Problem :

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 their 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 American market.

Objective 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

In [34]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [35]:
```
#Reading the data set
df=pd.read_csv('bike_sharing.csv')
```

In [36]:
```
df.head()
```

Out[36]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspee
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0

Column Profiling:

- datetime: datetime
- season: season (1: spring, 2: summer, 3: fall, 4: winter)
- holiday : whether day is a holiday or not (extracted from ttp://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
  - 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 Pellets + 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

Exploration

In [37]:

```
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
```

In [38]:
```
#Checking null
df.isnull().sum()
```

Out[38]:

	0
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

In [39]:
```
#checking duplicates
df[df.duplicated()]
```

Out[39]:

datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed

In [40]:
```
def check(df,col):
  print(f"Unique values:{df[col].unique()}")
```

```
  print(f"Value counts: {df[col].value_counts()}")

col_list=['season','holiday','workingday','weather']

for col in col_list:
  print(col)
  check(df,col)
  print('-'*50)
```

```
season
Unique values:[1 2 3 4]
Value counts: season
4    2734
2    2733
3    2733
1    2686
Name: count, dtype: int64
--------------------------------------------------
holiday
Unique values:[0 1]
Value counts: holiday
0    10575
1      311
Name: count, dtype: int64
--------------------------------------------------
workingday
Unique values:[0 1]
Value counts: workingday
1    7412
0    3474
Name: count, dtype: int64
--------------------------------------------------
weather
Unique values:[1 2 3 4]
Value counts: weather
1    7192
2    2834
3     859
4       1
Name: count, dtype: int64
--------------------------------------------------
```

Correlation and heatmap

In [41]:
```
#correlation
df.select_dtypes(include=np.number).corr()
```

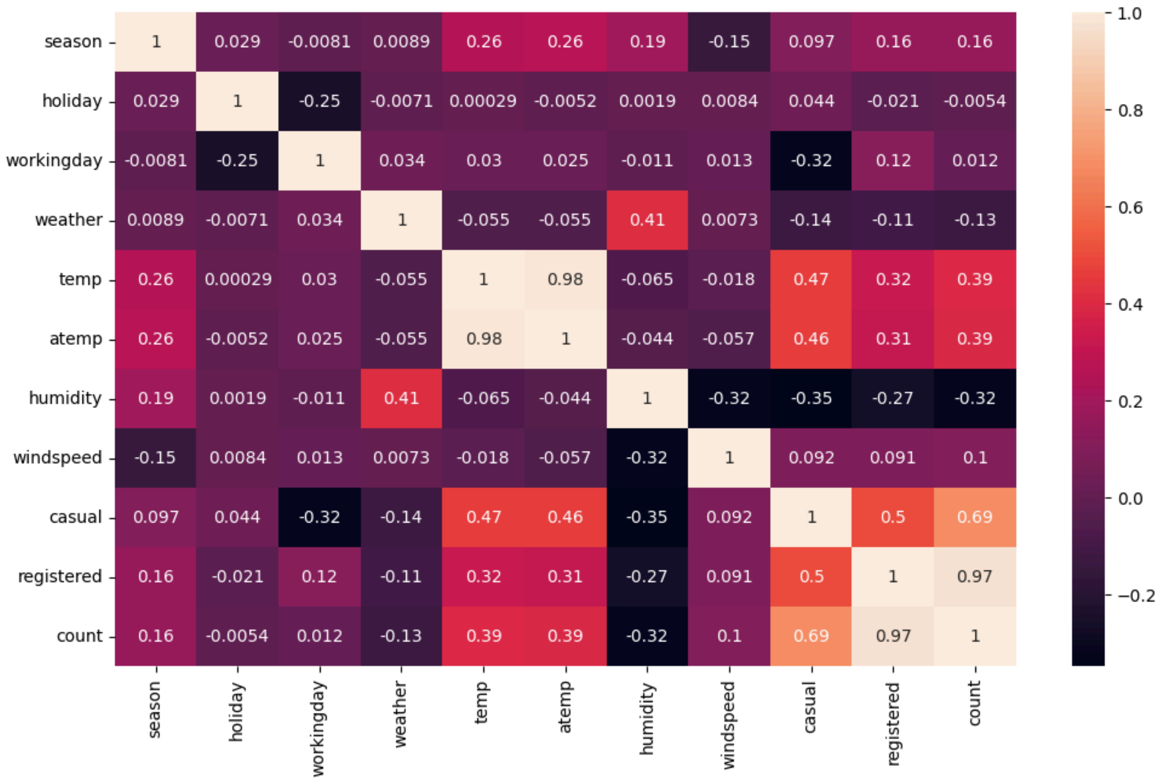
Out[41]:

	season	holiday	workingday	weather	temp	atemp	humidit
season	1.000000	0.029368	-0.008126	0.008879	0.258689	0.264744	0.19061
holiday	0.029368	1.000000	-0.250491	-0.007074	0.000295	-0.005215	0.00192
workingday	-0.008126	-0.250491	1.000000	0.033772	0.029966	0.024660	-0.01088
weather	0.008879	-0.007074	0.033772	1.000000	-0.055035	-0.055376	0.40624
temp	0.258689	0.000295	0.029966	-0.055035	1.000000	0.984948	-0.06494
atemp	0.264744	-0.005215	0.024660	-0.055376	0.984948	1.000000	-0.04353
humidity	0.190610	0.001929	-0.010880	0.406244	-0.064949	-0.043536	1.00000
windspeed	-0.147121	0.008409	0.013373	0.007261	-0.017852	-0.057473	-0.31860
casual	0.096758	0.043799	-0.319111	-0.135918	0.467097	0.462067	-0.34818
registered	0.164011	-0.020956	0.119460	-0.109340	0.318571	0.314635	-0.26545
count	0.163439	-0.005393	0.011594	-0.128655	0.394454	0.389784	-0.31737

In [42]:

```
plt.figure(figsize=(12, 7))
sns.heatmap(df.select_dtypes(include=np.number).corr(),annot=True)
```

Out[42]: <Axes: >



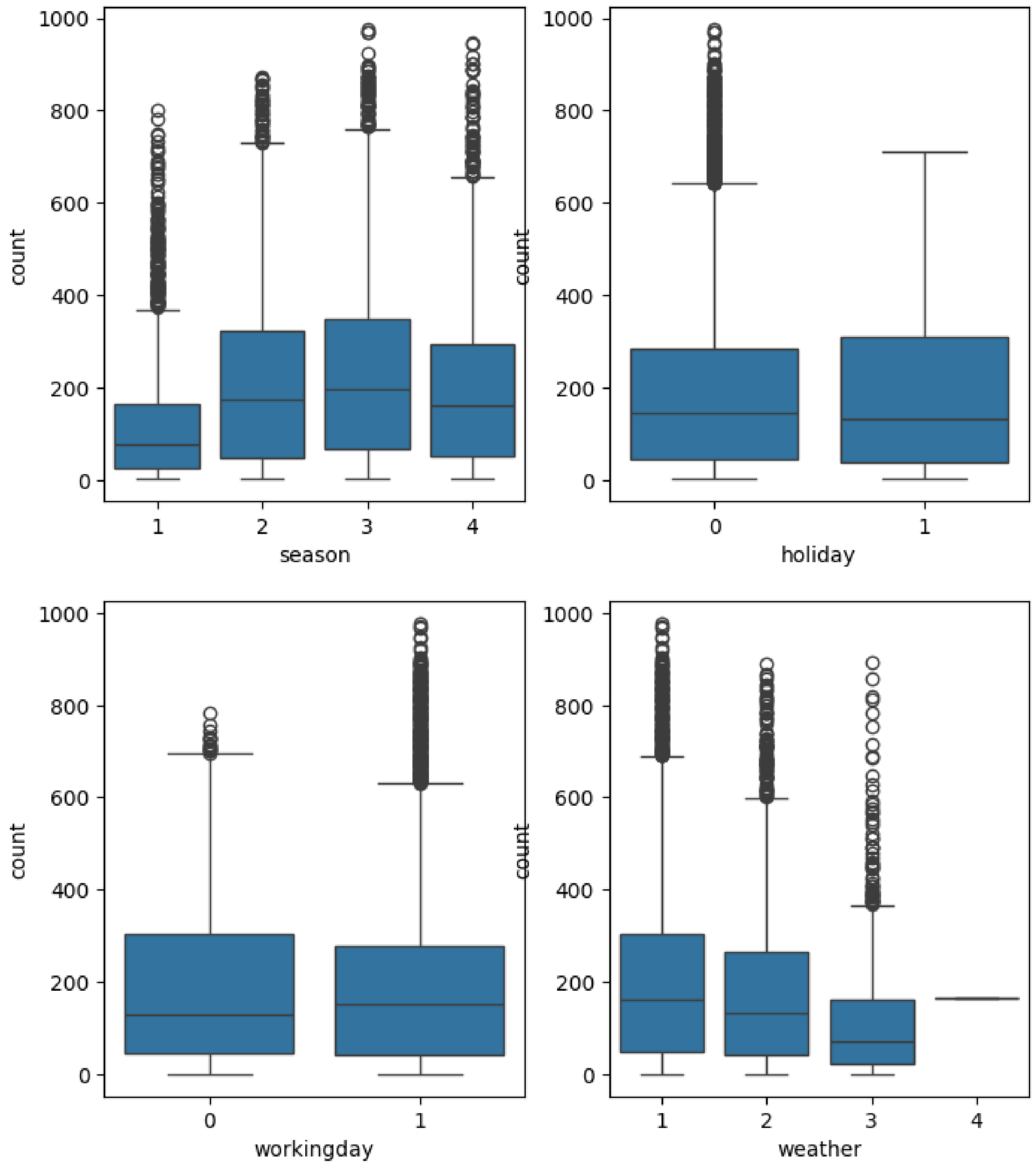
In [43]:

```
#Droping the highly correlated columns
df.drop(['atemp','casual','registered'],axis=1,inplace=True)
```

In [44]:

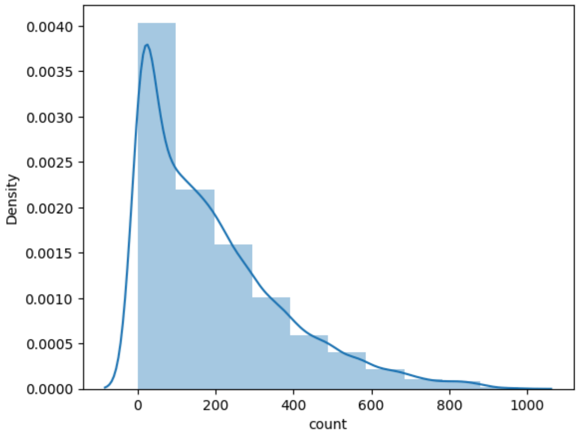
```
# Outlier detection
plt.figure(figsize=(8,25))
```

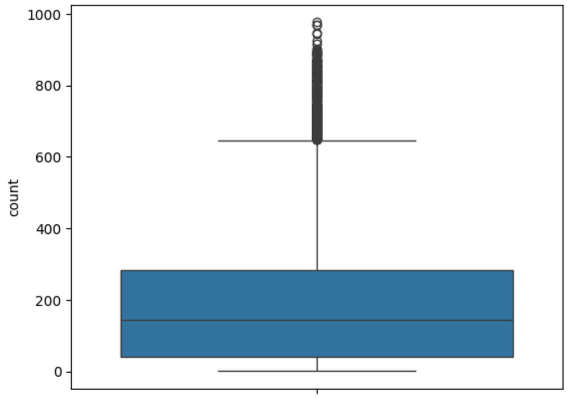
```
for plot in range(1,len(col_list)+1):
  plt.subplot(5,2,plot)
  sns.boxplot(x=df[col_list[plot-1]],y=df['count'])
plt.show()
```



In [45]:
```
# Distribution
plt.figure(figsize=(14,5))

#Histogram
plt.subplot(1,2,1)
sns.distplot(df['count'],bins=10)

#boxplot
plt.subplot(1,2,2)
sns.boxplot(y=df['count'])

plt.show()
```

```
/tmp/ipython-input-45-2107795470.py:6: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['count'],bins=10)
```



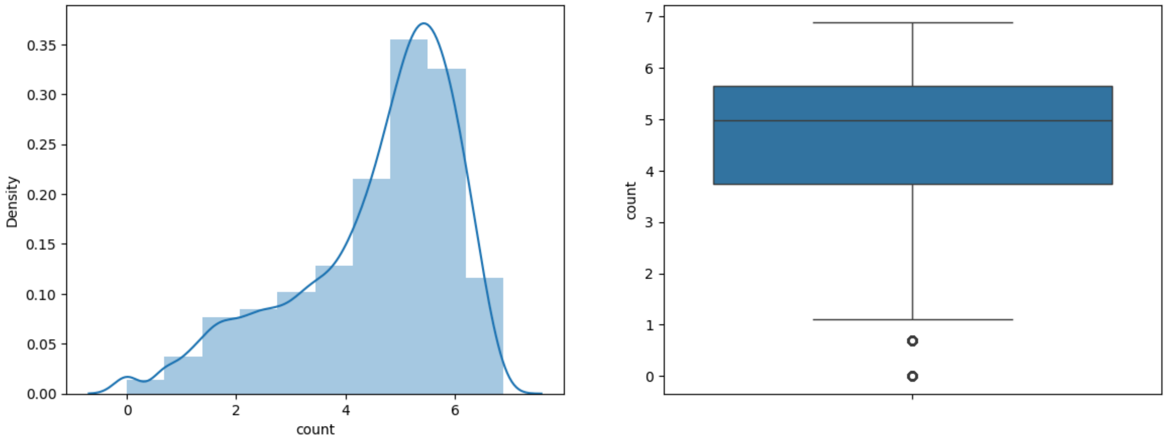


Dealing with outliers

1. Drop: Drop data can be harmful at times, without understanind the business context. Drop column outliers using IQR.
2. Treat: Transformating this data (log normal transformation).
3. Leave as it is.

In [46]:
```
#Distribution after applying log

plt.figure(figsize=(14,5))

#Histogram
plt.subplot(1,2,1)
sns.distplot(np.log(df['count']),bins=10)

#boxplot
plt.subplot(1,2,2)
sns.boxplot(y=np.log(df['count']))

plt.show()
```

```
/tmp/ipython-input-46-548753380.py:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(np.log(df['count']),bins=10)
```



Analysis

In [47]:
```
# 1. Workingday -
pd.DataFrame(df.groupby('workingday')['count'].describe())
```

Out[47]:

workingday	count	mean	std	min	25%	50%	75%	max
0	3474.0	188.506621	173.724015	1.0	44.0	128.0	304.0	783.0
1	7412.0	193.011873	184.513659	1.0	41.0	151.0	277.0	977.0

In [48]:
```
# 2. Holiday -
pd.DataFrame(df.groupby('holiday')['count'].describe())
```

Out[48]:

holiday	count	mean	std	min	25%	50%	75%	max
0	10575.0	191.741655	181.513131	1.0	43.0	145.0	283.0	977.0
1	311.0	185.877814	168.300531	1.0	38.5	133.0	308.0	712.0

In [49]:
```
# 3. Season -
pd.DataFrame(df.groupby('season')['count'].describe())
```

Out[49]:

season	count	mean	std	min	25%	50%	75%	max
1	2686.0	116.343261	125.273974	1.0	24.0	78.0	164.0	801.0
2	2733.0	215.251372	192.007843	1.0	49.0	172.0	321.0	873.0
3	2733.0	234.417124	197.151001	1.0	68.0	195.0	347.0	977.0
4	2734.0	198.988296	177.622409	1.0	51.0	161.0	294.0	948.0

In [50]:
```
# 4. Weather -
pd.DataFrame(df.groupby('weather')['count'].describe())
```

Out[50]:

	count	mean	std	min	25%	50%	75%	max
weather								
1	7192.0	205.236791	187.959566	1.0	48.0	161.0	305.0	977.0
2	2834.0	178.955540	168.366413	1.0	41.0	134.0	264.0	890.0
3	859.0	118.846333	138.581297	1.0	23.0	71.0	161.0	891.0
4	1.0	164.000000	NaN	164.0	164.0	164.0	164.0	164.0

Hypothises testing

In [51]:
```
def result(p_value, alpha):
  if p_value < alpha:
    print(f'As the p-value {p_value} is less than the level of significance, we
  else:
    print(f'As the p-value {p_value} is greater than the level of significance,
```

Is there any significant difference between the no. of bike rides on weekdays and weekends?

H_0 : The demand of bikes on weekdays is greater or similar to the demand of bikes on weekend.

H_a : The demand of bikes on weekdays is less than the demand of bikes on weekend.

Let μ_1 and μ_2 be the average no. of bikes rented on weekdays and weekends respectively.

Mathematically, the above formulated hypothesis can be written as:

$H_0 : \mu_1 >= \mu_2$

$H_a : \mu_1 < \mu_2$

In [52]:
```
alpha = 0.05
```

In [53]:
```
#Ttest because 2 samples
weekday=df[df['workingday']==1]['count'].sample(2999)
weekend=df[df['workingday']==0]['count'].sample(2999)
```

In [54]:
```
from scipy.stats import ttest_ind
```

In [55]:
```
t_value,p_value= ttest_ind(weekday,weekend,equal_var=False,alternative='less')

print('P_value:',p_value)
```

```
P_value: 0.4269189958888099
```

In [56]:
```
result(p_value, alpha)
```

```
As the p-value 0.4269189958888099 is greater than the level of significance, we f
ail to reject the null hypothesis.
```

Is the demand of bicycles on rent same for different weather conditions?

In [57]:
```
df=df[~(df['weather']==4)]
```

H_0 : The average no. of bike rides in different weather conditions are equal.

H_a : The average no. of bike rides in different weather conditions are not equal.

In [58]:
```
w1=df[df['weather']==1]['count'].sample(750)
w2=df[df['weather']==2]['count'].sample(750)
w3=df[df['weather']==2]['count'].sample(750)
```

In [59]:
```
df.groupby(['weather'])['count'].describe()
```

Out[59]:

weather	count	mean	std	min	25%	50%	75%	max
1	7192.0	205.236791	187.959566	1.0	48.0	161.0	305.0	977.0
2	2834.0	178.955540	168.366413	1.0	41.0	134.0	264.0	890.0
3	859.0	118.846333	138.581297	1.0	23.0	71.0	161.0	891.0

The ANOVA test has important assumptions that must be satisfied in order for the associated p-value to be valid.

- The samples are independent.
- Each sample is from a normally distributed population.
- The population variance of the groups are all equal.

Now, we will be using the following statistical tests to check the normality and euality of variance of the data set -

- 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.

In [62]:
```
from scipy.stats import shapiro # Shapiro-Wilk's test for Normality
```

> H_0 : Count follows normal distribution

against the alternative hypothesis

> H_a : Count doesn't follow normal distribution

In [63]:
```
# Assumption 1: Normality
```

```
w, p_value = shapiro(df['count'].sample(4999))
print('The p-value is : ', p_value)

result(p_value, alpha)
```

```
The p-value is :  2.341616484952168e-53
As the p-value 2.341616484952168e-53 is less than the level of significance, we r
eject the null hypothesis.
```

In [64]:
```
from scipy.stats import levene
```

Levene's test -

We will test the null hypothesis

> H_0: All the count variances are equal

against the alternative hypothesis

> H_a: At least one variance is different from the rest

In [68]:
```
#Assumption 2: Homogeneity of Variance

stat, p_value = levene(w1, w2, w3)
print('The p-value is : ', p_value)

result(p_value, alpha)
```

```
The p-value is :  0.03255021124728185
As the p-value 0.03255021124728185 is less than the level of significance, we rej
ect the null hypothesis.
```

In [66]:
```
from scipy.stats import f_oneway
```

In [67]:
```
test_stas ,p_value=f_oneway(w1,w2,w3)

print('P_value:',p_value)

result(p_value,alpha)
```

```
P_value: 0.004822962178434753
As the p-value 0.004822962178434753 is less than the level of significance, we re
ject the null hypothesis.
```

Insights from hypothesis testing -

1. The no. of bikes rented on weekdays is comparatively higher than on weekends.
2. The no. of bikes rented on regular days is comparatively higher than on holidays.
3. The demand of bicycles on rent differs under different weather conditions.
4. The demand of bicycles on rent is different during different seasons.
5. The weather conditions are surely dependent upon the ongoing season.

Miscellaneous observations -

The distribution of 'count' column wasn't actually normal or near normal. Infact the column's distribution is found to be a bit skewed towards right.

Generic recommendations -

- The demand of bikes on rent are usually higher during Weekdays.
- The demand of bikes on rent are usually higher during Regular days.
- The chances of person renting a bike are usually higher during Season 3.
- The chances of person renting a bike are usually higher during Weather condition 1.

We recommend the company to maintain the bike stocks accordingly.