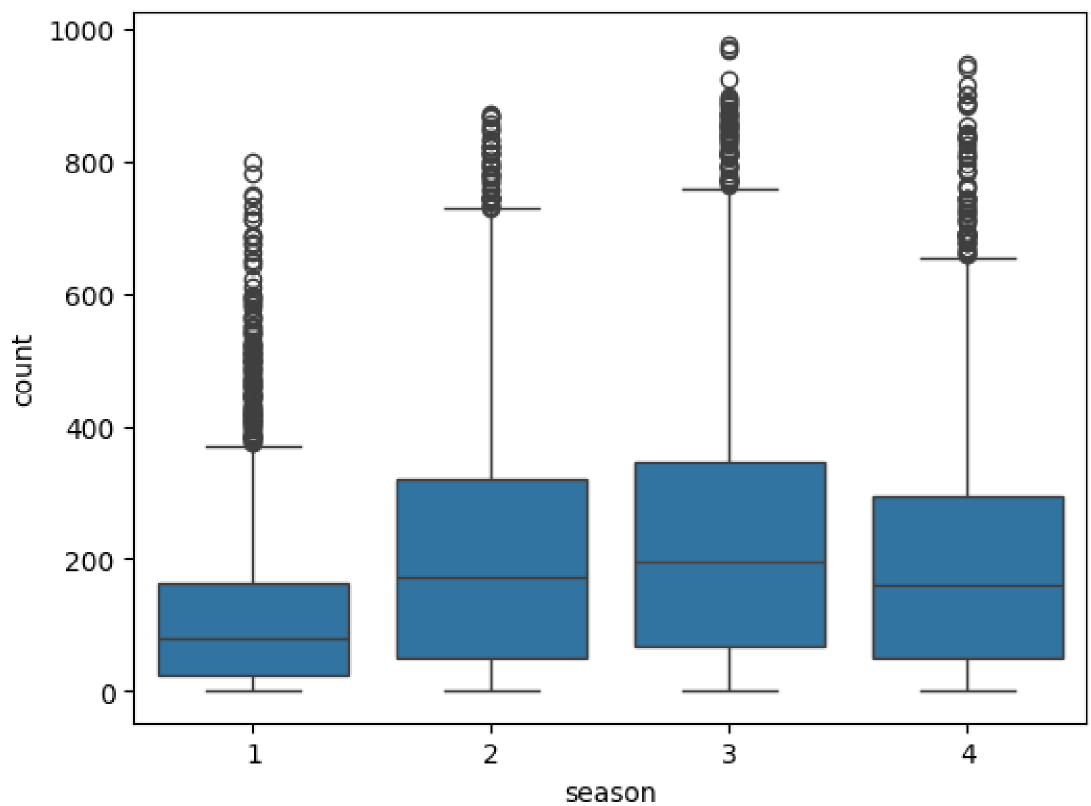
[31]:
```python
df["season"].unique()
```

[31]:
```
array([1, 2, 3, 4], dtype=object)
```

[32]:
```python
df["season"].value_counts()
```

[32]:
```
4    2734
2    2733
3    2733
1    2686
Name: season, dtype: int64
```

[33]:
```python
sns.boxplot(x='season', y='count', data=df)
plt.show()
```



[34]:
```
coun_g1 = df[df["season"]==1]["count"]
coun_g2 = df[df["season"]==2]["count"]
coun_g3 = df[df["season"]==3]["count"]
coun_g4 = df[df["season"]==4]["count"]

a,b,c,d = [round(coun_g1.mean(), 2),round(coun_g2.mean(),2),round(coun_g3.
↪mean(),2),round(coun_g4.mean(),2)]

print(a, end= " ")
print(b, end= " ")
print(c, end= " ")
print(d)
```

```
116.34 215.25 234.42 198.99
```

[35]:
```
from scipy.stats import f_oneway, kruskal   # Numeric Vs categorical for many␣
↪categories

# H0: All seasons's have same number of cycles rented.
# Ha: Atleast one or more seasons  have different number of cycles rented.

f_stats, p_value = f_oneway(coun_g1,coun_g2,coun_g3,coun_g4)
print(f"p_value : {p_value}")
print()

if p_value < 0.05:
    print("Reject H0")
    print("Different seasons have different number of cycles rented")
else:
    print("Fail to reject H0 or accept H0")
    print("All seasons have same number of cycles rented.")
```

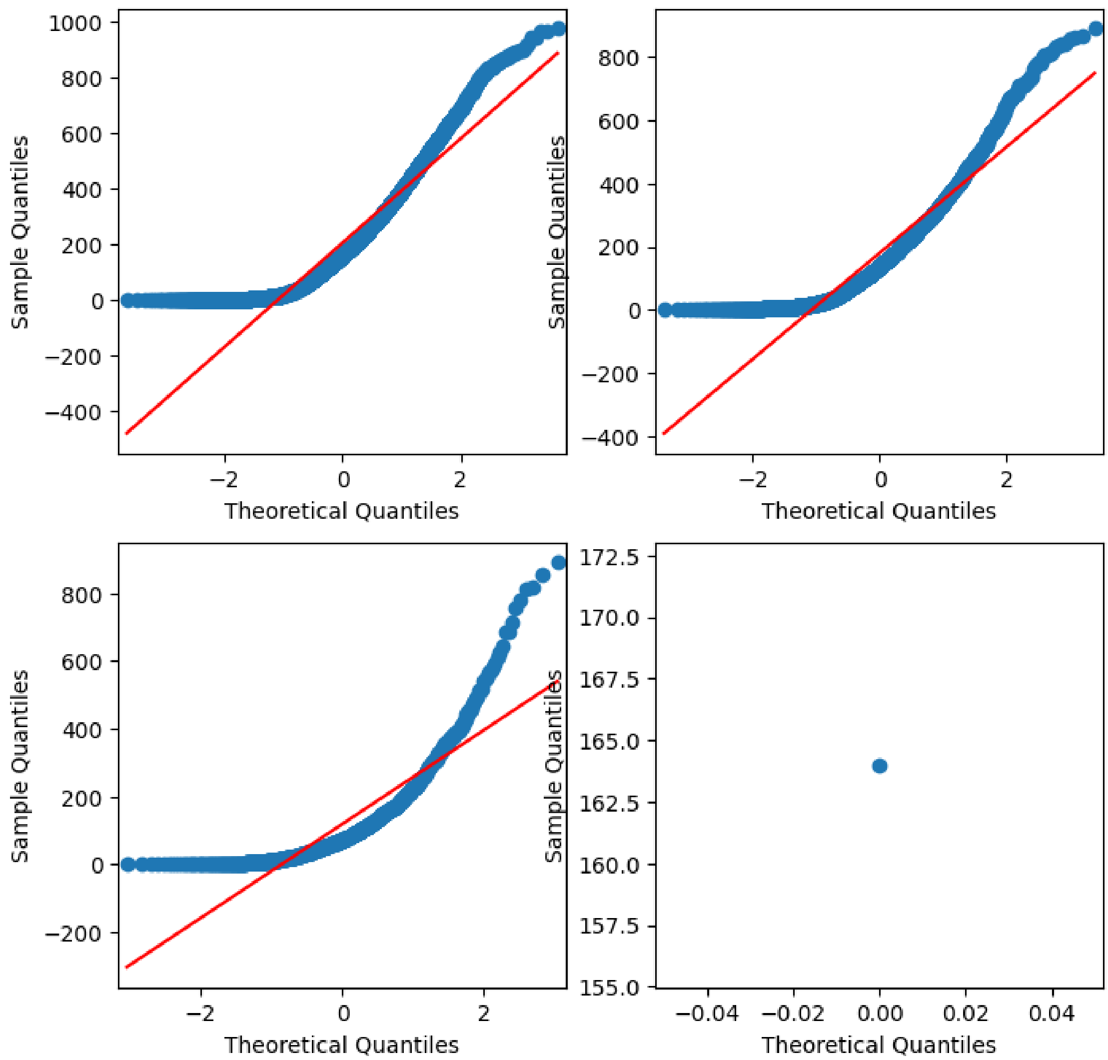
```
p_value : 6.164843386499654e-149

Reject H0
Different seasons have different number of cycles rented
```

10.3 Checking Assumptions of Anova test

10.3.1 QQ plot and histogram for weather

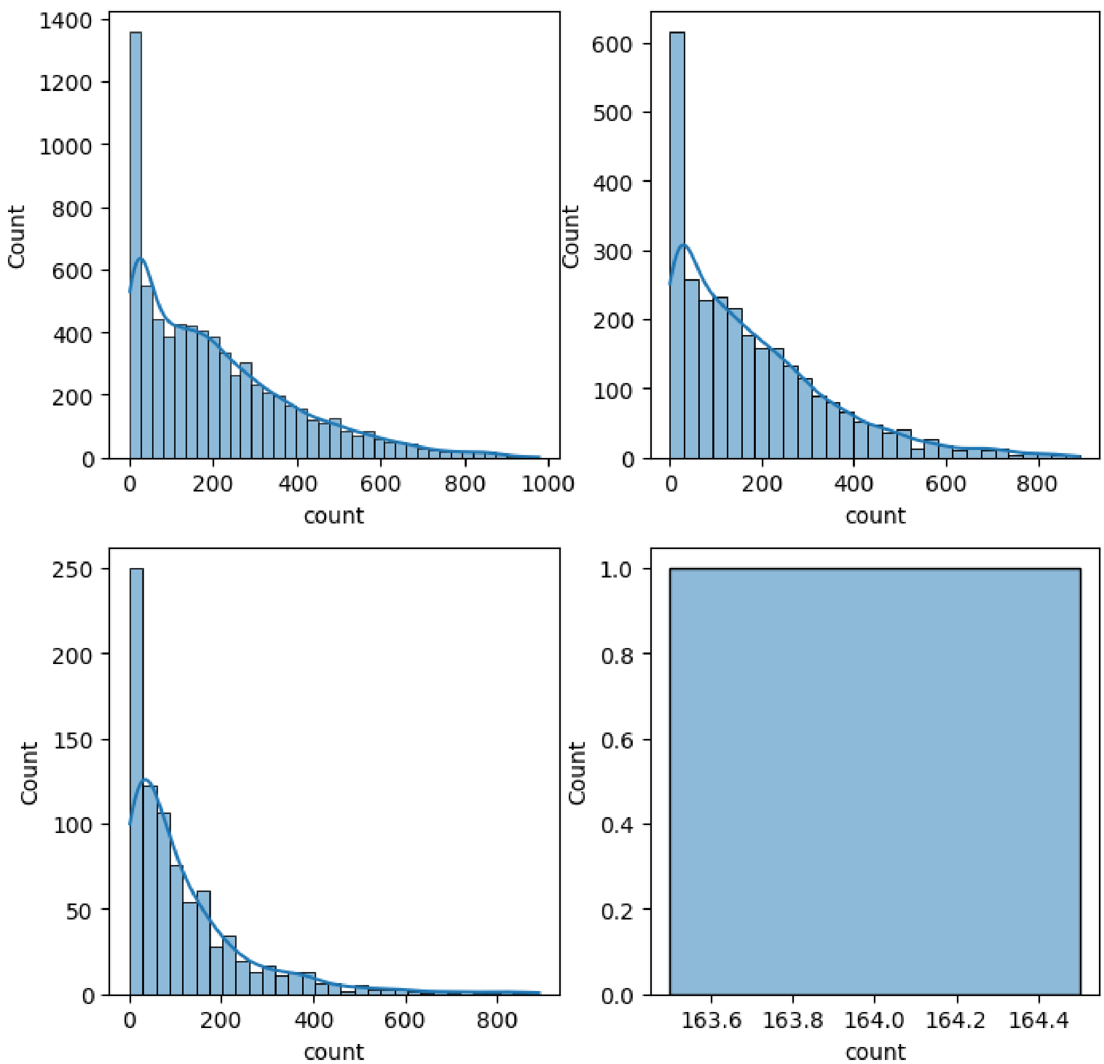
[36]:
```
import numpy as np
import statsmodels.api as sm
import pylab as py
```

```
a = [count_g1,count_g2,count_g3,count_g4]

fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(8, 8))

sm.qqplot(a[0], line = "s", ax = axis[0,0])
sm.qqplot(a[1], line = "s", ax = axis[0,1])
sm.qqplot(a[2], line = "s", ax = axis[1,0])
sm.qqplot(a[3], line = "s", ax = axis[1,1])

plt.show()
```



[37]:
```
a = [count_g1,count_g2,count_g3,count_g4]

fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(8, 8))
```

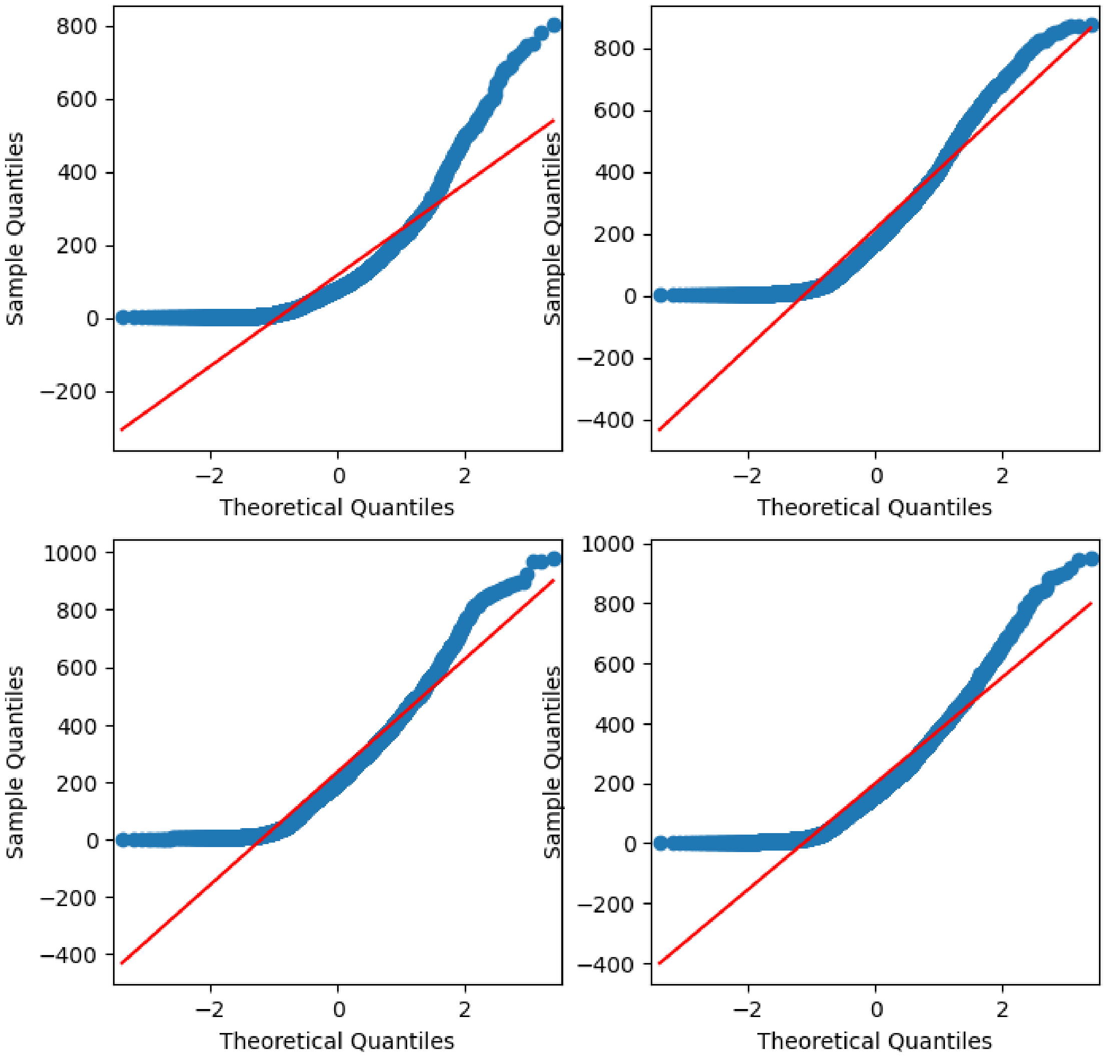
```
index = 0
for row in range(2):
  for col in range(2):
    sns.histplot(a[index], ax=axis[row, col], kde=True)
    index += 1
plt.show()
```



10.3.2 QQ plot and histogram for season.

[38]:
```
b = [coun_g1,coun_g2,coun_g3,coun_g4]

fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(8, 8))
```

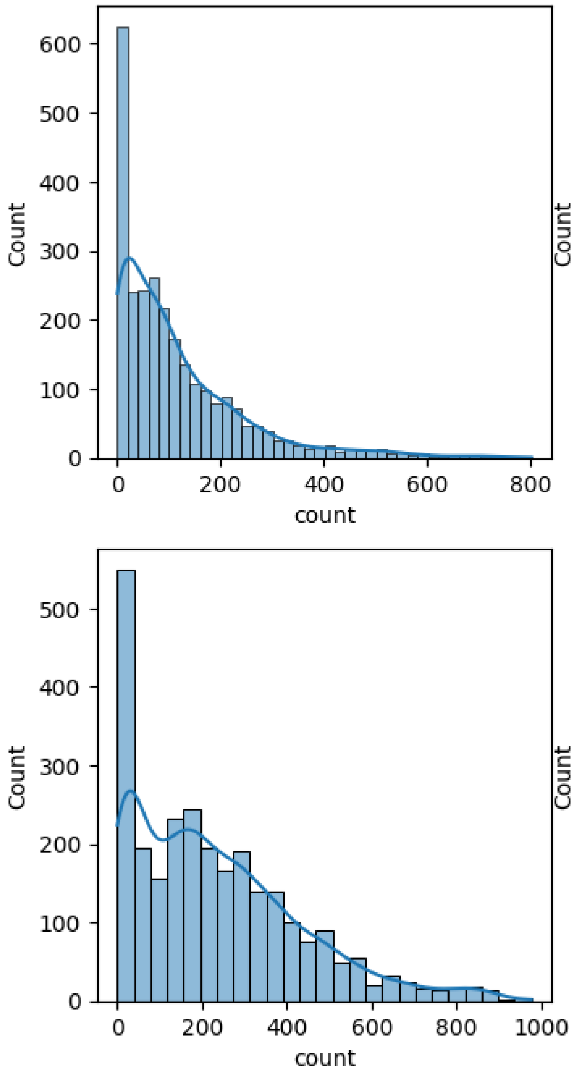
```
sm.qqplot(b[0], line = "s", ax = axis[0,0])
sm.qqplot(b[1], line = "s", ax = axis[0,1])
sm.qqplot(b[2], line = "s", ax = axis[1,0])
sm.qqplot(b[3], line = "s", ax = axis[1,1])

plt.show()
```



[39]:

```
b = [coun_g1,coun_g2,coun_g3,coun_g4]

fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(8, 8))

index = 0
for row in range(2):
  for col in range(2):
```

```
    sns.histplot(b[index], ax=axis[row, col], kde=True)
    index += 1
plt.show()
```



10.3.3 The above plots show data is not gaussian. Let us confirm the same via statiscal test.

10.3.4 Shapiro-Wilk test for Gaussian (Statistical Test for Normality)

Weather data

[40]:
```
count_g1_subset = count_g1.sample(100)

# H0: Data is Gaussian
# Ha: Data is not Gaussian
```

```
from scipy.stats import shapiro
from scipy.stats import levene

test_stat, p_value = shapiro(count_g1_subset)
print(p_value)
if p_value<0.05:
    print("Data is not gaussian")
else:
    print("Data is gaussian")
```

```
7.773400767518979e-08
Data is not gaussian
```

Season data

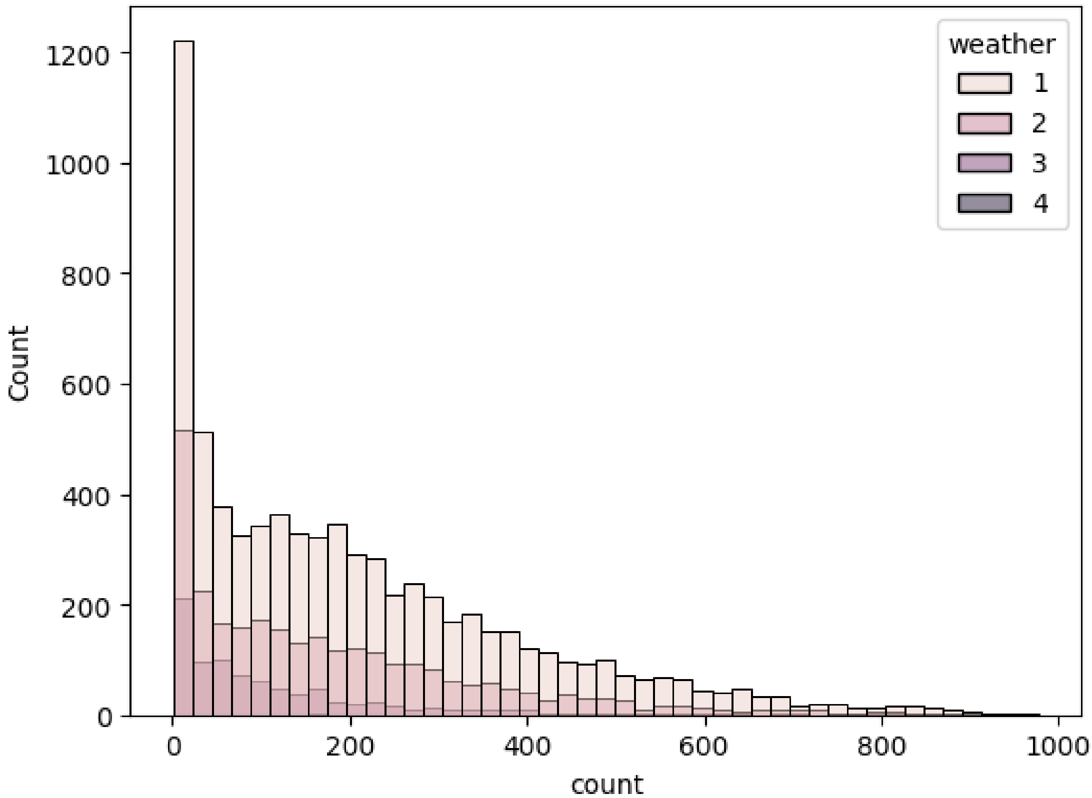
[41]:
```
coun_g1_subset = coun_g1.sample(100)

# H0: Data is Gaussian
# Ha: Data is not Gaussian
from scipy.stats import shapiro
from scipy.stats import levene

test_stat, p_value = shapiro(coun_g1_subset)
print(p_value)
if p_value<0.05:
    print("Data is not gaussian")
else:
    print("Data is gaussian")
```

```
1.2687553785362127e-10
Data is not gaussian
```

10.3.5 *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

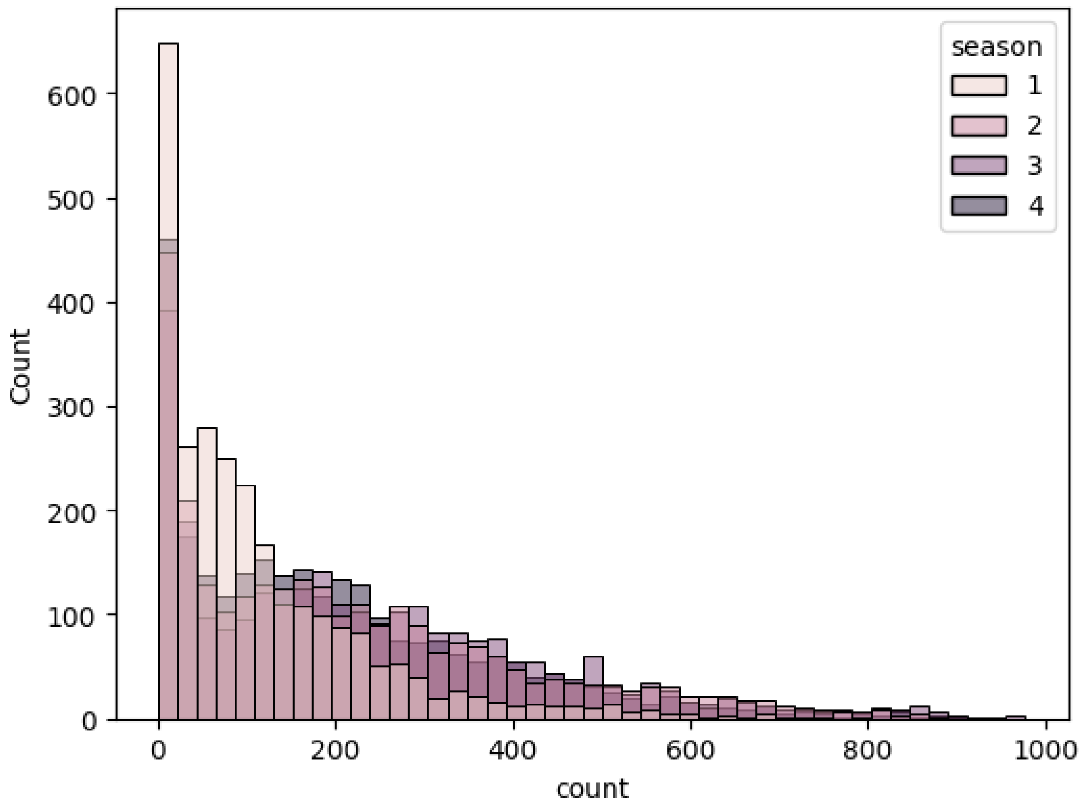
[42]:
```
sns.histplot(data= df, x="count", hue= "weather", color = "o")
plt.show()
```



[43]:
```
# H0: Variances are equal
# Ha: Variances are not equal
levene_stat, p_value = levene(count_g1,count_g2,count_g3,count_g4)
print(f'p-value : {p_value}')
if p_value < 0.05:
    print("Reject the null hypthesis.Variances are not similar.")
else:
    print("Variance are similar.")
```

```
p-value : 3.504937946833238e-35
Reject the null hypthesis.Variances are not similar.
```

[44]:
```
sns.histplot(data= df, x="count", hue= "season", color = "o")
plt.show()
```



[45]:
```
# H0: Variances are equal
# Ha: Variances are not equal
levene_stat, p_value = levene(coun_g1,coun_g2,coun_g3,coun_g4)
print(f'p-value : {p_value}')
if p_value < 0.05:
    print("Reject the null hypthesis. Variances are not similar.")
else:
    print("Variance are similar.")
```

```
p-value : 1.0147116860043298e-118
Reject the null hypthesis. Variances are not similar.
```

11 As per the QQ plots, histograms, Shapiro and Levene test the assumtions of Anova have failed. Hence we will use Kruskal test.

11.1 Weather

[46]:
```
kruskal_stat, p_value = stats.kruskal(count_g1,count_g2,count_g3,count_g4)
print(f"p_value : {p_value}")

if p_value<0.05:
  print("Since p-value is less than 0.05, we reject the null hypothesis")
  print('Different weather have different number of cycles rented.')
else :
  print("Failes to reject null hypothesis. All weathers has same number of␣
 ↪cycles rented.")
```

```
p_value : 3.501611300708679e-44
Since p-value is less than 0.05, we reject the null hypothesis
Different weather have different number of cycles rented.
```

11.2 Season

[47]:
```
kruskal_stat, p_value = stats.kruskal(coun_g1,coun_g2,coun_g3,coun_g4)
print(f"p_value : {p_value}")

if p_value<0.05:
  print("Since p-value is less than 0.05, we reject the null hypothesis")
  print('Different weather have different number of cycles rented.')
else :
  print("Failed to reject null hypothesis. All weathers has same number of␣
 ↪cycles rented.")
```

```
p_value : 2.479008372608633e-151
Since p-value is less than 0.05, we reject the null hypothesis
Different weather have different number of cycles rented.
```

12 Insights

- In summer and fall seasons more bikes are rented as compared to other seasons.
- It is seen there is increase in bike rentals on holidays.
- 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.

- 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.
- An ANOVA test on different seasons with respect to count,implies that population count means under different seasons are not the same, meaning there is a difference in the usage of Yulu bikes in different seasons.
- By performing an ANOVA test on different weather conditions except 4 with respect to count, we can infer that population count means under different weather conditions are the same, meaning there is a difference in the usage of Yulu bikes in different weather conditions.
- By performing a Chi2 test on season and weather (categorical variables), we can infer that there is an impact on weather dependent on season.
- The maximum number of holidays can be seen during the fall and winter seasons.
- There is a positive corelation between counts and temperature.
- There is a negative corelation between counts and humidity.

13 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 temperature 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.
- Consistent monitoring of seasonal weather forecast would help Yulu to be prepared for nature related decline in rented bikes due to rains, humidity,etc.
- As casual users are very less Yulu should focus on marketing startegy to bring more customers. for eg. first time user discount, friends and family discounts, referral bonuses etc.
- On non working days as count is low. We would recommend certain promotional campaigns to attracts uses on these days.
- In heavy rains as rent count is very low Yulu can introduce a different vehicle such as car or umbrella attached bike to encourage more users.