yulu-case-study

April 19, 2024

#YULU - ##Hypothesis Testing Business Case Study

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

[1]:
```
import numpy as np
import pandas as pd
from scipy.stats import f_oneway,shapiro,levene,kruskal,chi2_contingency
from scipy.stats import norm,binom
from scipy.stats import ttest_ind,ttest_rel,ttest_1samp
import statsmodels.api as sm
from scipy.stats import zscore
from statsmodels.formula.api import ols
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
!wget https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/
↪original/bike_sharing.csv?1642089089 -O 'yulu.csv'
```

```
--2024-04-19 10:03:02--  https://d2beiqkhq929f0.cloudfront.net/public_assets/ass
ets/000/001/428/original/bike_sharing.csv?1642089089
```

```
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)…
18.164.173.110, 18.164.173.18, 18.164.173.58, …
Connecting to d2beiqkhq929f0.cloudfront.net
(d2beiqkhq929f0.cloudfront.net)|18.164.173.110|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 648353 (633K) [text/plain]
Saving to: 'yulu.csv'

yulu.csv            100%[===================>] 633.16K  --.-KB/s    in 0.04s

2024-04-19 10:03:02 (16.5 MB/s) - 'yulu.csv' saved [648353/648353]
```

[3]:
```python
df=pd.read_csv('yulu.csv')
```

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)
- 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

[4]:
```python
df.head()
```

[4]:
```
              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
```

[5]:
```
df.tail()
```

[5]:
```
                  datetime  season  holiday  workingday  weather   temp  \
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
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
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    10886 non-null  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
```

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

[7]:
```
             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
```

[8]:
```
df.isna().sum()
```

[8]:
```
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
```

No Null Values in the given dataset.

[9]:
```
df.duplicated()
```

[9]:
```
0        False
1        False
2        False
3        False
4        False
         …
10881    False
10882    False
10883    False
10884    False
10885    False
Length: 10886, dtype: bool
```

No Duplicate values in the given dataset

[10]:
```python
df['season']=df['season'].astype('object')
df['holiday']=df['holiday'].astype('object')
df['workingday']=df['workingday'].astype('object')
df['weather']=df['weather'].astype('object')
```

[11]:
```python
df.dtypes
```

[11]:
```
datetime       object
season         object
holiday        object
workingday     object
weather        object
temp          float64
atemp         float64
humidity        int64
windspeed     float64
casual          int64
registered      int64
count           int64
dtype: object
```

[12]:
```python
df['datetime']=pd.to_datetime(df['datetime'])
```

[13]:
```python
df['datetime'].min()
```

[13]:
```
Timestamp('2011-01-01 00:00:00')
```

[14]:
```python
df['datetime'].max()
```

[14]:
```
Timestamp('2012-12-19 23:00:00')
```

[15]:
```python
df['datetime'].max() - df['datetime'].min()
```

[15]: Timedelta('718 days 23:00:00')

[16]:
```
#categorical columns

cat_cols = df.dtypes == "object"
cat_cols = list(cat_cols[cat_cols].index)
cat_cols
```

[16]: ['season', 'holiday', 'workingday', 'weather']

[17]:
```
#Numerical columns

num_cols = df.dtypes != "object"
num_cols = list(num_cols[num_cols].index)
num_cols.remove('datetime')
num_cols
```

[17]: ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count']

[18]:
```
df[num_cols].skew()
```

[18]:
```
temp          0.003691
atemp        -0.102560
humidity     -0.086335
windspeed     0.588767
casual        2.495748
registered    1.524805
count         1.242066
dtype: float64
```

[19]:
```
df[num_cols].kurt()
```

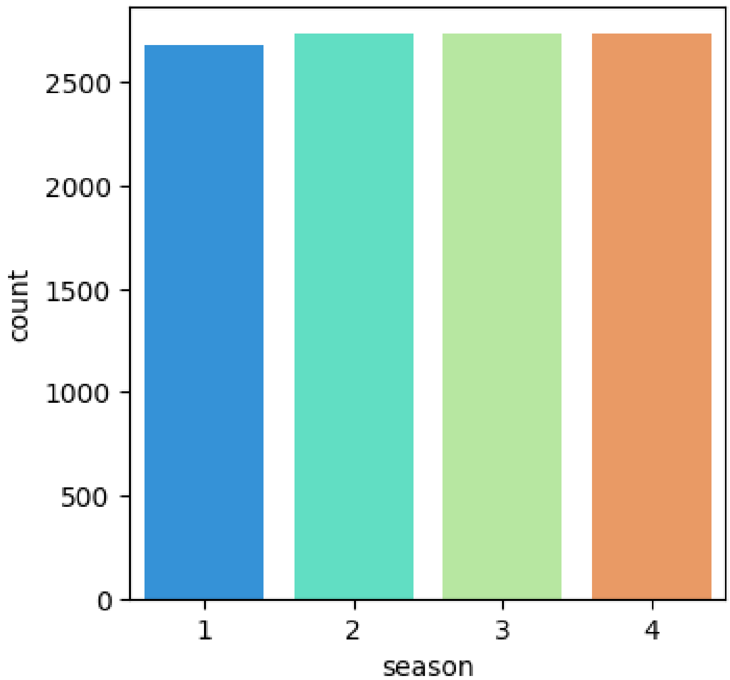
[19]:
```
temp         -0.914530
atemp        -0.850076
humidity     -0.759818
windspeed     0.630133
casual        7.551629
registered    2.626081
count         1.300093
dtype: float64
```

##Univariate Analysis

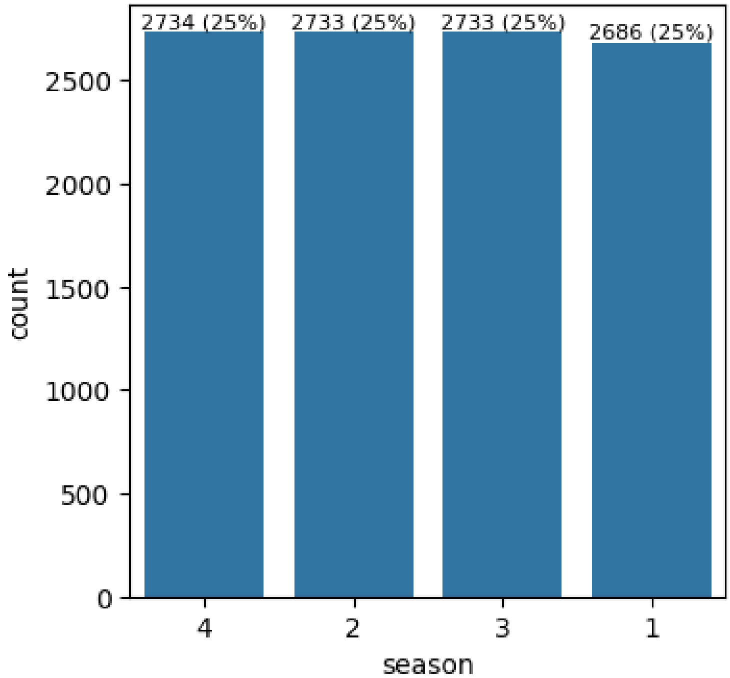
[20]:
```
plt.figure(figsize=(4,4))
sns.countplot(data = df, x = "season",palette='rainbow')
plt.show()
```

```
<ipython-input-20-493f818e79ac>:2: FutureWarning:
```

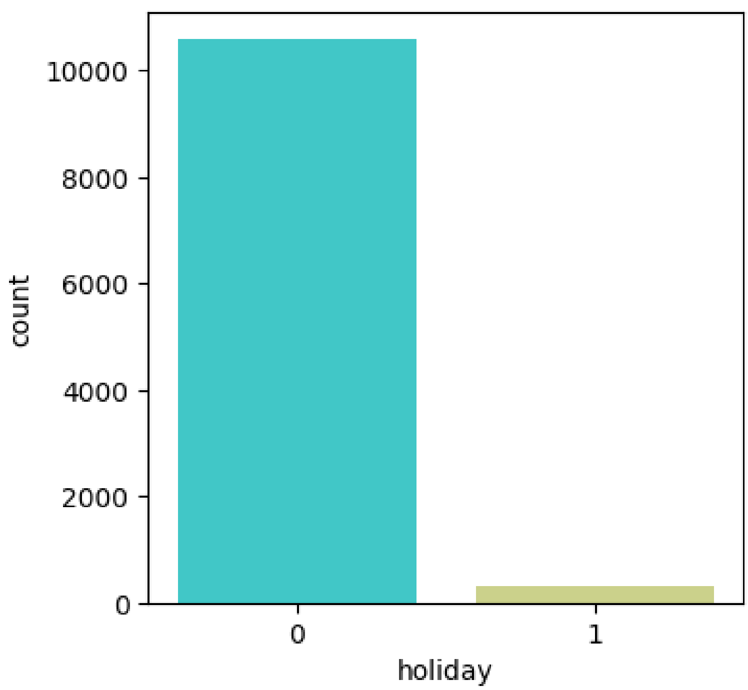
```
Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(data = df, x = "season",palette='rainbow')
```



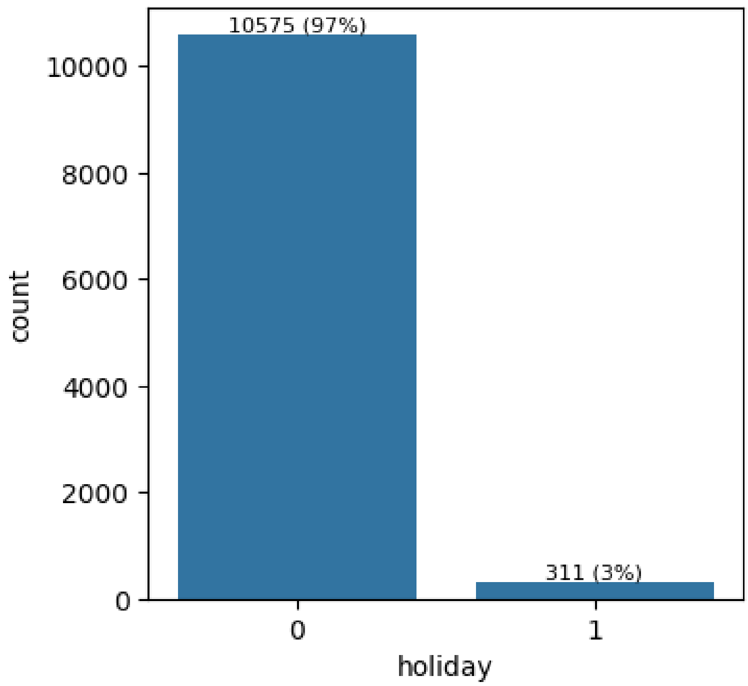
[21]:
```
plt.figure(figsize=(4,4))
ax = sns.countplot(x=df['season'],order=df['season'].
 ↪value_counts(ascending=False).index)
abs_values = df['season'].value_counts(ascending=False)
rel_values = df['season'].value_counts(ascending=False, normalize=True).values␣
 ↪* 100
lbls = [f'{p[0]} ({p[1]:.0f}%)' for p in zip(abs_values, rel_values)]
ax.bar_label(container=ax.containers[0], labels=lbls,fontsize=8)
plt.show()
```



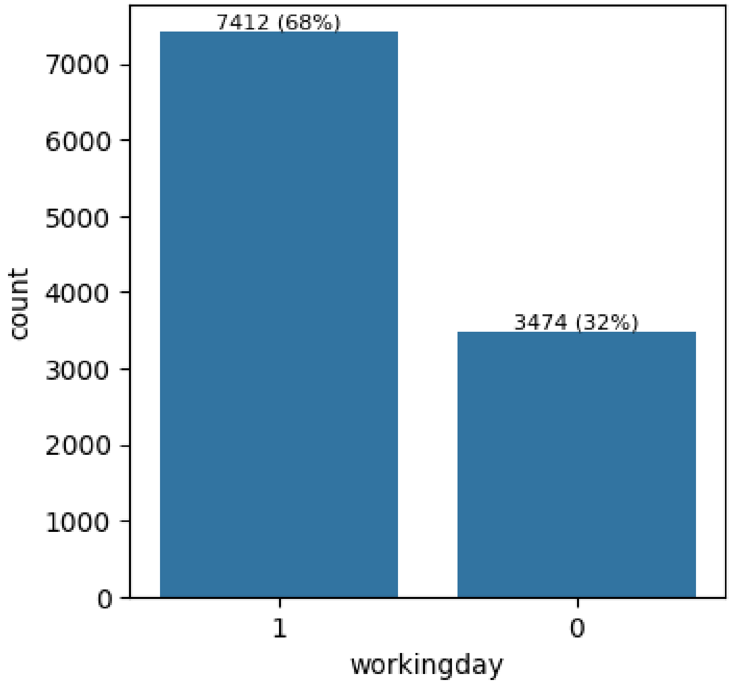
[79]:
```
plt.figure(figsize=(4,4))
sns.countplot(data = df, x = "holiday",palette='rainbow')
plt.show()
```



[23]:
```
plt.figure(figsize=(4,4))
ax = sns.countplot(x=df['holiday'],order=df['holiday'].
 ↪value_counts(ascending=False).index)
abs_values = df['holiday'].value_counts(ascending=False)
rel_values = df['holiday'].value_counts(ascending=False, normalize=True).values␣
 ↪* 100
lbls = [f'{p[0]} ({p[1]:.0f}%)' for p in zip(abs_values, rel_values)]
ax.bar_label(container=ax.containers[0], labels=lbls,fontsize=8)
plt.show()
```

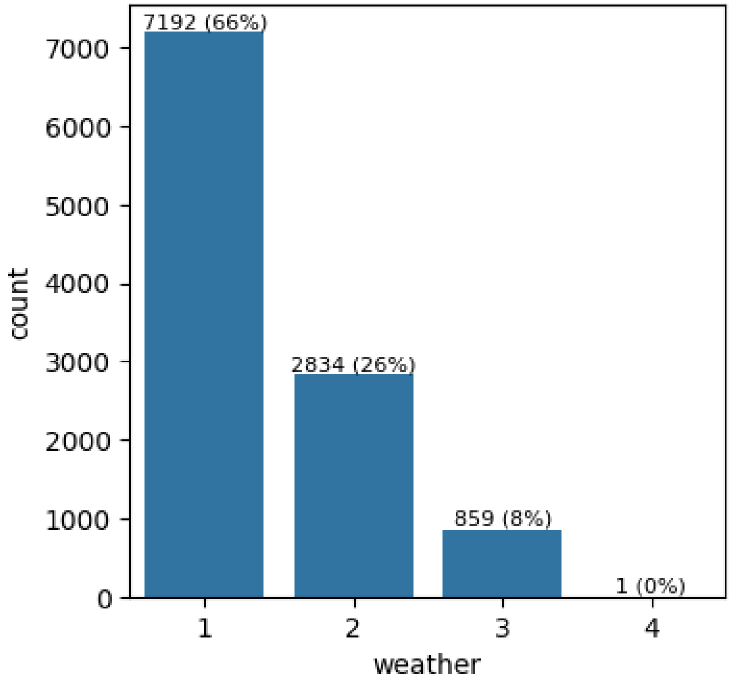


[24]:
```
plt.figure(figsize=(4,4))
ax = sns.countplot(x=df['workingday'],order=df['workingday'].
 ↪value_counts(ascending=False).index)
abs_values = df['workingday'].value_counts(ascending=False)
rel_values = df['workingday'].value_counts(ascending=False, normalize=True).
 ↪values * 100
lbls = [f'{p[0]} ({p[1]:.0f}%)' for p in zip(abs_values, rel_values)]
ax.bar_label(container=ax.containers[0], labels=lbls,fontsize=8)
plt.show()
```



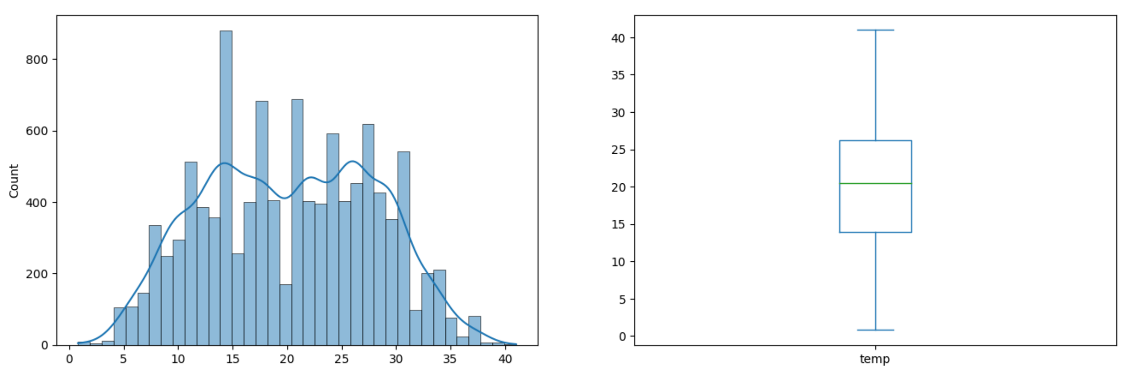
[25]:

```
plt.figure(figsize=(4,4))
ax = sns.countplot(x=df['weather'],order=df['weather'].
 ↪value_counts(ascending=False).index)
abs_values = df['weather'].value_counts(ascending=False)
rel_values = df['weather'].value_counts(ascending=False, normalize=True).values␣
 ↪* 100
lbls = [f'{p[0]} ({p[1]:.0f}%)' for p in zip(abs_values, rel_values)]
ax.bar_label(container=ax.containers[0], labels=lbls,fontsize=8)
plt.show()
```

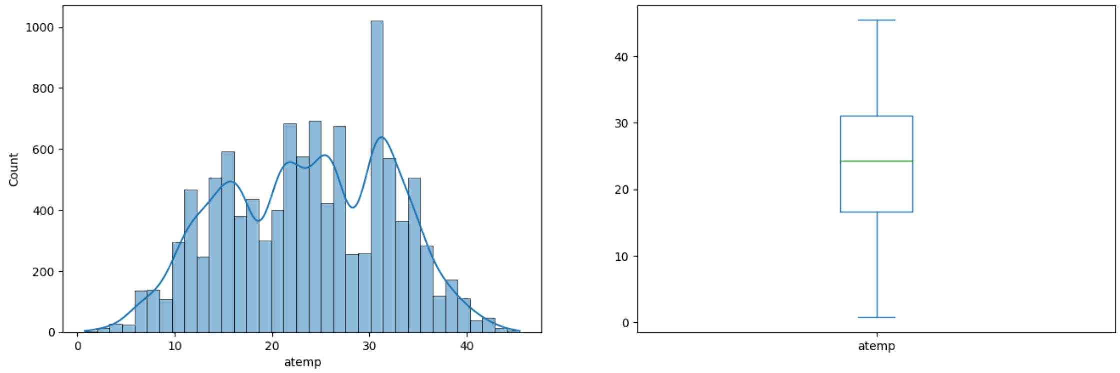


[26]:

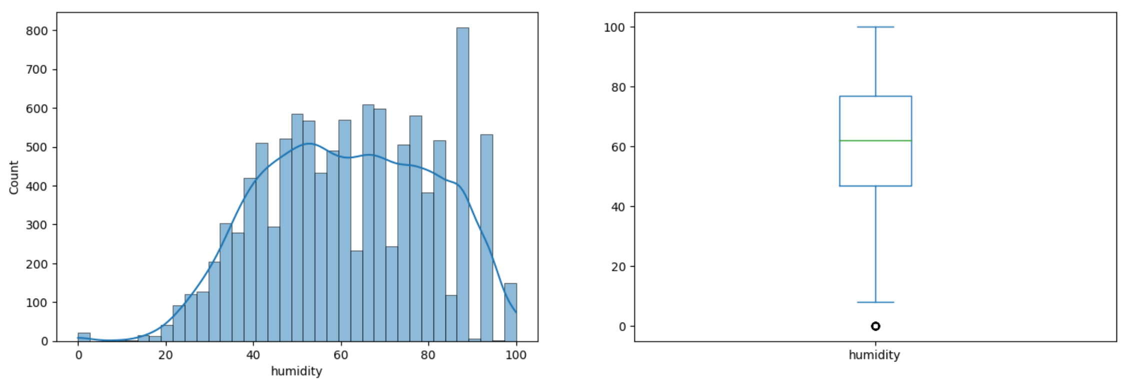
```
#Numerical columns analysis - ['temp', 'atemp', 'humidity', 'windspeed',
↪'casual', 'registered', 'count']
plt.subplot(121)
sns.histplot(df['temp'], kde=True)

plt.subplot(122)
df['temp'].plot.box(figsize=(16,5))
plt.show()
```



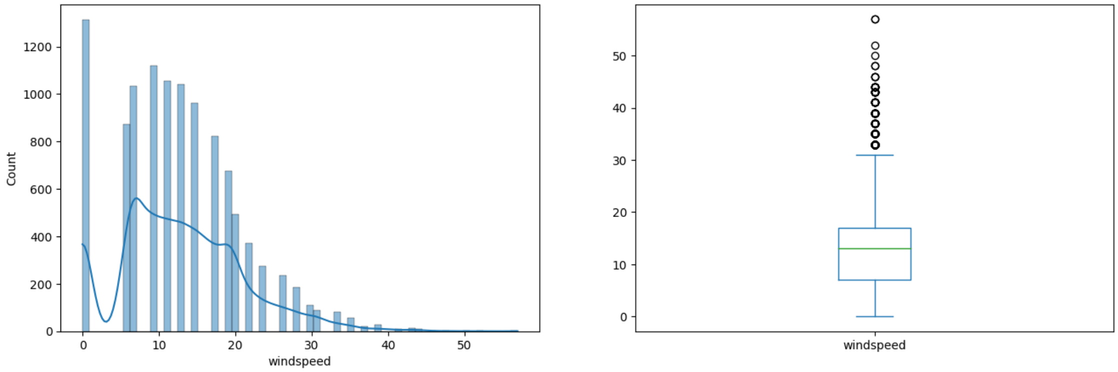
[27]:
```
plt.subplot(121)
sns.histplot(df['atemp'], kde=True)

plt.subplot(122)
df['atemp'].plot.box(figsize=(16,5))
plt.show()
```



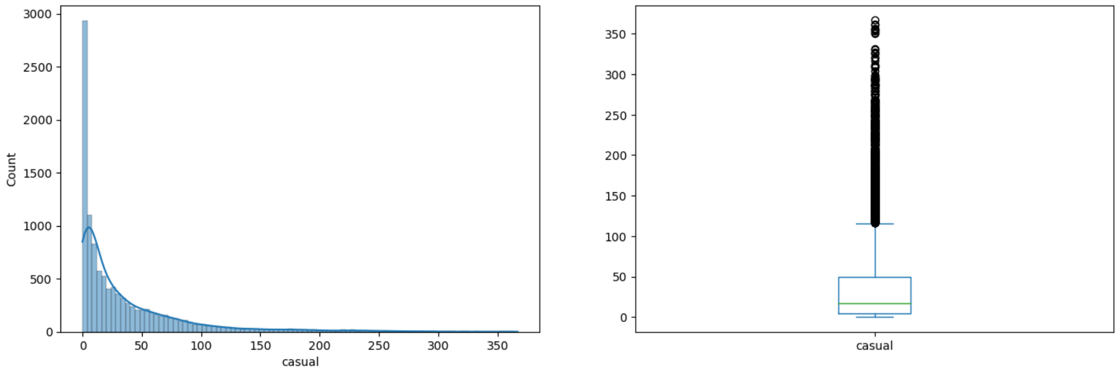
[28]:
```
plt.subplot(121)
sns.histplot(df['humidity'], kde=True)

plt.subplot(122)
df['humidity'].plot.box(figsize=(16,5))
plt.show()
```



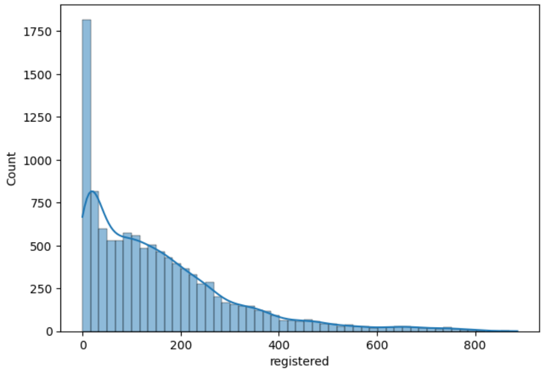
[29]:
```
plt.subplot(121)
sns.histplot(df['windspeed'], kde=True)

plt.subplot(122)
df['windspeed'].plot.box(figsize=(16,5))
plt.show()
```

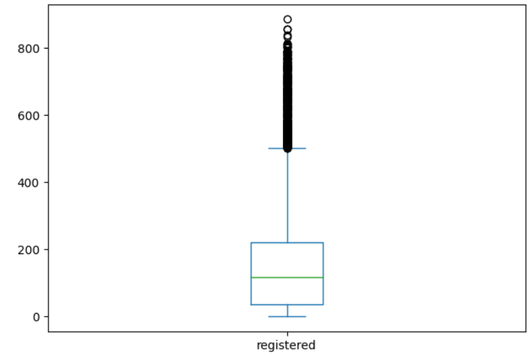


[30]:
```
plt.subplot(121)
sns.histplot(df['casual'], kde=True)

plt.subplot(122)
df['casual'].plot.box(figsize=(16,5))
plt.show()
```

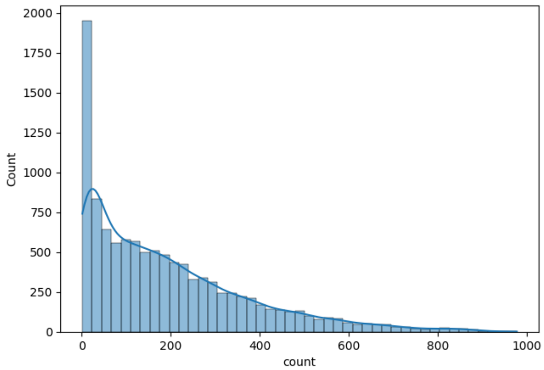


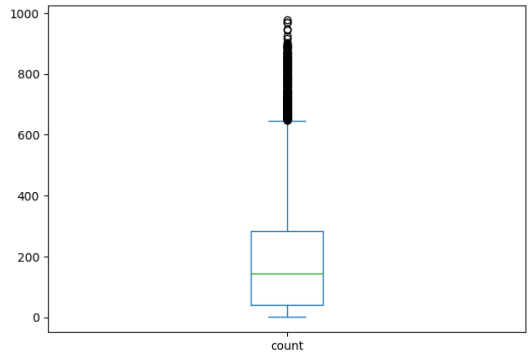
[31]:
```
plt.subplot(121)
sns.histplot(df['registered'], kde=True)

plt.subplot(122)
df['registered'].plot.box(figsize=(16,5))
plt.show()
```





[32]:
```python
plt.subplot(121)
sns.histplot(df['count'], kde=True)

plt.subplot(122)
df['count'].plot.box(figsize=(16,5))
plt.show()
```



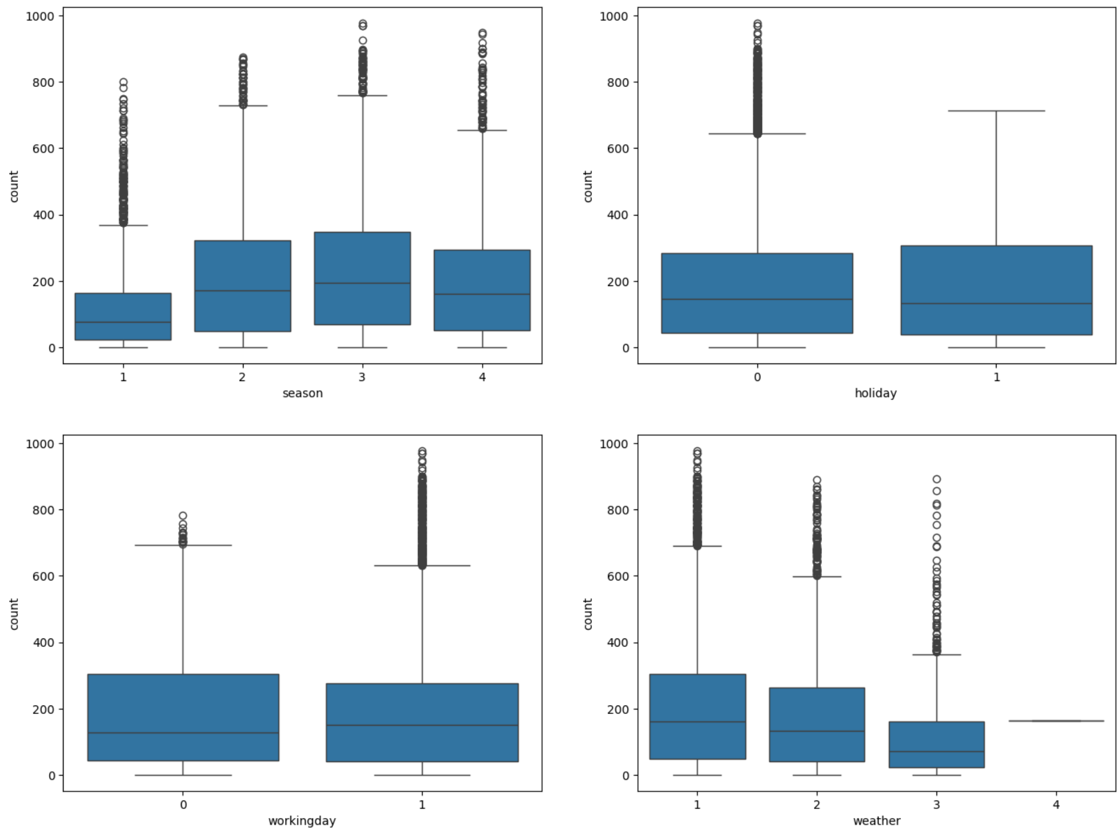


- There is no outlier in the temp column.
- There are few outliers present in humidity column.
- There are many outliers present in each of the columns : windspeed, casual, registered, count.

##Bivariate Analysis

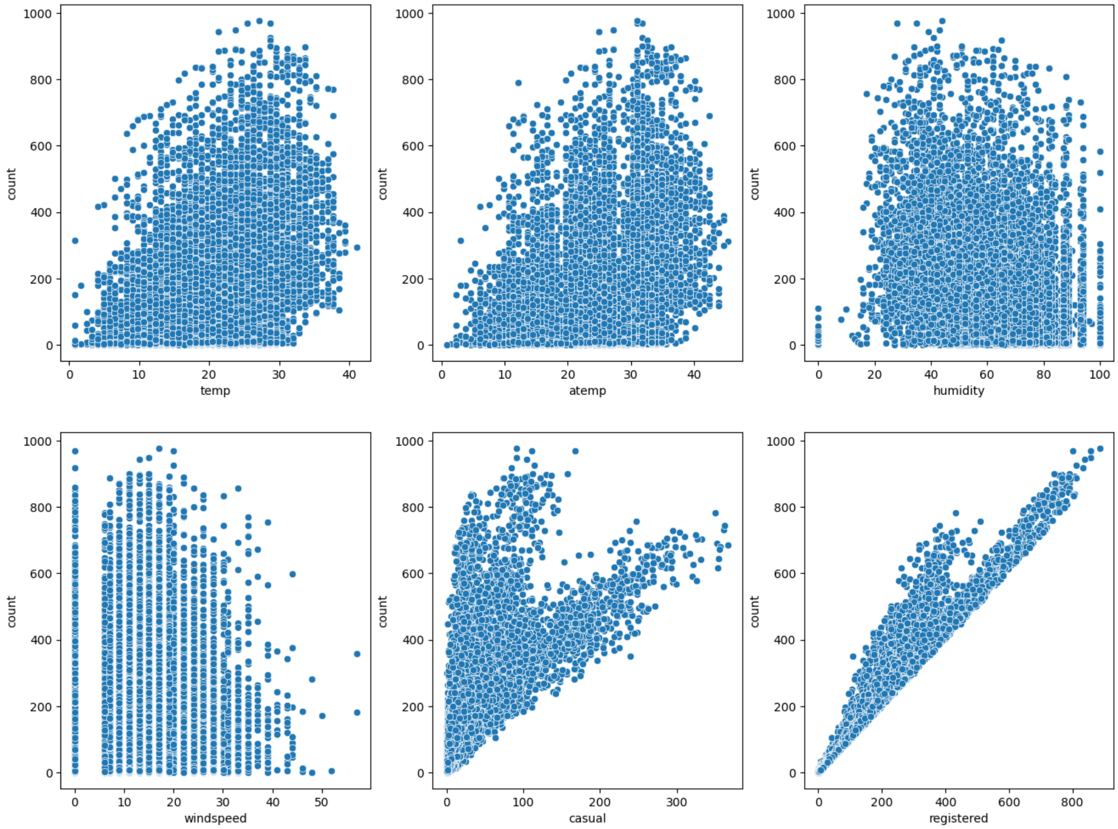
[80]:
```python
# plotting categorical variables againt count using boxplots
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(16, 12))

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

```
plt.show()
```



- 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.

[81]:
```
# plotting numerical variables againt count using scatterplot
fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(16, 12))

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

plt.show()
```



##Multivariate Analysis - Heatmap and Correlation

[34]:
```
val=df.corr()
val
```

[34]:
```
            datetime    season   holiday  workingday   weather      temp  \
datetime    1.000000  0.480021  0.010988   -0.003658 -0.005048  0.180986
season      0.480021  1.000000  0.029368   -0.008126  0.008879  0.258689
holiday     0.010988  0.029368  1.000000   -0.250491 -0.007074  0.000295
workingday -0.003658 -0.008126 -0.250491    1.000000  0.033772  0.029966
weather    -0.005048  0.008879 -0.007074    0.033772  1.000000 -0.055035
temp        0.180986  0.258689  0.000295    0.029966 -0.055035  1.000000
atemp       0.181823  0.264744 -0.005215    0.024660 -0.055376  0.984948
humidity    0.032856  0.190610  0.001929   -0.010880  0.406244 -0.064949
windspeed  -0.086888 -0.147121  0.008409    0.013373  0.007261 -0.017852
casual      0.172728  0.096758  0.043799   -0.319111 -0.135918  0.467097
registered  0.314879  0.164011 -0.020956    0.119460 -0.109340  0.318571
count       0.310187  0.163439 -0.005393    0.011594 -0.128655  0.394454

               atemp  humidity  windspeed    casual  registered     count
datetime    0.181823  0.032856  -0.086888  0.172728    0.314879  0.310187
```

```
season      0.264744  0.190610 -0.147121  0.096758    0.164011  0.163439
holiday    -0.005215  0.001929  0.008409  0.043799   -0.020956 -0.005393
workingday  0.024660 -0.010880  0.013373 -0.319111    0.119460  0.011594
weather    -0.055376  0.406244  0.007261 -0.135918   -0.109340 -0.128655
temp        0.984948 -0.064949 -0.017852  0.467097    0.318571  0.394454
atemp       1.000000 -0.043536 -0.057473  0.462067    0.314635  0.389784
humidity   -0.043536  1.000000 -0.318607 -0.348187   -0.265458 -0.317371
windspeed  -0.057473 -0.318607  1.000000  0.092276    0.091052  0.101369
casual      0.462067 -0.348187  0.092276  1.000000    0.497250  0.690414
registered  0.314635 -0.265458  0.091052  0.497250    1.000000  0.970948
count       0.389784 -0.317371  0.101369  0.690414    0.970948  1.000000
```

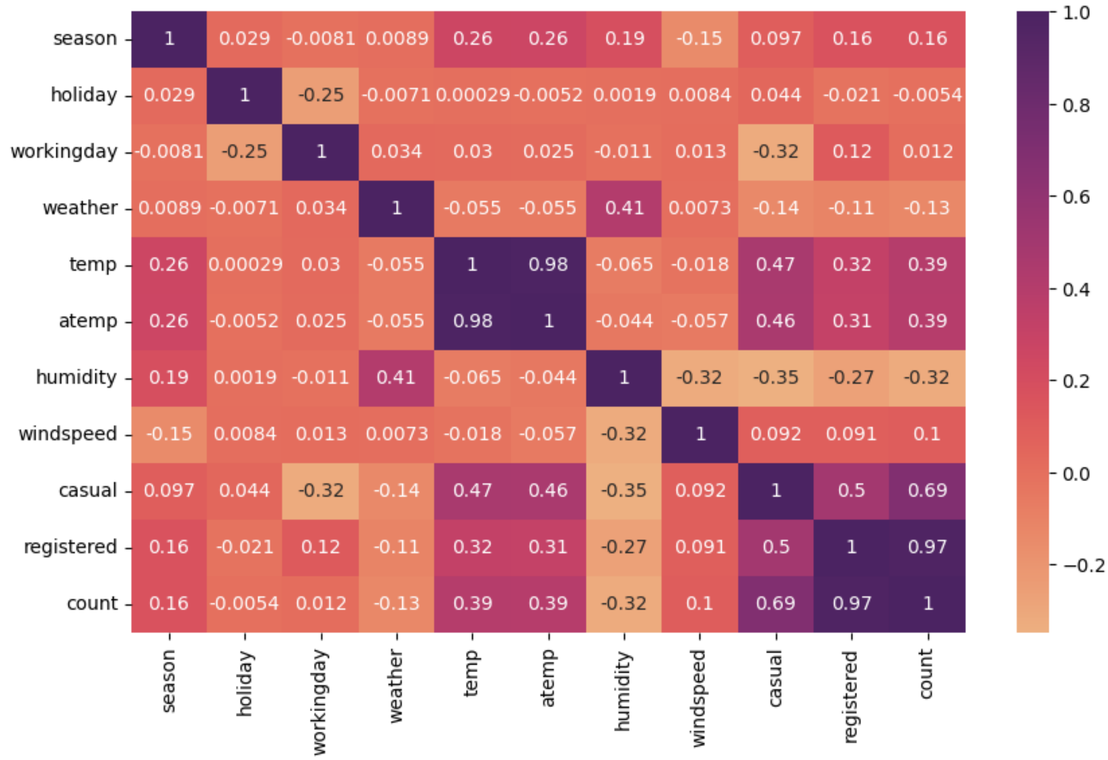
[35]:
```python
val.drop('datetime',axis=1,inplace=True)
val.drop('datetime',axis=0,inplace=True)
```

[36]:
```python
val
```

[36]:
```
              season   holiday  workingday   weather      temp     atemp  \
season      1.000000  0.029368   -0.008126  0.008879  0.258689  0.264744
holiday     0.029368  1.000000   -0.250491 -0.007074  0.000295 -0.005215
workingday -0.008126 -0.250491    1.000000  0.033772  0.029966  0.024660
weather     0.008879 -0.007074    0.033772  1.000000 -0.055035 -0.055376
temp        0.258689  0.000295    0.029966 -0.055035  1.000000  0.984948
atemp       0.264744 -0.005215    0.024660 -0.055376  0.984948  1.000000
humidity    0.190610  0.001929   -0.010880  0.406244 -0.064949 -0.043536
windspeed  -0.147121  0.008409    0.013373  0.007261 -0.017852 -0.057473
casual      0.096758  0.043799   -0.319111 -0.135918  0.467097  0.462067
registered  0.164011 -0.020956    0.119460 -0.109340  0.318571  0.314635
count       0.163439 -0.005393    0.011594 -0.128655  0.394454  0.389784

            humidity  windspeed    casual  registered     count
season      0.190610  -0.147121  0.096758    0.164011  0.163439
holiday     0.001929   0.008409  0.043799   -0.020956 -0.005393
workingday -0.010880   0.013373 -0.319111    0.119460  0.011594
weather     0.406244   0.007261 -0.135918   -0.109340 -0.128655
temp       -0.064949  -0.017852  0.467097    0.318571  0.394454
atemp      -0.043536  -0.057473  0.462067    0.314635  0.389784
humidity    1.000000  -0.318607 -0.348187   -0.265458 -0.317371
windspeed  -0.318607   1.000000  0.092276    0.091052  0.101369
casual     -0.348187   0.092276  1.000000    0.497250  0.690414
registered -0.265458   0.091052  0.497250    1.000000  0.970948
count      -0.317371   0.101369  0.690414    0.970948  1.000000
```

[37]:
```python
plt.figure(figsize=(10,6))
sns.heatmap(val,annot=True,cmap='flare')
```

[37]: <Axes: >



- Very High Correlation (> 0.97) exists between columns [atemp, temp] and [count, registered]
- Moderate positive correlation (0.5 - 0.7) exists between columns [casual, count], [casual, registered].
- Low Positive correlation (0.3 - 0.5) exists between columns [count, temp], [count, atemp], [casual, atemp]

[38]:
```
df['workingday'].value_counts()
```

[38]:
```
workingday
1    7412
0    3474
Name: count, dtype: int64
```

[82]:
```
rides_weekday = df[df['workingday']==1]['count']
rides_weekend = df[(df['workingday']!=1)]['count']
```

###Formulation of Hypothesis:

Null Hypothesis:

H0 : There is no significant difference between the number of bike rides on Weekdays and Weekends

Alternate Hypothesis:

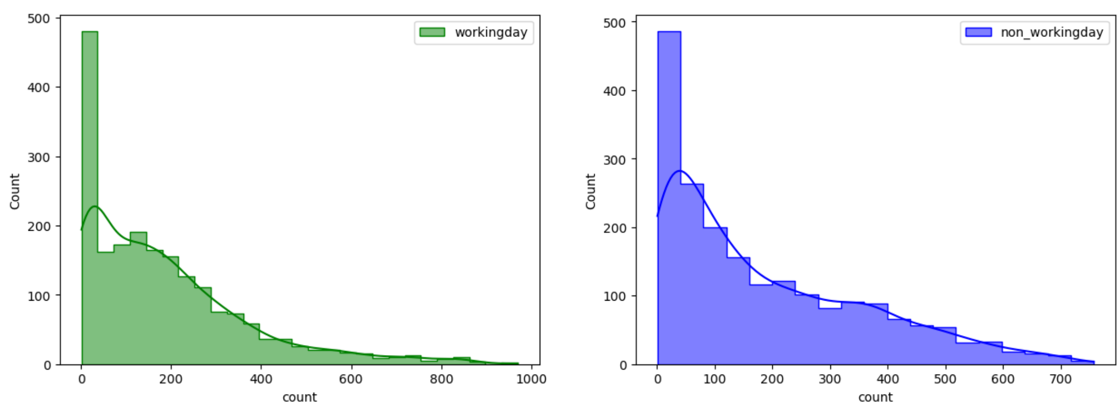
Ha : There is a significant difference between the number of bike rides on Weekdays and Weekends

Test selected : 2- Sample Independent T-test

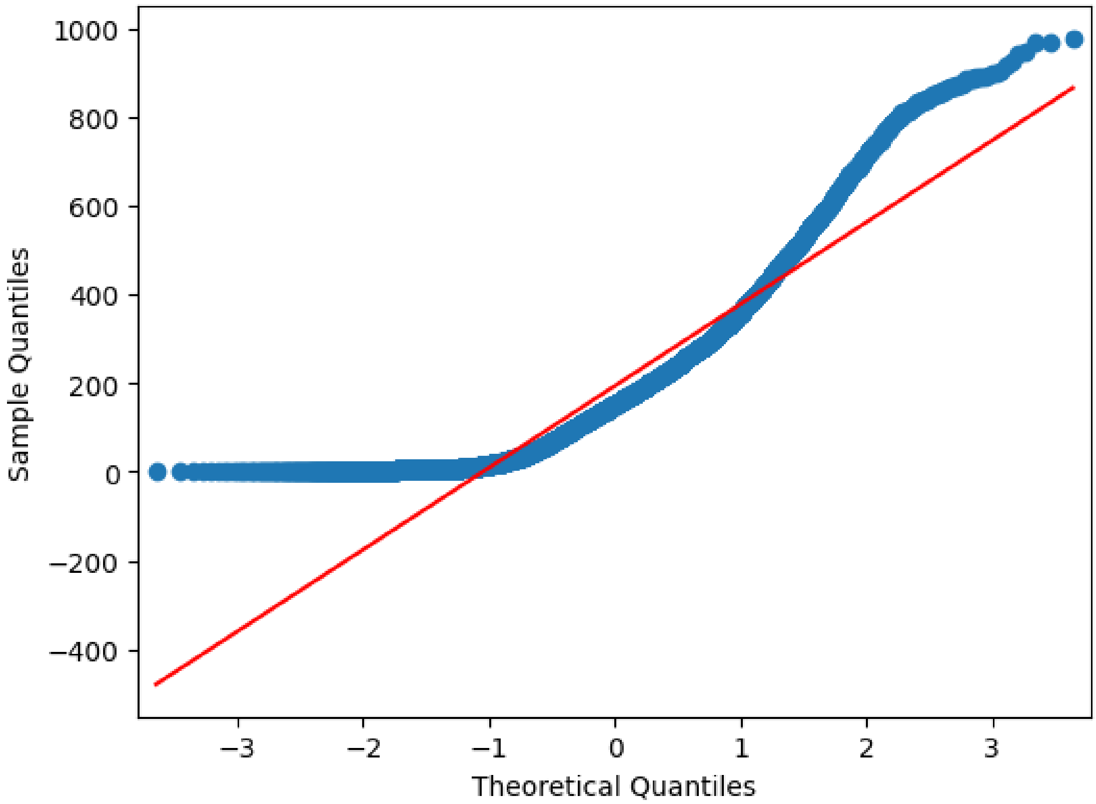
alpha value = 0.05 (recommended)

[84]:
```
plt.figure(figsize = (15, 5))
plt.subplot(1, 2, 1)
sns.histplot(df.loc[df['workingday'] == 1, 'count'].sample(2000),
             element = 'step', color = 'green', kde = True, label =␣
 ↪'workingday')
plt.legend()
plt.subplot(1, 2, 2)
sns.histplot(df.loc[df['workingday'] == 0, 'count'].sample(2000),
             element = 'step', color = 'blue', kde = True, label =␣
 ↪'non_workingday')
plt.legend()
plt.plot()
```

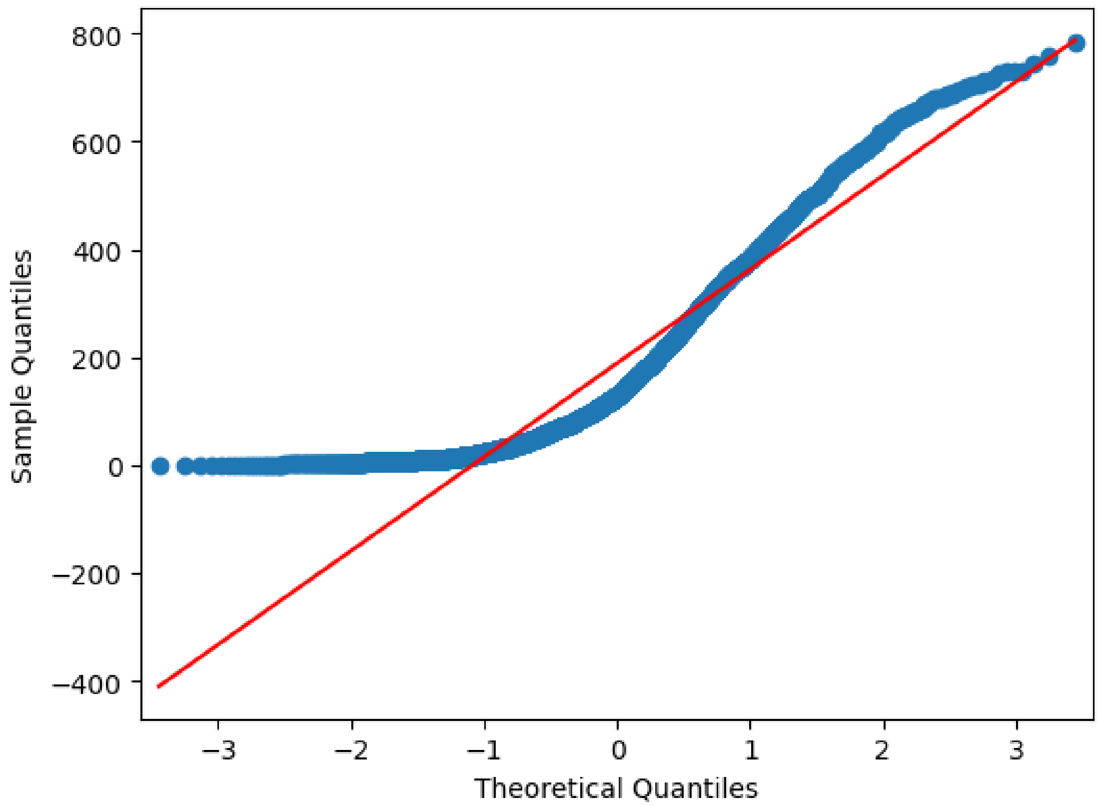
[84]: []



[88]:
```
sm.qqplot(rides_weekday,line='s')
plt.show()
```



[90]:
```
sm.qqplot(rides_weekend,line='s')
plt.show()
```



[83]:

```
t_stat,p_val = ttest_ind(rides_weekday,rides_weekend)
print("T Statistic Value is: ", t_stat)
print("p_value is: ",p_val)

alpha=0.05

if p_val>alpha:
  print("P is high, so we Fail to Reject the Null Hypothesis H0")
  print("Thus, we say that --> There is no significant difference between the␣
 ↪number of bike rides on Weekdays and Weekends\nWe don't have the sufficient␣
 ↪evidence to say that working day has effect on the number of cycles being␣
 ↪rented.")
else:
  print("P is low, so we Reject the Null Hypothesis H0")
  print("Thus, we say that --> There is a significant difference between the␣
 ↪number of bike rides on Weekdays and Weekends")
```

```
T Statistic Value is:  1.2096277376026694
p_value is:  0.22644804226361348
P is high, so we Fail to Reject the Null Hypothesis H0
Thus, we say that --> There is no significant difference between the number of
```

```
bike rides on Weekdays and Weekends
We don't have the sufficient evidence to say that working day has effect on the
number of cycles being rented.
```

Check if the demand of bicycles on rent is the same for different Weather conditions?

Null Hypothesis:

H0 : Demand of bicycles on rent is the same for different Weather conditions

Alternate Hypothesis:

Ha : Demand of bicycles on rent is different for different Weather conditions

Test selected : One-way ANOVA/ Kruskal Wallis

alpha value = 0.05 (recommended)

Check assumptions of the test–>

i. Normality

1. Use Histogram, Q-Q Plot, Skewness & Kurtosis
2. Shapiro-Wilk's test

ii. Equality Variance

1. Levene's test

[41]:
```
df['weather'].value_counts()
```

[41]:
```
weather
1    7192
2    2834
3     859
4       1
Name: count, dtype: int64
```

[42]:
```
#for weather - 1: Clear, Few clouds, partly cloudy
weather_1_demand = df[df['weather']==1]['count']

#for weather - 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
weather_2_demand = df[df['weather']==2]['count']

#for weather - 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds,␣
↪Light Rain +Scattered clouds
weather_3_demand = df[df['weather']==3]['count']

#for weather - 4: Heavy Rain + Ice Pellets + Thunderstorm + Mist, Snow + Fog
weather_4_demand = df[df['weather']==4]['count']
```

[43]:
```
#Checking for Skewness:

print(weather_1_demand.skew(), weather_2_demand.skew(), weather_3_demand.
 ↪skew(), weather_4_demand.skew())
```

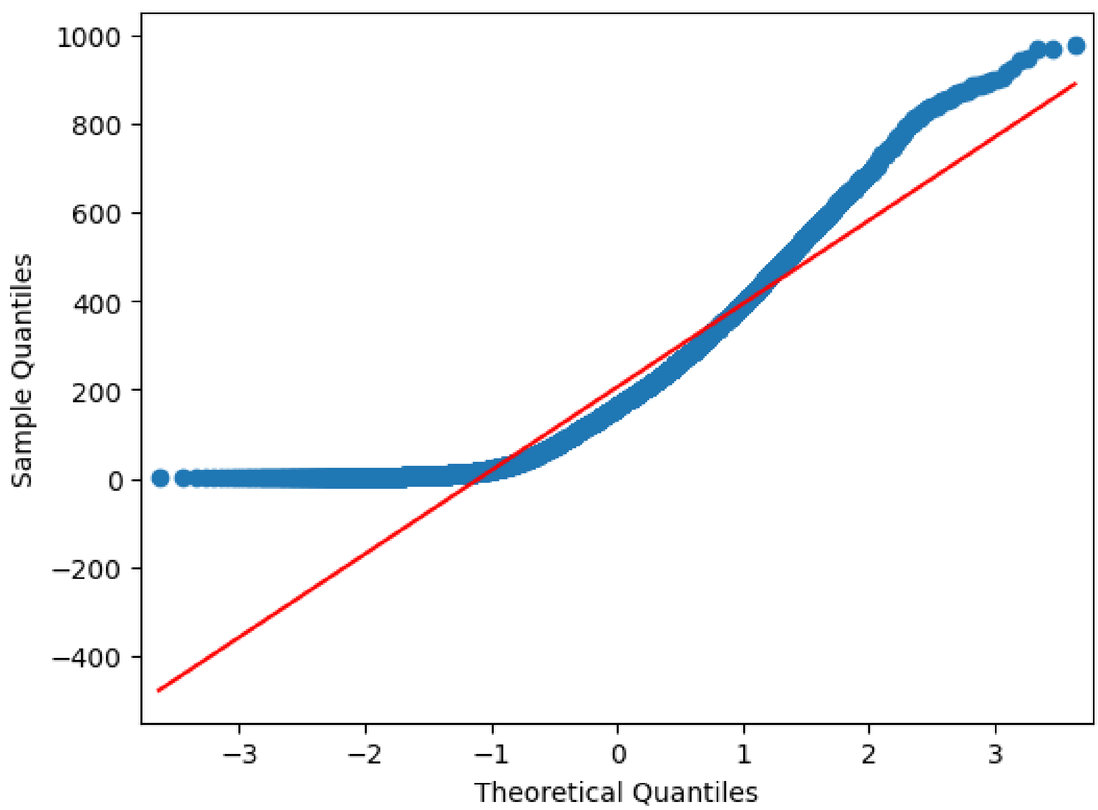
1.1398572666918205 1.294444423357868 2.1871371080456594 nan

[44]:
```
#kurtosis value check

print(weather_1_demand.kurt(), weather_2_demand.kurt(), weather_3_demand.
 ↪kurt(), weather_4_demand.kurt())
```

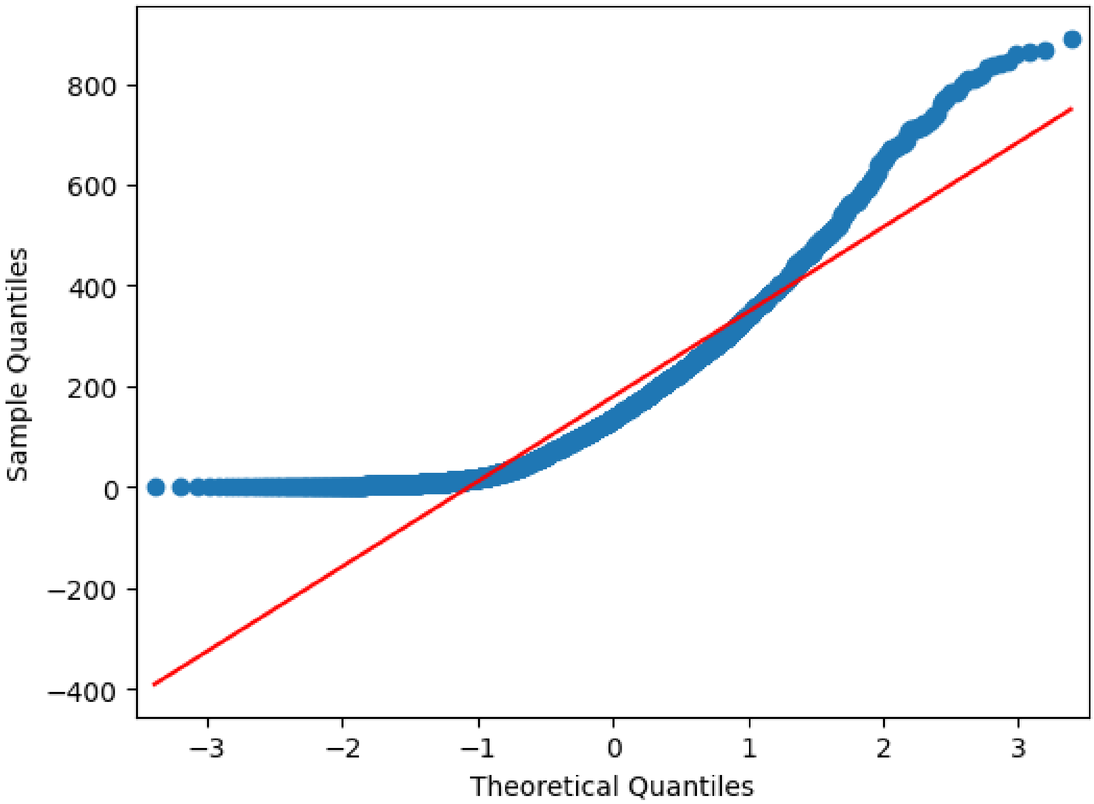
0.964719852310354 1.5884304891319174 6.003053730759276 nan

QQ Plots

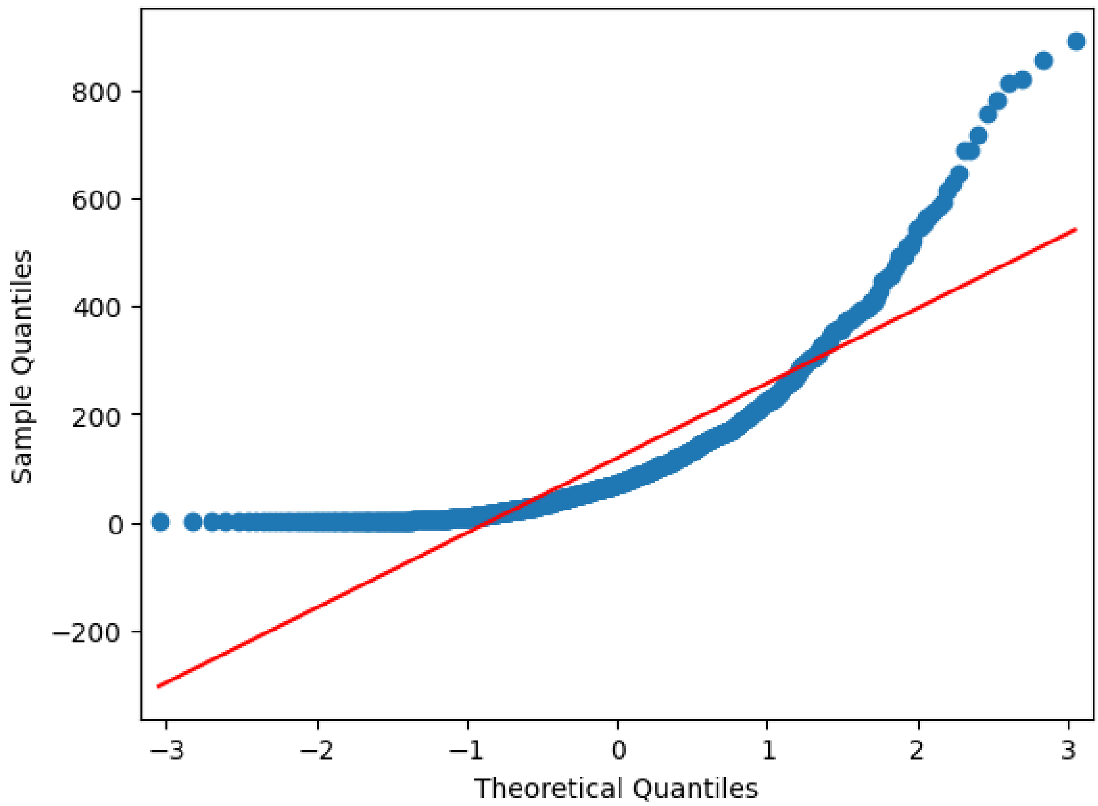
[45]:
```
sm.qqplot(weather_1_demand,line='s')
plt.show()
```



[46]:
```
sm.qqplot(weather_2_demand,line='s')
plt.show()
```

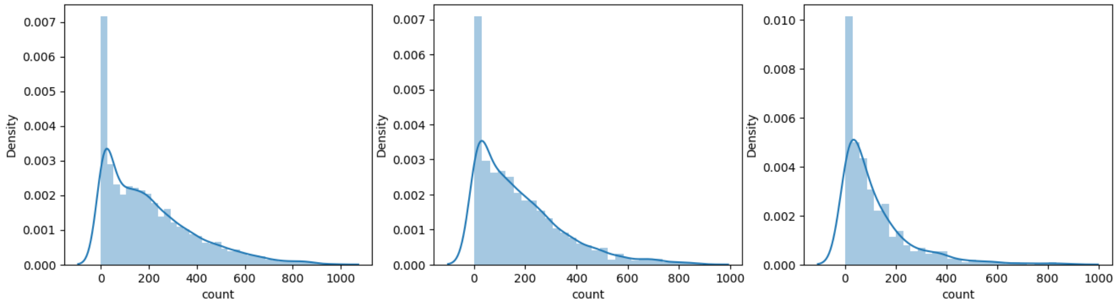


[47]:
```
sm.qqplot(weather_3_demand,line='s')
plt.show()
```



[91]:
```
import warnings
warnings.filterwarnings('ignore')
plt.figure(figsize=(16,4))
plt.subplot(1,3,1)
sns.distplot(weather_1_demand)
plt.subplot(1,3,2)
sns.distplot(weather_2_demand)
plt.subplot(1,3,3)
sns.distplot(weather_3_demand)
#plt.subplot(2,2,4)
#sns.distplot(weather_4_demand) -- SInce this has only 1 record, does not make␣
↪sense to draw the normality graph/ distplot for weather 4
```

[91]:
```
<Axes: xlabel='count', ylabel='Density'>
```



Shapiro Test for Normality check

[49]:
```
shapiro(weather_1_demand)
```

```
/usr/local/lib/python3.10/dist-packages/scipy/stats/_morestats.py:1882:
UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

[49]:
```
ShapiroResult(statistic=0.8909230828285217, pvalue=0.0)
```

[50]:
```
shapiro(weather_2_demand)
```

[50]:
```
ShapiroResult(statistic=0.8767687082290649, pvalue=9.781063280987223e-43)
```

[51]:
```
shapiro(weather_3_demand)
```

[51]:
```
ShapiroResult(statistic=0.7674332857131958, pvalue=3.876090133422781e-33)
```

[52]:
```
levene(weather_1_demand,weather_2_demand,weather_3_demand,weather_4_demand)
```

[52]:
```
LeveneResult(statistic=54.85106195954556, pvalue=3.504937946833238e-35)
```

The distributions are not Normal or does not have equal variances. GOing further with Kruskal Test.

[53]:
```
kruskal(weather_1_demand,weather_2_demand,weather_3_demand,weather_4_demand)
```

[53]:
```
KruskalResult(statistic=205.00216514479087, pvalue=3.501611300708679e-44)
```

[54]:
```
k_stat,p_val =
 ↪kruskal(weather_1_demand,weather_2_demand,weather_3_demand,weather_4_demand)
print("T Statistic Value is: ", k_stat)
print("p_value is: ",p_val)

alpha=0.05
```

```
if p_val>alpha:
  print("P is high, so we Fail to Reject the Null Hypothesis H0")
  print("Thus, we say that --> The demand of bicycles on rent is the same for␣
 ↪different Weather conditions")
else:
  print("P is low, so we Reject the Null Hypothesis H0")
  print("Thus, we say that --> The demand of bicycles on rent is different for␣
 ↪different Weather conditions")
```

```
T Statistic Value is:  205.00216514479087
p_value is:  3.501611300708679e-44
P is low, so we Reject the Null Hypothesis H0
Thus, we say that --> The demand of bicycles on rent is different for different
Weather conditions
```

[92]:
```
#Checking that One way Anova would also give the same result of hypothesis.
f_oneway(weather_1_demand,weather_2_demand,weather_3_demand,weather_4_demand)
```

[92]:
```
F_onewayResult(statistic=65.53024112793271, pvalue=5.482069475935669e-42)
```

Check if the demand of bicycles on rent is the same for different Seasons?

[56]:
```
df['season'].value_counts()
```

[56]:
```
season
4    2734
2    2733
3    2733
1    2686
Name: count, dtype: int64
```

[57]:
```
#for season - spring
season_spring_demand = df[df['season']==1]['count']

#for season - summer
season_summer_demand = df[df['season']==2]['count']

#for season - fall
season_fall_demand = df[df['season']==3]['count']

#for season - winter
season_winter_demand = df[df['season']==4]['count']
```

Null Hypothesis:

H0 : Demand of bicycles on rent is the same for different Seasons

Alternate Hypothesis:

Ha : Demand of bicycles on rent is different for different Seasons

Test selected : One-way ANOVA/ Kruskal Wallis

alpha value = 0.05 (recommended)

[58]:
```
#Checking for Skewness:

print(season_spring_demand.skew(),season_summer_demand.
 ↪skew(),season_fall_demand.skew(),season_winter_demand.skew())
```

```
1.8880559001782309 1.0032642267278118 0.9914946474772749 1.172117329762622
```

[59]:
```
#Kurtosis value check:

print(season_spring_demand.kurt(),season_summer_demand.
 ↪kurt(),season_fall_demand.kurt(),season_winter_demand.kurt())
```

```
4.31475739331681 0.42521337827415717 0.6993825795653992 1.2734853552995302
```

Shapiro Test for Normality Check

[60]:
```
shapiro(season_spring_demand)
```

[60]:
```
ShapiroResult(statistic=0.8087388873100281, pvalue=0.0)
```

[61]:
```
shapiro(season_summer_demand)
```

[61]:
```
ShapiroResult(statistic=0.900481641292572, pvalue=6.039093315091269e-39)
```

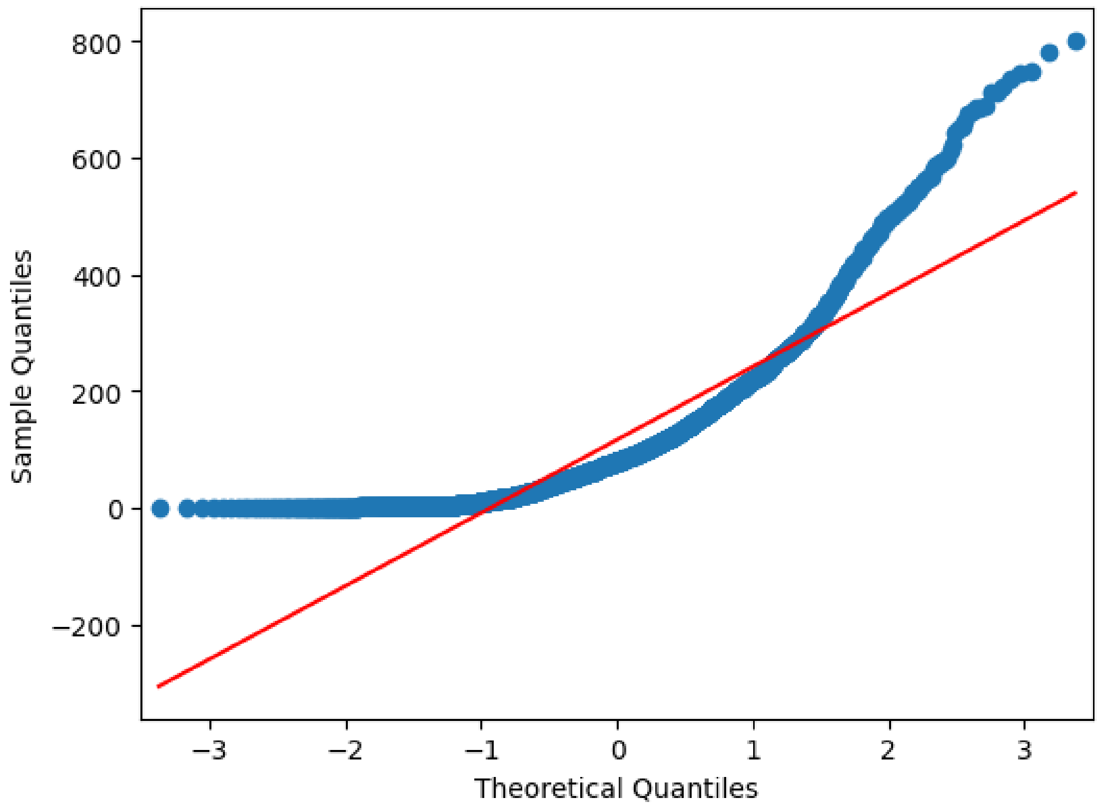
[62]:
```
shapiro(season_fall_demand)
```

[62]:
```
ShapiroResult(statistic=0.9148160815238953, pvalue=1.043458045587339e-36)
```

[63]:
```
shapiro(season_winter_demand)
```

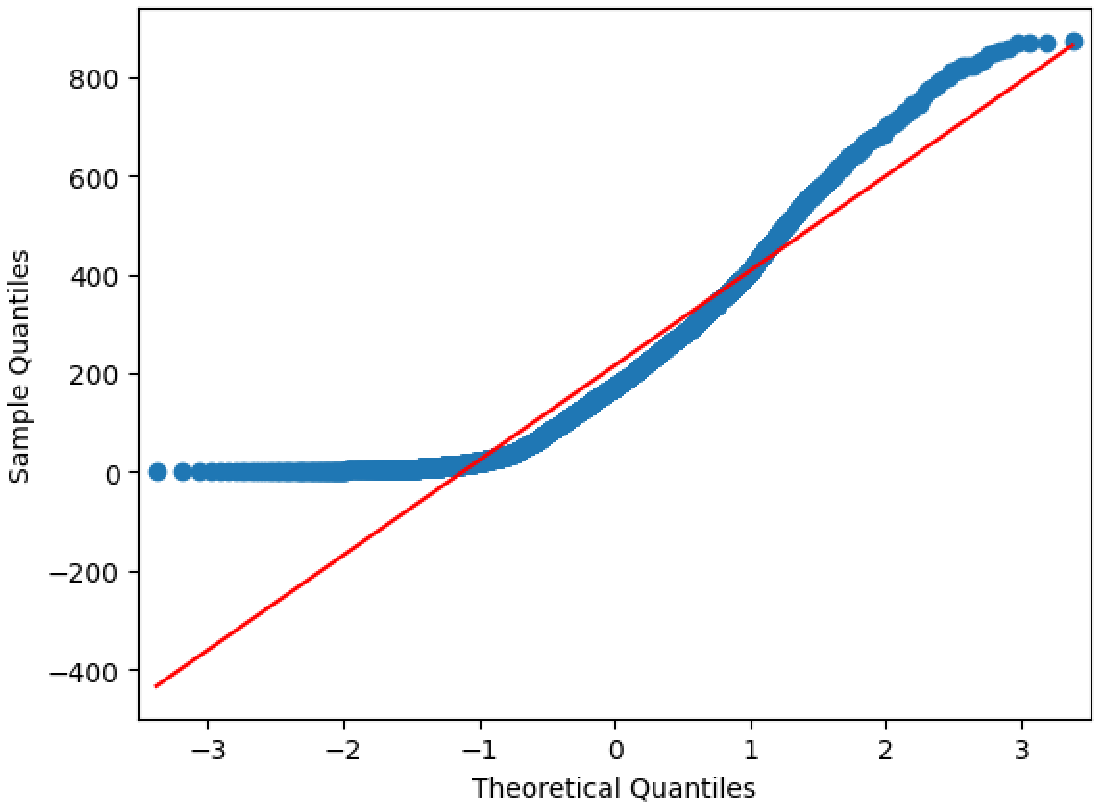
[63]:
```
ShapiroResult(statistic=0.8954644799232483, pvalue=1.1301682309549298e-39)
```

[64]:
```
sm.qqplot(season_spring_demand,line='s')
plt.show()
```

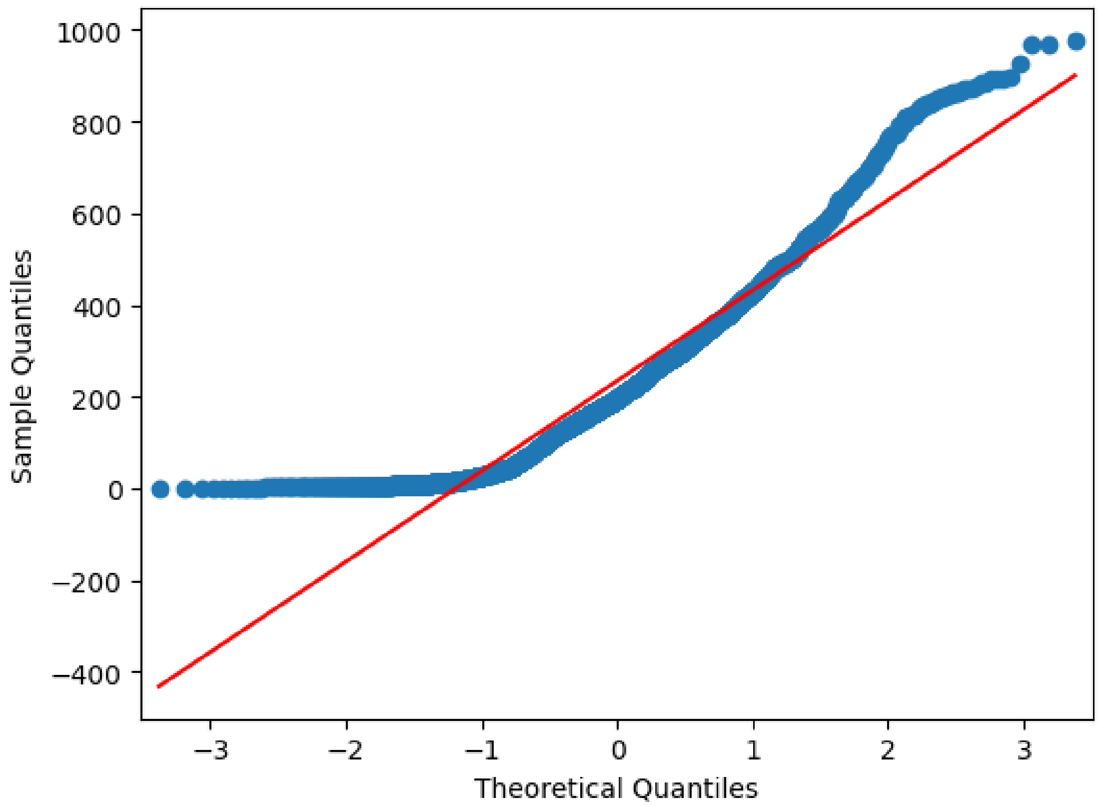


[65]:

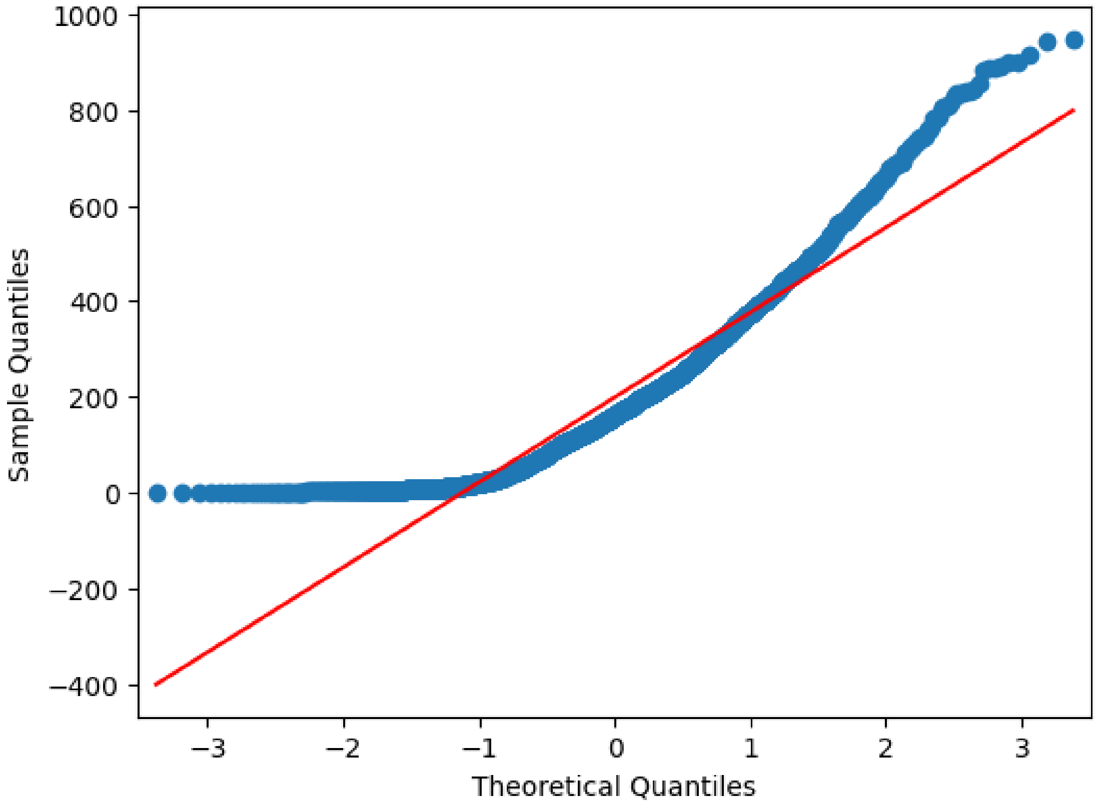
```
sm.qqplot(season_summer_demand,line='s')
plt.show()
```



[66]:
```
sm.qqplot(season_fall_demand,line='s')
plt.show()
```



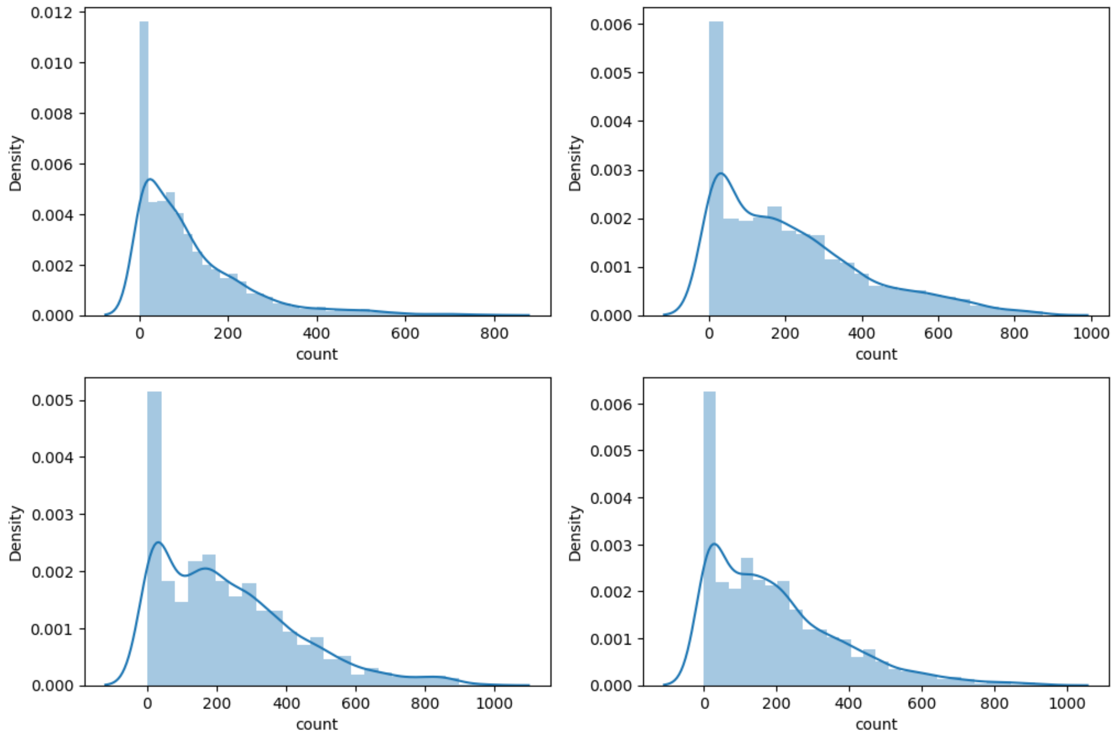
[67]:
```
sm.qqplot(season_winter_demand,line='s')
plt.show()
```



[68]:
```
import warnings
warnings.filterwarnings("ignore")
```

[69]:
```
plt.figure(figsize=(12,8))
plt.subplot(2,2,1)
sns.distplot(season_spring_demand)
plt.subplot(2,2,2)
sns.distplot(season_summer_demand)
plt.subplot(2,2,3)
sns.distplot(season_fall_demand)
plt.subplot(2,2,4)
sns.distplot(season_winter_demand)
```

[69]: <Axes: xlabel='count', ylabel='Density'>



[70]:
```
levene(season_spring_demand,season_summer_demand,season_fall_demand,season_winter_demand)
```

[70]: LeveneResult(statistic=187.7706624026276, pvalue=1.0147116860043298e-118)

Data distribution are not Guassian/ Normal and do have have equal variances. Going ahead with Kruskal Wallis Test

[71]:
```
kruskal(season_spring_demand,season_summer_demand,season_fall_demand,season_winter_demand)
```

[71]: KruskalResult(statistic=699.6668548181988, pvalue=2.479008372608633e-151)

[72]:
```
k_stat,p_val =␣
 ↪kruskal(season_spring_demand,season_summer_demand,season_fall_demand,season_winter_demand)
print("T Statistic Value is: ", k_stat)
print("p_value is: ",p_val)

alpha=0.05

if p_val>alpha:
  print("P is high, so we Fail to Reject the Null Hypothesis H0")
  print("Thus, we say that --> The demand of bicycles on rent is the same for␣
 ↪different Seasons")
else:
  print("P is low, so we Reject the Null Hypothesis H0")
```

```
print("Thus, we say that --> The demand of bicycles on rent is different for␣
↪different Seasons")
```

```
T Statistic Value is:  699.6668548181988
p_value is:  2.479008372608633e-151
P is low, so we Reject the Null Hypothesis H0
Thus, we say that --> The demand of bicycles on rent is different for different
Seasons
```

[93]:
```
#Checking that one way Anova would also give the same results of hypothesis.
f_oneway(season_spring_demand,season_summer_demand,season_fall_demand,season_winter_demand)
```

[93]:
```
F_onewayResult(statistic=236.94671081032106, pvalue=6.164843386499654e-149)
```

Check if the Weather conditions are significantly different during different Seasons?

Null Hypothesis:

H0 : Weather conditions are same during different Seasons

Alternate Hypothesis:

Ha : Weather conditions are significantly different during different Seasons

Test selected : Chi-square test

alpha value = 0.05 (recommended)

[74]:
```
table=pd.crosstab(df['weather'],df['season'])
table
```

[74]:

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

[75]:
```
table_norm=pd.crosstab(df['weather'],df['season'],normalize=True)
table_norm
```

[75]:

season	1	2	3	4
weather				
1	0.161584	0.165442	0.177292	0.156348
2	0.065681	0.065038	0.055484	0.074132
3	0.019383	0.020577	0.018280	0.020669
4	0.000092	0.000000	0.000000	0.000000

[76]:
```
chi2_contingency(table)
```

[76]:
```
Chi2ContingencyResult(statistic=49.15865559689363,
pvalue=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]]))
```

[77]:
```
pval=chi2_contingency(table)[1]
print("p_value is: ",pval)

alpha=0.05

if p_val>alpha:
  print("P is high, so we Fail to Reject the Null Hypothesis H0")
  print("Thus, we say that --> Weather conditions are same during different␣
↪Seasons")
else:
  print("P is low, so we Reject the Null Hypothesis H0")
  print("Thus, we say that --> Weather conditions are different during␣
↪different Seasonss")
```

```
p_value is:  1.5499250736864862e-07
P is low, so we Reject the Null Hypothesis H0
Thus, we say that --> Weather conditions are different during different Seasonss
```

##Insights:

1. The data is given from Timestamp('2011-01-01 00:00:00') to Timestamp('2012-12-19 23:00:00'). The total time period for which the data is given is '718 days 23:00:00'.
2. More bikes are rented during summer and fall seasons
3. Bike retals during holidays are more than weekdays.
4. Weekends and holidays have more customers of bike rentals than weekdays
5. Whenever there is rain, thunderstorm, snow or fog, there were less bikes were rented.
6. Whenever the humidity is less than 20, number of bikes rented is very very low.
7. Whenever the temperature is less than 10, number of bikes rented is less.
8. Whenever the windspeed is greater than 35, number of bikes rented is less.

##Recommendations:

1. Provide offers for casual users and Bring in loyalty points for registered users and encourage them to take Yulu more and spread the word.

2. Social media interaction with Youth given that the price of Petrol and Diesel are sky-rocketing.

3. Maintainance of bikes post heavy weather/ seasons needs to be done. Also, reduce the number of bikes available during heavy rains or thunderstorms to avoid inventory damage or loss.

4. Seasonal Marketing: There is a clear seasonal pattern in the count of rental bikes, Yulu can adjust its marketing strategies accordingly. Focus on promoting bike rentals during the spring and summer months when there is higher demand. Offer seasonal discounts or special packages to attract more customers during these periods.

5. Time-based Pricing: Take advantage of the hourly fluctuation in bike rental counts throughout the day. Consider implementing time-based pricing where rental rates are lower during off-peak hours and higher during peak hours. This can encourage customers to rent bikes during less busy times, balancing out the demand and optimizing the resources.
6. Weather-based Promotions: Recognize the impact of weather on bike rentals. Create weather-based promotions that target customers during clear and cloudy weather, as these conditions show the highest rental counts. Yulu can offer weather-specific discounts to attract more customers during these favorable weather conditions.

[ ]:

```

```