YULU

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.

Let's explore the dataset. Here's my Notebook Link.

#Importing the Python libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

Hypothesis Testing

from scipy.stats import f, f_oneway, shapiro, ttest_ind, levene, kruskal,chi2_contingency

Reading the CSV file:

df=pd.read_csv("yulu_dataset.csv")

```
[ ]: df.shape
```

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

```
[ ]: 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
```

```python
df.dtypes
```

```
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
```

Detect Null values

```python
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
```

Check for Duplicates

```python
df.duplicated().sum()
```

```
0
```

Statistical summary ¶

```
# Categorical variable
df.describe(include='object')
```

	datetime
count	10886
unique	10886
top	2011-01-01 00:00:00
freq	1

```
#Numerical variable
df.describe()
```

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000

Data Wrangling

```
df["datetime"] = pd.to_datetime(df['datetime'])
df = df.astype({'season': object, 'weather': object, 'holiday': object, 'workingday': object})
df.info()
```

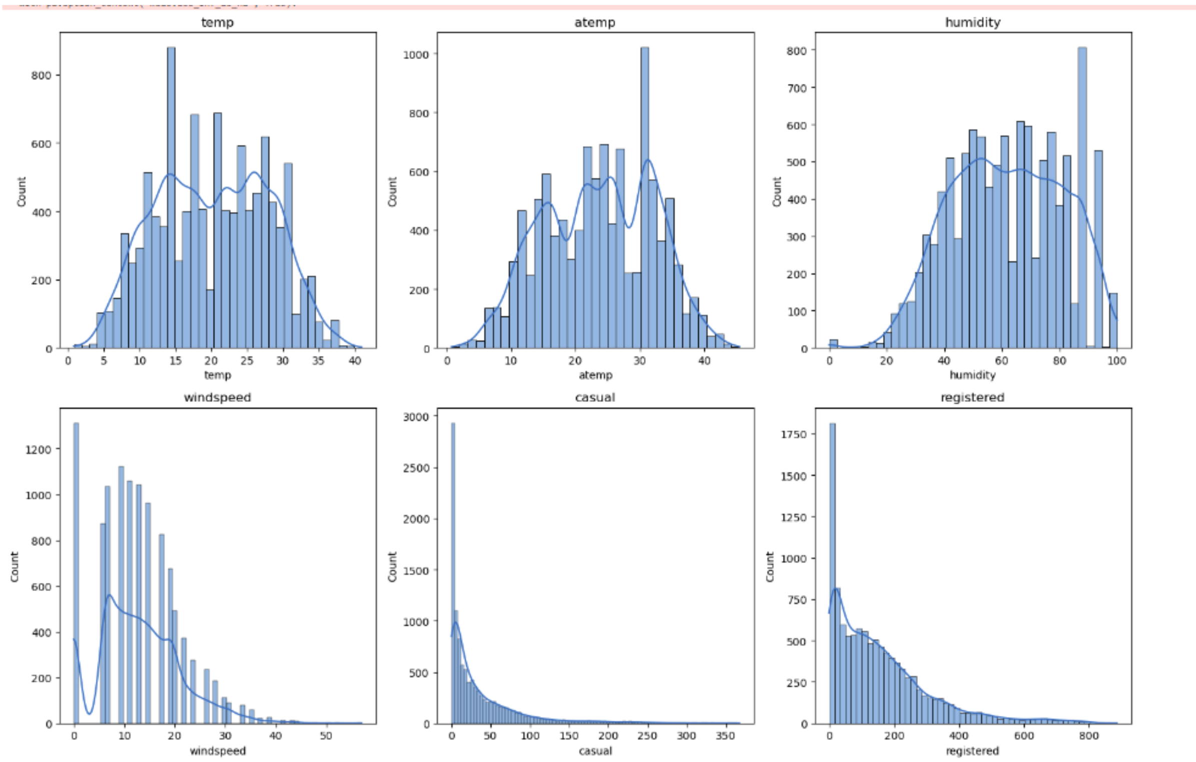
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    10886 non-null  datetime64[ns]
 1   season      10886 non-null  object
 2   holiday     10886 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
```

```
df['season'] = df['season'].map({
    1: 'Spring', 2: 'Summer', 3: 'Fall', 4: 'Winter'
})
```

Distribution of Numerical & Categorical variables:

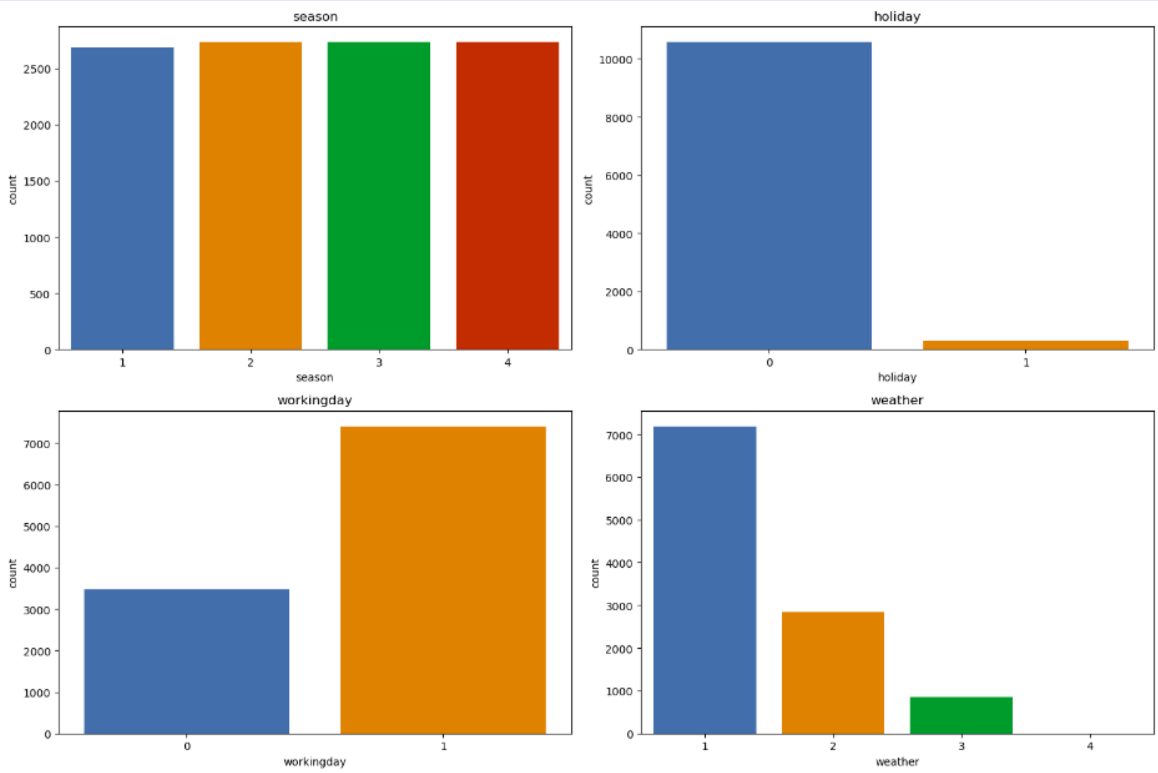
```
numerical = [ 'temp','atemp', 'humidity', 'windspeed', 'casual', 'registered']
# Create subplots
```

```
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))
# Flatten the axes so that we can iterate over them easily
axes = axes.flatten()
# Iterate over numerical variables and plot histplots
for i, var in enumerate(numerical):
    sns.histplot(x=df[var], ax=axes[i],kde=True)
    axes[i].set_title(var)
# Adjust layout
plt.tight_layout()
plt.show()
```



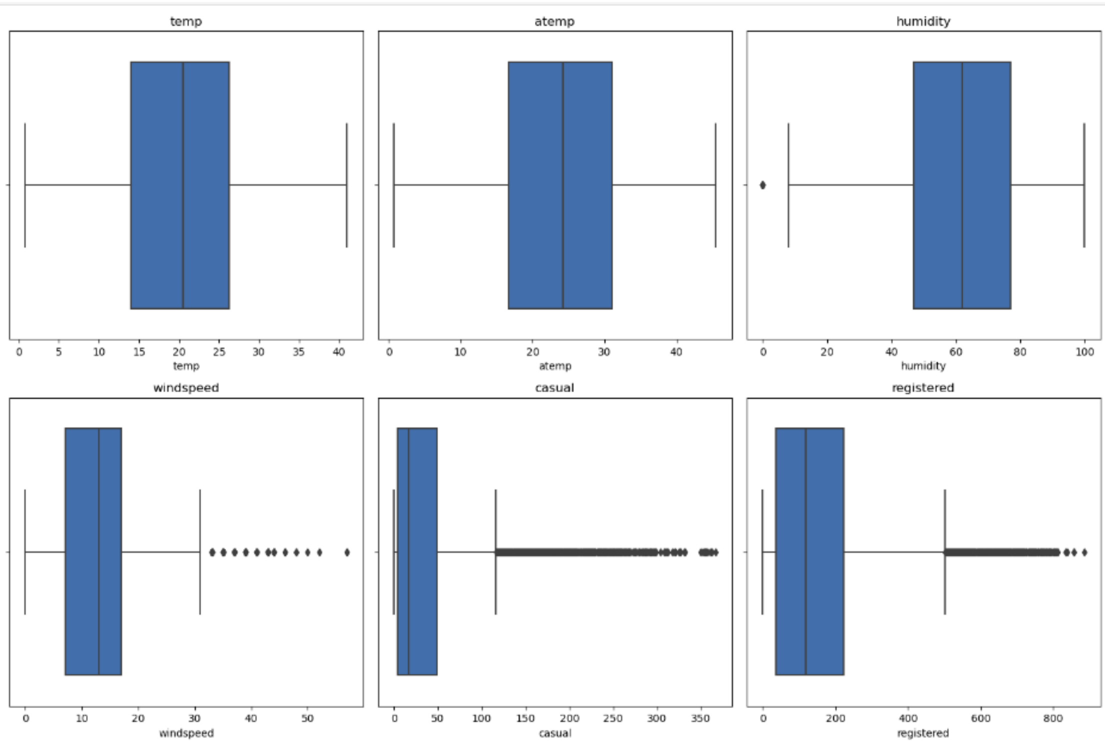

```
categorical = [ 'season','holiday', 'workingday', 'weather']
# Create subplots
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))
# Flatten the axes so that we can iterate over them easily
axes = axes.flatten()
# Iterate over categorical variables and plot countplots
for i, var in enumerate(categorical):
    sns.countplot(x=df[var], ax=axes[i])
```

```
    axes[i].set_title(var)
# Adjust layout
plt.tight_layout()
plt.show()
```



Detect and handle Outliers:

```
numerical = [ 'temp','atemp', 'humidity', 'windspeed', 'casual', 'registered']
# Create subplots
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))
# Flatten the axes so that we can iterate over them easily
axes = axes.flatten()
# Iterate over numerical variables and plot boxplots
for i, var in enumerate(numerical):
    sns.boxplot(x=df[var], ax=axes[i])
    axes[i].set_title(var)
# Adjust layout
plt.tight_layout()
plt.show()
```




```
# Calculate 5th and 95th percentiles for each column
percentile_5 = df[numerical].quantile(0.05)
percentile_95 = df[numerical].quantile(0.95)
# Clip the data for each column between the 5th and 95th percentiles
df[numerical] = df[numerical].clip(percentile_5, percentile_95, axis=1)
# Output the clipped data
df[numerical]
```

	temp	atemp	humidity	windspeed	casual	registered
0	9.84	14.395	81	0.0000	3	13
1	9.02	13.635	80	0.0000	8	32
2	9.02	13.635	80	0.0000	5	27
3	9.84	14.395	75	0.0000	3	10
4	9.84	14.395	75	0.0000	0	4
...	...	...	...	...	...	...
10881	15.58	19.695	50	26.0027	7	329
10882	14.76	17.425	57	15.0013	10	231
10883	13.94	15.910	61	15.0013	4	164
10884	13.94	17.425	61	6.0032	12	117
10885	13.12	16.665	66	8.9981	4	84

10886 rows × 6 columns

```
q1 = df['count'].quantile(0.25)
q3 = df['count'].quantile(0.75)
iqr = q3-q1
iqr
df = df[(df['count']>(q1-1.5*iqr)) & (df['count'] < (q3 + 1.5 *iqr))]
df
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	Spring	0	0	1	9.84	14.395	81	0.0000	3	13	16
1	2011-01-01 01:00:00	Spring	0	0	1	9.02	13.635	80	0.0000	8	32	40
2	2011-01-01 02:00:00	Spring	0	0	1	9.02	13.635	80	0.0000	5	27	32
3	2011-01-01 03:00:00	Spring	0	0	1	9.84	14.395	75	0.0000	3	10	13
4	2011-01-01 04:00:00	Spring	0	0	1	9.84	14.395	75	0.0000	0	4	1
...	...	...	...	...	...	...	...	...	...	...	...	...
10881	2012-12-19 19:00:00	Winter	0	1	1	15.58	19.695	50	26.0027	7	329	336
10882	2012-12-19 20:00:00	Winter	0	1	1	14.76	17.425	57	15.0013	10	231	241
10883	2012-12-19 21:00:00	Winter	0	1	1	13.94	15.910	61	15.0013	4	164	168
10884	2012-12-19 22:00:00	Winter	0	1	1	13.94	17.425	61	6.0032	12	117	129
10885	2012-12-19 23:00:00	Winter	0	1	1	13.12	16.665	66	8.9981	4	84	88

10583 rows × 12 columns

Relationship between the Dependent and Independent Variables:

Plot visualizing difference between count of casual user and registered users and Temperature

plt.figure(figsize=(18, 6))

sns.scatterplot(x ="temp",y = "count",data = df)

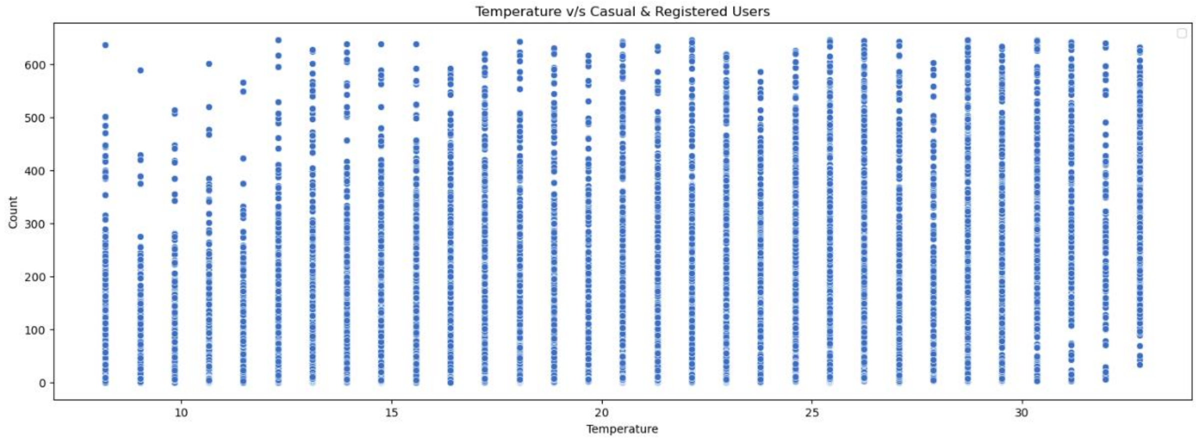
plt.xlabel('Temperature')

plt.ylabel('Count')

plt.title('Temperature v/s Casual & Registered Users')

plt.legend()

plt.show()



No. of bike rides on Weekdays and Weekends:

Hypothesis for 2-Sample T-Test:

Null Hypothesis (H0): The mean number of electric cycles rented is the same on working days and non-working days.

Alternative Hypothesis (H1): The mean number of electric cycles rented is different on working days compared to non-working days.

Assumptions for the Test:

- The samples are independent.
- The samples are randomly drawn from the population.
- The data follows a normal distribution, or the sample size is large enough to apply the Central Limit Theorem.

Test Procedure: Separate the counts of rented cycles into two groups based on whether the day is a working day or not. Perform the 2-sample t-test to compare the means of these two groups. Evaluate the p-value to determine if the result is statistically significant (typically using a significance level of 0.05).

```
holiday_count=df[df['holiday'] == 1]['count']

non_holiday_count=df[df['holiday'] == 0]['count']

# Perform 2-sample t-test

t_stat, p_value = ttest_ind(holiday_count, non_holiday_count)

t_stat, p_value

if p_value<0.05:

    print("Reject Ho")

    print("The mean number of electric cycles rented is different on holidays compared to non-holidays")

else:

    print("Fail to reject Ho")

    print("The mean number of electric cycles rented is the same on holidays and non-holidays")
```

```
Fail to reject Ho
The mean number of electric cycles rented is the same on holidays and non-holidays
```

Number of Bicycles Rented in Different Seasons:

```
# Performing Anova test for effect of Weather
weather_1= df[df["weather"]==1]["count"]
weather_2= df[df["weather"]==2]["count"]
weather_3= df[df["weather"]==3]["count"]
weather_4= df[df["weather"]==4]["count"]
f_stat, p_value = f_oneway(weather_1, weather_2, weather_3,weather_4)
print("f_stat: ", f_stat)
print("p-value: ", p_value)
#Significance level is 5%
alpha = 0.05
if p_value < alpha:
    print("Reject Ho: The mean number of electric cycles rented differs across weather conditions")
else:
    print("Failed to reject Ho: The mean number of electric cycles rented is the same across different weather conditions")
```

```
f_stat:  64.38048872136727
p-value:  3.029209202309234e-41
Reject Ho: The mean number of electric cycles rented differs across weather conditions
```

#**Normality test**

```
sstat,pvalue=shapiro(df["count"].sample(4999))

print(pvalue)

if pvalue<0.05:

  print("Normally distributed")

else:

  print("Not normally distributed")
```

0.0
Normally distributed

```
from scipy.stats import kstest

kstat,pvalue=kstest(weather_1,weather_2,weather_3,weather_4)

print(pvalue)

if pvalue<0.05:

  print("Normally distributed")

else:

  print("Not normally distributed")
```

1.9541397976486484e-06
Normally distributed

```
from scipy.stats import levene

lstat,pvalue=levene(weather_1,weather_2,weather_3,weather_4)

print(pvalue)

if pvalue<0.05:

  print("Reject Ho,variance is not equal ")

else:

  print("Fail to reject Ho,variance is equal ")
```

2.0385458926668884e-37
Reject Ho,variance is not equal

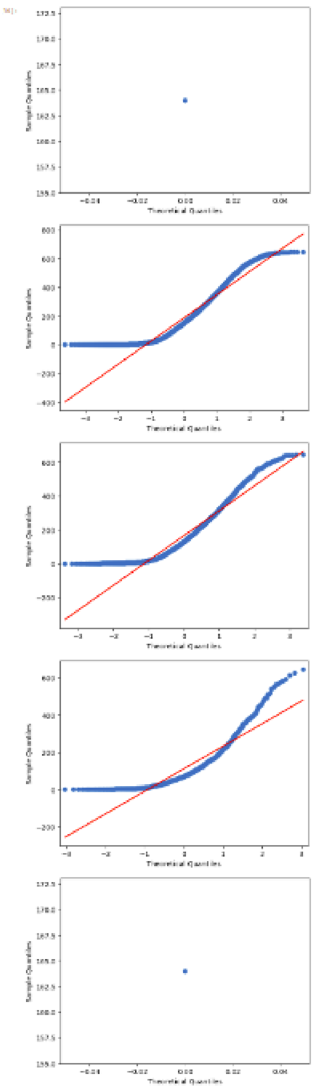
```
from statsmodels.graphics.gofplots import qqplot

qqplot(weather_1,line="s")

qqplot(weather_2,line="s")

qqplot(weather_3,line="s")

qqplot(weather_4,line="s")
```



Number of Bicycles Rented in Different Seasons:

```
df["season"].value_counts()
```

```
season
Spring    2670
Winter    2664
Summer    2633
Fall      2616
Name: count, dtype: int64
```

season1 = df[df['season']=="Spring"]['count'].sample(500)

season2 = df[df['season']=="Winter"]['count'].sample(500)

season3 = df[df['season']=="Summer"]['count'].sample(500)

season4 = df[df['season']=="Fall"]['count'].sample(500)

plt.figure(figsize=(20,10))

#histogram for winter season

plt.subplot(2,2,1)

sns.histplot(season1,kde=True,color='mediumseagreen')

plt.title('Season1')

#histogram for fall season

plt.subplot(2,2,2)

sns.histplot(season2,kde=True,color='mediumseagreen')

plt.title('Season2')

#histogram for summer season

plt.subplot(2,2,3)

sns.histplot(season3,kde=True,color='mediumseagreen')

plt.title('Season3')

#histogram for spring season

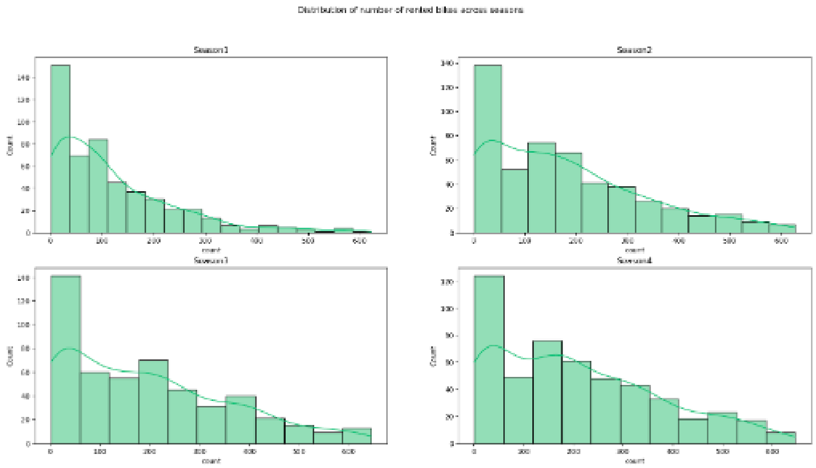
plt.subplot(2,2,4)

sns.histplot(season4,kde=True,color='mediumseagreen')

plt.title('Season4')

```
plt.suptitle('Distribution of number of rented bikes across seasons')

plt.show()
```



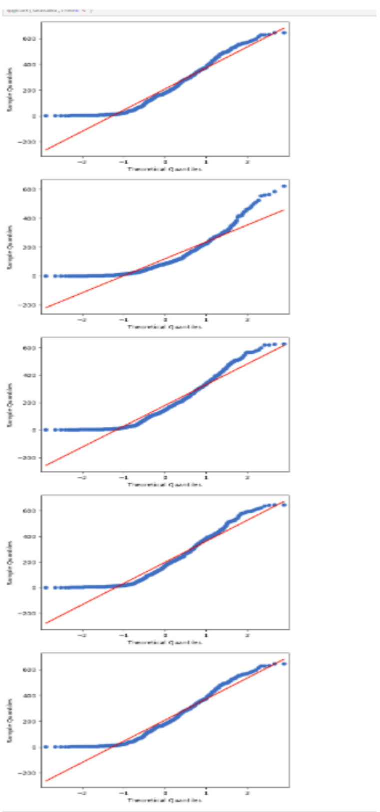

```
from statsmodels.graphics.gofplots import qqplot

qqplot(season1,line="s")

qqplot(season2,line="s")

qqplot(season3,line="s")

qqplot(season4,line="s")
```



```
h_stat, p_value = kruskal(season1,season2,season3,season4)

print("H statistic vale: ", h_stat)

print("P Value: ", p_value)

# Significance level is 5%

alpha = 0.05

if p_value < alpha:

    print("Reject Ho: The mean number of electric cycles rented differs across seasons")

else:

    print("Failed to reject Ho: The mean number of electric cycles rented is the same across different seasons.")
```

```
H statistic vale:  92.58271938974767
P Value:  6.105765141635713e-20
Reject Ho: The mean number of electric cycles rented differs across seasons
```

[52]:

ANOVA Test:

Null Hypothesis (H0): The mean number of electric cycles rented is the same across different seasons.

Alternative Hypothesis (H1): The mean number of electric cycles rented differs across seasons.

```
# Performing Anova test for effect of Season

f_stat, p_value = f_oneway(season1,season2, season3,season4)

print("f_stat: ", f_stat)

print("p-value: ", p_value)

# Significance level is 5%

alpha = 0.05

if p_value < alpha:

    print("Reject Ho: The mean number of electric cycles rented differs across seasons")

else:

    print("Failed to reject Ho: The mean number of electric cycles rented is the same across different seasons.")
```

```
f_stat:  34.32126380422523
```

p-value: 1.2617006587752654e-21
Reject Ho: The mean number of electric cycles rented differs across seasons

```
from scipy.stats import levene

lstat,pvalue=levene(season1,season2, season3,season4)

print(pvalue)

if pvalue<0.05:

  print("Reject Ho,variance is not equal ")

else:

  print("Fail to reject Ho,variance is equal ")
```

9.30264271270202e-19
Reject Ho,variance is not equal

Chi-Square Test:

Hypothesis: to check if weather conditions are dependent on the season.

Null Hypothesis (H0): Weather conditions are independent of the season.

Alternative Hypothesis (H1): Weather conditions are dependent on the season.

Test Procedure:

Create a contingency table (cross-tabulation) of weather conditions versus seasons.

Perform the Chi-square test of independence on this table.

Evaluate the p-value to determine if the result is statistically significant (typically using a significance level of 0.05).

```
# Creating a contingency table of weather conditions vs seasons

contingency_table = pd.crosstab(df['season'], df['weather'])

# Perform Chi-square test of independence

chi2, p_value, dof, expected = chi2_contingency(contingency_table)

chi2, p_value, dof, expected

# Checking for significance level of 5%

alpha = 0.05

if p_value < alpha:

    print("Reject Ho - Weather is dependent on the season")

else:
```

print("Failed to reject Ho- Weather is independent of the Season")

Reject Ho - Weather is dependent on the season

Inference:

We reject the null hypothesis (H0) and accept the alternative hypothesis (H1): Weather conditions are dependent on the season. This implies that the likelihood of experiencing certain types of weather conditions varies with the season, which is a logical outcome considering seasonal weather patterns.

Insights:

- Seasonal and Weather Considerations: Yulu should consider seasonal and weather variations in their operational and marketing strategies. For example, increasing fleet availability during favourable seasons and weather conditions could boost usage.
- Focus on Working Days: While working days did not show a significant difference in rentals statistically, operational focus during these days might still be beneficial given the slightly higher usage trends.
- Based on hypothesis testing, weather and season do have effects on the bike rentals
- Weather and Season are dependent on each other based on the analysis.

Recommendations:

- To beat the Temperature barrier Yulu can try using sunshield for bikes which can cover the user riding the Yulu bikes. Which can eventually increase the count of total users.
- Windspeed also is big factor for Yulu bikes, where more windspeed has reduced the number of users using the Yulu bikes. Yulu can try attaching Helmets to face the windspeed and thus eventually users will prefer to travel via Yulu bikes over local services like Metro, Buses, etc.
- Continuous Monitoring: Ongoing analysis of usage patterns, especially in response to changes in weather, urban infrastructure, and customer preferences, is recommended.

- Since the season has effect on rentals, company must have high stock in rentals to cater the demand during the seasonal time in comparison to the other off-season times.
- Since the registered users are highest contributors, this shows positive sign of the service provided by the company and must continue maintaining the levels during the highest demand spike seasons too.
- Based on the weather conditions, rentals happen mostly during the clear sky and in other conditions can take bikes for maintenance.