# yulu-project

April 10, 2023

## 1 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.

## 2 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

[24]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import binom
from scipy.stats import norm
from scipy.stats import expon
from scipy.stats import poisson
from scipy.stats import lognorm
import io
from scipy import stats
```

[25]:
```
##Importing the dataset Yulu Project
# from google.colab import files
```

[26]:
```
!wget https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/
↪original/bike_sharing.csv
```

```
--2023-04-10 17:28:05--  https://d2beiqkhq929f0.cloudfront.net/public_assets/ass
ets/000/001/428/original/bike_sharing.csv
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)…
108.157.172.173, 108.157.172.176, 108.157.172.183, …
Connecting to d2beiqkhq929f0.cloudfront.net
(d2beiqkhq929f0.cloudfront.net)|108.157.172.173|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 648353 (633K) [text/plain]
Saving to: 'bike_sharing.csv.1'

bike_sharing.csv.1  100%[===================>] 633.16K  --.-KB/s    in 0.1s

2023-04-10 17:28:05 (5.33 MB/s) - 'bike_sharing.csv.1' saved [648353/648353]
```

[27]:
```
yulu = pd.read_csv('bike_sharing.csv')
```

[28]:
```
yulu.head()
```

[28]:
```
              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
```

[29]:
```
yulu.tail()
```

[29]:
```
                  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
```

## 3 Column Profiling:

- datetime: datetime season: season (1: spring, 2: summer, 3: fall, 4: winter)
- holiday: whether day is a holiday or not
- 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 checking the structure & characteristics of the dataset

[30]:
```python
#Length of data
len(yulu)
```

[30]:
```
10886
```

[31]:
```python
#checking datatypes
yulu.dtypes
```

[31]:
```
datetime       object
season          int64
holiday         int64
workingday      int64
weather         int64
temp          float64
atemp         float64
```

```
humidity        int64
windspeed     float64
casual          int64
registered      int64
count           int64
dtype: object
```

[32]:
```python
yulu.shape
```

[32]:
```
(10886, 12)
```

[33]:
```python
yulu.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
```

[34]:
```python
#number of unique values in given dataset
for i in yulu.columns:
    print(i,":",yulu[i].nunique())
```

```
datetime : 10886
season : 4
holiday : 2
workingday : 2
weather : 4
temp : 49
atemp : 60
humidity : 89
windspeed : 28
casual : 309
registered : 731
```

```
count : 822
```

[35]:
```
yulu.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
```

[36]:
```
# statistical summary of the dataset
yulu.describe()
```

[36]:

| | 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
```

[37]:
```
# Minimum and Maximum values of Numerical columns such as -
#  "season","holiday","workingday","weather","humidity"␣
 ↪,"registered","count","windspeed","temp","atemp","datetime".
L = ["season","holiday","workingday","weather","humidity"␣
 ↪,"registered","count","windspeed","temp","atemp","datetime"]
for i in L:
    print("Maximum value of ",i,"is:",yulu[i].max())
    print("Minimum value of ",i,"is:",yulu[i].min())
```

```
Maximum value of  season is: 4
Minimum value of  season is: 1
Maximum value of  holiday is: 1
Minimum value of  holiday is: 0
Maximum value of  workingday is: 1
Minimum value of  workingday is: 0
Maximum value of  weather is: 4
Minimum value of  weather is: 1
Maximum value of  humidity is: 100
Minimum value of  humidity is: 0
Maximum value of  registered is: 886
Minimum value of  registered is: 0
Maximum value of  count is: 977
Minimum value of  count is: 1
Maximum value of  windspeed is: 56.9969
Minimum value of  windspeed is: 0.0
Maximum value of  temp is: 41.0
Minimum value of  temp is: 0.82
Maximum value of  atemp is: 45.455
Minimum value of  atemp is: 0.76
Maximum value of  datetime is: 2012-12-19 23:00:00
Minimum value of  datetime is: 2011-01-01 00:00:00
```

[38]:
```
#Statistical Summary
yulu.describe(include="all")
```

[38]:
```
                      datetime        season       holiday    workingday  \
count                    10886  10886.000000  10886.000000  10886.000000
```

```
unique                 10886        NaN        NaN        NaN
top      2011-01-01 00:00:00        NaN        NaN        NaN
freq                       1        NaN        NaN        NaN
mean                     NaN   2.506614   0.028569   0.680875
std                      NaN   1.116174   0.166599   0.466159
min                      NaN   1.000000   0.000000   0.000000
25%                      NaN   2.000000   0.000000   0.000000
50%                      NaN   3.000000   0.000000   1.000000
75%                      NaN   4.000000   0.000000   1.000000
max                      NaN   4.000000   1.000000   1.000000

             weather         temp         atemp      humidity     windspeed  \
count   10886.000000  10886.00000  10886.000000  10886.000000  10886.000000
unique           NaN          NaN           NaN           NaN           NaN
top              NaN          NaN           NaN           NaN           NaN
freq             NaN          NaN           NaN           NaN           NaN
mean        1.418427     20.23086     23.655084     61.886460     12.799395
std         0.633839      7.79159      8.474601     19.245033      8.164537
min         1.000000      0.82000      0.760000      0.000000      0.000000
25%         1.000000     13.94000     16.665000     47.000000      7.001500
50%         1.000000     20.50000     24.240000     62.000000     12.998000
75%         2.000000     26.24000     31.060000     77.000000     16.997900
max         4.000000     41.00000     45.455000    100.000000     56.996900

              casual    registered         count
count   10886.000000  10886.000000  10886.000000
unique           NaN           NaN           NaN
top              NaN           NaN           NaN
freq             NaN           NaN           NaN
mean       36.021955    155.552177    191.574132
std        49.960477    151.039033    181.144454
min         0.000000      0.000000      1.000000
25%         4.000000     36.000000     42.000000
50%        17.000000    118.000000    145.000000
75%        49.000000    222.000000    284.000000
max       367.000000    886.000000    977.000000
```

[39]:
```python
#mode value of each column
print("Mode values of all columns (both categorical and numerical)")
yulu.mode()
```

```
Mode values of all columns (both categorical and numerical)
```

[39]:
```
             datetime  season  holiday  workingday  weather   temp  \
0 2011-01-01 00:00:00     4.0      0.0         1.0      1.0  14.76
1 2011-01-01 01:00:00     NaN      NaN         NaN      NaN    NaN
2 2011-01-01 02:00:00     NaN      NaN         NaN      NaN    NaN
```

```
3      2011-01-01 03:00:00     NaN      NaN          NaN      NaN    NaN
4      2011-01-01 04:00:00     NaN      NaN          NaN      NaN    NaN
…                    …     …      …         …      …    …
10881  2012-12-19 19:00:00     NaN      NaN          NaN      NaN    NaN
10882  2012-12-19 20:00:00     NaN      NaN          NaN      NaN    NaN
10883  2012-12-19 21:00:00     NaN      NaN          NaN      NaN    NaN
10884  2012-12-19 22:00:00     NaN      NaN          NaN      NaN    NaN
10885  2012-12-19 23:00:00     NaN      NaN          NaN      NaN    NaN

       atemp  humidity  windspeed  casual  registered  count
0      31.06      88.0        0.0     0.0         3.0    5.0
1        NaN       NaN        NaN     NaN         NaN    NaN
2        NaN       NaN        NaN     NaN         NaN    NaN
3        NaN       NaN        NaN     NaN         NaN    NaN
4        NaN       NaN        NaN     NaN         NaN    NaN
…      …       …        …      …          …    …
10881    NaN       NaN        NaN     NaN         NaN    NaN
10882    NaN       NaN        NaN     NaN         NaN    NaN
10883    NaN       NaN        NaN     NaN         NaN    NaN
10884    NaN       NaN        NaN     NaN         NaN    NaN
10885    NaN       NaN        NaN     NaN         NaN    NaN

[10886 rows x 12 columns]
```

# 5 Missing values Outlier Detection

[40]:
```
## #checking null values in every column of our data
yulu.isnull().sum()
```

[40]:
```
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
```

[41]:
```
yulu.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
```

# 6 Univariate Analysis

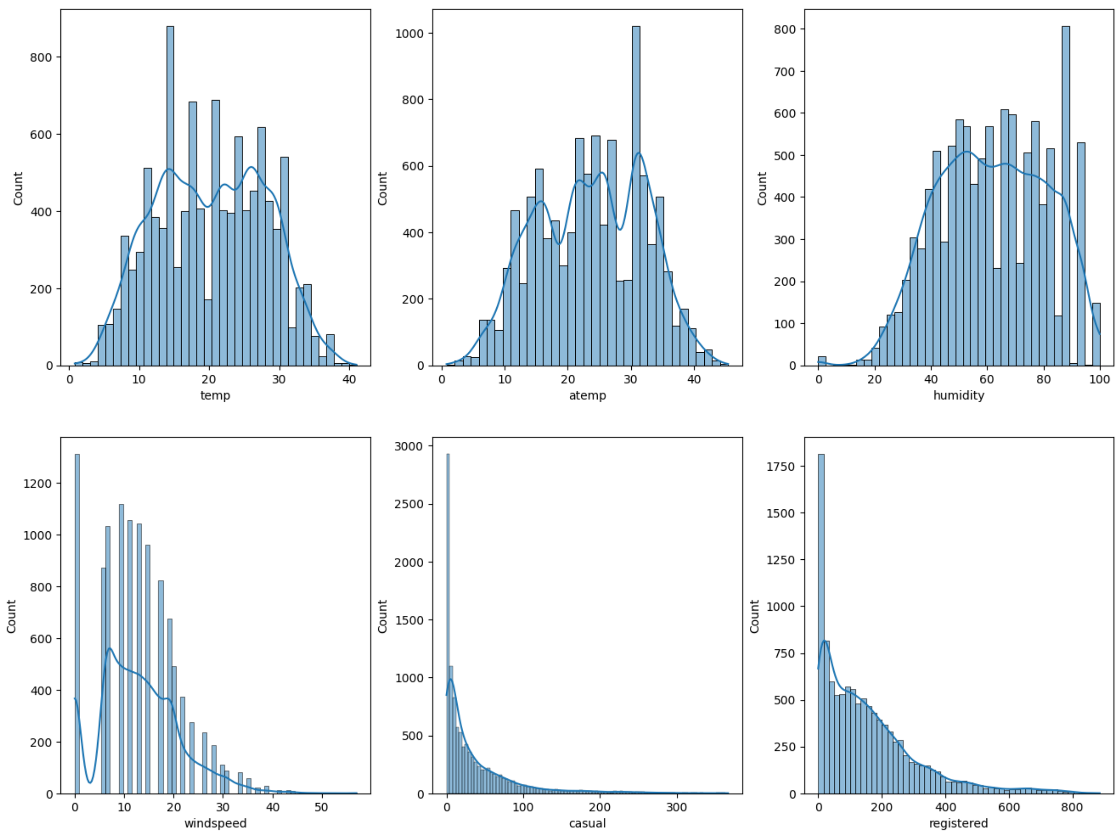
[42]:

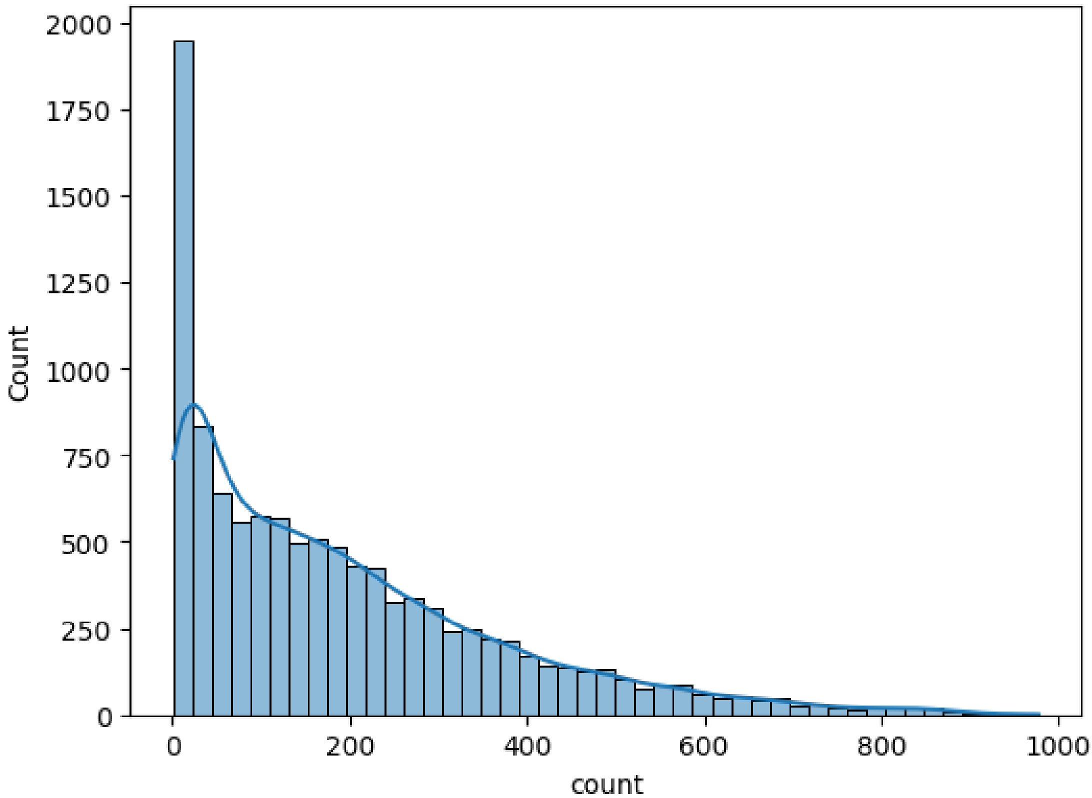
```
# understanding the distribution for numerical variables
num_cols = ['temp', 'atemp', 'humidity', 'windspeed', 'casual',␣
↪'registered','count']

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(yulu[num_cols[index]], ax=axis[row, col], kde=True)
        index += 1

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


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

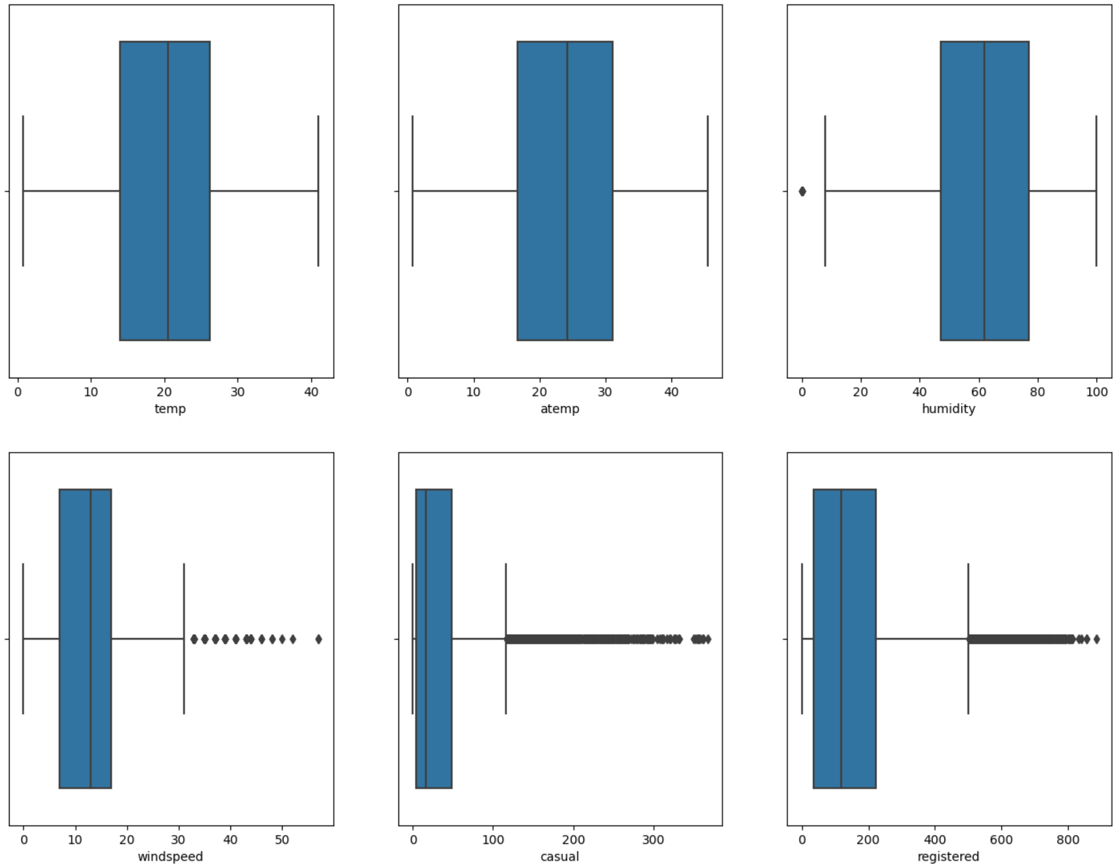



- casual, registered and count somewhat looks like Log Normal Distribution
- temp, atemp and humidity looks like they follows the Normal Distribution
- windspeed follows the binomial distribution

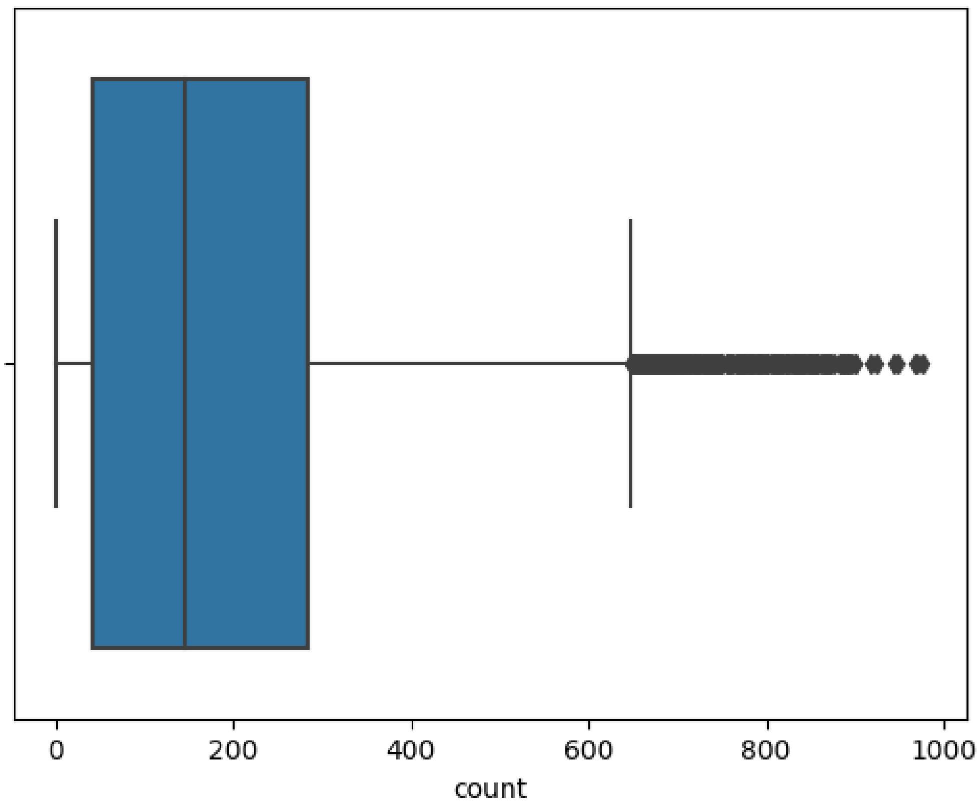
[43]:
```
# plotting box plots to detect outliers in the data
fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(16, 12))

index = 0
for row in range(2):
    for col in range(3):
        sns.boxplot(x=yulu[num_cols[index]], ax=axis[row, col])
        index += 1

plt.show()
sns.boxplot(x=yulu[num_cols[-1]])
plt.show()
```



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




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

[44]:
```
yulu['datetime'] = pd.to_datetime(yulu['datetime'])

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

[45]:
```
# minimum datetime and maximum datetime
yulu['datetime'].min(), yulu['datetime'].max()
```

[45]:
```
(Timestamp('2011-01-01 00:00:00'), Timestamp('2012-12-19 23:00:00'))
```

[46]:
```
# number of unique values in each categorical columns
yulu[cat_cols].melt().groupby(['variable', 'value'])[['value']].count()
```

[46]:
```
                    value
variable   value
holiday    0        10575
           1          311
season     1         2686
```

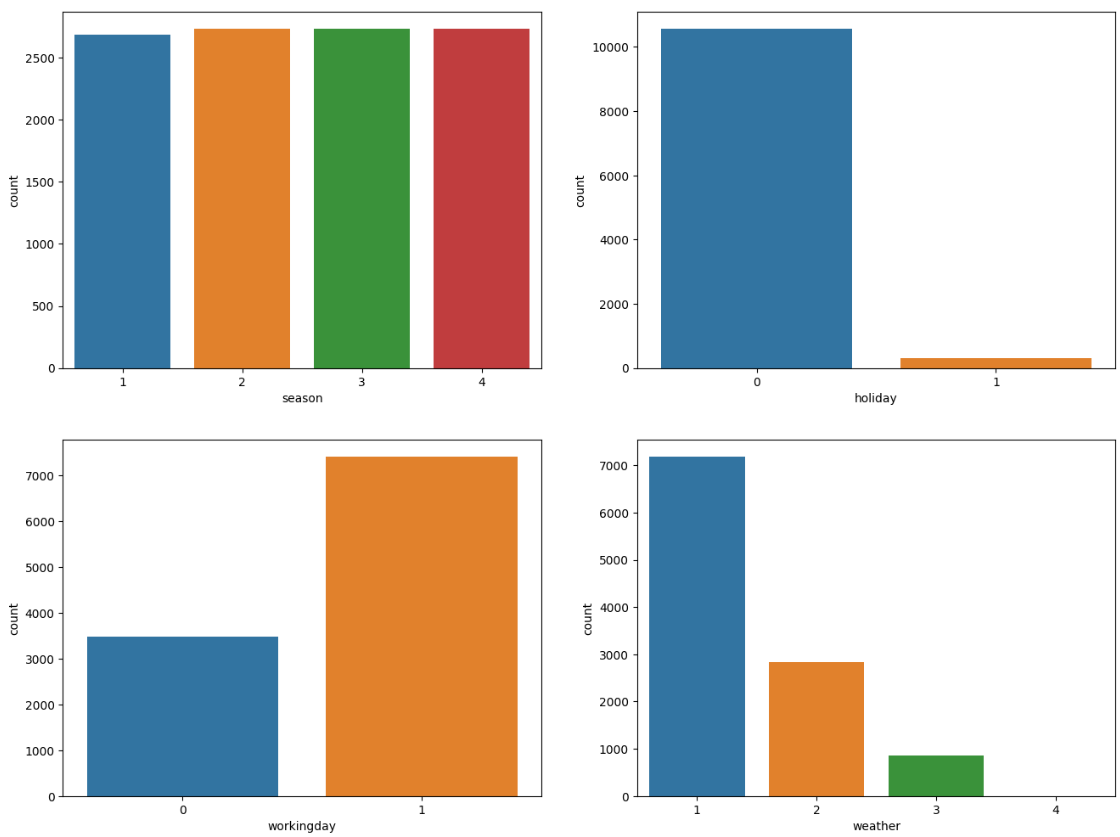
```
            2        2733
            3        2733
            4        2734
weather     1        7192
            2        2834
            3         859
            4           1
workingday  0        3474
            1        7412
```

[47]:

```python
# countplot of each categorical column
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=yulu, x=cat_cols[index], ax=axis[row, col])
        index += 1

plt.show()
```



Data looks common as it should be like equal number of days in each season, more working days and weather is mostly Clear, Few clouds, partly cloudy, partly cloudy.

Based on the provided variable-value table, we can see that the dataset includes information about holidays, seasons, weather, and working days. Here's an analysis of the data:

Holidays: The dataset includes information on whether a day is a holiday or not. Based on the values provided, we can see that out of the total number of observations (10,886), only 311 days are holidays. This suggests that holidays are relatively rare, and may not have a significant impact on overall bike rental demand.

Seasons: The dataset includes information on which season each observation falls under. Based on the values provided, we can see that the dataset has an almost equal number of observations for each season (around 2,700). This suggests that bike rental demand is relatively consistent across seasons, although it may be worth investigating further to see if there are any season-specific trends or patterns.

Weather: The dataset includes information on the weather conditions for each observation. Based on the values provided, we can see that the majority of observations (7,192) have a weather rating of 1, which suggests good weather conditions. This is followed by a weather rating of 2 (2,834 observations), which suggests slightly unfavorable weather conditions, and a weather rating of 3 (859 observations), which suggests bad weather conditions. There is also one observation with a weather rating of 4. This information can be used to optimize bike availability and pricing during certain weather conditions to encourage more rentals.

Working days: The dataset includes information on whether a day is a working day or not. Based on the values provided, we can see that the majority of observations (7,412) are working days, while the remaining 3,474 observations are non-working days. This suggests that bike rental demand may be higher on working days, likely due to commuters using bikes for transportation to and from work.

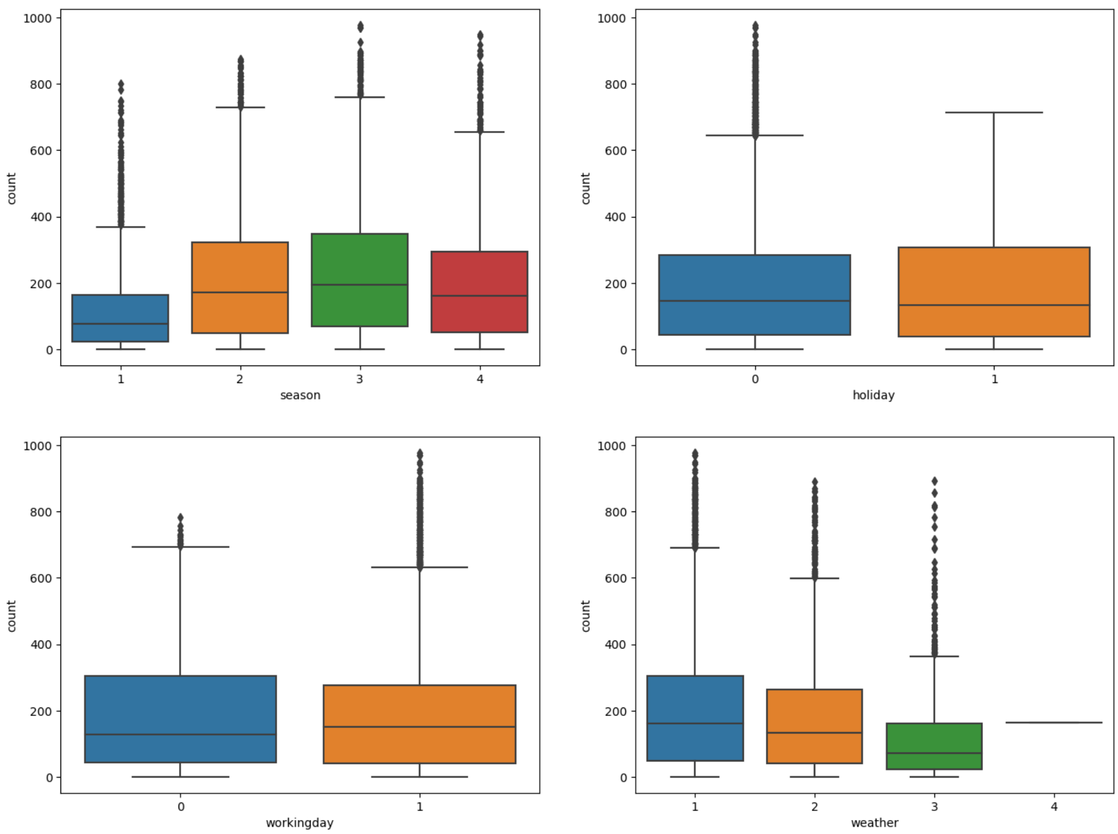
Overall, the dataset provides valuable information on various factors that may impact bike rental demand, such as weather conditions and working days. This information can be used to optimize bike availability, pricing, and promotions to encourage more rentals and improve the overall user experience.

# 7 Bi-variate Analysis

[48]:
```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=yulu, x=cat_cols[index], y='count', ax=axis[row, col])
        index += 1

plt.show()
```

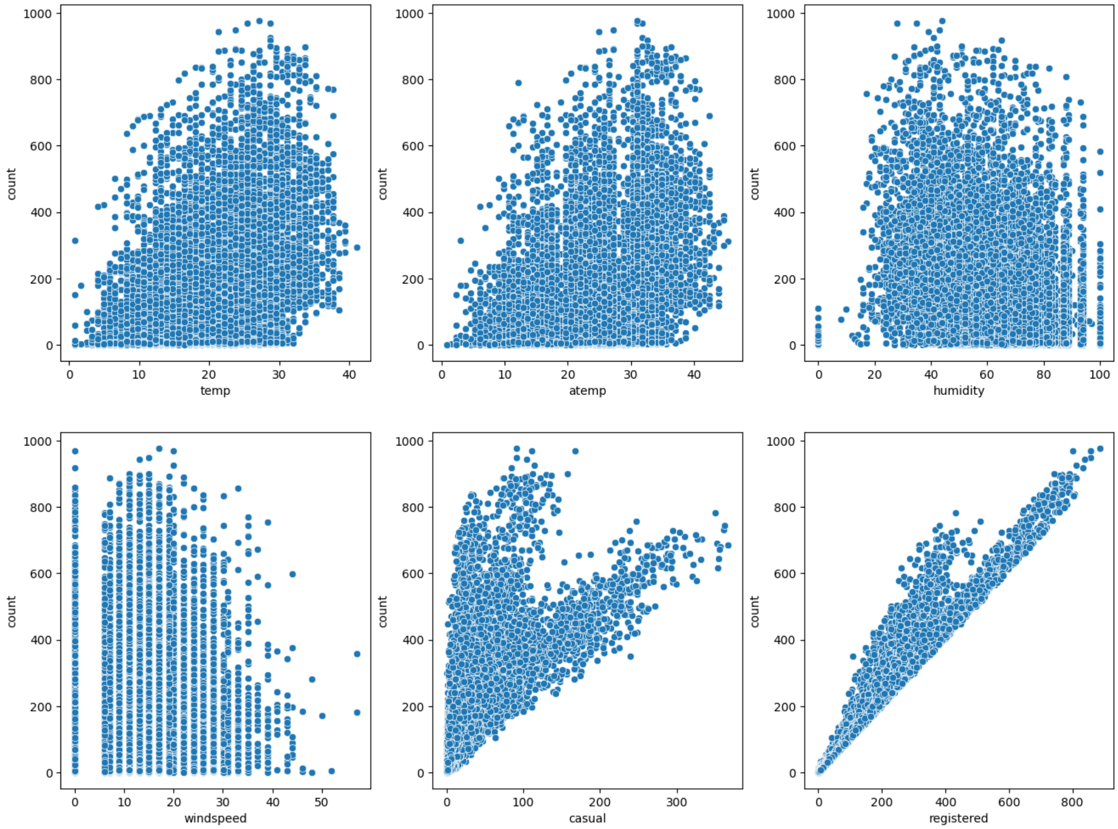



- In summer and fall seasons more bikes are rented as compared to other seasons.
- Whenever its a holiday more bikes are rented.
- It is also clear from the workingday also that whenever day is holiday or weekend, slightly more bikes were rented.
- Whenever there is rain, thunderstorm, snow or fog, there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

[49]:
```
# 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=yulu, x=num_cols[index], y='count', ax=axis[row,␣
↪col])
        index += 1

plt.show()
```




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

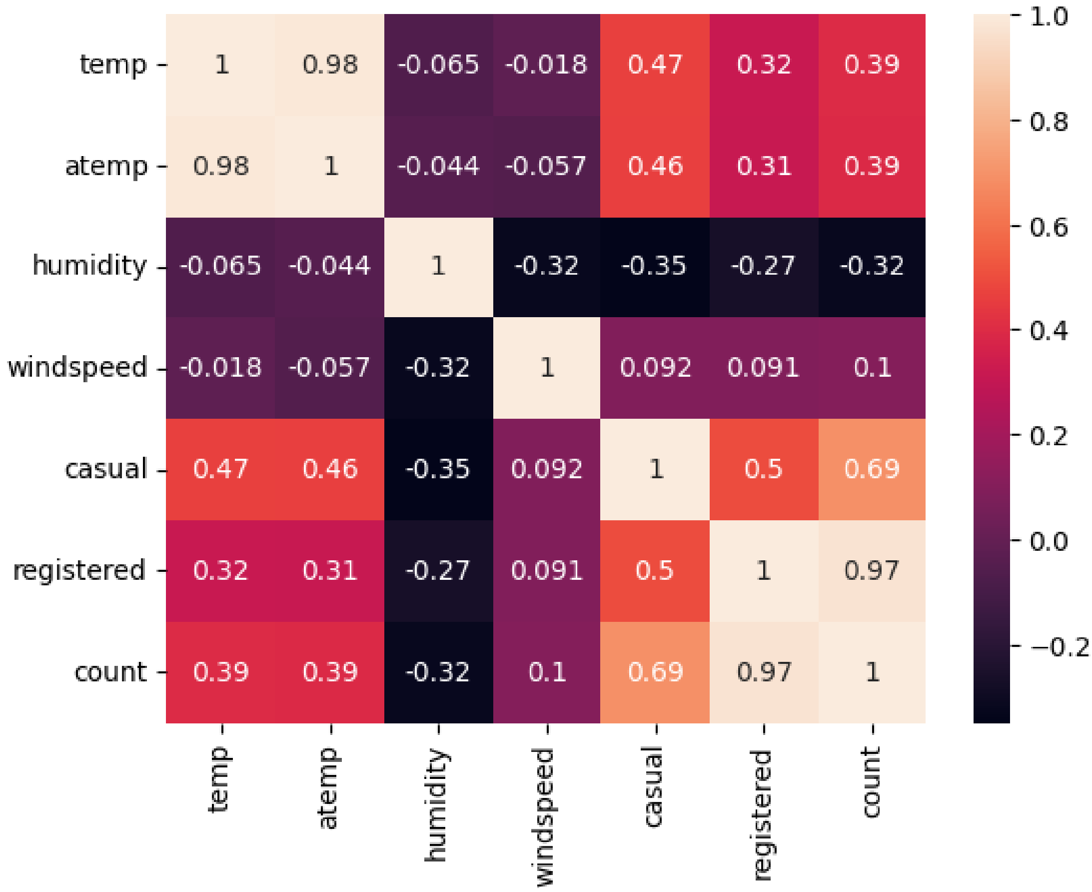
Whenever the temperature is less than 10, number of bikes rented is less.

Whenever the windspeed is greater than 35, number of bikes rented is less.

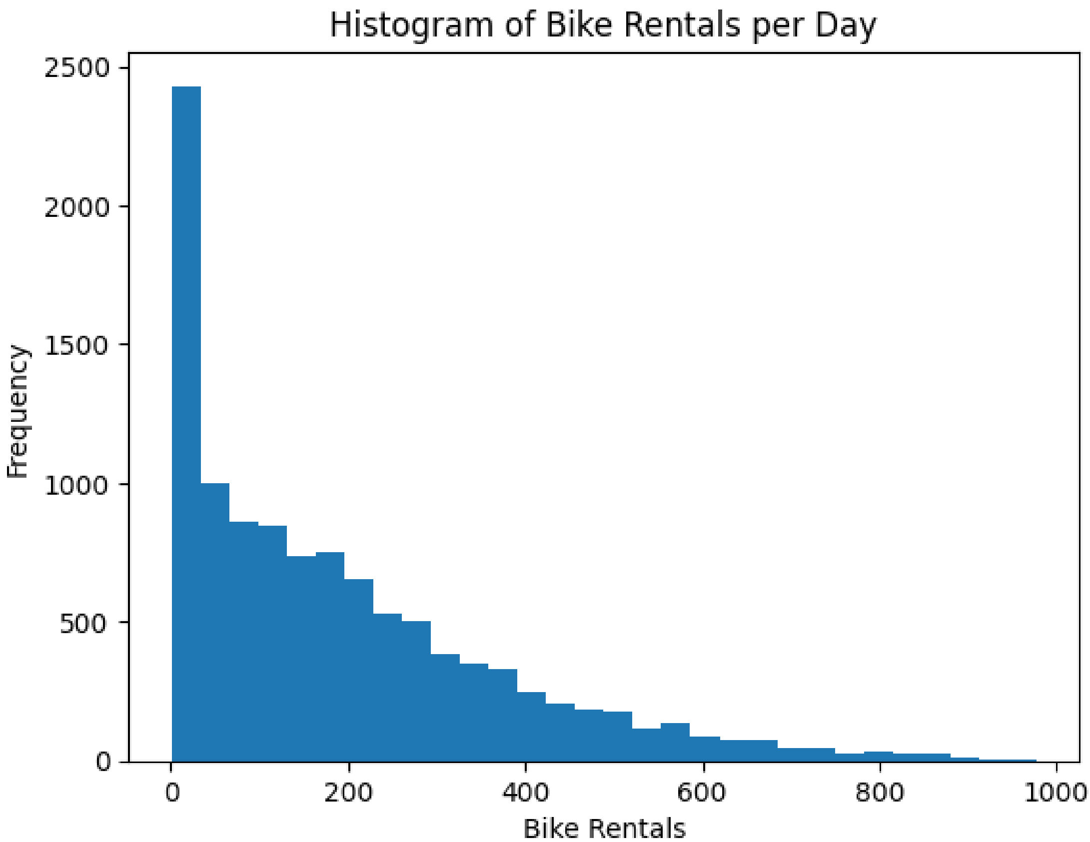
[50]:
```
# understanding the correlation between count and numerical variables
yulu.corr()['count']
```

[50]:
```
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
```

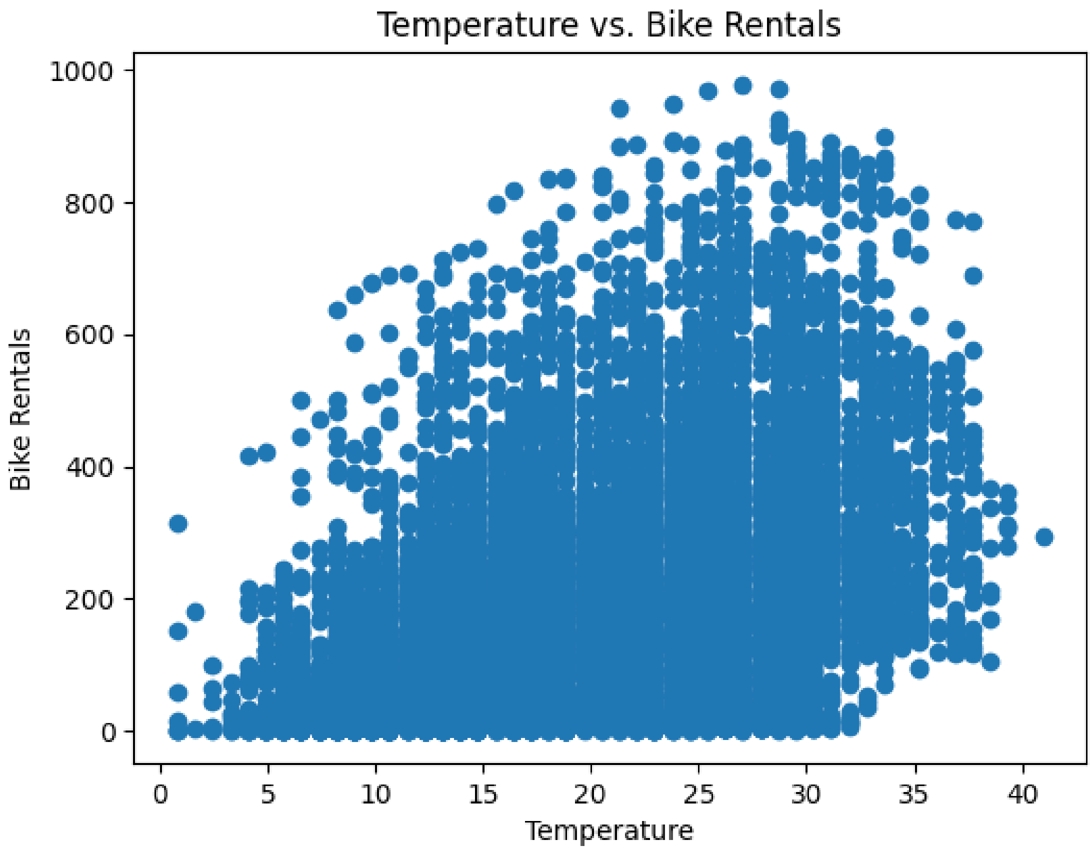
[51]:
```
sns.heatmap(yulu.corr(), annot=True)
plt.show()
```




[52]:
```
plt.hist(yulu['count'], bins=30)
plt.xlabel('Bike Rentals')
plt.ylabel('Frequency')
plt.title('Histogram of Bike Rentals per Day')
plt.show()
```




[53]:
```
# Creating a scatter plot of temperature vs. bike rentals
plt.scatter(yulu['temp'], yulu['count'])
plt.xlabel('Temperature')
plt.ylabel('Bike Rentals')
plt.title('Temperature vs. Bike Rentals')
plt.show()
```




## 8 Hypothesis Testing

[54]:
```
#Select an appropriate test to check whether:
#Working Day has effect on number of electric cycles rented
```

## 9 2- Sample T-Test to check if Working Day has an effect on the number of electric cycles rented

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

[55]:
```
data_group1 = yulu[yulu['workingday']==0]['count'].values
data_group2 = yulu[yulu['workingday']==1]['count'].values
```

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

[55]: (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

[56]:
```
stats.ttest_ind(a=data_group1, b=data_group2, equal_var=True)
```

[56]: Ttest_indResult(statistic=-1.2096277376026694, pvalue=0.22644804226361348)

[57]:
```
# Split the data into working days and non-working days
working_day = yulu[yulu['workingday'] == 1]['count']
non_working_day = yulu[yulu['workingday'] == 0]['count']

# Perform a 2-sample t-test
t_statistic, p_value = stats.ttest_ind(working_day, non_working_day)

# Print the results
print("T-Statistic:", t_statistic)
print("P-Value:", p_value)

if p_value < 0.05:
    print("Reject the null hypothesis")
else:
    print("Fail to reject the null hypothesis")
```

```
T-Statistic: 1.2096277376026694
P-Value: 0.22644804226361348
Fail to reject the null hypothesis
```

Since pvalue is greater than 0.05 so we can not reject the Null hypothesis. We don't have the sufficient evidence to say that working day has effect on the number of cycles being rented.

In this code, we extract the 'count' values for working days and non-working days using boolean indexing with the 'workingday' column, which takes a value of 1 for working days and 0 for non-working days. The t-test is then performed on these two sets of values to test whether there is a significant difference in the number of bike rentals between working days and non-working days.

## 10 No. of cycles rented similar or different in different seasons

Null Hypothesis: Number of cycles rented is similar in different season.

Alternate Hypothesis: Number of cycles rented is not similar in different season.

Significance level (alpha): 0.05

Here, we will use the ANOVA to test the hypothesis defined

[58]:
```
se1 = yulu[yulu['season']==1]['count'].values
se2 = yulu[yulu['season']==2]['count'].values
se3 = yulu[yulu['season']==3]['count'].values
se4 = yulu[yulu['season']==4]['count'].values
```

[59]:
```
# conduct the one-way anova
stats.f_oneway(se1,se2,se3,se4)
```

[59]:
```
F_onewayResult(statistic=236.94671081032106, pvalue=6.164843386499654e-149)
```

Since p-value is more than 0.05, we fail to reject the null hypothesis. This implies that Number of cycles rented is similar in different season conditions

## 11 No. of cycles rented similar or different in different weather

Null Hypothesis: Number of cycles rented is similar in different weather.

Alternate Hypothesis: Number of cycles rented is not similar in different weather.

Significance level (alpha): 0.05

Here, we will use the ANOVA to test the hypothesis defined

[60]:
```
# defining the data groups for the ANOVA

we1 = yulu[yulu['weather']==1]['count'].values
we2 = yulu[yulu['weather']==2]['count'].values
we3 = yulu[yulu['weather']==3]['count'].values
we4 = yulu[yulu['weather']==4]['count'].values
```

[61]:
```
stats.f_oneway(we1,we2,we3,we4)
```

[61]:
```
F_onewayResult(statistic=65.53024112793271, pvalue=5.482069475935669e-42)
```

Since p-value is more than 0.05, we fail to reject the null hypothesis. This implies that Number of cycles rented is similar in different weather conditions

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

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.

[62]:
```
data_table = pd.crosstab(yulu['season'], yulu['weather'])
print("Observed values:")
data_table
```

```
Observed values:
```

[62]:

| 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 |

[63]:
```
val = stats.chi2_contingency(data_table)
expected_values = val[3]
expected_values
```

[63]:
```
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]])
```

[64]:
```
nrows, ncols = 4, 4
dof = (nrows-1)*(ncols-1)
print("degrees of freedom: ", dof)
alpha = 0.05
```

```
degrees of freedom:  9
```

[65]:
```
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("\nSince p-value is less than the alpha 0.05, We reject the Null␣
↪Hypothesis. Meaning that\
    Weather is dependent on the season.")
else:
    print("Since p-value is greater than the alpha 0.05, We do not reject the␣
↪Null Hypothesis")
```

```
chi-square test statistic:  44.09441248632364
```

```
critical value: 16.918977604620448
p-value: 1.3560001579371317e-06

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

## 13 Insights

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.

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.

## 14 Recommendations

In summer and fall seasons the company should have more bikes in stock to be rented. Because the demand in these seasons is higher as compared to other seasons.

With a significance level of 0.05, workingday has no effect on the number of bikes being rented.

In very low humid days, company should have less bikes in the stock to be rented.

Whenever temprature is less than 10 or in very cold days, company should have less bikes.

Whenever the windspeed is greater than 35 or in thunderstorms, company should have less bikes in stock to be rented.

Increase the number of bikes available during peak hours: Based on the number of rentals during certain hours, it may be worthwhile to increase the number of bikes available during those times to ensure that there are enough bikes to meet demand.

Analyze the effect of weather on bike rentals: Since weather conditions can have a significant impact on bike rentals, it may be worth analyzing the relationship between weather and rental numbers to optimize operations accordingly.

For example, if it's found that rentals drop significantly during rainy days, it may be worthwhile to implement promotions or other incentives to encourage more rentals during those times.

Improve user experience: The number of rentals may be increased by improving the user experience, such as by improving bike availability, optimizing pricing, and implementing promotions or incentives. Additionally, the analysis of user feedback and usage patterns can help identify areas where improvements can be made to the overall experience.

Improve bike maintenance during low usage periods: Since the number of rentals drops significantly during some periods, it may be possible to conduct bike maintenance during those times without disrupting service. This can help ensure that bikes are in good condition and ready for use during peak hours.

[65]:
```

```