BUSINESS CASE
YULU-HYPOTHESIS TESTING

Akanksha Trivedi

Yulu is India’s leading micro-mobility service provider, which offers unique vehicles for the daily commute.

Yulu was founded in 2017 by Amit Gupta, RK Misra, and Naveen Dachuri. In November 2023, the company's CFO Anuj Tewari was elevated to the role of co-founder. The headquarters of Yulu is in Bengaluru. Yulu has launched its services in Indore, Madhya Pradesh. Yulu service is actively used for the last-mile commute as well as last-mile deliveries, making a real impact on India's goal of electrifying its mobility by 2030. Yulu provides the safest commute solution through a user-friendly mobile app to enable shared and sustainable commuting. In 2022, Yulu has partnered Canadian auto parts manufacturer Magna International to start a battery-as-a-service (BaaS) business, Yuma Energy.

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.

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?

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)
- working day: 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.

Import Libraries

```
[ ]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     from scipy import stats
```

Loading the dataset

```
df = pd.read_csv("bike1.csv")
```

Let us check the first five data

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	01-01-2011 00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	01-01-2011 01:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	01-01-2011 02:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	01-01-2011 03:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	01-01-2011 04:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

Shape of the data

```
print(f"# rows: {df.shape[0]} \n# columns: {df.shape[1]}")
```

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

Info of the data

```
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
```

Data type of all the attributes needs to change into proper data type

```
df['datetime'] = pd.to_datetime(df['datetime'],format = "%d-%m-%Y %H:%M")
cat_cols = ['season','holiday','workingday','weather']
for col in cat_cols:
  df[col] = df[col].astype('object')
```

```
df.dtypes
```

```
datetime      datetime64[ns]
season                object
holiday               object
workingday            object
weather               object
temp                 float64
atemp                float64
humidity               int64
windspeed            float64
casual                 int64
registered             int64
count                  int64
dtype: object
```

Ranges of attributes:

```
df.iloc[:,1:].describe(include = "all")
```

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
count	10886.0	10886.0	10886.0	10886.0	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
unique	4.0	2.0	2.0	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	4.0	0.0	1.0	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	2734.0	10575.0	7412.0	7192.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	NaN	NaN	NaN	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191.574132
std	NaN	NaN	NaN	NaN	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.144454
min	NaN	NaN	NaN	NaN	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	NaN	NaN	NaN	NaN	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.000000
50%	NaN	NaN	NaN	NaN	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.000000
75%	NaN	NaN	NaN	NaN	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.000000
max	NaN	NaN	NaN	NaN	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.000000

Identify the missing values

```
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
```

There are no null values across the entire dataset.

Minimum and maximum values

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

```
Timestamp('2011-01-01 00:00:00')
```

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

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

Number of unique values in each categorical column

```
df[cat_cols].melt().groupby(['variable', 'value'])[['value']].count()
```

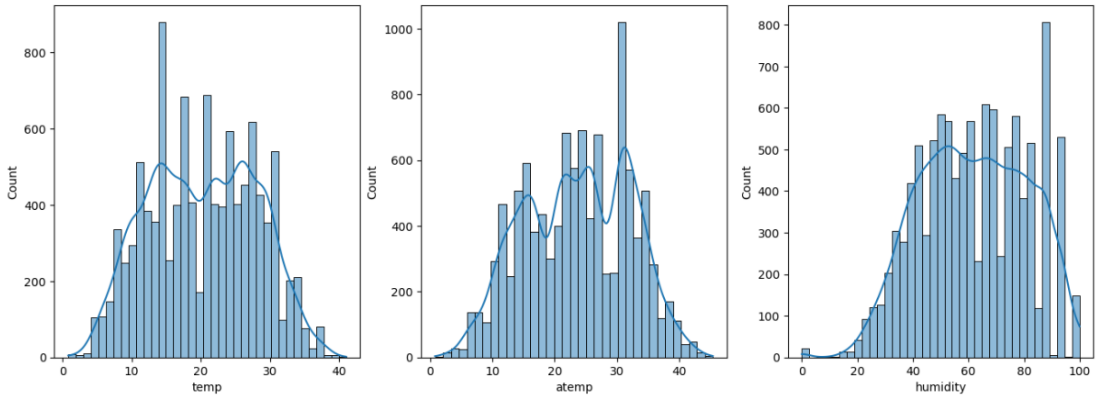
variable	value	value
holiday	0	10575
	1	311
season	1	2686
	2	2733
	3	2733
	4	2734
weather	1	7192
	2	2834
	3	859
	4	1
workingday	0	3474
	1	7412

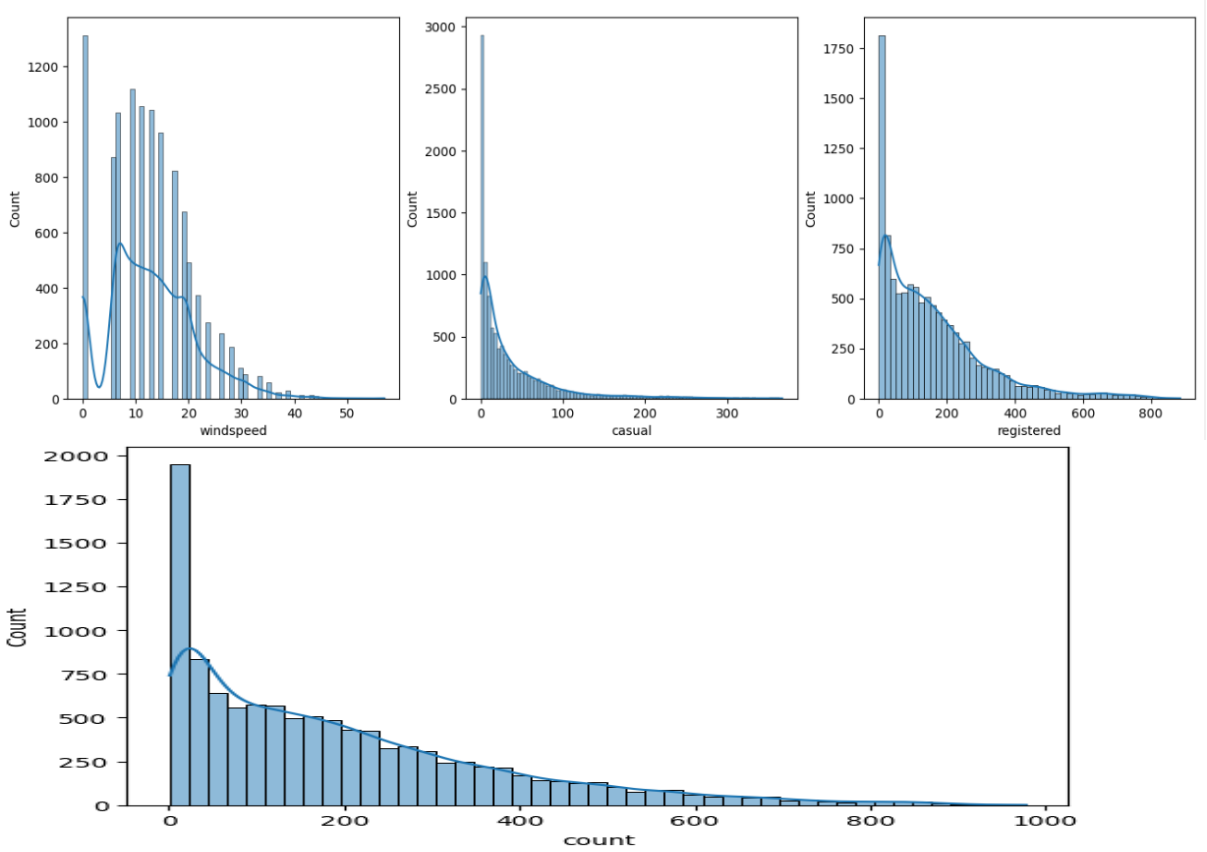
Univariate analysis

Distribution of numerical variables-

```
a = ['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[a[index]], ax=axis[row, col], kde=True)
        index += 1

plt.show()
sns.histplot(df[a[-1]], kde=True)
plt.show()
```





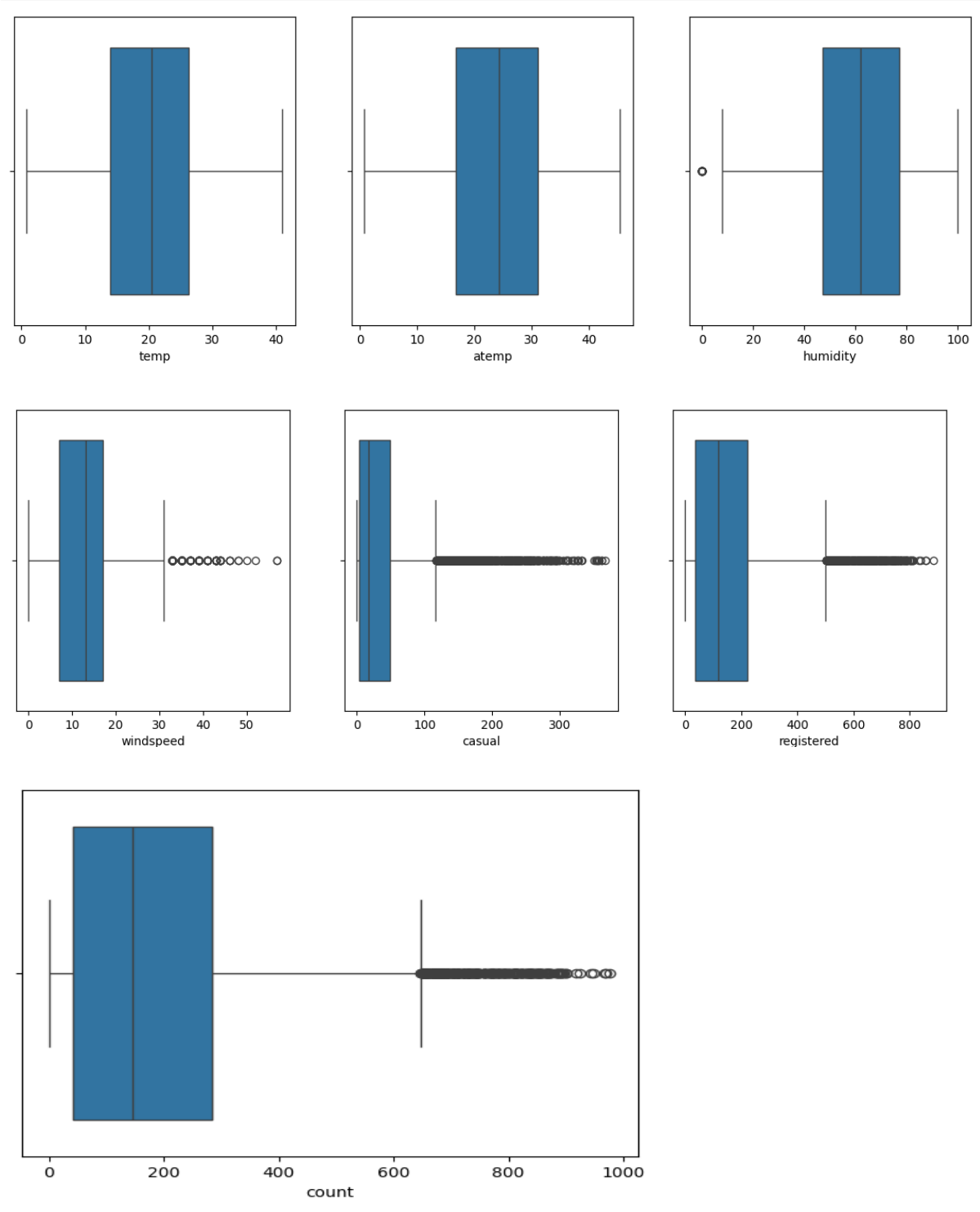
Insights

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

Detect Outliers in the data

```
fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(14, 10))
index = 0
for row in range(2):
    for col in range(3):
        sns.boxplot(x=df[a[index]], ax=axis[row, col])
        index += 1

plt.show()
sns.boxplot(x=df[a[-1]])
plt.show()
```

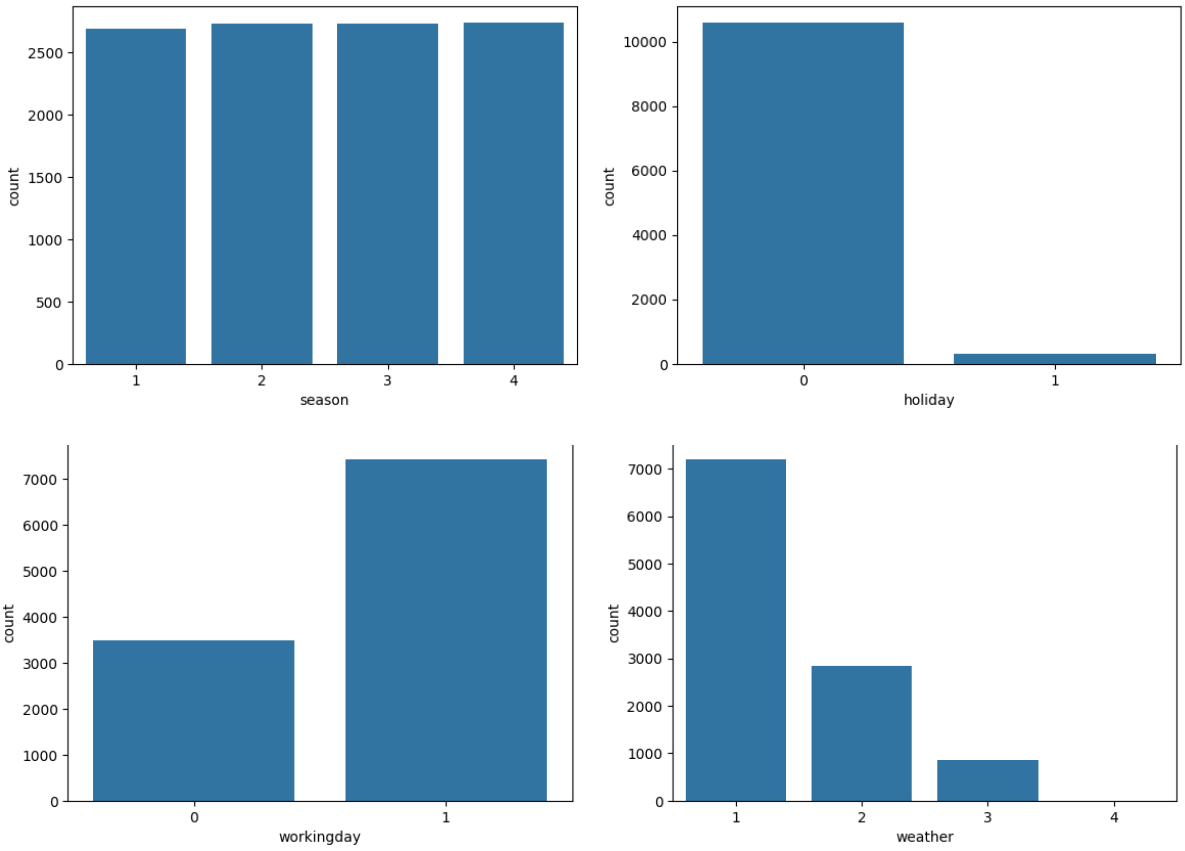


Insights

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

Count plot of each categorical column

```
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(14, 10))
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()
```

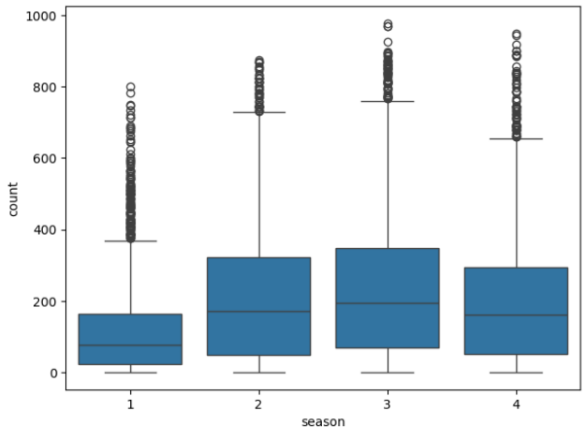


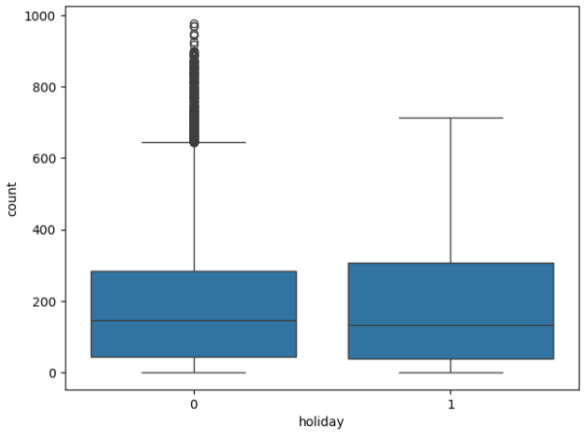
Bivariate analysis

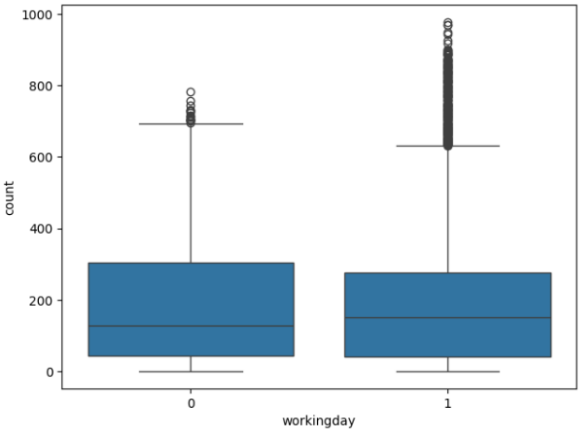
Plotting boxplot - categorical variables against count-

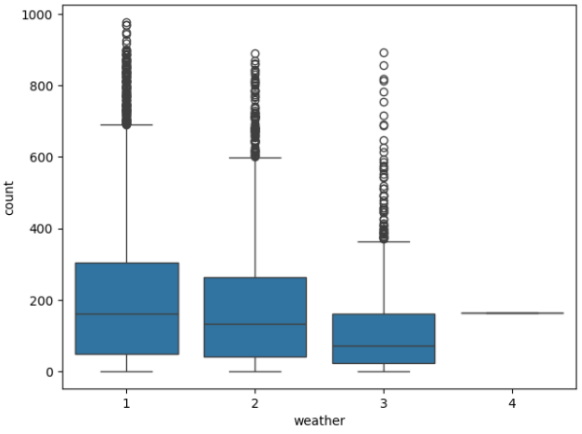
```
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(16, 12))
index = 0
for row in range(2):
    for col in range(2):
        sns.boxplot(data=df, x=cat_cols[index], y='count', ax=axis[row, col])
        index += 1

plt.show()
```







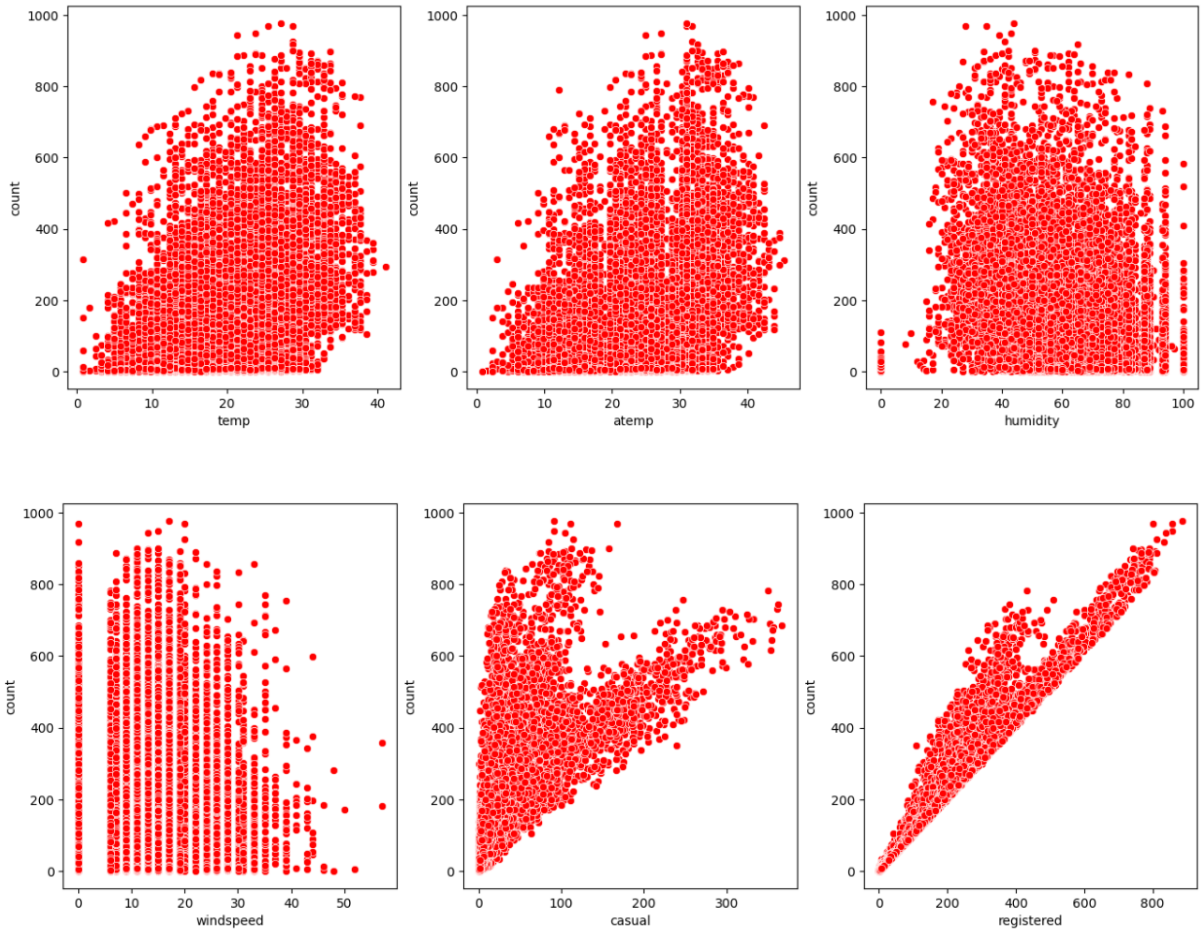


Insights

- In **summer** and **fall** seasons more bikes are rented as compared to other seasons.
- Whenever it is a **holiday** more bikes are rented.
- It is also clear from the working day also that whenever day is holiday or weekend, slightly more bikes rented.
- Whenever there is **rain, thunderstorm, snow or fog**, there were less bikes rented.

Plotting scatterplot -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=a[index], y='count',color = "red", ax=axis[row, col])
        index += 1
plt.show()
```



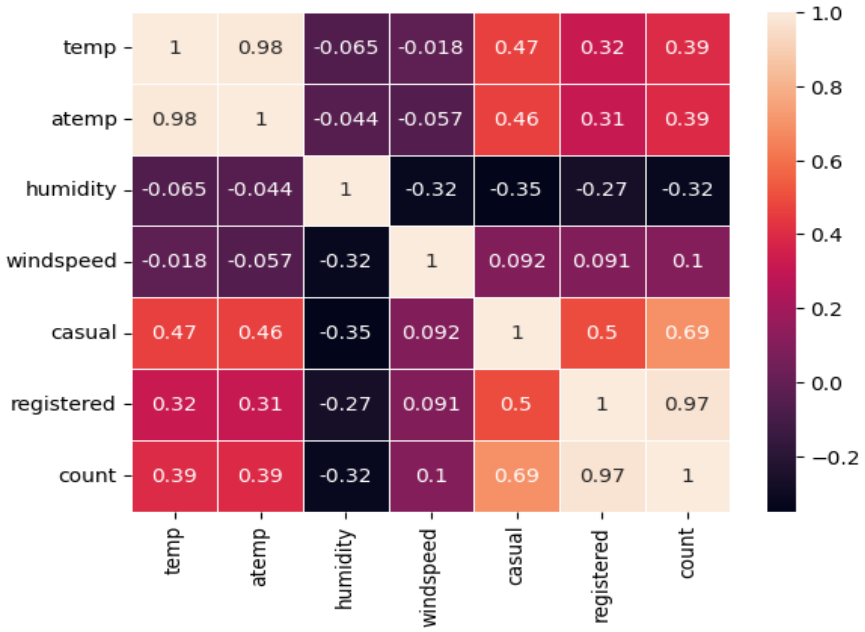
Insights

- Whenever the humidity is less than 20, number of bikes rented is 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.

Correlation between count and numerical values

```
df1 = df[["temp","atemp","humidity","windspeed","casual","registered","count"]]
print(df1.corr())
```

```
sns.heatmap(df1.corr(),annot=True, linewidth = .5)
plt.show()
```



Insights:

- Strong positive relationship between count and registered.
- Weak positive relationship between count and temperature.
- Weak negative relationship between count and humidity.

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

First, use **chi-square test** to test

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

```
Observed values:
```

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

```
val = stats.chi2_contingency(data_table)
expected_values = val[3]
expected_values
```

```
array([[1.77454639e+03, 6.99258130e+02, 2.11948742e+02, 2.46738931e-01],
       [1.80559765e+03, 7.11493845e+02, 2.15657450e+02, 2.51056403e-01],
       [1.80559765e+03, 7.11493845e+02, 2.15657450e+02, 2.51056403e-01],
       [1.80625831e+03, 7.11754180e+02, 2.15736359e+02, 2.51148264e-01]])
```

```
nrows, ncols = 4, 4
dof = (nrows-1)*(ncols-1)
print("degrees of freedom: ", dof)
alpha = 0.05
chi_sqr = sum([(o-e)**2/e for o, e in zip(data_table.values, expected_values)])
chi_sqr_statistic = chi_sqr[0] + chi_sqr[1]
print("chi-square test statistic: ", chi_sqr_statistic)
critical_val = stats.chi2.ppf(q=1-alpha, df=dof)
print(f"critical value: {critical_val}")
p_val = 1-stats.chi2.cdf(x=chi_sqr_statistic, df=dof)
print(f"p-value: {p_val}")
if p_val <= alpha:
    print("Weather is dependent on the season.")
else:
    print( "Weather is independent on the season.")
```

```
degrees of freedom:  9
chi-square test statistic:  44.09441248632364
critical value: 16.918977604620448
p-value: 1.3560001579371317e-06
Weather is dependent on the season.
```

Hypothesis testing 2

Null Hypothesis: Working day has no effect on the number of cycles rented.

Alternate Hypothesis: Working day has effect on the number of cycles rented.

Significance level (alpha): 0.05

use the **2-Sample T-Test**-

```
data1 = df[df['workingday']==0]['count'].values
data2 = df[df['workingday']==1]['count'].values

np.var(data1), np.var(data2)
```

```
(30171.346098942427, 34040.69710674686)
```

Here, the ratio is 34040.70 / 30171.35 which is less than 4:1. So we can consider that the given data groups have equal variance.

```
stats.ttest_ind(a=data1, b=data2, equal_var=True)
```

```
TtestResult(statistic=-1.2096277376026694, pvalue=0.22644804226361348, df=10884.0)
```

Since p-value>alpha so we do not reject the null hypothesis. We do not have the sufficient evidence to say that working day has effect on the number 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

use the **ANOVA** to test –

Since p-value<alpha so we reject the null hypothesis. So, Number of cycles rented is not similar in different weather and season.

```
g1 = df[df['weather']==1]['count'].values
g2 = df[df['weather']==2]['count'].values
g3 = df[df['weather']==3]['count'].values
g4 = df[df['weather']==4]['count'].values

g5 = df[df['season']==1]['count'].values
g6 = df[df['season']==2]['count'].values
g7 = df[df['season']==3]['count'].values
g8 = df[df['season']==4]['count'].values

# conduct the one-way anova
stats.f_oneway(g1, g2, g3, g4, g5, g6, g7, g8)
```

```
F_onewayResult(statistic=127.96661249562491, pvalue=2.8074771742434642e-185)
```

Recommendations:

- Yulu can focus on promoting bikes rental during the summer and spring season. Seasonal discounts, offers and vouchers can attract more customers.
- Yulu can offer weather specific discounts to attract more customers during favourable weather conditions like clear and cloudy weather.
- We have noticed that mostly customers are registered, and the remaining are casual. Yulu can provide loyalty programs, exclusive offers, or personalized recommendations for registered users to encourage repeat business and for casual users, focus on providing a seamless rental experience and promoting the benefits of bike rentals for occasional use.
- Whenever the humidity is less than 20, number of bikes rented is extremely low. So Yulu can provide customer comfort zone like providing umbrella, raincoat and water bottles, tea. These small touches can contribute to a positive customer experience.
- Since Yulu focusses on providing a sustainable solution for vehicular pollution, it can give exclusive discounts on the occasions like Zero Emissions Day (21st September), Earth Day (22nd April), World Environment Day (5th June) etc to attract new users.