```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import math

#Loading the dataframe into a variable
df = pd.read_csv('/content/drive/MyDrive/SCALER/BUSINESS CASES/YULU/bike_sharing.csv')
```

# PROBLEM STETEMENT

**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.

**Problem Statement:**

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

# INITIAL OBSERVATION ON THE DATASET

```python
#This tells the number of rows & columns present in the dataframe. Totally there are 10886 rows  & 12 columns representing the electric cycle demands on a hourly basis.
df.shape
```

```
(10886, 12)
```

```python
df.head()
```

| | 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 |

```
# This gives the summary of a DataFrame's structure with the type of the data ecach column has.
# We can see there are a total of 10886 entries for in the dataframe.
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
```

```
#This gives the unique value present in each of the columns in our dataset.
df.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
dtype: int64
```

```
#Date Range during which the data was collected
print('Start Date: ', df['datetime'].min())
print('End Date: ', df['datetime'].max())

Start Date:  2011-01-01 00:00:00
End Date:  2012-12-19 23:00:00
```

# DATA CLEANING AND PRE-PROCESSING

Data Cleaning: Finding out missing values and duplicate entries

```
df.isnull().sum()
#There are no missing values in our dataset.

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

np.any(df.duplicated())
#There are no duplicate entries in our dataset.

False
```

Data Pre-Processing

```
#Converting numerical values for weather & season to its respective
categorical values
def weather_func(x):
  if x == 1:
    return 'Clear'
  elif x == 2:
    return 'Mist'
  elif x == 3:
    return 'Light Rain'
  elif x == 4:
    return 'Heavy Rain'

def season_func(x):
```

```
  if x == 1:
    return 'Spring'
  elif x == 2:
    return 'Summer'
  elif x == 3:
    return 'Fall'
  elif x == 4:
    return 'Winter'

df['season'] = df['season'].apply(season_func)
df['weather'] = df['weather'].apply(weather_func)

#Updating datatypes for few columns
df['datetime'] = pd.to_datetime(df['datetime'])
df['holiday'] = df['holiday'].astype('object')
df['workingday'] = df['workingday'].astype('object')

#Data types of the columns after the update
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  datetime64[ns]
 1   season      10886 non-null  object
 2   holiday     10886 non-null  object
 3   workingday  10886 non-null  object
 4   weather     10886 non-null  object
 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: datetime64[ns](1), float64(3), int64(4), object(4)
memory usage: 1020.7+ KB

#This gives the statistical summary for the numerical data in the
dataframe.
df.describe(include=np.number)

# The min temperature observed was  degree Celsius and maximum was
degree Celsius.
# The maximum humidity and windspeed observed were 100 and 56.9
respectively.
```

| | temp | atemp | humidity | windspeed | casual | registered | count |
|---|---|---|---|---|---|---|---|
| count | 10886.00000 | 10886.000000 | 10886.000000 | 10886.000000 | 10886.000000 | 10886.000000 | 10886.000000 |
| mean | 20.23086 | 23.655084 | 61.886460 | 12.799395 | 36.021955 | 155.552177 | 191.574132 |
| std | 7.79159 | 8.474601 | 19.245033 | 8.164537 | 49.960477 | 151.039033 | 181.144454 |
| min | 0.82000 | 0.760000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 |
| 25% | 13.94000 | 16.665000 | 47.000000 | 7.001500 | 4.000000 | 36.000000 | 42.000000 |
| 50% | 20.50000 | 24.240000 | 62.000000 | 12.998000 | 17.000000 | 118.000000 | 145.000000 |
| 75% | 26.24000 | 31.060000 | 77.000000 | 16.997900 | 49.000000 | 222.000000 | 284.000000 |
| max | 41.00000 | 45.455000 | 100.000000 | 56.996900 | 367.000000 | 886.000000 | 977.000000 |

```
#This gives the statistical summary for the categorical data in the
dataframe.
df.describe(include='object')

# Most occurred Season value is Winter and in weather is Clear.
```

| | season | holiday | workingday | weather |
|---|---|---|---|---|
| count | 10886 | 10886 | 10886 | 10886 |
| unique | 4 | 2 | 2 | 4 |
| top | Winter | 0 | 1 | Clear |
| freq | 2734 | 10575 | 7412 | 7192 |

# Exploratory Data Analysis

**Descriptive Analysis**

```
#Distribution of Season in our dataset
df.season.value_counts()
```

```
season
Winter    2734
Summer    2733
Fall      2733
Spring    2686
Name: count, dtype: int64
```

```
#Distribution of Holiday in our dataset
df.holiday.value_counts()
```

```
holiday
0    10575
1      311
Name: count, dtype: int64
```

```
#Distribution of Working Day in our dataset
df.workingday.value_counts()
```

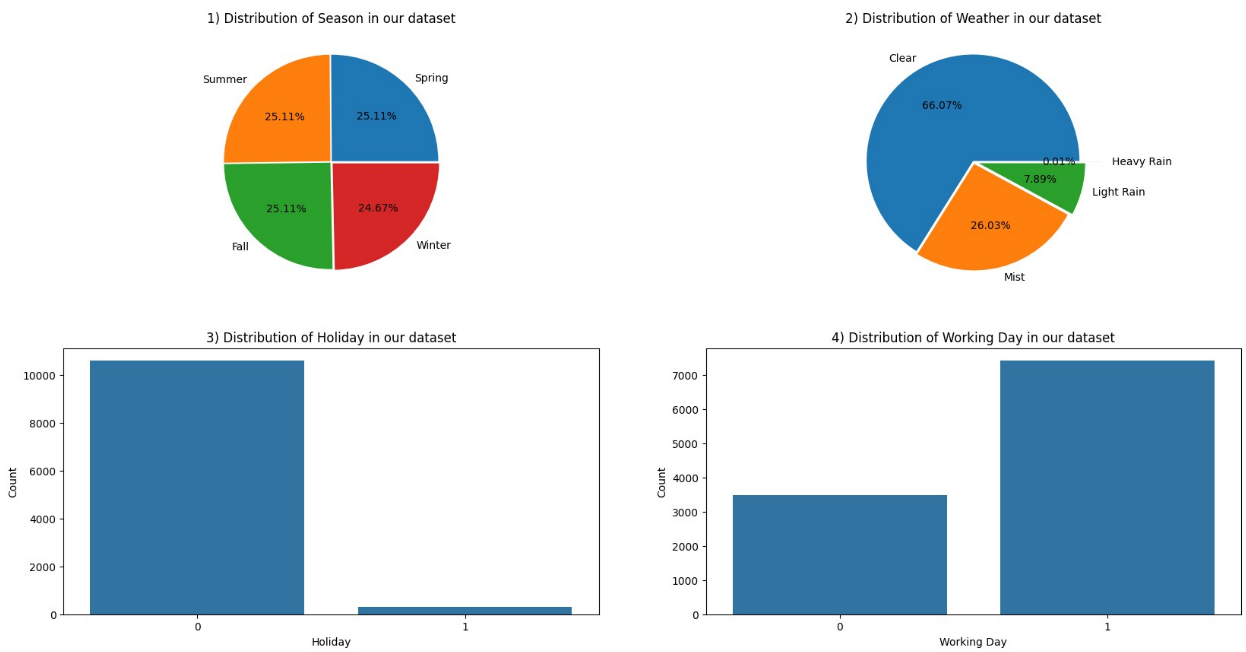
```
workingday
1    7412
0    3474
Name: count, dtype: int64
```

```
#Distribution of Weather in our dataset
df.weather.value_counts()
```

```
weather
Clear         7192
Mist          2834
Light Rain     859
Heavy Rain       1
Name: count, dtype: int64
```

```
fig, axes = plt.subplots(2,2,figsize=(20,10))

#1) Univariate Analysis - Distribution of Season in our dataset
axes[0,0].pie(x=df['season'].value_counts(),labels=df['season'].unique
(),autopct='%.2f%%',explode=[0.01,0.01,0.01,0.02])
axes[0,0].set_title('1) Distribution of Season in our dataset')
```

```
#2) Univariate Analysis - Distribution of Weather in our dataset
axes[0,1].pie(x=df['weather'].value_counts(),labels=df['weather'].uniq
ue(),autopct='%.2f%%',explode=[0.01, 0.02, 0.05, 0.2])
axes[0,1].set_title('2) Distribution of Weather in our dataset')

#3) Univariate Analysis - Distribution of Holiday in our dataset
sns.countplot(x='holiday',data=df,ax=axes[1,0])
axes[1,0].set_xlabel('Holiday')
axes[1,0].set_ylabel('Count')
axes[1,0].set_title('3) Distribution of Holiday in our dataset')

#4) Univariate Analysis - Distribution of Working Day in our dataset
sns.countplot(x='workingday',ax=axes[1,1],data=df)
axes[1,1].set_xlabel('Working Day')
axes[1,1].set_ylabel('Count')
axes[1,1].set_title('4) Distribution of Working Day in our dataset')
#axes[1,0].tick_params(labelrotation=90,axis='x')



plt.show()

#Insights for these graphs:
#1) We can see that all the four seasons are almost equally
distributed.
#2) We can see most of the time it was a clear wather when users
rented the electric cycle.
#3) Distribution of Non-Holiday days is more in our dataset compared
to Holiday days.
#4) Distribution of Working Days is more in our dataset compared to
Non-Working Days.
```




```python
#Number of electric cycle rented during different Seasons
df.groupby('season')['count'].sum()
```

```
season
Fall      640662
Spring    312498
Summer    588282
Winter    544034
Name: count, dtype: int64
```

```python
#Number of electric cycle rented based on Weather Conidtions
df.groupby('weather')['count'].sum()
```

```
weather
Clear         1476063
Heavy Rain        164
Light Rain     102089
Mist           507160
Name: count, dtype: int64
```

```python
#Number of electric cycle rented on Holiday & Non-holiday Days
df.groupby('holiday')['count'].sum()
```

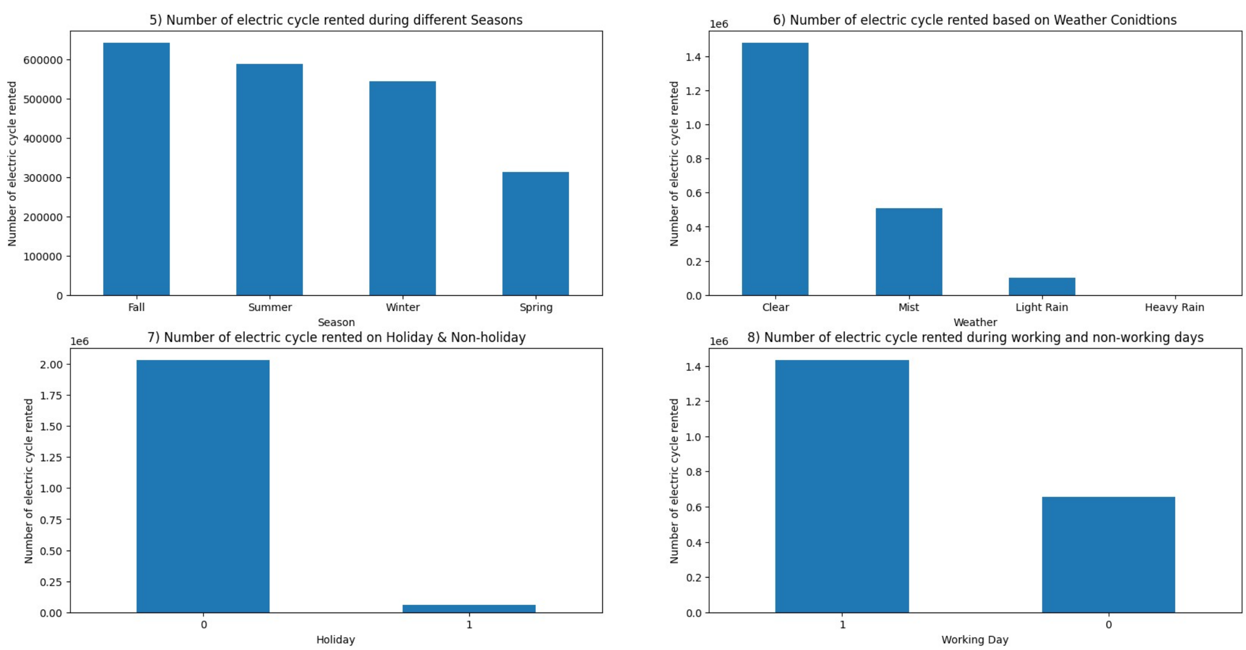
```
holiday
0    2027668
1      57808
Name: count, dtype: int64
```

```
#Number of electric cycle rented during working and non-working days
df.groupby('workingday')['count'].sum()
```

```
workingday
0     654872
1    1430604
Name: count, dtype: int64
```

```
fig, axes = plt.subplots(2,2,figsize=(20,10))

#5) Bivariate Analysis - Number of electric cycle rented during
different Seasons
df.groupby('season')
['count'].sum().sort_values(ascending=False).plot(kind='bar',ax=axes[0
,0])
axes[0,0].set_xlabel('Season')
axes[0,0].set_ylabel('Number of electric cycle rented')
axes[0,0].set_title('5) Number of electric cycle rented during
different Seasons')
axes[0,0].tick_params(labelrotation=0,axis='x')

#6) Bivariate Analysis - Number of electric cycle rented based on
Weather Conidtions
df.groupby('weather')
['count'].sum().sort_values(ascending=False).plot(kind='bar',ax=axes[0
,1])
axes[0,1].set_xlabel('Weather')
axes[0,1].set_ylabel('Number of electric cycle rented')
axes[0,1].set_title('6) Number of electric cycle rented based on
Weather Conidtions')
axes[0,1].tick_params(labelrotation=0,axis='x')

#7) Bivariate Analysis - Number of electric cycle rented on Holiday &
Non-holiday
df.groupby('holiday')
['count'].sum().sort_values(ascending=False).plot(kind='bar',ax=axes[1
,0])
axes[1,0].set_xlabel('Holiday')
axes[1,0].set_ylabel('Number of electric cycle rented')
axes[1,0].set_title('7) Number of electric cycle rented on Holiday &
```

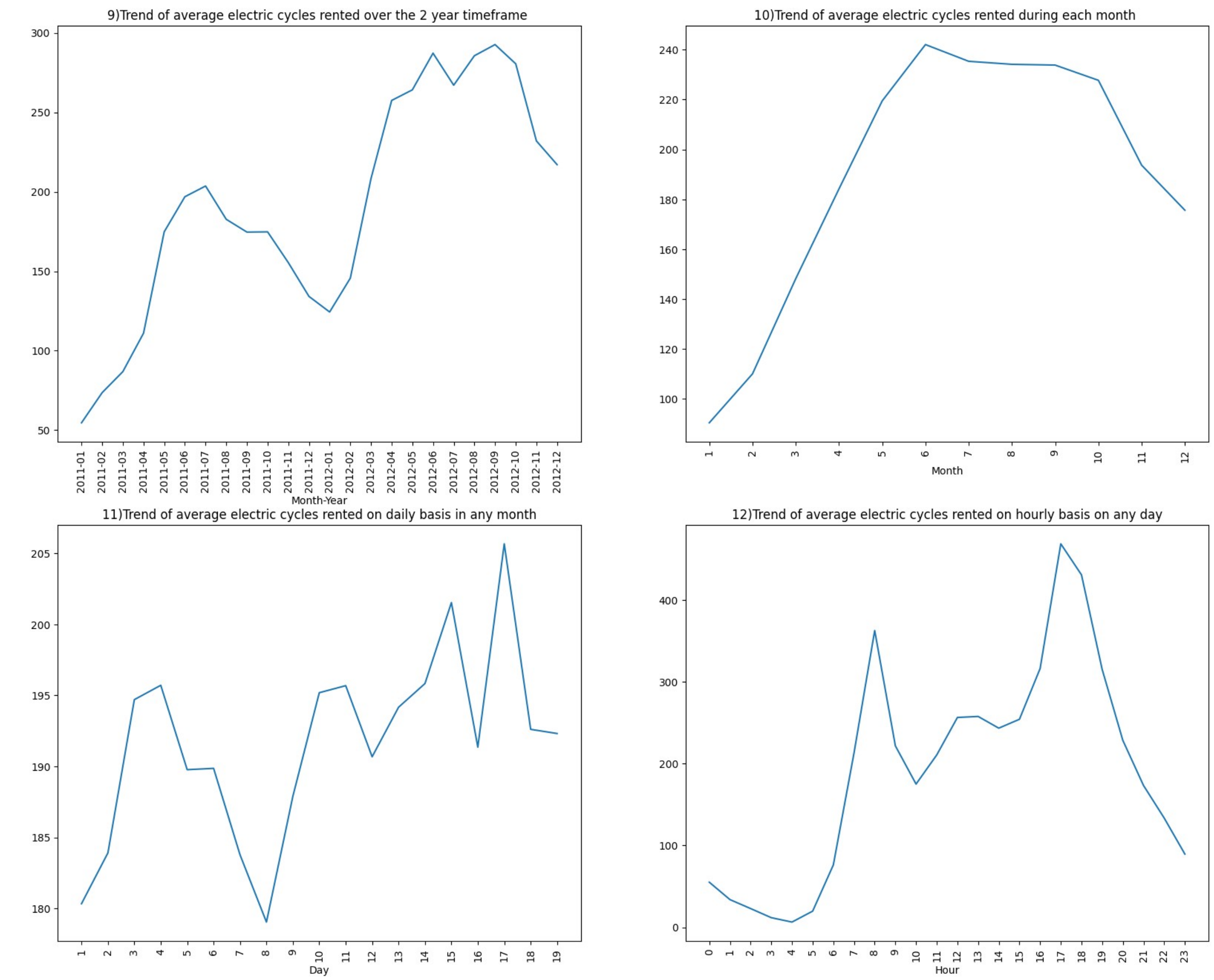
```
Non-holiday')
axes[1,0].tick_params(labelrotation=0,axis='x')

#8) Bivariate Analysis - Number of electric cycle rented during
working and non-working days
df.groupby('workingday')
['count'].sum().sort_values(ascending=False).plot(kind='bar',ax=axes[1
,1])
axes[1,1].set_xlabel('Working Day')
axes[1,1].set_ylabel('Number of electric cycle rented')
axes[1,1].set_title('8) Number of electric cycle rented during working
and non-working days')
axes[1,1].tick_params(labelrotation=0,axis='x')



plt.show()

#Insights for these graphs:
#5) More Number of electric cycle were rented during Fall Seasons.
#6) More Number of electric cycle were rented during a Clear Weather
Conidtion.
#7) Maximum users rented the electric cycle during the Non Holiday
Days.
#8) Maximum users rented the electric cycle to commute to their office
that is during the Working Days.
```




```
# Adding few date columns for durther analysis
df['Month-Year'] = df['datetime'].dt.strftime('%Y-%m')
df['Month'] = df['datetime'].dt.month
```

```
df['Day'] = df['datetime'].dt.day
df['Hour'] = df['datetime'].dt.hour

fig, axes = plt.subplots(2,2,figsize=(20,16))

#9) Bivariate Analysis - Trend of average electric cycles rented over
the months in 2011 & 2012.
month_year_count = df.groupby('Month-Year')['count'].mean()
sns.lineplot(x=month_year_count.index,
y=month_year_count.values,ax=axes[0,0])
axes[0,0].tick_params(labelrotation=90,axis='x')
axes[0,0].set_title('9)Trend of average electric cycles rented over
the 2 year timeframe ')

#10) Bivariate Analysis - Trend of average electric cycles rented
during each month.
month_count = df.groupby('Month')['count'].mean()
sns.lineplot(x=month_count.index, y=month_count.values,ax=axes[0,1])
axes[0,1].tick_params(labelrotation=90,axis='x')
axes[0,1].set_xticks(range(1,13,1))
axes[0,1].set_title('10)Trend of average electric cycles rented during
each month ')

#11) Bivariate Analysis - Trend of average electric cycles rented on
daily basis in any month.
day_count = df.groupby('Day')['count'].mean()
sns.lineplot(x=day_count.index,y=day_count.values,ax=axes[1,0])
axes[1,0].tick_params(labelrotation=90,axis='x')
axes[1,0].set_xticks(range(1,len(day_count.index)+1))
axes[1,0].set_title('11)Trend of average electric cycles rented on
daily basis in any month')

#12) Bivariate Analysis - Trend of average electric cycles rented on
hourly basis on any day.
hour_count = df.groupby('Hour')['count'].mean()
sns.lineplot(x=hour_count.index,y=hour_count.values,ax=axes[1,1])
axes[1,1].tick_params(labelrotation=90,axis='x')
axes[1,1].set_xticks(range(0,len(hour_count.index)))
axes[1,1].set_title('12)Trend of average electric cycles rented on
hourly basis on any day')

plt.show()


#Insights for these graphs:
#9) There is stable growth in the number of cycles rented over the
months with a spike between May and October month and a drop from
November month. The same trend is almost beign oberved for both the
years - 2011 & 2012.
#10) There is a linear growth in number of cycles rented from January
```

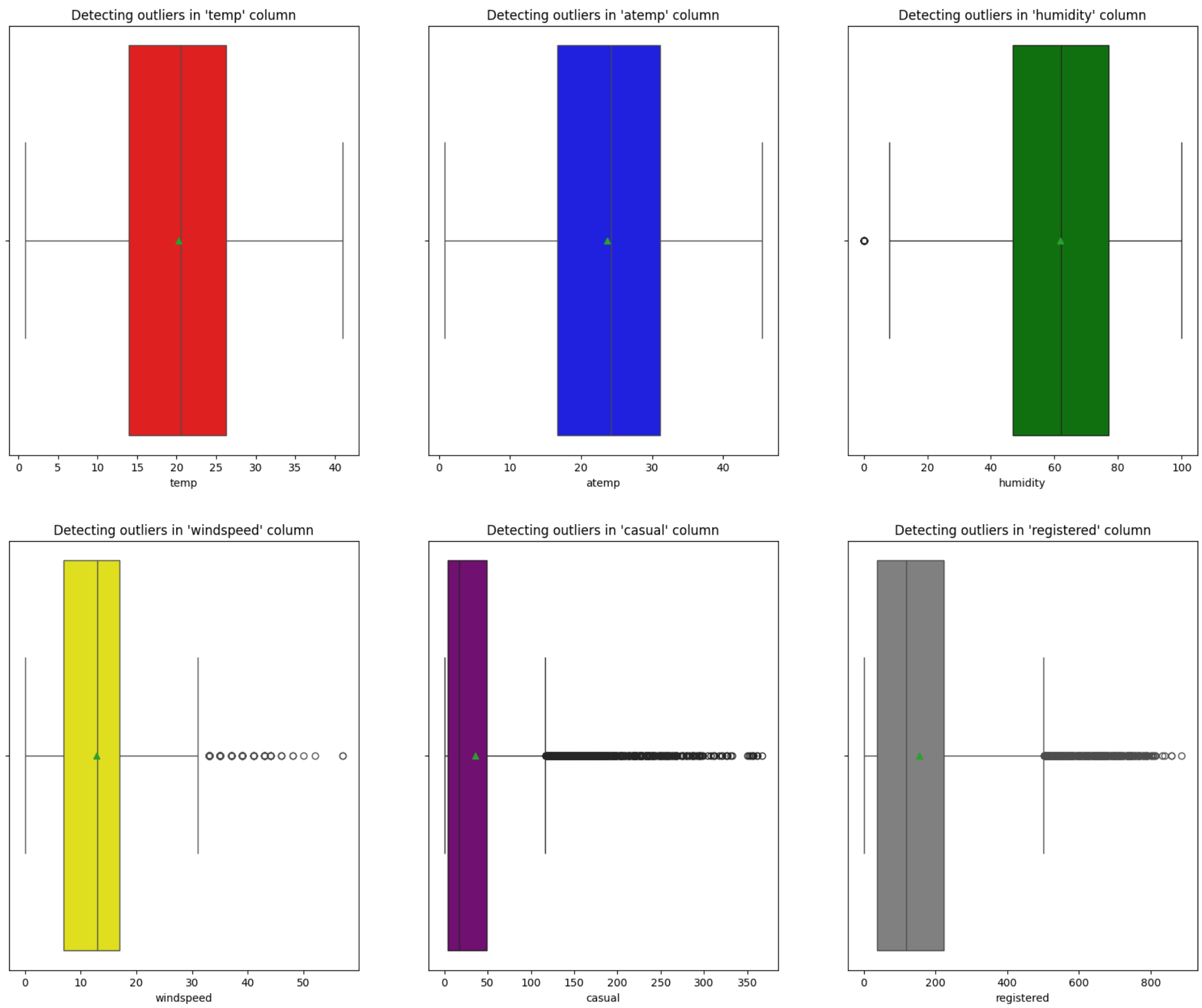
```
to June and then it is quite constant from June to October and then
dropped back from November.
#11) This shows the Trend of average electric cycles rented on daily
basis in any month. We can see that the data was collected mostly on
the first 19 days in any month.
#12) In any given day, on average more users rent the electric cycles
mainly between 7 AM to 9 AM in the Morning and 4 PM to 8 PM in the
Night.
```




# Outlier Detection using Boxplot

```
# Detecting Outliers in the numerical columns
columns = ['temp','atemp', 'humidity', 'windspeed', 'casual',
'registered']
colors =['red', 'blue', 'green', 'yellow', 'purple', 'gray']
count = 1
plt.figure(figsize = (20, 16))
```

```
for i in columns:
    plt.subplot(2, 3, count)
    plt.title(f"Detecting outliers in '{i}' column")
    sns.boxplot(data = df, x = df[i], color = colors[count - 1],
showmeans = True)
    plt.plot()
    count += 1

#Insights:
#There is no outlier in the temp and atemp columns.
#There are few outliers present in humidity and windspeed columns.
#There are more outliers present in casual and registered columns.
```

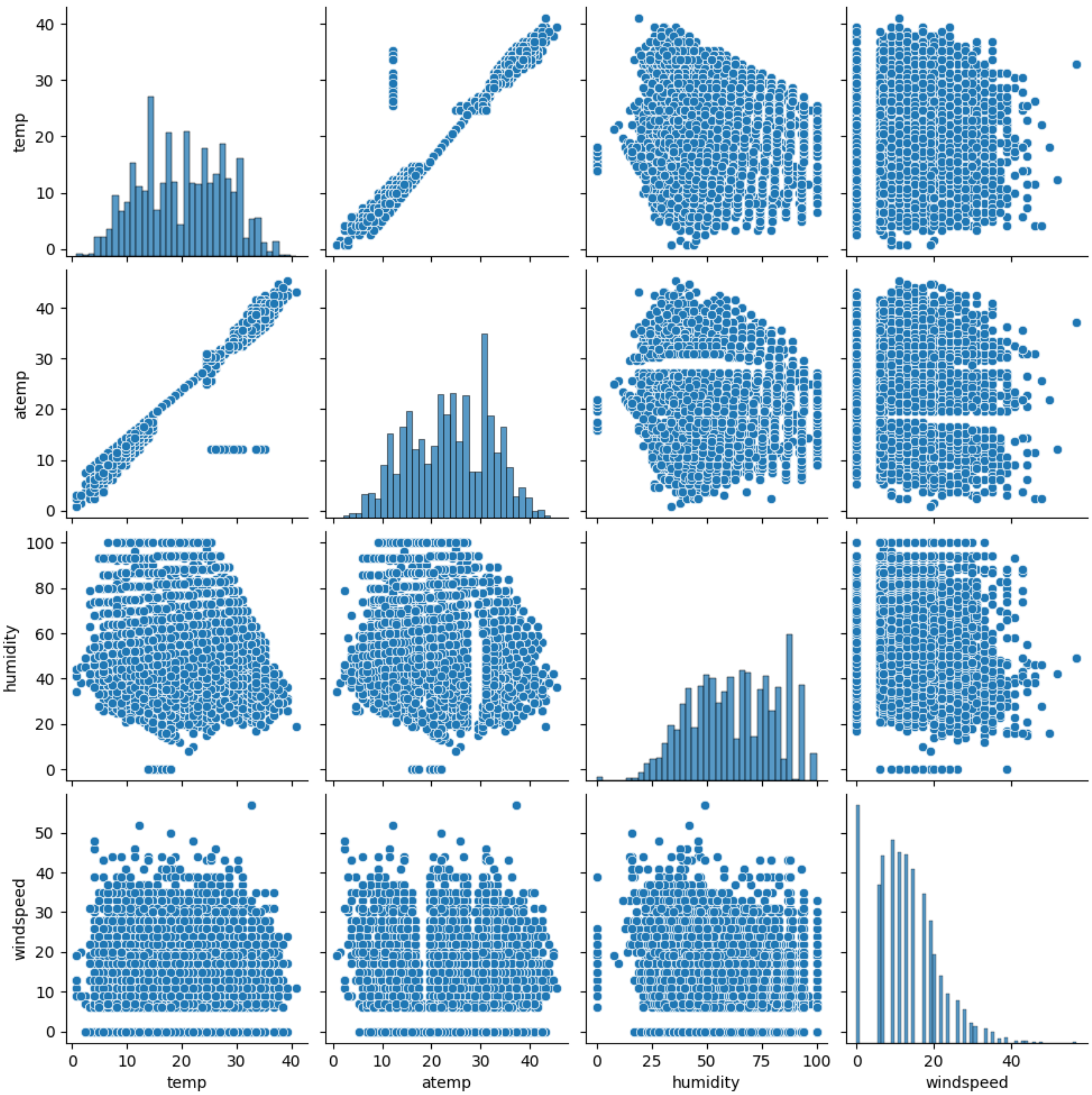


# Co-Relation, PairPlot and HeatMaps

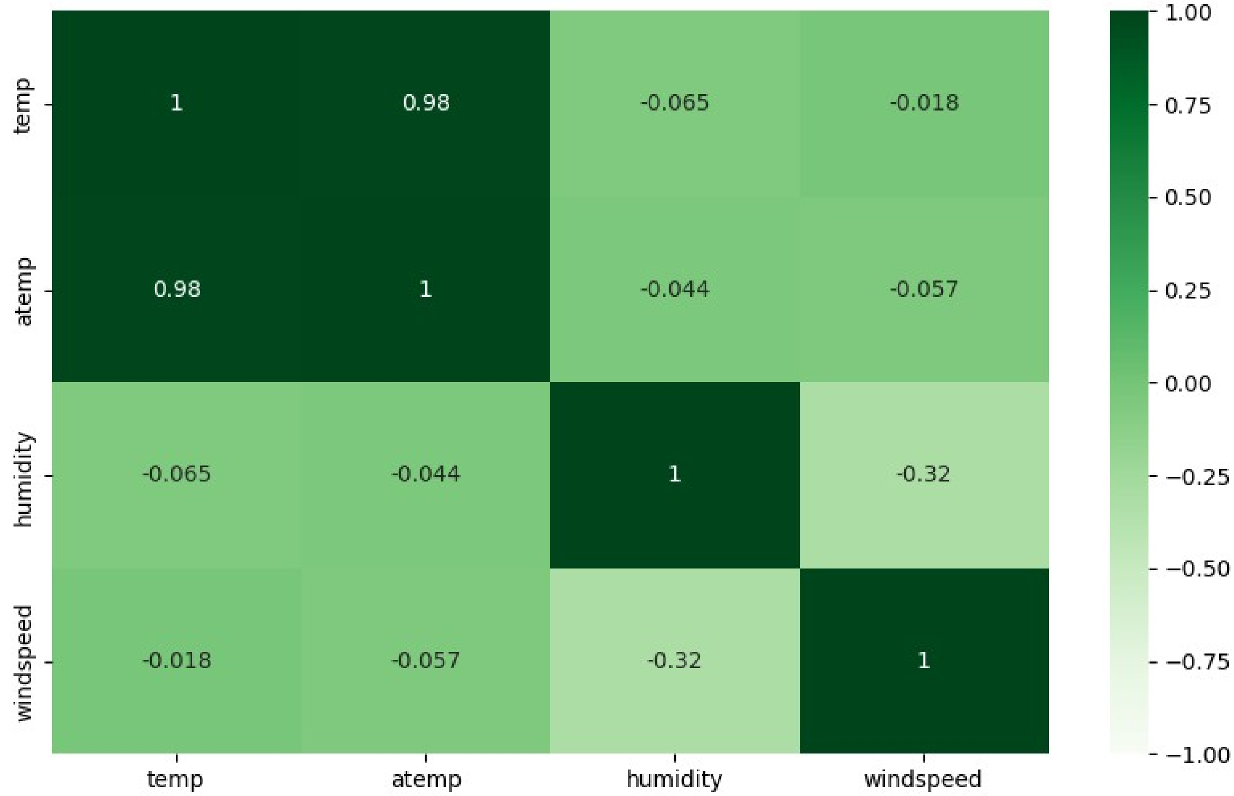
```
#Co-Relation: Measures the relationship between 2 numerical columns in the dataframe.

df[['temp','atemp','humidity','windspeed']].corr()

# temp and atemp seems to be highly correlated this is because both are measured in Celsius and almost have similar values.
# windspeed and humidity are negatively correlated
```

| | temp | atemp | humidity | windspeed |
|---|---|---|---|---|
| temp | 1.000000 | 0.984948 | -0.064949 | -0.017852 |
| atemp | 0.984948 | 1.000000 | -0.043536 | -0.057473 |
| humidity | -0.064949 | -0.043536 | 1.000000 | -0.318607 |
| windspeed | -0.017852 | -0.057473 | -0.318607 | 1.000000 |

```
#PairPlot - Plots the relationshipt between the numerical variables in the dataframe.
sns.pairplot(df[['temp','atemp','humidity','windspeed']])

<seaborn.axisgrid.PairGrid at 0x7f9caab208b0>
```




```
# HeatMap to visualize the co-relation between 2 numerical variables
in the dataframe
plt.figure(figsize = (10, 6))

cr = df[['temp','atemp','humidity','windspeed']].corr()
sns.heatmap(data = cr, cmap = 'Greens', annot = True, vmin = -1, vmax
= 1)
plt.plot()

[]
```




# HYPOTHESIS TESTING

**Q1) check if Working Day has an effect on the number of electric cycles rented**

Step 1)

Ho: The average cycles rented on working(1) and non-working(0) days are same(Working Day has no effect on number of cycles rented).

Ha: The average cycles rented on working(1) and non-working(0) days are different(Working Day has effect on number of cycles rented)

Step 2)

Test Statistic/Distribution: We use T-Test Statistic here as the sample size of both the group is large and hence T-Test behaves like a Z-Test in this case. Before proceeding with T-Test we will check its assumption of normality & equality in variance. Once the assumptions are met we will proceed with next steps.

Step 3)

We will choose to perform two-tailed test here.

Step 4)

Find out the p-value and we will take the significance level as 5% which is the default value if it's not given and also this dataset is not a critical datset. So alpha = 0.05

Step 5)

Compare p-value with alpha.

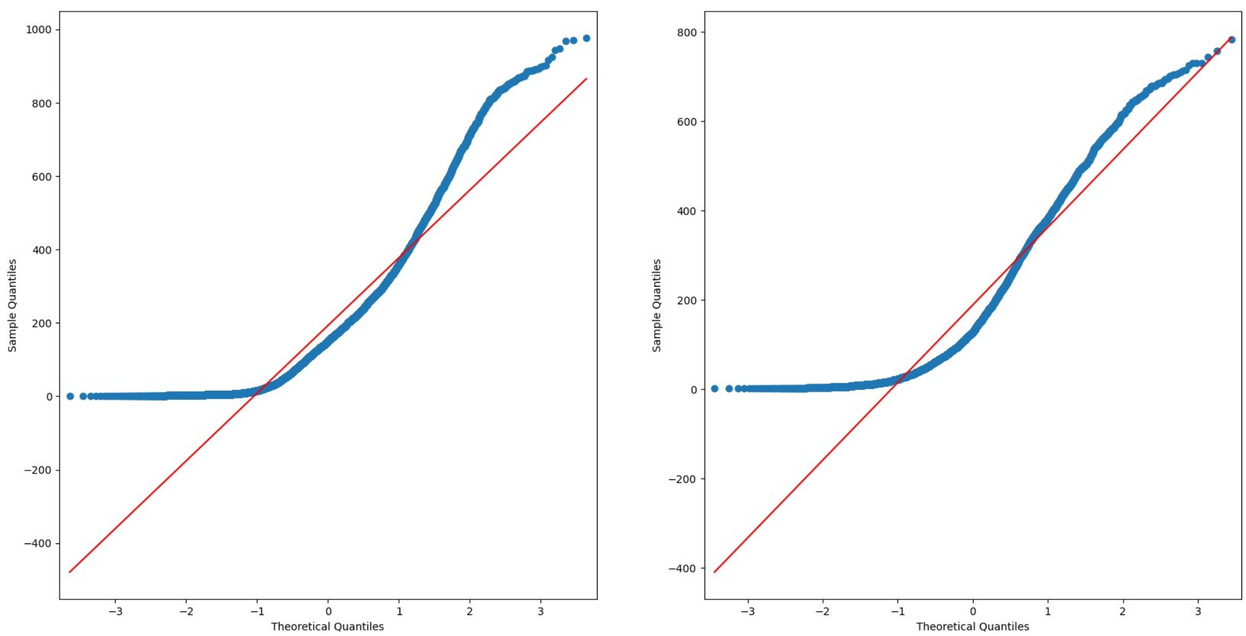
if p-value < alpha:

Reject Null Hypothesis(Ho)

else:

Fail to Reject Null Hypothesis(Ho)

```
from statsmodels.graphics.gofplots import qqplot
#Checking for the assumptions
#QQ Plot for checking normality
working_day = df[df['workingday'] == 1]['count']
non_working_day = df[df['workingday'] == 0]['count']

fig, axes = plt.subplots(1,2,figsize=(20,10))
qqplot(working_day, line='s', ax=axes[0])
qqplot(non_working_day, line='s', ax=axes[1])
plt.show()
```




```
from scipy.stats import levene
#Checking for equality in variance
#H0: The variances across the 2 groups are equal.
#Ha: The variances across the 2 groups are not equal.
l_stat, p_val = levene(working_day, non_working_day)
```

```python
alpha = 0.05
print('Test Statistic: ', l_stat)
print('p-value: ', p_val)

if p_val < alpha:
  print('Reject Null Hypothesis(Ho). The variances across the 2 groups are not equal')
else:
  print('Fail to Reject Null Hypothesis(Ho). The variances across the 2 groups are equal')
```

```
Test Statistic:  0.004972848886504472
p-value:  0.9437823280916695
Fail to Reject Null Hypothesis(Ho). The variances across the 2 groups are equal
```

```python
size_working_day = len(working_day)
size_non_working_day = len(non_working_day)
print('Size of Working Day: ', size_working_day)
print('Size of Non-Working Day: ', size_non_working_day)
```

```
Size of Working Day:  7412
Size of Non-Working Day:  3474
```

```python
# We can see the 2 groups does not have normal distribution but the variances across the 2 groups are equal. Since the sample sizes of both groups are very large even though they do not meet normality we can still go with Independent 2-sample T-Test
# because at large sample size T-test behaves as Z-Test and follows CLT. So we will do both Independent 2- Sample T-Test and Mann-Whitney U rank test
```

```python
from scipy.stats import mannwhitneyu
m_stat, pval = mannwhitneyu(working_day, non_working_day)
print('Test Statistic: ', m_stat)
print('p-value: ', pval)
if pval < alpha:
  print('Reject Null Hypothesis(Ho). The average cycles rented on working(1) and non-working(0) days are different(Working Day has effect on number of cycles rented)')
else:
  print('Fail to Reject Null Hypothesis(Ho). The average cycles rented on working(1) and non-working(0) days are same(Working Day has no effect on number of cycles rented)')
```

```
Test Statistic:  12868495.5
p-value:  0.9679139953914079
Fail to Reject Null Hypothesis(Ho). The average cycles rented on working(1) and non-working(0) days are same(Working Day has no effect on number of cycles rented)
```

```
from scipy.stats import ttest_ind
t_stat, pval = ttest_ind(working_day, non_working_day)
print('Test Statistic: ', t_stat)
print('p-value: ', pval)
if pval < alpha:
  print('Reject Null Hypothesis(Ho). The average cycles rented on
working(1) and non-working(0) days are different(Working Day has
effect on number of cycles rented)')
else:
  print('Fail to Reject Null Hypothesis(Ho). The average cycles rented
on working(1) and non-working(0) days are same(Working Day has no
effect on number of cycles rented)')
```

```
Test Statistic:  1.2096277376026694
p-value:  0.22644804226361348
Fail to Reject Null Hypothesis(Ho). The average cycles rented on
working(1) and non-working(0) days are same(Working Day has no effect
on number of cycles rented)
```

**Q2) Check if No. of cycles rented is similar or different in different weather conditions.**

Step 1)

Ho: The average number of cycles rented are same during different Weather conditions(Weather has no effect on number of cycles rented).

Ha: The average number of cycles rented are not same during different Weather conditions(Weather has effect on number of cycles rented).

Step 2)

Test Statistic/Distribution: We use one way ANOVA here as there are more than two groups in weather. Before proceeding with AOVA we will check its assumption of normality & equality in variance. Once the assumptions are met we will proceed with next steps.

Step 3)

We will choose to perform two-tailed test here.

Step 4)

Find out the p-value and we will consider alpha = 0.05

Step 5)

Compare p-value with alpha.

if p-value < alpha:

Reject Null Hypothesis(Ho)

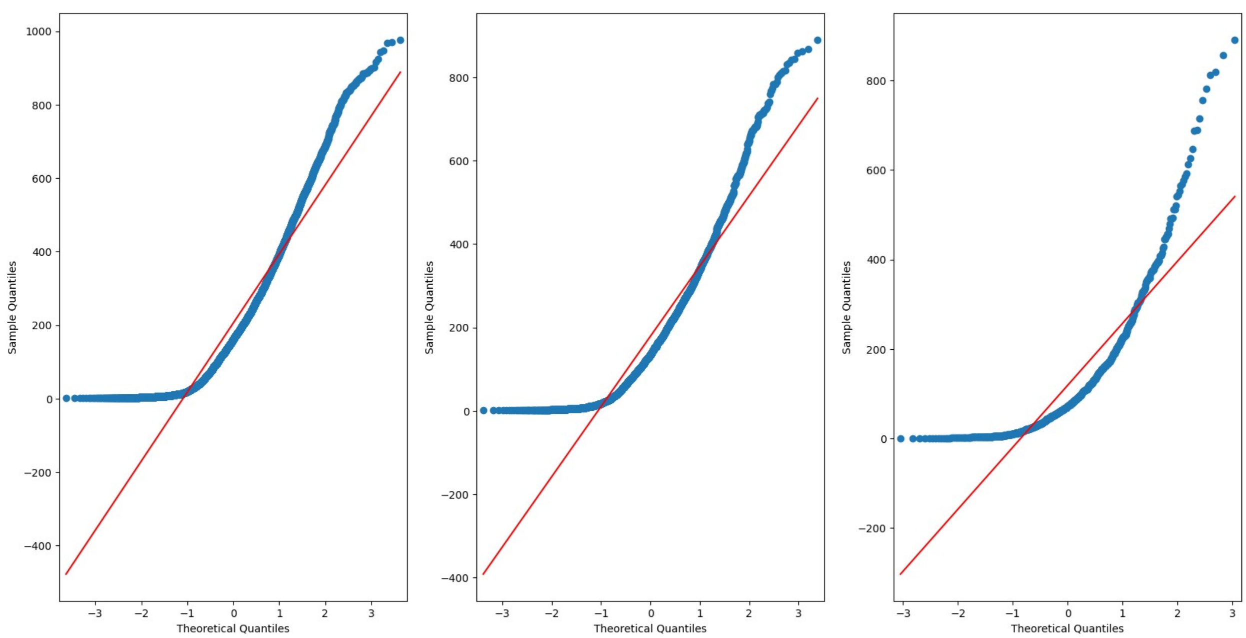
else:

Fail to Reject Null Hypothesis(Ho)

```
df.weather.value_counts() # We will not consider Heav Rain value when we do analysis on weather column as it has only 1 row
```

```
weather
Clear          7192
Mist           2834
Light Rain      859
Heavy Rain        1
Name: count, dtype: int64
```

```
from statsmodels.graphics.gofplots import qqplot
#Checking for the assumptions
#QQ Plot for checking normality
clear_weather = df[df['weather'] == 'Clear']['count']
mist_weather = df[df['weather'] == 'Mist']['count']
light_rain_weather = df[df['weather'] == 'Light Rain']['count']
heavy_rain_weather = df[df['weather'] == 'Heavy Rain']['count']

fig, axes = plt.subplots(1,3,figsize=(20,10))
qqplot(clear_weather, line='s', ax=axes[0])
qqplot(mist_weather, line='s', ax=axes[1])
qqplot(light_rain_weather, line='s', ax=axes[2])

plt.show()
```




```
from scipy.stats import levene
#Checking for equality in variance
#H0: The variances across the 2 groups are equal.
```

```
#Ha: The variances across the 2 groups are not equal.
l_stat, p_val = levene(clear_weather, mist_weather,
light_rain_weather)
alpha = 0.05
print('Test Statistic: ', l_stat)
print('p-value: ', p_val)

if p_val < alpha:
  print('Reject Null Hypothesis(Ho). The variances across the 2 groups
are not equal')
else:
  print('Fail to Reject Null Hypothesis(Ho). The variances across the
2 groups are equal')
```

```
Test Statistic:  81.67574924435011
p-value:  6.198278710731511e-36
Reject Null Hypothesis(Ho). The variances across the 2 groups are not
equal
```

```
# We can see that the 3 groups are not normally distributed and also
as per the levene's test atleast one of the three groups has a
different variance and hence the assumptions of ANOVA are not met.
# So we need to perform non-parametric test and we will use Kruskal-
Wallis test here.

#Ho: The median of number of cycles rented are same during different
Weather conditions.Weather has no effect on the median number of
cycles rented.

#Ha: The median of number of cycles rented are not same during
different Weather conditions.Weather has effect on the median number
of cycles rented.

from scipy.stats import kruskal
k_stat, pval = kruskal(clear_weather, mist_weather,
light_rain_weather)
print('Test Statistic: ', k_stat)
print('p-value: ', pval)

if pval < alpha:
  print('Reject Null Hypothesis(Ho). The median of number of cycles
rented are not same during different Weather conditions.Weather has
effect on the median number of cycles rented.')
else:
  print('Fail to Reject Null Hypothesis(Ho). The median of number of
cycles rented are same during different Weather conditions.Weather has
no effect on the median number of cycles rented')
```

```
Test Statistic:  204.95566833068537
p-value:  3.122066178659941e-45
Reject Null Hypothesis(Ho). The median of number of cycles rented are
```

```
not same during different Weather conditions.Weather has effect on the
median number of cycles rented.
```

**Q3 Check if No. of cycles rented is similar or different in different Seasons.**

Step 1)

Ho: The average number of cycles rented are same during different Seasons (Weather has no effect on number of cycles rented).

Ha: The average number of cycles rented are not same during different Seasons (Weather has effect on number of cycles rented).

Step 2)

Test Statistic/Distribution: We use one way ANOVA here as there are more than two groups in weather. Before proceeding with AOVA we will check its assumption of normality & equality in variance. Once the assumptions are met we will proceed with next steps.

Step 3)

We will choose to perform two-tailed test here.

Step 4)

Find out the p-value and we will consider alpha = 0.05

Step 5)

Compare p-value with alpha.

if p-value < alpha:

Reject Null Hypothesis(Ho)

else:

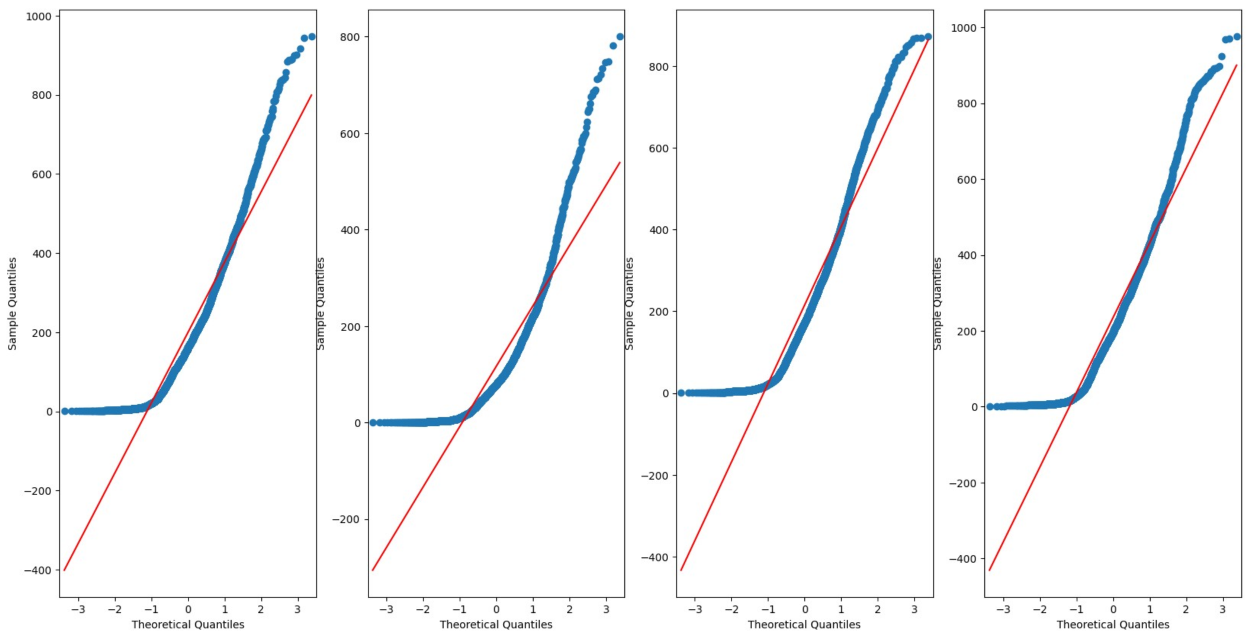
Fail to Reject Null Hypothesis(Ho)

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

```
season
Winter    2734
Summer    2733
Fall      2733
Spring    2686
Name: count, dtype: int64
```

```
#Checking for the assumptions
#QQ Plot for checking normality
```

```
winter_season = df[df['season'] == 'Winter']['count']
spring_season = df[df['season'] == 'Spring']['count']
summer_season = df[df['season'] == 'Summer']['count']
fall_season = df[df['season'] == 'Fall']['count']

fig, axes = plt.subplots(1,4,figsize=(20,10))
qqplot(winter_season, line='s', ax=axes[0])
qqplot(spring_season, line='s', ax=axes[1])
qqplot(summer_season, line='s', ax=axes[2])
qqplot(fall_season, line='s', ax=axes[3])

plt.show()
```




```
#Checking for equality in variance
#H0: Variances are equal across groups.
#Ha: Variances are not equal across groups.
l_stat, p_val = levene(winter_season, spring_season, summer_season,
fall_season)
alpha = 0.05
print('Test Statistic: ', l_stat)
print('p-value: ', p_val)

if p_val < alpha:
  print('Reject Null Hypothesis(Ho). Variances are not equal across
groups')
else:
  print('Fail to Reject Null Hypothesis(Ho). Variances are equal
across groups')
```

```
Test Statistic:  187.7706624026276
p-value:  1.0147116860043298e-118
Reject Null Hypothesis(Ho). Variances are not equal across groups

# We can see that the 4 groups are not normally distributed and also
as per the levene's test atleast one of the three groups has a
different variance and hence the assumptions of ANOVA are not met.
# So we need to perform non-parametric test and we will use Kruskal-
Wallis test here.

#Ho: The median of number of cycles rented are same during different
Seasons conditions.Season has no effect on the median number of cycles
rented.

#Ha: The median of number of cycles rented are not same during
different Seasons conditions.Season has effect on the median number of
cycles rented.

k_stat, pval = kruskal(winter_season, spring_season, summer_season,
fall_season)
print('Test Statistic: ', k_stat)
print('p-value: ', pval)

if pval < alpha:
  print('Reject Null Hypothesis(Ho). The median of number of cycles
rented are not same during different Seasons conditions.Season has
effect on the median number of cycles rented.')
else:
  print('Fail to Reject Null Hypothesis(Ho). The median of number of
cycles rented are same during different Seasons conditions.Season has
no effect on the median number of cycles rented.')

Test Statistic:  699.6668548181988
p-value:  2.479008372608633e-151
Reject Null Hypothesis(Ho). The median of number of cycles rented are
not same during different Seasons conditions.Season has effect on the
median number of cycles rented.
```

**Q4) Is there a significant difference in the number of cycles rented considering both weather conditions and seasons simultaneously?**

Main Effects:

1) Weather vs number of cycles rented.

Ho: The mean number of cycles rented is the same across all weather conditions. Ha: The mean number of cycles rented differs across weather conditions.

2) Season vs number of cycles rented.

Ho: The mean number of cycles rented is the same across all seasons. Ha: The mean number of cycles rented differs across seasons.

Interaction effect:

1) Effect of Both Weather and Season on number of cycles rented.

Ho: There is no interaction effect between weather and season on the mean number of cycles rented (i.e., the effect of weather is consistent across all seasons and vice-versa).

Ha: There is an interaction effect between weather and season on the mean number of cycles rented (i.e., the effect of weather varies depending on the season and vice-versa).

```
df[['weather' , 'season']].value_counts()
```

```
weather     season
Clear       Fall      1930
            Summer    1801
            Spring    1759
            Winter    1702
Mist        Winter     807
            Spring     715
            Summer     708
            Fall       604
Light Rain  Winter     225
            Summer     224
            Spring     211
            Fall       199
Heavy Rain  Spring       1
Name: count, dtype: int64
```

```
from scipy.stats import shapiro
weathers  = [w for w in df['weather'].unique() if w != 'Heavy Rain']

for weather in weathers:
    for season in df['season'].unique():
        df_ws = df[(df['weather'] == weather) & (df['season'] ==
season)]
        stat, p_value = shapiro(df_ws['count'])

        if p_value < alpha:
            print(f'{weather}-{season} Group does not follow Normal
Distribution')
            print('-----------')
        else:
            print(f'{weather}-{season} Group follows Normal
Distribution')
            print('-----------')
```

```
Clear-Spring Group does not follow Normal Distribution
-----------
Clear-Summer Group does not follow Normal Distribution
-----------
Clear-Fall Group does not follow Normal Distribution
-----------
Clear-Winter Group does not follow Normal Distribution
-----------
Mist-Spring Group does not follow Normal Distribution
-----------
Mist-Summer Group does not follow Normal Distribution
-----------
Mist-Fall Group does not follow Normal Distribution
-----------
Mist-Winter Group does not follow Normal Distribution
-----------
Light Rain-Spring Group does not follow Normal Distribution
-----------
Light Rain-Summer Group does not follow Normal Distribution
-----------
Light Rain-Fall Group does not follow Normal Distribution
-----------
Light Rain-Winter Group does not follow Normal Distribution
-----------
```

```
weather_season_groups = [df[(df['weather'] == weather) & (df['season']
== season)]['count']
                         for weather in weathers
                         for season in df['season'].unique()]

stat, p_value = levene(*weather_season_groups)
print('Test Statistic: ', stat)
print('p-value: ', p_value)

if p_value < alpha:
  print('Reject Null Hypothesis(Ho). The variances across the groups
are not equal')
else:
  print('Fail to Reject Null Hypothesis(Ho). The variances across the
groups are equal')

# We cannot continue with two-way ANOVA Test as Normality and Equality
in variance Test does not meet.
```

```
Test Statistic:  67.50325013129238
p-value:  5.616798787507102e-147
Reject Null Hypothesis(Ho). The variances across the groups are not
equal
```

**Q5) check if Weather is dependent on the season**

Step 1)

Ho: Weather and season are independent. (There is no association between weather and season.)

Ha: Weather and season are dependent. (There is an association between weather and season.)

Step 2)

Test Statistic/Distribution: We use Chi-Squared test for independence as both the variables are categotical variables.

Step 3)

We will choose to perform two-tailed test here.

Step 4)

Find out the p-value and we will consider alpha = 0.05

Step 5)

Compare p-value with alpha.

if p-value < alpha:

Reject Null Hypothesis(Ho)

else:

Fail to Reject Null Hypothesis(Ho)

```
weather_season = pd.crosstab(df['weather'], df['season']).drop('Heavy Rain', axis=0)
weather_season
```

| season<br>weather | Fall | Spring | Summer | Winter |
|---|---|---|---|---|
| Clear | 1930 | 1759 | 1801 | 1702 |
| Light Rain | 199 | 211 | 224 | 225 |
| Mist | 604 | 715 | 708 | 807 |

```
from scipy.stats import chi2_contingency
stat, p_value, dof, expected = chi2_contingency(weather_season)
print('Test Statistic: ', stat)
print('p-value: ', p_value)
```

```
if p_value < alpha:
  print('Reject Null Hypothesis(Ho). Weather and season are
dependent.')
else:
  print('Fail to Reject Null Hypothesis(Ho). Weather and season are
independent.')
```

```
Test Statistic:  46.101457310732485
p-value:  2.8260014509929403e-08
Reject Null Hypothesis(Ho). Weather and season are dependent.
```

**Q6) Check if windspeed and humidity are related**

Step 1)

Ho: There is no monotonic relationship between windspeed and humidity

Ha: There is a monotonic relationship between windspeed and humidity

Step 2)

Test Statistic/Distribution: We use Spearman's rank correlation Test here as both the variables are categotical numerical and the distribution is non-linear.

Step 3)

We will choose to perform two-tailed test here.

Step 4)

Find out the p-value and we will consider alpha = 0.05

Step 5)

Compare p-value with alpha.

if p-value < alpha:

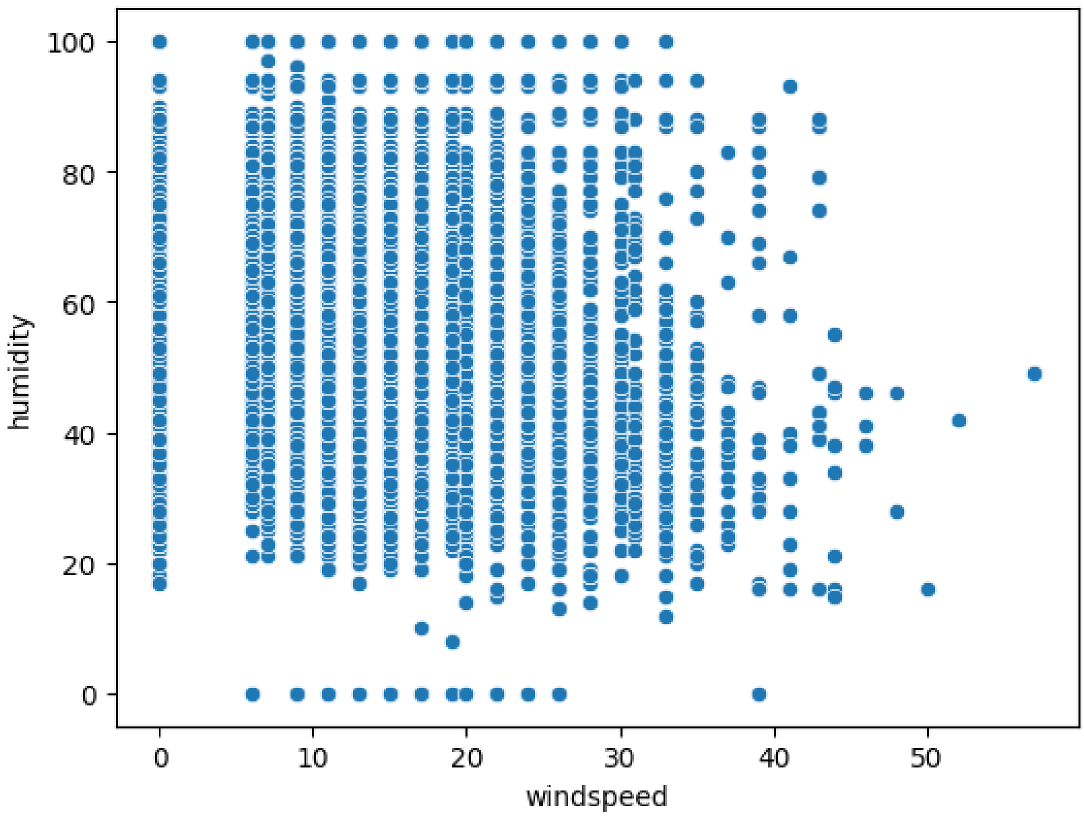
Reject Null Hypothesis(Ho)

else:

Fail to Reject Null Hypothesis(Ho)

```
sns.scatterplot(x='windspeed', y='humidity', data=df)
```

```
<Axes: xlabel='windspeed', ylabel='humidity'>
```




```
# The distribution tells us a monotonic relationship where as
windspeend increases, the humidity decreases and vice-versa in a non
linear way. Hence we go with the Spearman's rank correlation here.

#H0: There is no monotonic relationship between windspeed and
humidity. The Spearman's rank correlation coefficient (ρ) is equal to
zero.
#Ha: There is a monotonic relationship between windspeed and humidity.
The Spearman's rank correlation coefficient (ρ) is not equal to zero.

from scipy.stats import spearmanr
windspeed = df['windspeed']
humidity = df['humidity']

stest, p_value = spearmanr(windspeed, humidity)
print('Test Statistic: ', stest)
print('p-value: ', p_value)

if p_value < alpha:
  print('Reject Null Hypothesis(Ho). There is a monotonic relationship
between windspeed and humidity.')
else:
  print('Fail to Reject Null Hypothesis(Ho). There is no monotonic
relationship between windspeed and humidity.')
```

```
if stest > 0:
  print(f"Since Spearman's coefficient is {stest}, this indicates  a positive monotonic relationship between windspeed and humidity")
else:
  print(f"Since Spearman's coefficient is {stest}, this indicates  a negative monotonic relationship between windspeed and humidity")
```

```
Test Statistic:  -0.32444686812751267
p-value:  3.0980404677025436e-265
Reject Null Hypothesis(Ho). There is a monotonic relationship between windspeed and humidity.
Since Spearman's coefficient is -0.32444686812751267, this indicates a negative monotonic relationship between windspeed and humidity
```

# OVERALL INSIGHTS AND RECOMMENDATIONS

**INSIGHTS**

1) The data was collected during the below Time frame.

Start Date: 2011-01-01 00:00:00

End Date: 2012-12-19 23:00:00

2) More Number of electric cycle were rented by the customers during Fall Seasons and also when the Weather is in Clear Condition.

3) Maximum users rented the electric cycle during the Non Holiday Days.

4) Maximum users rented the electric cycle to commute to their office that is during the Working Days.

5) There is stable growth in the number of cycles rented over the months with a spike between May and October month and a drop from November month. The same trend is almost beign oberved for both the years - 2011 & 2012.

6) There is a linear growth in number of cycles rented from January to June and then it is quite constant from June to October and then dropped back from November.

7) The data was collected mostly on the first 19 days in any month.

8) In any given day, on average more users rent the electric cycles mainly between 7 AM to 9 AM in the Morning and 4 PM to 8 PM in the Night.

9) The average cycles rented on working(1) and non-working(0) days are same which means that the Working Day has no effect on number of cycles rented.

10) The median of number of cycles rented are not same during different Weather conditions which tells that the Weather has effect on the median number of cycles rented.

11) The median of number of cycles rented are not same during different Seasons conditions.Season has effect on the median number of cycles rented.

12) Weather and season are dependent as per the Hypothesis Testing.

**RECOMMENDATIONS**

1) Seasonal Marketing: There is a difference in number of electric cycles rented across each season.Yulu can adjust its marketing strategies accordingly to improve sales.

2) Feedback and Reviews: Add Feedback and Reviews section in the App to get the feedbacks from the customers on the Application and which also helps in identifying areas for improvement, understand customer preferences, and tailor the services to meet the customer expectations.

3) Social Media Marketing: The company should try implementing marketing strategies like Celebrity / Athlete endorsements, Influencer Marketing to have a better reach to the customers which inturn helps in generating a demand and improve sales.

4) Pricing based on the Hour of the Day: Seeing the fluctuations in the demand for the electric cycles rented by the customers for each Hour in a day, the company can set the rental price accordingly like low rental rates during Non-Peak hours and high-rental rates during Peak hours.