yulu_hypothesis_testing

April 4, 2025

[2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[1]:
```
from google.colab import files
df=files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving bike_sharing.txt to bike_sharing.txt
```

[3]:
```
df=pd.read_csv("bike_sharing.txt")
df.head()
```

[3]:

	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	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

[ ]:
```
df.tail()
```

[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
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						

```
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
```

[ ]:
```
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
```

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

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

[ ]:
```
df.describe(include='all')
```

[ ]:
```
                     datetime        season       holiday    workingday  \
count                   10886  10886.000000  10886.000000  10886.000000
unique                  10886           NaN           NaN           NaN
top     2012-12-19 23:00:00           NaN           NaN           NaN
freq                        1           NaN           NaN           NaN
mean                      NaN      2.506614      0.028569      0.680875
std                       NaN      1.116174      0.166599      0.466159
min                       NaN      1.000000      0.000000      0.000000
25%                       NaN      2.000000      0.000000      0.000000
50%                       NaN      3.000000      0.000000      1.000000
75%                       NaN      4.000000      0.000000      1.000000
max                       NaN      4.000000      1.000000      1.000000

              weather         temp         atemp      humidity     windspeed  \
```

count	10886.000000	10886.00000	10886.000000	10886.000000	10886.000000
unique	NaN	NaN	NaN	NaN	NaN
top	NaN	NaN	NaN	NaN	NaN
freq	NaN	NaN	NaN	NaN	NaN
mean	1.418427	20.23086	23.655084	61.886460	12.799395
std	0.633839	7.79159	8.474601	19.245033	8.164537
min	1.000000	0.82000	0.760000	0.000000	0.000000
25%	1.000000	13.94000	16.665000	47.000000	7.001500
50%	1.000000	20.50000	24.240000	62.000000	12.998000
75%	2.000000	26.24000	31.060000	77.000000	16.997900
max	4.000000	41.00000	45.455000	100.000000	56.996900

	casual	registered	count
count	10886.000000	10886.000000	10886.000000
unique	NaN	NaN	NaN
top	NaN	NaN	NaN
freq	NaN	NaN	NaN
mean	36.021955	155.552177	191.574132
std	49.960477	151.039033	181.144454
min	0.000000	0.000000	1.000000
25%	4.000000	36.000000	42.000000
50%	17.000000	118.000000	145.000000
75%	49.000000	222.000000	284.000000
max	367.000000	886.000000	977.000000

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

[ ]:

	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
```

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

[ ]:
```
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
```

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

[ ]:
```
np.int64(0)
```

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

[ ]:
```
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
```

[4]:
```
cat_col=['season','holiday','workingday','weather']
df[cat_col]=df[cat_col].astype('category')
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  category
 2   holiday     10886 non-null  category
 3   workingday  10886 non-null  category
 4   weather     10886 non-null  category
 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: category(4), float64(3), int64(4), object(1)
memory usage: 723.7+ KB
```

[5]:
```
date_time=pd.to_datetime(df['datetime'])
date_time.dtype
```

[5]:
```
dtype('<M8[ns]')
```

[6]:
```
#replacing number with category
df['season']=df['season'].map({1:'spring',2:'summer',3:"fall",4:"winter"})
df['holiday']=df['holiday'].map({0:'no',1:'yes'})
df['workingday']=df['workingday'].map({0:'no',1:"yes"})
df['weather']=df['weather'].map({1:'clear',2:"cloudy",3:"light_rain",4:
 ↪"heavy_rain"})
df.head()
```

[6]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	spring	no	no	clear	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	spring	no	no	clear	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	spring	no	no	clear	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	spring	no	no	clear	9.84	14.395					
4	2011-01-01 04:00:00	spring	no	no	clear	9.84	14.395					

```
3        75        0.0       3          10      13
4        75        0.0       0           1       1
```

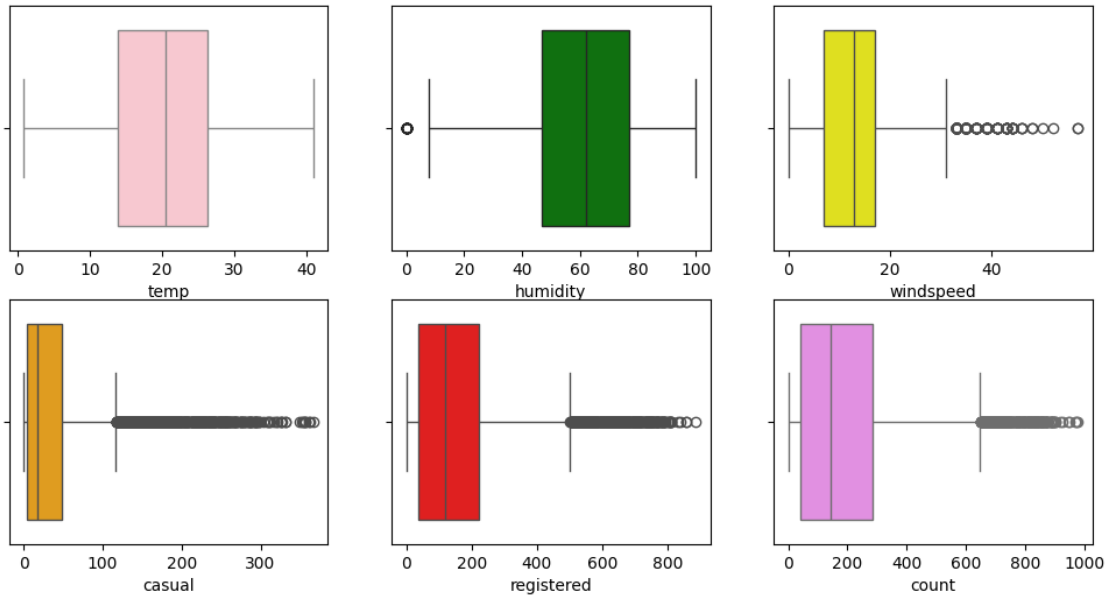
[7]:
```
df.describe()
```

[7]:
```
              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
```

[8]:
```
df.describe(include='category')
```

[8]:
```
        season holiday workingday weather
count    10886   10886      10886   10886
unique       4       2          2       4
top     winter      no        yes   clear
freq      2734   10575       7412    7192
```

[9]:
```
#detecting outliers
fig,axis=plt.subplots(2,3,figsize=(12,6))
sns.boxplot(data=df,x=df['temp'],ax=axis[0,0],color='Pink')
sns.boxplot(data=df,x=df['humidity'],ax=axis[0,1],color='green')
sns.boxplot(x=df['windspeed'],data=df,ax=axis[0,2],color='yellow')
sns.boxplot(x=df['casual'],ax=axis[1,0],color='orange')
sns.boxplot(x=df['registered'],data=df,ax=axis[1,1],color='red')
sns.boxplot(x=df['count'],data=df,ax=axis[1,2],color='violet')
plt.show()
```



[23]:
```
#removing outliers
continuous_var=['temp','humidity','windspeed','casual','registered','count']
df_filtered=df.copy()
for var in continuous_var:
  q1=df[var].quantile(0.25)
  q3=df[var].quantile(0.75)
  iqr=q3-q1
  lower_bound=q1-(1.5*iqr)
  upper_bound=q3+(1.5*iqr)
  df_filtered=df_filtered[(df_filtered[var]>=lower_bound)&␣
 ↪(df_filtered[var]<=upper_bound)]
df=df_filtered
```

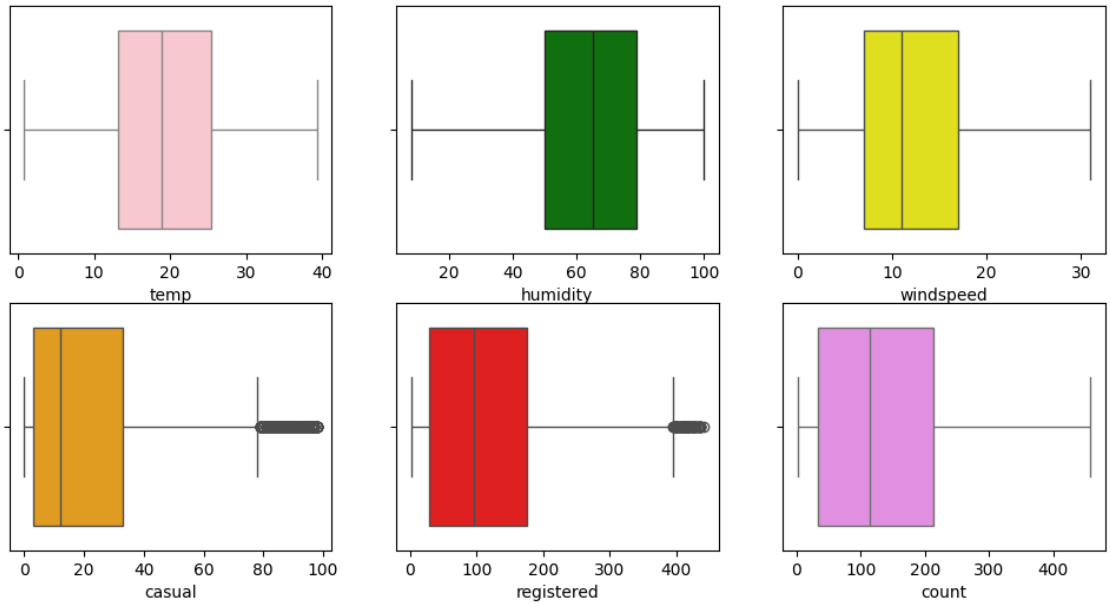
[28]:
```
df.describe()
```

[28]:

	temp	atemp	humidity	windspeed	casual	registered	count
count	8945.000000	8945.000000	8945.000000	8945.000000	8945.000000	8945.000000	8945.000000
mean	19.376110	22.767514	64.233203	12.047171	21.527781		
std	7.582856	8.264151	18.611223	7.432740	23.682250		
min	0.820000	1.515000	8.000000	0.000000	0.000000		
25%	13.120000	15.910000	50.000000	7.001500	3.000000		
50%	18.860000	22.725000	65.000000	11.001400	12.000000		
75%	25.420000	29.545000	79.000000	16.997900	33.000000		
max	39.360000	45.455000	100.000000	31.000900	98.000000		

```
mean    115.094243   136.622023
std      99.262201   114.542722
min       1.000000     2.000000
25%      28.000000    33.000000
50%      95.000000   114.000000
75%     175.000000   214.000000
max     442.000000   458.000000
```

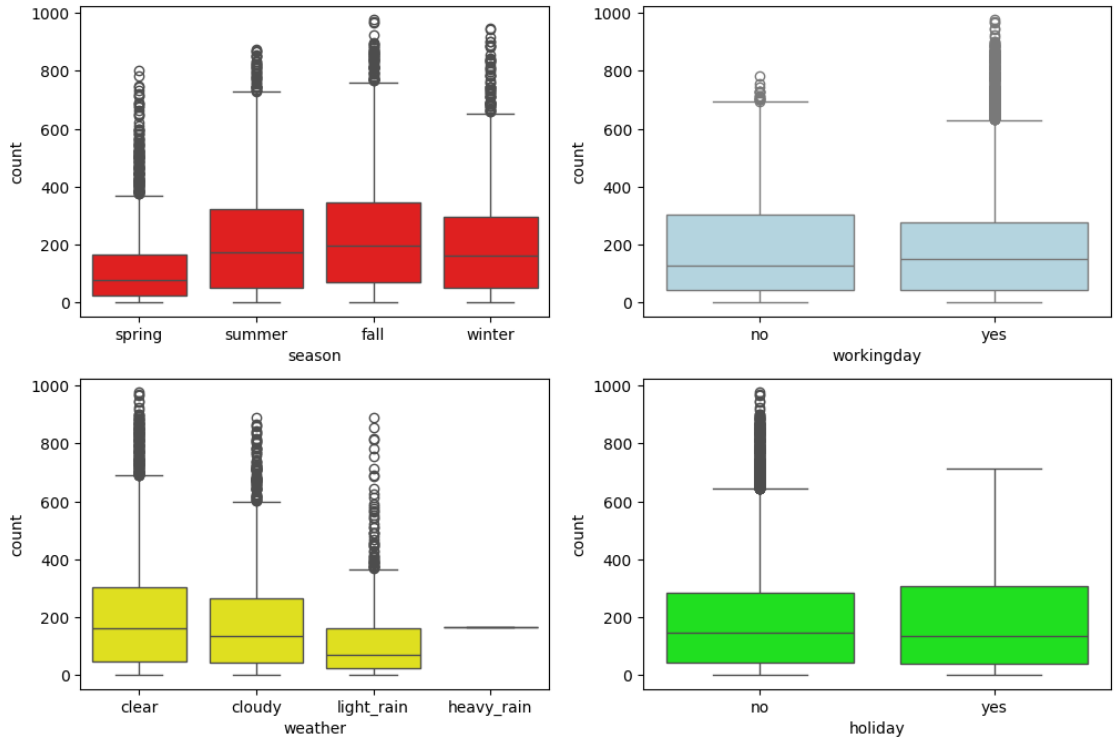
[30]:
```
fig,axis=plt.subplots(2,3,figsize=(12,6))
sns.boxplot(data=df,x=df['temp'],ax=axis[0,0],color='Pink')
sns.boxplot(data=df,x=df['humidity'],ax=axis[0,1],color='green')
sns.boxplot(x=df['windspeed'],data=df,ax=axis[0,2],color='yellow')
sns.boxplot(x=df['casual'],ax=axis[1,0],color='orange')
sns.boxplot(x=df['registered'],data=df,ax=axis[1,1],color='red')
sns.boxplot(x=df['count'],data=df,ax=axis[1,2],color='violet')
plt.show()
```



removed the outliers where the variables foloow normal distribution but in casual and registered outliers are not removed because they are skewed distribution.

[ ]:
```
plt.figure(figsize=(12,8))
plt.subplot(2,2,1)
sns.boxplot(x='season',y='count',data=df,color="red")
plt.subplot(2,2,2)
sns.boxplot(x="workingday",y='count',data=df,color="lightblue")
plt.subplot(2,2,3)
sns.boxplot(x='weather',y='count',data=df,color="yellow")
```

```
plt.subplot(2,2,4)
sns.boxplot(x='holiday',y='count',data=df,color="lime")
plt.show()
```



Insights:

No outliers in temperature and humidity.

and in other numerical columns outliers are present.

In season, spring and winter have more unusual values compared to other seasons.

In weather category, light_rain has unexpected patterns while heavy rain doesnt have any pattern.

Comparing workind day and holiday, working day have typically more unexpectional values than holidays. This need to be noticed.

Distribution of categorical and numerical columns

[ ]:
```
holiday=(df['holiday'].value_counts(normalize = True) * 100)
holiday
```

[ ]:
```
holiday
no     97.14312
yes     2.85688
Name: proportion, dtype: float64
```

[ ]:
```
workingday=(df['workingday'].value_counts(normalize = True) * 100)
workingday
```

[ ]:
```
workingday
yes    68.087452
no     31.912548
Name: proportion, dtype: float64
```

[ ]:
```
weather=(df['weather'].value_counts(normalize = True) * 100)
weather
```

[ ]:
```
weather
clear         66.066507
cloudy        26.033437
light_rain     7.890869
heavy_rain     0.009186
Name: proportion, dtype: float64
```

Insights

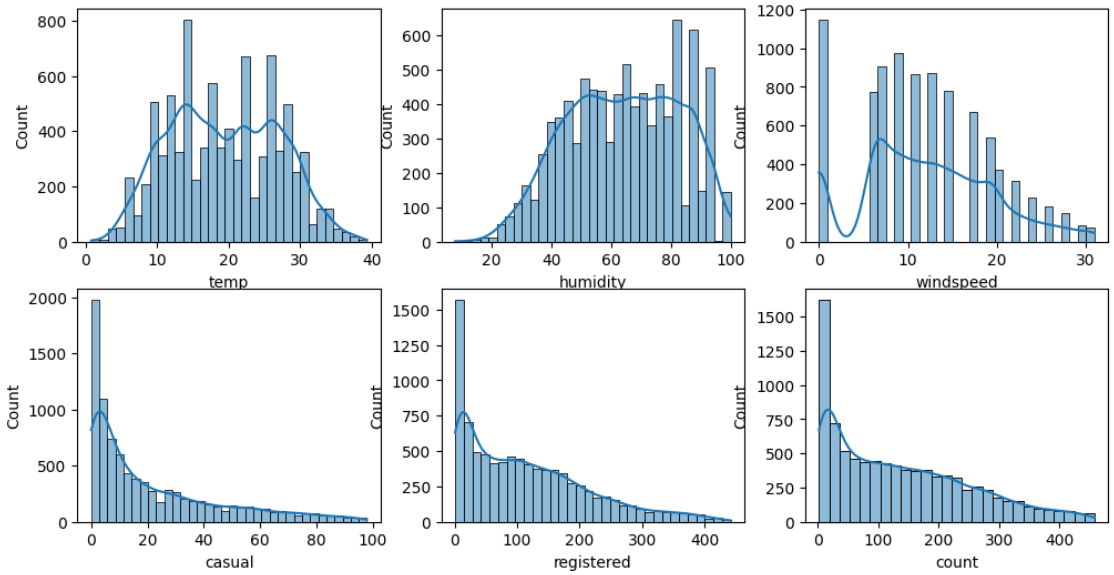
All four seasons carry similar percentage, means season doesnot impact in elictric bike usage.

During holidays people not prefer to use bikes.

But in working days over 68% perfered using bikes.

When sky is clear people are interested to ride bikes but during heavy rain only 0.009 % are used bikes.

[34]:
```
#for numerical columns

fig,axis=plt.subplots(2,3,figsize=(12,6))
sns.histplot(df['temp'],ax=axis[0,0],kde=True)
sns.histplot(df['humidity'],ax=axis[0,1],kde=True)
sns.histplot(df['windspeed'],ax=axis[0,2],kde=True)
sns.histplot(df['casual'],ax=axis[1,0],kde=True)
sns.histplot(df['registered'],ax=axis[1,1],kde=True)
sns.histplot(df['count'],ax=axis[1,2],kde=True)
plt.show()
```



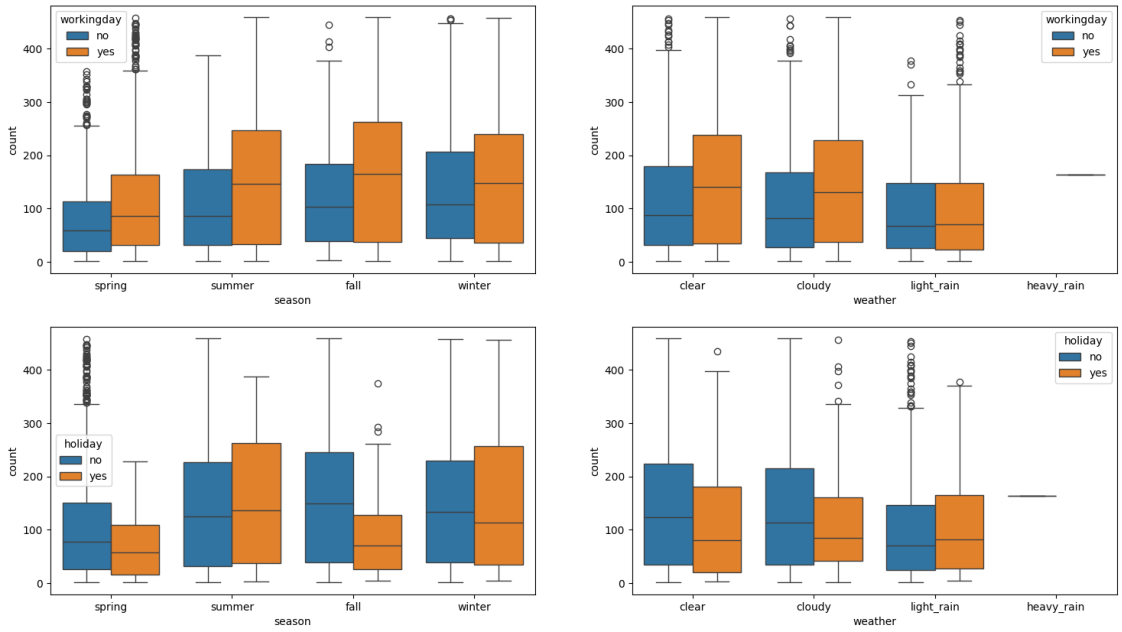
Insights

Temperature and humidity closely related to normal distribution

Windspeed represents binomial distribution

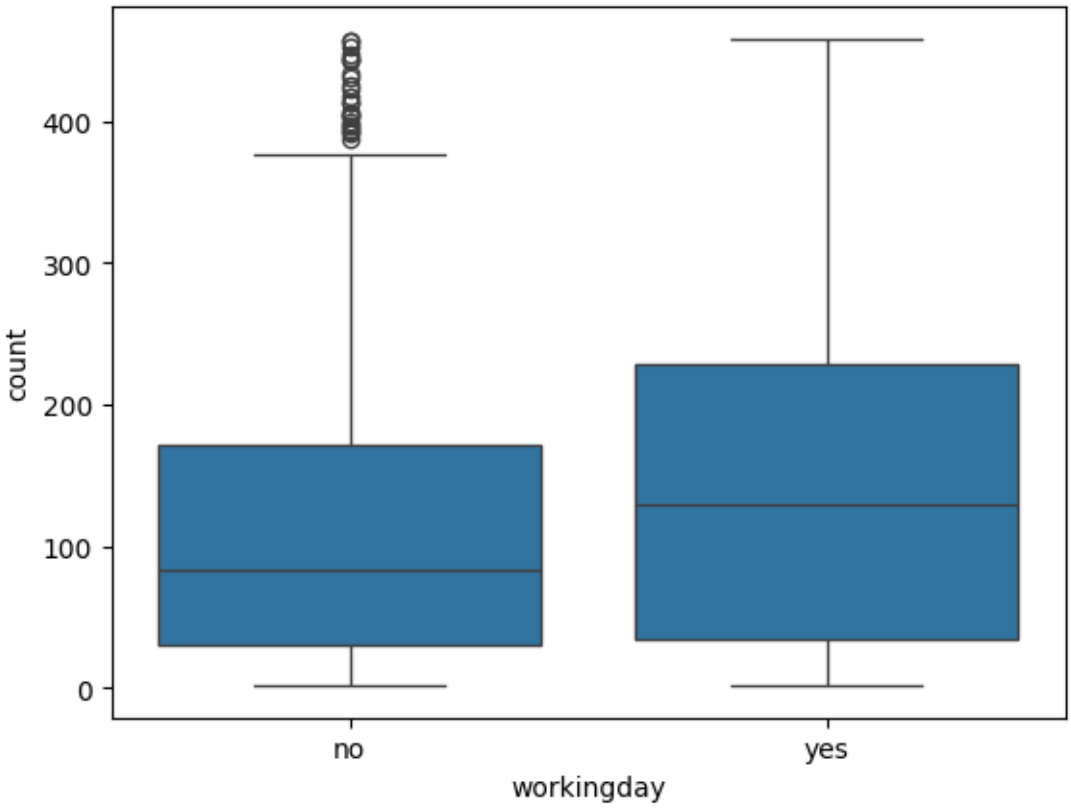
casual, registered and count shows log normal or exponential distribution

[35]:
```
#bivariate analysis
fig,axis=plt.subplots(figsize=(18,10),nrows=2,ncols=2)
sns.boxplot(x='season',y='count',hue='workingday',data=df,ax=axis[0,0])
sns.boxplot(x='weather',y='count',hue='workingday',data=df,ax=axis[0,1])
sns.boxplot(x='season',y='count',hue='holiday',data=df,ax=axis[1,0])
sns.boxplot(x='weather',y='count',hue='holiday',data=df,ax=axis[1,1])
plt.show()
```



[41]:
```
#hypothesis testing
#check whether Working Day has effect on number of electric cycles rented

df.groupby('workingday')['count'].describe()

sns.boxplot(x=df['workingday'],y=df['count'])
plt.show()
```

```
<ipython-input-41-ea775ba5121b>:7: FutureWarning: The default of observed=False
is deprecated and will be changed to True in a future version of pandas. Pass
observed=False to retain current behavior or observed=True to adopt the future
default and silence this warning.
  df.groupby('workingday')['count'].describe()
```



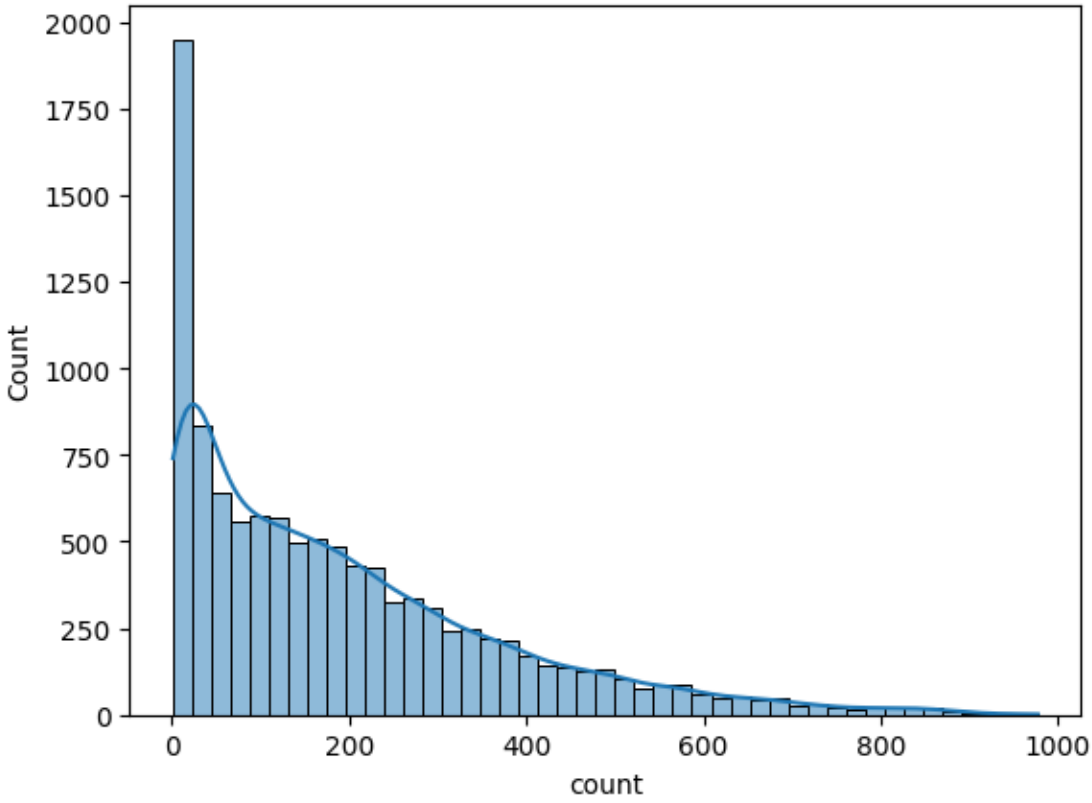
[44]:
```
#null hpothesis: working day has no effect on number of cycles rented
#alternate hypothesis: working day has an effect on cycles  rented
#alpha =0.05

from scipy.stats import ttest_ind
t_stat,p_val=ttest_ind(df[df['workingday']=='yes']['count'].
 ↪values,df[df['workingday']=='no']['count'].values)
p_val
```

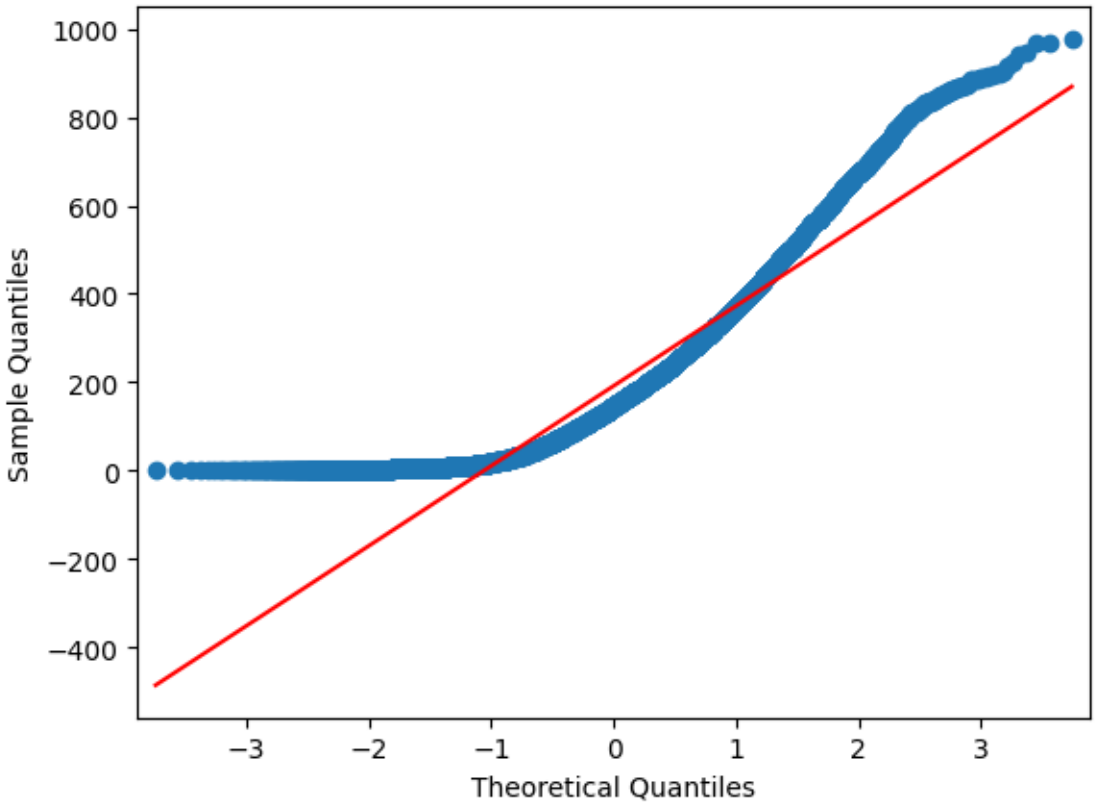
[44]:
```
np.float64(1.1664810025457815e-42)
```

Since p_value is less than alpha we are reject null hypothesis .Hence there is a significant in number of bikes rented in working day,

[23]:
```
#No. of cycles rented similar or different in different seasons

#Null hypothesis: number of cycles rented in different season and weather are␣
 ↪similar
#Alternate hypothesis: number of cycles rented in different seasons and weather␣
 ↪are different
#asssumptions of annova
```

```
s1= df[df['season']=='spring']['count'].values
s2=df[df['season']=='summer']['count'].values
s3=df[df['season']=='fall']['count'].values
s4 =df[df['season']=='winter']['count'].values

w1=df[df['weather']=='clear']['count'].values
w2=df[df['weather']=='cloudy']['count'].values
w3=df[df['weather']=='light_rain']['count'].values
w4=df[df['weather']=='heavy_rain']['count'].values

sns.histplot(x=df['count'],kde=True)
plt.show()

#qq plot to find normality
from statsmodels.api import qqplot
qqplot(df['count'],line='s')
plt.show()
```





[21]:
```
from scipy.stats import levene
t_stat,p_val=levene(s1,s2,s3,s4)
print(p_val)

t_stat, p_val =levene(w1,w2,w3,w4)
print(p_val)
```

```
1.0147116860043298e-118
3.504937946833238e-35
```

From the graph its clear that it doesnt follow normal distribution Levene test is used to check same variance . from the output its clear that variance is not same so one assumption is failed and cannot use anova test

[22]:
```
#kruskals test
from scipy.stats import kruskal
t_stat,p_val=kruskal(s1,s2,s3,s4)
print(p_val)

t_sta,p_val=kruskal(w1,w2,w3,w4)
print(p_val)
```

```
2.479008372608633e-151
3.501611300708679e-44
```

Since p-value is less than 0.05, we reject the null hypothesis. This implies that Number of cycles rented is not similar in different weather and season conditions.

[25]:
```
#is weather and season dependent

#to check this we can use chisquare contigency since 2 categorical variables␣
 ↪are there
#null_hypothesis: weather and season are independent
#Alternate_hypothesis:both are dependent

from scipy.stats import chi2_contingency
contigency_table=pd.crosstab(df['weather'],df['season'])
contigency_table
```

[25]:

season	spring	summer	fall	winter
weather				
clear	1759	1801	1930	1702
cloudy	715	708	604	807
light_rain	211	224	199	225
heavy_rain	1	0	0	0

[28]:
```
p_val=chi2_contingency(contigency_table)
p_val
```

[28]:
```
Chi2ContingencyResult(statistic=np.float64(49.15865559689363),
pvalue=np.float64(1.5499250736864862e-07), dof=9,
expected_freq=array([[1.77454639e+03, 1.80559765e+03, 1.80559765e+03,
1.80625831e+03],
       [6.99258130e+02, 7.11493845e+02, 7.11493845e+02, 7.11754180e+02],
       [2.11948742e+02, 2.15657450e+02, 2.15657450e+02, 2.15736359e+02],
       [2.46738931e-01, 2.51056403e-01, 2.51056403e-01, 2.51148264e-01]]))
```

Pvalue is less than alpha. weather and season are dependent and is statistically significant

Recommendation 1. Focus on Weather Conditions:

Recommendation: Prioritize bike availability and marketing efforts during clear weather conditions, as these periods demonstrate the highest demand. Consider offering weather-based discounts or promotions to encourage usage during cloudy or light rain.

Justification: analysis reveals that clear weather strongly correlates with increased bike rentals, while heavy rain significantly dampens demand. Capitalizing on favorable weather and mitigating the impact of adverse weather can optimize bike usage.

2. Address Seasonal Variations:

Recommendation: Implement seasonal pricing strategies, adjusting rates based on demand patterns. Explore targeted marketing campaigns during Spring and Winter to address unusual value patterns

observed in these seasons.

Justification: While initial analysis suggested seasonality might not be a major factor, further investigation revealed potential variations in demand during Spring and Winter. Fine-tuning pricing and marketing efforts can optimize revenue and bike utilization across seasons.

3. Leverage Working Day Insights:

Recommendation: Focus on serving the regular commuter segment during working days and explore opportunities to expand casual ridership during holidays.

Justification: analysis indicates that bike usage is significantly higher on working days, highlighting the importance of catering to commuters. However, exploring initiatives to encourage casual use during holidays can further broaden your customer base.