Yulu Hypothesis Testing

December 12, 2023

[1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # data visualisation library
import seaborn as sns #data visualisation library built on the top of the␣
↪matplotlib
import plotly.graph_objs as go
import plotly.subplots as sp
import plotly.express as px
import datetime as dt
from scipy.stats import ttest_ind,levene,shapiro,f_oneway,chi2_contingency
from statsmodels.graphics.gofplots import qqplot
```

1 Defining Problem Statement and Analysing basic metrics

[2]:
```
df=pd.read_csv("/Users/senth/Downloads/bike_sharing.csv")
df
```

[2]:
```
                  datetime  season  holiday  workingday  weather   temp  \
0      2011-01-01 00:00:00       1        0           0        1   9.84
1      2011-01-01 01:00:00       1        0           0        1   9.02
2      2011-01-01 02:00:00       1        0           0        1   9.02
3      2011-01-01 03:00:00       1        0           0        1   9.84
4      2011-01-01 04:00:00       1        0           0        1   9.84
…                      …       …        …          …        …     …
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
0      14.395        81     0.0000       3          13     16
1      13.635        80     0.0000       8          32     40
2      13.635        80     0.0000       5          27     32
3      14.395        75     0.0000       3          10     13
4      14.395        75     0.0000       0           1      1
…         …        …          …       …          …      …
```

```
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

[10886 rows x 12 columns]
```

[3]:
```
df.shape
```

[3]:
```
(10886, 12)
```

[4]:
```
df.head(5)
```

[4]:
```
              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
```

[5]:
```
df.columns
```

[5]:
```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count'],
      dtype='object')
```

[6]:
```
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
```

Inference:

1. All the columns execpt datetime are numerical columns.
2. There are no missing values in the dataframe.
3. 'datetime' attribute is of object type, we should convert it to datetime type.
4. There are various categorical columns (season, holiday, workingday, weather) that appear to be integer type, we should convert them to categorical type.

[7]:
```python
df.describe().T
```

[7]:
```
              count        mean         std   min      25%      50%       75%  \
season      10886.0    2.506614    1.116174  1.00   2.0000    3.000    4.0000
holiday     10886.0    0.028569    0.166599  0.00   0.0000    0.000    0.0000
workingday  10886.0    0.680875    0.466159  0.00   0.0000    1.000    1.0000
weather     10886.0    1.418427    0.633839  1.00   1.0000    1.000    2.0000
temp        10886.0   20.230860    7.791590  0.82  13.9400   20.500   26.2400
atemp       10886.0   23.655084    8.474601  0.76  16.6650   24.240   31.0600
humidity    10886.0   61.886460   19.245033  0.00  47.0000   62.000   77.0000
windspeed   10886.0   12.799395    8.164537  0.00   7.0015   12.998   16.9979
casual      10886.0   36.021955   49.960477  0.00   4.0000   17.000   49.0000
registered  10886.0  155.552177  151.039033  0.00  36.0000  118.000  222.0000
count       10886.0  191.574132  181.144454  1.00  42.0000  145.000  284.0000

                 max
season        4.0000
holiday       1.0000
workingday    1.0000
weather       4.0000
temp         41.0000
atemp        45.4550
humidity    100.0000
windspeed    56.9969
casual      367.0000
registered  886.0000
count       977.0000
```

[8]:
```python
df.describe(include='object').T
```

[8]:
```
          count unique                  top freq
datetime  10886  10886  2011-01-01 00:00:00    1
```

2 Non-Graphical Analysis: Value counts and unique attributes

[9]:
```
df.nunique()
```

[9]:
```
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
```

[10]:
```
for col in df.columns:
    value_count=df[col].value_counts(normalize=True)*100
    print(f"----Value counts of {col} column ---- ")
    print()
    print(value_count.round(2))
    print()
    print()
```

```
----Value counts of datetime column ----

2011-01-01 00:00:00    0.01
2012-05-01 21:00:00    0.01
2012-05-01 13:00:00    0.01
2012-05-01 14:00:00    0.01
2012-05-01 15:00:00    0.01
                       …
2011-09-02 04:00:00    0.01
2011-09-02 05:00:00    0.01
2011-09-02 06:00:00    0.01
2011-09-02 07:00:00    0.01
2012-12-19 23:00:00    0.01
Name: datetime, Length: 10886, dtype: float64


----Value counts of season column ----

4    25.11
2    25.11
3    25.11
1    24.67
```

```
Name: season, dtype: float64


----Value counts of holiday column ----

0    97.14
1     2.86
Name: holiday, dtype: float64


----Value counts of workingday column ----

1    68.09
0    31.91
Name: workingday, dtype: float64


----Value counts of weather column ----

1    66.07
2    26.03
3     7.89
4     0.01
Name: weather, dtype: float64


----Value counts of temp column ----

14.76    4.29
26.24    4.16
28.70    3.92
13.94    3.79
18.86    3.73
22.14    3.70
25.42    3.70
16.40    3.67
22.96    3.63
27.06    3.62
24.60    3.58
12.30    3.54
21.32    3.33
17.22    3.27
13.12    3.27
29.52    3.24
10.66    3.05
18.04    3.01
20.50    3.00
30.34    2.75
```

```
9.84     2.70
15.58    2.34
9.02     2.28
31.16    2.22
8.20     2.10
27.88    2.06
23.78    1.86
32.80    1.86
11.48    1.66
19.68    1.56
6.56     1.34
33.62    1.19
5.74     0.98
7.38     0.97
31.98    0.90
34.44    0.73
35.26    0.70
4.92     0.55
36.90    0.42
4.10     0.40
37.72    0.31
36.08    0.21
3.28     0.10
0.82     0.06
38.54    0.06
39.36    0.06
2.46     0.05
1.64     0.02
41.00    0.01
Name: temp, dtype: float64


----Value counts of atemp column ----

31.060    6.16
25.760    3.89
22.725    3.73
20.455    3.67
26.515    3.63
16.665    3.50
25.000    3.35
33.335    3.34
21.210    3.27
30.305    3.22
15.150    3.10
21.970    3.01
24.240    3.00
17.425    2.88
```

```
31.820    2.75
34.850    2.60
27.275    2.59
32.575    2.50
11.365    2.49
14.395    2.47
29.545    2.36
19.695    2.34
15.910    2.33
12.880    2.27
13.635    2.18
34.090    2.06
12.120    1.79
28.790    1.61
23.485    1.56
10.605    1.52
35.605    1.46
9.850     1.17
18.180    1.13
36.365    1.13
37.120    1.08
9.090     0.98
37.880    0.89
28.030    0.73
7.575     0.69
38.635    0.68
6.060     0.67
39.395    0.62
6.820     0.58
8.335     0.58
18.940    0.41
40.150    0.41
40.910    0.36
5.305     0.23
42.425    0.22
41.665    0.21
3.790     0.15
4.545     0.10
3.030     0.06
43.940    0.06
2.275     0.06
43.180    0.06
44.695    0.03
0.760     0.02
1.515     0.01
45.455    0.01
Name: atemp, dtype: float64
```

```
----Value counts of humidity column ----

88    3.38
94    2.98
83    2.90
87    2.65
70    2.38
      …
8     0.01
10    0.01
97    0.01
96    0.01
91    0.01
Name: humidity, Length: 89, dtype: float64


----Value counts of windspeed column ----

0.0000     12.06
8.9981     10.29
11.0014     9.71
12.9980     9.57
7.0015      9.50
15.0013     8.83
6.0032      8.01
16.9979     7.57
19.0012     6.21
19.9995     4.52
22.0028     3.42
23.9994     2.52
26.0027     2.16
27.9993     1.72
30.0026     1.02
31.0009     0.82
32.9975     0.73
35.0008     0.53
39.0007     0.25
36.9974     0.20
43.0006     0.11
40.9973     0.10
43.9989     0.07
46.0022     0.03
56.9969     0.02
47.9988     0.02
51.9987     0.01
50.0021     0.01
Name: windspeed, dtype: float64
```

```
----Value counts of casual column ----

0      9.06
1      6.13
2      4.47
3      4.02
4      3.25
       …
332    0.01
361    0.01
356    0.01
331    0.01
304    0.01
Name: casual, Length: 309, dtype: float64


----Value counts of registered column ----

3      1.79
4      1.75
5      1.63
6      1.42
2      1.38
       …
570    0.01
422    0.01
678    0.01
565    0.01
636    0.01
Name: registered, Length: 731, dtype: float64


----Value counts of count column ----

5      1.55
4      1.37
3      1.32
6      1.24
2      1.21
       …
801    0.01
629    0.01
825    0.01
589    0.01
636    0.01
Name: count, Length: 822, dtype: float64
```

2.1 Data Preprocessing

[11]:
```
df.isnull().sum()
```

[11]:
```
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
```

2.2 Data-type conversion

[12]:
```
df['datetime']=pd.to_datetime(df['datetime'])
df['datetime']
```

[12]:
```
0        2011-01-01 00:00:00
1        2011-01-01 01:00:00
2        2011-01-01 02:00:00
3        2011-01-01 03:00:00
4        2011-01-01 04:00:00
                  …
10881    2012-12-19 19:00:00
10882    2012-12-19 20:00:00
10883    2012-12-19 21:00:00
10884    2012-12-19 22:00:00
10885    2012-12-19 23:00:00
Name: datetime, Length: 10886, dtype: datetime64[ns]
```

[13]:
```
def season_type(x):
    if x==1:
        return 'spring'
    elif x==2:
        return 'summer'
    elif x==3:
        return 'fall'
    else:
```

```
        return 'winter'
```

[14]:
```
df['season']=df['season'].apply(lambda x:season_type(x))
df['season']
```

[14]:
```
0        spring
1        spring
2        spring
3        spring
4        spring
          …
10881    winter
10882    winter
10883    winter
10884    winter
10885    winter
Name: season, Length: 10886, dtype: object
```

[15]:
```
df['season']= pd.Categorical(df['season'])
df['weather']=pd.Categorical(df['weather'])
df['holiday']=pd.Categorical(df['holiday'])
df['workingday']=pd.Categorical(df['workingday'])
```

[16]:
```
df[['datetime','season','weather','holiday','workingday']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    10886 non-null  datetime64[ns]
 1   season      10886 non-null  category
 2   weather     10886 non-null  category
 3   holiday     10886 non-null  category
 4   workingday  10886 non-null  category
dtypes: category(4), datetime64[ns](1)
memory usage: 128.3 KB
```

[17]:
```
df.describe().T
```

[17]:
```
              count        mean         std   min      25%      50%       75%  \
temp        10886.0   20.230860    7.791590  0.82  13.9400   20.500   26.2400
atemp       10886.0   23.655084    8.474601  0.76  16.6650   24.240   31.0600
humidity    10886.0   61.886460   19.245033  0.00  47.0000   62.000   77.0000
windspeed   10886.0   12.799395    8.164537  0.00   7.0015   12.998   16.9979
casual      10886.0   36.021955   49.960477  0.00   4.0000   17.000   49.0000
registered  10886.0  155.552177  151.039033  0.00  36.0000  118.000  222.0000
count       10886.0  191.574132  181.144454  1.00  42.0000  145.000  284.0000
```

```
                 max
temp         41.0000
atemp        45.4550
humidity    100.0000
windspeed    56.9969
casual      367.0000
registered  886.0000
count       977.0000
```

[18]:
```
df.describe(include='category').T
```

[18]:

	count	unique	top	freq
season	10886	4	winter	2734
holiday	10886	2	0	10575
workingday	10886	2	1	7412
weather	10886	4	1	7192

2.3 Handling outliers

[19]:
```
outliers={}
for col in df.select_dtypes(include=np.number):
    z_score= np.abs((df[col]-df[col].mean()))/df[col].std()
    column_outliers=df[z_score > 3][col]
    outliers[col]=column_outliers

for col,outlier_values in outliers.items():
    print(f"Outliers for {col} column")
    print(outlier_values)
    print()
```

```
Outliers for temp column
Series([], Name: temp, dtype: float64)

Outliers for atemp column
Series([], Name: atemp, dtype: float64)

Outliers for humidity column
1091    0
1092    0
1093    0
1094    0
1095    0
1096    0
1097    0
1098    0
1099    0
1100    0
```

```
1101    0
1102    0
1103    0
1104    0
1105    0
1106    0
1107    0
1108    0
1109    0
1110    0
1111    0
1112    0
Name: humidity, dtype: int64

Outliers for windspeed column
265      39.0007
613      39.0007
750      43.9989
752      40.9973
753      40.9973
          …
9481     43.0006
9482     43.0006
9484     39.0007
9754     39.0007
10263    43.0006
Name: windspeed, Length: 67, dtype: float64

Outliers for casual column
1384     219
1385     240
1935     196
2127     195
2129     206
        …
10226    195
10227    262
10228    292
10229    304
10230    260
Name: casual, Length: 292, dtype: int64

Outliers for registered column
6611     623
6634     614
6635     638
6649     628
6658     642
```

```
        …
10702    670
10726    655
10750    623
10846    652
10870    665
Name: registered, Length: 235, dtype: int64

Outliers for count column
6658     782
6659     749
6683     746
6779     801
6849     757
        …
9935     834
9944     890
9945     788
10519    743
10534    759
Name: count, Length: 147, dtype: int64
```

Observations:

1. There no outliers in 'temp' and 'atemp' column.
2. Outliers are evident within the 'humidity' and 'windspeed' columns based on the observations.
3. Outliers are noticeable in the counts of casual and registered users, though drawing definite conclusions necessitates analyzing their relationship with independent variables..

3 Univariate Analysis

3.0.1 Distribution of Working day

[20]:
```
workingday_df=df.groupby(['workingday']).
  ↪agg(number_of_cycles_rented=('count','sum')).reset_index()
workingday_df
```

[20]:

	workingday	number_of_cycles_rented
0	0	654872
1	1	1430604

[21]:
```
labels= workingday_df['workingday']
values= workingday_df['number_of_cycles_rented']

#create pie chart
#Create pie chart
colors = ['#C0E0DE','#4F7CAC']
```

```
pie_chart = go.Figure(go.Pie(labels=labels, values=values))
pie_chart.update_traces(hoverinfo='label+value', textinfo='percent',␣
 ↪textfont_size=20,
                  marker=dict(colors=colors, line=dict(color='#000000',␣
 ↪width=2)))

# Create bar chart
bar_chart = go.Figure(go.Bar(x=labels, y=values, marker_color=colors))

# Create subplots
fig = sp.make_subplots(rows=1, cols=2, column_widths=[0.5, 0.5],␣
 ↪specs=[[{"type": "bar"}, {"type": "pie"}]],
                       subplot_titles=("Bar Chart", "Pie Chart"))

# Add charts to subplots
fig.add_trace(bar_chart.data[0], row=1, col=1)
fig.add_trace(pie_chart.data[0], row=1, col=2)

# Update layout
fig.update_layout(showlegend=False,title_text="Distribution of Working day",
                xaxis=dict(title='Working␣
 ↪Day',titlefont_size=16,tickfont_size=14,),
                yaxis=dict(title='Number of Cycles␣
 ↪rented',titlefont_size=16,tickfont_size=14,))

# Show subplots
fig.show()
```

3.0.2 Distribution of Season

[22]:
```
season_df=df.groupby(['season']).agg(number_of_cycles_rented=('count','sum')).
 ↪reset_index()
season_df
```

[22]:

	season	number_of_cycles_rented
0	fall	640662
1	spring	312498
2	summer	588282
3	winter	544034

[23]:
```
labels= season_df['season']
values= season_df['number_of_cycles_rented']

#create pie chart
# Create pie chart
colors = ['#D4D2A5','#FCDEBE','#ddbea9','#ffc8dd']
```

```
pie_chart = go.Figure(go.Pie(labels=labels, values=values))
pie_chart.update_traces(hoverinfo='label+value', textinfo='percent',␣
↪textfont_size=20,
                  marker=dict(colors=colors, line=dict(color='#000000',␣
↪width=2)))

# Create bar chart
bar_chart = go.Figure(go.Bar(x=labels, y=values, marker_color=colors))

# Create subplots
fig = sp.make_subplots(rows=1, cols=2, column_widths=[0.5, 0.5],␣
↪specs=[[{"type": "bar"}, {"type": "pie"}]],
                       subplot_titles=("Bar Chart", "Pie Chart"))

# Add charts to subplots

fig.add_trace(bar_chart.data[0], row=1, col=1)
fig.add_trace(pie_chart.data[0], row=1, col=2)

# Update layout
fig.update_layout(showlegend=False,title_text="Distribution of Season",
                xaxis=dict(title='Season',titlefont_size=16,tickfont_size=14,),
                yaxis=dict(title='Number of Cycles␣
↪rented',titlefont_size=16,tickfont_size=14,))

# Show subplots
fig.show()
```

Insights:

1. During the fall season, approximately 30.7% of cycles are rented.
2. In the summer season, around 28.2% of cycles are rented.
3. The winter season records a rental rate of about 26.1% for cycles.
4. The lowest rental rate, at just 15%, is observed in the spring season.

3.0.3 Distribution of Weather

[24]:
```
weather_df=df.groupby(['weather']).agg(number_of_cycles_rented=('count','sum')).
↪reset_index()
weather_df
```

[24]:

	weather	number_of_cycles_rented
0	1	1476063
1	2	507160
2	3	102089
3	4	164

```
[25]: labels= weather_df['weather']
values= weather_df['number_of_cycles_rented']

#create pie chart
# Create pie chart
colors =  ["#b9e769","#efea5a","#f1c453"]

pie_chart = go.Figure(go.Pie(labels=labels, values=values))
pie_chart.update_traces(hoverinfo='label+value', textinfo='percent',␣
 ↪textfont_size=20,
                  marker=dict(colors=colors, line=dict(color='#000000',␣
 ↪width=2)))

# Create bar chart
bar_chart = go.Figure(go.Bar(x=labels, y=values, marker_color=colors))

# Create subplots
fig = sp.make_subplots(rows=1, cols=2, column_widths=[0.5, 0.5],␣
 ↪specs=[[{"type": "bar"}, {"type": "pie"}]],
                       subplot_titles=("Bar Chart", "Pie Chart"))
fig.add_trace(bar_chart.data[0], row=1, col=1)
fig.add_trace(pie_chart.data[0], row=1, col=2)

# Update layout
fig.update_layout(showlegend=False,title_text="Distribution of Weather types",
                xaxis=dict(title='Weather␣
 ↪types',titlefont_size=16,tickfont_size=14,),
                yaxis=dict(title='Number of Cycles␣
 ↪rented',titlefont_size=16,tickfont_size=14,))

# Show subplots
fig.show()
```

Insights:

1. Weather condition 1 experiences the highest rental rate, with approximately 70.8% of cycles rented.
2. In weather condition 2, around 24.3% of cycles are rented.
3. Weather condition 3 has a rental rate of approximately 4.9% for cycles.
4. Weather condition 4 exhibits an exceptionally low rental rate, with only 0.00786% of cycles being rented.

3.0.4 Hourly Trends in Average Cycle Rentals

```
[28]: hour_df=df.groupby(df['datetime'].dt.hour).
 ↪agg(average_cycles_rented=('count','mean')).reset_index()
hour_df
```

[28]:

	datetime	average_cycles_rented
0	0	55.138462
1	1	33.859031
2	2	22.899554
3	3	11.757506
4	4	6.407240
5	5	19.767699
6	6	76.259341
7	7	213.116484
8	8	362.769231
9	9	221.780220
10	10	175.092308
11	11	210.674725
12	12	256.508772
13	13	257.787281
14	14	243.442982
15	15	254.298246
16	16	316.372807
17	17	468.765351
18	18	430.859649
19	19	315.278509
20	20	228.517544
21	21	173.370614
22	22	133.576754
23	23	89.508772

[29]:

```
fig = px.line(hour_df, x='datetime', y='average_cycles_rented', markers=True)
fig.update_xaxes(tickvals=list(range(25)))
fig.update_layout(title='Average cycles rented in hourly basis',
                  xaxis_title='Hours',
                  yaxis_title='Average cycles rented')
fig.show()
```

Insights:

1. The highest average count of rental bikes is observed at 5 PM, closely followed by 6 PM and 8 AM. This indicates distinct peak hours during the day when cycling is most popular.
2. Conversely, the lowest average count of rental bikes occurs at 4 AM, with 3 AM and 5 AM also showing low counts. These hours represent the early morning period with the least demand for cycling.
3. Notably, there is an increasing trend in cycle rentals between 5 AM and 8 AM, suggesting a surge in demand during the early morning hours as people start their day.
4. Additionally, there is a decreasing trend in cycle rentals from 5 PM to 11 PM, indicating a gradual decline in demand as the day progresses into the evening and nighttime.

3.0.5 Montly trend in average cycle rentals

[30]:
```
month_df=df.groupby(df['datetime'].dt.month).
    ↪agg(average_cycles_rented=('count','mean')).reset_index()
month_df
```

[30]:

	datetime	average_cycles_rented
0	1	90.366516
1	2	110.003330
2	3	148.169811
3	4	184.160616
4	5	219.459430
5	6	242.031798
6	7	235.325658
7	8	234.118421
8	9	233.805281
9	10	227.699232
10	11	193.677278
11	12	175.614035

[31]:
```
fig = px.line(month_df, x='datetime', y='average_cycles_rented', markers=True)
fig.update_xaxes(tickvals=list(range(1,13)),ticktext=['Jan', 'Feb', 'Mar',␣
    ↪'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])
fig.update_layout(title='Average cycles rented on monthly basis',
                  xaxis_title='Month',
                  yaxis_title='Average cycles rented')
fig.show()
```

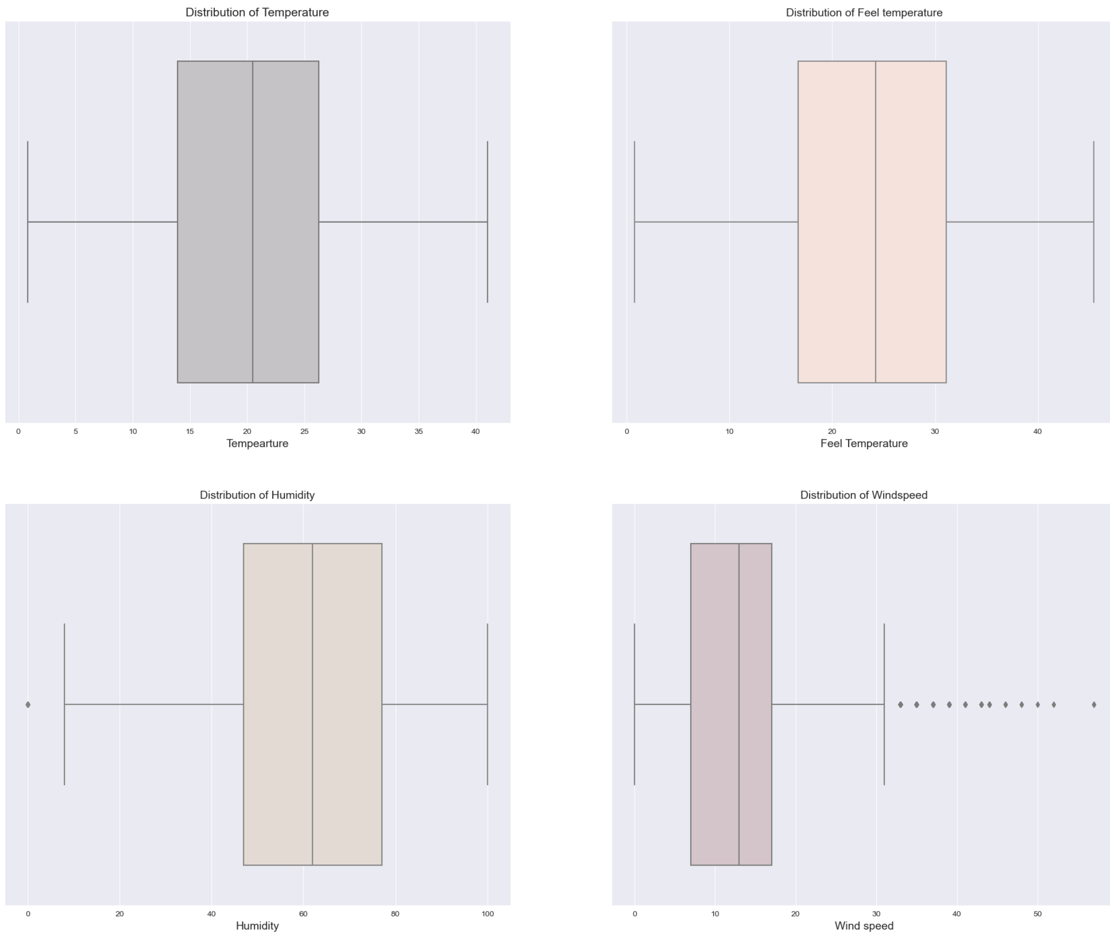
Insights:

1. The highest average hourly count of rental bikes occurs in June, July, and August, reflecting the peak demand during summer.
2. Conversely, the lowest average hourly count of rental bikes is found in January, February, and March, which are the winter months with reduced cycling activity.
3. Notably, there is an increasing trend in average bike rentals from February to June, corresponding to the shift from winter to spring and summer.
4. Conversely, a decreasing trend in average bike rentals is observed from October to December due to the onset of winter.

3.0.6 Distribution of temp, atemp, humidity and windspeed

[33]:
```
sns.set_style('darkgrid')
plt.figure(figsize=(25,20))

# Boxplot for temp column
plt.subplot(2,2,1)
sns.boxplot(data=df,x='temp',color='#C5C3C6')
plt.xlabel('Tempearture',fontsize=14)
```

```
plt.title('Distribution of Temperature',fontsize=15)

#Boxplot for feel temperature
plt.subplot(2,2,2)
sns.boxplot(data=df,x='atemp',color='#F9E0D9')
plt.xlabel('Feel Temperature',fontsize=14)
plt.title('Distribution of Feel temperature',fontsize=14)

#Boxplot for Humidity
plt.subplot(2,2,3)
sns.boxplot(data=df,x='humidity',color='#E6DBD0')
plt.xlabel('Humidity',fontsize=14)
plt.title('Distribution of Humidity',fontsize=14)

#Boxplot for Wind Speed
plt.subplot(2,2,4)
sns.boxplot(data=df,x='windspeed',color='#D6C3C9')
plt.xlabel('Wind speed',fontsize=14)
plt.title('Distribution of Windspeed',fontsize=14)


plt.show()
```



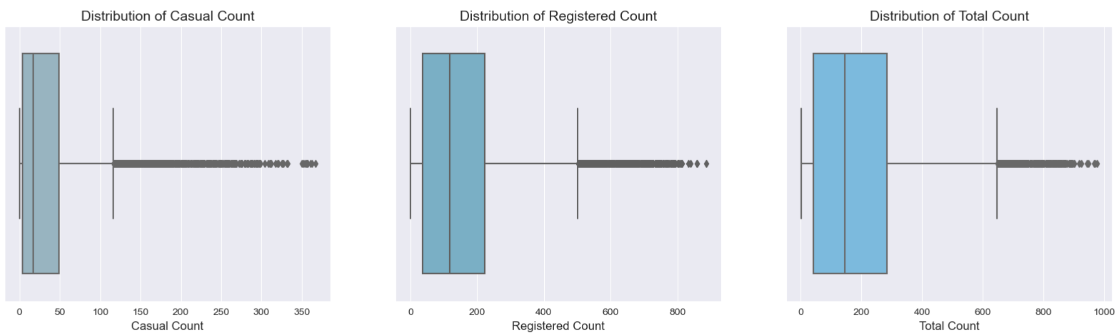
Inference:

1. No outliers are detected in the 'temp' and 'atemp' columns, suggesting that the temperature-related data points fall within the expected range.
2. In the 'humidity' column, a single value is identified as an outlier, implying an unusual humidity measurement distinct from the others.
3. The 'windspeed' column contains 12 outlier values, indicating instances where wind speed measurements significantly deviate from the typical range.

3.0.7 Distribution of Casual count, Registered count and Total count

[34]:
```
plt.figure(figsize=(20,5))

# Boxplot for temp column
plt.subplot(1,3,1)
sns.boxplot(data=df,x='casual',color='#91B7C7')
plt.xlabel('Casual Count',fontsize=12)
plt.title('Distribution of Casual Count',fontsize=14)
```

```
#Boxplot for feel temperature
plt.subplot(1,3,2)
sns.boxplot(data=df,x='registered',color='#6EB4D1')
plt.xlabel('Registered Count',fontsize=12)
plt.title('Distribution of Registered Count',fontsize=14)

#Boxplot for Humidity
plt.subplot(1,3,3)
sns.boxplot(data=df,x='count',color='#6CBEED')
plt.xlabel('Total Count',fontsize=12)
plt.title('Distribution of Total Count',fontsize=14)


plt.show()
```



Inference:

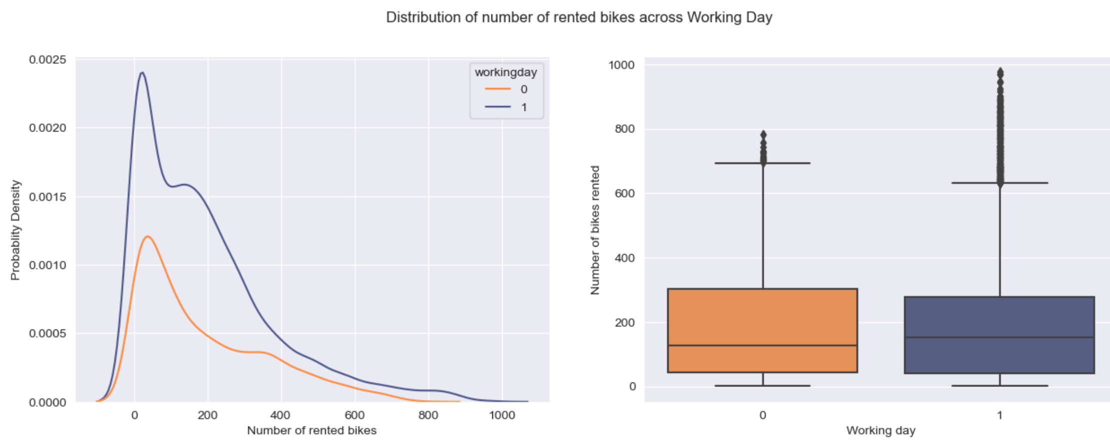
1. The box plot clearly indicates the presence of outliers in the number of casual and registered users. However, further analysis against independent variables is needed before making definitive comments.
2. The box plot reveal data skewness. As we proceed, we will decide whether to address outliers or perform variable transformation. In this case, given the significant number of outliers, variable transformation, specifically Log Transformation, seems to be a more appropriate approach.

4 Bivariate Analysis

4.1 Distribution of count of rented bikes across working day

[35]:
```
plt.figure(figsize=(15,5))
# KDE plot
plt.subplot(1,2,1)
sns.set_style('darkgrid')
sns.kdeplot(data=df,x='count',hue='workingday',palette=['#FF8C42','#4E598C'])
plt.xlabel('Number of rented bikes')
```

```
plt.ylabel('Probablity Density')

# Box plot
plt.subplot(1,2,2)
sns.boxplot(data=df,y='count',x='workingday',palette=['#FF8C42','#4E598C'])
plt.xlabel('Working day')
plt.ylabel('Number of bikes rented')

plt.suptitle('Distribution of number of rented bikes across Working Day')
plt.show()
```



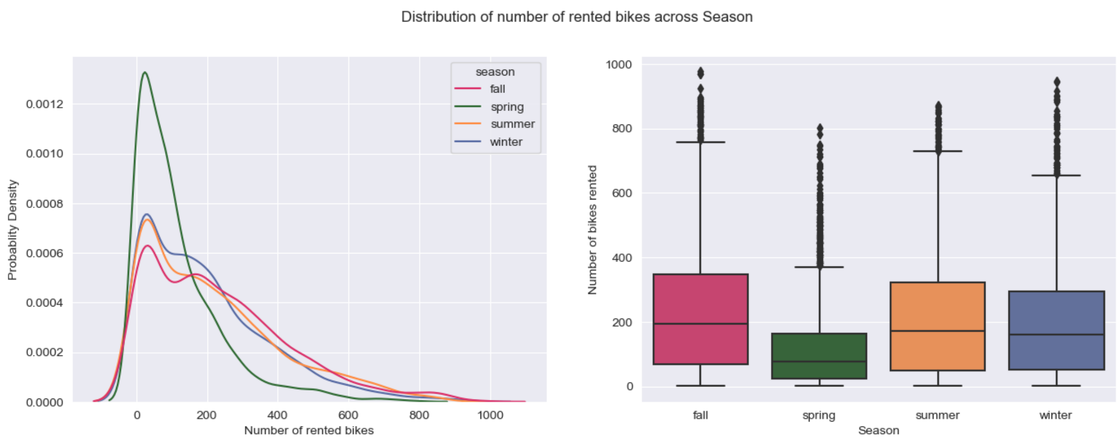
Inference:

The probability of renting bikes on a working day appears to be higher than on a non-working day, as evidenced by our univariate analysis, where 68.6% of bike rentals occurred on working days compared to 31.4% on non-working days. However, we will further investigate this through hypothesis testing to determine if the working day indeed has a statistically significant effect on the number of cycles rented."

4.2 Distribution of count of rented bikes across Season

[36]:
```
plt.figure(figsize=(15,5))
# KDE plot
plt.subplot(1,2,1)
sns.set_style('darkgrid')
sns.
 ↪kdeplot(data=df,x='count',hue='season',palette=['#DB3069','#306B34','#FF8C42','#586BA4'])
plt.xlabel('Number of rented bikes')
plt.ylabel('Probablity Density')

# Box plot
plt.subplot(1,2,2)
```

```
sns.
  ↪boxplot(data=df,y='count',x='season',palette=['#DB3069','#306B34','#FF8C42','#586BA4'])
plt.xlabel('Season')
plt.ylabel('Number of bikes rented')

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



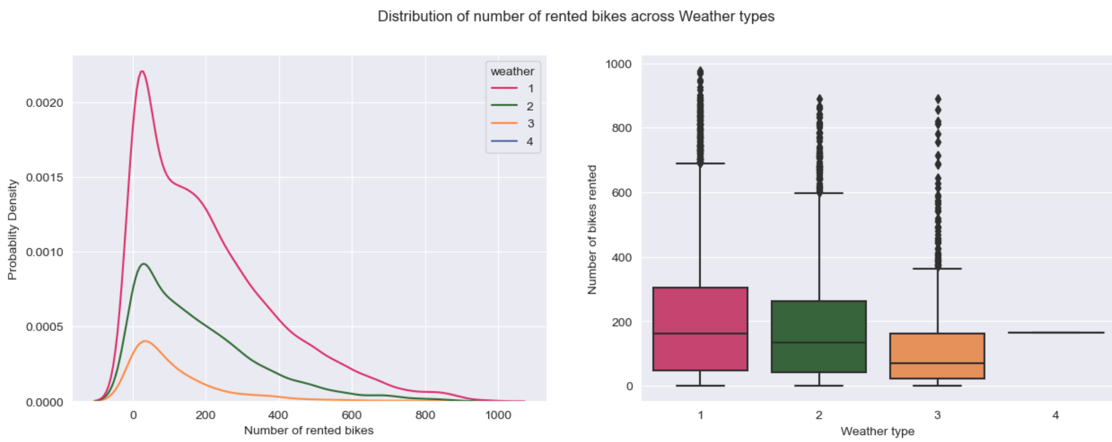
Inference:

The probability of renting a bike during the fall season appears to be higher compared to other seasons. Conversely, the probability of renting bikes during the winter and spring seasons is lower in comparison to summer and fall. To investigate this further, we will conduct an ANOVA test to determine if the season has a statistically significant effect on bike rentals.

4.3 Distribution of count of rented bikes across Weather types

[37]:
```
plt.figure(figsize=(15,5))

# KDE plot
plt.subplot(1,2,1)
sns.set_style('darkgrid')
sns.
  ↪kdeplot(data=df,x='count',hue='weather',palette=['#DB3069','#306B34','#FF8C42','#586BA4'],w
plt.xlabel('Number of rented bikes')
plt.ylabel('Probablity Density')

# Box plot
plt.subplot(1,2,2)
sns.
  ↪boxplot(data=df,y='count',x='weather',palette=['#DB3069','#306B34','#FF8C42','#586BA4'])
plt.xlabel('Weather type')
```

```
plt.ylabel('Number of bikes rented')

plt.suptitle('Distribution of number of rented bikes across Weather types')
plt.show()
```



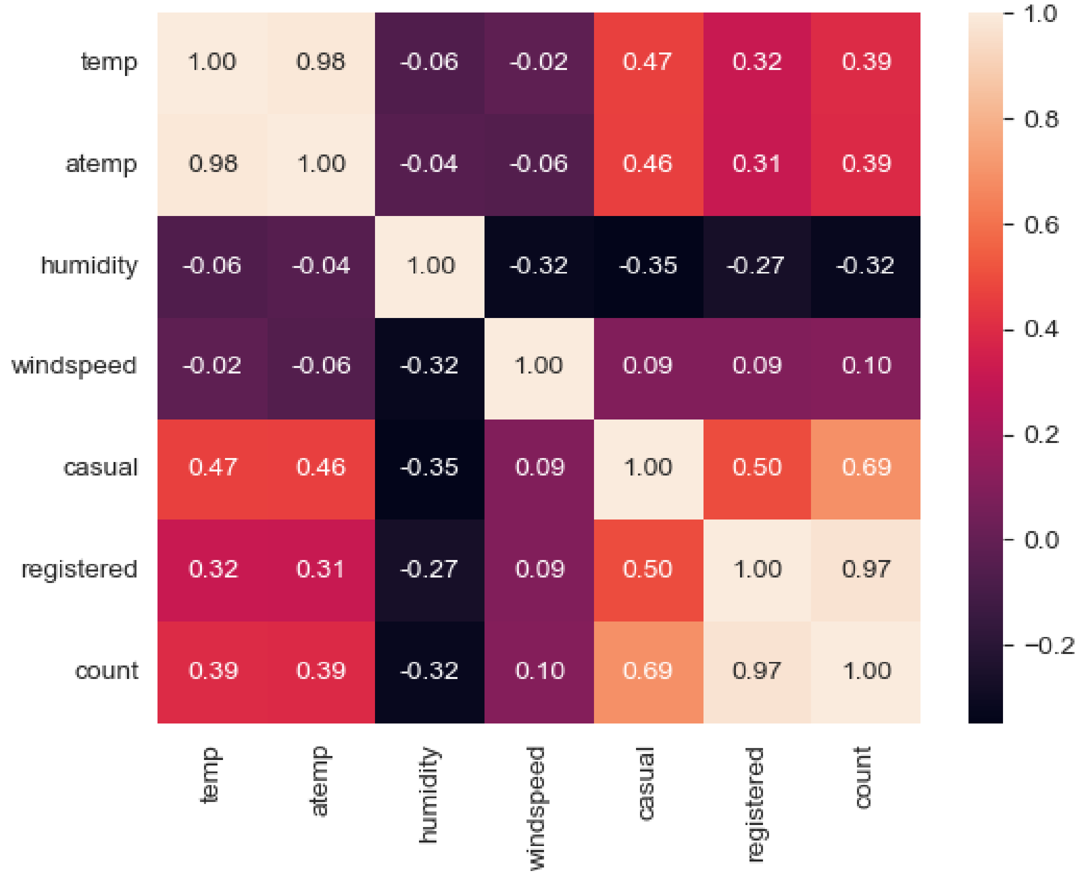
Inference:

The probability of renting a bike during weather condition 1 appears to be higher than in other weather types. This is supported by our univariate analysis, where approximately 70.8% of bike rentals occur in weather condition 1, while the remaining weather types collectively account for approximately 29% of bike rentals. However, we will further investigate this by conducting an ANOVA test to establish whether weather type indeed has a statistically significant effect on the number of bikes rented.

4.4 Heatmap and Correlation

[39]:
```
sns.heatmap(df.corr(),annot=True,cmap='rocket',fmt='.2f')
plt.show()
```

```
C:\Users\senth\AppData\Local\Temp\ipykernel_21584\336913126.py:1: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future
version, it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
```



Insights:

1. he weak positive correlation of 0.39 between temperature and the number of bikes rented suggests that, on average, fewer people prefer to use electric cycles during the daytime between 12 PM to 3 PM. This observation aligns with our univariate analysis, where we discovered that the average number of cycles rented during this time frame was lower compared to other times of the day. A similar correlation pattern is also observed in the case of "feels-like" temperature, reinforcing this trend.
2. The negative correlation between humidity and the number of cycles rented indicates that people tend to avoid using electric bikes during high humidity conditions. This avoidance can be attributed to the discomfort caused by the heavy and sticky air, leading to sweating and a general sense of unease. Moreover, the reduced efficiency of electric bikes in high humidity, resulting in increased air resistance and potential battery performance issues, contributes to the preference for alternative transportation or indoor activities in such conditions.
3. The presence of a weak positive correlation between windspeed and the number of cycles rented indicates that there is a subset of individuals who appear to favor using electric cycles during windy conditions for the sheer enjoyment of the experience. While this preference contributes to a slight increase in bike rentals on windier days, it's essential to recognize that this effect is not particularly strong, as indicated by the weak correlation. This suggests that the enjoyment of cycling in windy conditions is a relatively niche preference among riders.

5 Hypothesis Testing

5.1 2 Sample T test

5.1.1 Does Working day has an effect on the number of bikes rented?

Formulating Null and Alternative Hypotheses

To answer the above question we first set up Null and Alternate Hypothesis:

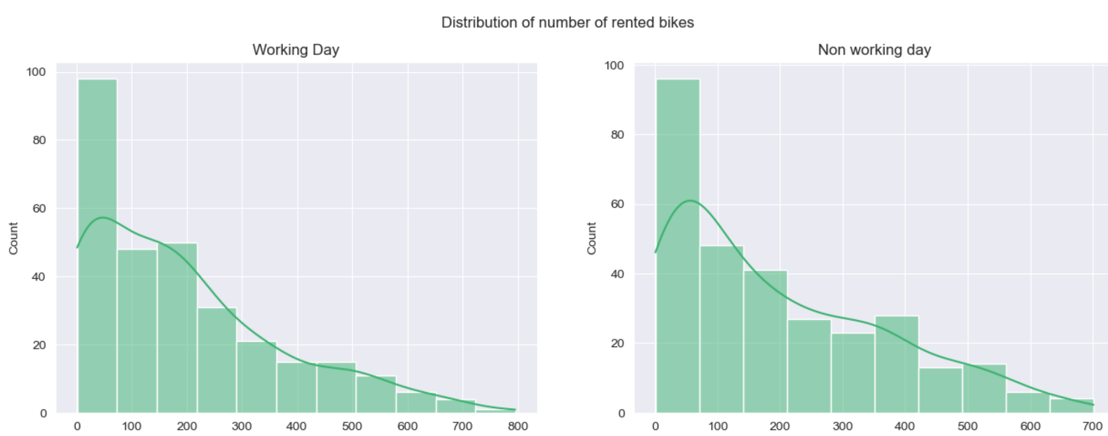
H0 : Working day does not have an effect on number of cycles rented Ha: Working day does have an effect on number of cycles rented

Solution: To test the above hypothesis, we use Two sample T Test

Generate a sample of 300 bike rentals, randomly selected from both working days and non-working days

[40]:
```
workingday_sample=df[df['workingday']==1]['count'].sample(300)
nonworkingday_sample=df[df['workingday']==0]['count'].sample(300)
```

[41]:
```
plt.figure(figsize=(15,5))

#histogram for working day sample
plt.subplot(1,2,1)
sns.histplot(workingday_sample,kde=True,color='mediumseagreen')
plt.title('Working Day')

#histogram for non working day sample
plt.subplot(1,2,2)
sns.histplot(nonworkingday_sample,kde=True,color='mediumseagreen')
plt.title('Non working day')

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

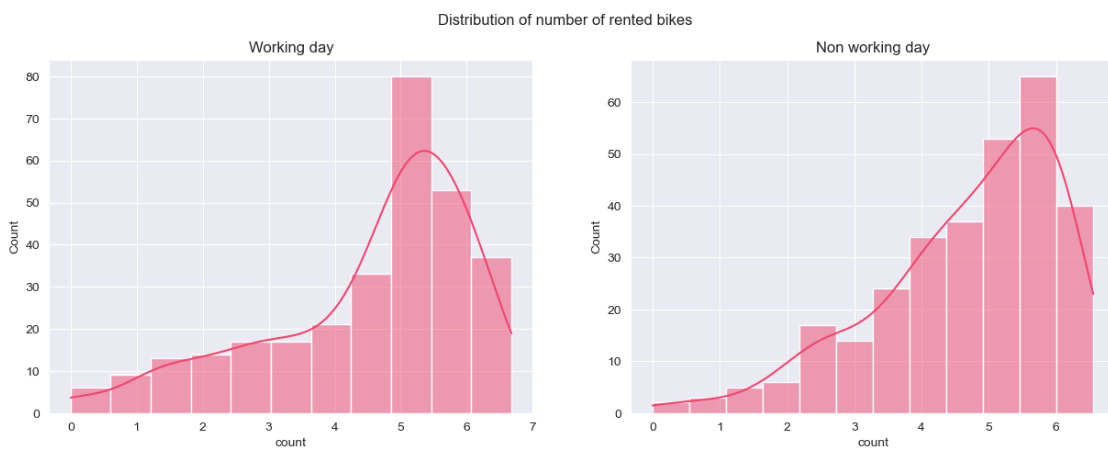


Inference:

1. The counts of rented cycles on both working and non-working days do not follow a normal distribution.
2. We can try to convert the distribution to normal by applying log transformation

5.1.2 Converting sample distribution to normal by applying log transformation

[42]:
```
plt.figure(figsize=(15,5))

#histogram for working day sample
plt.subplot(1,2,1)
sns.histplot(np.log(workingday_sample),kde=True,color='#ef476f')
plt.title('Working day')

#histogram for non working day sample
plt.subplot(1,2,2)
sns.histplot(np.log(nonworkingday_sample),kde=True,color='#ef476f')
plt.title('Non working day')

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



Inference:

Upon implementing a log transformation on our continuous variables, we observed a substantial improvement in achieving a distribution that closely resembles normality for both the workingday_sample and nonworkingday_sample

5.1.3 Performing the Wilk-Shapiro test for the workingday sample

We select the level of signifiance as 5% and the null and alternate hypothesis is as follows:

H0 : The Working day samples are normally distributed Ha: The Working day samples are not normally distributed

[43]:
```
test_stat,p_value= shapiro(np.log(workingday_sample))
print("test stat :",test_stat)
print("p value :",p_value)
alpha = 0.05
if p_value< alpha:
 print("Reject Ho: The working day samples are not normally distributed ")
else:
 print("Fail to Reject Ho: The working day samples are normally distributed")
```

```
test stat : 0.8912758827209473
p value : 8.112738589640192e-14
Reject Ho: The working day samples are not normally distributed
```

Inference:

From the above output, we see that the p value is far less than 0.05, Hence we reject the null hypothesis. We have sufficient evidence to say that the working day sample data does not come from normal distribution.

5.1.4 Performing the Wilk-Shapiro test for the non-working day sample

We select the level of signifiance as 5% and the null and alternate hypothesis is as follows:

H0 : The non working day samples are normally distributed Ha: The non working day samples are not normally distributed

[44]:
```
test_stat,p_value= shapiro(np.log(nonworkingday_sample))
print("test stat :",test_stat)
print("p value :",p_value)
alpha = 0.05
if p_value< alpha:
 print("Reject Ho: The non working day samples are not normally distributed ")
else:
 print("Fail to Reject Ho: The non working day samples are normally␣
 ↪distributed")
```

```
test stat : 0.9212508201599121
p value : 1.8011125577088727e-11
Reject Ho: The non working day samples are not normally distributed
```

Inference:

1. From the above output, we see that the p value is far less than 0.05, Hence we reject the null hypothesis.
2. We have sufficient evidence to say that the non working day sample data does not come from normal distribution.

5.1.5 Homegenity of Variance test : Levene's Test

We select the level of signifiance as 5% and the null and alternate hypothesis is as follows:

H0 : Variance is equal in both working day count and non working day count samples Ha: Variances is not equal

[45]:
```
test_stat,p_value= levene(np.log(workingday_sample),np.
 ↪log(nonworkingday_sample))
print("test stat :",test_stat)
print("p value :",p_value)
alpha = 0.05
if p_value< alpha:
 print("Reject Ho: Variance is not equal ")
else:
 print("Fail to Reject Ho: Variance is equal in both working day count and non␣
 ↪working day count samples")
```

```
test stat : 1.6823474489176666
p value : 0.19511302333521052
Fail to Reject Ho: Variance is equal in both working day count and non working
day count samples
```

Inference:

1. Since pvalue is not less than 0.05, we fail to reject null hypothesis.
2. This means we do not have sufficient evidence to say that variance across workingday count and non workingday count is significantly different thus making the assumption of homogenity of variances true

5.2 T-Test and final conclusion

1. 3 out of 4 assumptions for T test has been satified.
2. Although the sample distribution did not meet the criteria of passing the normality test, we proceed with the T-test as per the given instructions.

For T-Test we select the level of signifiance as 5% and the null and alternate hypothesis is as follows:

H0 : Working day does not have an effect on number of cycles rented Ha: Working day does have an effect on number of cycles rented

[46]:
```
t_stat,p_value= ttest_ind(np.log(workingday_sample),np.
 ↪log(nonworkingday_sample),equal_var=True)
print("f stat :",t_stat)
print("p value :",p_value)
alpha = 0.05
if p_value< alpha:
 print("Reject Ho: Working day does have an effect on number of cycles rented ")
else:
 print("Fail to Reject Ho: Working day does not have an effect on number of␣
 ↪cycles rented")
```

```
f stat : -1.3496662089486253
p value : 0.17763377005031294
Fail to Reject Ho: Working day does not have an effect on number of cycles
rented
```

Conclusion:

1. Since the p-value of our test is greater than alpha which is 0.05, we fail to reject the null hypothesis of this test.
2. We do not have sufficient evidence to conclude that working days have a significant effect on the number of cycles rented. This suggests that there is no significant difference in the number of cycles rented on working days versus non-working days.

6 Anova test

6.0.1 Are number of cycles rented similar or different in different weather conditions?

6.1 To perform such an analyis, we use ANOVA test:

1. ANOVA, which stands for Analysis of Variance, is a statistical technique used to assess whether there is a statistically significant difference among the means of two or more categorical groups. It achieves this by testing for variations in means by examining the variance within and between these groups.

2. The 4 different weather conditions are as follows:

   – Clear, Few clouds, partly cloudy, partly cloudy – Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist – Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds – Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

3. We have to check if there is any significant difference in the number of bikes rented across different weather conditions. To analyse this, we use Annova test.

[47]:
```
df['weather'].value_counts()
```

[47]:
```
1    7192
2    2834
3     859
4       1
Name: weather, dtype: int64
```

We shall setup the Null and Alternate Hypothesis to check if there is any effect of weather on the number of cycles rented.

1. H0 : The mean number of cycles rented is the same across all three different weather types.

2. Ha : There is at least one weather type with a mean number of cycles rented that significantly differs from the overall mean of the dependent variable.

Assumptions for ANOVA Test:

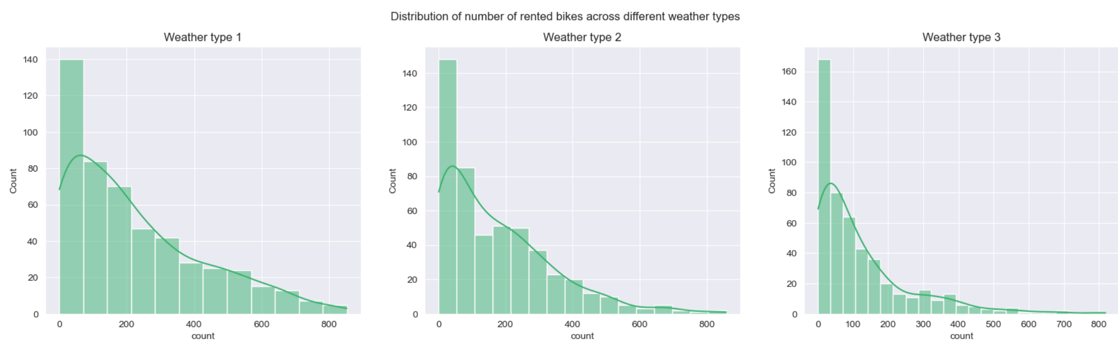
1. The distributions of data of each group should follow the Gaussian Distribution.

2. The variance of each group should be the same or close to each other.
3. The total n observations present should be independent of each other.

6.1.1 Normality Test: Shapiro-Wilk Test

[48]:
```
sample_1= df[df['weather']==1]['count'].sample(500)
sample_2 = df[df['weather']==2]['count'].sample(500)
sample_3 = df[df['weather']==3]['count'].sample(500)
```

[49]:
```
plt.figure(figsize=(20,5))

#histogram for weather condition 1
plt.subplot(1,3,1)
sns.histplot(sample_1,kde=True,color='mediumseagreen')
plt.title('Weather type 1')

#histogram for weather condition 2
plt.subplot(1,3,2)
sns.histplot(sample_2,kde=True,color='mediumseagreen')
plt.title('Weather type 2')

#histogram for weather condition 3
plt.subplot(1,3,3)
sns.histplot(sample_3,kde=True,color='mediumseagreen')
plt.title('Weather type 3')

plt.suptitle('Distribution of number of rented bikes across different weather␣
↪types')
plt.show()
```

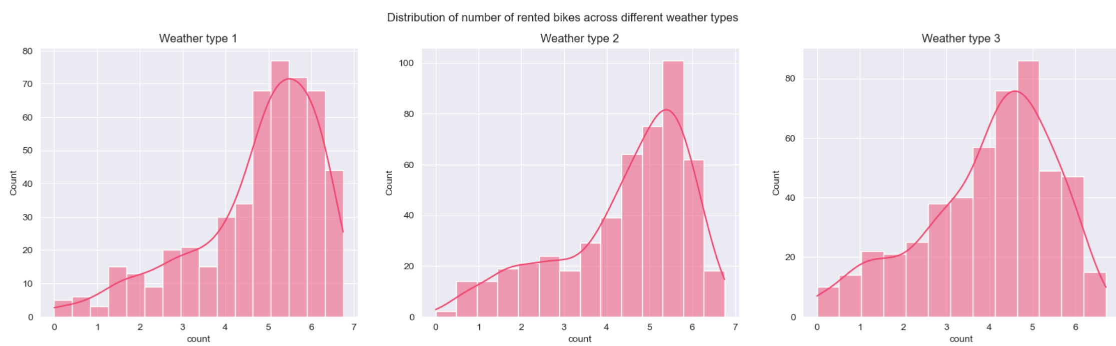


Inference:

We see that none of the graphs are normally distributed. Hence we apply log transformation to make these distributions near to normal

[50]:
```
log_1=np.log(sample_1)
log_2=np.log(sample_2)
log_3=np.log(sample_3)
```

[51]:
```
plt.figure(figsize=(20,5))

#histogram for weather condition 1
plt.subplot(1,3,1)
sns.histplot(log_1,kde=True,color='#ef476f')
plt.title('Weather type 1')

#histogram for weather condition 2
plt.subplot(1,3,2)
sns.histplot(log_2,kde=True,color='#ef476f')
plt.title('Weather type 2')

#histogram for weather condition 3
plt.subplot(1,3,3)
sns.histplot(log_3,kde=True,color='#ef476f')
plt.title('Weather type 3')

plt.suptitle('Distribution of number of rented bikes across different weather␣
↪types')
plt.show()
```



Inference:

After using a log transformation on the data for each weather type, we noticed a substantial improvement in making the data look more like a normal distribution.

6.1.2 Shapiro-Wilk Test for weather type 1 sample data

We select the level of signifiance as 5% and the null and alternate hypothesis is as follows:

1. H0 : The sample follows a normal distribution
2. Ha: The sample does not follow a normal distribution

[52]:
```
test_stat,p_value= shapiro(log_1)
print("test stat :",test_stat)
print("p value :",p_value)
alpha = 0.05
if p_value< alpha:
 print("Reject Ho: The sample does not follow a normal distribution")
else:
 print("Fail to Reject Ho:The sample follows a normal distribution")
```

```
test stat : 0.9003000259399414
p value : 1.5339812996889682e-17
Reject Ho: The sample does not follow a normal distribution
```

Inference:

Even after applying the log transformation, the sample does not conform to a normal distribution, as demonstrated by the Shapiro-Wilk test.

6.1.3 Shapiro-Wilk Test for weather type 2 sample data

We select the level of signifiance as 5% and the null and alternate hypothesis is as follows:

1. H0 : The sample follows a normal distribution
2. Ha : The sample does not follow a normal distribution

[53]:
```
test_stat,p_value= shapiro(log_2)
print("test stat :",test_stat)
print("p value :",p_value)
alpha = 0.05
if p_value< alpha:
 print("Reject Ho: The sample does not follow a normal distribution")
else:
 print("Fail to Reject Ho:The sample follows a normal distribution")
```

```
test stat : 0.9103980660438538
p value : 1.3363751305394404e-16
Reject Ho: The sample does not follow a normal distribution
```

Inference:

Even after applying the log transformation, the sample does not conform to a normal distribution, as demonstrated by the Shapiro-Wilk test.

6.1.4 Shapiro-Wilk Test for weather type 3 sample data

We select the level of signifiance as 5% and the null and alternate hypothesis is as follows:

1. H0 : The sample follows a normal distribution
2. Ha : The sample does not follow a normal distribution

[54]:
```
test_stat,p_value= shapiro(log_3)
print("test stat :",test_stat)
```

```
print("p value :",p_value)
alpha = 0.05
if p_value< alpha:
 print("Reject Ho: The sample does not follow a normal distribution")
else:
 print("Fail to Reject Ho:The sample follows a normal distribution")
```

```
test stat : 0.9491229057312012
p value : 4.342783077593282e-12
Reject Ho: The sample does not follow a normal distribution
```

Inference:

Even after applying the log transformation, the sample does not conform to a normal distribution, as demonstrated by the Shapiro-Wilk test.

Final Conclusion: None of the weather type samples adhere to a normal distribution even after applying the log-normal transformation, indicating that the normality assumption of the ANOVA test is not met.

6.1.5 Homegenity of Variance test : Levene's Test

We select the level of signifiance as 5% and the null and alternate hypothesis is as follows:

1. H0 : The variance is equal across all groups
2. Ha : The variance is not equal across the groups

[55]:
```
test_stat,p_value= levene(log_1,log_2,log_3,center='median')
print("test stat :",test_stat)
print("p value :",p_value)
alpha = 0.05
if p_value< alpha:
 print("Reject Ho: Variance is not equal across the groups ")
else:
 print("Fail to Reject Ho: Variance is equal across all groups")
```

```
test stat : 1.0396633834315243
p value : 0.3538288323243305
Fail to Reject Ho: Variance is equal across all groups
```

Inference:

1. Since pvalue is not less than 0.05, we fail to reject the null hypothesis.
2. This means we do not have sufficient evidence to claim a significant difference in variance across the different weather types. Therefore, the assumption of homogeneity of variances can be considered valid.

Final Conclusion:

1. Since the p-value obtained from our test is less than the predetermined alpha level of 0.05, we have sufficient evidence to reject the null hypothesis for this test.

2. Indeed, this indicates that we have collected sufficient evidence to conclude that there is a significant difference in the mean number of cycles rented across all weather conditions.
3. Additionally, this suggests that weather conditions do have a notable effect on the number of cycles rented.

6.2 Are number of cycles rented similar or different in different season ?

6.3 To perform such an analyis, we use ANOVA test.

1. ANNOVA, which stands for Analysis of Variance, is a statistical technique used to assess whether there is a statistically significant difference among the means of two or more categorical groups. It achieves this by testing for variations in means by examining the variance within and between these groups.
2. Here we have 4 different seasons namely spring,summer,fall & winter.
3. With the Annova test, we can find out if the different seasons have same or different effect amongst the number of cycles rented.

6.3.1 Formulating Null and Alternative Hypotheses¶

We shall setup the Null and Alternate Hypothesis to check if there is any effect of season on the number of cycles rented.

1. H0: All the 4 different seasons have equal means
2. Ha: There is atleast one season that differs significantly from the overall mean of dependent variable.

6.3.2 Normality Test: Wilkin Shapiro Test

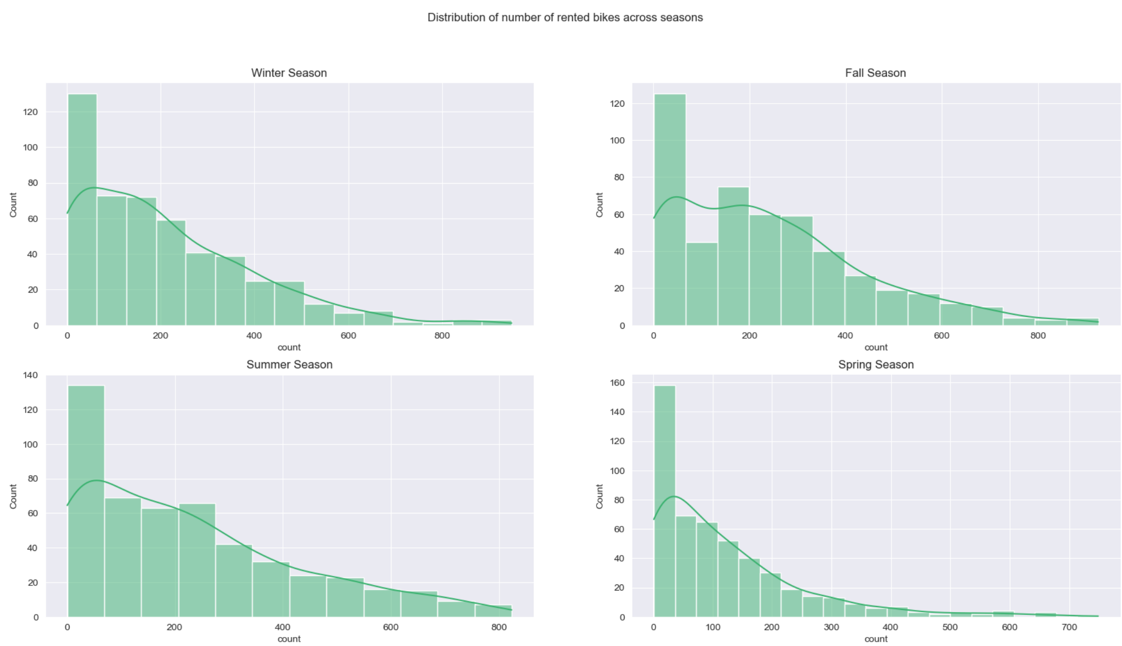
[74]:
```
df['season'].value_counts()
```

[74]:
```
winter    2734
fall      2733
summer    2733
spring    2686
Name: season, dtype: int64
```

[75]:
```
winter_sample=df[df['season']=='winter']['count'].sample(500)
fall_sample = df[df['season']=='fall']['count'].sample(500)
summer_sample = df[df['season']=='summer']['count'].sample(500)
spring_sample = df[df['season']=='spring']['count'].sample(500)
```

[76]:
```
plt.figure(figsize=(20,10))

#histogram for winter season
plt.subplot(2,2,1)
sns.histplot(winter_sample,kde=True,color='mediumseagreen')
plt.title('Winter Season')

#histogram for fall season
```

```python
plt.subplot(2,2,2)
sns.histplot(fall_sample,kde=True,color='mediumseagreen')
plt.title('Fall Season')

#histogram for summer season
plt.subplot(2,2,3)
sns.histplot(summer_sample,kde=True,color='mediumseagreen')
plt.title('Summer Season')

#histogram for spring season
plt.subplot(2,2,4)
sns.histplot(spring_sample,kde=True,color='mediumseagreen')
plt.title('Spring Season')

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

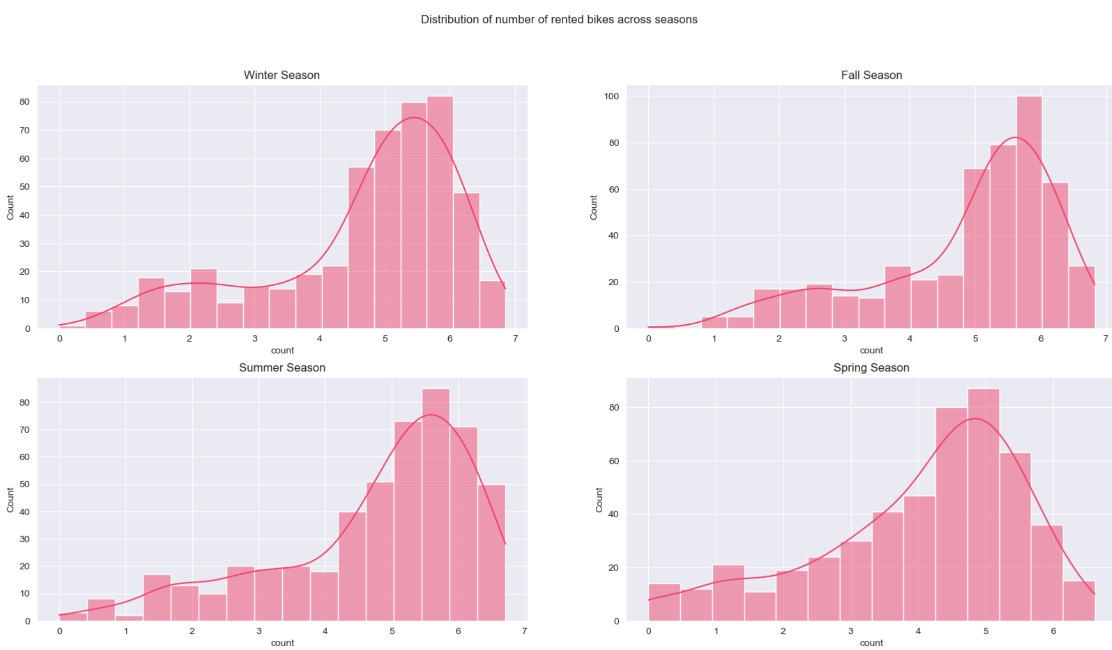


Inference:

We see that none of the graphs are normally distributed. Hence we apply log transformation to make these distributions near to normal

[77]:
```python
log_winter=np.log(winter_sample)
log_fall=np.log(fall_sample)
log_summer=np.log(summer_sample)
log_spring=np.log(spring_sample)
```

[78]:
```
plt.figure(figsize=(20,10))

#histogram for winter season
plt.subplot(2,2,1)
sns.histplot(log_winter,kde=True,color='#ef476f')
plt.title('Winter Season')

#histogram for fall season
plt.subplot(2,2,2)
sns.histplot(log_fall,kde=True,color='#ef476f')
plt.title('Fall Season')

#histogram for summer season
plt.subplot(2,2,3)
sns.histplot(log_summer,kde=True,color='#ef476f')
plt.title('Summer Season')

#histogram for spring season
plt.subplot(2,2,4)
sns.histplot(log_spring,kde=True,color='#ef476f')
plt.title('Spring Season')

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



Inference:

After applying a log transformation to the samples of each season, it can be inferred that a significant improvement was observed in achieving data distributions that closely resemble normality for each of the seasons.

6.3.3 Shapiro-Wilk Test for winter season sample data

We select the level of signifiance as 5% and the null and alternate hypothesis is as follows:

1. H0 : The sample follows a normal distribution
2. Ha: The sample does not follow a normal distribution

[79]:
```
test_stat,p_value= shapiro(log_winter)
print("test stat :",test_stat)
print("p value :",p_value)
alpha = 0.05
if p_value< alpha:
 print("Reject Ho: The sample does not follow a normal distribution")
else:
 print("Fail to Reject Ho:The sample follows a normal distribution")
```

```
test stat : 0.885298490524292
p value : 8.109409374782921e-19
Reject Ho: The sample does not follow a normal distribution
```

Inference:

Even after applying the log transformation, the sample does not conform to a normal distribution, as demonstrated by the Shapiro-Wilk test.

6.3.4 Shapiro-Wilk Test for fall season sample data

We select the level of signifiance as 5% and the null and alternate hypothesis is as follows:

1. H0 : The sample follows a normal distribution
2. Ha: The sample does not follow a normal distribution

[81]:
```
test_stat,p_value= shapiro(log_fall)
print("test stat :",test_stat)
print("p value :",p_value)
alpha = 0.05
if p_value< alpha:
 print("Reject Ho: The sample does not follow a normal distribution")
else:
 print("Fail to Reject Ho:The sample follows a normal distribution")
```

```
test stat : 0.8858478665351868
p value : 8.985255607711857e-19
Reject Ho: The sample does not follow a normal distribution
```

Inference:

Even after applying the log transformation, the sample does not follow a normal distribution, as demonstrated by the Shapiro-Wilk test.

6.3.5 Shapiro-Wilk Test for summer season sample data

We select the level of signifiance as 5% and the null and alternate hypothesis is as follows:

1. H0 : The sample follows a normal distribution
2. Ha: The sample does not follow a normal distribution

[82]:
```
test_stat,p_value= shapiro(log_summer)
print("test stat :",test_stat)
print("p value :",p_value)
alpha = 0.05
if p_value< alpha:
 print("Reject Ho: The sample does not follow a normal distribution")
else:
 print("Fail to Reject Ho:The sample follows a normal distribution")
```

```
test stat : 0.8904457092285156
p value : 2.1507241865583005e-18
Reject Ho: The sample does not follow a normal distribution
```

Inference:

Even after applying the log transformation, the sample does not follow a normal distribution, as demonstrated by the Shapiro-Wilk test.

6.3.6 Shapiro-Wilk Test for spring season sample data

We select the level of signifiance as 5% and the null and alternate hypothesis is as follows:

1. H0 : The sample follows a normal distribution
2. Ha: The sample does not follow a normal distribution

[83]:
```
test_stat,p_value= shapiro(log_spring)
print("test stat :",test_stat)
print("p value :",p_value)
alpha = 0.05
if p_value< alpha:
 print("Reject Ho: The sample does not follow a normal distribution")
else:
 print("Fail to Reject Ho:The sample follows a normal distribution")
```

```
test stat : 0.9281446933746338
p value : 9.541294214128818e-15
Reject Ho: The sample does not follow a normal distribution
```

Inference:

Even after applying the log transformation, the sample does not follow a normal distribution, as demonstrated by the Shapiro-Wilk test

Homegenity of Variance test : Levene's Test We select the level of signifiance as 5% and the null and alternate hypothesis is as follows:

H0 : The variance is equal across all groups Ha : The variance is not equal across the groups

[84]:
```
test_stat,p_value=␣
 ↪levene(log_winter,log_fall,log_summer,log_spring,center='median')
print("test stat :",test_stat)
print("p value :",p_value)
alpha = 0.05
if p_value< alpha:
 print("Reject Ho: Variance is not equal across the groups ")
else:
 print("Fail to Reject Ho: Variance is equal across all groups")
```

```
test stat : 1.6725956851748855
p value : 0.17086624597649322
Fail to Reject Ho: Variance is equal across all groups
```

Inference:

Since pvalue is not less than 0.05, we fail to reject the null hypothesis. This means we do not have sufficient evidence to claim a significant difference in variance across the different seasons. Therefore, the assumption of homogeneity of variances can be considered valid.

6.3.7 ANOVA Test and Final Conclusion¶

2 out of 3 assumptions for ANOVA test have been satisfied. We continue to do the test since we have been instructed to do so.

For ANOVA Test we select the level of signifiance as 5% and the null and alternate hypothesis is as follows:

1. H0 : The mean number of cycles rented is the same across different seasons
2. Ha: At least one season has a mean number of cycles rented that is significantly different from the others.

[85]:
```
f_stat,p_value= f_oneway(log_winter,log_fall, log_summer,log_spring)
print("f stat :",f_stat)
print("p value :",p_value)
alpha = 0.05
if p_value< alpha:
 print("Reject Ho: At least one season has a mean number of cycles rented that␣
 ↪is significantly different from the others. ")
else:
 print("Fail to Reject Ho: The mean number of cycles rented is the same across␣
 ↪different seasons ")
```

```
f stat : 33.02568358826368
p value : 7.879540191567301e-21
```

```
Reject Ho: At least one season has a mean number of cycles rented that is
significantly different from the others.
```

Conclusion:

1. Since the p-value obtained from our test is less than the predetermined alpha level of 0.05, we have sufficient evidence to reject the null hypothesis for this test.
2. Indeed, this implies that we have gathered enough evidence to conclude that there is a significant difference in the mean number of cycles rented across all seasons.

7 Chi-square test

7.0.1 Is weather type dependent on the season

7.0.2 To perform such an analysis we perform Chi square test

A Chi-Square Test of Independence is used to determine whether or not there is a significant association between two categorical variables

The Pearson's Chi-Square statistical hypothesis is a test for independence between categorical variables.

Also it is important to know that Chi-Square test is non parametric test meaning that it is distribution free (need not have gaussian distribution)

[67]:
```
data=pd.crosstab(df['weather'],df['season'])
data
```

[67]:

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

[68]:
```
df_removed_weather=df[~(df['weather']==4)]
```

[69]:
```
data=pd.crosstab(df_removed_weather['weather'],df_removed_weather['season'])
data
```

[69]:

season weather	fall	spring	summer	winter
1	1930	1759	1801	1702
2	604	715	708	807
3	199	211	224	225

[70]:
```
x_stat,p_value,dof,expected=chi2_contingency(data)
```

[71]:
```
expected.min()
```

[71]: 211.88929719797886

[72]:
```
(len(expected[expected<5])/len(expected))*100
```

[72]: 0.0

Inference:

All of the data points have expected values greater than 5, indicating that the assumption related to the expected values being greater than 5 is satisfied for the chi-square test.

7.0.3 Chi-Square Test and Final Conclusion¶

We shall setup Null and alternate Hypotheis to check if Weather is dependent on season

1. H0: Weather is not dependent on the season
2. Ha: Weather is dependent on the season, meaning they are associated or related

We consider level of significance as 0.05

[73]:
```
x_stat,p_value,dof,expected=chi2_contingency(data)
print("X stat :",x_stat)
print("p value :",p_value)
alpha = 0.05
if p_value< alpha:
 print("Reject Ho: Weather is dependent on the season")
else:
 print("Fail to Reject Ho: Weather is not dependent on the season")
```

```
X stat : 46.101457310732485
p value : 2.8260014509929403e-08
Reject Ho: Weather is dependent on the season
```

7.1 Final Conclusion

1. Since the p-value obtained from our test is less than the predetermined alpha level of 0.05, we have sufficient evidence to reject the null hypothesis for this test.
2. Indeed, this suggests that we have gathered enough evidence to conclude that there is a dependence between weather and the season.

8 Insights

1. On working days, 68.6% of cycles are rented, whereas on non-working days, 31.4% of cycles are rented.
2. Despite the fact that 68.6% of cycles are rented on working days compared to 31.4% on non-working days, our t-test analysis does not provide sufficient evidence to conclude that working days have a significant effect on the number of cycles rented. This finding suggests that there is no statistically significant difference in the number of cycles rented between working days and non-working days.
3. During the fall season, approximately 30.7% of cycles are rented.
4. In the summer season, around 28.2% of cycles are rented.
5. The winter season records a rental rate of about 26.1% for cycles.

6. The lowest rental rate, at just 15%, is observed in the spring season.
7. The ANOVA test results indicate a clear and statistically significant difference in the mean number of cycles rented across all seasons. This underscores the notion that the season plays a substantial role in influencing the number of bikes rented.
8. Weather condition 1 experiences the highest rental rate, with approximately 70.8% of cycles rented.
9. In weather condition 2, around 24.3% of cycles are rented.
10. Weather condition 3 has a rental rate of approximately 4.9% for cycles
11. Weather condition 4 exhibits an exceptionally low rental rate, with only 0.00786% of cycles being rented.
12. The ANOVA test results indicate a significant difference in the mean number of cycles rented across all weather conditions, which strongly suggests that weather types have a significant effect on the number of cycles rented.
13. The chi-square test results reveal a statistically significant association between weather type and the season.
14. The highest average count of rental bikes is observed at 5 PM, closely followed by 6 PM and 8 AM. This indicates distinct peak hours during the day when cycling is most popular.
15. Conversely, the lowest average count of rental bikes occurs at 4 AM, with 3 AM and 5 AM also showing low counts. These hours represent the early morning period with the least demand for cycling.
16. Notably, there is an increasing trend in cycle rentals between 5 AM and 8 AM, suggesting a surge in demand during the early morning hours as people start their day.
17. Additionally, there is a decreasing trend in cycle rentals from 5 PM to 11 PM, indicating a gradual decline in demand as the day progresses into the evening and nighttime.
18. The highest average hourly count of rental bikes occurs in June, July, and August, reflecting the peak demand during summer.
19. Conversely, the lowest average hourly count of rental bikes is found in January, February, and March, which are the winter months with reduced cycling activity.
20. Notably, there is an increasing trend in average bike rentals from February to June, corresponding to the shift from winter to spring and summer.
21. Conversely, a decreasing trend in average bike rentals is observed from October to December due to the onset of winter.
22. The weak positive correlation of 0.39 between temperature and the number of bikes rented suggests that, on average, fewer people prefer to use electric cycles during the daytime between 12 PM to 3 PM. This observation aligns with our univariate analysis, where we discovered that the average number of cycles rented during this time frame was lower compared to other times of the day. A similar correlation pattern is also observed in the case of "feels-like" temperature, reinforcing this trend.
23. The negative correlation between humidity and the number of cycles rented indicates that people tend to avoid using electric bikes during high humidity conditions. This avoidance can be attributed to the discomfort caused by the heavy and sticky air, leading to sweating and a general sense of unease. Moreover, the reduced efficiency of electric bikes in high humidity, resulting in increased air resistance and potential battery performance issues, contributes to the preference for alternative transportation or indoor activities in such conditions.
24. The presence of a weak positive correlation between windspeed and the number of cycles rented indicates that there is a subset of individuals who appear to favor using electric cycles during windy conditions for the sheer enjoyment of the experience. While this preference contributes to a slight increase in bike rentals on windier days, it's essential to recognize that

this effect is not particularly strong, as indicated by the weak correlation. This suggests that the enjoyment of cycling in windy conditions is a relatively niche preference among riders.

9 Recommendations

1. Actionable Insight : Despite the fact that 68.6% of cycles are rented on working days compared to 31.4% on non-working days, our t-test analysis does not provide sufficient evidence to conclude that working days have a significant effect on the number of cycles rented. This finding suggests that there is no statistically significant difference in the number of cycles rented between working days and non-working days.

Recommendations:

Yulu can consider adjusting its fleet allocation and marketing efforts to better align with customer demand. While working days might not significantly impact rentals, Yulu can focus on peak hours within both working and non-working days to ensure bikes are available when and where they are needed most. Yulu can engage with users through notifications and alerts to inform them of bike availability and incentives during specific timeframes, encouraging rentals during periods with lower demand.

2. Actionable Insight : The ANOVA test results indicate a clear and statistically significant difference in the mean number of cycles rented across all seasons. This underscores the notion that the season plays a substantial role in influencing the number of bikes rented.

Recommendations:

Yulu can introduce season-specific promotions and discounts to incentivize bike rentals during peak seasons. For example, offering discounts during summer to encourage more rides can attract additional customers. Ensure that bike availability is well-managed to meet the increased demand during peak seasons. This includes bike maintenance, distribution, and tracking to prevent shortages or excess bikes. During seasons with adverse weather conditions, such as rain or snow, Yulu can provide weather-ready bikes equipped with features like fenders and all-weather tires. This ensures that riders are comfortable and safe during inclement weather.

3. Actionable Insight : The ANOVA test results indicate a significant difference in the mean number of cycles rented across all weather conditions, which strongly suggests that weather types have a significant effect on the number of cycles rented.

Recommendations:

During seasons with adverse weather conditions, such as rain or snow, Yulu can provide weather-ready bikes equipped with features like fenders and all-weather tires. This ensures that riders are comfortable and safe during inclement weather. During adverse weather, prioritize rider safety by providing guidelines and recommendations for riding in specific conditions. Ensure bikes are well-maintained and equipped with safety features.

4. Actionable Insight: The lowest average count of rental bikes occurs from 1 am to 4 am.

Recommendations:

1. Designate the hours from 1 am to 4 am as the primary maintenance window. During this time, perform routine checks and maintenance on the bikes, ensuring they are in optimal

condition for the next day's rentals. This includes inspecting brakes, tires, gears, and electric components (if applicable).

2. As a significant number of bikes are likely to be unused during these hours, Yulu can capitalize on this downtime to charge electric bike batteries. Implement a comprehensive battery charging program, ensuring that all electric bikes are fully charged and ready for use during peak hours.
3. After maintenance and charging, strategically deploy bikes to high-demand areas in preparation for the morning rush. Ensure that bikes are available at key locations, such as transportation hubs, offices, and residential areas.
4. Use data analytics to fine-tune this strategy over time. Analyze bike utilization patterns, maintenance needs, and charging efficiency to continually optimize the process.
5. By leveraging this off-peak time for maintenance and charging, Yulu can enhance bike availability during peak hours, improve customer satisfaction, and maximize operational efficiency.

[ ]:

```

```