Bikesharing_Company_Demand_Forecasting_by_Diptyajit_Das

April 14, 2024

1 Problem Statement: Analyzing Factors Affecting Demand for Shared Electric Cycles

A prominent micro-mobility service provider in India aims to understand the determinants influencing demand for its shared electric cycles. By analyzing various factors such as urban infrastructure, commuting habits, economic indicators, and environmental conditions, we aim to uncover insights that can drive strategic decisions and enhance the adoption of sustainable transportation solutions.

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
from scipy.stats import ttest_ind # T-test for independent samples
from scipy.stats import shapiro # Shapiro-Wilk’s test for Normality
from scipy.stats import levene # Levene's test for Equality of Variance
from scipy.stats import f_oneway # One-way ANOVA

#Non-Paramteric
from scipy.stats import kruskal
from scipy.stats import mannwhitneyu

from scipy.stats import chi2_contingency # Chi-square test of independence
```

[3]:
```
#Regression
from statsmodels.formula.api import ols
import statsmodels.api as sm
```

[4]:
```
#Warnings
import warnings
warnings.simplefilter('ignore')
```

[5]:
```
df=pd.read_csv('bike_sharing.csv')
```

[6]:
```
df.head()
```

[6]:
```
              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
```

2 Part 1 : Structure

[7]:
```
df.shape
```

[7]:
```
(10886, 12)
```

[8]:
```
df.describe()
```

[8]:
```
             season       holiday    workingday       weather         temp  \
count  10886.000000  10886.000000  10886.000000  10886.000000  10886.00000
mean       2.506614      0.028569      0.680875      1.418427     20.23086
std        1.116174      0.166599      0.466159      0.633839      7.79159
min        1.000000      0.000000      0.000000      1.000000      0.82000
25%        2.000000      0.000000      0.000000      1.000000     13.94000
50%        3.000000      0.000000      1.000000      1.000000     20.50000
75%        4.000000      0.000000      1.000000      2.000000     26.24000
max        4.000000      1.000000      1.000000      4.000000     41.00000

              atemp      humidity     windspeed        casual    registered  \
count  10886.000000  10886.000000  10886.000000  10886.000000  10886.000000
mean      23.655084     61.886460     12.799395     36.021955    155.552177
std        8.474601     19.245033      8.164537     49.960477    151.039033
min        0.760000      0.000000      0.000000      0.000000      0.000000
25%       16.665000     47.000000      7.001500      4.000000     36.000000
50%       24.240000     62.000000     12.998000     17.000000    118.000000
75%       31.060000     77.000000     16.997900     49.000000    222.000000
max       45.455000    100.000000     56.996900    367.000000    886.000000

              count
count  10886.000000
mean     191.574132
std      181.144454
min        1.000000
25%       42.000000
50%      145.000000
```

```
75%      284.000000
max      977.000000
```

2.1 10886 rows and 12 columns

[9]:
```python
df.isna().sum()
```

[9]:
```
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
```

[10]:
```python
len(df[df.duplicated()])
```

[10]:
```
0
```

[11]:
```python
df.dtypes
```

[11]:
```
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
```

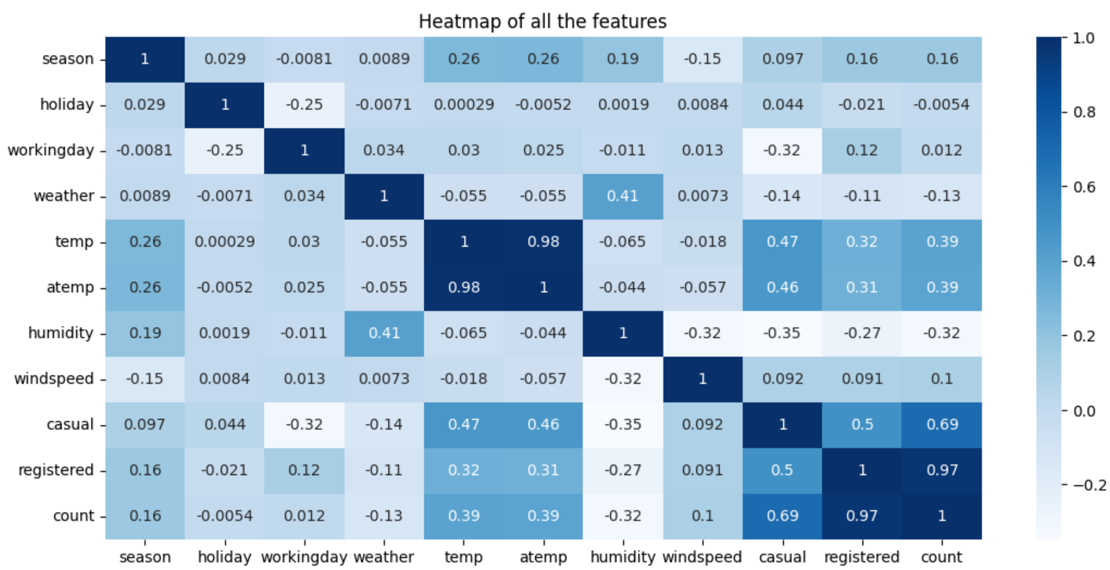
2.1.1 Dataset Summary

- **datetime**: Date and time of the observation (datetime)
- **season**: Season of the year (1: spring, 2: summer, 3: fall, 4: winter)
- **holiday**: Whether the day is a holiday or not (0: not a holiday, 1: holiday)
- **workingday**: Whether the day is a working day (0: weekend or holiday, 1: working day)

- **weather**: Weather condition (1: Clear, Few clouds, partly cloudy, partly cloudy, 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist, 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds, 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog)
- **temp**: Temperature in Celsius (float)
- **atemp**: Feeling temperature in Celsius (float)
- **humidity**: Humidity (integer)
- **windspeed**: Wind speed (float)
- **casual**: Count of casual users (integer)
- **registered**: Count of registered users (integer)
- **count**: Total count of rental bikes including both casual and registered users (integer)

3 Part 2 : Relationship

[12]:
```
plt.figure(figsize=(13, 6))
sns.heatmap(df[['season', 'holiday', 'workingday', 'weather', 'temp',
        'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count']].
 ↪corr(),annot=True,cmap='Blues')
plt.title('Heatmap of all the features')
plt.show()
```

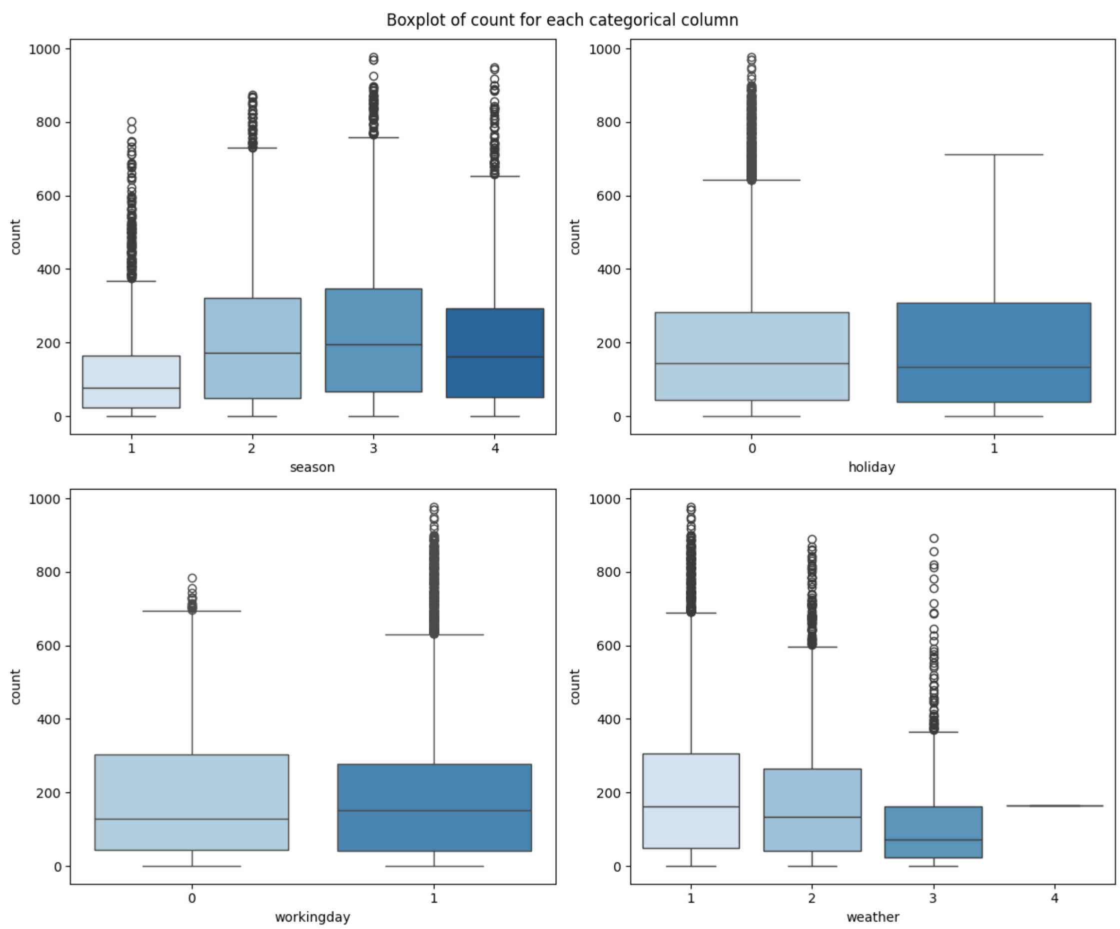


- Apparent temperature(atemp) and temperature(temp) are strongly positively correlated as they are similar measures.
- Casual and registered and total count are strongly positively correlated which is expected.
- Temperature and casual riders count are mildly positively correlated.

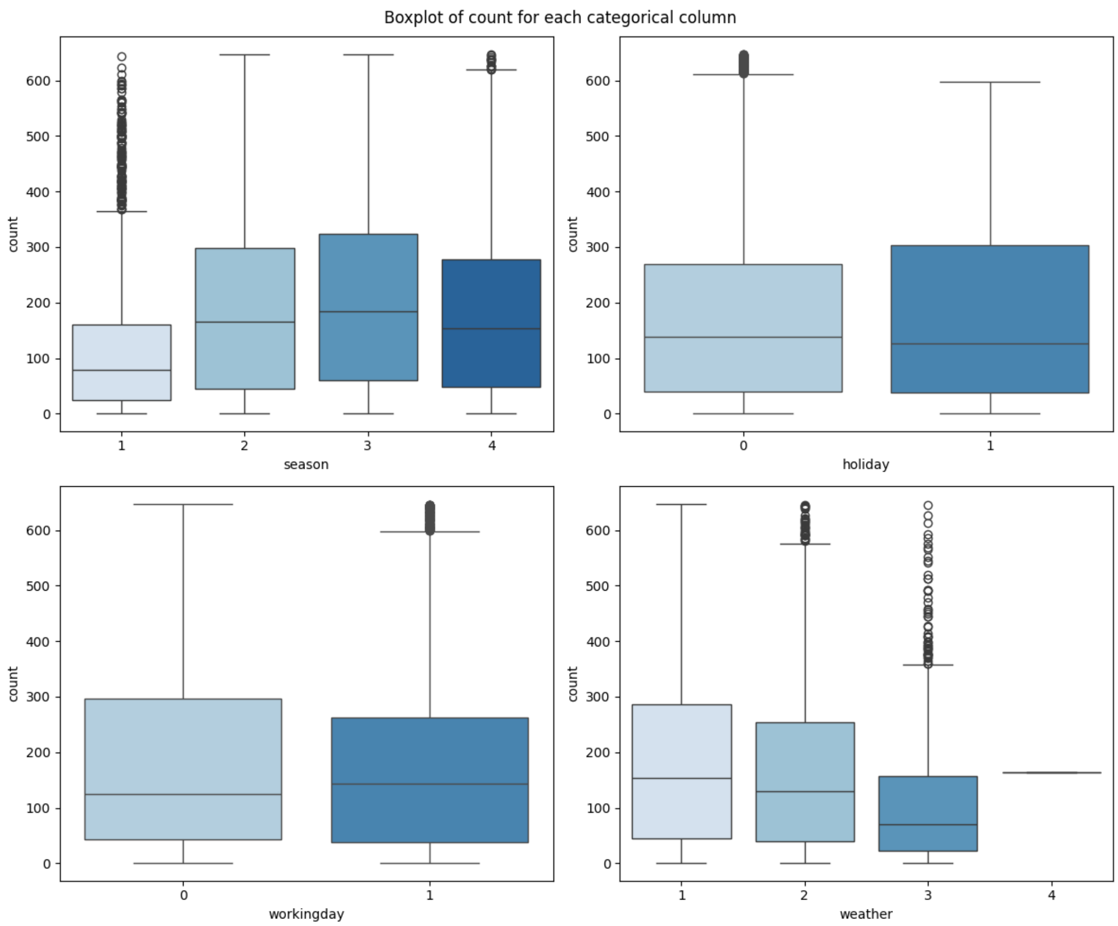
3.1 Dropping 'atemp' column and keeping 'casual' and 'registered' to use for regression.

[13]:
```
df.drop(columns=['atemp'],inplace=True)
```

[14]:
```
def outlier_checker():
    fig, ax = plt.subplots(2, 2, figsize=(12, 10))
    cols = ['season', 'holiday', 'workingday', 'weather']

    for i in range(len(cols)):
        c = cols[i]
        sns.boxplot(data=df, x=c, y='count', ax=ax[i // 2, i %␣
↪2],palette='Blues')
    plt.suptitle('Boxplot of count for each categorical column')
    plt.tight_layout()
    plt.show()
outlier_checker()
```

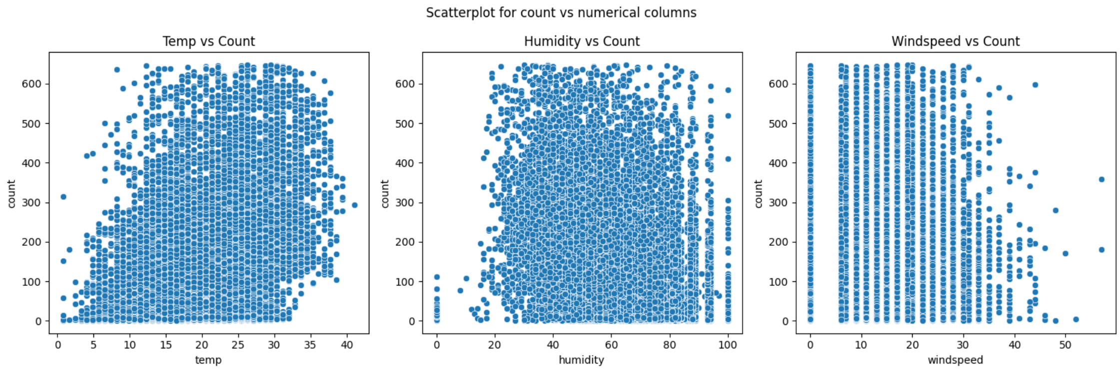


[15]:
```
def remove_outliers_iqr(data, column):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return data[(data[column] >= lower_bound) & (data[column] <= upper_bound)]

df = remove_outliers_iqr(df, 'count')
outlier_checker()
```

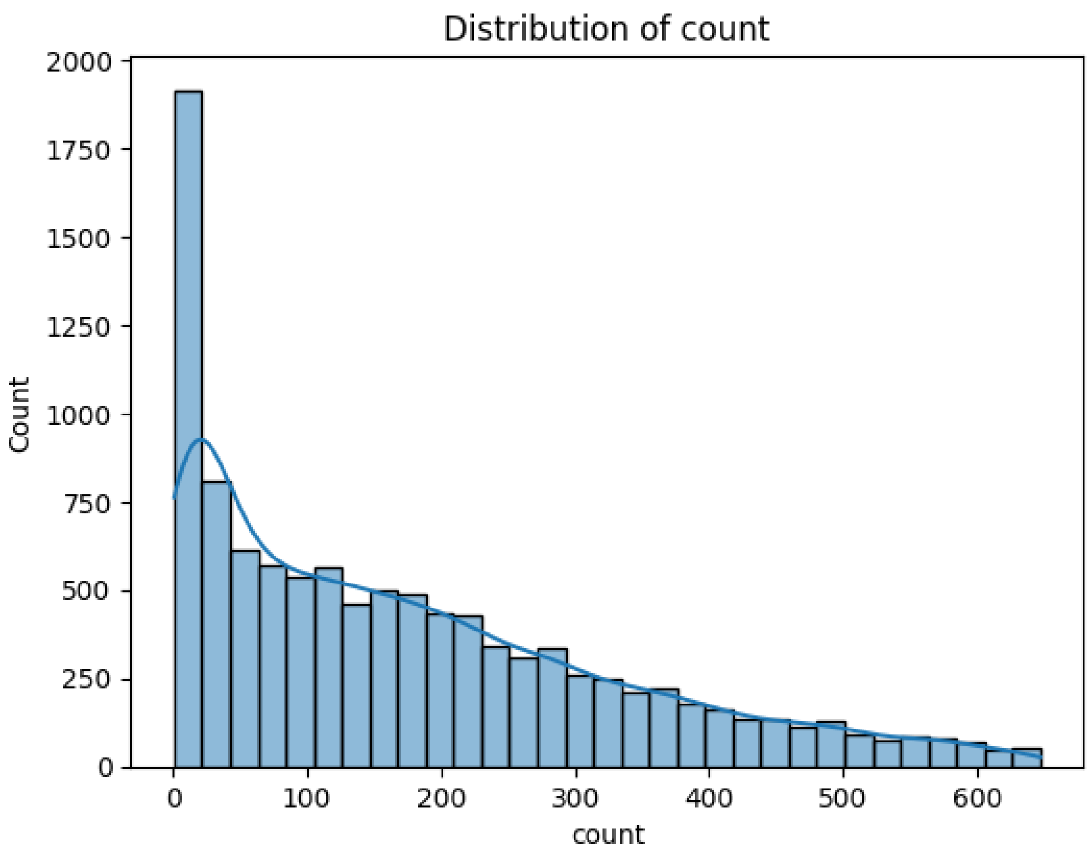


[16]:
```
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

ncols = ['temp', 'humidity', 'windspeed']

for i, col in enumerate(ncols):
    sns.scatterplot(data=df, x=col, y='count', ax=axes[i],palette='Blues')
    axes[i].set_title(f'{col.capitalize()} vs Count')
```

```
plt.suptitle('Scatterplot for count vs numerical columns')
plt.tight_layout()
plt.show()
```

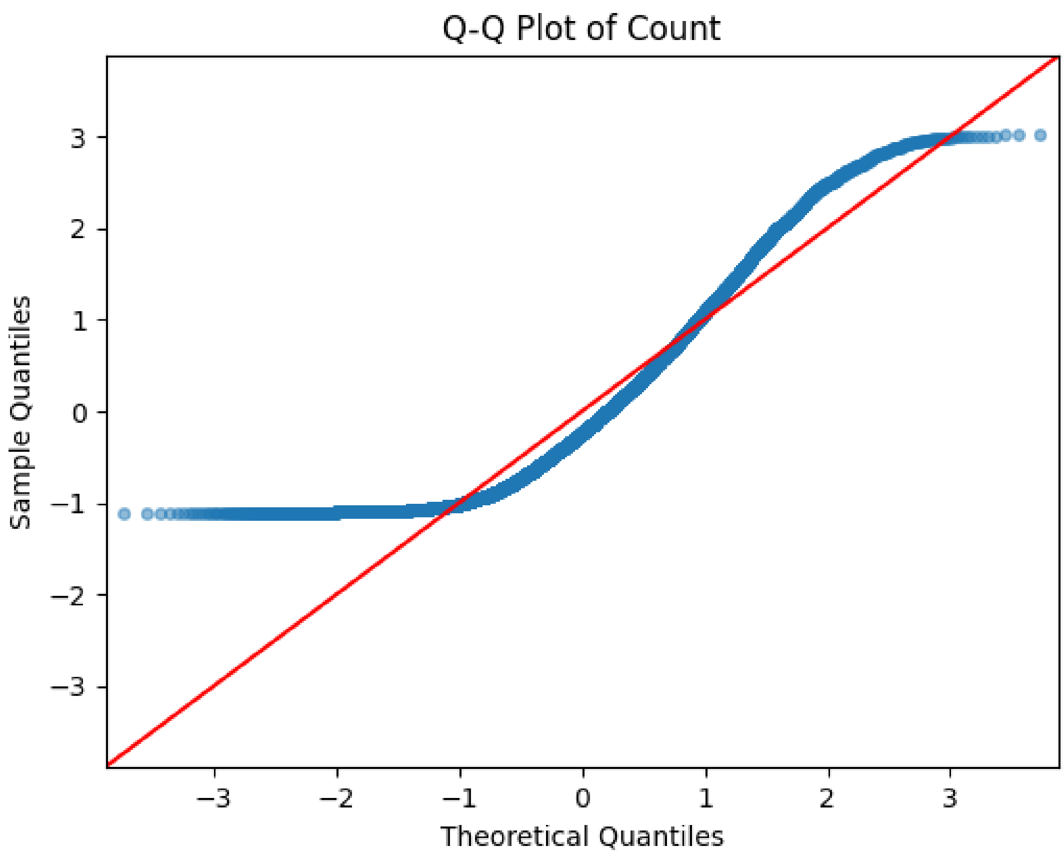


[17]:
```
sns.histplot(data=df,x='count',kde=True,palette='Blues')
plt.title('Distribution of count')
plt.show()
```



[18]:

```
normal_distribution = sm.ProbPlot(df['count'], fit=True)

qq_plot = normal_distribution.qqplot(line='45', alpha=0.5, color='blue',␣
↪markersize=4)

plt.title('Q-Q Plot of Count')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')

plt.show()
```



3.2 There is no missing or duplicated data.

3.3 String column: 'datetime'

3.4 Integer columns: 'season', 'holiday', 'workingday', 'weather', 'humidity', 'casual', 'registered', 'count'

3.5 Float columns: 'temp', 'atemp', 'windspeed'

3.6 After iqr treatment outliers have decreased but count is still right skewed so continuing with the original data.

3.7 Significance level alpha is considered as .05 if not mentioned otherwise.

[19]:
```
df=pd.read_csv('bike_sharing.csv')
```

4 Part 3: Is the average count of bike rides higher on working days compared to non-working days?

[20]:
```
df.groupby('workingday')['count'].describe()
```

[20]:

workingday	count	mean	std	min	25%	50%	75%	max
0	3474.0	188.506621	173.724015	1.0	44.0	128.0	304.0	783.0
1	7412.0	193.011873	184.513659	1.0	41.0	151.0	277.0	977.0

$H_0 : \mu_w <= \mu_n$

$H_1 : \mu_w > \mu_n$

- μ_w is average count in working days and μ_n is average count in non-working days.

[21]:
```
w,n=df[df['workingday']==1].
↪sample(8000,random_state=95,replace=True)['count'],df[df['workingday']==0].
↪sample(4000,random_state=95,replace=True)['count']
```

4.0.1 Check for equal variance

[22]:
```
levene(w,n)
```

[22]: LeveneResult(statistic=0.3356469728273234, pvalue=0.5623635910302649)

4.0.2 Equal variances as pvalue of Levene test > .05

[23]:
```
ttest_ind(w,n,alternative='greater')
```

[23]: TtestResult(statistic=1.7405913020723884, pvalue=0.0408904398116283, df=11998.0)

4.1 At slightly higher size of samples pvalue<.05 so we reject null. We conclude that average count of rides in workingdays is greater than that of non working days.

4.2 Is the average count of bike rides higher on regular days compared to holidays?

[24]:
```
df.groupby('holiday')['count'].describe()
```

[24]:

holiday	count	mean	std	min	25%	50%	75%	max
0	10575.0	191.741655	181.513131	1.0	43.0	145.0	283.0	977.0
1	311.0	185.877814	168.300531	1.0	38.5	133.0	308.0	712.0

$H_0 : \mu_r <= \mu_h$

$H_1 : \mu_r > \mu_h$

- μ_r is average count in regular days and μ_h is average count in holidays.

[25]:
```
r,h=df[df['holiday']==0].
↪sample(20000,random_state=95,replace=True)['count'],df[df['holiday']==1].
↪sample(10000,random_state=95,replace=True)['count']
```

4.2.1 Check for equal variance

[26]:
```
levene(r,h)
```

[26]:
```
LeveneResult(statistic=0.31476547365524954, pvalue=0.5747747075542864)
```

4.2.2 Equal variances as pvalue of Levene test > .05

[27]:
```
ttest_ind(r,h,alternative='greater')
```

[27]:
```
TtestResult(statistic=1.979267953300377, pvalue=0.023897486766671795,
df=29998.0)
```

4.3 At slightly higher size of samples we get a pvalue <.05. So we reject null concluding that more average bike rides happen in regular days than in holidays.

5 Part 4 : Is the demand of bicycles on rent same for different weather conditions?

[28]:
```
df.groupby('weather')['count'].describe()
```

[28]:

weather	count	mean	std	min	25%	50%	75%	max
1	7192.0	205.236791	187.959566	1.0	48.0	161.0	305.0	977.0

2	2834.0	178.955540	168.366413	1.0	41.0	134.0	264.0	890.0
3	859.0	118.846333	138.581297	1.0	23.0	71.0	161.0	891.0
4	1.0	164.000000	NaN	164.0	164.0	164.0	164.0	164.0

5.0.1 We can exclude weather category 4 since there is only one record.

[29]:
```
w1 = df[df['weather'] == 1].sample(5000, random_state=95, replace=True)['count']
w2 = df[df['weather'] == 2].sample(3000, random_state=95, replace=True)['count']
w3 = df[df['weather'] == 3].sample(1000, random_state=95, replace=True)['count']
```

$H_0 : \mu_1 = \mu_2 = \mu_3$

H_1: Average mean counts of the three weather conditions are not equal.

- μ_1, μ_2, μ_3 are the mean counts for weather conditions 1,2,3 respectively.

5.0.2 Normality and equal variance of each group check

[30]:
```
shapiro(df.sample(100,random_state=95,replace=True)['count'])
```

[30]:
```
ShapiroResult(statistic=0.8826003074645996, pvalue=2.34207959692867e-07)
```

[31]:
```
levene(w1,w2,w3)
```

[31]:
```
LeveneResult(statistic=74.88710632006602, pvalue=5.554991383504963e-33)
```

5.1 One-way ANOVA

[32]:
```
f_oneway(w1,w2,w3)
```

[32]:
```
F_onewayResult(statistic=92.62935313319629, pvalue=1.5141150091398818e-40)
```

5.1.1 Clearly target variable is not normal and also the groups donot have equal variance so ANOVA results are not trustworthy.

5.1.2 Using (non-parametric) Kruskal Wallis test.

[33]:
```
kruskal(w1,w2,w3)
```

[33]:
```
KruskalResult(statistic=203.04619280997002, pvalue=8.111094207044942e-45)
```

5.2 Pvalue is well below .05 so we can reject null and conclude that average rides count is different for different weather conditions.

[61]:
```
label_map = {'w1': w1, 'w2': w2, 'w3': w3}
pairs = [('w1', 'w2'), ('w1', 'w3'), ('w2', 'w3')]
for i, (group1, group2) in enumerate(pairs, start=1):
    data1 = label_map[group1]
    data2 = label_map[group2]
```

```python
    print(f"Pair {i}: ({group1}, {group2})")
    #Levene Test
    levene_statistic, levene_pvalue = levene(data1, data2)
    print(f"Levene Test p-value: {levene_pvalue}")

    # Determine equality of variance based on Levene test result
    if levene_pvalue < 0.01:
        equal_var = False
    else:
        equal_var = True

    # Two Sample Independent T Test
    statistic, p_value = ttest_ind(data1, data2, equal_var=equal_var,␣
↪alternative='greater')
    print(f"2 Sample Independent T Test Statistic: {statistic}")
    print(f"P-value: {p_value}")

    # Check for significance based on p-value
    alpha = 0.05/3 #Bonferroni correction
    if p_value < alpha:
        print("Reject the null hypothesis. 1st group mean is significantly␣
↪greater than 2nd group mean.")
    else:
        print("Fail to reject the null hypothesis. 1st group mean is␣
↪significantly smaller or equal than 2nd group mean.")

    print()
```

```
Pair 1: (w1, w2)
Levene Test p-value: 3.921264841454833e-09
2 Sample Independent T Test Statistic: 5.666545284713696
P-value: 7.58111517905297e-09
Reject the null hypothesis. 1st group mean is significantly greater than 2nd
group mean.

Pair 2: (w1, w3)
Levene Test p-value: 9.725848590143342e-31
2 Sample Independent T Test Statistic: 15.632137392336668
P-value: 5.590532733876443e-52
Reject the null hypothesis. 1st group mean is significantly greater than 2nd
group mean.

Pair 3: (w2, w3)
Levene Test p-value: 5.426106244289857e-15
2 Sample Independent T Test Statistic: 10.793830501966399
P-value: 9.797281641825225e-27
Reject the null hypothesis. 1st group mean is significantly greater than 2nd
```

group mean.

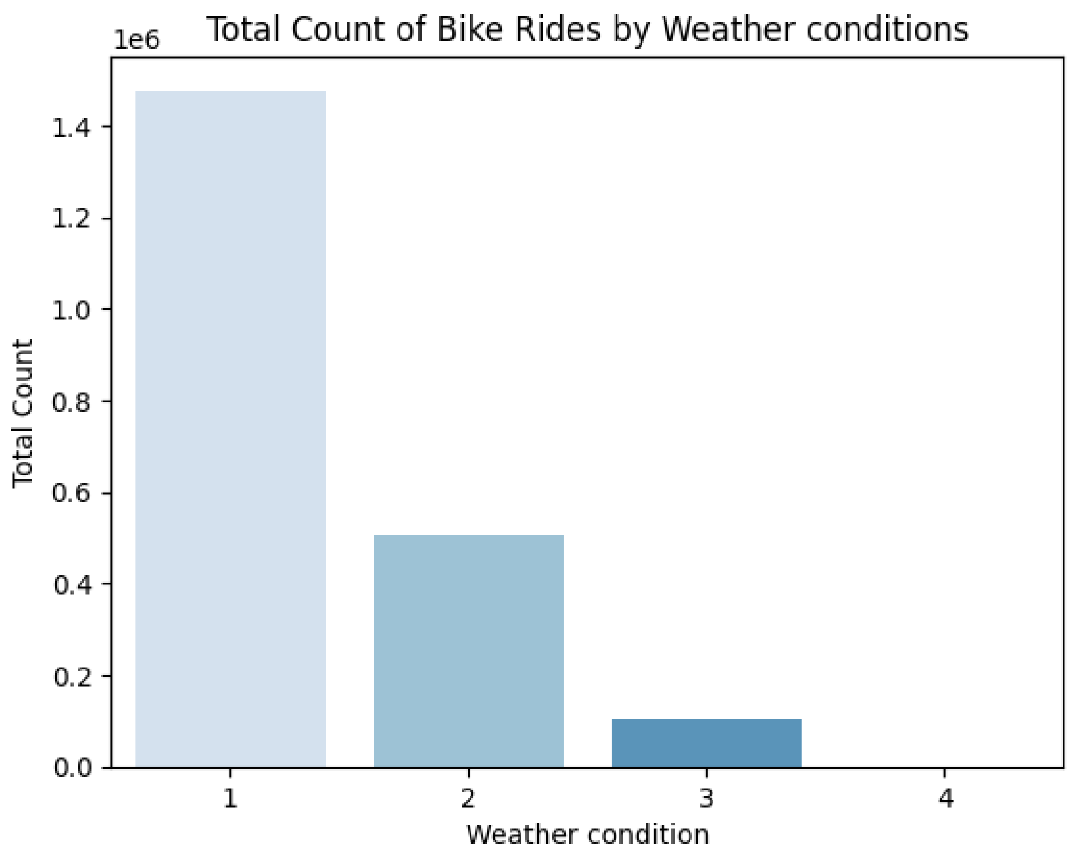
[62]:
```
for i, (group1, group2) in enumerate(pairs, start=1):
    data1 = label_map[group1]
    data2 = label_map[group2]
    print(f"Pair {i}: ({group1}, {group2})")

    # Perform the Mann-Whitney U Test
    statistic, p_value = mannwhitneyu(data1, data2, alternative='greater')
    print(f"Mann-Whitney U Test Statistic: {statistic}")
    print(f"P-value: {p_value}")

    # Check for significance based on p-value
    alpha = 0.05/3 #Bonferroni correction
    if p_value < alpha:
        print("Reject the null hypothesis. 1st group mean is significantly␣
↪greater than 2nd group mean.")
    else:
        print("Fail to reject the null hypothesis. 1st group mean is␣
↪significantly smaller or equal than 2nd group mean.")
    print()
```

```
Pair 1: (w1, w2)
Mann-Whitney U Test Statistic: 7932587.5
P-value: 7.604144323986111e-06
Reject the null hypothesis. 1st group mean is significantly greater than 2nd
group mean.

Pair 2: (w1, w3)
Mann-Whitney U Test Statistic: 3199906.0
P-value: 8.114092010805387e-45
Reject the null hypothesis. 1st group mean is significantly greater than 2nd
group mean.

Pair 3: (w2, w3)
Mann-Whitney U Test Statistic: 1854677.5
P-value: 1.7269516899113813e-29
Reject the null hypothesis. 1st group mean is significantly greater than 2nd
group mean.
```

5.2.1 We can use the non-parametric Mann-Whitney U test (also known as the Wilcoxon rank-sum test) instead of the t-test in this scenario because the population is extremely skewed.

5.3 We can conclude that each pair of weather conditions have significantly different number of average rides count.

[36]:
```python
g=df.groupby('weather',as_index=False)['count'].sum()
sns.barplot(data=g, x='weather', y='count',palette='Blues')
plt.xlabel('Weather condition')
plt.ylabel('Total Count')
plt.title('Total Count of Bike Rides by Weather conditions')
plt.show()
```



5.4 Weather condition 1 requires most number of bikes.

5.5 We can conclude $\mu_1 > \mu_2 > \mu_3$.

6 Part 5 : Is the demand of bicycles on rent same for different seasons?

[37]:
```
df.groupby('season')['count'].describe()
```

[37]:

season	count	mean	std	min	25%	50%	75%	max
1	2686.0	116.343261	125.273974	1.0	24.0	78.0	164.0	801.0
2	2733.0	215.251372	192.007843	1.0	49.0	172.0	321.0	873.0
3	2733.0	234.417124	197.151001	1.0	68.0	195.0	347.0	977.0
4	2734.0	198.988296	177.622409	1.0	51.0	161.0	294.0	948.0

[38]:
```
s1 = df[df['season'] == 1].sample(5000, random_state=95, replace=True)['count']
s2 = df[df['season'] == 2].sample(5000, random_state=95, replace=True)['count']
s3 = df[df['season'] == 3].sample(5000, random_state=95, replace=True)['count']
s4 = df[df['season'] == 4].sample(5000, random_state=95, replace=True)['count']
```

$H_0 : \mu_1 = \mu_2 = \mu_3 = \mu_4$

H_1: Average mean count of the four seasons are not equal.

- $\mu_1, \mu_2, \mu_3, \mu_4$ are the mean counts for seasons 1,2,3,4 respectively.

6.0.1 Normality and equal variance of each group check

[39]:
```
shapiro(df.sample(100,random_state=95,replace=True)['count'])
```

[39]:
```
ShapiroResult(statistic=0.8826003074645996, pvalue=2.34207959692867e-07)
```

[40]:
```
levene(s1,s2,s3,s4)
```

[40]:
```
LeveneResult(statistic=364.1057885155847, pvalue=2.9387905737363694e-230)
```

6.1 One-way ANOVA

[41]:
```
f_oneway(s1,s2,s3,s4)
```

[41]:
```
F_onewayResult(statistic=447.44708851025274, pvalue=2.1540340699147184e-281)
```

6.1.1 Clearly target variable is not normal and also the groups donot have equal variance so ANOVA results are not trustworthy.

6.1.2 Using (non-parametric) Kruskal Wallis test.

[42]:
```
kruskal(s1,s2,s3,s4)
```

[42]:
```
KruskalResult(statistic=1301.4766257573244, pvalue=7.037738268222606e-282)
```

6.2 Pvalue is well below .05 so we can reject null and conclude that average rides count is different for different seasons.

[43]:
```
label_map = {'s1': s1, 's2': s2, 's3': s3, 's4': s4}
pairs = [('s2', 's1'), ('s3', 's1'), ('s4', 's1'), ('s3', 's2'), ('s2', 's4'),␣
↪('s3', 's4')]
for i, (group1, group2) in enumerate(pairs, start=1):
    print(f"Pair {i}: ({group1}, {group2})")

    #Levene Test
    levene_statistic, levene_pvalue = levene(label_map[group1],␣
↪label_map[group2])
    print(f"Levene Test p-value: {levene_pvalue}")

    # Determine equality of variance based on Levene test result
    if levene_pvalue < 0.01:
        equal_var = False
    else:
        equal_var = True

    # Two Sample Independent T Test
    statistic, p_value = ttest_ind(label_map[group1], label_map[group2],␣
↪equal_var=equal_var, alternative='greater')
    print(f"2 Sample Independent T Test Statistic: {statistic}")
    print(f"P-value: {p_value}")

    # Check for significance based on p-value
    alpha = 0.05/6 #Bonferroni correction
    if p_value < alpha:
        print("Reject the null hypothesis. 1st group mean is significantly␣
↪greater than 2nd group mean.")
    else:
        print("Fail to reject the null hypothesis. 1st group mean is␣
↪significantly smaller or equal than 2nd group mean.")
    print()
```

```
Pair 1: (s2, s1)
Levene Test p-value: 4.198246319509843e-179
2 Sample Independent T Test Statistic: 31.10255307501512
```

```
P-value: 1.5179298418397769e-201
Reject the null hypothesis. 1st group mean is significantly greater than 2nd
group mean.

Pair 2: (s3, s1)
Levene Test p-value: 4.188823199758719e-198
2 Sample Independent T Test Statistic: 35.9781897991829
P-value: 5.263614629041002e-264
Reject the null hypothesis. 1st group mean is significantly greater than 2nd
group mean.

Pair 3: (s4, s1)
Levene Test p-value: 2.850918182746981e-116
2 Sample Independent T Test Statistic: 27.369599134572923
P-value: 9.91887836393212e-159
Reject the null hypothesis. 1st group mean is significantly greater than 2nd
group mean.

Pair 4: (s3, s2)
Levene Test p-value: 0.09438608500247757
2 Sample Independent T Test Statistic: 4.621756826665715
P-value: 1.926338439642033e-06
Reject the null hypothesis. 1st group mean is significantly greater than 2nd
group mean.

Pair 5: (s2, s4)
Levene Test p-value: 1.6021085189431552e-09
2 Sample Independent T Test Statistic: 4.559932886358439
P-value: 2.5891103837130133e-06
Reject the null hypothesis. 1st group mean is significantly greater than 2nd
group mean.

Pair 6: (s3, s4)
Levene Test p-value: 1.4404991539288338e-14
2 Sample Independent T Test Statistic: 9.292134084386216
P-value: 9.160077936247514e-21
Reject the null hypothesis. 1st group mean is significantly greater than 2nd
group mean.
```

[44]:

```
for i, (group1, group2) in enumerate(pairs, start=1):
    data1 = label_map[group1]
    data2 = label_map[group2]
    print(f"Pair {i}: ({group1}, {group2})")

    # Perform the Mann-Whitney U Test
    statistic, p_value = mannwhitneyu(data1, data2, alternative='greater')
```

```
    print(f"Mann-Whitney U Test Statistic: {statistic}")
    print(f"P-value: {p_value}")

    # Check for significance based on p-value
    alpha = 0.05/6 #Bonferroni correction
    if p_value < alpha:
        print("Reject the null hypothesis. 1st group mean is significantly␣
↪greater than 2nd group mean.")
    else:
        print("Fail to reject the null hypothesis. 1st group mean is␣
↪significantly smaller or equal than 2nd group mean.")
    print()
```

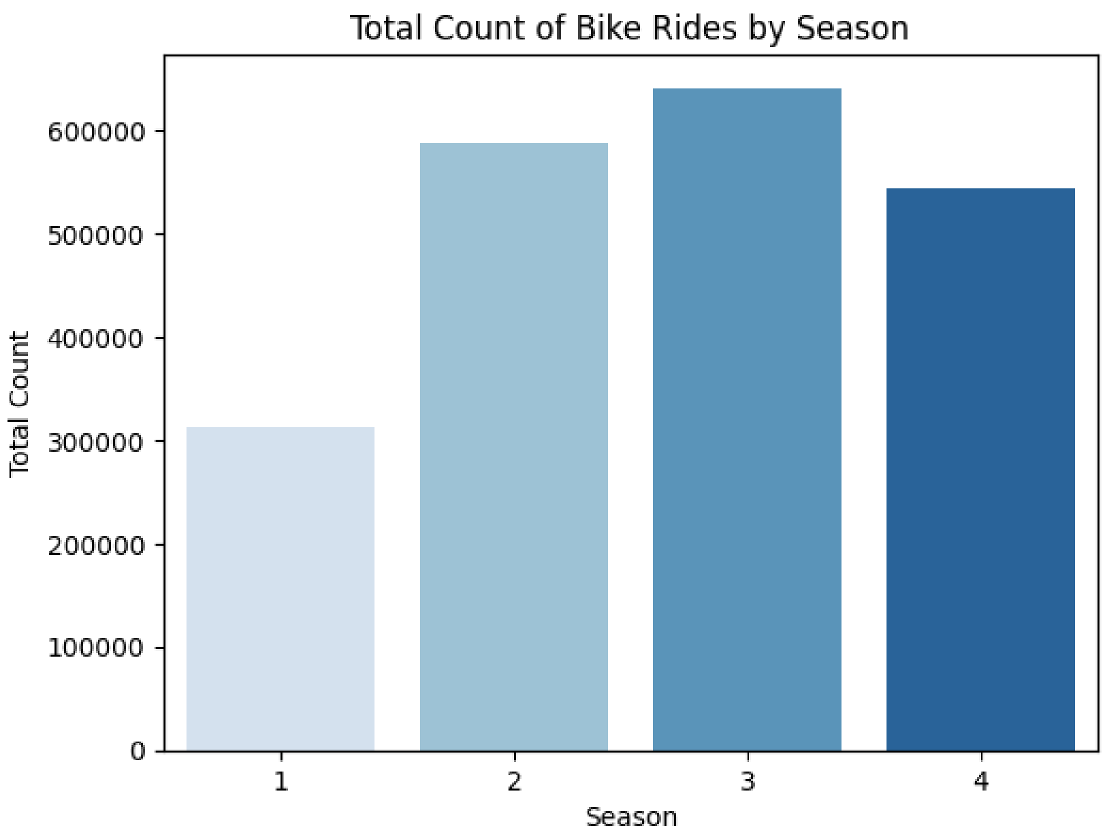
```
Pair 1: (s2, s1)
Mann-Whitney U Test Statistic: 16463120.0
P-value: 2.9092933783255415e-166
Reject the null hypothesis. 1st group mean is significantly greater than 2nd
group mean.

Pair 2: (s3, s1)
Mann-Whitney U Test Statistic: 17237437.0
P-value: 1.4881141164282895e-236
Reject the null hypothesis. 1st group mean is significantly greater than 2nd
group mean.

Pair 3: (s4, s1)
Mann-Whitney U Test Statistic: 16213158.5
P-value: 3.091805575709368e-146
Reject the null hypothesis. 1st group mean is significantly greater than 2nd
group mean.

Pair 4: (s3, s2)
Mann-Whitney U Test Statistic: 13236006.0
P-value: 1.7074041316865722e-07
Reject the null hypothesis. 1st group mean is significantly greater than 2nd
group mean.

Pair 5: (s2, s4)
Mann-Whitney U Test Statistic: 12971815.0
P-value: 0.000540227135499256
Reject the null hypothesis. 1st group mean is significantly greater than 2nd
group mean.

Pair 6: (s3, s4)
Mann-Whitney U Test Statistic: 13746451.0
P-value: 2.928616838532851e-18
Reject the null hypothesis. 1st group mean is significantly greater than 2nd
```

group mean.

- We can use the non-parametric Mann-Whitney U test (also known as the Wilcoxon rank-sum test) instead of the t-test in this scenario because the population is extremely skewed.

6.3 We can conclude that each pair of seasons have significantly different number of average rides count.

[45]:
```
g=df.groupby('season',as_index=False)['count'].sum()
sns.barplot(data=g, x='season', y='count',palette='Blues')
plt.xlabel('Season')
plt.ylabel('Total Count')
plt.title('Total Count of Bike Rides by Season')
plt.show()
```



6.4 Season 3 requires most number of bikes.

6.5 We can conclude $\mu_3 > \mu_2 > \mu_4 > \mu_1$.

7 Two-way ANOVA [Assumptions are not met, results might not be proper]

[46]:
```
test=ols('count ~ C(weather) * C(season)',data=df).fit()
sm.stats.anova_lm(test,typ=2)
```

[46]:

	sum_sq	df	F	PR(>F)
C(weather)	9.034656e+06	3.0	99.621868	1.337843e-43
C(season)	2.887549e+06	3.0	31.839954	1.630869e-14
C(weather):C(season)	8.382528e+05	9.0	3.081036	5.150817e-03
Residual	3.286889e+08	10873.0	NaN	NaN

7.1 Previous results are verified as we can conclude that weather and season have both main effect and interaction effect on count.

8 Part 6: Are weather conditions dependent on seasons?

H_0: weather and season are not dependent on each other.

H_1: weather and season are dependent on each other.

[47]:
```
cont=pd.crosstab(df['season'],df['weather'])
chi2_contingency(cont)
```

[47]:
```
Chi2ContingencyResult(statistic=49.158655596893624,
pvalue=1.549925073686492e-07, dof=9, expected_freq=array([[1.77454639e+03,
6.99258130e+02, 2.11948742e+02, 2.46738931e-01],
       [1.80559765e+03, 7.11493845e+02, 2.15657450e+02, 2.51056403e-01],
       [1.80559765e+03, 7.11493845e+02, 2.15657450e+02, 2.51056403e-01],
       [1.80625831e+03, 7.11754180e+02, 2.15736359e+02, 2.51148264e-01]]))
```

8.1 We can reject null as pvalue<.05 and conclude that weather and season are dependent on each other.

8.2 Are holidays dependent on seasons?

H_0: holiday and season are not dependent on each other.

H_1: holiday and season are dependent on each other.

[48]:
```
cont=pd.crosstab(df['season'],df['holiday'])
chi2_contingency(cont)
```

[48]:
```
Chi2ContingencyResult(statistic=20.82338817816167,
pvalue=0.00011455163312609901, dof=3, expected_freq=array([[2609.26419254,
76.73580746],
```

```
       [2654.92145875,   78.07854125],
       [2654.92145875,   78.07854125],
       [2655.89288995,   78.10711005]]))
```

8.3 We can reject null as pvalue<.05 and conclude that holiday and season are dependent on each other.

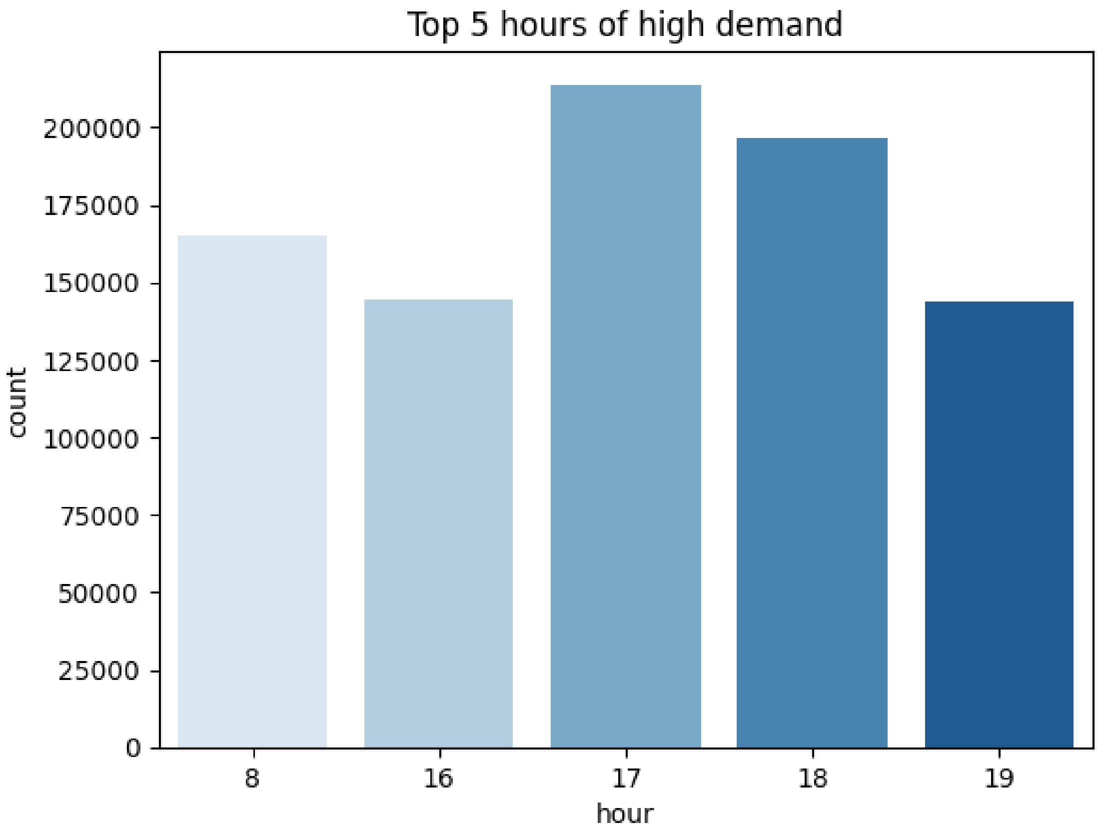
9 Hourly demand analysis

[49]:
```python
df['hour'] = pd.to_datetime(df['datetime']).dt.hour
g=df.groupby('hour', as_index=False)['count'].sum().
  ↪sort_values('count',ascending=False).head(5)
g
```

[49]:

	hour	count
17	17	213757
18	18	196472
8	8	165060
16	16	144266
19	19	143767

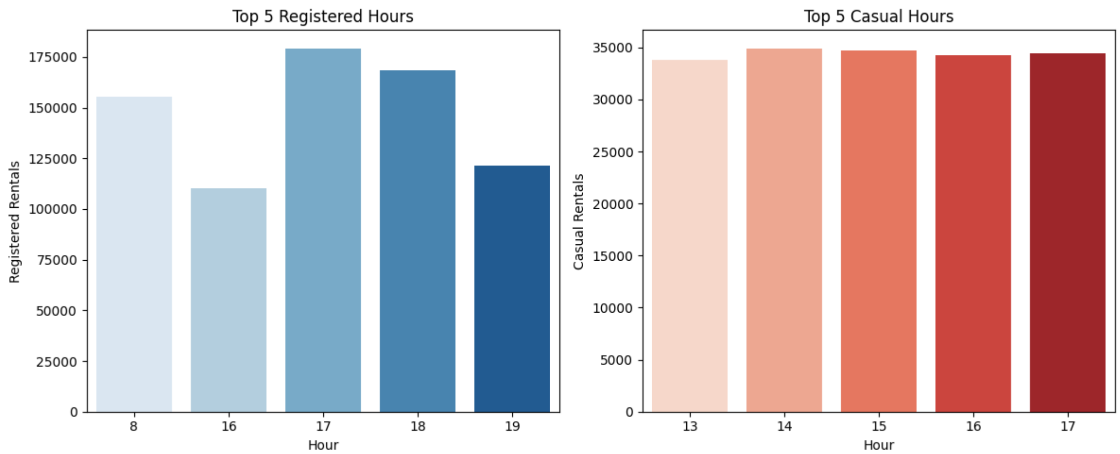
[50]:
```python
sns.barplot(data=g,x='hour',y='count',palette='Blues')
plt.title('Top 5 hours of high demand')
plt.show()
```



9.1 Need more bikes from 4pm-7pm and also in the morning around 8am.

[51]:
```
hourly_rentals = df.groupby('hour', as_index=False)[['casual', 'registered']].
    ↪sum()
top_registered_hours = hourly_rentals.sort_values(by='registered',␣
    ↪ascending=False).head(5)
top_casual_hours = hourly_rentals.sort_values(by='casual', ascending=False).
    ↪head(5)

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
sns.barplot(x='hour', y='registered', data=top_registered_hours,␣
    ↪palette='Blues')
plt.xlabel('Hour')
plt.ylabel('Registered Rentals')
plt.title('Top 5 Registered Hours')

plt.subplot(1, 2, 2)
sns.barplot(x='hour', y='casual', data=top_casual_hours, palette='Reds')
plt.xlabel('Hour')
```

```
plt.ylabel('Casual Rentals')
plt.title('Top 5 Casual Hours')

plt.tight_layout()
plt.show()
```



[52]:
```
print("Top 5 hours for registered rentals:")
print(top_registered_hours)

print("\nTop 5 hours for casual rentals:")
print(top_casual_hours)
```

```
Top 5 hours for registered rentals:
    hour  casual  registered
17    17   34401      179356
18    18   27997      168475
8      8    9802      155258
19    19   22378      121389
16    16   34238      110028

Top 5 hours for casual rentals:
    hour  casual  registered
14    14   34925       76085
15    15   34669       81291
17    17   34401      179356
16    16   34238      110028
13    13   33771       83780
```

9.2 The conclusion from the top 5 hours for registered and casual rentals suggests different patterns in bike rental behavior:

1. **Top 5 Hours for Registered Rentals:**

- The hours with the highest number of registered rentals are during typical commuting hours, particularly in the late afternoon (17:00 to 18:00) and early morning (08:00). This indicates that registered users, who are likely commuters or regular users, heavily utilize the bike-sharing service during their daily commute to and from work or school.

2. **Top 5 Hours for Casual Rentals:**
   - The hours with the highest number of casual rentals are during the afternoon (13:00 to 15:00), with a peak at 14:00. This suggests that casual users, who may be tourists or occasional riders, prefer renting bikes during the midday period, perhaps for leisurely activities or sightseeing.

9.3 We can get the number of casual and registered counts through regression by using independent variables such as atemp, humidity, windspeed and hour.

10 Part 7 : Regression

10.1 Casual count

[53]:
```
X=df[['temp', 'humidity', 'windspeed','hour']]
y=df['casual']
X = sm.add_constant(X)
modelcasual = sm.OLS(y, X).fit()
modelcasual.summary()
```

[53]:

Dep. Variable:	casual	**R-squared:**	0.344
Model:	OLS	**Adj. R-squared:**	0.344
Method:	Least Squares	**F-statistic:**	1427.
Date:	Sun, 14 Apr 2024	**Prob (F-statistic):**	0.00
Time:	20:04:31	**Log-Likelihood:**	-55728.
No. Observations:	10886	**AIC:**	1.115e+05
Df Residuals:	10881	**BIC:**	1.115e+05
Df Model:	4		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	13.0505	2.190	5.959	0.000	8.758	17.343
temp	2.7238	0.050	54.046	0.000	2.625	2.823
humidity	-0.7244	0.022	-32.987	0.000	-0.767	-0.681
windspeed	-0.0807	0.050	-1.604	0.109	-0.179	0.018
hour	1.1893	0.059	20.138	0.000	1.074	1.305

Omnibus:	6024.289	**Durbin-Watson:**	0.188
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	53789.368
Skew:	2.541	**Prob(JB):**	0.00
Kurtosis:	12.632	**Cond. No.**	392.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[54]:
```
residuals = modelcasual.resid
shapiro_test = shapiro(residuals)
print("Shapiro-Wilk test p-value:", shapiro_test.pvalue)
```

```
Shapiro-Wilk test p-value: 0.0
```

[55]:
```
name = ['Lagrange multiplier statistic', 'p-value', 'f-value', 'f p-value']
breusch_pagan_test = sm.stats.diagnostic.het_breuschpagan(modelcasual.resid,␣
↪modelcasual.model.exog)
print(dict(zip(name, breusch_pagan_test)))
```

```
{'Lagrange multiplier statistic': 678.2729617305428, 'p-value':
1.7642127735552764e-145, 'f-value': 180.75248458645208, 'f p-value':
3.371255953757685e-150}
```

[56]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif_data = pd.DataFrame()
vif_data["Feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(len(X.
↪columns))]
print(vif_data)
```

```
     Feature        VIF
0      const  31.877682
1       temp   1.024825
2   humidity   1.186919
3  windspeed   1.120297
4       hour   1.108654
```

10.2 Registered count

[57]:
```
X=df[['temp', 'humidity', 'windspeed','hour']]
y=df['registered']
X = sm.add_constant(X)
modelregistered = sm.OLS(y, X).fit()
modelregistered.summary()
```

[57]:

Dep. Variable:	registered	**R-squared:**	0.241
Model:	OLS	**Adj. R-squared:**	0.241
Method:	Least Squares	**F-statistic:**	863.8
Date:	Sun, 14 Apr 2024	**Prob (F-statistic):**	0.00
Time:	20:04:31	**Log-Likelihood:**	-68566.
No. Observations:	10886	**AIC:**	1.371e+05
Df Residuals:	10881	**BIC:**	1.372e+05
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P> |t|	[0.025	0.975]
const	57.9804	7.122	8.141	0.000	44.020	71.941
temp	5.1318	0.164	31.311	0.000	4.811	5.453
humidity	-1.3041	0.071	-18.263	0.000	-1.444	-1.164
windspeed	-0.0103	0.164	-0.063	0.950	-0.331	0.310
hour	6.4629	0.192	33.651	0.000	6.086	6.839

Omnibus:	3887.645	**Durbin-Watson:**	0.561
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	13648.828
Skew:	1.810	**Prob(JB):**	0.00
Kurtosis:	7.122	**Cond. No.**	392.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[58]:
```
residuals = modelregistered.resid
shapiro_test = shapiro(residuals)
print("Shapiro-Wilk test p-value:", shapiro_test.pvalue)
```

```
Shapiro-Wilk test p-value: 0.0
```

[59]:
```
name = ['Lagrange multiplier statistic', 'p-value', 'f-value', 'f p-value']
breusch_pagan_test = sm.stats.diagnostic.het_breuschpagan(modelregistered.
 ↪resid, modelcasual.model.exog)
print(dict(zip(name, breusch_pagan_test)))
```

```
{'Lagrange multiplier statistic': 169.6826061494763, 'p-value':
1.223450883130963e-35, 'f-value': 43.07254931092141, 'f p-value':
6.520222697852765e-36}
```

[60]:
```
vif_data = pd.DataFrame()
vif_data["Feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(len(X.
 ↪columns))]
print(vif_data)
```

```
     Feature        VIF
0      const  31.877682
1       temp   1.024825
2   humidity   1.186919
3  windspeed   1.120297
4       hour   1.108654
```

10.3 Conclusion

- We can utilize the model to predict bike counts based on the given independent variables by using model.predict(temp, humidity, windspeed, hour).
- While most assumptions linearity, independence, and no multicollinearity among indpenedent variables are met, the homoscedasticity and normality of residuals assumptions are violated.
- The low p-values of both models indicate that the models are good fits for the data.

- Only the p-value of windspeed is greater than .05 so we can neglect the effect of windspeed.
- Notably, the negative intercept for humidity suggests that an increase in humidity is associated with a decrease in bike counts.
- We have three options:
  - **a)** Proceed with the current model despite the violation of the assumptions.
  - **b)** Explore creating new combinations of independent variables like dropping windspeed and iterate to improve the model.
  - **c)** Explore other models like polynomial regression.

11 Insights

1. **Average Rides on Working Days vs. Non-working Days:**
   - The analysis indicates that the average count of bike rides on working days is significantly higher than on non-working days.
   - This suggests that more bike rides occur during regular working days compared to non-working days, possibly due to commuters using bike services for daily transportation to work or school.
2. **Average Rides on Holidays vs. Regular Days:**
   - The analysis indicates that the average count of bike rides on regular days is significantly higher than on holidays.
   - Also could be due to the same reason that regular days need more transportation.
3. **Average Rides Across Weather Conditions:**
   - Each pair of weather conditions significantly affects the average number of bike rides.
   - Weather condition 1 requires most bikes.
   - This implies that weather conditions play a crucial role in determining bike ride usage, with certain weather conditions likely leading to higher or lower ride counts for e.g. humid weather is not favored by users.
   - Check whether there is an error in collecting data as there is only one record of Weather condition 4.
4. **Average Rides Across Seasons:**
   - The analysis reveals that different seasons have a significant impact on the average number of bike rides.
   - Season 3 requires most number of bikes.
   - This suggests that seasonal variations influence bike ride usage patterns, with factors such as temperature, daylight hours, and seasonal activities affecting ride counts.
5. **Peak Demand Hours for Registered Rentals:**
   - Registered rentals peak during commuting hours, notably between 17:00 and 18:00, indicating heavy usage by commuters returning home from work or school.
   - Another significant peak occurs in the morning around 08:00, suggesting high demand during the morning commute hours.
6. **Peak Demand Hours for Casual Rentals:**
   - Casual rentals show a different pattern, with peak hours occurring in the afternoon, particularly between 13:00 and 15:00, with a notable peak at 14:00.
   - This trend indicates that casual users, likely tourists or occasional riders, prefer renting bikes during midday hours, possibly for leisure activities or sightseeing.

12 Recommendations

1. **Optimize Service Capacity :**
    - Allocate additional resources and bikes during weekdays and regular days, especially during peak commuting hours in the morning and evening, to meet the high demand from registered users.
    - Ensure sufficient bike availability at popular commuting locations such as offices, schools, and transportation hubs during peak hours.
2. **Promotional Strategies for Holidays:**
    - Implement targeted marketing campaigns and promotions to encourage bike usage on holidays and weekends, leveraging incentives such as discounted fares or special offers.
    - Partner with local businesses and event organizers to promote bike-sharing as a convenient and eco-friendly transportation option for holiday activities and events.
3. **Weather-Responsive Service Planning:**
    - Implement the model which dynamically predicts the count of users based on weather conditions and hour of day.
    - Prepare more bikes for weather condition 1.
    - Introduce weather-dependent promotions or discounts to encourage bike rides during favorable conditions and counteract the effects of adverse weather, such as high humidity, on ride demand.
    - Leverage warmer weather conditions to attract more casual riders to the service.
4. **Seasonal Promotions and Events:**
    - More bikes should be available for season 3.
    - Design seasonal promotions or events tailored to specific weather conditions and seasonal activities to attract riders during off-peak seasons.
    - Collaborate with local tourism boards, event organizers, and community organizations to promote bike-sharing as a recreational and leisure activity during peak tourist seasons.
5. **Recommendation for Bike Supply:**
    - To meet increased demand, additional bikes should be available during peak hours, especially from 16:00 to 19:00 in the evening and around 08:00 in the morning.
    - Use the models to predict the counts and plan accordingly.
6. **Enhanced User Experience and Accessibility:**
    - Improve bike-sharing infrastructure and accessibility by expanding docking stations and bike lanes in high-demand areas.
    - Invest in user-friendly mobile apps and digital platforms to streamline the rental process, provide real-time updates on bike availability, and enhance the overall user experience for both registered and casual riders.

[ ]:

```
```