```python
''' 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
'''
```

```
' datetime: datetime\nseason: season (1: spring, 2: summer, 3: fall,
4: winter)\nholiday: whether day is a holiday or not (extracted from
http://dchr.dc.gov/page/holiday-schedule)\nworkingday: if day is
neither weekend nor holiday is 1, otherwise is 0.\nweather:\n    1:
Clear, Few clouds, partly cloudy, partly cloudy\n    2: Mist + Cloudy,
Mist + Broken clouds, Mist + Few clouds, Mist\n    3: Light Snow,
Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered
clouds\n    4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow +
Fog\ntemp: temperature in Celsius\natemp: feeling temperature in
Celsius\nhumidity: humidity\nwindspeed: wind speed\ncasual: count of
casual users\nregistered: count of registered users\ncount: count of
total rental bikes including both casual and registered\n'
```

```python
'''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'''
```

```
'The company wants to know:\n\nWhich variables are significant in
predicting the demand for shared electric cycles in the Indian
market?\nHow well those variables describe the electric cycle demands'
```

```python
import pandas as pd
import numpy as np
import seaborn as sns

df = pd.read_csv('Yulu.csv')
```

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

df.shape

(10886, 12)

df.isnull().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

                 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]
```

```
df.season.value_counts()
```

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

```
df.weather.value_counts()
```

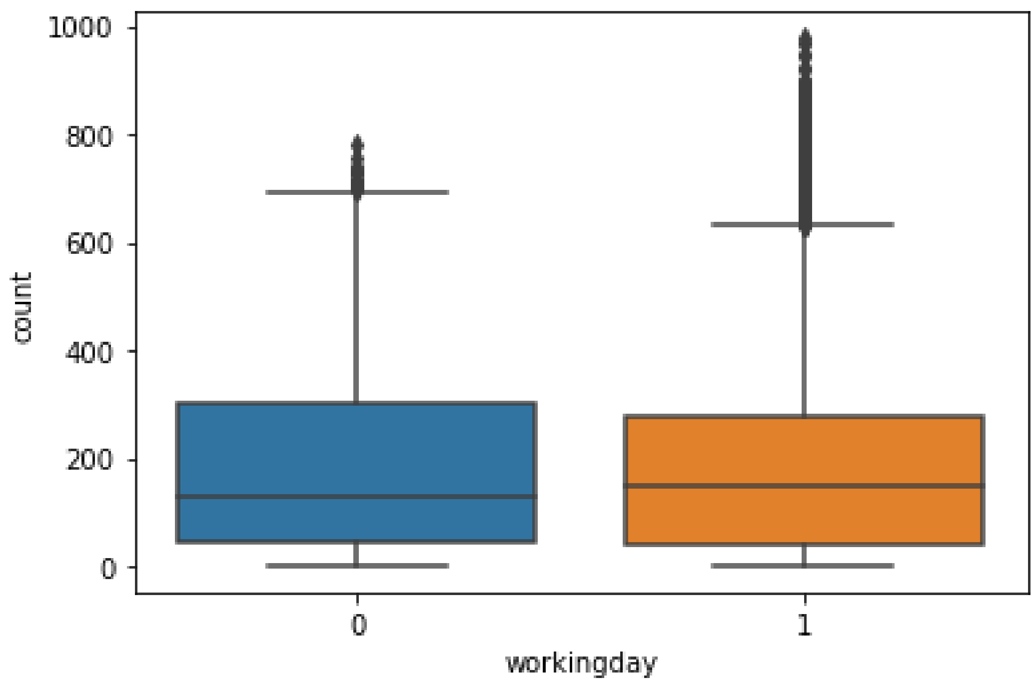
```
1    7192
2    2834
3     859
4       1
Name: weather, dtype: int64
```

```
df.workingday.value_counts()
```

```
1    7412
0    3474
Name: workingday, dtype: int64
```

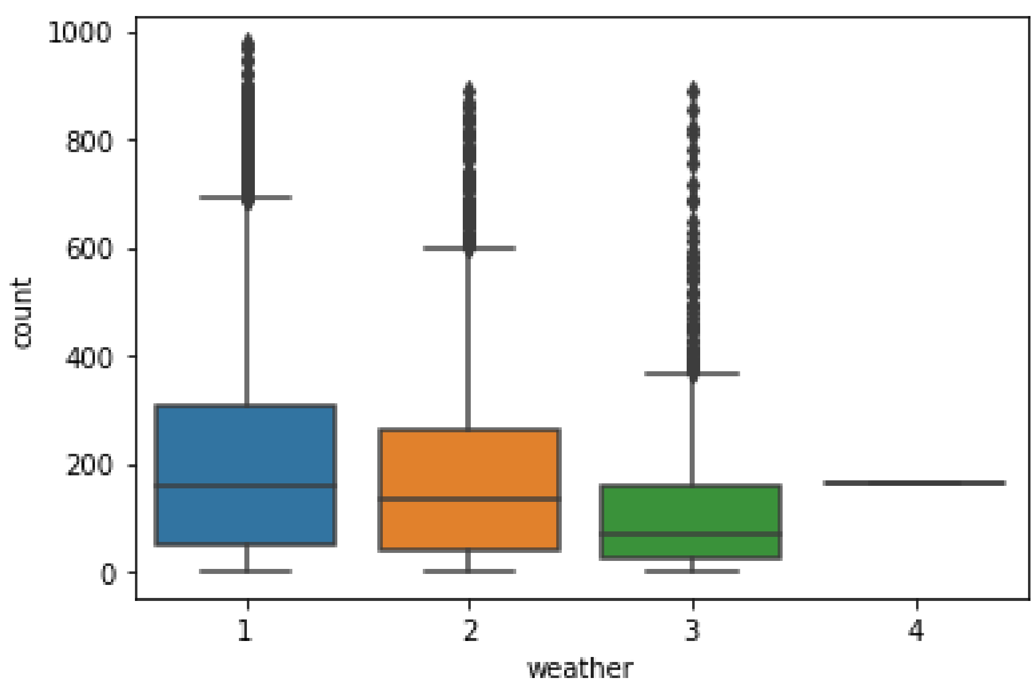
```
sns.boxplot(data=df , x= 'workingday', y = 'count')
```

```
<AxesSubplot:xlabel='workingday', ylabel='count'>
```



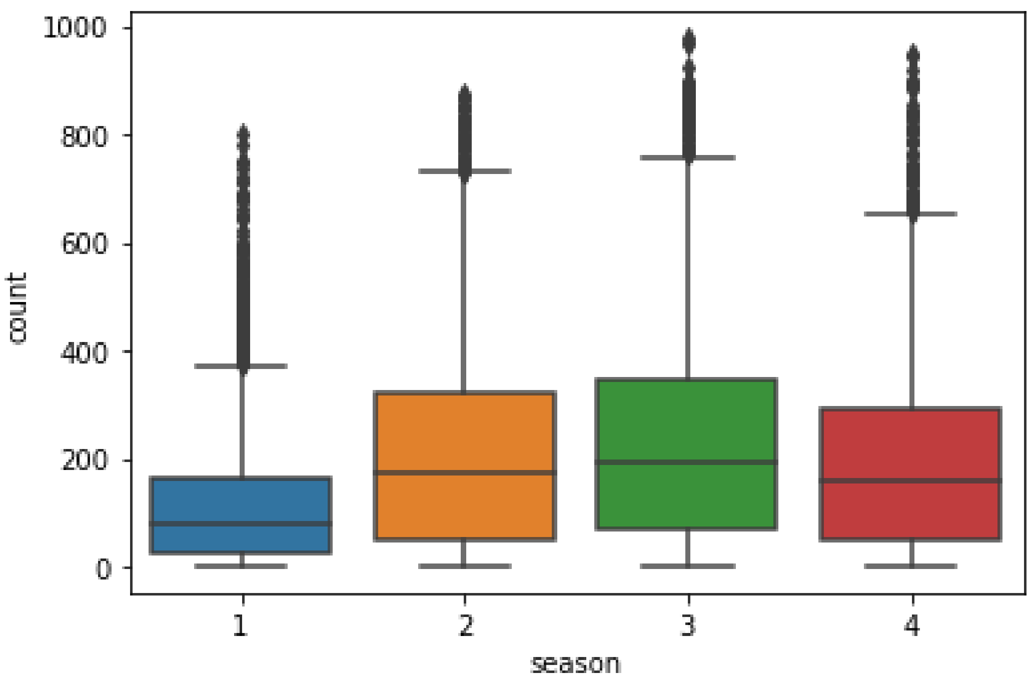

```
sns.boxplot(data=df , x= 'weather', y = 'count')
```

```
<AxesSubplot:xlabel='weather', ylabel='count'>
```

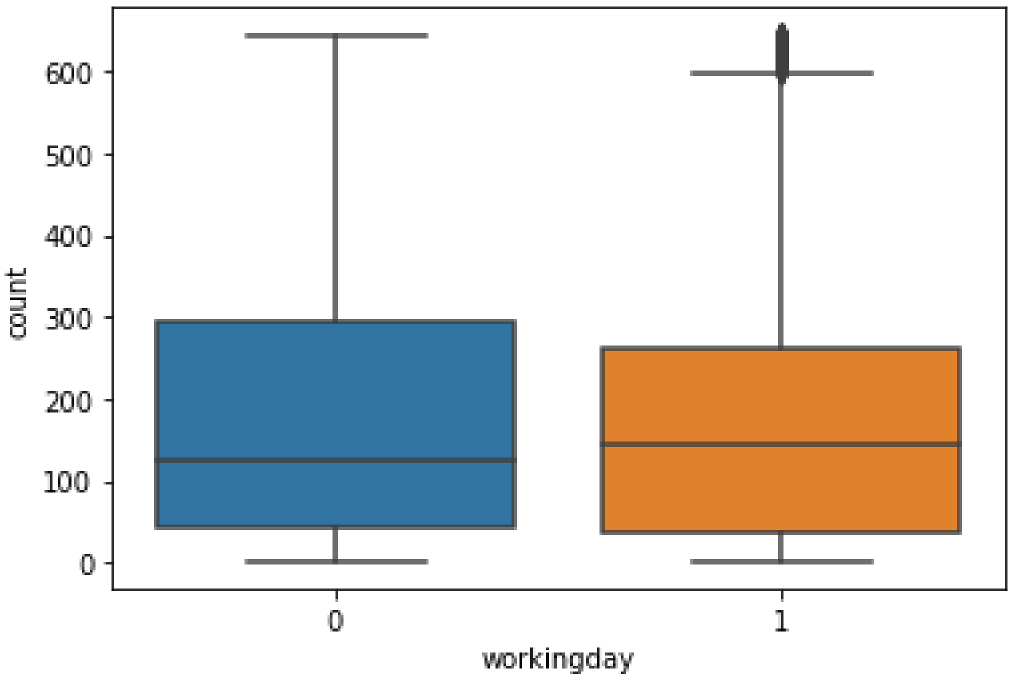



```
sns.boxplot(data=df , x= 'season', y = 'count')
```

```
<AxesSubplot:xlabel='season', ylabel='count'>
```




```
q1 = df['count'].quantile(0.25)
q3 = df['count'].quantile(0.75)
iqr = q3-q1
iqr

242.0

#Removing Outlier
df = df[(df['count']>(q1-1.5*iqr)) & (df['count']<(q3+1.5*iqr))]

df.shape

(10583, 12)

sns.boxplot(data=df , x= 'workingday', y = 'count')

<AxesSubplot:xlabel='workingday', ylabel='count'>
```

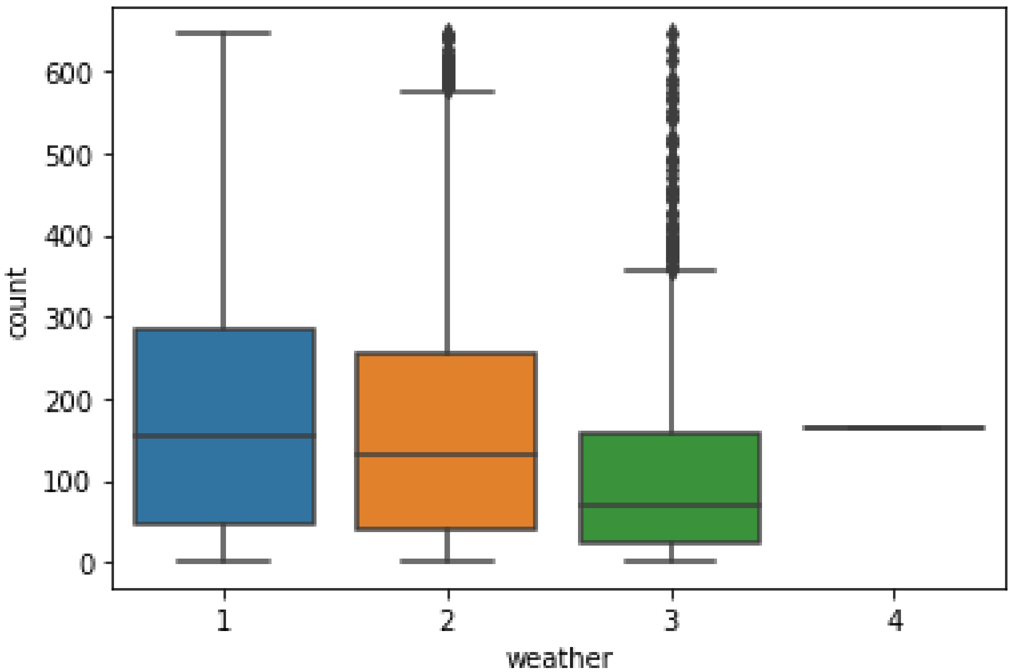


With visual analysis we see that the count doesn't depend much on the working day

Need to check using statistical methods The t test as compared with z test is its advantage for small sample comparison. As n increases, t approaches to z. The advantage of t test disappears, and t distribution simply becomes z distribution. In other words, with large n. t test is just close to z test.

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

```
<AxesSubplot:xlabel='weather', ylabel='count'>
```



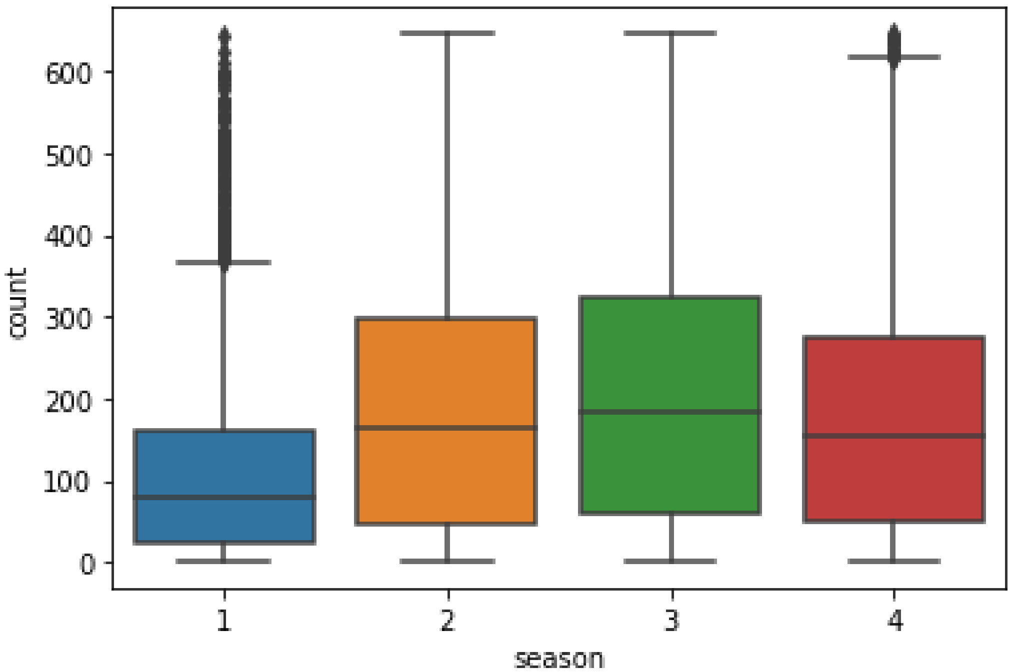
Weather Impact

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

From the above plot we can infer that when the weather is 4 , user dont go for using yulu season 1 being the highest user and weather 2 a little less but weather 3 very less comapred to 1

```
sns.boxplot(data=df , x= 'season', y = 'count')
```

```
<AxesSubplot:xlabel='season', ylabel='count'>
```



Season Impact

season: season (1: spring, 2: summer, 3: fall, 4: winter) from he above plot we acn infer that season 1 sees less count of yulu rides compared to the rest

Step 1: Define the null and alternate hypotheses

H0: The count on weekday is LESS THAN or equal to the count on weekend. Ha : The count on weekday is greater than count on weekend.

Step 2: Select Appropriate test

This is a one-tailed test concerning two population means from two independent populations. As the population standard deviations are unknown, the two sample independent t-test will be the appropriate test for this problem.

Step 3: Decide the significance level

As given in the problem statement, we select α = 0.05.

Step 4: Collect and prepare data

```
weekday = df[df['workingday'] == 1]['count'].sample(3422)
weekend = df[df['workingday'] == 0]['count'].sample(3422)

print('The sample standard deviation of the count on weekday is:',
round(weekday.std(),2))
print('The sample standard deviation of the count on weekend is:',
round(weekend.std(),2))
```

```
The sample standard deviation of the count on weekday is: 150.8
The sample standard deviation of the count on weekend is: 163.78
```

Step 5: Calculate the p-value

```
# import the required function
from scipy.stats import ttest_ind
# find the p-value
test_stat, p_value = ttest_ind(weekday, weekend, equal_var = False,
alternative = 'greater')
print('The p-value is', p_value)
```

```
The p-value is 0.9955488213007847
```

```
# print the conclusion based on p-value
if p_value < 0.05:
 print('As the p-value is less than the level of significance, we
reject the null hypothesis')
else:
 print('As the p-value is greater than the level of significance, we
fail to reject the null hypothesis')
```

```
As the p-value is greater than the level of significance, we fail to
reject the null hypothesis
```

Which means The count on weekday is LESS THAN or equal to the count on weekend.

Is the demand of electric cycles same for different weather?

```
df.weather.value_counts()
```

```
1    6962
2    2770
3     850
4       1
Name: weather, dtype: int64
```

```
df=df[~(df['weather']==4)]
```

```
w1 = df[df['weather'] == 1]['count'].sample(850)
w2= df[df['weather'] == 2]['count'].sample(850)
w3 = df[df['weather'] == 3]['count'].sample(850)
```

```
df.groupby(['weather'])['count'].describe()
```

```
          count        mean         std  min   25%    50%    75%
max
weather

1        6962.0  187.131140  161.333785  1.0  45.0  153.0  286.0
646.0
```

```
2         2770.0  166.117690  146.992422  1.0  39.0  130.0  254.0
646.0
3          850.0  111.862353  121.233389  1.0  23.0   70.5  157.0
646.0
```

Step 1: Define the null and alternate hypotheses

H0:The mean count in different weather are equal. Ha:The mean count in different weather are different.

Step 2: Select Appropriate test

This is a problem, concerning three population means. One-way ANOVA could be the appropriate test here provided normality and equality of variance assumptions are verified. For testing of normality, Shapiro-Wilk's test is applied to the response variable. For equality of variance, Levene test is applied to the response variable.

Shapiro-Wilk's test

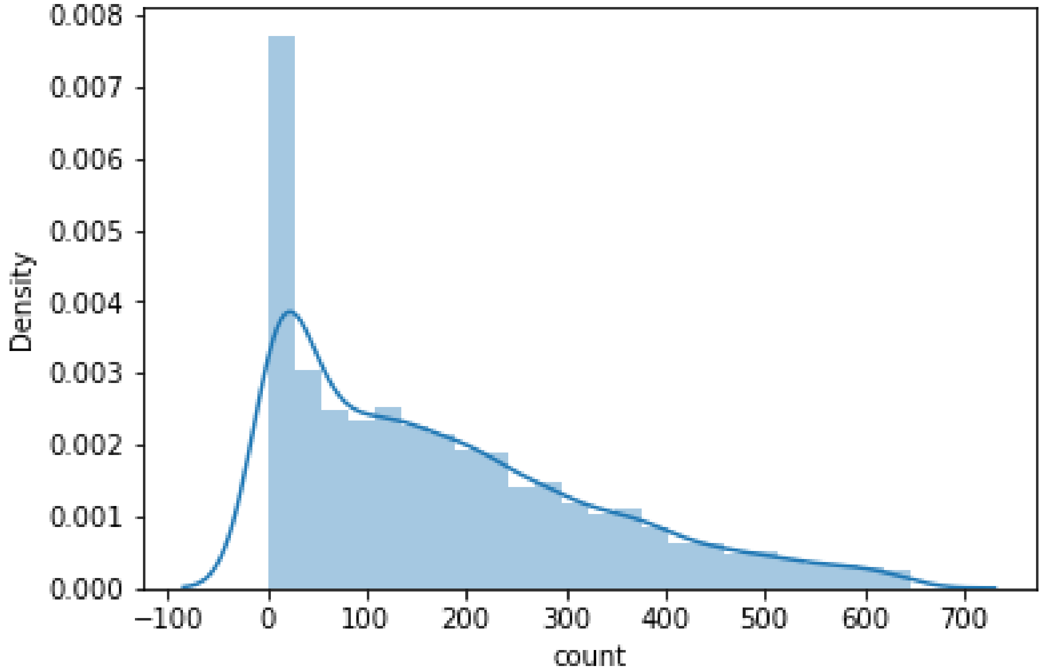
We will test the null hypothesis

H0: Count follows normal distribution Ha: Count doesn't follow normal distribution

```
# Assumption 1: Normality
# import the required function
from scipy.stats import shapiro
# find the p-value
w, p_value = shapiro(df['count'].sample(4999))
print('The p-value is', p_value)

The p-value is 0.0
```

as p<0.05 that ,eans we reject the null hypothesis which means the distribution does not follow normal distribution

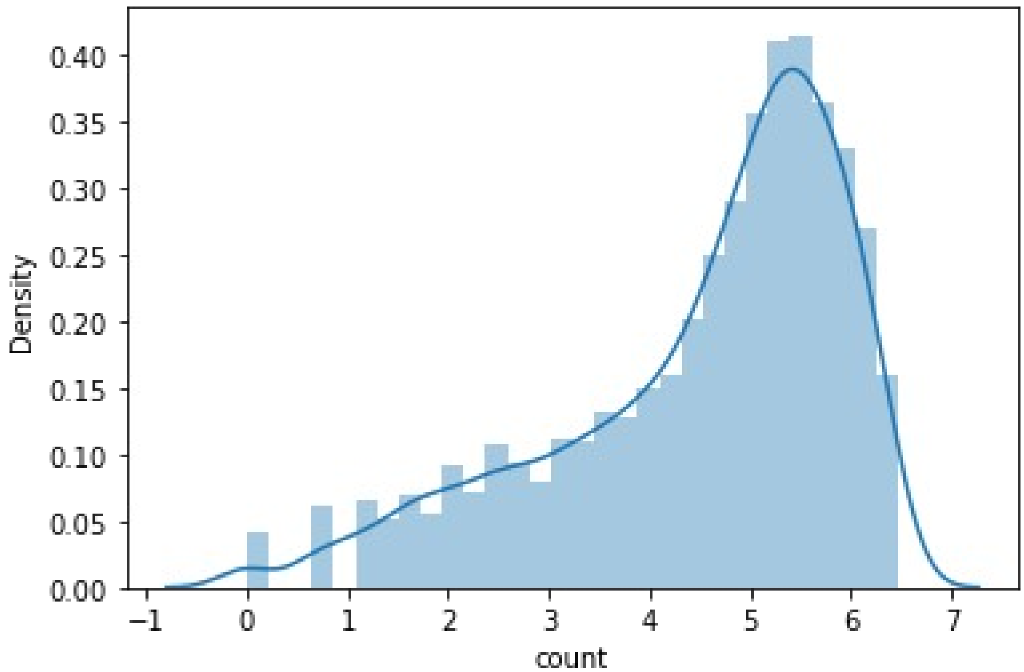
```
sns.distplot(df['count'].sample(4999))

/opt/anaconda3/lib/python3.8/site-packages/seaborn/
distributions.py:2551: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)

<AxesSubplot:xlabel='count', ylabel='Density'>
```




```
#as the distribution is not normal we try to do a log normal
distribution and try to make it a normal distribution
import numpy as np
sns.distplot(np.log(df['count'].sample(4999)))
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/
distributions.py:2551: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<AxesSubplot:xlabel='count', ylabel='Density'>
```



Still it doesnot follow the normal distribution but we can still go for a anova test

Levene’s test

We will test the null hypothesis H0:All the count variances are equal Ha:At least one variance is different from the rest

```
#Assumption 2: Homogeneity of Variance
#import the required function
from scipy.stats import levene
statistic, p_value = levene( w1,
 w2,
w3)
# find the p-value
print('The p-value is', p_value)
```

```
The p-value is 8.960356701049056e-20
```

```
p_value<0.05
```

```
True
```

That means we can accept the null hypothesis , so the variance test is passed

```
print(w1.var(), w2.var(), w3.var())
```

```
26227.263875840068 21685.391389177563 14697.534623432406
```

ANOVA

```
# import the required function
from scipy.stats import f_oneway
# find the p-value
test_stat, p_value = f_oneway(w1,w2,w3)
# print the p-value
print('The p-value is', p_value)
```

```
The p-value is 1.0991993365274465e-27
```

As the p-value 1.457244731807399e-25 is less than the level of significance, we

reject the null hypothesis. which means weather has effect on the bicycle count

Recommendations

we can observe that during rains and bad weather there is a less count we can give offers during those times or we can move the bicycles to nearby places

```
df.season.value_counts()
```

```
1    2669
4    2664
2    2633
3    2616
Name: season, dtype: int64
```

```
s1 = df[df['season'] == 1]['count'].sample(2000)
s2= df[df['season'] == 2]['count'].sample(2000)
s3 = df[df['season'] == 3]['count'].sample(2000)
s4 = df[df['season'] == 4]['count'].sample(2000)
```

```
df.groupby(['season'])['count'].describe()
```

```
         count        mean         std  min    25%    50%     75%
max
season

1       2669.0  112.775946  116.902627  1.0  24.00   78.0  161.00
644.0
2       2633.0  195.653627  166.170802  1.0  45.00  165.0  299.00
646.0
3       2616.0  210.484327  164.055532  1.0  59.75  185.0  323.25
646.0
4       2664.0  184.404655  154.563069  1.0  48.75  154.0  276.25
646.0
```

```python
# import the required function
from scipy.stats import f_oneway
# find the p-value
test_stat, p_value = f_oneway(s1,s2,s3,s4)
# print the p-value
print('The p-value is', p_value)
```

```
The p-value is 1.3598773512803993e-97
```

As the p-value 1.3598773512803993e-97 is less than the level of significance, we

reject the null hypothesis. which means season has effect on the bicycle count

Check Weather is dependent on season (check between 2 predictor variable)

```python
df1=pd.crosstab(df['weather'],df['season'])

from scipy.stats import chi2_contingency

chi2, p, dof, expected = chi2_contingency(df1)

p
```

```
6.75312212866461e-08
```

p<0.05 reject the null hypothesis. which means weather has effect on season or viceversa