Business Case: Yulu – Hypothesis Testing

Nikhil K A

SCALER DSML

Business Case: Yulu – Hypothesis Testing

Yulu has recently suffered considerable dips in its revenues. They have contracted a consulting company to understand the factors on which the demand for these shared electric cycles depends. Specifically, they want to understand the factors affecting the demand for these shared electric cycles in the Indian market.

The company wants to know:

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

How well those variables describe the electric cycle demands?

1. Defining Problem Statement and perform Exploratory Data Analysis.

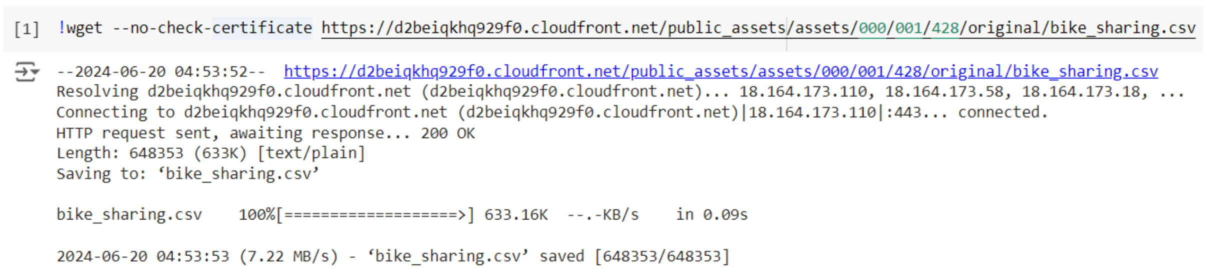
Definition of problem

Yulu is India's leading micro-mobility service provider, which offers unique vehicles for the daily commute. Starting off as a mission to eliminate traffic congestion in India, Yulu provides the safest commute solution through a user-friendly mobile app to enable shared, solo and sustainable commuting. Yulu zones are located at all the appropriate locations (including metro stations, bus stands, office spaces, residential areas, corporate offices, etc) to make those first and last miles smooth, affordable, and convenient. The business case envisages analysing of the given Yulu dataset and using different methods like visual and non-visual analysis and statistical analysis and formulate insights which will help Yulu in decision making. The business case leverages Python's robust data analytics and visualization capabilities to extract valuable insights from the data set, purchasing patterns, and product performance metrics. By harnessing Python libraries such as Pandas, NumPy, SciPy, Matplotlib and Seaborn, the case aims to gain a comprehensive understanding of user preferences, market trends, and market dynamics. Through data-driven analysis and visualization techniques, the case tries to optimize content recommendations to cater to diverse customer segments. This data-centric approach empowers Yulu to make informed decisions, drive customer retention and growth, and maintain a leading position in the ever-evolving retail industry landscape.

The data set have the following columns:

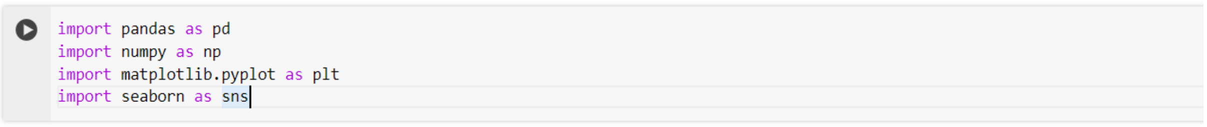
1	datetime	:	datetime
2	season	:	season (1: spring, 2: summer, 3: fall, 4: winter)
3	holiday	:	whether day is a holiday or not (extracted from http://dchr.dc.gov/page/holiday-schedule)
4	workingday	:	if day is neither weekend nor holiday is 1, otherwise is 0.
5	weather	:	1: Clear, Few clouds, partly 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
6	temp	:	temperature in Celsius
7	atemp	:	feeling temperature in Celsius
8	humidity	:	humidity
9	windspeed	:	wind speed
10	casual	:	count of casual users
11	registered	:	count of registered users
12	count	:	count of total rental bikes including both casual and registered

The data set is downloaded as 'bike_sharing.csv' and saved as dataframe named 'df'.

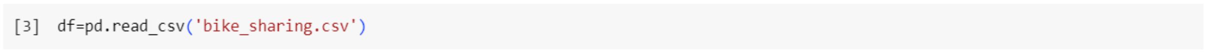


```
[1] !wget --no-check-certificate https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv
```

```
--2024-06-20 04:53:52--  https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)... 18.164.173.110, 18.164.173.58, 18.164.173.18, ...
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)|18.164.173.110|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 648353 (633K) [text/plain]
Saving to: 'bike_sharing.csv'

bike_sharing.csv    100%[===================>] 633.16K  --.-KB/s    in 0.09s

2024-06-20 04:53:53 (7.22 MB/s) - 'bike_sharing.csv' saved [648353/648353]
```



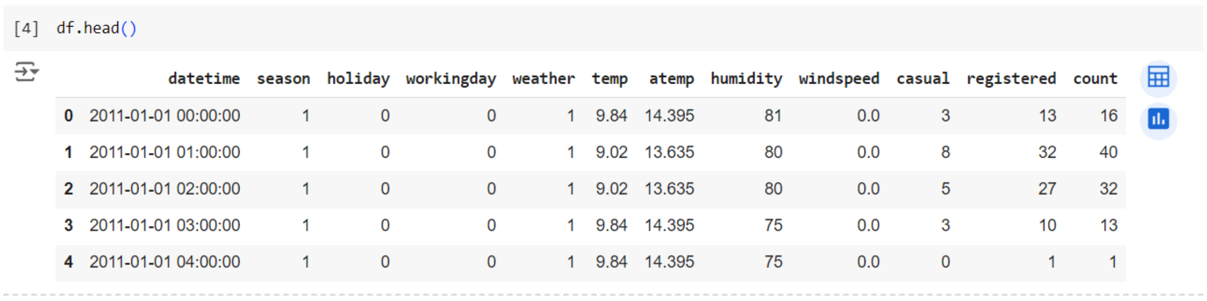
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```



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

Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), statistical summary

The data sample is observed by df.head()

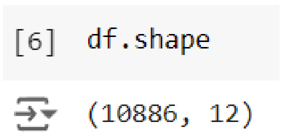


```
[4] df.head()
```

	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

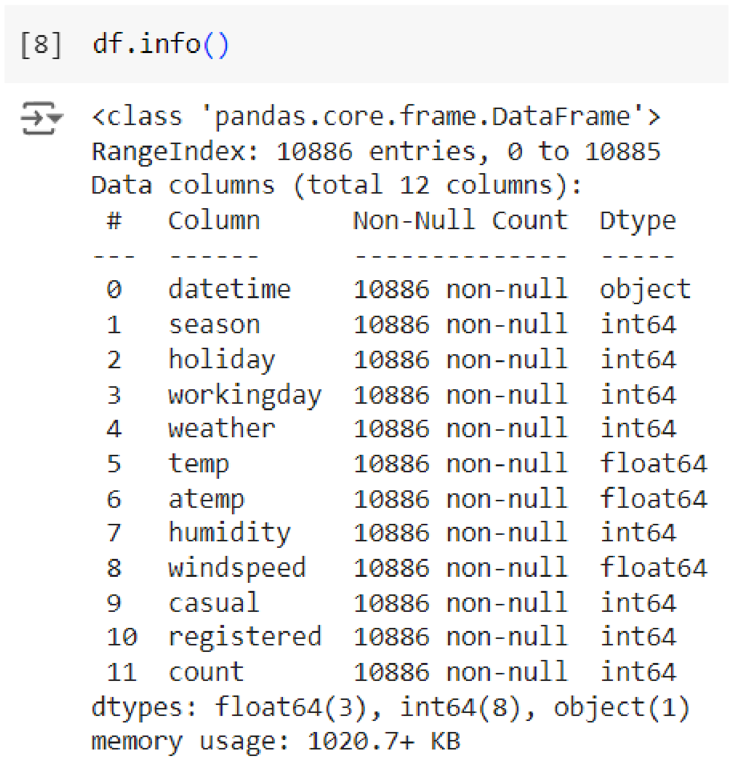
The data is divided into 12 columns and there are 10886 rows in the dataset.

Shape of the dataframe : df.shape showed



```
[6] df.shape

(10886, 12)
```

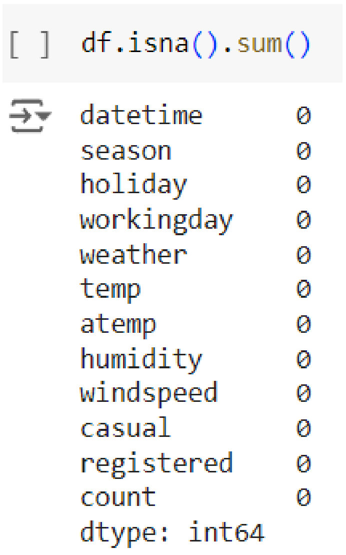
The basic information about dataframe. df.info()



```
[8] 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
```

Data type of 1 of the 12 columns are object type, three are of float64 type and remaining columns being int64.

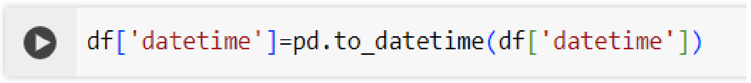
Detecting missing values by isna().



```
[ ]  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
```

It shows there are no null values or missing values in the dataset.

The datetime column in the dataset is of object datatype. The data is converted into datetime type by using pd.to_datetime.



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

Value counts and unique attributes.

The dataset consists of data of customers from first January 2011 to nineteenth December 2012.

```
[18] df['date']=df['datetime'].dt.date
```

```
[19] df['date'].min(),df['date'].max()
```

```
(datetime.date(2011, 1, 1), datetime.date(2012, 12, 19))
```

The dataset shows there are data regarding 456 days.

```
[21] df['date'].nunique()
```

```
456
```

Since the data contains entries of same date, drop duplicate method is used to create a new dataframe with unique entries of dates.

The 456 days are equally divided into four seasons.

```
[22] df1=df.drop_duplicates(subset='date')
```

```
[26] df1['season'].value_counts()
```

```
season
1    114
2    114
3    114
4    114
Name: count, dtype: int64
```

The data shows that among the 456 days, 311 are working days and 145 are either holidays or weekends.

```
[27] df1['workingday'].value_counts()
```

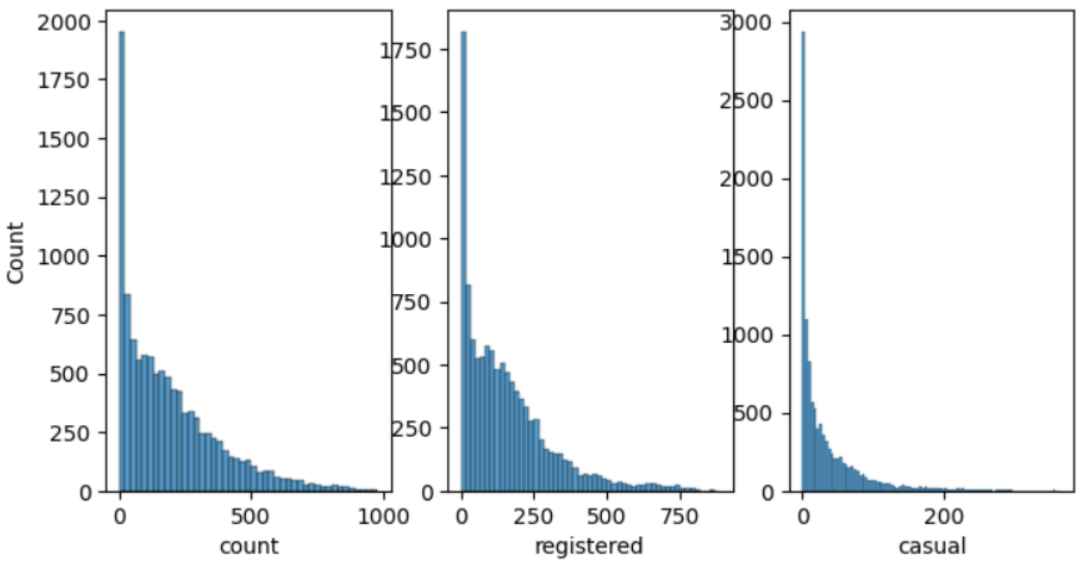
```
workingday
1    311
0    145
Name: count, dtype: int64
```

Visual Analysis - Univariate & Bivariate

For continuous variable(s): Distplot, countplot, histogram for univariate analysis

The variation of count of total rentals, registered users and casual rentals are ploted using histplot. The variation shows an exponential decrease with increase in number of rentals for all the three cases.

```
[59] plt.figure(figsize=(8,4))
     plt.subplot(1,3,1)
     sns.histplot(df['count'])
     ax=plt.subplot(1,3,2)
     sns.histplot(df['registered'],ax=ax)
     ax.set(ylabel='')
     ax=plt.subplot(1,3,3)
     sns.histplot(df['casual'],ax=ax)
     ax.set(ylabel='')
     plt.show()
```

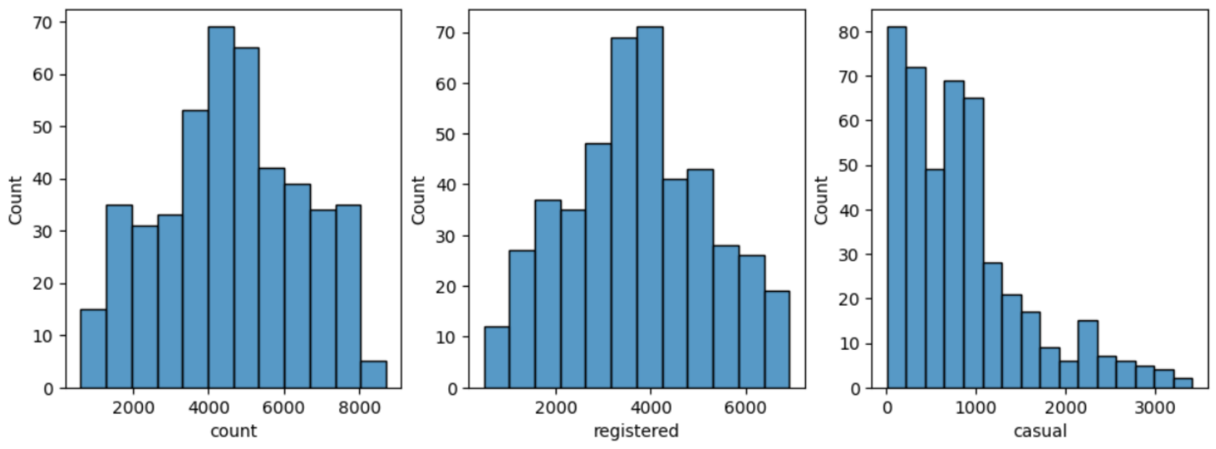


Here, the entries are for each hour. A dataframe is created by grouping the entries of each date using groupby method.

```
df1=df.groupby(['date','season','holiday','workingday']).aggregate({'casual':sum,'registered':sum,'count':sum})
#grouped by date, retaining the data about season, holiday and workingday columns and sum of hiring counts
df1.reset_index(inplace=True) #reseting the index from date and multilevel column indexing
```

The count of rentals per day is plotted using histplot.

```
[62] plt.figure(figsize=(12,4))
     plt.subplot(1,3,1)
     sns.histplot(df1['count'])
     plt.subplot(1,3,2)
     sns.histplot(df1['registered'])
     plt.subplot(1,3,3)
     sns.histplot(df1['casual'])
     plt.show()
```



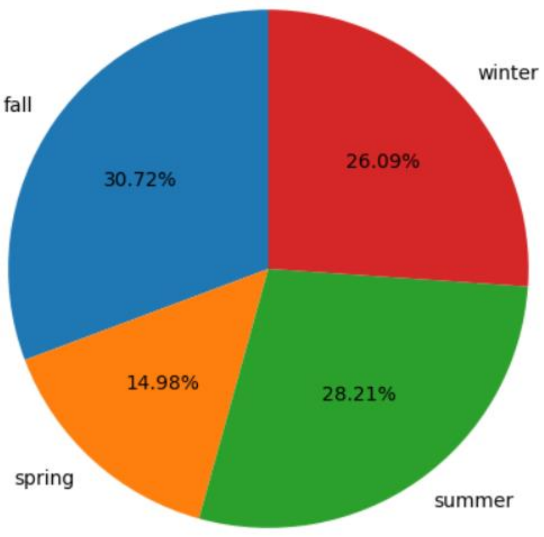
Bivariate Analysis (Relationships between important variables such as workday and count, season and count, weather and count.

The seasons has been provided by numeric codes. Apply function is used to substitute them with the names of seasons.

```
[7]  def season (s):
       if(s==1):
         return 'spring'
       elif(s==2):
         return 'summer'
       elif(s==3):
         return 'fall'
       else:
         return 'winter'
     df['seasonw']=df['season'].apply(season)
```

The following represents the distribution of the number of rentals across seasons

```
[15]  season_count=df.groupby('seasonw')['count'].sum()
      plt.figure(figsize = (6,6))
      plt.pie(season_count,
              labels=season_count.index,
              startangle=90,
              autopct = '%.2f%%')
      plt.show()
```

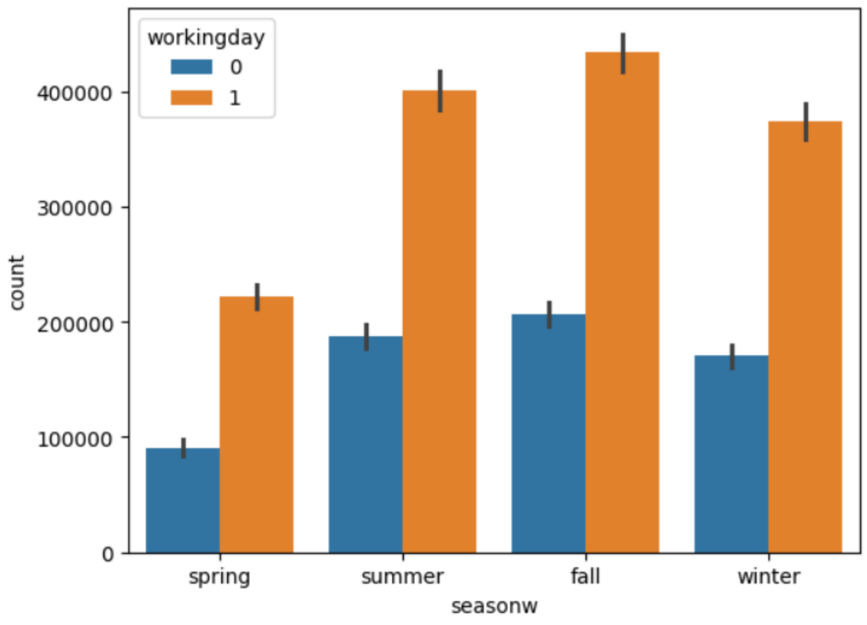


Fall season contributes to greatest number of rentals among all seasons. Spring is surprisingly the least.

Distribution of number of rentals among seasons and working/holidays.

```
sns.barplot(data=df,x='seasonw',y='count',hue='workingday',estimator='sum')
```

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

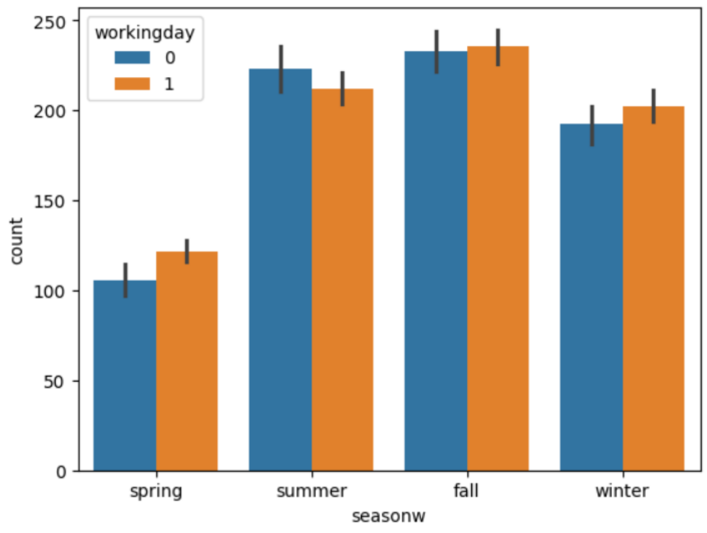


In all four seasons, holidays provide for only half of the rentals taken during working days. This is because the number of non-working days are less.

When observed for mean value,

```
[16] sns.barplot(data=df,x='seasonw',y='count',hue='workingday',estimator='mean')
```

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

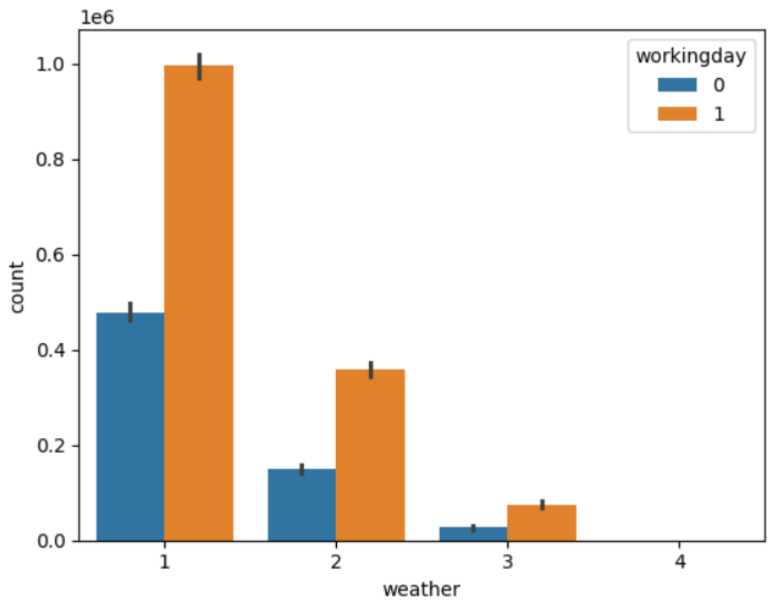


It is observed that the mean rental count of working and non-working days are more or less same in all four seasons.

The distribution of count of rentals across weather conditions.

```
[24] sns.barplot(x='weather',hue='workingday',y='count',data=df,estimator='sum')
```

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

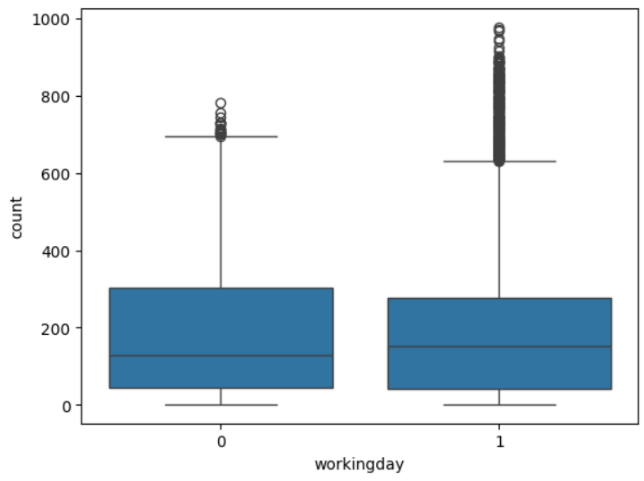


The ratio of count of rentals during working days and holidays remains same. Most number of rentals being during 1: Clear, Few clouds, partly cloudy, partly cloudy weather. Followed by 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist. And heavy rain being the least contributing.

Boxplot has been plotted using count and workingday,

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

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



The count of rentals during working days being slightly high, but with a large number of outliers in the upper side limit.

Illustrate the insights based on EDA

Comments on range of attributes, outliers of various attributes

- The data contains details of yulu bikes rented from first January 2011 to nineteenth December 2012.
- The dataset shows there are data regarding 456 days.
- The data is divided into 12 columns and there are 10886 rows in the dataset.
- The data shows the count of registered and casual users of yulu bikes over the above-mentioned period over different seasons and weather conditions.
- The counts of rentals over working days, or seasons or weather conditions have shown outliers on the upper end.

Comments on the distribution of the variables and relationship between them

- The major part of rentals is during working days since in the time period, more days were working days.
- The average rental counts of working and non-working days are close.
- The season which saw greatest number of rentals is fall and the least was observed in spring.
- The number of rentals is found to be reducing from clear weather condition having greatest number of rentals to heavy rain weather having lowest.

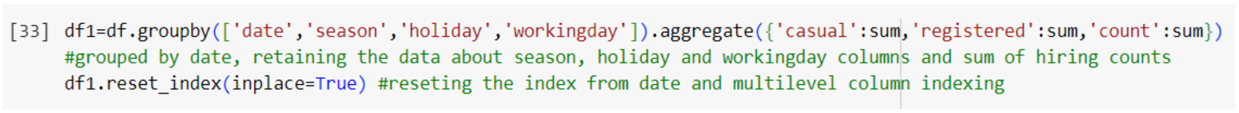
Comments for each univariate and bivariate plots

- The distribution of count of customers show variation depending on the status of the day, (working day or not), season and weather conditions.
- When considering the sum of total rentals, working days are largest contributors but when analysed on the mean number of rentals per day, there wasn't much difference between working days and non-working days.
- The total number of rentals consist of registered customers and casual customers combined. Among these the number of casual customers are found to be considerably small while registered customers play the major role.
- Fall and summer seasons are found to be providing positive atmosphere for bike rentals.
- Considering weather conditions, clear weather is contributes to larger number of bike rentals where heavy rains show a very small number of rentals over the time period.

2. Hypothesis Testing (30 Points):

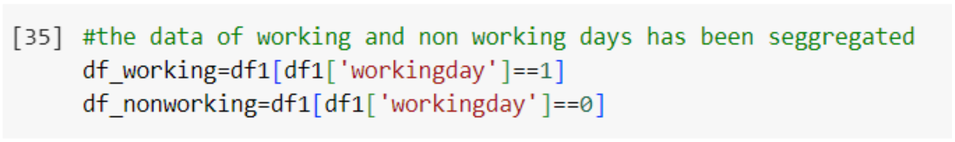
i. 2- Sample T-Test to check if Working Day has an effect on the number of electric cycles rented

Firstly, the data given are hourly based. For getting day based data, the dataframe is grouped by the date parameter.



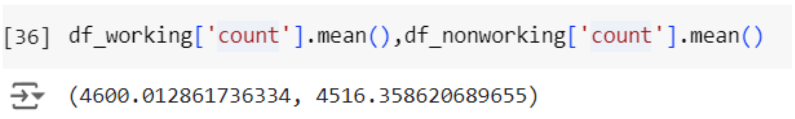
```
[33] df1=df.groupby(['date','season','holiday','workingday']).aggregate({'casual':sum,'registered':sum,'count':sum})
     #grouped by date, retaining the data about season, holiday and workingday columns and sum of hiring counts
     df1.reset_index(inplace=True) #reseting the index from date and multilevel column indexing
```

The data regarding the rentals during working days and non-working days has been separated into two sets of data.



```
[35] #the data of working and non working days has been seggregated
     df_working=df1[df1['workingday']==1]
     df_nonworking=df1[df1['workingday']==0]
```

Now, the mean value for count of rentals is obtained.



```
[36] df_working['count'].mean(),df_nonworking['count'].mean()

(4600.012861736334, 4516.358620689655)
```

The mean value of rentals of working day and nonworking day are different with mean of working day rentals being higher.

Establishing hypothesis:

H_0 = There is no effect of working day on the number of electric cycles rented.

H_a = There is an effect of working day on the number of electric cycles rented.

In other words,

H_0=The mean value of counts of rentals in working day and non-working day are equal, i.e., $\mu_w = \mu_n$

where,

μ_w = The mean value of count of rentals during working days

μ_n = The mean value of count of rentals during non-working days

H_a = The mean value of counts of rentals in working day and non-working day are not equal, i.e., $\mu_w \neq \mu_n$

The distribution of count of non working and working days are plotted using histplot.

```
[39] sns.histplot(df_working['count'],bins=50)
```

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

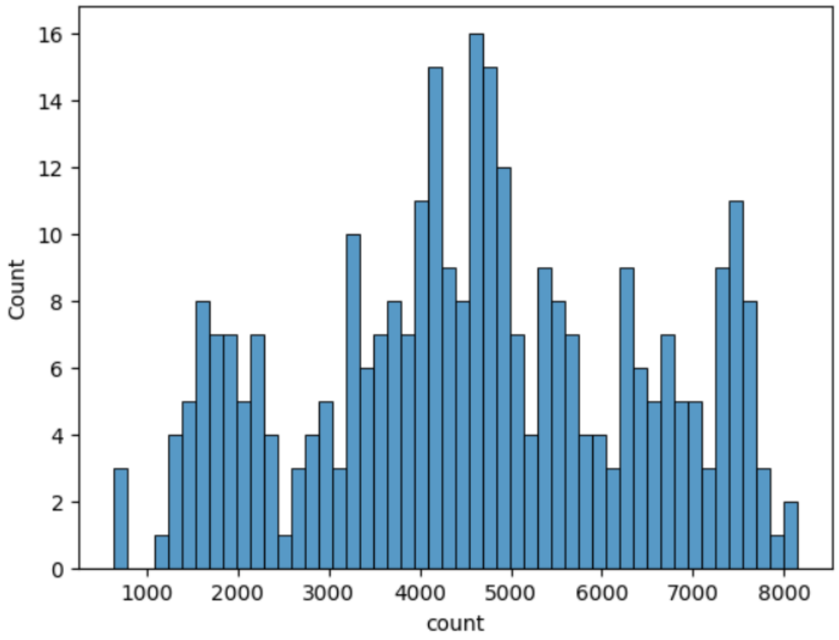


Figure 1 Histogram of count of rental on working days

```
[42] sns.histplot(df_nonworking['count'],bins=50)
```

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

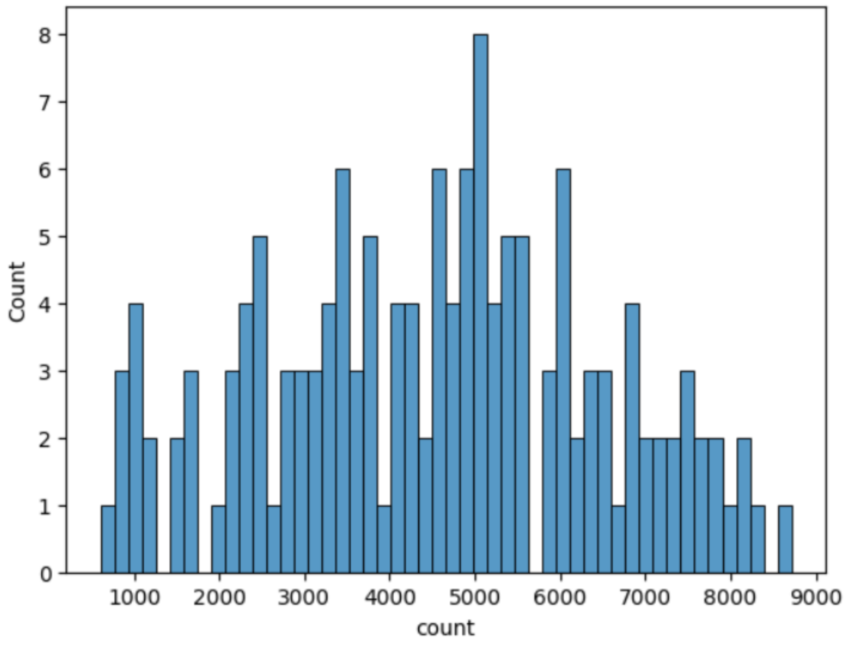


Figure 2 Histogram of count of rental on non-working days

Since the distribution is not exactly Gaussian and the population standard deviation is not known, T-test will be appropriate for hypothesis testing. The test is for inequality and hence, two-sample t-test will be appropriate.

```
[1]  from scipy.stats import ttest_ind
```

'ttest_ind' is imported from scipy.stats library since the two sets of data are independent.

```
[41] ttest_ind(df_working['count'],df_nonworking['count'])

     TtestResult(statistic=0.4447722161488l995, pvalue=0.656696335987859, df=454.0)
```

The test statistics and p-value are obtained. Assuming a confidence level of 95%, i.e., α =0.05, p-value greater than α. Hence,

Failed to reject null hypothesis.

Working day has no effect on the number of electric bikes rented.

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

1. <u>Weather</u>

Here, the daily basis data cannot be taken since the weather may change during the same day.

The data has been segregated based on weather.

```
[44] #segregating the count based on weather
     df_weather1=df[df['weather']==1]
     df_weather2=df[df['weather']==2]
     df_weather3=df[df['weather']==3]
     df_weather4=df[df['weather']==4]
```

Since the data is of Numerical-Categorical combination with four different categories,
ANOVA test is suggested.

Establishing hypothesis:

H_0 = There is no effect of weather on the number of electric cycles rented.
H_a = There is an effect of weather on the number of electric cycles rented.

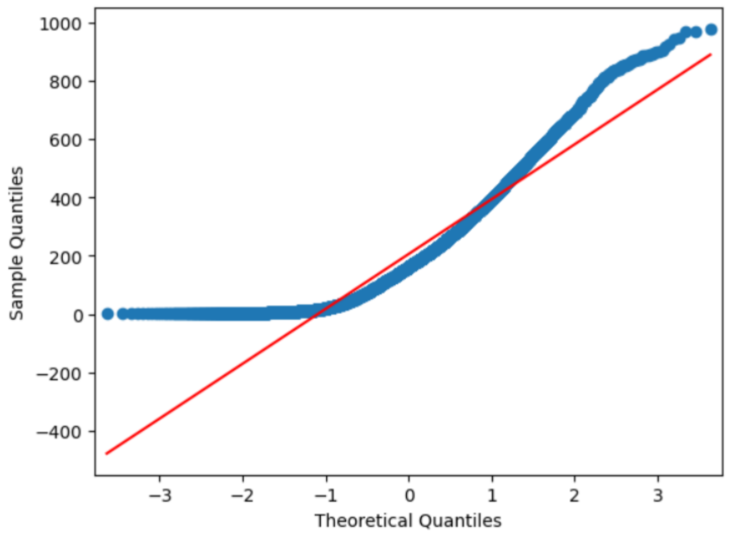
Assumptions for ANOVA.

- The data should be following normal distribution.
- The data should be independent across each record.
- Equal variance in each group.

Testing normalcy by qq plot.

```
[45] #Assumptions in ANOVA
     from statsmodels.graphics.gofplots import qqplot
```

```
plt.figure(figsize=(12,4))
qqplot(df_weather1['count'],line='s')
```



For double checking, Shapiros test is used.

```
[51] from scipy.stats import shapiro
```

```
[53] countsub=df_weather1['count'].sample(200)
     shapiro(countsub)
```

```
ShapiroResult(statistic=0.8852612376213074, pvalue=3.1881906836783713e-11)
```

```
[55] countsub2=df_weather2['count'].sample(200)
     shapiro(countsub2)
```

```
ShapiroResult(statistic=0.8723453283309937, pvalue=6.005850764628784e-12)
```

```
[56] countsub3=df_weather3['count'].sample(200)
     shapiro(countsub3)
```

```
ShapiroResult(statistic=0.7585265040397644, pvalue=8.141081148500356e-17)
```

All the shapiros tests gave a very small value of p-value indicating that all the counts follow normal distribution except the fourth. The values corresponding to weather 4 is not enough to check for normalcy.

Levenes test for equal variance.

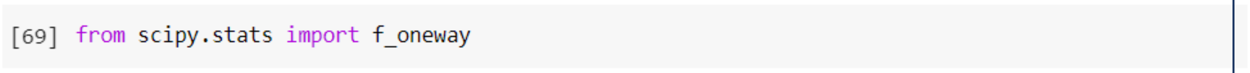
```
[66] from scipy.stats import levene
```

```
[68] levene(df_weather1['count'],df_weather2['count'],df_weather3['count'],df_weather4['count'])
```

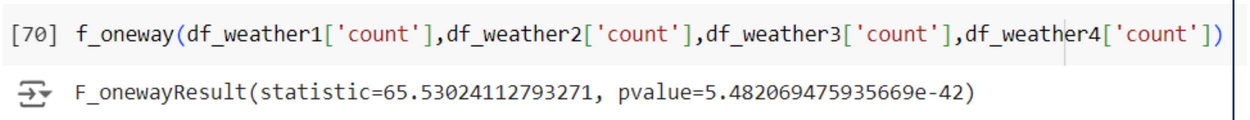
```
LeveneResult(statistic=54.85106195954556, pvalue=3.504937946833238e-35)
```

Levenes test gave p-value very less than alpha indicating that the count values are having equal variance.

ANOVA across different weather groups.



```
[69] from scipy.stats import f_oneway
```



```
[70] f_oneway(df_weather1['count'],df_weather2['count'],df_weather3['count'],df_weather4['count'])

F_onewayResult(statistic=65.53024112793271, pvalue=5.482069475935669e-42)
```

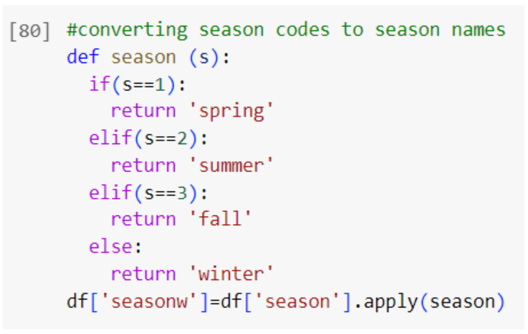
Here, p-value < α assuming a confidence level of 95%. i.e., α =0.05

Hence, H_0 is rejected.

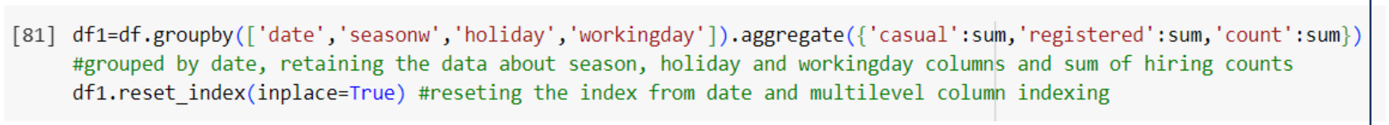
There is some effect of weather on the number of electric cycles rented.

2.<u>Season</u>

Here, the daily basis data is taken since the season won't change during the same day.

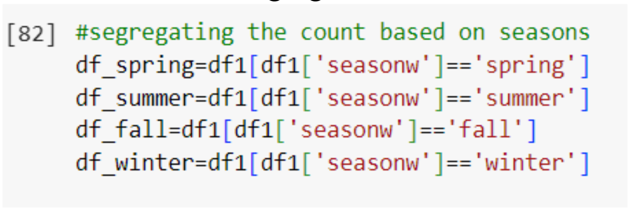


```
[80] #converting season codes to season names
     def season (s):
       if(s==1):
         return 'spring'
       elif(s==2):
         return 'summer'
       elif(s==3):
         return 'fall'
       else:
         return 'winter'
     df['seasonw']=df['season'].apply(season)
```



```
[81] df1=df.groupby(['date','seasonw','holiday','workingday']).aggregate({'casual':sum,'registered':sum,'count':sum})
     #grouped by date, retaining the data about season, holiday and workingday columns and sum of hiring counts
     df1.reset_index(inplace=True) #reseting the index from date and multilevel column indexing
```

The data has been segregated based on season.



```
[82] #segregating the count based on seasons
     df_spring=df1[df1['seasonw']=='spring']
     df_summer=df1[df1['seasonw']=='summer']
     df_fall=df1[df1['seasonw']=='fall']
     df_winter=df1[df1['seasonw']=='winter']
```

Since the data is of Numerical-Categorical combination with four different categories,
ANOVA test is suggested.

Establishing hypothesis:

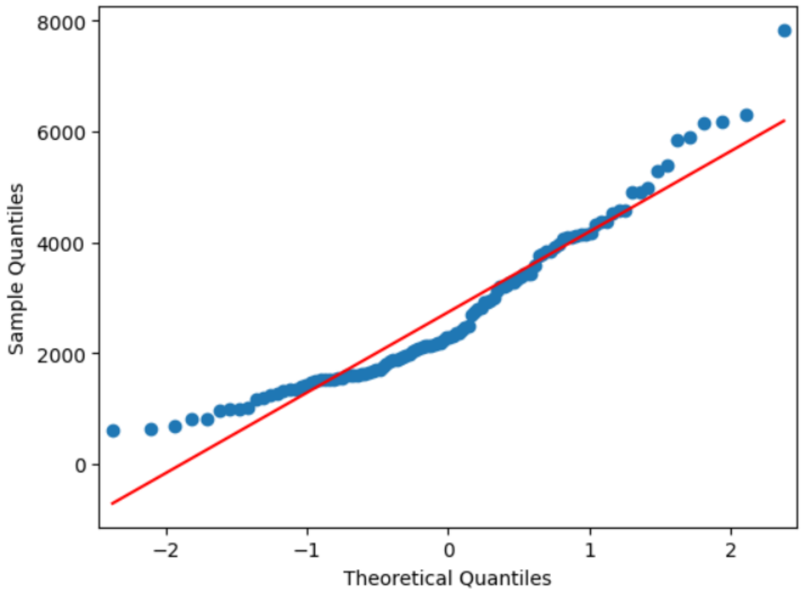
H_0 = There is no effect of season on the number of electric cycles rented.
H_a = There is an effect of season on the number of electric cycles rented.

Assumptions for ANOVA.

- The data should be following normal distribution.
- The data should be independent across each record.
- Equal variance in each group.

Testing normalcy by qq plot.

```
[83] #Assumptions in ANOVA
     from statsmodels.graphics.gofplots import qqplot
     qqplot(df_spring['count'],line='s')
```



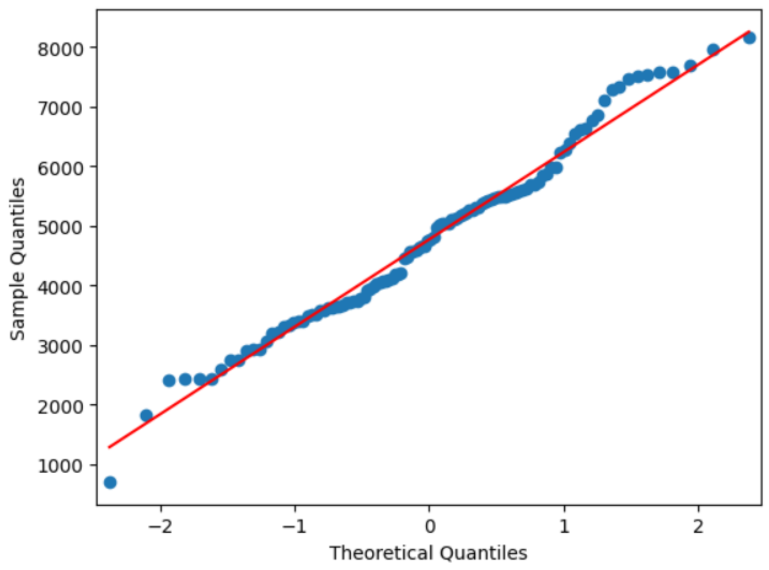
For double checking, Shapiros test is used.

```
[84] from scipy.stats import shapiro
     shapiro(df_spring['count'])

ShapiroResult(statistic=0.9294025301933289, pvalue=1.4321534763439558e-05)

[85] shapiro(df_summer['count'])

ShapiroResult(statistic=0.9752597212791443, pvalue=0.032791439443826675)

[86] shapiro(df_fall['count'])

ShapiroResult(statistic=0.9641115069389343, pvalue=0.003765953006222844)

[87] shapiro(df_winter['count'])

ShapiroResult(statistic=0.9835520386695862, pvalue=0.17639805376529694)
```

All the shapiros tests gave a very small value of p-value indicating that all the counts follow normal distribution except the fourth. The values corresponding to winter is not showing normalcy.

Re-checking winter data with qq plot

```
[88] qqplot(df_winter['count'],line='s')
```



It shows the winter data is almost normal

Levenes test for equal variance.

```
[89] from scipy.stats import levene
     levene(df_spring['count'],df_summer['count'],df_fall['count'],df_winter['count'])
```

```
LeveneResult(statistic=1.5071252673249398, pvalue=0.21194448921499898)
```

Levenes test gave p-value more than alpha indicating that the count values are not having equal variance.

Since all the assumptions for ANOVA is not found to be satisfied, Kruskal test is used. Kruskal test across different weather groups:

```
[93] from scipy.stats import kruskal
     kruskal(df_spring['count'],df_summer['count'],df_fall['count'],df_winter['count'])
```

```
KruskalResult(statistic=152.13785107938023, pvalue=9.111105591373848e-33)
```

Here, p-value << α assuming a confidence level of 95%. i.e., α =0.05

Hence, H_0 is rejected.

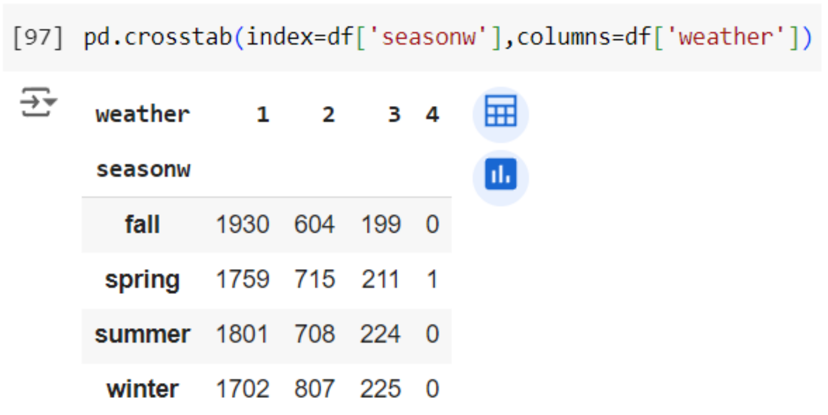
There is some effect of season on the number of electric cycles rented.

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

Establishing hypothesis

H_0 = There is no effect of season on weather.
H_a = There is an effect of season on weather.

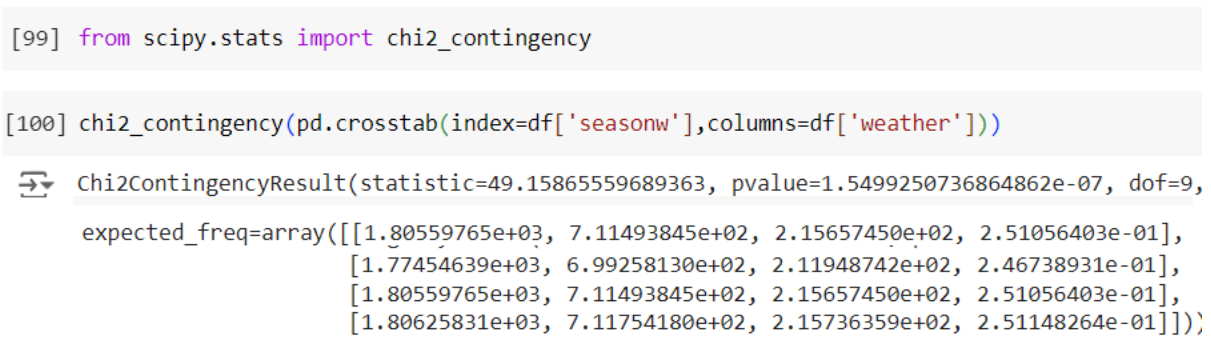
Cross tab is used to create matrix of weather and seasons.



```
[97] pd.crosstab(index=df['seasonw'],columns=df['weather'])
```

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

Now, chi2_contingency is used to do chisquare test on this crosstab.



```
[99] from scipy.stats import chi2_contingency
```

```
[100] chi2_contingency(pd.crosstab(index=df['seasonw'],columns=df['weather']))

Chi2ContingencyResult(statistic=49.15865559689363, pvalue=1.5499250736864862e-07, dof=9,
 expected_freq=array([[1.80559765e+03, 7.11493845e+02, 2.15657450e+02, 2.51056403e-01],
                      [1.77454639e+03, 6.99258130e+02, 2.11948742e+02, 2.46738931e-01],
                      [1.80559765e+03, 7.11493845e+02, 2.15657450e+02, 2.51056403e-01],
                      [1.80625831e+03, 7.11754180e+02, 2.15736359e+02, 2.51148264e-01]]))
```

p-value = 1.55 e^{-7}. Assuming a confidence level of 95% i.e., α =0.05

p-value<< α and H_0 is rejected.

Weather is dependent on seasons.