YuluCasestudy

July 17, 2024

1 Business Case: Yulu - Hypothesis Testing

About Yulu:

Yulu, India's pioneering micro-mobility service provider, has embarked on a mission to revolutionize daily commutes by offering unique, sustainable transportation solutions However, recent revenue setbacks have prompted Yulu to seek the expertise of a consulting company to delve into the factors influencing the demand for their shared electric cycles specifically in the Ind amarket

Problem Statement:

The Compny wants to know - which variables are significant in predicting the demand for shared electric cycle in Indian market and - how well those variables describe the electric cycle demand

[1]:
```
#importing basic libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
#reading dataset into dataframe

df=pd.read_csv('https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/
↪001/428/original/bike_sharing.csv?1642089089')
df.head()
```

[2]:
```
              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
```

[3]:
```
#checking structure and characterestics of data

df.shape
```

[3]: (10886, 12)

[4]:
```
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
```

[5]:
```
df.duplicated().any()
```

[5]: False

- The dataset has 10886 rows and 12 columns (7 continuous columns and 4 categorical columns and 1 datetime columnwhich here used mostly as categorical)
- No Null/missing values.
- No duplicated data.

[6]:
```
#since the season,holiday,workingday,weather are all categorical variables and␣
↪date has to be in datetime format,converting datatypes according

df['datetime'] = pd.to_datetime(df['datetime'])
for col in ['season','holiday','workingday','weather']:
    df[col]=df[col].astype('str')
```

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    10886 non-null  datetime64[ns]
 1   season      10886 non-null  object
 2   holiday     10886 non-null  object
 3   workingday  10886 non-null  object
 4   weather     10886 non-null  object
 5   temp        10886 non-null  float64
 6   atemp       10886 non-null  float64
 7   humidity    10886 non-null  int64
 8   windspeed   10886 non-null  float64
 9   casual      10886 non-null  int64
 10  registered  10886 non-null  int64
 11  count       10886 non-null  int64
dtypes: datetime64[ns](1), float64(3), int64(4), object(4)
memory usage: 1020.7+ KB
```

[7]:
```python
df.describe(include=['object'])
```

[7]:
```
       season holiday workingday weather
count   10886   10886      10886   10886
unique      4       2          2       4
top         4       0          1       1
freq     2734   10575       7412    7192
```

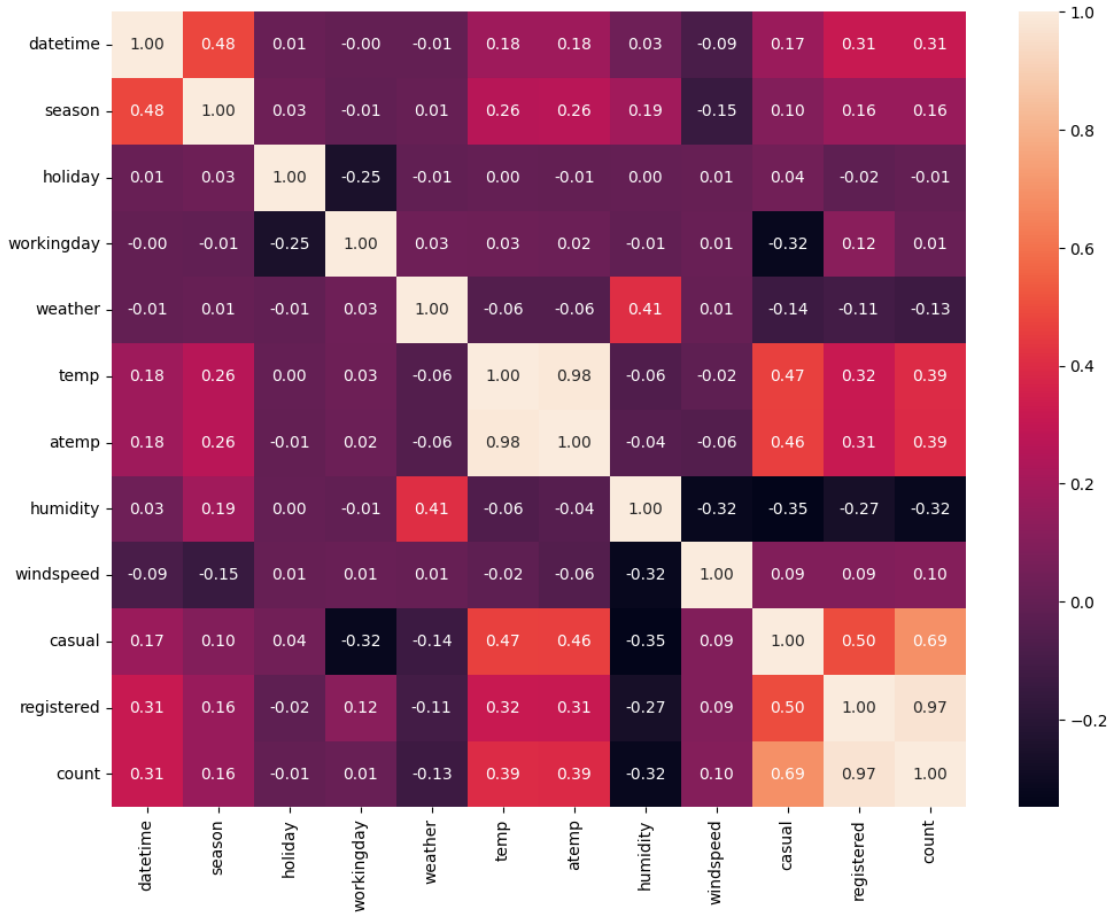
[8]:
```python
df.describe(include=['int64','float64'])
```

[8]:
```
              temp         atemp      humidity     windspeed        casual  \
count  10886.00000  10886.000000  10886.000000  10886.000000  10886.000000
mean      20.23086     23.655084     61.886460     12.799395     36.021955
std        7.79159      8.474601     19.245033      8.164537     49.960477
min        0.82000      0.760000      0.000000      0.000000      0.000000
25%       13.94000     16.665000     47.000000      7.001500      4.000000
50%       20.50000     24.240000     62.000000     12.998000     17.000000
75%       26.24000     31.060000     77.000000     16.997900     49.000000
max       41.00000     45.455000    100.000000     56.996900    367.000000

         registered         count
count  10886.000000  10886.000000
mean     155.552177    191.574132
std      151.039033    181.144454
min        0.000000      1.000000
```

```
25%       36.000000     42.000000
50%      118.000000    145.000000
75%      222.000000    284.000000
max      886.000000    977.000000
```

- clear weather(weather 1) recorded more rental bike sales compartitive to other weathers.
- More rental bikes are recorded on Non holidays.
- Registered users are more
- casual users are very less

[9]:
```
#checking how the correlation is between one factor to other

plt.figure(figsize=(12,9))
sns.heatmap(df.corr(), fmt='.2f',annot=True, cbar=True )
plt.show()
```



	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
datetime	1.00	0.48	0.01	-0.00	-0.01	0.18	0.18	0.03	-0.09	0.17	0.31	0.31
season	0.48	1.00	0.03	-0.01	0.01	0.26	0.26	0.19	-0.15	0.10	0.16	0.16
holiday	0.01	0.03	1.00	-0.25	-0.01	0.00	-0.01	0.00	0.01	0.04	-0.02	-0.01
workingday	-0.00	-0.01	-0.25	1.00	0.03	0.03	0.02	-0.01	0.01	-0.32	0.12	0.01
weather	-0.01	0.01	-0.01	0.03	1.00	-0.06	-0.06	0.41	0.01	-0.14	-0.11	-0.13
temp	0.18	0.26	0.00	0.03	-0.06	1.00	0.98	-0.06	-0.02	0.47	0.32	0.39
atemp	0.18	0.26	-0.01	0.02	-0.06	0.98	1.00	-0.04	-0.06	0.46	0.31	0.39
humidity	0.03	0.19	0.00	-0.01	0.41	-0.06	-0.04	1.00	-0.32	-0.35	-0.27	-0.32
windspeed	-0.09	-0.15	0.01	0.01	0.01	-0.02	-0.06	-0.32	1.00	0.09	0.09	0.10
casual	0.17	0.10	0.04	-0.32	-0.14	0.47	0.46	-0.35	0.09	1.00	0.50	0.69
registered	0.31	0.16	-0.02	0.12	-0.11	0.32	0.31	-0.27	0.09	0.50	1.00	0.97
count	0.31	0.16	-0.01	0.01	-0.13	0.39	0.39	-0.32	0.10	0.69	0.97	1.00

- casual users are less in workingdays
- when the humidity is high, sales are low
- when the temperature is high,sales are comparitively high

[10]:

```
# Data Mapping

season_mapping = {'1':'spring', '2':'summer', '3':'fall', '4':'winter'}
df['season'] = df['season'].map(lambda x: season_mapping[x])

holiday_mapping = {'0':'no', '1':'yes'}
df['holiday'] = df['holiday'].map(lambda x: holiday_mapping[x])

working_day_mapping = {'0':'no', '1':'yes'}
df['workingday'] = df['workingday'].map(lambda x: working_day_mapping[x])

weather_mapping = {'1':'clear', '2':'partly_clear', '3':'rain', '4':'intense'}
df['weather'] = df['weather'].map(lambda x: weather_mapping[x])
```

[11]:

```
#For Date time analysis

df['date']=df['datetime'].dt.date
df['Month']=df['datetime'].dt.month
df['Year']=df['datetime'].dt.year
df['Day']=df['datetime'].dt.day
df['Dayoftheweek']=df['datetime'].dt.day_name()
df.head()
```

[11]:

```
             datetime  season holiday workingday weather  temp   atemp  \
0 2011-01-01 00:00:00  spring      no         no   clear  9.84  14.395
1 2011-01-01 01:00:00  spring      no         no   clear  9.02  13.635
2 2011-01-01 02:00:00  spring      no         no   clear  9.02  13.635
3 2011-01-01 03:00:00  spring      no         no   clear  9.84  14.395
4 2011-01-01 04:00:00  spring      no         no   clear  9.84  14.395

   humidity  windspeed  casual  registered  count        date  Month  Year  \
0        81        0.0       3          13     16  2011-01-01      1  2011
1        80        0.0       8          32     40  2011-01-01      1  2011
2        80        0.0       5          27     32  2011-01-01      1  2011
3        75        0.0       3          10     13  2011-01-01      1  2011
4        75        0.0       0           1      1  2011-01-01      1  2011

   Day Dayoftheweek
0    1     Saturday
1    1     Saturday
2    1     Saturday
3    1     Saturday
4    1     Saturday
```

[12]:

```
df.groupby('Year')['count'].mean().sort_values(ascending=False)
```

[12]: Year
```
2012    238.560944
2011    144.223349
Name: count, dtype: float64
```

[13]:
```
df.groupby('Month')['count'].mean().sort_values(ascending=False)
```

[13]: Month
```
6     242.031798
7     235.325658
8     234.118421
9     233.805281
10    227.699232
5     219.459430
11    193.677278
4     184.160616
12    175.614035
3     148.169811
2     110.003330
1      90.366516
Name: count, dtype: float64
```

[14]:
```
df.groupby('Day')['count'].mean().sort_values(ascending=False)
```

[14]: Day
```
17    205.660870
15    201.527875
14    195.829268
4     195.705575
11    195.679577
10    195.183566
3     194.696335
13    194.160279
18    192.605684
19    192.311847
16    191.353659
12    190.675393
6     189.860140
5     189.765217
9     187.897391
2     183.910995
7     183.773519
1     180.333913
8     179.041812
Name: count, dtype: float64
```

[15]:
```
df.groupby('Dayoftheweek')['count'].mean().sort_values(ascending=False)
```

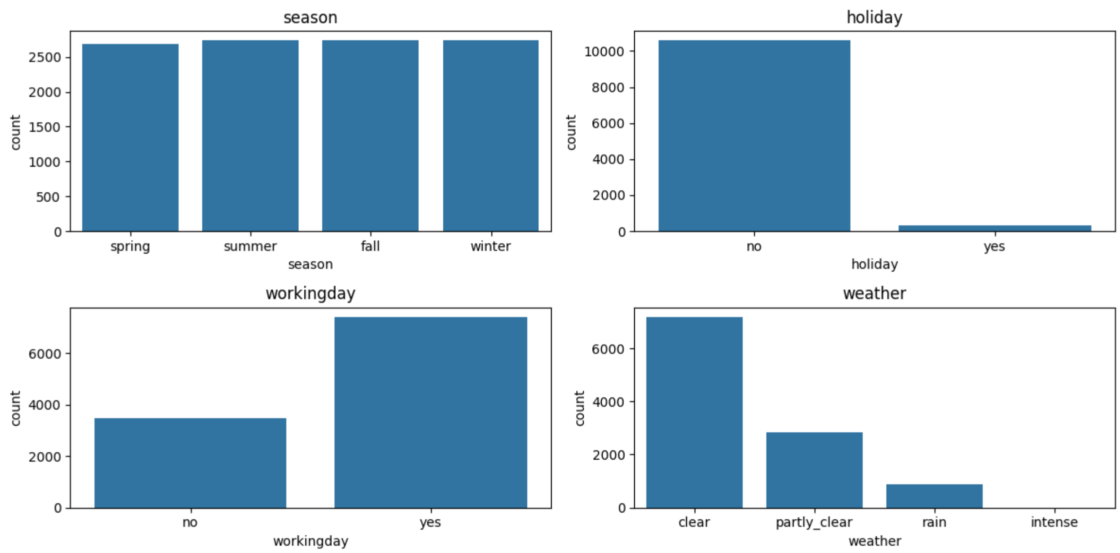
[15]:
```
Dayoftheweek
Friday       197.844343
Thursday     197.296201
Saturday     196.665404
Monday       190.390716
Tuesday      189.723847
Wednesday    188.411348
Sunday       180.839772
Name: count, dtype: float64
```

[16]:
```
df.groupby('date')[['Dayoftheweek','count']].aggregate({'Dayoftheweek':
↪'first','count':'mean'}).sort_values(by='count',ascending=False)
```

[16]:
```
           Dayoftheweek       count
date
2012-09-15     Saturday  363.083333
2012-05-19     Saturday  345.583333
2012-09-09       Sunday  342.791667
2012-10-05       Friday  339.833333
2012-06-02     Saturday  338.333333
…                 …           …
2011-01-09       Sunday   34.250000
2011-04-16     Saturday   33.125000
2011-12-07    Wednesday   29.375000
2011-03-10     Thursday   28.318182
2011-03-06       Sunday   26.304348

[456 rows x 2 columns]
```

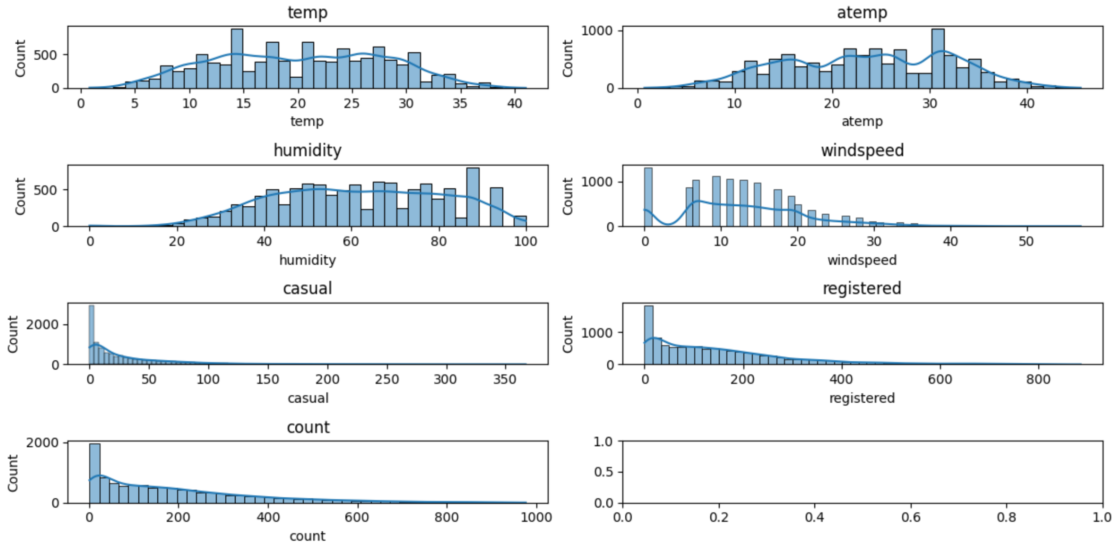
- 2012 has more sales than it's previous year.
- Almost all the weekdays and weekends recorded same count of rental bikes while the months starting from May to October recorded comparitively high rental bikes

[17]:
```
#UNIVARIATE ANALYSIS
#i)CATEGORICAL COLUMNS
fig,axes = plt.subplots(nrows=2,ncols=2,figsize=(12,6))

axes = axes.flatten()
categories = ['season','holiday','workingday','weather']

for i,category in enumerate(categories):
    sns.countplot(data=df,x=category,ax=axes[i])
    axes[i].set_title(category)

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

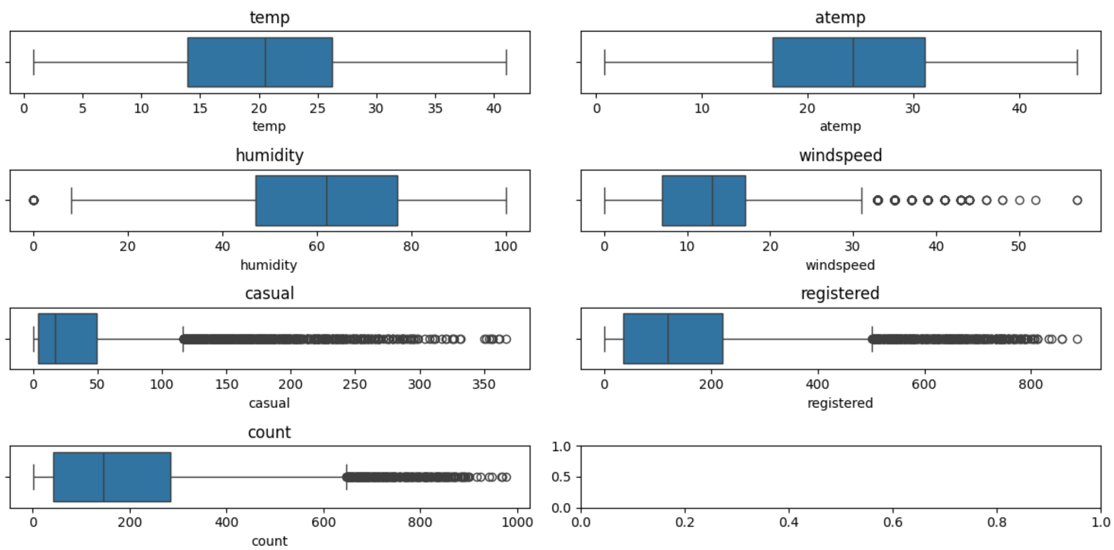


- The distribution of all seasons are similar
- on Non holidays, working days, more rental bike sales distribution is recorded.
- clear weather recorded more sales

[18]:
```
#2.Continuous columns
fig,axes = plt.subplots(nrows=4,ncols=2,figsize=(12,6))

axes = axes.flatten()
numeric_var = df.select_dtypes(include=['int64','float64'])

for i,category in enumerate(numeric_var):
    sns.histplot(data=df,x=category,kde=True,ax=axes[i])
    axes[i].set_title(category)

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

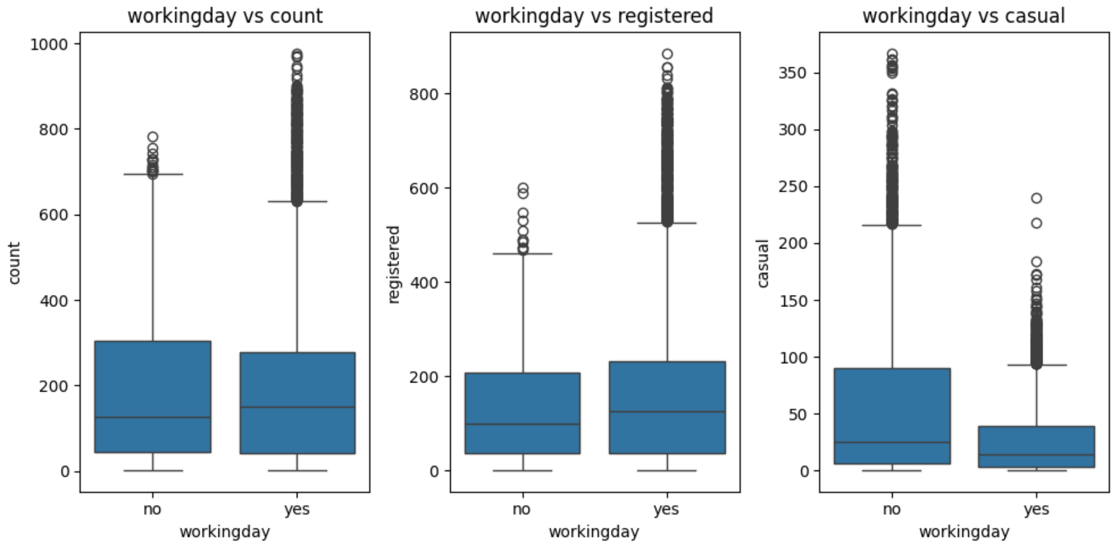


[19]:
```
#checking for outliers

fig,axes = plt.subplots(nrows=4,ncols=2,figsize=(12,6))

axes = axes.flatten()
numeric_var = df.select_dtypes(include=['int64','float64'])

for i,category in enumerate(numeric_var):
    sns.boxplot(data=df,x=category,ax=axes[i])
    axes[i].set_title(category)

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

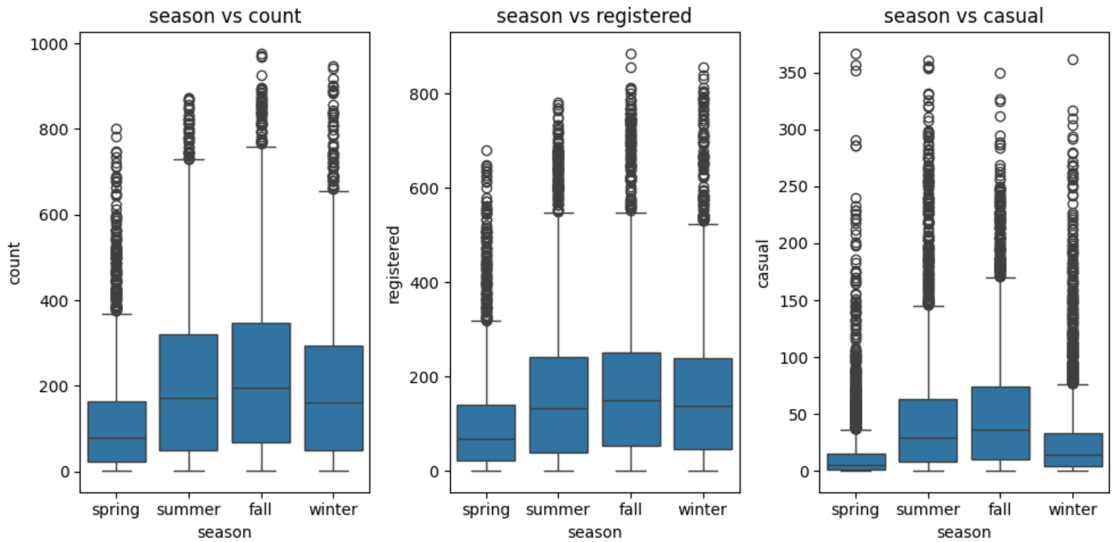


casual,registered,count has huge outliers

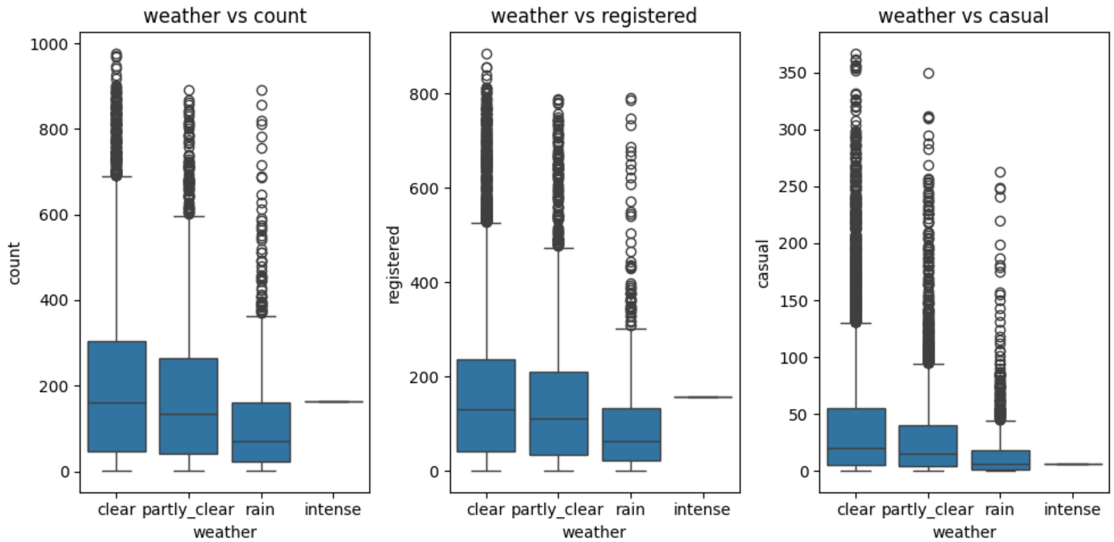
[20]:
```python
#THE BIVARIATE ANALYSIS OF IMPORTANT VARIABLES

plt.figure(figsize=(10,5))

plt.subplot(1,3,1)
sns.boxplot(data=df,x='workingday',y='count')
plt.title('workingday vs count')

plt.subplot(1,3,2)
sns.boxplot(data=df,x='workingday',y='registered')
plt.title('workingday vs registered')

plt.subplot(1,3,3)
sns.boxplot(data=df,x='workingday',y='casual')
plt.title('workingday vs casual')

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



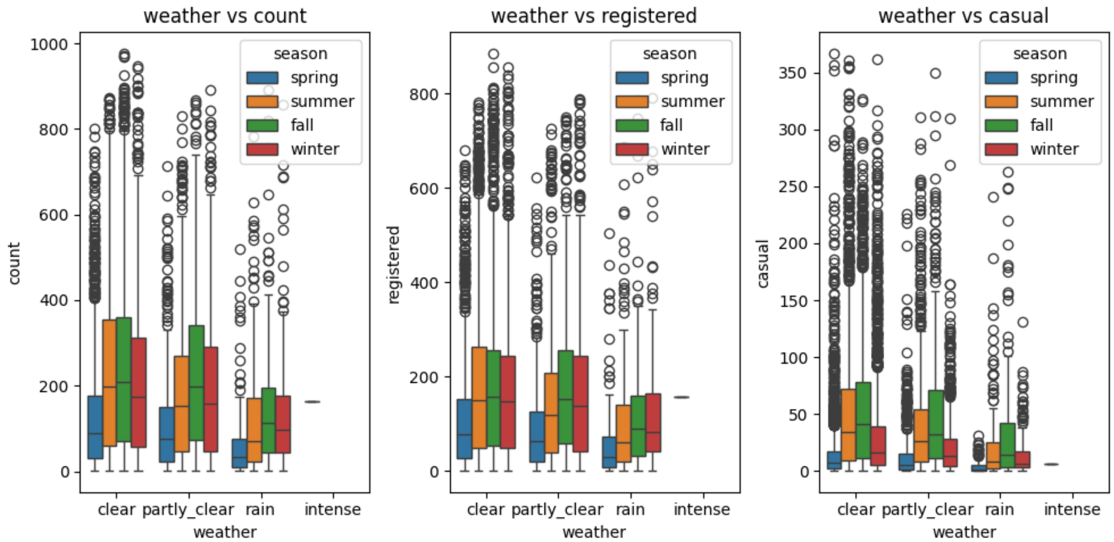
[21]:
```
plt.figure(figsize=(10,5))

plt.subplot(1,3,1)
sns.boxplot(data=df,x='season',y='count')
plt.title('season vs count')

plt.subplot(1,3,2)
sns.boxplot(data=df,x='season',y='registered')
plt.title('season vs registered')

plt.subplot(1,3,3)
sns.boxplot(data=df,x='season',y='casual')
plt.title('season vs casual')

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



[22]:
```
plt.figure(figsize=(10,5))

plt.subplot(1,3,1)
sns.boxplot(data=df,x='weather',y='count')
plt.title('weather vs count')

plt.subplot(1,3,2)
sns.boxplot(data=df,x='weather',y='registered')
plt.title('weather vs registered')

plt.subplot(1,3,3)
sns.boxplot(data=df,x='weather',y='casual')
plt.title('weather vs casual')

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



[23]:
```
plt.figure(figsize=(10,5))

plt.subplot(1,3,1)
sns.boxplot(data=df,x='weather',y='count',hue='season')
plt.title('weather vs count')

plt.subplot(1,3,2)
sns.boxplot(data=df,x='weather',y='registered',hue='season')
plt.title('weather vs registered')

plt.subplot(1,3,3)
sns.boxplot(data=df,x='weather',y='casual',hue='season')
plt.title('weather vs casual')

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



- More rental bike sales are in clear weather.
- More Casual sales are in fall season

1.0.1 Check if there is any significant difference between the no. of bike rides on Weekdays and Weekends?

Setting up Null and Alternate Hypothesis:

H0 : No significant difference between no.of bike rides on weekdays and weekends

Ha : There is significant difference between no.of bike rides on weekdays and weekends

- **Significance level : 0.05**

[24]:
```
df.groupby('workingday')['count'].mean()
```

[24]:
```
workingday
no     188.506621
yes    193.011873
Name: count, dtype: float64
```

- Using **2 sample Independent ttest** to check whether the mean of these two samples or values of these samples is significantly different

[25]:
```
#calculate the p-value:

import scipy.stats as stats
tstats,pval = stats.
 ↪ttest_ind(df[df['workingday']=='no']['count'],df[df['workingday']=='yes']['count'])
print("pvalue: ",pval)
```

```
pvalue:  0.22644804226361348
```

pvalue > 0.05, hence *fail to reject null hypothesis.*

No significant difference between bike rides on working and non working days or there is equal trend of bikes rented in both working and Non working days

1.0.2 Check if there is any significant difference between the no. of bike rides on holidays and non holidays?

Setting up Null and Alternate Hypothesis:

H0 : No significant difference between no.of bike rides on holidays and Non holidays

Ha : There is significant difference between no.of bike rides on holidays and Non holidays

- **Significance level : 0.05**

[26]:
```
df.groupby('holiday')['count'].mean()
```

[26]:
```
holiday
no     191.741655
yes    185.877814
Name: count, dtype: float64
```

- Using **2 sample Independent ttest** to check whether the mean of these two samples or values of these samples is significantly different

[27]:
```
#calculate the p-value:

import scipy.stats as stats
tstats,pval = stats.
 ↪ttest_ind(df[df['holiday']=='no']['count'],df[df['holiday']=='yes']['count'])
print("pvalue: ",pval)
```

```
pvalue:  0.5736923883271103
```

pvalue > 0.05, hence *fail to reject null hypothesis.*

No significant difference between bike rides on holidays and non holidays or there is equal trend of bikes rented in both holidays and Non holidays

1.0.3 Check if the demand of bicycles on rent is the same for different Weather conditions?

Setting up Null and Alternate Hypothesis:

H0 : No significant difference between no.of bike rides in different weather conditions

Ha : There is significant difference between no.of bike rides in different weather conditions

- **Significance level : 0.05**

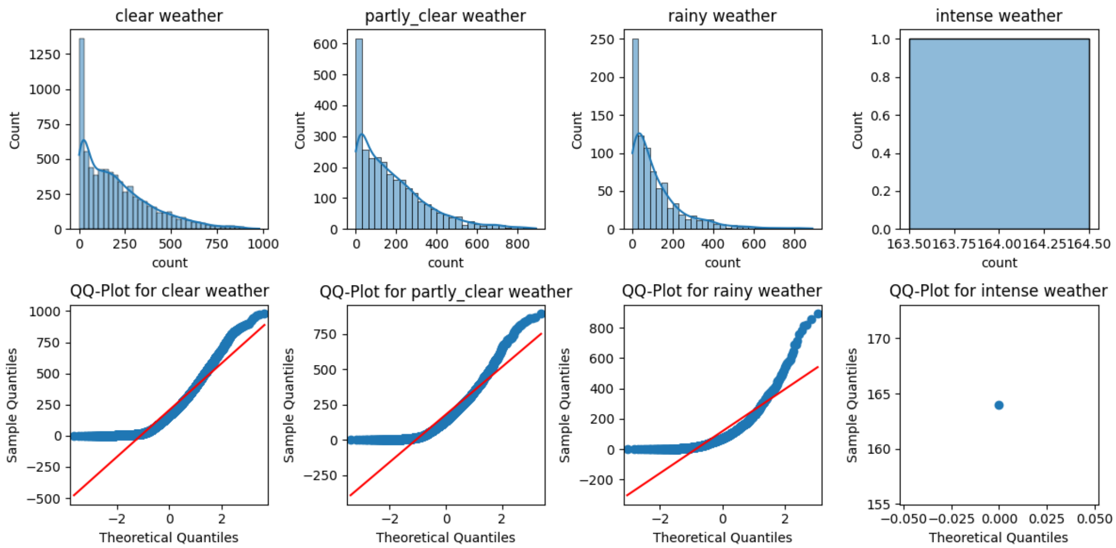
[28]:
```
df.groupby('weather')['count'].mean()
```

[28]:
```
weather
clear           205.236791
intense         164.000000
partly_clear    178.955540
rain            118.846333
Name: count, dtype: float64
```

[29]:
```python
#Setting up of data

w1 = df[df['weather']=='clear']['count']
w2 = df[df['weather']=='partly_clear']['count']
w3 = df[df['weather']=='rain']['count']
w4 = df[df['weather']=='intense']['count']
```

[30]:
```python
#Assessing Normality

#1. Graphical Methods

fig,axes = plt.subplots(nrows=2,ncols=4,figsize=(12,6))
axes = axes.flatten()

sns.histplot(w1,kde=True,ax=axes[0])
axes[0].set_title('clear weather')

sns.histplot(w2,kde=True,ax=axes[1])
axes[1].set_title('partly_clear weather')

sns.histplot(w3,kde=True,ax=axes[2])
axes[2].set_title('rainy weather')

sns.histplot(w4,kde=True,ax=axes[3])
axes[3].set_title('intense weather')

from statsmodels.api import qqplot
qqplot(w1,line='s',ax=axes[4])
axes[4].set_title('QQ-Plot for clear weather')

qqplot(w2,line='s',ax=axes[5])
axes[5].set_title('QQ-Plot for partly_clear weather')

qqplot(w3,line='s',ax=axes[6])
axes[6].set_title('QQ-Plot for rainy weather')

qqplot(w4,line='s',ax=axes[7])
axes[7].set_title('QQ-Plot for intense weather')

plt.tight_layout()
```

```
plt.show()
```



The density curves are not symmetric(skewed) and qqplot doesn't follow the line which clearly shows the distribution is not normal distribution.

[31]:

```
# 2. Statistical tests
#H0 : The data follows normal distribution ,Ha: the data doesn't follow normal distribution

from scipy.stats import shapiro
test_stat1,p_val1 = shapiro(w1)
test_stat2,p_val2 = shapiro(w2)
test_stat2,p_val3 = shapiro(w3)
#test_stat2,p_val4 = shapiro(w4)

print("p_value of clear weather is ",p_val1)
print("p_value of partly_clear weather is ",p_val2)
print("p_value of rainy weather is ",p_val3)
#print("p_value of weather 4 is ",p_val4)
```

```
p_value of clear weather is  0.0
p_value of partly_clear weather is  9.781063280987223e-43
p_value of rainy weather is  3.876090133422781e-33
```

all pvalues are clearly < 0.05, rejecting null hypothesis. The data doesn't follow normal distribution

[32]:

```
# 3. Summary Statistics:
# i) skewness
print("skewness of clear weather data is ",w1.skew())
print("skewness of partly_clear weather data is ",w2.skew())
```

```
print("skewness of rainy weather data is ",w3.skew())
#print("skewness of intense weather data is ",w4.skew())

#ii)kurtosis
print("kurtosis of clear weather data is ",w1.kurt())
print("kurtosis of partly_clear weather data is ",w2.kurt())
print("kurtosis of rainy weather data is ",w3.kurt())
#print("kurtosis of intense weather data is ",w4.kurt())
```

```
skewness of clear weather data is  1.1398572666918205
skewness of partly_clear weather data is  1.294444423357868
skewness of rainy weather data is  2.1871371080456594
kurtosis of clear weather data is  0.964719852310354
kurtosis of partly_clear weather data is  1.5884304891319174
kurtosis of rainy weather data is  6.003053730759276
```

The data in all the weather types are positively skewed(>0) and have both flatten and peak kurtosis

[33]:
```
#equal variance
#H0: The data groups are equally variant, Ha: The data groups are not equally variant
from scipy.stats import levene
levene(w1,w2,w3)
```

[33]:
```
LeveneResult(statistic=81.67574924435011, pvalue=6.198278710731511e-36)
```

pvalue<0.05, rejecting null hypothesis. Those are not equally variant

since all the assumptions are false, we can use kruskals as per theory

[34]:
```
from scipy.stats import kruskal
kruskal(w1,w2,w3,w4)
```

[34]:
```
KruskalResult(statistic=205.00216514479087, pvalue=3.501611300708679e-44)
```

But as per the test asked to do in the problem given, **performing one way anova testing**

[35]:
```
import scipy.stats as stats
stats.f_oneway(w1,w2,w3,w4)
```

[35]:
```
F_onewayResult(statistic=65.53024112793271, pvalue=5.482069475935669e-42)
```

pvalue < 0.05, hence *rejecting null hypothesis.*

The demand of bicycles on rent has impact in different weather conditions

1.0.4 Check if the demand of bicycles on rent is the same for different Seasons?

Setting up Null and Alternate Hypothesis:

H0 : No significant difference between no.of bike rides in different seasons

Ha : There is significant difference between no.of bike rides in different seasons

- **Significance level : 0.05**

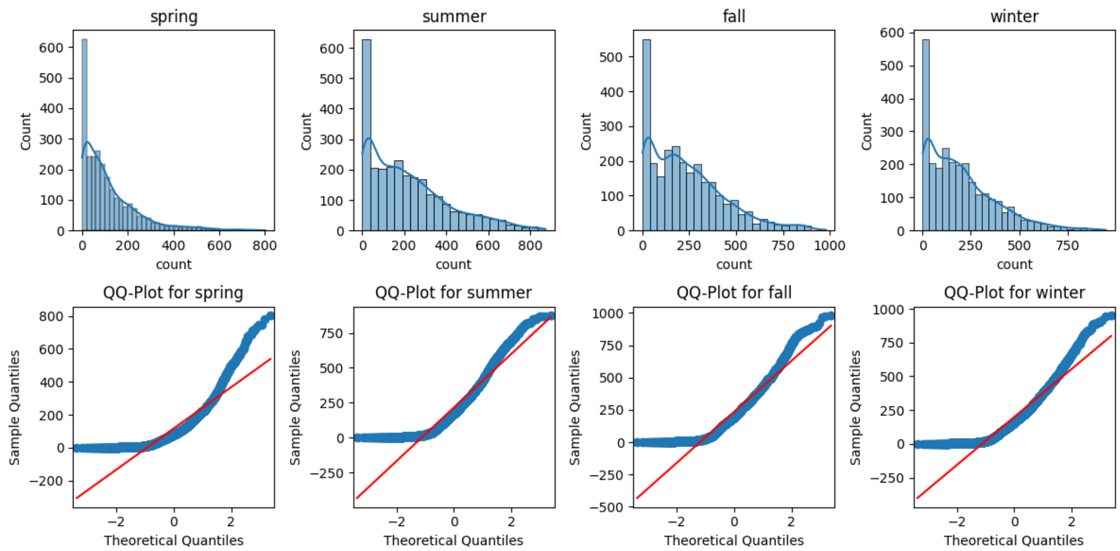
[36]:
```
df.groupby('season')['count'].mean()
```

[36]:
```
season
fall      234.417124
spring    116.343261
summer    215.251372
winter    198.988296
Name: count, dtype: float64
```

[37]:
```
#Setting up of data

s1 = df[df['season']=='spring']['count']
s2 = df[df['season']=='summer']['count']
s3 = df[df['season']=='fall']['count']
s4 = df[df['season']=='winter']['count']
```

[38]:
```
#Assessing Normality

#1. Graphical Methods

fig,axes = plt.subplots(nrows=2,ncols=4,figsize=(12,6))
axes = axes.flatten()

sns.histplot(s1,kde=True,ax=axes[0])
axes[0].set_title('spring')

sns.histplot(s2,kde=True,ax=axes[1])
axes[1].set_title('summer')

sns.histplot(s3,kde=True,ax=axes[2])
axes[2].set_title('fall')

sns.histplot(s4,kde=True,ax=axes[3])
axes[3].set_title('winter')

from statsmodels.api import qqplot
qqplot(s1,line='s',ax=axes[4])
axes[4].set_title('QQ-Plot for spring')

qqplot(s2,line='s',ax=axes[5])
axes[5].set_title('QQ-Plot for summer')

qqplot(s3,line='s',ax=axes[6])
axes[6].set_title('QQ-Plot for fall')

qqplot(s4,line='s',ax=axes[7])
```

```
axes[7].set_title('QQ-Plot for winter')

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



The density curves are not symmetric(skewed) and qqplot doesn't follow the line which clearly shows the distribution is not normal distribution.

[39]:
```
# 2. Statistical tests
#H0 : The data follows normal distribution ,Ha: the data doesn't follow normal distribution

from scipy.stats import shapiro
test_stat1,p_val1 = shapiro(s1)
test_stat2,p_val2 = shapiro(s2)
test_stat2,p_val3 = shapiro(s3)
test_stat2,p_val4 = shapiro(s4)

print("p_value of spring is ",p_val1)
print("p_value of summer is ",p_val2)
print("p_value of fall is ",p_val3)
print("p_value of winter is ",p_val4)
```

```
p_value of spring is  0.0
p_value of summer is  6.039093315091269e-39
p_value of fall is  1.043458045587339e-36
p_value of winter is  1.1301682309549298e-39
```

all pvalues are clearly < 0.05, rejecting null hypothesis. The data doesn't follow normal distribution

[40]:
```
#3. Summary Statistics:
# i) skewness
print("skewness of spring data is ",s1.skew())
print("skewness of summer data is ",s2.skew())
print("skewness of fall data is ",s3.skew())
print("skewness of winter data is ",s4.skew())

#ii)kurtosis
print("kurtosis of spring data is ",s1.kurt())
print("kurtosis of summer data is ",s2.kurt())
print("kurtosis of fall data is ",s3.kurt())
print("kurtosis of winter data is ",s4.kurt())
```

```
skewness of spring data is  1.8880559001782309
skewness of summer data is  1.0032642267278118
skewness of fall data is  0.9914946474772749
skewness of winter data is  1.172117329762622
kurtosis of spring data is  4.31475739331681
kurtosis of summer data is  0.42521337827415717
kurtosis of fall data is  0.6993825795653992
kurtosis of winter data is  1.2734853552995302
```

The data in all the season types are positively skewed(>0) and have both flatten and peak kurtosis

[41]:
```
#equal variance
#H0: The data groups are equally variant, Ha: The data groups are not equally␣
↪variant
from scipy.stats import levene
levene(s1,s2,s3,s4)
```

[41]:
```
LeveneResult(statistic=187.7706624026276, pvalue=1.0147116860043298e-118)
```

pvalue<0.05, rejecting null hypothesis. Those are not equally variant

since all the assumptions are false, we can use kruskals as per theory

[42]:
```
from scipy.stats import kruskal
kruskal(s1,s2,s3,s4)
```

[42]:
```
KruskalResult(statistic=699.6668548181988, pvalue=2.479008372608633e-151)
```

But as per the test asked to do in the problem given, **performing one way anova testing**

[43]:
```
import scipy.stats as stats
stats.f_oneway(s1,s2,s3,s4)
```

[43]:
```
F_onewayResult(statistic=236.94671081032106, pvalue=6.164843386499654e-149)
```

pvalue < 0.05(as the significance level given), hence rejecting null hypothesis.

The demand of bicycles does has impact on different seasons

1.0.5 Check if the Weather conditions are significantly different during different Seasons?

Setting up Null and Alternate Hypothesis:

H0 : The Weather conditions are not significantly different during different Seasons

Ha : The Weather conditions are significantly different during different Seasons

- **Significance level : 0.05**

Applying chi-square test since it is categorical - categorical with the significance level given as 0.05

[44]:
```
#creating contingency table

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

contingency_table
```

[44]:
```
season        fall  spring  summer  winter
weather
clear         1930    1759    1801    1702
intense          0       1       0       0
partly_clear   604     715     708     807
rain           199     211     224     225
```

[45]:
```
#since the values should be more than 5 in each cell for statistical stability,␣
↪removing the intense

ct_temp = df[df['weather']!='intense']
ct_temp['weather'].value_counts()
```

[45]:
```
weather
clear           7192
partly_clear    2834
rain             859
Name: count, dtype: int64
```

[46]:
```
contingency_table_updated = pd.crosstab(ct_temp['weather'],ct_temp['season'])
contingency_table_updated
```

[46]:
```
season        fall  spring  summer  winter
weather
clear         1930    1759    1801    1702
partly_clear   604     715     708     807
rain           199     211     224     225
```

[47]:
```
stats.chi2_contingency(contingency_table_updated)
```

[47]:
```
Chi2ContingencyResult(statistic=46.101457310732485,
pvalue=2.8260014509929403e-08, dof=6, expected_freq=array([[1805.76352779,
1774.04869086, 1805.76352779, 1806.42425356],
        [ 711.55920992,  699.06201194,  711.55920992,  711.81956821],
        [ 215.67726229,  211.8892972 ,  215.67726229,  215.75617823]]))
```

$pvalue < alpha$, hence rejecting null hypothesis

The Weather conditions are significantly different during different Seasons

1.1 Final Insights:

- There is no significance difference between no.of bikes on weekdays and weekends (as per above ttest).
- There is no significance difference between no.of bikes on holidays and non holidays (as per above ttest).
- The no.of bikes varies in different weathers and seasons (as per anova test).
- The weather is significantly different in different seasons (as per chisquare test).
- More rental bike sales are in clear weather.
- casual users are less in workingdays and more casual user sales are in fall season
- when the humidity is high, sales are low
- when the temperature is high,sales are comparitively highn

1.2 Recommendations:

- To increase the casual users, offers like first ride discount can be implemented.
- Advertising or Promotional activities should be done irrespective of holidays/Nonholidays and either workingday or not, since all those has equal demand.
- Ensure to maintain more bikes in fall season and clear weather conditions as it recorded high bike rides comparitively
- Any holiday/promotional campaign or bicycle marathons can be started to attract more users.
- Ensure to maintain quality of bikes and prices to retain the already registered users.