Imported necessary libraries and loading data

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import ttest_ind, ttest_rel, chi2, chi2_contingency, f_oneway
```

```
!gdown 1s74_TfPuURd92v_uzZxmT6-WTmVC85Mv
```

```
Downloading...
From: https://drive.google.com/uc?id=1s74_TfPuURd92v_uzZxmT6-WTmVC85Mv
To: /content/yulu.txt
100% 648k/648k [00:00<00:00, 7.17MB/s]
```

```
df=pd.read_csv('/content/yulu.txt')
df
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	regis
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0000	3	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000	8	
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000	5	
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000	3	
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000	0	
...	...	...	...	...	...	...	...	...	...	...	...
	2012-12-										

Exploring Data

```
df.shape
```

```
(10886, 12)
```

```
df.info()
```

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

```
# Datatype of following attributes needs to changed to proper data type

# datetime - to datetime
# season - to categorical
```

```
# holiday - to categorical
# workingday - to categorical
# weather - to categorical
```

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

```
df['season'] = df['season'].astype('object')
df['holiday'] = df['holiday'].astype('object')
df['workingday'] = df['workingday'].astype('object')
df['weather'] = df['weather'].astype('object')
```

```
df.info()
```

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

we can see that the few attributes has been changed to categorical values

```
df.describe()
```

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

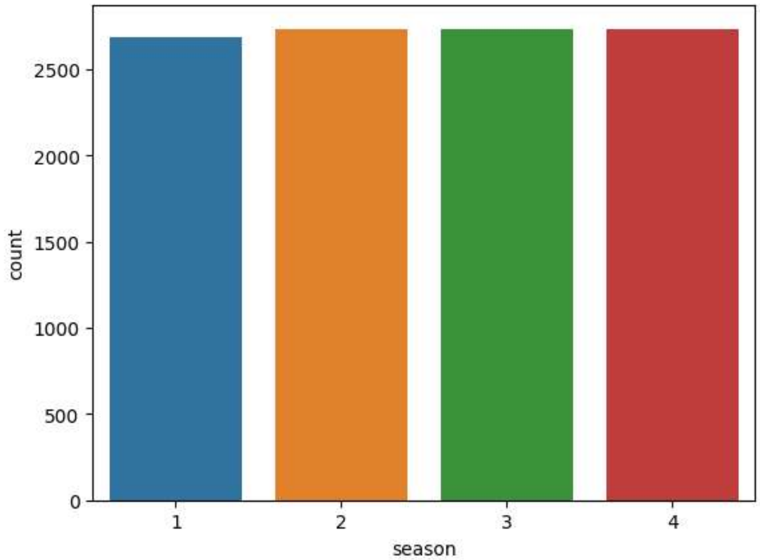
```
df.isna().sum()
```

```
datetime      0
season        0
holiday       0
workingday    0
weather       0
temp          0
atemp         0
humidity      0
windspeed     0
casual        0
registered    0
count         0
dtype: int64
```

we can see that there are no null values

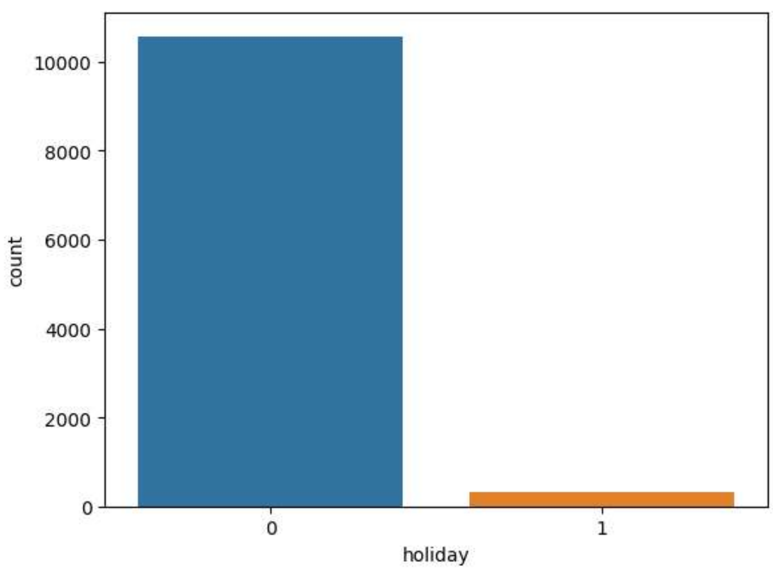
Univariate Analysis

```
sns.countplot(data = df, x = 'season')
plt.show()
```

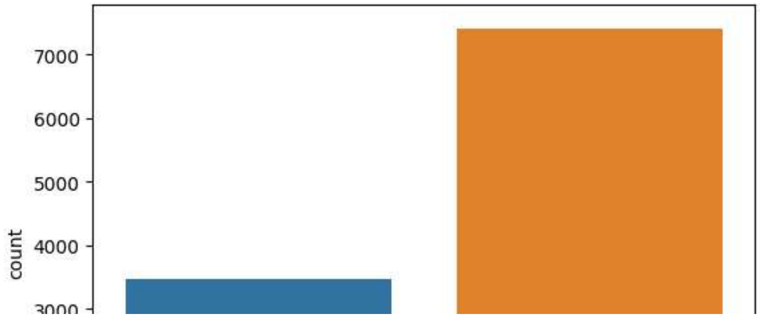


season: season (1: spring, 2: summer, 3: fall, 4: winter)

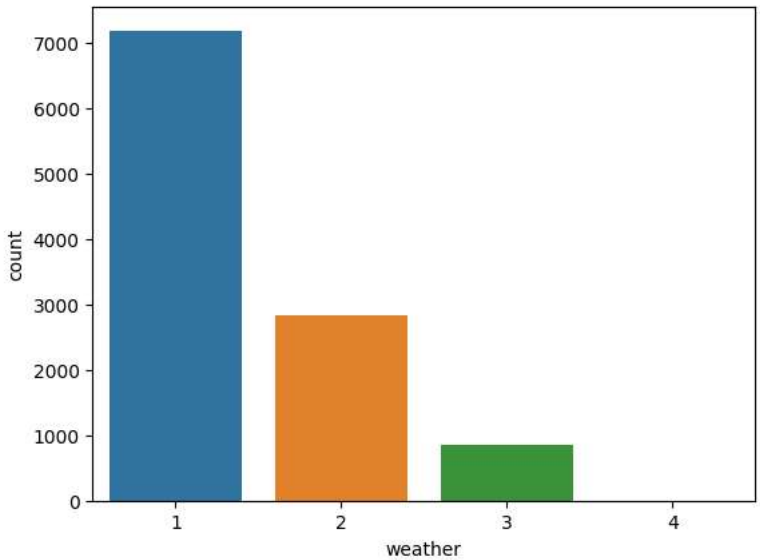
```
sns.countplot(data = df, x = 'holiday')
plt.show()
```




```
sns.countplot(data = df, x = 'workingday')
plt.show()
```




```
sns.countplot(data = df, x = 'weather')
plt.show()
```



weather: 1: Clear, Few clouds, partly cloudy, partly cloudy

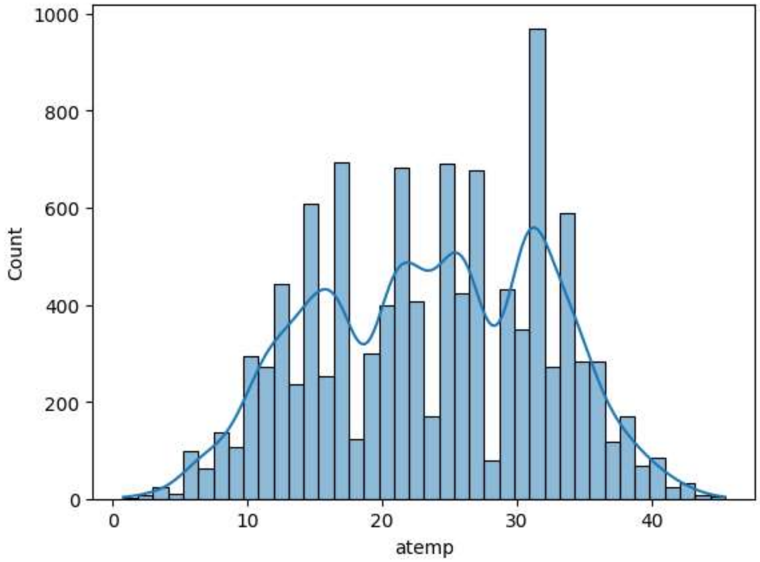
2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist

3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds

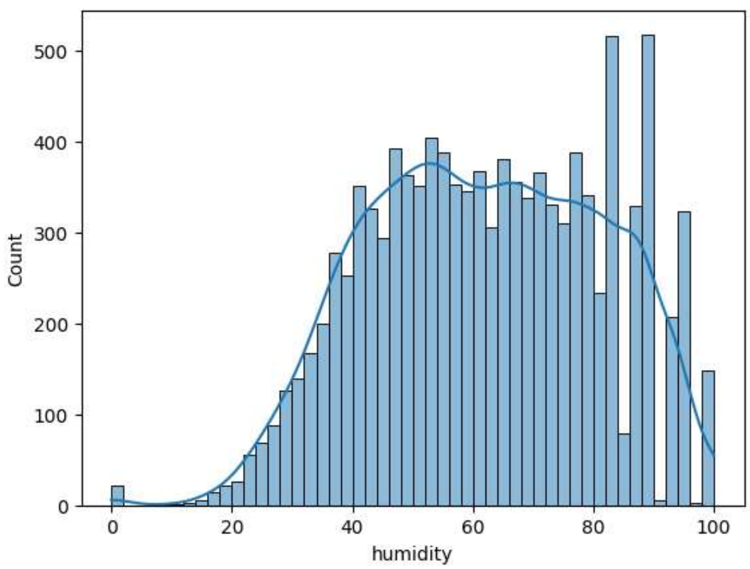
4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

```
sns.histplot(data = df, x = 'temp', kde = True, bins = 40)
plt.show()
```


```
sns.histplot(data = df, x = 'atemp', kde = True, bins = 40)
plt.show()
```

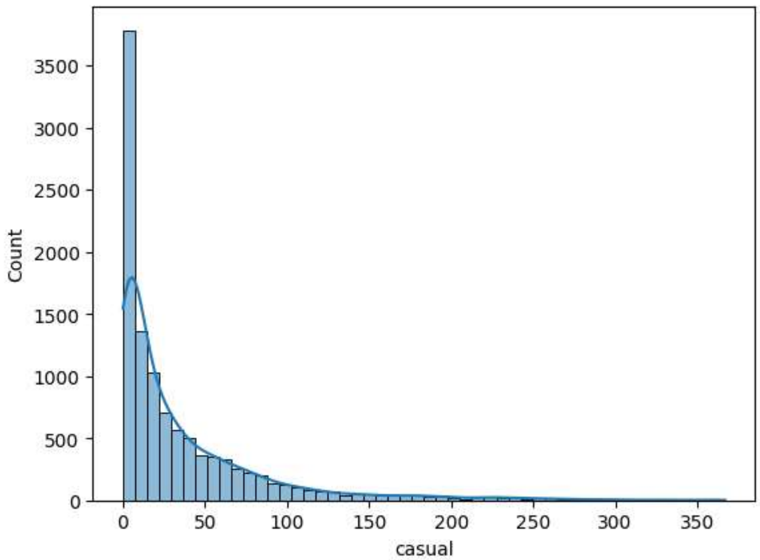



```
sns.histplot(data = df, x = 'humidity', kde = True, bins = 50)
plt.show()
```

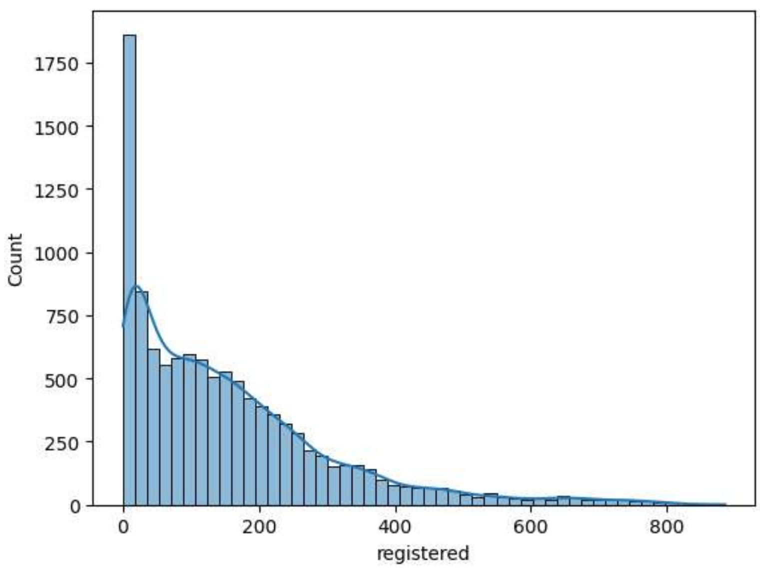



```
sns.histplot(data = df, x = 'windspeed', kde = True, bins = 50)
plt.show()
```

```
sns.histplot(data = df, x = 'casual', kde = True, bins = 50)
plt.show()
```




```
sns.histplot(data = df, x = 'registered', kde = True, bins = 50)
plt.show()
```



outlier detection

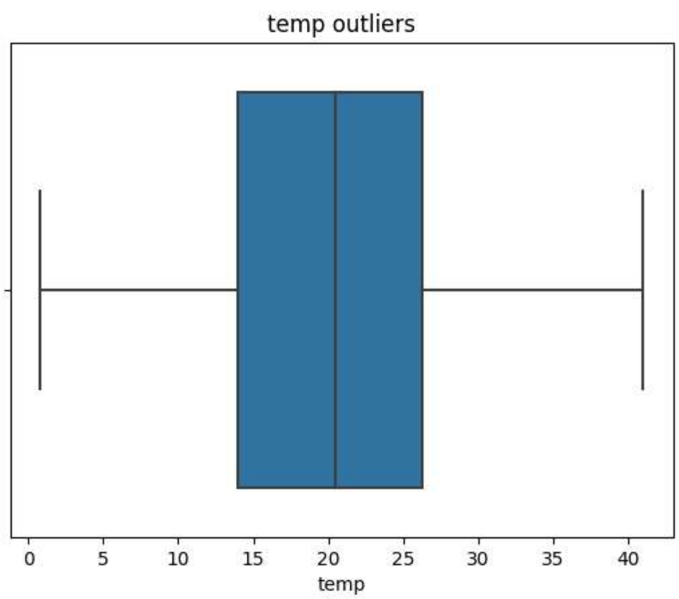
```
# Selecting numerical columns from the DataFrame
numerical_columns = df.select_dtypes(include=['number'])

# Calculating quartiles and IQR for numerical columns
Q1 = numerical_columns.quantile(0.25)
Q3 = numerical_columns.quantile(0.75)
IQR = Q3 - Q1

# Calculating lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
```

```
upper_bound = Q3 + 1.5 * IQR

# Finding outliers for numerical columns
outliers = ((numerical_columns < lower_bound) | (numerical_columns > upper_bound)).sum()

print("Outliers")
print(outliers)
```

```
Outliers
temp            0
atemp           0
humidity       22
windspeed     227
casual        749
registered    423
count         300
dtype: int64
```

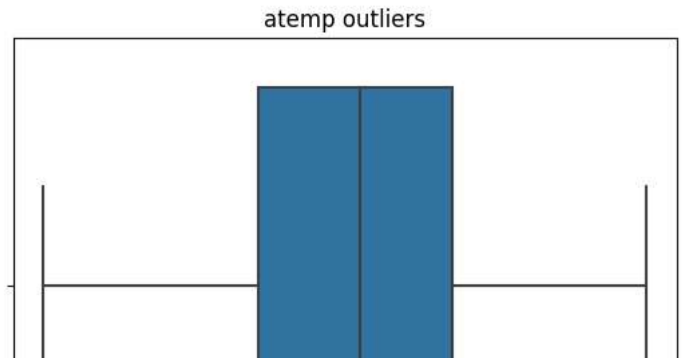
As we can see here season have the most outlier

```
# Outlier detection
plt.subplot()
plt.title('temp outliers')
sns.boxplot(data = df, x = 'temp')
plt.show()
```

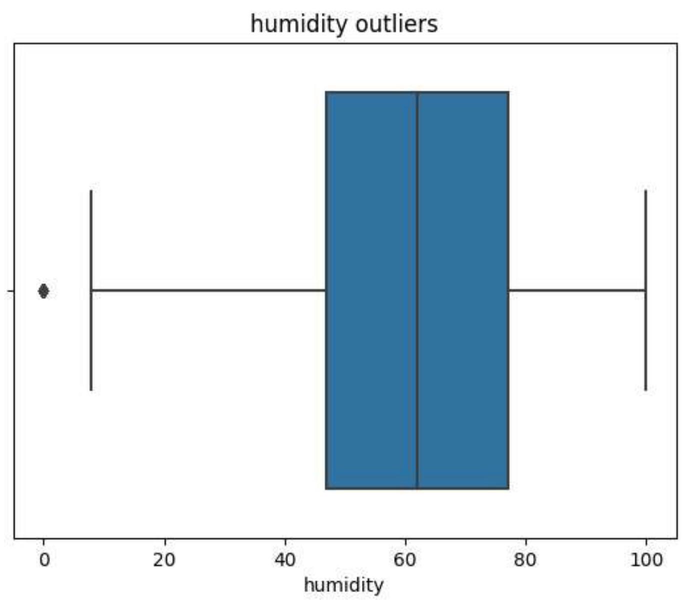


we can see there are no outliers in temp

```
# Outlier detection
plt.subplot()
plt.title('atemp outliers')
sns.boxplot(data = df, x = 'atemp')
plt.show()
```

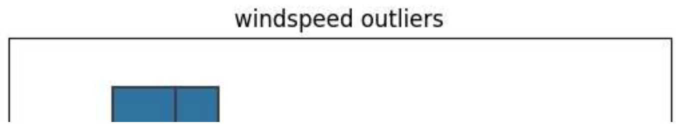



```
# Outlier detection
plt.subplot()
plt.title('humidity outliers')
sns.boxplot(data = df, x = 'humidity')
plt.show()
```

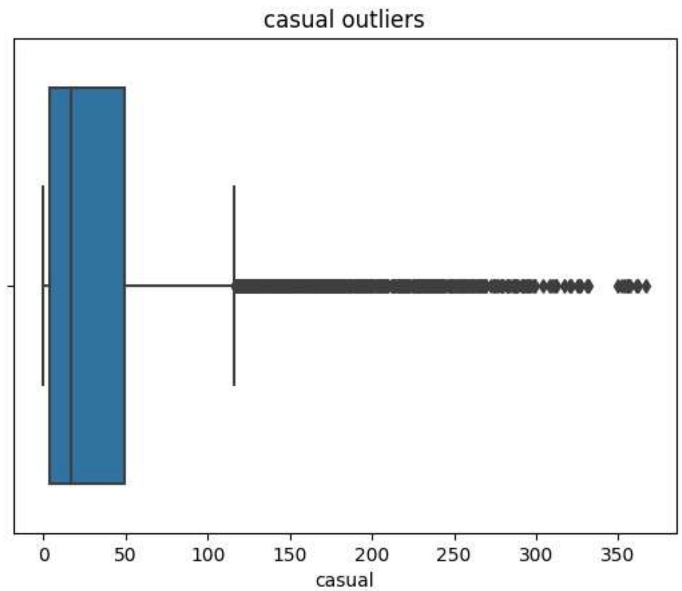


There are few outliers present in humidity column.

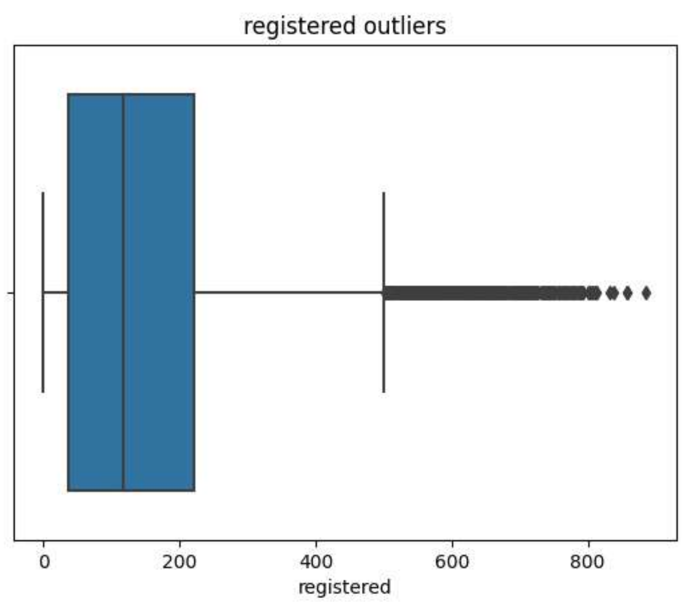
```
# Outlier detection
plt.subplot()
plt.title('windspeed outliers')
sns.boxplot(data = df, x = 'windspeed')
plt.show()
```



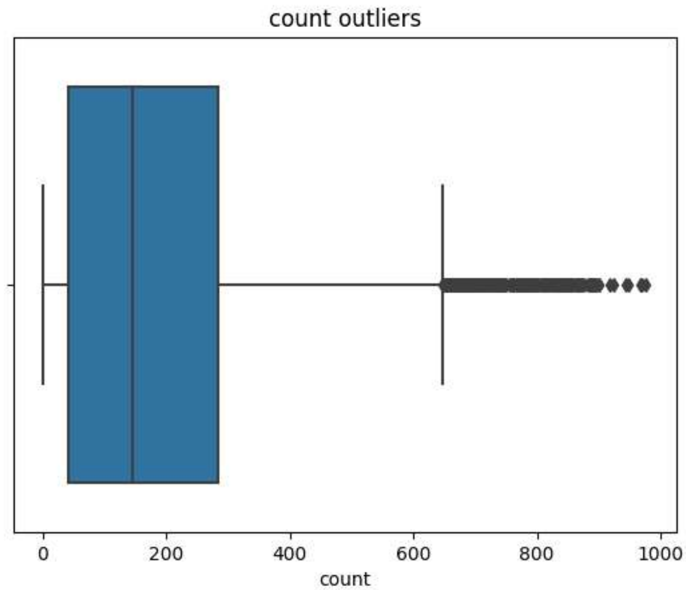

```
# Outlier detection
plt.subplot()
plt.title('casual outliers')
sns.boxplot(data = df, x = 'casual')
plt.show()
```




```
# Outlier detection
plt.subplot()
plt.title('registered outliers')
sns.boxplot(data = df, x = 'registered')
plt.show()
```



```
# Outlier detection
plt.subplot()
plt.title('count outliers')
sns.boxplot(data = df, x = 'count')
plt.show()
```

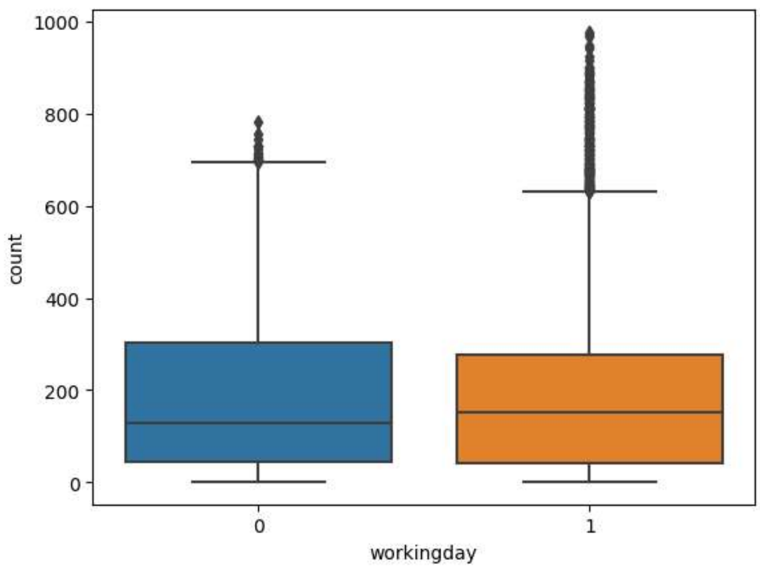


There are many outliers present in each of the columns : windspeed, casual, registered, count

Bivariate Analysis

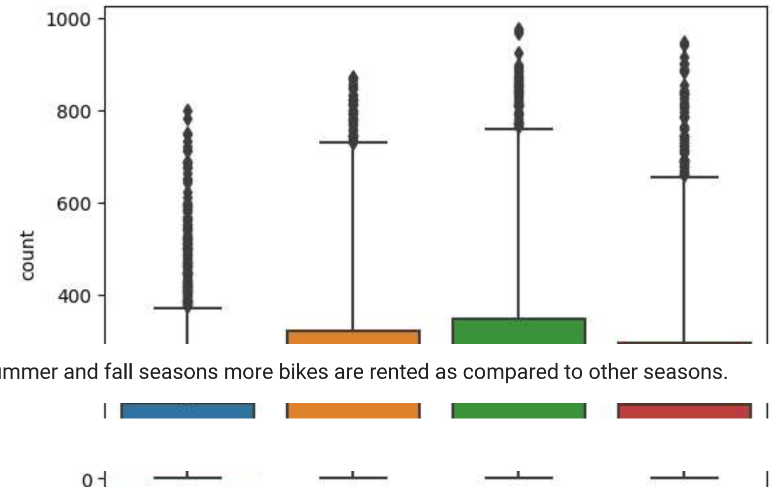
```
sns.boxplot(data = df, x = 'workingday', y = 'count')
```

```
<Axes: xlabel='workingday', ylabel='count'>
```




```
sns.boxplot(data = df, x = 'season', y='count')
```

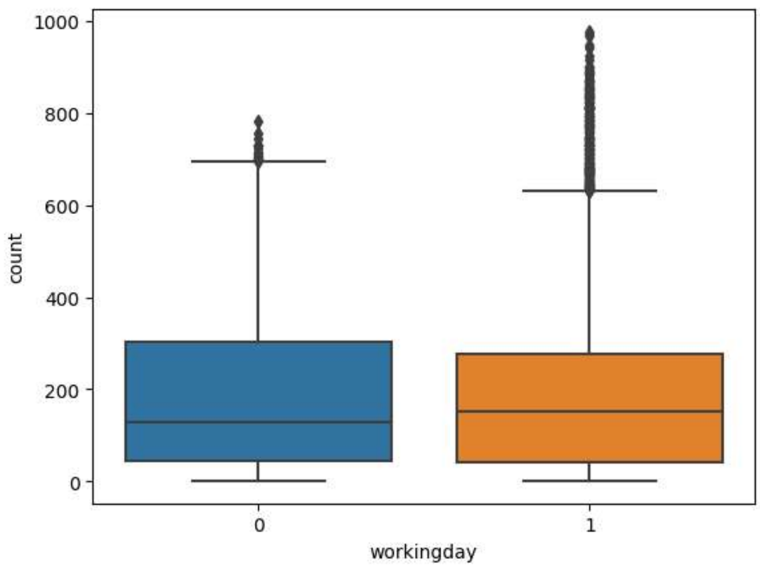
```
<Axes: xlabel='season', ylabel='count'>
```



In summer and fall seasons more bikes are rented as compared to other seasons.

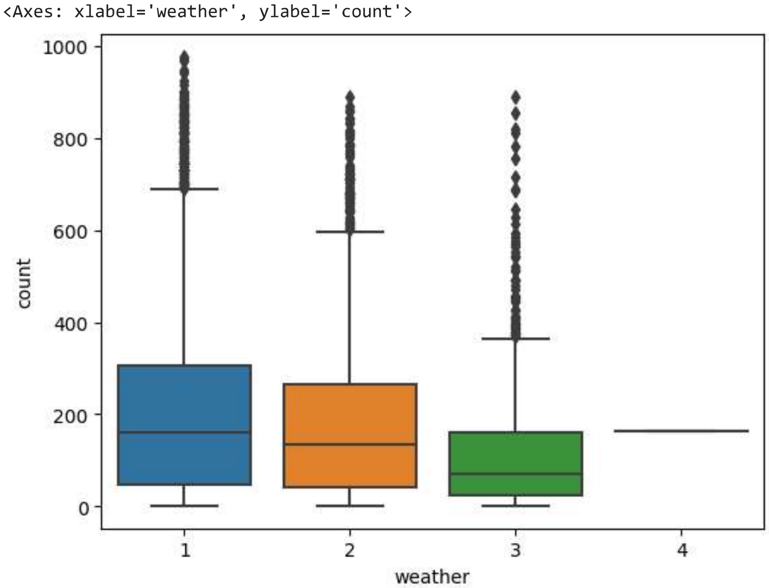
```
sns.boxplot(data = df, x = 'workingday', y = 'count')
```

```
<Axes: xlabel='workingday', ylabel='count'>
```




```
sns.boxplot(data = df, x = 'weather', y = 'count')
```

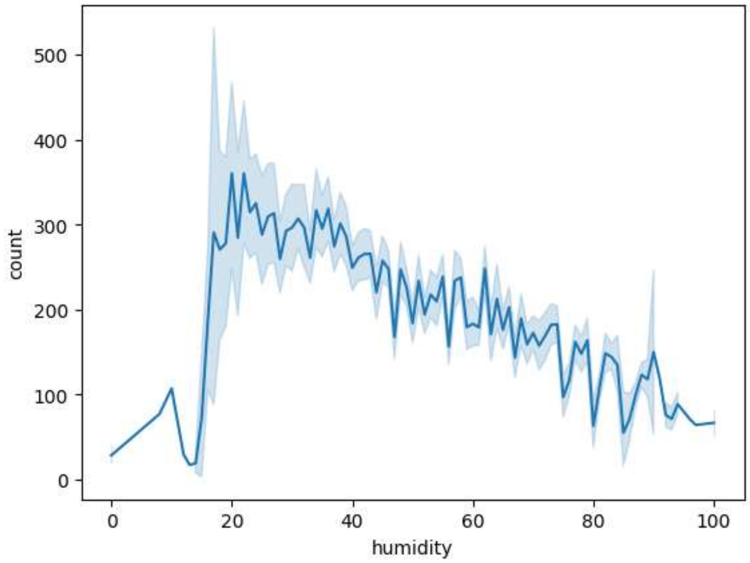
```
<Axes: xlabel='weather', ylabel='count'>
```



Whenever there is rain, thunderstorm, snow or fog, there were less bikes were rented.

```
sns.lineplot(data = df, x = 'humidity', y = 'count')
```

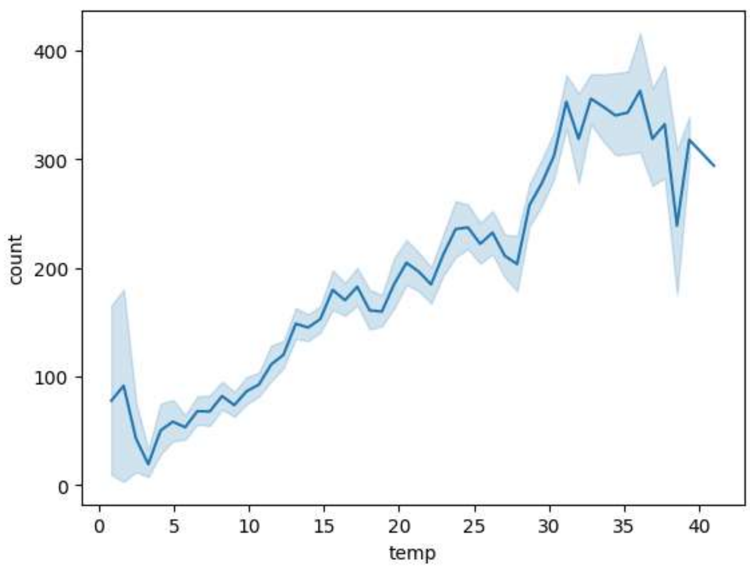
```
<Axes: xlabel='humidity', ylabel='count'>
```



Whenever the humidity is less than 20, number of bikes rented is very very low.

```
sns.lineplot(data = df, x = 'temp', y = 'count')
```

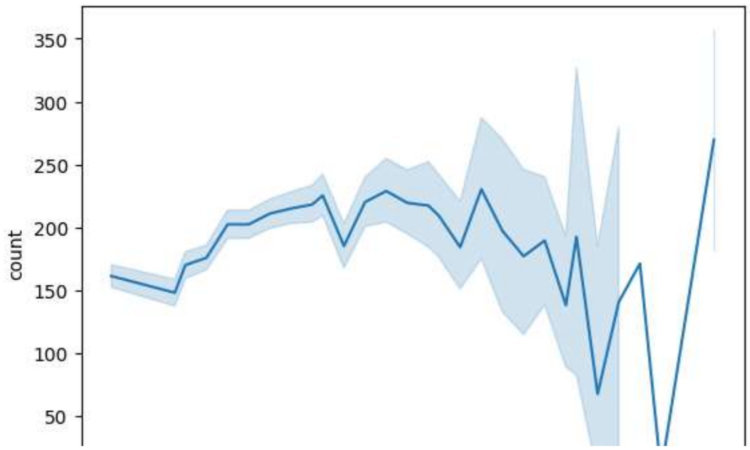
```
<Axes: xlabel='temp', ylabel='count'>
```



Whenever the temperature is less than 10, number of bikes rented is less.

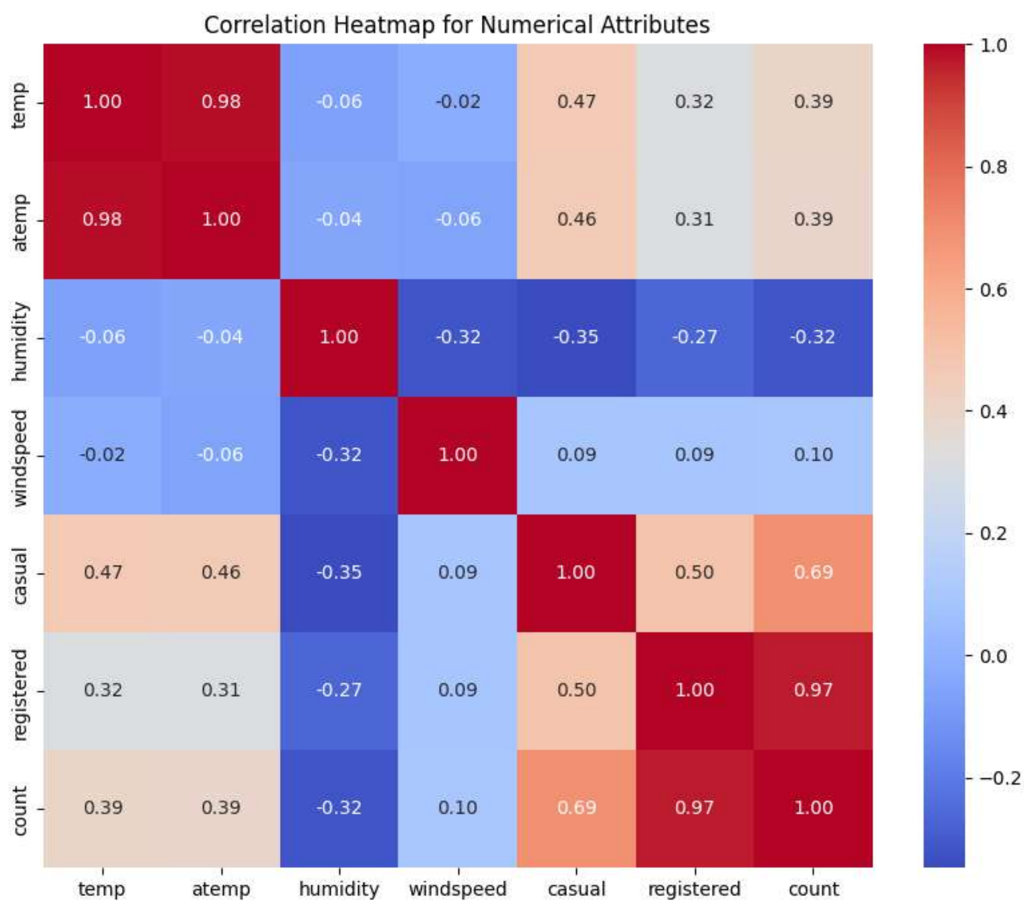
```
sns.lineplot(data = df, x = 'windspeed', y = 'count')
```

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



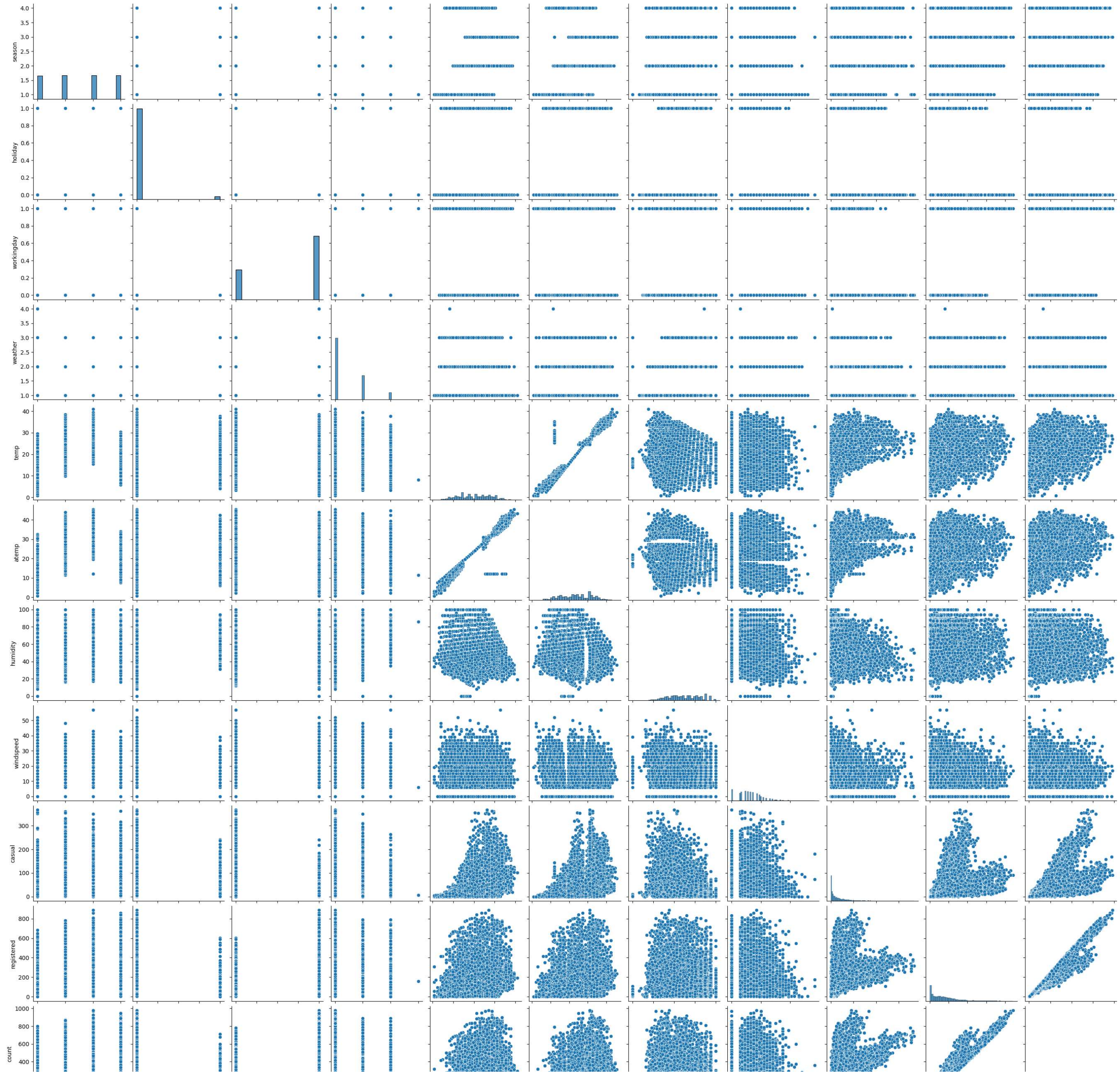
Whenever the windspeed is greater than 35, number of bikes rented is less.

```
#  Correlation heatmap for numerical attributes
numerical_corr_matrix = df.select_dtypes(include='number').corr()
plt.figure(figsize=(10, 8))
sns.heatmap(numerical_corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap for Numerical Attributes')
plt.show()
```




```
sns.pairplot(data=df)
```

```
<seaborn.axisgrid.PairGrid at 0x7902e94b5a80>
```



Hypothesis testing

```
# 2- Sample T-Test to check if Working Day has an effect on the number of electric cycles rented
df.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

```
working_day = df[df['workingday']==1]['count'].values
working_day
```

```
array([  5,   2,   1, ..., 168, 129,  88])
```

```
non_working_day = df[df['workingday']==0]['count'].values
non_working_day
```

```
array([ 16,  40,  32, ..., 106,  89,  33])
```

```
np.var(working_day)
```

```
34040.69710674686
```

```
np.var(non_working_day)
```

```
30171.346098942427
```

Before conducting the two-sample T-Test we need to find if the given data groups have the same variance. If the ratio of the larger data groups to the small data group is less than 4:1 then we can consider that the given data groups have equal variance.

Here, the ratio is 34040.70 / 30171.35 which is less than 4:1

Ho : Working day has no effect on the number of cycles being rented.

Ha : Working day has effect on the number of cycles being rented.

Alpha : 0.05

We will use the 2-Sample T-Test to test the hypothess defined above

```
stats, pvalue = ttest_ind(working_day, non_working_day, equal_var = True)
print('stats', stats)
print('pvalue', pvalue)
```

```
stats 1.2096277376026694
pvalue 0.22644804226361348
```

```
alpha = 0.05
if pvalue<alpha:
  print('reject Ho')
else:
  print('fail to reject Ho')
```

```
fail to reject Ho
```

Since pvalue is greater than 0.05 so we can not reject the Null hypothesis. We don't have the sufficient evidence to say that working day has effect on the number of cycles being rented

ANNOVA to check if No. of cycles rented is similar or different in different 1. weather 2. season

Ho : Number of cycles rented is similar in different weather and season.

Ha : Number of cycles rented is not similar in different weather and season.

alpha: 0.05

Here, we will use the ANOVA to test the hypothess defined above

```
w1 = df[df['weather']==1]['count'].values
w2 = df[df['weather']==2]['count'].values
w3 = df[df['weather']==3]['count'].values
w4 = df[df['weather']==4]['count'].values

s1 = df[df['season']==1]['count'].values
s2 = df[df['season']==2]['count'].values
s3 = df[df['season']==3]['count'].values
s4 = df[df['season']==4]['count'].values

stats, pvalue = f_oneway(w1, w2, w3, w4, s1, s2, s3, s4)
print('stats', stats)
print('pvalue', pvalue)
```

```
stats 127.96661249562491
pvalue 2.8074771742434642e-185
```

```
alpha = 0.05
if pvalue<alpha:
  print('Reject Ho')
else:
  print('Fail to reject Ho')
```

```
Reject Ho
```

Since p-value is less than 0.05, we reject the null hypothesis. This implies that Number of cycles rented is not similar in different weather and season conditions

Chi-square test to check if Weather is dependent on the season

H0 : Weather is independent of the season

Ha : Weather is not independent of the season

alpha: 0.05

We will use chi-square test to test hypyothesis defined above.

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

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

```
stats, pvalue, dof, expected_freq = chi2_contingency(data_table)
print('stats', stats)
print('pvalue', pvalue)
print('degrees of freedom', dof)
print('expected_freq', expected_freq)
```

```
stats 49.158655596893624
pvalue 1.549925073686492e-07
degrees of freedom 9
expected_freq [[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]]
```

```
alpha = 0.05
if pvalue<alpha:
  print('reject Ho')
```

```
else:
  print('fail to reject Ho')
```

```
reject Ho
```

Since p-value is less than the alpha 0.05, We reject the Null Hypothesis. Meaning that Weather is dependent on the season.

Which variables are significant in predicting the demand for shared electric cycles in the Indian market?

Based on our analysis, we tested the significance of various variables on the demand for shared electric cycles. Here got some insights like as given:

Working Day: We conducted a two-sample T-Test to determine if working days significantly affect the number of cycles being rented. The results indicate that there isn't sufficient evidence to conclude that working day significantly influences demand. Therefore, it may not be a significant predictor of demand.

Weather and Season: We used ANOVA to assess the impact of weather and season on the number of cycles rented. The analysis revealed that the number of cycles rented varies significantly across different weather and season conditions. This suggests that both weather and season are significant factors in predicting demand.

We aslo performed a chi-square test, which indicated that weather is dependent on the season.

How well those variables describe the electric cycle demands?

Weather and season significantly describe electric cycle demand, while other variables' impact on demand needs further analysis.

✓ 0s completed at 7:22 PM