Yulu

April 9, 2023

[46]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pylab

import scipy.stats as stats
from scipy.stats import f,f_oneway #Anova Testing
from scipy.stats import kruskal #Kruskal Testing
from scipy.stats import mannwhitneyu # Alternate for anova if data is not normal
from scipy.stats import ttest_ind,ttest_rel #Two Sample Test for independent␣
↪variable
from scipy.stats import norm
from scipy.stats import chi2_contingency,chi2 #Test for two categorical Values
from scipy.stats import shapiro #Test for normality
```

[6]:
```
df = pd.read_csv("bike_sharing.csv")
```

[7]:
```
df
```

[7]:
```
                  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]
```

[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
```

[9]:
```
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
```

[10]:
```
df["datetime"] = pd.to_datetime(df["datetime"])
```

[11]:
```
df.head(3)
```

[11]:
```
             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

   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
```

[12]:
```
df["weather"].value_counts()
```

[12]:
```
1    7192
2    2834
3     859
4       1
Name: weather, dtype: int64
```

[13]:
```
print(df["season"].value_counts())
print(df["workingday"].value_counts())
```

```
4    2734
2    2733
3    2733
1    2686
Name: season, dtype: int64
1    7412
0    3474
```

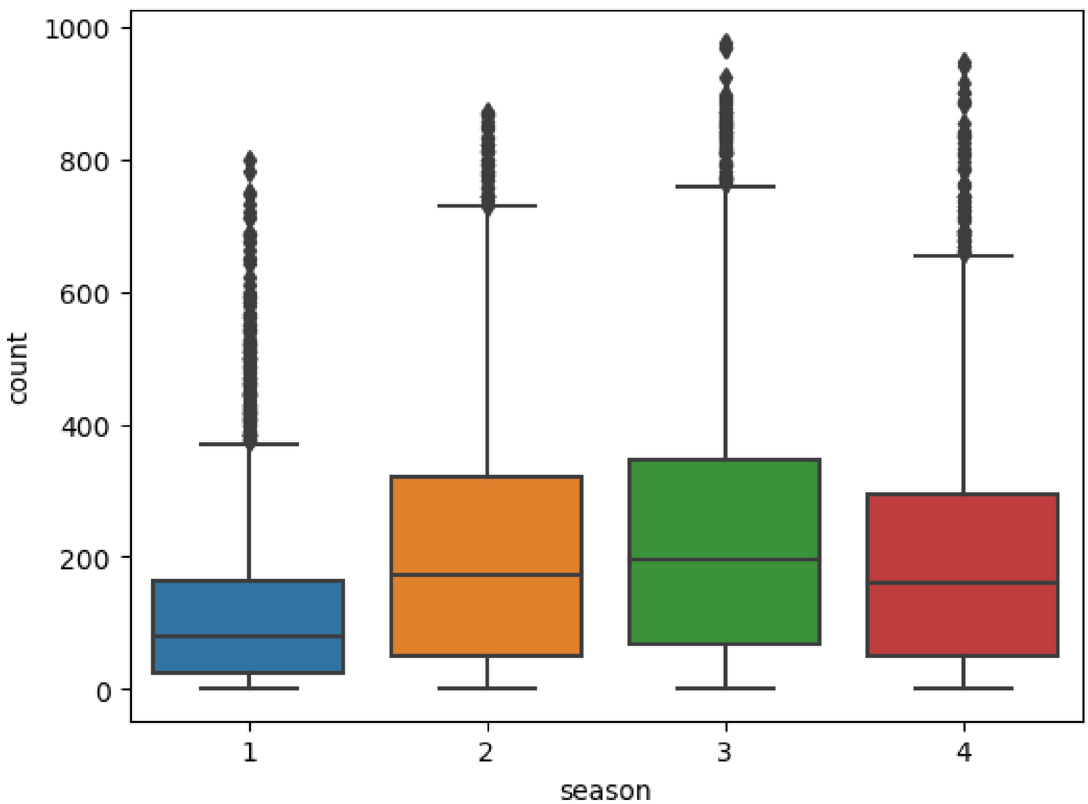
```
Name: workingday, dtype: int64
```

[14]:
```
df["holiday"].value_counts()
```

[14]:
```
0    10575
1      311
Name: holiday, dtype: int64
```

[15]:
```
sns.boxplot(data=df,x="season",y="count")
```

[15]:
```
<AxesSubplot:xlabel='season', ylabel='count'>
```



[16]:
```
df.shape
```

[16]:
```
(10886, 12)
```

Test to check our Number of cycles rented is normal or not

[17]:
```
# Majority we will use count variable,
# But from plot we are not able to check its a gaussian distribution or not
# Let's statistically prove that by shapiro test
```

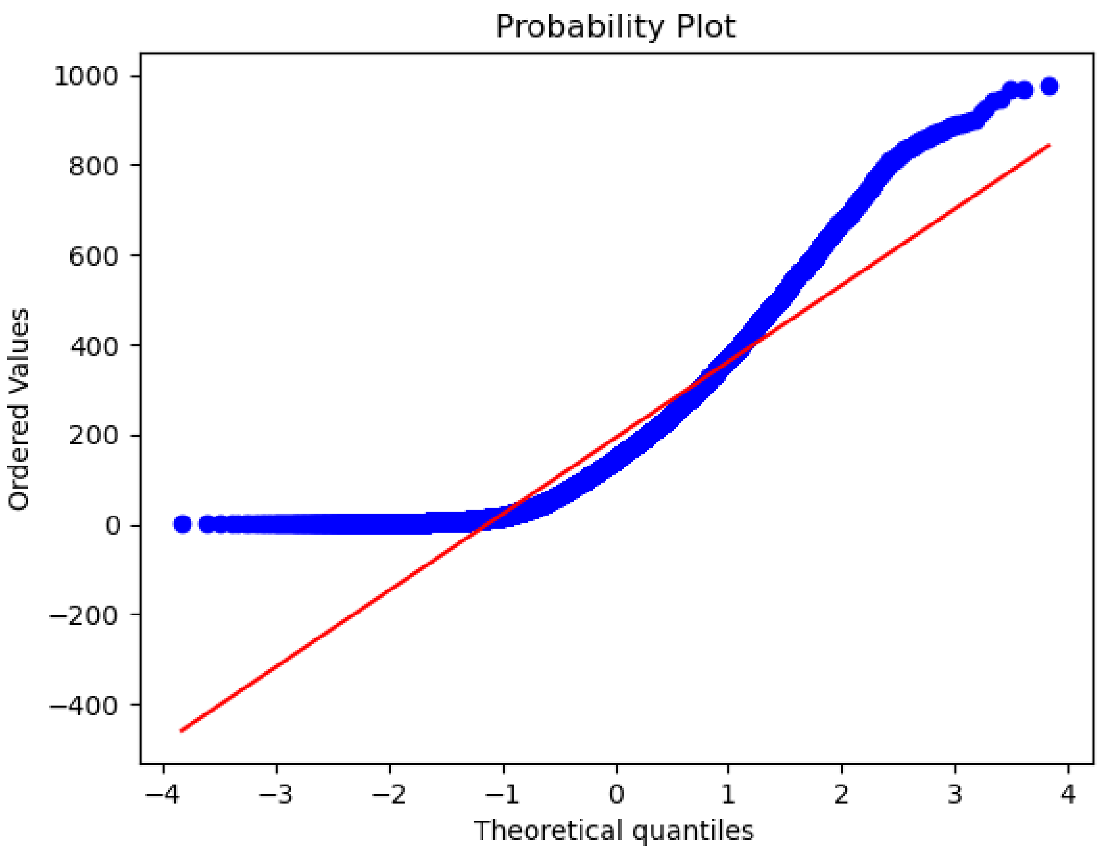
[18]:
```
# Shapiro Test
#______________
# HO: Number cycle rented is Normally distributed
# Ha: Number cycle rented is Not normally distributed
# Significant Value: 0.05
```

[19]:
```
alpha = 0.05

kruskal_stat, p_value = shapiro(df["count"])
if p_value<alpha:
    print("Reject Null Hypothesis")
else:
    print("Fail to reject Null Hypothesis")
print("Test Statistic Value: ",kruskal_stat)
print("P_value:",p_value)
print("Critical Value: ")
```

```
Reject Null Hypothesis
Test Statistic Value:  0.8783695697784424
P_value: 0.0
Critical Value:
```

```
c:\Users\revan\anaconda3\lib\site-packages\scipy\stats\_morestats.py:1800:
UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

[49]:
```
# Graphical checking of normality
# Quartile-Quartile plot
stats.probplot(df["count"],dist="norm",plot=pylab)
pylab.show()
```



[ ]:
```
# we can see that the points are nor lying in straight line
# From test and graphical representation we can see that count feature is not␣
↪normally distributed
# Hence we will use kruskal instead of anova.
```

Hypothetical testing between season and count

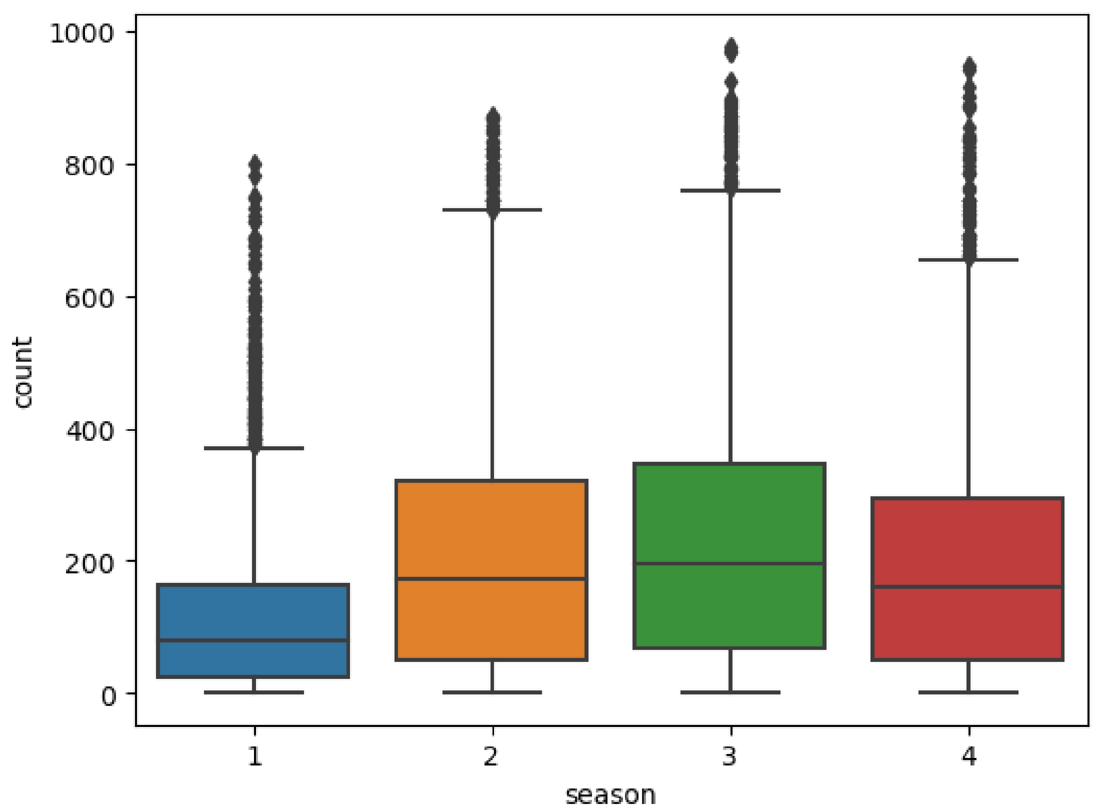
[20]:
```
df[(df["count"] >0) & (df["count"] <100)]
```

[20]:
```
                  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
…                      …     …       …         …        …     …
10864  2012-12-19 02:00:00       4        0           1        1  11.48
10865  2012-12-19 03:00:00       4        0           1        1  10.66
10866  2012-12-19 04:00:00       4        0           1        1   9.84
10867  2012-12-19 05:00:00       4        0           1        1  10.66
```

```
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
…          …       …        …     …       …     …
10864   15.150        65     6.0032       1           2      3
10865   13.635        75     8.9981       0           5      5
10866   12.120        75     8.9981       1           6      7
10867   14.395        75     6.0032       2          29     31
10885   16.665        66     8.9981       4          84     88

[4312 rows x 12 columns]
```

[21]:
```
sns.boxplot(data=df,x="season",y="count")
```

[21]:
```
<AxesSubplot:xlabel='season', ylabel='count'>
```



[22]:
```
# Anova or Kruskal Walli's Test
# Assumption for Anova:
# ---------------------

# 1. The population from which samples are drawn should be normally distributed.
 ↪ -- False
# 2. Independence of cases: the sample cases should be independent of each␣
 ↪other. -- True
# 3. Homogeneity of variance: Homogeneity means that the variance among the␣
 ↪groups should be approximately equal. -- True

# Our data doesn't meet the requirements to conduct anova test for these two␣
 ↪variables, Hence we are going to use Kruskal Wallis test
#␣
 ↪-----------------------------------------------------------------------------------------------

# H0 : Mean of count for all season is same
# Ha : Mean of each season count is varies
# Significant Value: 0.05
# Critical Value: 2.605725028634713
```

[23]:
```
alpha = 0.05
cr = f.ppf(1-alpha,dfn=3,dfd=10886-3)
kruskal_stat, p_value = kruskal(
                        df[df["season"]==1]["count"],
                        df[df["season"]==2]["count"],
                        df[df["season"]==3]["count"],
                        df[df["season"]==4]["count"],
)
if p_value<alpha:
    print("Reject Null Hypothesis")
else:
    print("Fail to reject Null Hypothesis")
print("Test Statistic Value: ",kruskal_stat)
print("P_value:",p_value)
print("Critical Value: ", cr)
```

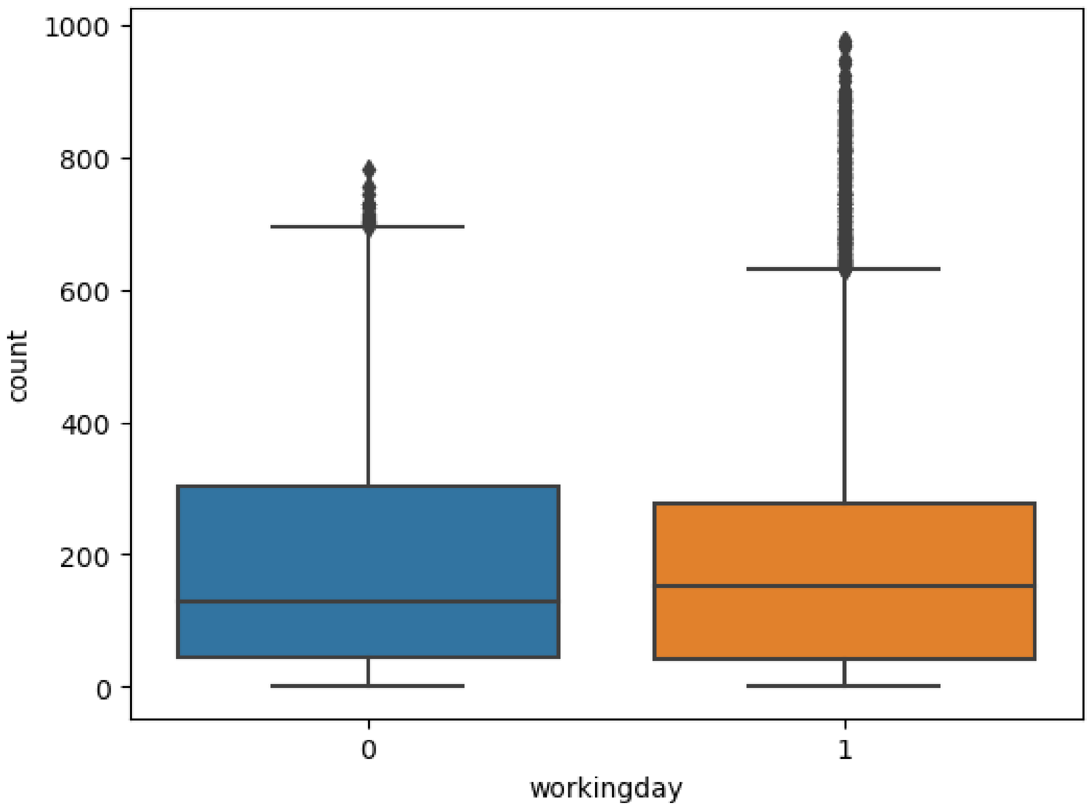
```
Reject Null Hypothesis
Test Statistic Value:  699.6668548181988
P_value: 2.479008372608633e-151
Critical Value:  2.605725028634713
```

[24]:
```
# After Test
# we rejecting our null hypothesis, which means one group season data mean is␣
 ↪not identical to other season's data
# From the above test we can identified that No. of cycles rented are varies in␣
 ↪different seasons.
```

Hypothetical testing between Working Day and Number of electric cycles rented
#Working Day has effect on number of electric cycles rented

[25]:
```
sns.boxplot(data=df,x="workingday",y="count")
```

[25]: <AxesSubplot:xlabel='workingday', ylabel='count'>



[26]:
```
# Anova or ttest_ind(Two groups of sample only)
# Assumption for Anova:
# ---------------------

# 1. The population from which samples are drawn should be normally distributed.
 ↪ -- False
# 2. Independence of cases: the sample cases should be independent of each␣
 ↪other. -- True
# 3. Homogeneity of variance: Homogeneity means that the variance among the␣
 ↪groups should be approximately equal. -- True

# Our data doesn't meet the requirements to conduct anova test for these two␣
 ↪variables, Hence we are going to use mannwhitneyu
# Because our dependent variable is not normally distributed
```

```
#␣
 ↪--------------------------------------------------------------------------------------------

# H0 : Mean of count for working day and holiday is same
# Ha : Mean of count varies for working day and holiday
# Significant Value: 0.05
# Critical Value: 2.605725028634713
```

[27]:
```
alpha = 0.05
cr = f.ppf(1 - alpha,dfn=1,dfd=10886-1)
kruskal_stat, p_value = mannwhitneyu(
                        df[df["workingday"]==0]["count"],
                        df[df["workingday"]==1]["count"],
                        )
if p_value<alpha/2:
    print("Reject Null Hypothesis")
else:
    print("Fail to reject Null Hypothesis")
print("Test Statistic Value: ",kruskal_stat)
print("P_value:",p_value)
print("Critical Value: ", cr)
```

```
Fail to reject Null Hypothesis
Test Statistic Value:  12880792.5
P_value: 0.9679139953914079
Critical Value:  3.842313268641915
```

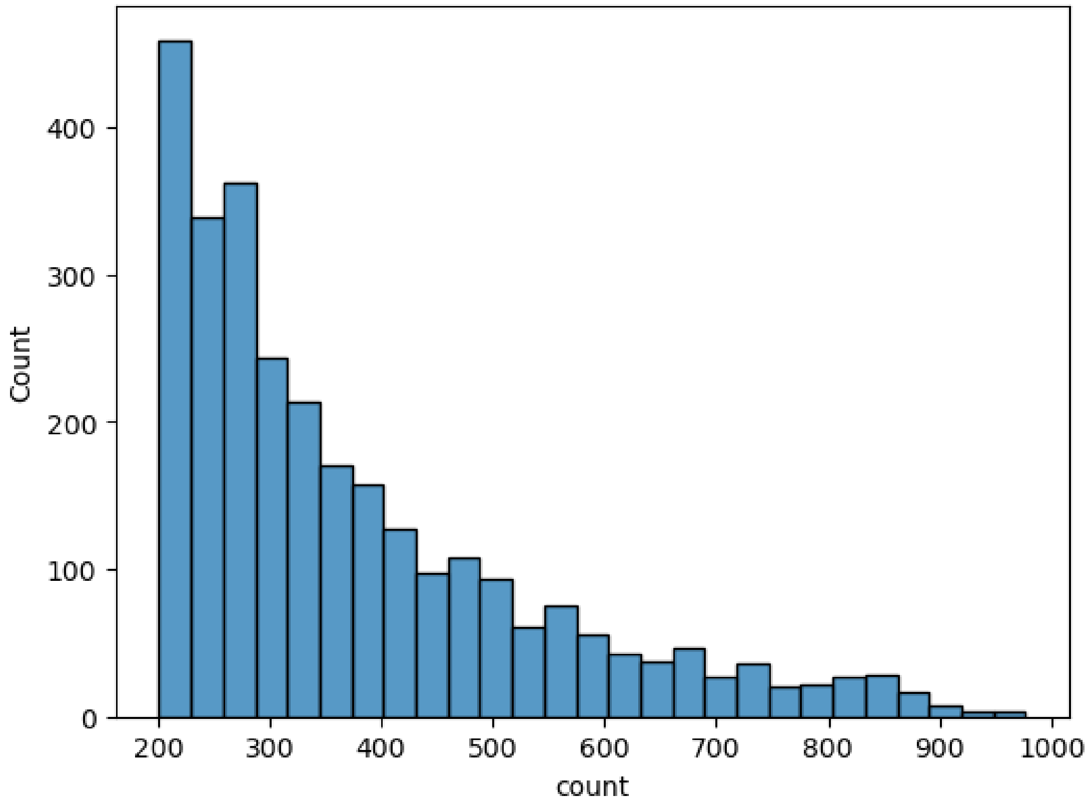
[28]:
```
# We conduct a manwhitneyu test because our our sample data of count doesn't␣
 ↪follow normal distribution
# Our test failed to reject null hypothesis, which means that the working day␣
 ↪won't cause anything in number of cycles rented
# From the test we found that irrespective of working day or holiday cycles are␣
 ↪rented by people
```

[29]:
```
df.groupby("workingday")["count"].mean()
```

[29]:
```
workingday
0    188.506621
1    193.011873
Name: count, dtype: float64
```

[30]:
```
sns.histplot((df[(df["workingday"] == 1)&(df["count"] > 200)]["count"]))
```

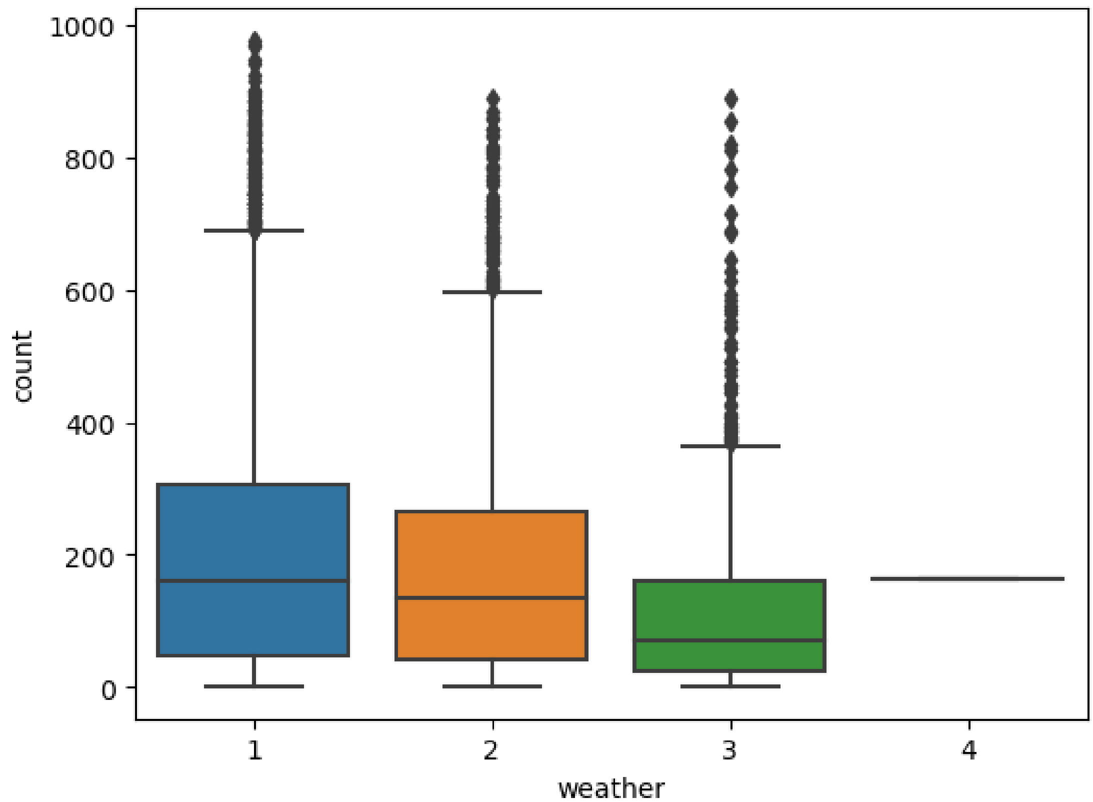
[30]:
```
<AxesSubplot:xlabel='count', ylabel='Count'>
```



Hypothetical testing for Weather and count

[31]:
```
sns.boxplot(data=df,x="weather",y="count")
```

[31]: <AxesSubplot:xlabel='weather', ylabel='count'>



[32]:
```
# Weather Characteristics
#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
# __________________________________________________________
# Analysis from data
# There are  many outliers in weather and count relation
# Weather creates a major impact in count of cycles
# Lets proove the above analysis statistically
```

[33]:
```
# Anova or kruskal wills
# Assumption for Anova:
# ---------------------

# 1. The population from which samples are drawn should be normally distributed.
↪ -- False
# 2. Independence of cases: the sample cases should be independent of each␣
↪other. -- True
```

```
# 3. Homogeneity of variance: Homogeneity means that the variance among the␣
↪groups should be approximately equal. -- True

# Our count data is not normally distributed and we can't use any normal␣
↪distribution tests here
# Hence we will go with Kruskal's will test to find whether the weather feature␣
↪creates any impact on count data
#␣
↪-------------------------------------------------------------------------------------

# H0 : Weather doesn't make any impact on cycles rented
# Ha : Weather makes a particular amount of impact on cycles rented
# Significant Value: 0.05
# Critical Value: 2.605725028634713
```

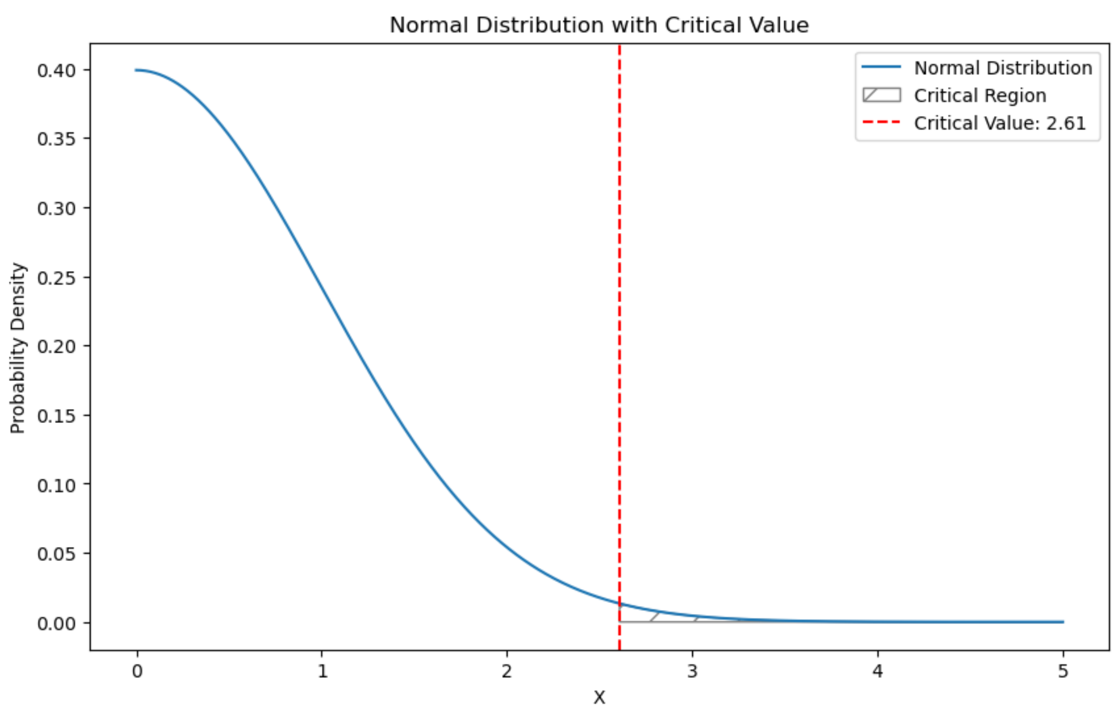
[34]:
```
alpha = 0.05  # Significant Value
cr = f.ppf(1-alpha,dfn=3,dfd=10886-3)  #dfn = 4 groups - 1 group and dfd =␣
↪total group - dfn
kruskal_stat, p_value = kruskal(
                        df[df["weather"]==1]["count"],
                        df[df["weather"]==2]["count"],
                        df[df["weather"]==3]["count"],
                        df[df["weather"]==4]["count"],
                        )
if p_value<alpha:
    print("Reject Null Hypothesis")
else:
    print("Fail to reject Null Hypothesis")
print("Test Statistic Value: ",kruskal_stat)
print("P_value:",p_value)
print("Critical Value: ", cr)
```

```
Reject Null Hypothesis
Test Statistic Value:  205.00216514479087
P_value: 3.501611300708679e-44
Critical Value:  2.605725028634713
```

[35]:
```
# From the above test, we can accept alternate hypothesis, because our p_value␣
↪is very lower than significance level
# So from kruskal hypothetical test we found that the data of weather makes a␣
↪great impact on cycles rented
# The mean of each group is varies from another group level in count of cycles␣
↪rented
```

[36]:
```
mu = 0
sigma = 1
```

```
# Calculate the critical value using the inverse survival function (ppf)
alpha = 0.05  # significance level
crit_value = f.ppf(1-alpha,dfn=3,dfd=10886-3)
# Generate some data to plot the normal distribution
x = np.linspace(0, 5, 1000)
y = norm.pdf(x, loc=mu, scale=sigma)


fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(x, y, label='Normal Distribution')
ax.fill_between(x, 0, y, where=x>=crit_value, hatch='/', edgecolor='gray', facecolor='none', label='Critical Region')



ax.axvline(x=crit_value, color='r', linestyle='--', label=f'Critical Value: {crit_value:.2f}')

# Add labels and legend to the plot
ax.set_xlabel('X')
ax.set_ylabel('Probability Density')
ax.set_title('Normal Distribution with Critical Value')
ax.legend()

plt.show()
```



[37]:
```
df.groupby("weather")["count"].mean()
```

[37]:
```
weather
1    205.236791
2    178.955540
3    118.846333
4    164.000000
Name: count, dtype: float64
```

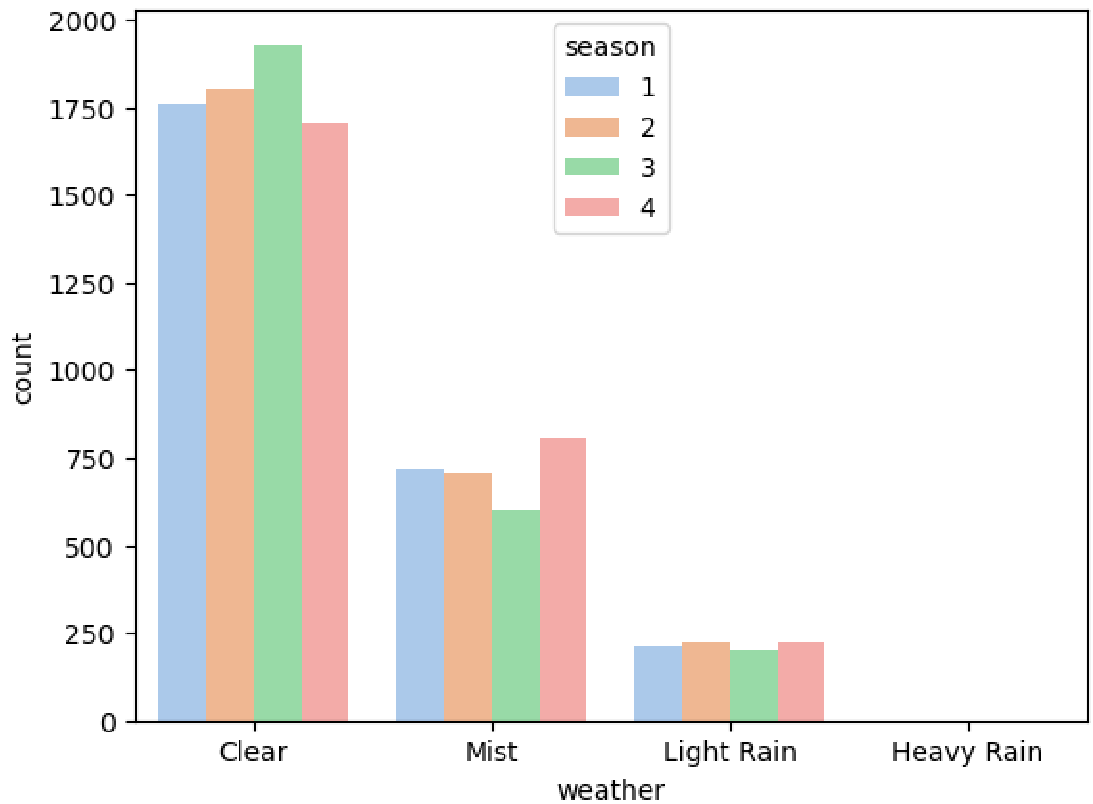
0.0.1 Hypothetical testing between weather and season

[38]:
```
df["weather"].value_counts(),df["season"].value_counts()
```

[38]:
```
(1    7192
 2    2834
 3     859
 4       1
 Name: weather, dtype: int64,
 4    2734
 2    2733
 3    2733
 1    2686
 Name: season, dtype: int64)
```

[39]:
```
weather_labels = {1: "Clear", 2: "Mist", 3: "Light Rain", 4: "Heavy Rain"}
sns.countplot(data=df,x="weather", hue="season",palette="pastel")

plt.xticks(ticks=[0, 1, 2, 3], labels=weather_labels.values())
```

[39]:
```
([<matplotlib.axis.XTick at 0x2388f3e9880>,
  <matplotlib.axis.XTick at 0x2388f3e9850>,
  <matplotlib.axis.XTick at 0x2388f3d6f40>,
  <matplotlib.axis.XTick at 0x2388f446280>],
 [Text(0, 0, 'Clear'),
  Text(1, 0, 'Mist'),
  Text(2, 0, 'Light Rain'),
  Text(3, 0, 'Heavy Rain')])
```



[40]:
```
pd.crosstab(index=df["weather"],columns=df["season"],margins=True)
```

[40]:

season	1	2	3	4	All
weather					
1	1759	1801	1930	1702	7192
2	715	708	604	807	2834
3	211	224	199	225	859
4	1	0	0	0	1
All	2686	2733	2733	2734	10886

[41]:
```
# Weather Characteristics
#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

# season:
# 1: spring,
# 2: summer,
# 3: fall,
```

```
# 4: winter

# _______________________________________________________________
# Analysis from data
# On an average cycles rented in clear day is greater than other weathers and␣
 ↪seasons
# Does weather impact season? yes from data we can see weather  makes impact on␣
 ↪season
# Lets prove statistically
```

[42]:
```
# ChiSquare test
# Assumption for Chisquare:
# ----------------------

# The data is categorical: Yes
# The observations are independent: Yes
# The expected frequencies are greater than 5: Yes
# The sample size is large: Yes

# Both weather and season are categorical values
# observation are totally independent
# In our data we won't consider heavy rain parameter, why beacause it doesn't␣
 ↪have enough sample to prove. Hence we will ignore that
#␣
 ↪___________________________________________________________________________________________________

# H0 : There is no association between the weather and season,
# Ha : There is a significant association between them.
# Significant Value: 0.05
# Critical Value: 16.918977604620448
```

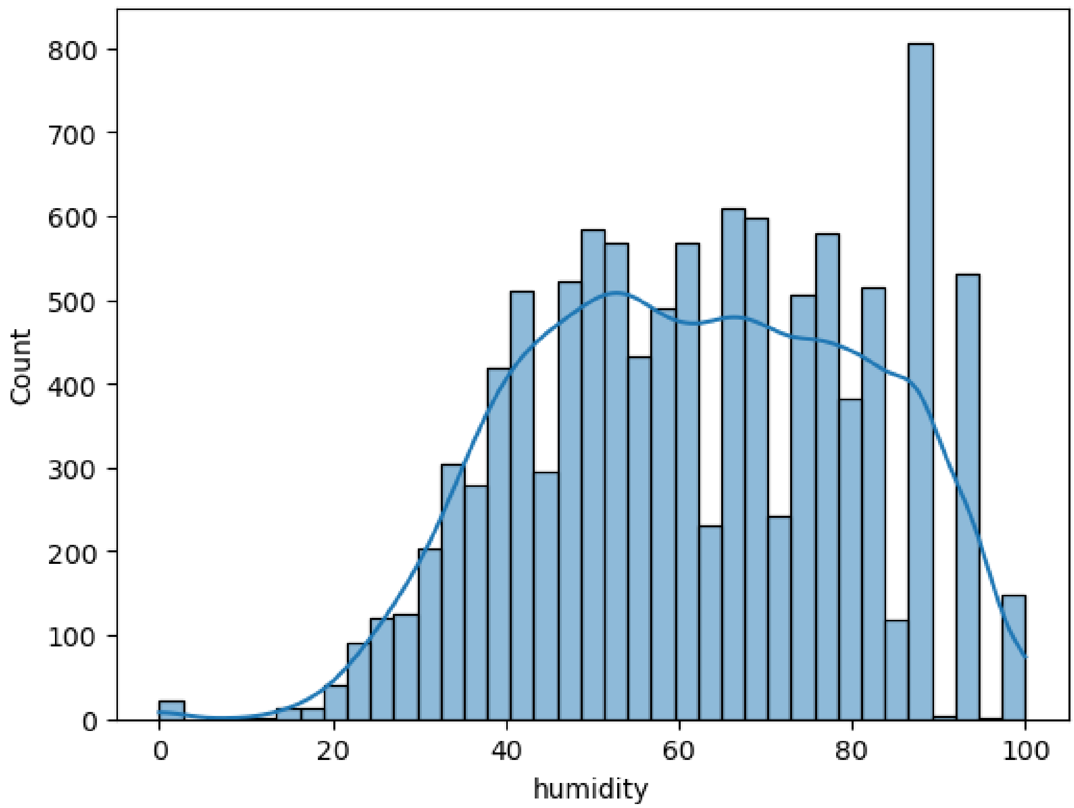
[43]:
```
alpha = 0.05  # Significant Value
cr = chi2.ppf(1-alpha,df=9)  #dfn = (4-1)*(4-1)
chi_stat, p_value,dof,exp_freq = chi2_contingency(pd.crosstab(df[df["weather"]!
 ↪=4]["weather"],df["season"]))
if p_value<alpha:
    print("Reject Null Hypothesis")
else:
    print("Fail to reject Null Hypothesis")
print("Test Statistic Value: ",chi_stat)
print("P_value:",p_value)
print("Critical Value: ", cr)
print("Expected Values: ", exp_freq)
```

```
Reject Null Hypothesis
Test Statistic Value:  46.101457310732485
P_value: 2.8260014509929403e-08
```

```
Critical Value:  16.918977604620448
Expected Values:  [[1774.04869086 1805.76352779 1805.76352779 1806.42425356]
 [ 699.06201194  711.55920992  711.55920992  711.81956821]
 [ 211.8892972   215.67726229  215.67726229  215.75617823]]
```

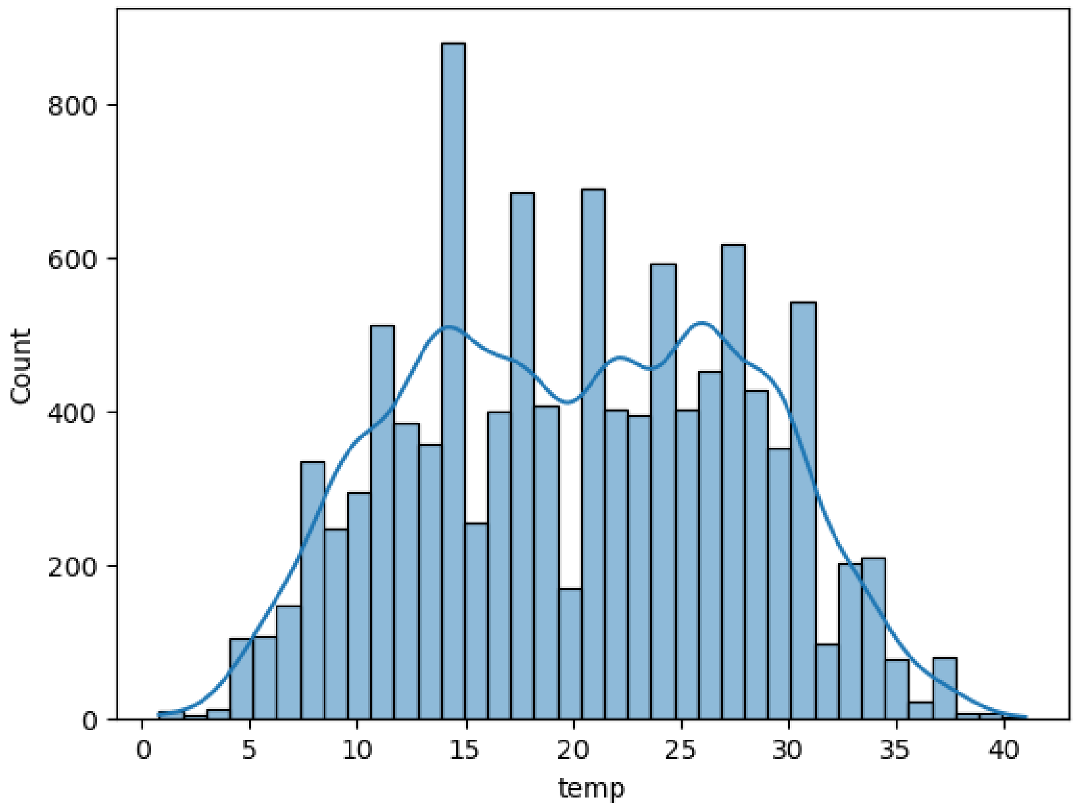
[44]:
```
sns.histplot(data=df,x="humidity",kde=True)
```

[44]: <AxesSubplot:xlabel='humidity', ylabel='Count'>



[45]:
```
sns.histplot(data=df,x="temp",kde=True)
```

[45]: <AxesSubplot:xlabel='temp', ylabel='Count'>


Count
temp
0
200
400
600
800
0
5
10
15
20
25
30
35
40

[ ]: