Business Case: Yulu - Hypothesis Testing

[45]
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import ttest_1samp,ttest_ind,shapiro,levene,chi2_contingency
```

[2]
```
df = pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv?1642089089")
df
```

	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	0.0000	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...
10881	2012-12-19 19:00:00	4	0	1	1	15.58	19.695	50	26.0027	7	329	336
10882	2012-12-19 20:00:00	4	0	1	1	14.76	17.425	57	15.0013	10	231	241
10883	2012-12-19 21:00:00	4	0	1	1	13.94	15.910	61	15.0013	4	164	168
10884	2012-12-19 22:00:00	4	0	1	1	13.94	17.425	61	6.0032	12	117	129
10885	2012-12-19 23:00:00	4	0	1	1	13.12	16.665	66	8.9981	4	84	88

10886 rows × 12 columns

Basic Analysis

Shape of Data

[ ]
```
print("Rows in the data: ",df.shape[0])
print("Columns in the data: ",df.shape[1])
```

```
Rows in the data:  10886
Columns in the data:  12
```

First 5 rows

[ ]
```
df.head(5)
```

	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

Columns in DataFrame

```
[ ] df.columns
```

```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count'],
      dtype='object')
```

Datatype of all Attributes

```
[ ] df.dtypes
```

```
datetime       object
season          int64
holiday         int64
workingday      int64
weather         int64
temp          float64
atemp         float64
humidity        int64
windspeed     float64
casual          int64
registered      int64
count           int64
dtype: object
```

```
[ ] 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
```

```
[ ] # Missing Values
    df.isnull().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
```

There are no missing values present in dataset.

Statistical Summary

Statistical Summary of Numeric columns

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

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191.574132
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.144454
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.000000
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.000000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.000000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.000000

Non-Graphical Analysis: Value counts and unique attributes

Unique Values

```
[ ] df.nunique()
```

```
datetime      10886
season            4
holiday           2
workingday        2
weather           4
temp             49
atemp            60
humidity         89
windspeed        28
casual          309
registered      731
count           822
dtype: int64
```

Value Counts

```
[ ] df.season.value_counts()
```

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

```
[ ] df.holiday.value_counts()
```

```
0    10575
1      311
Name: holiday, dtype: int64
```

Data-type conversion

Converting datetime column to type datetime from object type

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

```
0       2011-01-01 00:00:00
1       2011-01-01 01:00:00
2       2011-01-01 02:00:00
3       2011-01-01 03:00:00
4       2011-01-01 04:00:00
                ...
10881   2012-12-19 19:00:00
10882   2012-12-19 20:00:00
10883   2012-12-19 21:00:00
10884   2012-12-19 22:00:00
10885   2012-12-19 23:00:00
Name: datetime, Length: 10886, dtype: datetime64[ns]
```

Converting season column from numerical to categorical

```
[ ] def season_type(x):
        if x==1:
            return 'spring'
        elif x==2:
            return 'summer'
        elif x==3:
            return 'fall'
        else:
            return 'winter'
```

```
[ ] df['season']=df['season'].apply(lambda x:season_type(x))
    df['season']
```

```
0        spring
1        spring
2        spring
3        spring
4        spring
          ...
10881    winter
10882    winter
10883    winter
10884    winter
10885    winter
Name: season, Length: 10886, dtype: object
```

Converting season,weather,holiday and workingday columns into categorical

```
[3] df['season']= pd.Categorical(df['season'])
    df['weather']=pd.Categorical(df['weather'])
    df['holiday']=pd.Categorical(df['holiday'])
    df['workingday']=pd.Categorical(df['workingday'])
```

```
[ ] df[['datetime','season','weather','holiday','workingday']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    10886 non-null  datetime64[ns]
 1   season      10886 non-null  category
 2   weather     10886 non-null  category
 3   holiday     10886 non-null  category
 4   workingday  10886 non-null  category
dtypes: category(4), datetime64[ns](1)
memory usage: 128.3 KB
```

Statistical Summary after data-type conversion

```
#Statistical summary of  numeric columns
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

```
#Statistical summary of  categorical columns

df.describe(include='category')
```

	season	holiday	workingday	weather
count	10886	10886	10886	10886
unique	4	2	2	4
top	winter	0	1	1
freq	2734	10575	7412	7192

Univariate Analysis

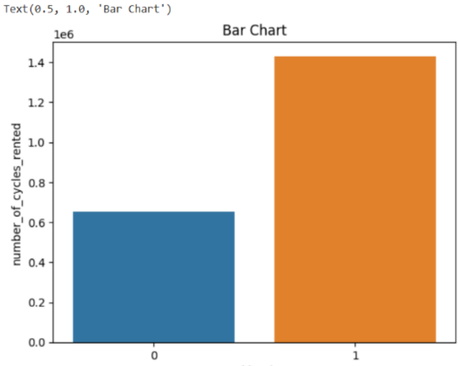
Distribution of Working day

```
workingday_df=df.groupby(['workingday']).agg(number_of_cycles_rented=('count','sum')).reset_index()
workingday_df
```

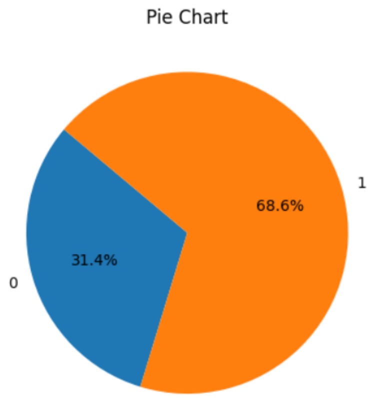
	workingday	number_of_cycles_rented
0	0	654872
1	1	1430604

```
labels= workingday_df['workingday']
values= workingday_df['number_of_cycles_rented']
sns.barplot(x="workingday",y="number_of_cycles_rented",data=workingday_df)
plt.title("Bar Chart")
```

```
Text(0.5, 1.0, 'Bar Chart')
```



```
[ ] sizes = workingday_df["number_of_cycles_rented"]
    labels = workingday_df["workingday"]

    # Create a pie chart
    plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)

    # Add a title
    plt.title("Pie Chart")

    # Display the pie chart
    plt.show()
```



---

observation:

On working days, 68.6% of cycles are rented, whereas on non-working days, 31.4% of cycles are rented.

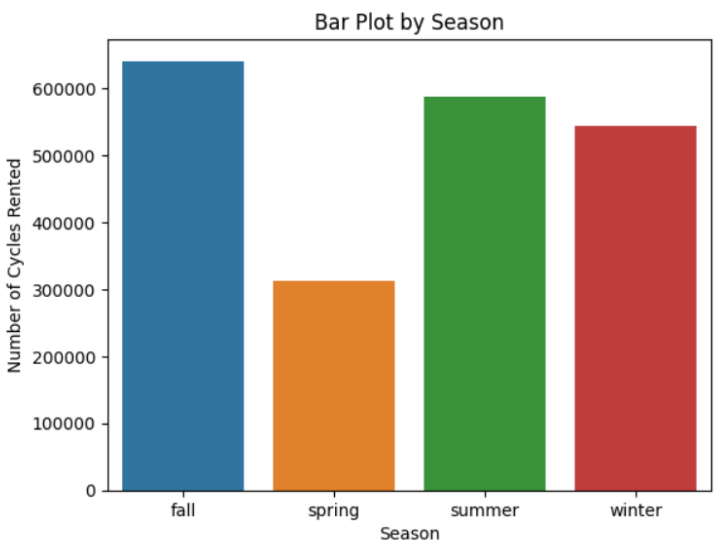
Distribution of Season

season_df=df.groupby(['season']).agg(number_of_cycles_rented=('count','sum')).reset_index() season_df

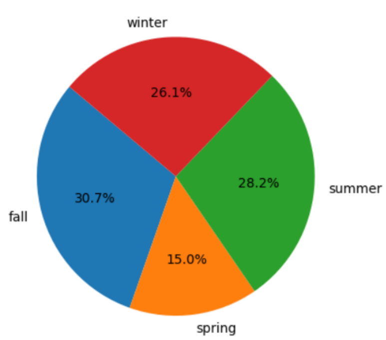
```
[ ] season_df=df.groupby(['season']).agg(number_of_cycles_rented=('count','sum')).reset_index()
    season_df
```

	season	number_of_cycles_rented
0	fall	640662
1	spring	312498
2	summer	588282
3	winter	544034

```
[ ] # Define the data to be plotted
    labels = season_df['season']
    values = season_df['number_of_cycles_rented']

    # Create a bar plot using Seaborn
    sns.barplot(x=labels, y=values)

    # Optionally, add labels and a title
    plt.xlabel("Season")
    plt.ylabel("Number of Cycles Rented")
    plt.title("Bar Plot by Season")

    # Show the plot
    plt.show()
```




```
[ ] sizes = season_df["number_of_cycles_rented"]
    labels = season_df["season"]
    plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
    plt.show()
```



observations:

- During the fall season, approximately 30.7% of cycles are rented.
- In the summer season, around 28.2% of cycles are rented.
- The winter season records a rental rate of about 26.1% for cycles.
- The lowest rental rate, at just 15%, is observed in the spring season.

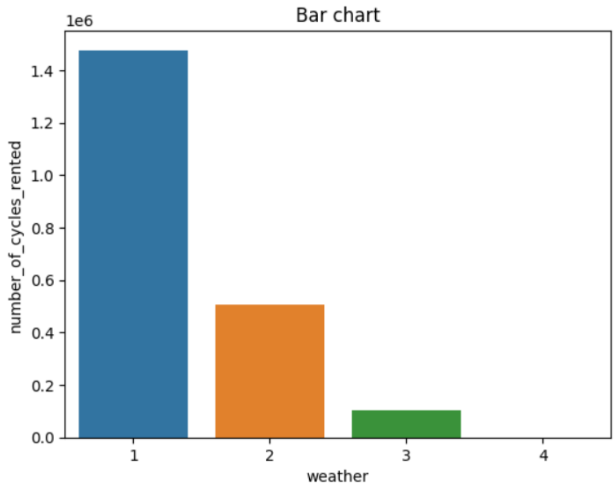
Distribution of Weather

[4]
```
weather_df=df.groupby(['weather']).agg(number_of_cycles_rented=('count','sum')).reset_index()
weather_df
```

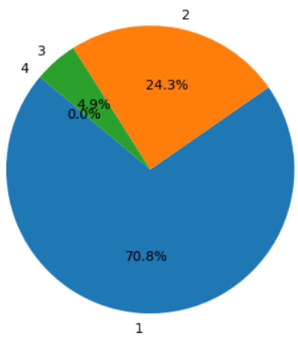
	weather	number_of_cycles_rented
0	1	1476063
1	2	507160
2	3	102089
3	4	164

[7]
```
labels= weather_df['weather']
values= weather_df['number_of_cycles_rented']
sns.barplot(x=labels,y=values,data=weather_df)
plt.title("Bar chart")
```

```
Text(0.5, 1.0, 'Bar chart')
```



[14]
```
# Define the data to be plotted
values = weather_df['number_of_cycles_rented']
labels = weather_df['weather']

# Create a pie chart
plt.pie(values, labels=labels,autopct='%1.1f%%', startangle=140)

# Add a title
plt.title("Pie Chart")

# Display the pie chart
plt.show()
```

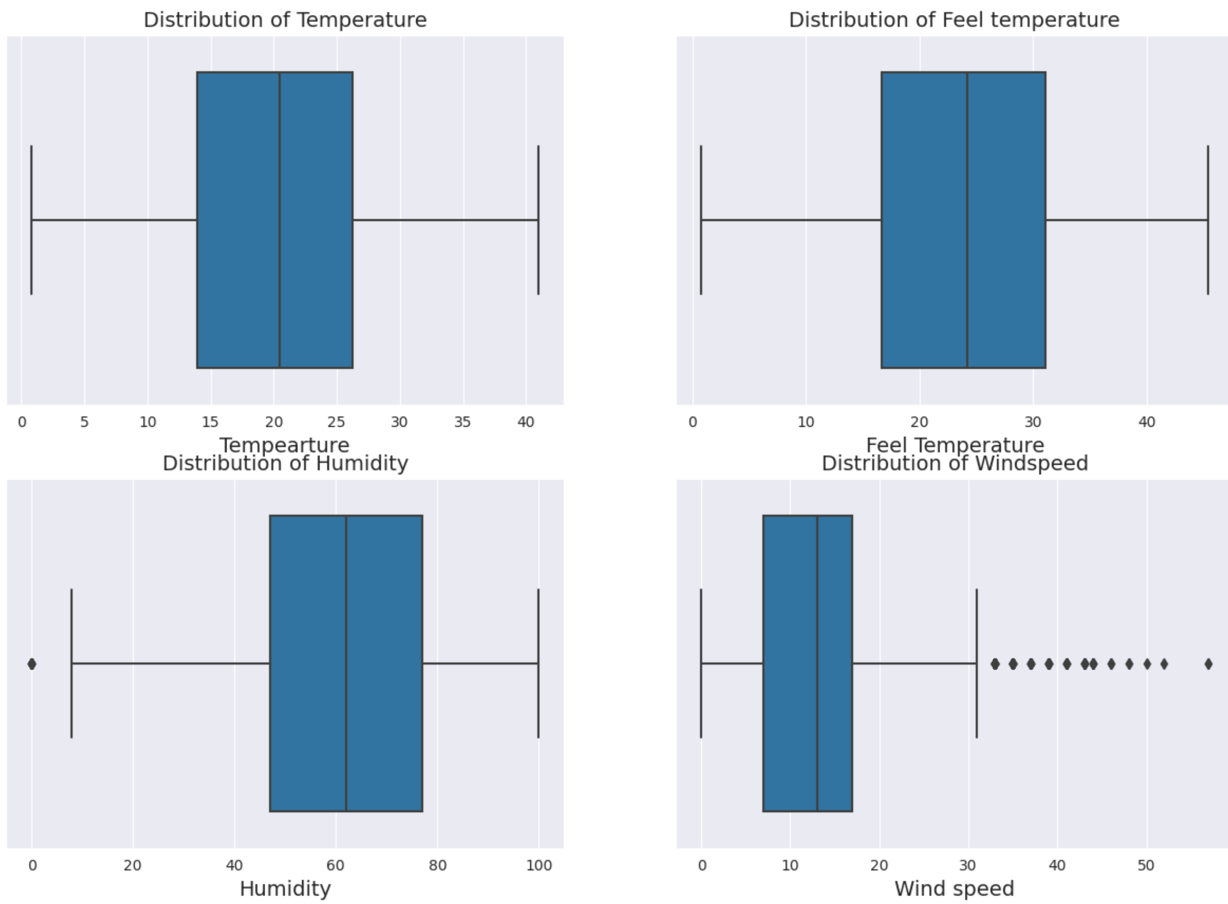


observations:

- Weather condition 1 experiences the highest rental rate, with approximately 70.8% of cycles rented.
- In weather condition 2, around 24.3% of cycles are rented.
- Weather condition 3 has a rental rate of approximately 4.9% for cycles.
- Weather condition 4 exhibits an exceptionally low rental rate, with only 0.00786% of cycles being rented.

Distribution of temp, atemp, humidity and windspeed

```
plt.figure(figsize=(15,10))

# Boxplot for temp column
plt.subplot(2,2,1)
sns.boxplot(data=df,x='temp')
plt.xlabel('Tempearture',fontsize=14)
plt.title('Distribution of Temperature',fontsize=14)

#Boxplot for feel temperature
plt.subplot(2,2,2)
sns.boxplot(data=df,x='atemp')
plt.xlabel('Feel Temperature',fontsize=14)
plt.title('Distribution of Feel temperature',fontsize=14)

#Boxplot for Humidity
plt.subplot(2,2,3)
sns.boxplot(data=df,x='humidity')
plt.xlabel('Humidity',fontsize=14)
plt.title('Distribution of Humidity',fontsize=14)

#Boxplot for Wind Speed
plt.subplot(2,2,4)
sns.boxplot(data=df,x='windspeed',)
plt.xlabel('Wind speed',fontsize=14)
plt.title('Distribution of Windspeed',fontsize=14)


plt.show()
```



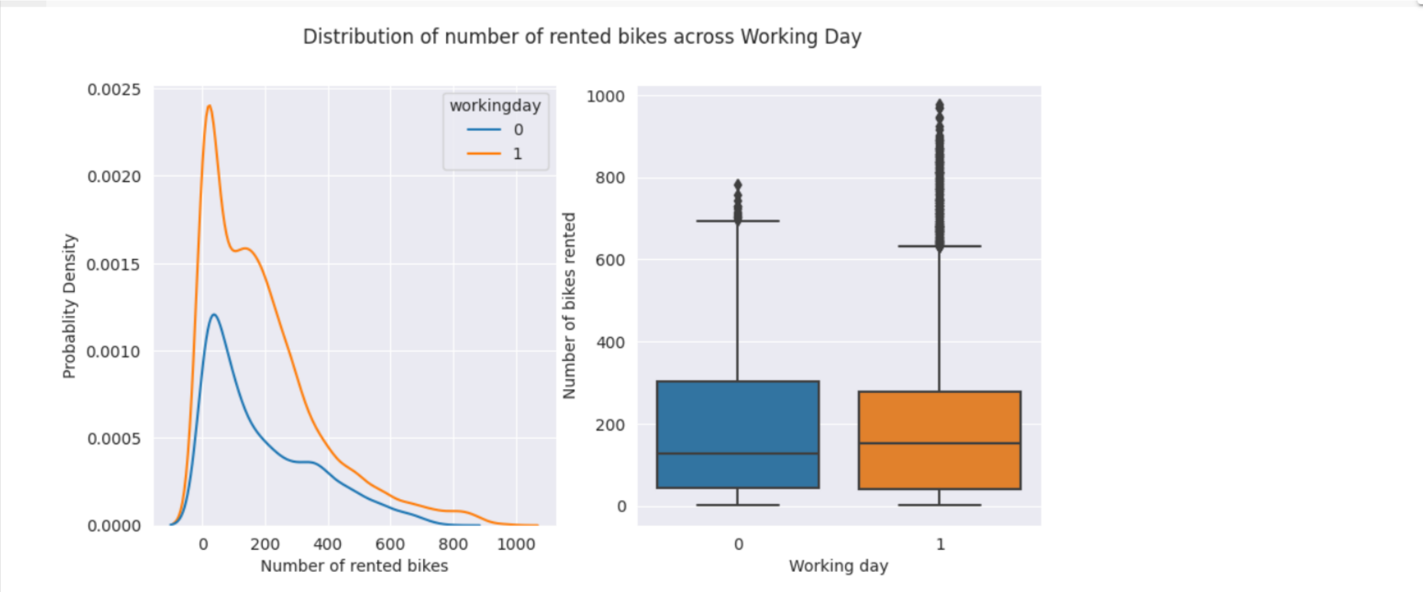
observation:

- No outliers are detected in the 'temp' and 'atemp' columns, suggesting that the temperature-related data points fall within the expected range.
- In the 'humidity' column, a single value is identified as an outlier, implying an unusual humidity measurement distinct from the others.
- The 'windspeed' column contains 12 outlier values, indicating instances where wind speed measurements significantly deviate from the typical range.

Bivariate Analysis

Distribution of count of rented bikes across working day

```
plt.figure(figsize=(10,5))
# KDE plot
plt.subplot(1,2,1)
sns.kdeplot(data=df,x='count',hue='workingday')
plt.xlabel('Number of rented bikes')
plt.ylabel('Probablity Density')

# Box plot
plt.subplot(1,2,2)
sns.boxplot(data=df,y='count',x='workingday',)
plt.xlabel('Working day')
plt.ylabel('Number of bikes rented')

plt.suptitle('Distribution of number of rented bikes across Working Day')
plt.show()
```

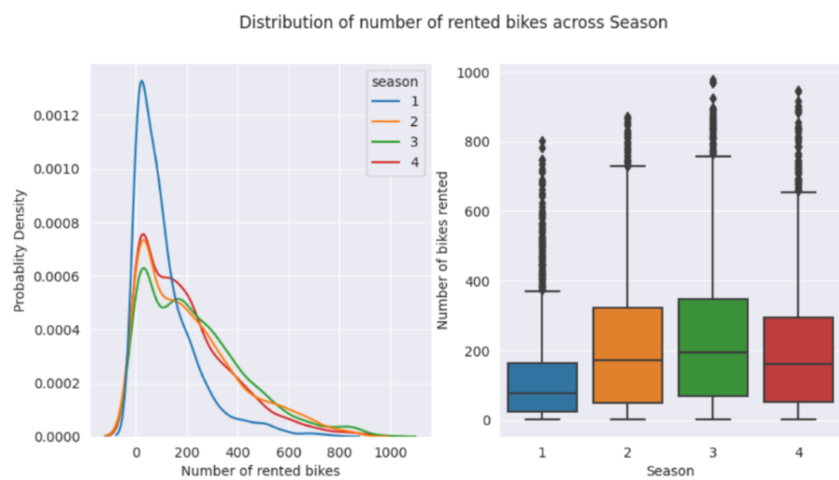


observation:

The probability of renting bikes on a working day appears to be higher than on a non-working day, as evidenced by our univariate analysis, where 68.6% of bike rentals occurred on working days compared to 31.4% on non-working days. However, we will further investigate this through hypothesis testing to determine if the working day indeed has a statistically significant effect on the number of cycles rented."

Distribution of count of rented bikes across Season

```
[23] plt.figure(figsize=(10,5))
     # KDE plot
     plt.subplot(1,2,1)
     sns.kdeplot(data=df,x='count',hue='season')
     plt.xlabel('Number of rented bikes')
     plt.ylabel('Probablity Density')

     # Box plot
     plt.subplot(1,2,2)
     sns.boxplot(data=df,y='count',x='season')
     plt.xlabel('Season')
     plt.ylabel('Number of bikes rented')

     plt.suptitle('Distribution of number of rented bikes across Season')
     plt.show()
```



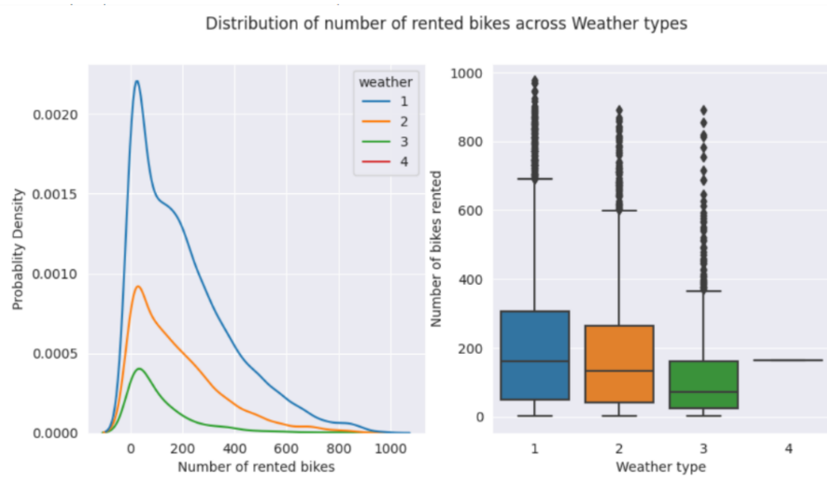
observation:

The probability of renting a bike during the fall season appears to be higher compared to other seasons. Conversely, the probability of renting bikes during the winter and spring seasons is lower in comparison to summer and fall. To investigate this further, we will conduct an ANOVA test to determine if the season has a statistically significant effect on bike rentals.

Distribution of count of rented bikes across Weather types

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

     # KDE plot
     plt.subplot(1,2,1)
     sns.kdeplot(data=df,x='count',hue='weather')
     plt.xlabel('Number of rented bikes')
     plt.ylabel('Probablity Density')

     # Box plot
     plt.subplot(1,2,2)
     sns.boxplot(data=df,y='count',x='weather')
     plt.xlabel('Weather type')
     plt.ylabel('Number of bikes rented')

     plt.suptitle('Distribution of number of rented bikes across Weather types')
     plt.show()
```

```
<ipython-input-24-eacc1dd2cde0>:5: UserWarning: Dataset has 0 variance; skipping density estimate. Pass `warn_singular=False` to disable this warning.
  sns.kdeplot(data=df,x='count',hue='weather')
```



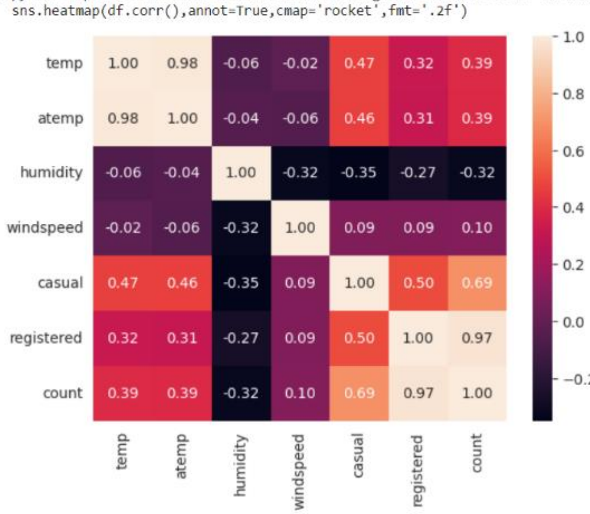
observation:

The probability of renting a bike during weather condition 1 appears to be higher than in other weather types. This is supported by our univariate analysis, where approximately 70.8% of bike rentals occur in weather condition 1, while the remaining weather types collectively account for approximately 29% of bike rentals. However, we will further investigate this by conducting an ANOVA test to establish whether weather type indeed has a statistically significant effect on the number of bikes rented.

Heatmap and Correlation

```
[25] sns.heatmap(df.corr(),annot=True,cmap='rocket',fmt='.2f')
     plt.show()
```

```
<ipython-input-25-c2e8d88a43ba>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to Fal
  sns.heatmap(df.corr(),annot=True,cmap='rocket',fmt='.2f')
```



observation:

* The weak positive correlation of 0.39 between temperature and the number of

bikes rented suggests that, on average, fewer people prefer to use electric cycles during the daytime between 12 PM to 3 PM. This observation aligns with our univariate analysis, where we discovered that the average number of cycles rented during this time frame was lower compared to other times of the day. A similar correlation pattern is also observed in the case of "feels-like" temperature, reinforcing this trend.

* The negative correlation between humidity and the number of cycles rented indicates that people tend to avoid using electric bikes during high humidity conditions. This avoidance can be attributed to the discomfort caused by the heavy and sticky air, leading to sweating and a general sense of unease. Moreover, the reduced efficiency of electric bikes in high humidity, resulting in increased air resistance and potential battery performance issues, contributes to the preference for alternative transportation or indoor activities in such conditions.

* The presence of a weak positive correlation between windspeed and the number of cycles rented indicates that there is a subset of individuals who appear to favor using electric cycles during windy conditions for the sheer enjoyment of the experience. While this preference contributes to a slight increase in bike rentals on windier days, it's essential to recognize that this effect is not particularly strong, as indicated by the weak correlation. This suggests that the enjoyment of cycling in windy conditions is a relatively niche preference among riders.

Hypothesis Testing

Does Working day has an effect on the number of bikes rented?

Formulating Null and Alternative Hypotheses

To answer the above question we first set up Null and Alternate Hypothesis:

- H0 : Working day does not have an effect on number of cycles rented
- Ha: Working day does have an effect on number of cycles rented

Solution: To test the above hypothesis, we use Two sample T Test

Assumptions of a T Test

- Independence : The observations in one sample are independent of the observations in the other sample.
- Normality : Both samples are approximately normally distributed.
- Homogenity of Variances : Both samples have approximately the same variance.
- Random Sampling : Both samples were obtained using random sampling method

Normality Check: Wilkin Shapiron Test

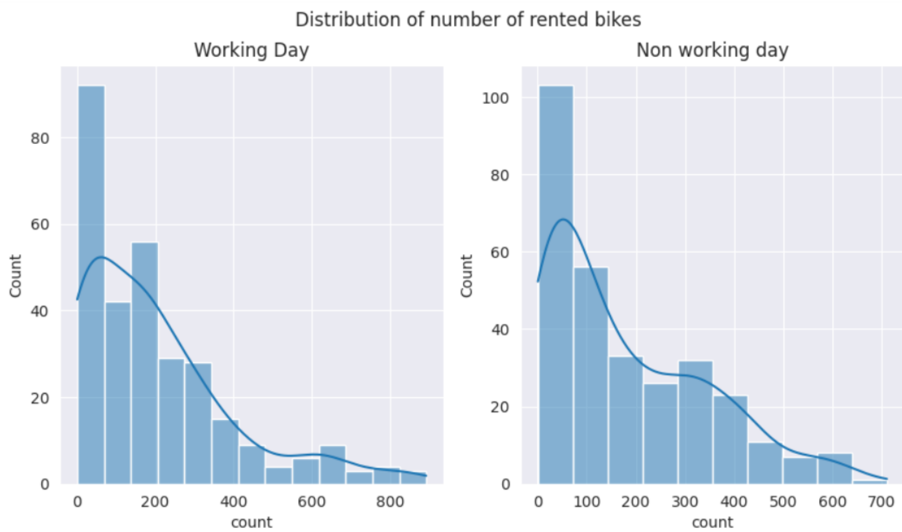
- To conduct the above experiment, we shall take the samples randomly, and also the number of electric cyles rented on Working day and non working day are independent.
- We however have to check for Normality and homogenity of Variances

Generate a sample of 300 bike rentals, randomly selected from both working days and non-working days

```
[26] workingday_sample=df[df['workingday']==1]['count'].sample(300)
     nonworkingday_sample=df[df['workingday']==0]['count'].sample(300)
```

To check normality we can use histogram

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

#histogram for working day sample
plt.subplot(1,2,1)
sns.histplot(workingday_sample,kde=True)
plt.title('Working Day')

#histogram for non working day sample
plt.subplot(1,2,2)
sns.histplot(nonworkingday_sample,kde=True)
plt.title('Non working day')

plt.suptitle('Distribution of number of rented bikes')
plt.show()
```

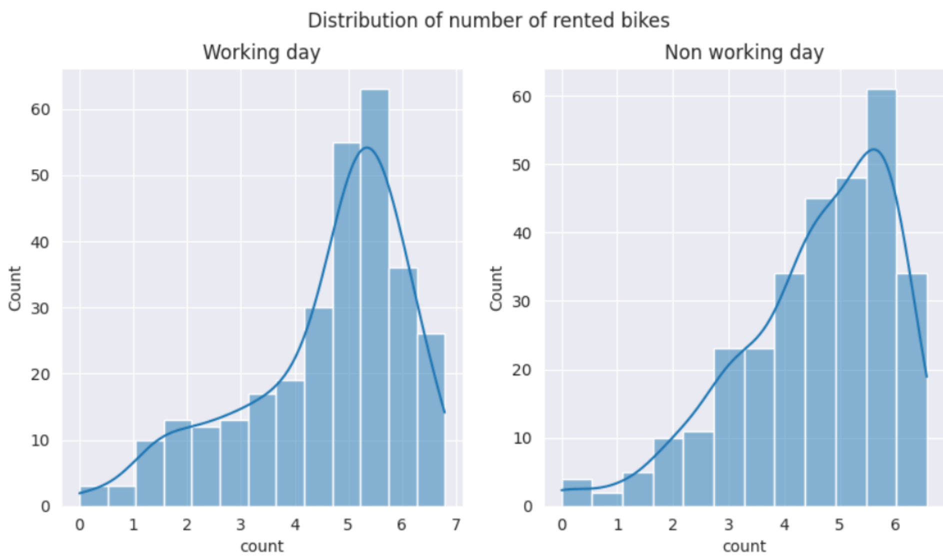


observation:

- The counts of rented cycles on both working and non-working days do not follow a normal distribution.
- We can try to convert the distribution to normal by applying log transformation

Converting sample distribution to normal by applying log transformation

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

     #histogram for working day sample
     plt.subplot(1,2,1)
     sns.histplot(np.log(workingday_sample),kde=True)
     plt.title('Working day')

     #histogram for non working day sample
     plt.subplot(1,2,2)
     sns.histplot(np.log(nonworkingday_sample),kde=True)
     plt.title('Non working day')

     plt.suptitle('Distribution of number of rented bikes')
     plt.show()
```



observation:

Upon implementing a log transformation on our continuous variables, we observed a substantial improvement in achieving a distribution that closely resembles normality for both the workingday_sample and nonworkingday_sample.

We will now conduct the Wilk-Shapiro Test to assess the normality of the log-normal distribution obtained in the previous step

Performing the Wilk-Shapiro test for the workingday sample

We select the level of signifiance as 5% and the null and alternate hypothesis is as follows:

- H0 : The Working day samples are normally distributed
- Ha: The Working day samples are not normally distributed

```
test_stat,p_value= shapiro(np.log(workingday_sample))
print("test stat :",test_stat)
print("p value :",p_value)
alpha = 0.05
if p_value< alpha:
 print("Reject Ho: The working day samples are not normally distributed ")
else:
 print("Fail to Reject Ho: The working day samples are normally distributed")
```

```
test stat : 0.9099098443984985
p value : 2.0253870195580115e-12
Reject Ho: The working day samples are not normally distributed
```

observation:

- From the above output, we see that the p value is far less than 0.05, Hence we reject the null hypothesis.
- We have sufficient evidence to say that the non working day sample data does not come from normal distribution.

Homegenity of Variance test : Levene's Test

We select the level of signifiance as 5% and the null and alternate hypothesis is as follows:

- H0 : Variance is equal in both working day count and non working day count samples
- Ha: Variances is not equal

```
[36] test_stat,p_value= levene(np.log(workingday_sample),np.log(nonworkingday_sample))
print("test stat :",test_stat)
print("p value :",p_value)
alpha = 0.05
if p_value< alpha:
 print("Reject Ho: Variance is not equal ")
else:
 print("Fail to Reject Ho: Variance is equal in both working day count and non working day count samples")
```

```
test stat : 0.2687846461786804
p value : 0.604339982940315
Fail to Reject Ho: Variance is equal in both working day count and non working day count samples
```

observation:

- Since pvalue is not less than 0.05, we fail to reject null hypothesis.
- This means we do not have sufficient evidence to say that variance across workingday count and non workingday count is significantly different thus making the assumption of homogenity of variances true

T-Test and final conclusion

- 3 out of 4 assumptions for T test has been satified.
- Although the sample distribution did not meet the criteria of passing the normality test, we proceed with the T-test as per the given instructions.

For T-Test we select the level of signifiance as 5% and the null and alternate hypothesis is as follows:

- H0 : Working day does not have an effect on number of cycles rented
- Ha: Working day does have an effect on number of cycles rented

```
[37] t_stat,p_value= ttest_ind(np.log(workingday_sample),np.log(nonworkingday_sample),equal_var=True)
print("f stat :",t_stat)
print("p value :",p_value)
alpha = 0.05
if p_value< alpha:
 print("Reject Ho: Working day does have an effect on number of cycles rented ")
else:
 print("Fail to Reject Ho: Working day does not have an effect on number of cycles rented")
```

```
f stat : 0.2668962988716935
p value : 0.7896410177551554
Fail to Reject Ho: Working day does not have an effect on number of cycles rented
```

Conclusion:

- Since the p-value of our test is greater than alpha which is 0.05, we fail to reject the null hypothesis of this test.
- we do not have sufficient evidence to conclude that working days have a significant effect on the number of cycles rented. This suggests that there is no significant difference in the number of cycles rented on working days versus non-working days.

Assumptions of Chi-Square Test

- Both variables are categorical:

In this dataset, season column has already been converted into categorical data and the weather column is nominal data. Hence it is safe to say that the above condition is satisfied.

- All observations are independent:

We are hoping that the sample provided by YULU has been obtained from random sampling upon which the condition is satisfied.

- Cells in the contingency table are mutually exclusive

Assuming each individual in the dataset was only surveyed once, this assumption should be met.

- Expected value of cells should be 5 or greater in at least 80% of cells and none less than 1

We shall check for this condition after the pearson chi-square test has been completed.

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

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

As previously mentioned, there is only one row in our dataset for weather type 4. We lack sufficient information to determine if it truly correlates with the season. To avoid potential biases and skewed results, it's advisable to exclude this rare weather type from our analysis.

```
[42] df_removed_weather=df[~(df['weather']==4)]
```

```
[43] data=pd.crosstab(df_removed_weather['weather'],df_removed_weather['season'])
data
```

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

```
[46] x_stat,p_value,dof,expected=chi2_contingency(data)
```

```
[47] expected.min()
```

```
211.88929719797886
```

```
[48] (len(expected[expected<5])/len(expected))*100
```

```
0.0
```

All of the data points have expected values greater than 5, indicating that the assumption related to the expected values being greater than 5 is satisfied for the chi-square test.

Chi-Square Test and Final Conclusion

We shall setup Null and alternate Hypotheis to check if Weather is dependent on season

- H0: Weather is not dependent on the season
- Ha: Weather is dependent on the season, meaning they are associated or related

We consider level of significance as 0.05

```
[49] x_stat,p_value,dof,expected=chi2_contingency(data)
     print("X stat :",x_stat)
     print("p value :",p_value)
     alpha = 0.05
     if p_value< alpha:
      print("Reject Ho: Weather is dependent on the season")
     else:
      print("Fail to Reject Ho: Weather is not dependent on the season")
```

```
X stat : 46.101457310732485
p value : 2.8260014509929403e-08
Reject Ho: Weather is dependent on the season
```

Final Conclusion

- Since the p-value obtained from our test is less than the predetermined alpha level of 0.05, we have sufficient evidence to reject the null hypothesis for this test.
- Indeed, this suggests that we have gathered enough evidence to conclude that there is a dependence between weather and the season.

Insights

1. On working days, 68.6% of cycles are rented, whereas on non-working days, 31.4% of cycles are rented.
2. Despite the fact that 68.6% of cycles are rented on working days compared to 31.4% on non-working days, our t-test analysis does not provide sufficient evidence to conclude that working days have a significant effect on the number of cycles rented. This finding suggests that there is no statistically significant difference in the number of cycles rented between working days and non-working days.
3. During the fall season, approximately 30.7% of cycles are rented.
4. In the summer season, around 28.2% of cycles are rented.
5. The winter season records a rental rate of about 26.1% for cycles.
6. The lowest rental rate, at just 15%, is observed in the spring season.
7. Weather condition 1 experiences the highest rental rate, with approximately 70.8% of cycles rented.
8. In weather condition 2, around 24.3% of cycles are rented.
9. Weather condition 3 has a rental rate of approximately 4.9% for cycles.
10. Weather condition 4 exhibits an exceptionally low rental rate, with only 0.00786% of cycles being rented.
11. The chi-square test results reveal a statistically significant association between weather type and the season.

Recommendations

Actionable Insight : Despite the fact that 68.6% of cycles are rented on working days compared to 31.4% on non-working days, our t-test analysis does not provide sufficient evidence to conclude that working days have a significant effect on the number of cycles rented. This finding suggests that there is no statistically significant difference in the number of cycles rented between working days and non-working days.

Recommendations:

1. Yulu can consider adjusting its fleet allocation and marketing efforts to better align with customer demand. While working days might not significantly impact rentals, Yulu can focus on peak hours within both working and non-working days to ensure bikes are available when and where they are needed most.
2. Yulu can engage with users through notifications and alerts to inform them of bike availability and incentives during specific timeframes, encouraging rentals during periods with lower demand.