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

In [95]:

```
# importing the required packages

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from IPython.display import display, Markdown
```

```
import warnings
warnings.filterwarnings('ignore')
```

In [96]:

```
df = pd.read_csv('C:/Users/pshashank3/Desktop/Data Science/Scaler/Datasets/Projects/yulu/bike_sharing.csv')
df.head()
```

Out[96]:

	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

In [97]:

```
display(Markdown("#### Shape of the data:"))
# no of rows amd columns in dataset
print(f"# rows: {df.shape[0]} \n# columns: {df.shape[1]}")
```

Shape of the data:

```
# rows: 10886
# columns: 12
```

In [98]:

```
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 [99]:

```
df.describe(include = 'all').T
```

Out[99]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
datetime	10886	10886	2011-08-09 09:00:00	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN
season	10886	NaN	NaN	NaN	2.50661	1.11617	1	2	3	4	4
holiday	10886	NaN	NaN	NaN	0.0285688	0.166599	0	0	0	0	1
workingday	10886	NaN	NaN	NaN	0.680875	0.466159	0	0	1	1	1
weather	10886	NaN	NaN	NaN	1.41843	0.633839	1	1	1	2	4
temp	10886	NaN	NaN	NaN	20.2309	7.79159	0.82	13.94	20.5	26.24	41
atemp	10886	NaN	NaN	NaN	23.6551	8.4746	0.76	16.665	24.24	31.06	45.455
humidity	10886	NaN	NaN	NaN	61.8865	19.245	0	47	62	77	100
windspeed	10886	NaN	NaN	NaN	12.7994	8.16454	0	7.0015	12.998	16.9979	56.9969
casual	10886	NaN	NaN	NaN	36.022	49.9605	0	4	17	49	367
registered	10886	NaN	NaN	NaN	155.552	151.039	0	36	118	222	886
count	10886	NaN	NaN	NaN	191.574	181.144	1	42	145	284	977

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

Datatype of following attributes needs to changed to proper data type

- **datetime** - to datetime
- **season** - to categorical
- **holiday** - to categorical
- **workingday** - to categorical
- **weather** - to categorical

As these got parsed and object and int/float datatype which we can override it.

In [100…
```
df['datetime'] = pd.to_datetime(df['datetime'])


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

In [101…
```
# detecting missing values in the dataset
df.isnull().sum()
```

Out[101…
```
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
```

There are no missing values present in the dataset.

Univariate Analysis

In [102…
```
# minimum datetime and maximum datetime
```

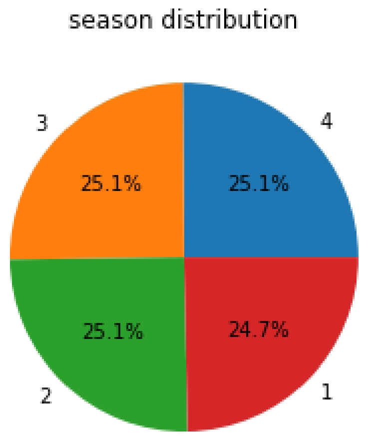
```
df['datetime'].min(), df['datetime'].max()
```

Out[102…
```
(Timestamp('2011-01-01 00:00:00'), Timestamp('2012-12-19 23:00:00'))
```

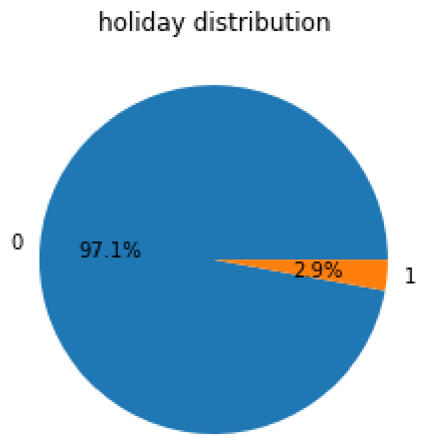
In [103…
```
# Analysis of each categorical column
display(Markdown("### Value Counts/pie plots of each feature:"))
for col in cat_cols:
    display(df[col].value_counts().to_frame())
    data = df[col].value_counts().values.tolist()
    lbls = df[col].value_counts().index.tolist()
    plt.pie(data, labels = lbls, autopct='%1.1f%%')
    plt.title(col + ' distribution')
    plt.show()
```

Value Counts/pie plots of each feature:

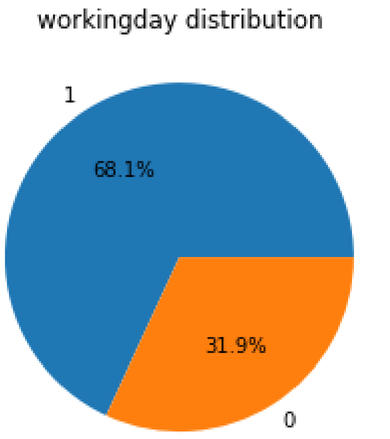
	season
4	2734
3	2733
2	2733
1	2686



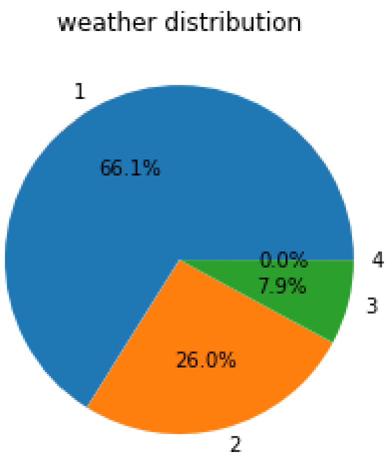
	holiday
0	10575
1	311



	workingday
1	7412
0	3474



	weather
1	7192
2	2834
3	859
4	1

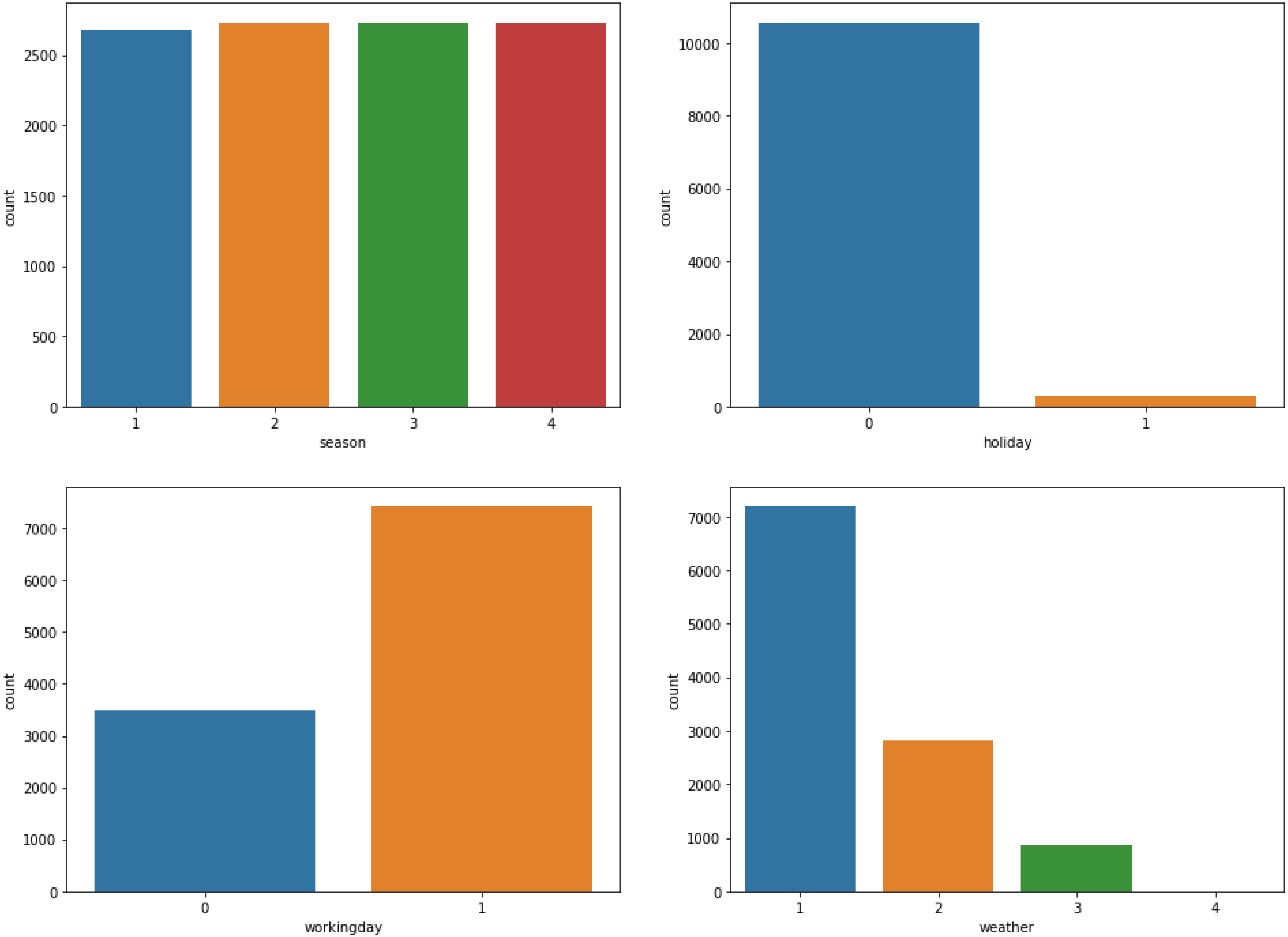


In [104…

```
display(Markdown("### Count plots of each feature:"))
```

```
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(16, 12))

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()
```

Count plots of each feature:


count
2500
2000
1500
1000
500
0
1 2 3 4
season
count
10000
8000
6000
4000
2000
0
0 1
holiday
count
7000
6000
5000
4000
3000
2000
1000
0
0 1
workingday
count
7000
6000
5000
4000
3000
2000
1000
0
1 2 3 4
weather

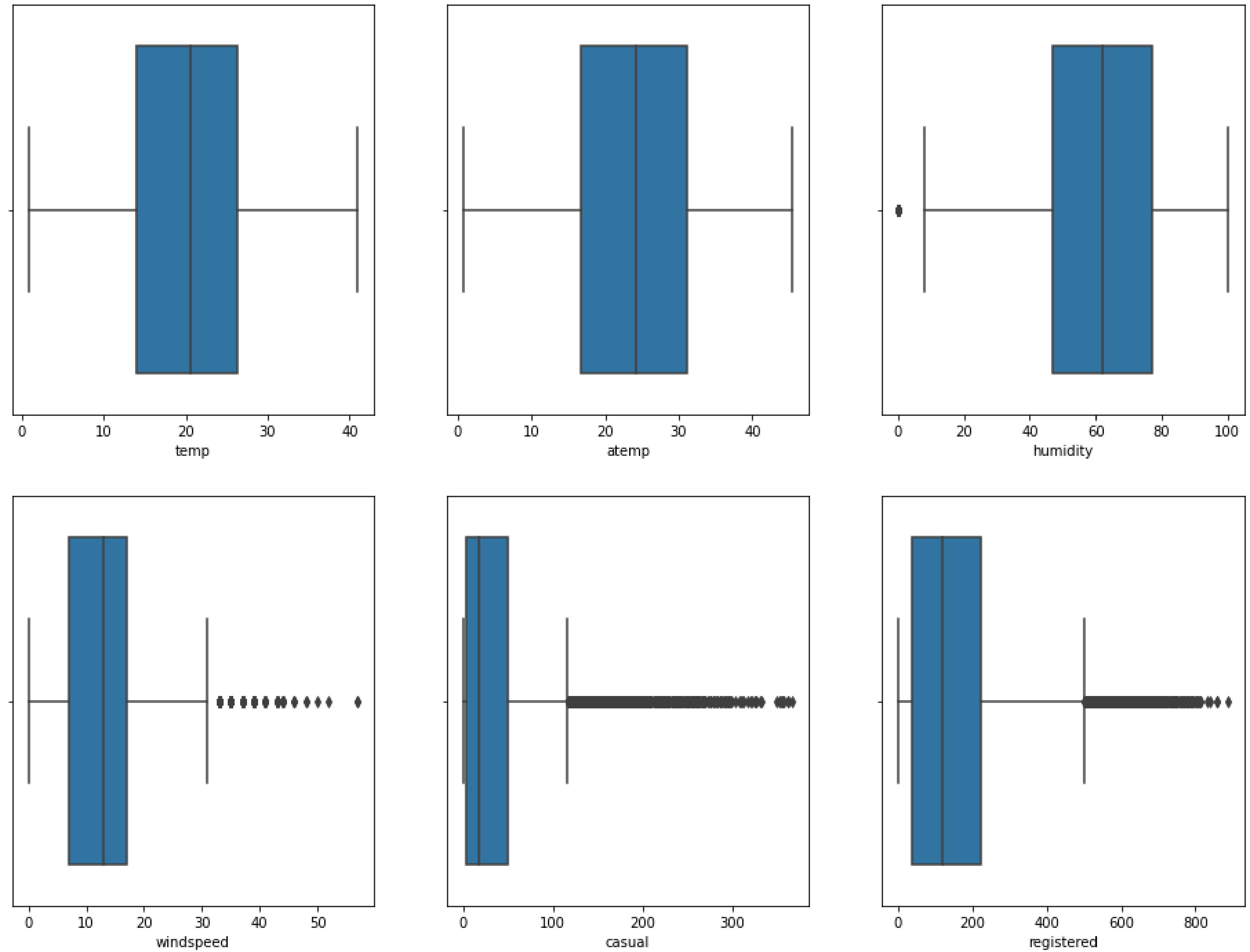
Data looks common as

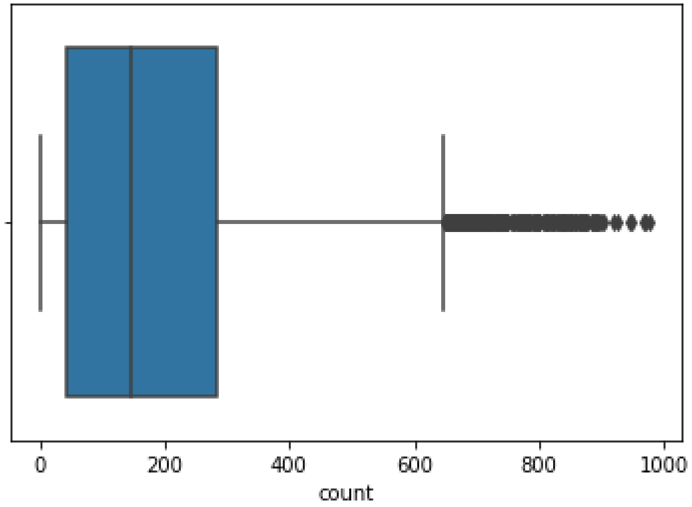
- equal number of days in each season
- less holidays
- more working days
- weather is mostly 1 i.e. `Clear, Few clouds, partly cloudy, partly cloudy` .

In [105…

```
# plotting box plots to detect outliers for each numerical variables
display(Markdown("### Bar plots of each feature:"))
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()
```

Bar plots of each feature:


0
10
20
30
40
temp
0
10
20
30
40
atemp
0
20
40
60
80
100
humidity
0
10
20
30
40
50
windspeed
0
100
200
300
casual
0
200
400
600
800
registered

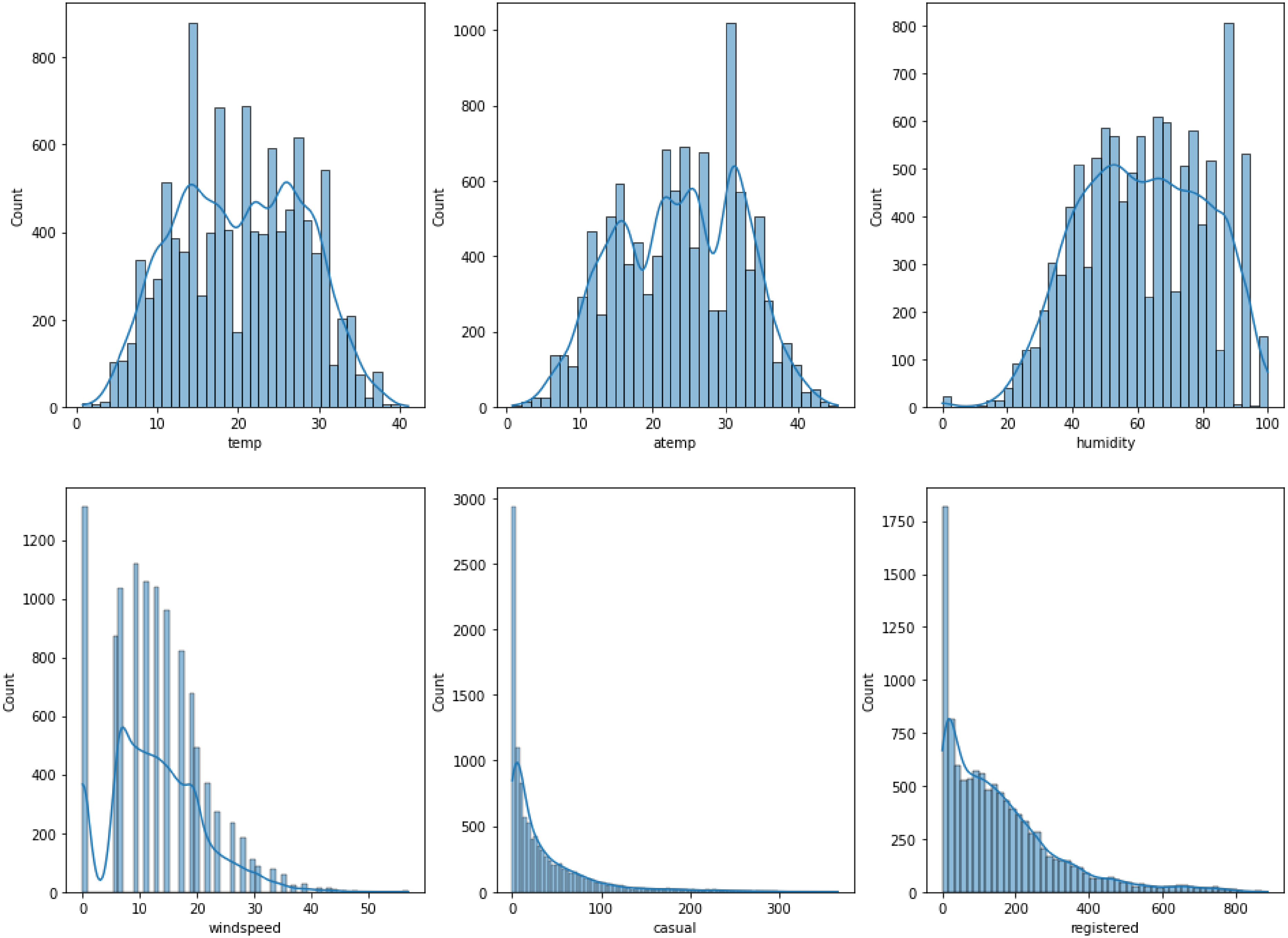


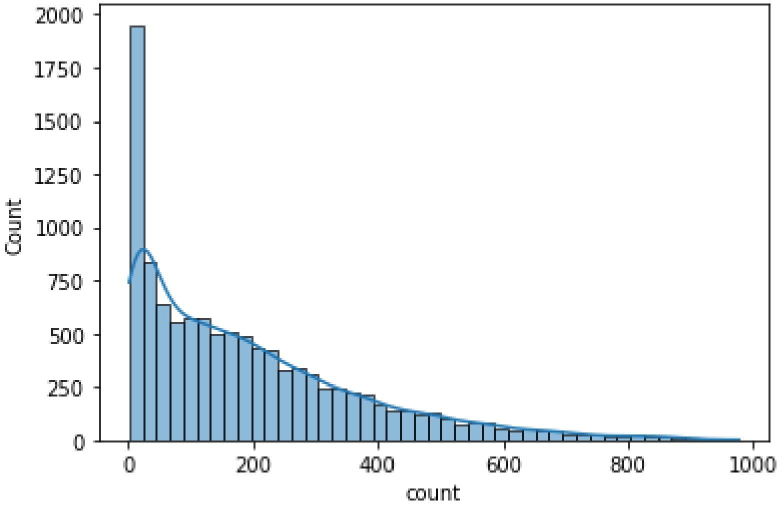
humidity, **casual**, **registered** and **count** have outliers in the data.

In [106…

```
# understanding the distribution for each numerical variables
display(Markdown("### Hist plots of each numerical feature:"))
num_cols = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered','count']

fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(16, 12))

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()
```

Hist plots of each numerical feature:


Count
temp
Count
atemp
Count
humidity
Count
windspeed
Count
casual
Count
registered



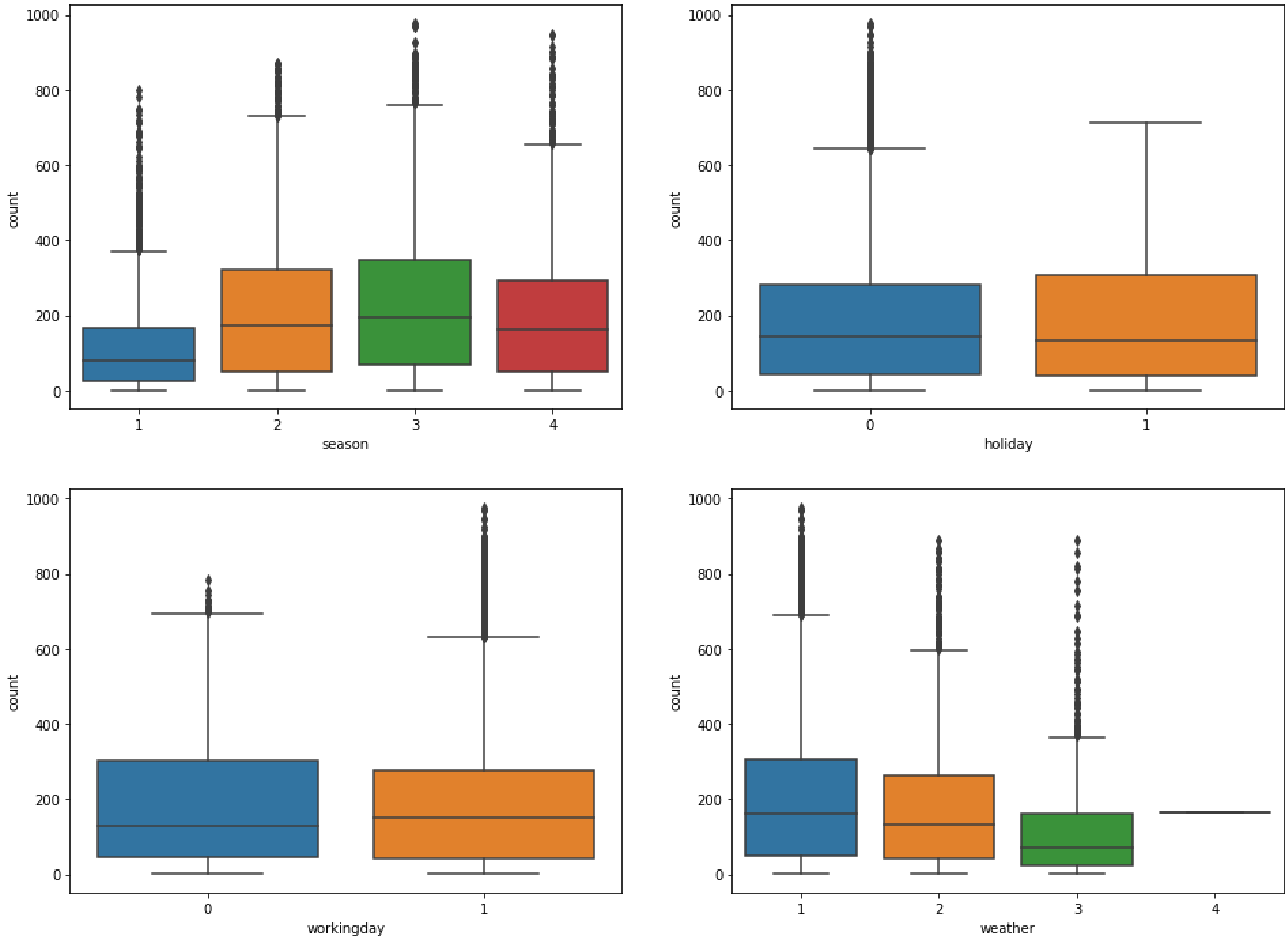
- **casual**, **registered** and **count** somewhat looks like **Log Normal Distribution**
- **temp**, **atemp** and **humidity** looks like they follows the **Normal Distribution**
- **windspeed** follows the **binomial distribution**

Bi-variate Analysis

In [107…

```
# plotting categorical variables againt count using boxplots
display(Markdown("### Bar plots of categorical variables againt count:"))
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()
```

Bar plots of categorical variables againt count:

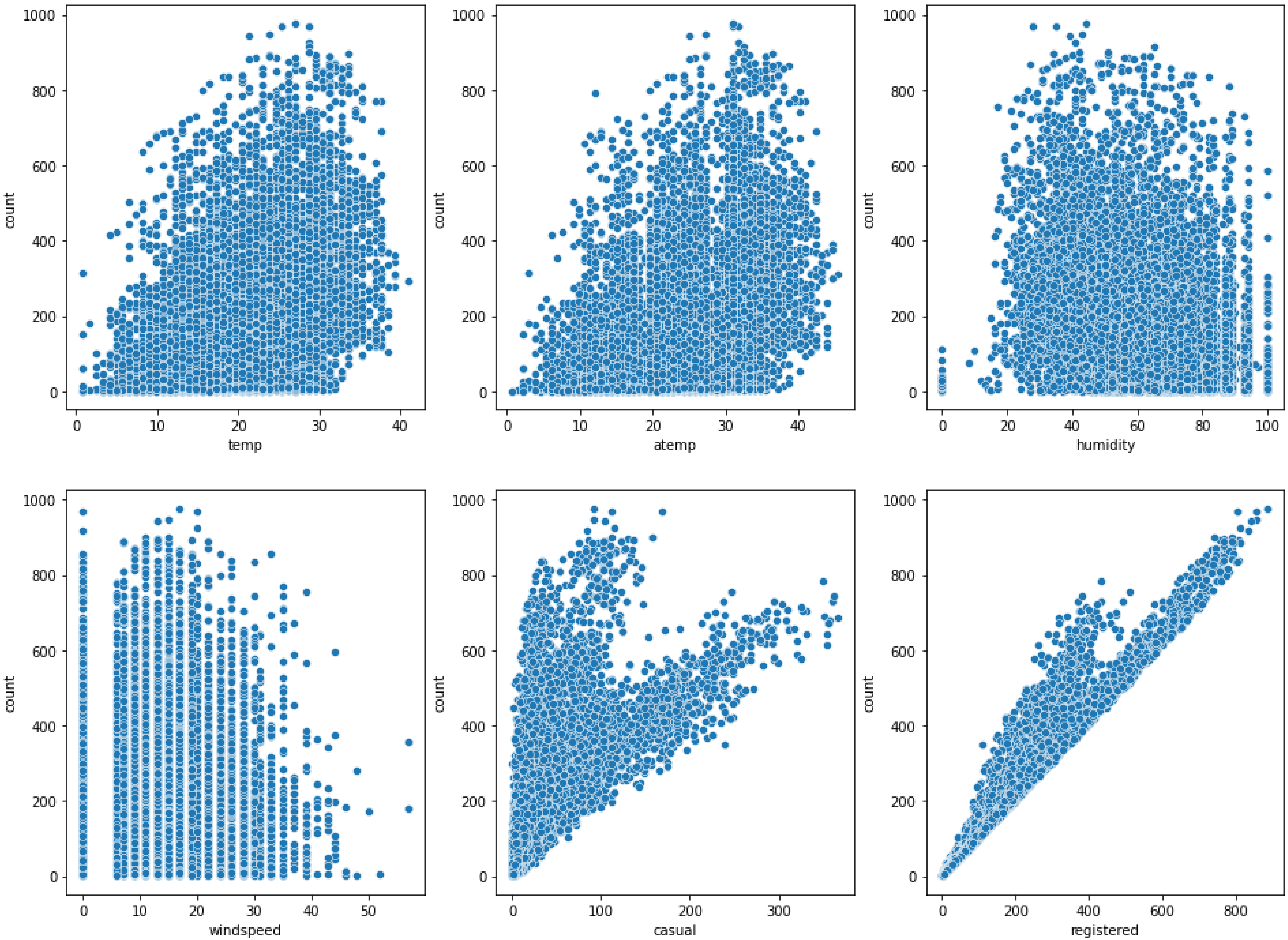

1000
800
600
400
200
0
count
1
2
3
4
season
holiday
0
1
workingday
weather

- In **fall** and **summer** seasons more bikes are rented as compared to other seasons.
- On **holiday** more bikes are rented.
- On **holiday** more bikes are rented.
- Whenever there is **rain, thunderstorm, snow or fog**, there were significantly less bikes were rented and when **Clear, Few clouds, partly cloudy, partly cloudy** the bike rent will ramp up.

In [108…

```
# plotting numerical variables againt count using scatterplot
display(Markdown("### scatterplot plots of numerical variables against count:"))
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()
```

scatterplot plots of numerical variables against count:


count
temp
atemp
humidity
windspeed
casual
registered

- count and both casual/registered columns seems positevely correlated.
- 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.

In [109…

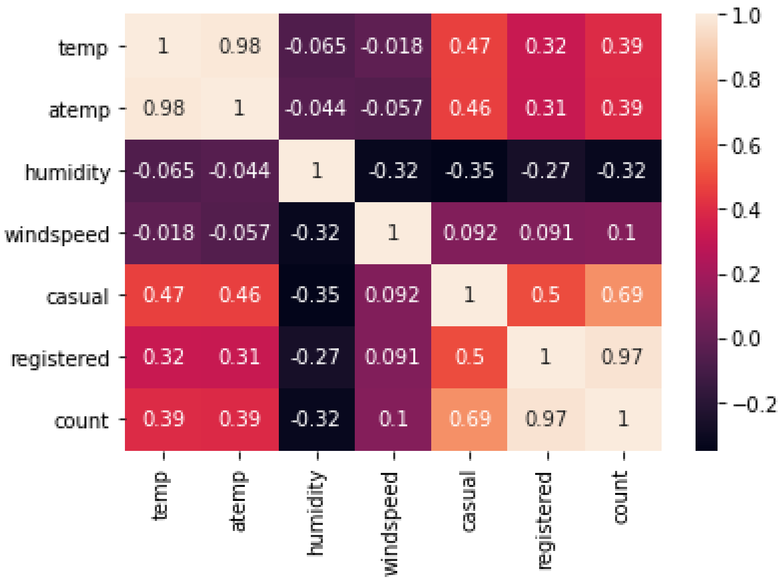
```
# understanding the correlation between count and numerical variables
df.corr()['count']
```

Out[109…

```
temp          0.394454
atemp         0.389784
humidity     -0.317371
windspeed     0.101369
casual        0.690414
registered    0.970948
count         1.000000
Name: count, dtype: float64
```

In [110…

```
sns.heatmap(df.corr(), annot=True)
plt.show()
```



Hypothesis Testing - 1

Null Hypothesis (H0): Weather is independent of the season

Alternate Hypothesis (H1): Weather is not independent of the season

Significance level (alpha): 0.05

We will use **chi-square test** to test hypyothesis defined above.

In [111…
```
data_table = pd.crosstab(df['season'], df['weather'])
print("Observed values:")
data_table
```

```
Observed values:
```

Out[111…

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

In [112…
```
alpha = 0.05 # taking the 95% CI
chi2, p_val, dof, expected = stats.chi2_contingency(data_table)
expected_values = val[3]
expected_values



print("chi-square test statistic: ", chi2)
print("degrees of freedom: ", dof)
print("p value : ",p_val)
```

```
if p_val <= alpha:
    print("\nSince p-value is less than the alpha 0.05, We reject the Null Hypothesis.\n\
Hence that Weather is dependent on the season.")
else:
    print("Since p-value is greater than the alpha 0.05, We fail to reject the Null Hypothesis.\n\
Hence that Weather is not dependent on the season.")
```

```
chi-square test statistic:  49.158655596893624
degrees of freedom:  9
p value :  1.549925073686492e-07

Since p-value is less than the alpha 0.05, We reject the Null Hypothesis.
Hence that Weather is dependent on the season.
```

Hypothesis Testing - 2

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

We will use the **2-Sample T-Test** to test the hypothess defined above

In [113…

```
data_group1 = df[df['workingday']==0]['count'].values
data_group2 = df[df['workingday']==1]['count'].values


np.var(data_group1), np.var(data_group2)
```

Out[113…

```
(30171.346098942427, 34040.69710674686)
```

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.

Here, the ratio is 34040.70 / 30171.35 which is less than 4:1

In [114…

```
stats.ttest_ind(a=data_group1, b=data_group2, equal_var=True)
```

Out[114…

```
Ttest_indResult(statistic=-1.2096277376026694, pvalue=0.22644804226361348)
```

Since pvalue is greater than 0.05 so we fail to reject the Null hypothesis. We consider working day has no effect on the count of cycles rented.

Hypothesis Testing - 3

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

Here, we will use the **ANOVA** to test the hypothess defined above

In [115…

```
# defining the data groups for the ANOVA
# 1. weather

gp1 = df[df['weather']==1]['count'].values
gp2 = df[df['weather']==2]['count'].values
gp3 = df[df['weather']==3]['count'].values
gp4 = df[df['weather']==4]['count'].values

gp5 = df[df['season']==1]['count'].values
gp6 = df[df['season']==2]['count'].values
gp7 = df[df['season']==3]['count'].values
gp8 = df[df['season']==4]['count'].values

# conduct the one-way anova
stats.f_oneway(gp1, gp2, gp3, gp4)
```

Out[115…
```
F_onewayResult(statistic=65.53024112793271, pvalue=5.482069475935669e-42)
```

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 weathers

In [116…
```python
# defining the data groups for the ANOVA
# 2. season

gp1 = df[df['season']==1]['count'].values
gp2 = df[df['season']==2]['count'].values
gp3 = df[df['season']==3]['count'].values
gp4 = df[df['season']==4]['count'].values

# conduct the one-way anova
stats.f_oneway(gp1, gp2, gp3, gp4)
```

Out[116…
```
F_onewayResult(statistic=236.94671081032106, pvalue=6.164843386499654e-149)
```

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 seasons

Inferences:

- There are no missing values in the dataset.
- **casual** and **registered** attributes might have outliers as their mean and median are very far away to one another, also the value of standard deviation is also high which tells us that there is high variance in the data of these attributes.
- Data looks common as
    - equal number of days in each season
    - less holiday
    - more working days
- humidity, casual, registered and count have outliers in the data.
- 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

- In fall and summer seasons more bikes are rented as compared to other seasons.
- On holiday more bikes are rented.
- On holiday more bikes are rented.
- Whenever there is **rain, thunderstorm, snow or fog,** there were significantly less bikes were rented and **when Clear, Few clouds, partly cloudy, partly cloudy** the bike rent will ramp up.
- count and both casual/registered columns seems positively correlated.
- 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.

Recommendations:

- From hypothesis testing using **chi-square test** we can consider Weather is dependent on the season.
- From hypothesis testing using **2-Sample T-Test** we consider working day has no effect on the count of cycles rented.
- From hypothesis testing using **2-Sample T-Test** we consider working day has no effect on the count of cycles rented.
- From hypothesis testing using **ANOVA** we can consider that Number of cycles rented is not similar in different weathers
- From hypothesis testing using **ANOVA** we can consider that Number of cycles rented is not similar in different seasons

In [ ]:

```

```