Business Case: Yulu - Hypothesis Testing

About Yulu

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!

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

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
import scipy.stats as spy

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

df.shape

(10886, 12)

df.columns

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

df.head()

{"repr_error":"'str' object has no attribute
'empty'","type":"dataframe","variable_name":"df"}

df.tail()
```

{"repr_error":"'str' object has no attribute
'empty'","type":"dataframe"}

```
np.any(df.isna())
```

```
False
```

```
np.any(df.duplicated())
```

```
False
```

```
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
```

Converting the datatype of datetime column from object to datetime

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

```
df['datetime'].min()
```

```
Timestamp('2011-01-01 00:00:00')
```

```
df['datetime'].max()
```

```
Timestamp('2012-12-19 23:00:00')
```

```
df['datetime'].max() - df['datetime'].min()
```

```
Timedelta('718 days 23:00:00')
```

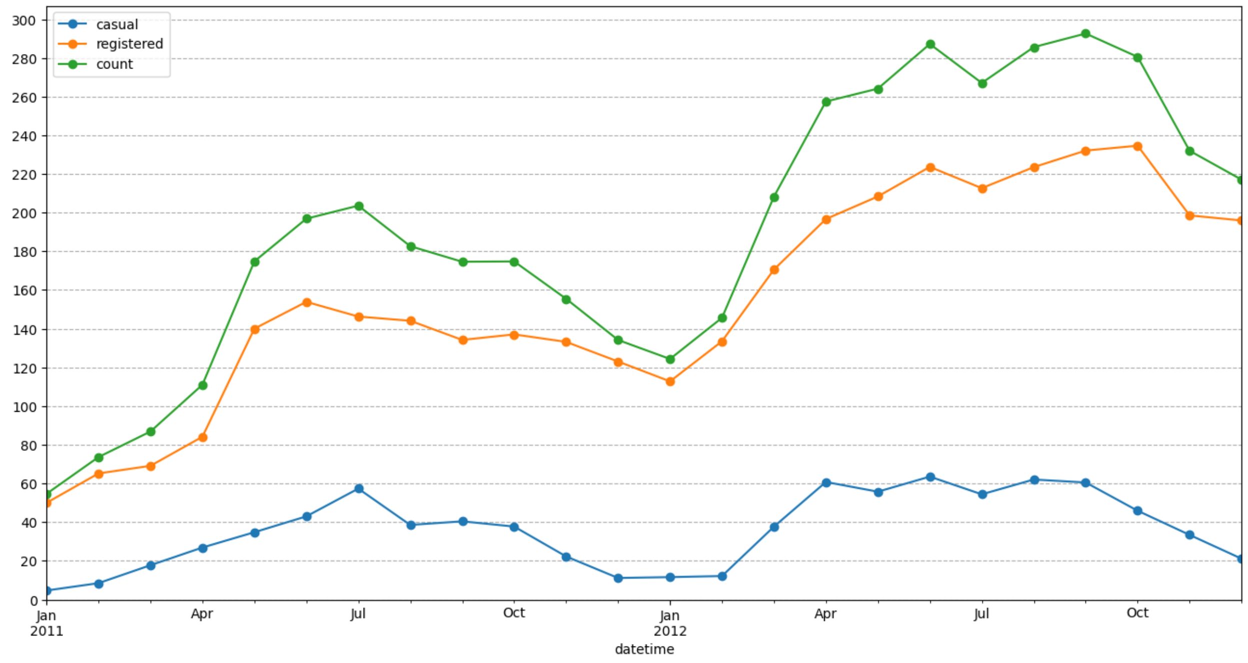
```
df['day'] = df['datetime'].dt.day_name()
```

```
df.set_index('datetime', inplace = True)
```

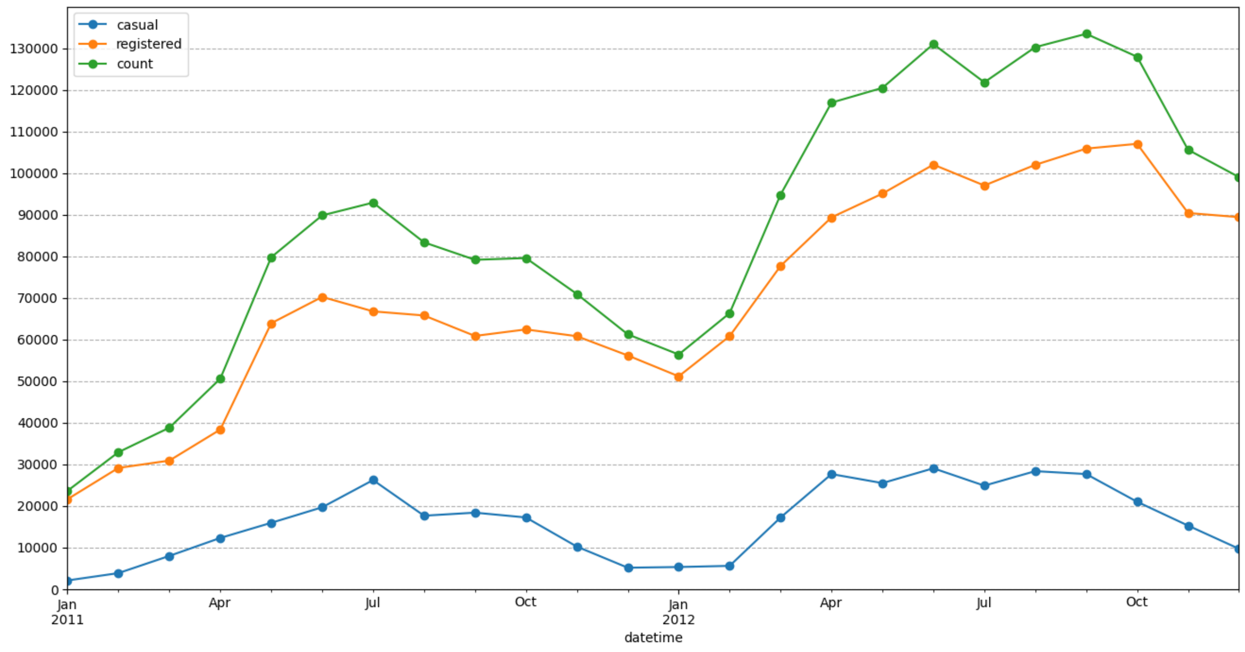
Slicing Data by Time

```
plt.figure(figsize = (16, 8))
df.resample('M')['casual'].mean().plot(kind = 'line', legend =
'casual', marker = 'o')
```

```
df.resample('M')['registered'].mean().plot(kind = 'line', legend =
'registered', marker = 'o')
df.resample('M')['count'].mean().plot(kind = 'line', legend = 'count',
marker = 'o')
plt.grid(axis = 'y', linestyle = '--')
plt.yticks(np.arange(0, 301, 20))
plt.ylim(0,)
plt.show()
```



```
plt.figure(figsize = (16, 8))

df.resample('M')['casual'].sum().plot(kind = 'line', legend =
'casual', marker = 'o')
df.resample('M')['registered'].sum().plot(kind = 'line', legend =
'registered', marker = 'o')
df.resample('M')['count'].sum().plot(kind = 'line', legend = 'count',
marker = 'o')

plt.grid(axis = 'y', linestyle = '--')
plt.yticks(np.arange(0, 130001, 10000))
plt.ylim(0,)
plt.show()
```



I want to know if there is an increase in the average hourly count of rental bikes from the year 2011 to 2012

```
df1 = df.resample('Y')['count'].mean().to_frame().reset_index()

df1['prev_count'] = df1['count'].shift(1)

df1['growth_percent'] = (df1['count'] - df1['prev_count']) * 100 /
df1['prev_count']
df1
```

{"summary":"{\n \"name\": \"df1\",\n \"rows\": 2,\n \"fields\": [\n   {\n   \"column\": \"datetime\",\n   \"properties\": {\n   \"dtype\": \"date\",\n   \"min\": \"2011-12-31 00:00:00\",\n   \"max\": \"2012-12-31 00:00:00\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"2012-12-31 00:00:00\",\n   \"2011-12-31 00:00:00\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"count\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 66.70675317750982,\n   \"min\": 144.223349317595,\n   \"max\": 238.56094436310394,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   238.56094436310394,\n   144.223349317595\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"prev_count\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": null,\n   \"min\": 144.223349317595,\n   \"max\": 144.223349317595,\n   \"num_unique_values\": 1,\n   \"samples\": [\n   144.223349317595\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\":

```
\"growth_percent\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": null,\n        \"min\":
65.41076427074762,\n        \"max\": 65.41076427074762,\n
\"num_unique_values\": 1,\n        \"samples\": [\n
65.41076427074762\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df1"}
```

This data suggests that there was substantial growth in the count of the variable over the course of one year. The mean total hourly count of rental bikes is 144 for the year 2011 and 239 for the year 2012. An annual growth rate of 65.41 % can be seen in the demand of electric vehicles on an hourly basis. It indicates positive growth and potentially a successful outcome or increasing demand for the variable being measured.

```
df.reset_index(inplace = True)
```

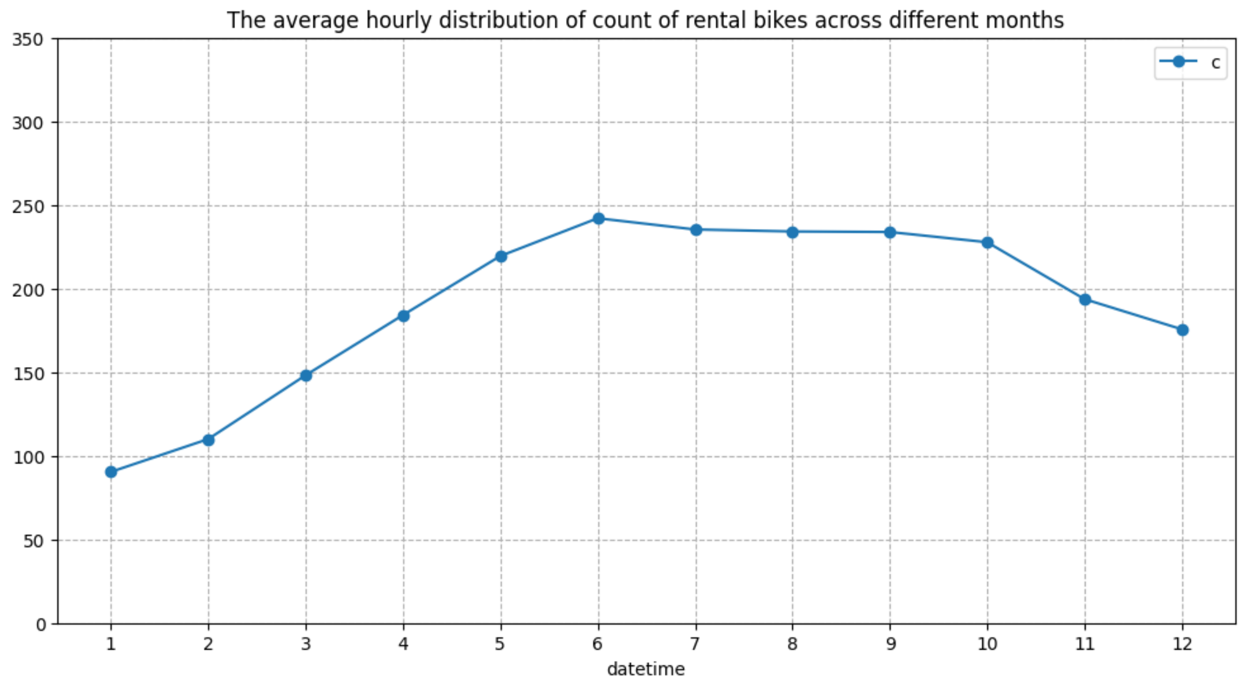
How does the average hourly count of rental bikes varies for different month ?

```
# Grouping the DataFrame by the month
df1 = df.groupby(by = df['datetime'].dt.month)
['count'].mean().reset_index()
df1.rename(columns = {'datetime' : 'month'}, inplace = True)

# Create a new column 'prev_count' by shifting the 'count' column one
position up
    # to compare the previous month's count with the current month's
count
df1['prev_count'] = df1['count'].shift(1)

# Calculating the growth percentage of 'count' with respect to the
'count' of previous month
df1['growth_percent'] = (df1['count'] - df1['prev_count']) * 100 /
df1['prev_count']
df1.set_index('month', inplace = True)
df1
```

```
{"summary":"{\n  \"name\": \"df1\",\n  \"rows\": 12,\n  \"fields\": [\
n    {\n      \"column\": \"count\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 51.54824348509813,\n
\"min\": 90.36651583710407,\n        \"max\": 242.03179824561403,\n
\"num_unique_values\": 12,\n        \"samples\": [\n
193.67727771679472,\n          227.6992316136114,\n
90.36651583710407\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"prev_count\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 53.81852870496405,\n        \"min\":
90.36651583710407,\n        \"max\": 242.03179824561403,\n
\"num_unique_values\": 11,\n        \"samples\": [\n
242.03179824561403,\n          90.36651583710407,\n
227.6992316136114\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"growth_percent\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 15.761504330004792,\n        \"min\": -
14.941619985151904,\n        \"max\": 34.69575131415657,\n
\"num_unique_values\": 11,\n        \"samples\": [\n          -
2.7707683037878277,\n          21.73018801790899,\n          -
14.941619985151904\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df1"}
```

The count of rental bikes shows an increasing trend from January to March, with a significant growth rate of 34.70% between February and March. The growth rate starts to stabilize from April to June, with a relatively smaller growth rate. From July to September, there is a slight decrease in the count of rental bikes, with negative growth rates. The count further declines from October to December, with the largest drop observed between October and November (-14.94%).

```
# Setting the figure size for the plot
plt.figure(figsize = (12, 6))

# Setting the title for the plot
plt.title("The average hourly distribution of count of rental bikes
across different months")

# Grouping the DataFrame by the month and calculating the mean of the
'count' column for each month.
    # Ploting the line graph using markers ('o') to represent the
average count per month.
df.groupby(by = df['datetime'].dt.month)['count'].mean().plot(kind =
'line', marker = 'o')

plt.ylim(0,)
plt.xticks(np.arange(1, 13))
plt.legend('count')
plt.yticks(np.arange(0, 400, 50))
plt.grid(axis = 'both', linestyle = '--')
plt.plot()

[]
```



What is the distribution of average count of rental bikes on an hourly basis in a single day ?

```
df1 = df.groupby(by = df['datetime'].dt.hour)
['count'].mean().reset_index()
df1.rename(columns = {'datetime' : 'hour'}, inplace = True)

# Create a new column 'prev_count' by shifting the 'count' column one
position up
    # to compare the previous hour's count with the current hour's
count
df1['prev_count'] = df1['count'].shift(1)

# Calculating the growth percentage of 'count' with respect to the
'count' of previous hour
df1['growth_percent'] = (df1['count'] - df1['prev_count']) * 100 /
df1['prev_count']
df1.set_index('hour', inplace = True)
df1
```

{"summary":"{\n \"name\": \"df1\",\n \"rows\": 24,\n \"fields\": [\
n   {\n   \"column\": \"count\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 133.24582163120547,\n
\"min\": 6.407239819004525,\n   \"max\": 468.765350877193,\n
\"num_unique_values\": 24,\n   \"samples\": [\n
362.7692307692308,\n   316.37280701754383,\n
55.13846153846154\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"prev_count\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 134.44474670604328,\n   \"min\":

```
6.407239819004525,\n        \"max\": 468.765350877193,\n
\"num_unique_values\": 23,\n        \"samples\": [\n
254.2982456140351,\n          221.78021978021977,\n
55.13846153846154\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"growth_percent\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 87.47730477156162,\n        \"min\": -
48.656179095378874,\n        \"max\": 285.77752633488507,\n
\"num_unique_values\": 23,\n        \"samples\": [\n
24.410141428078635,\n          -21.051431969081356,\n          -
38.59271751101322\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df1"}
```

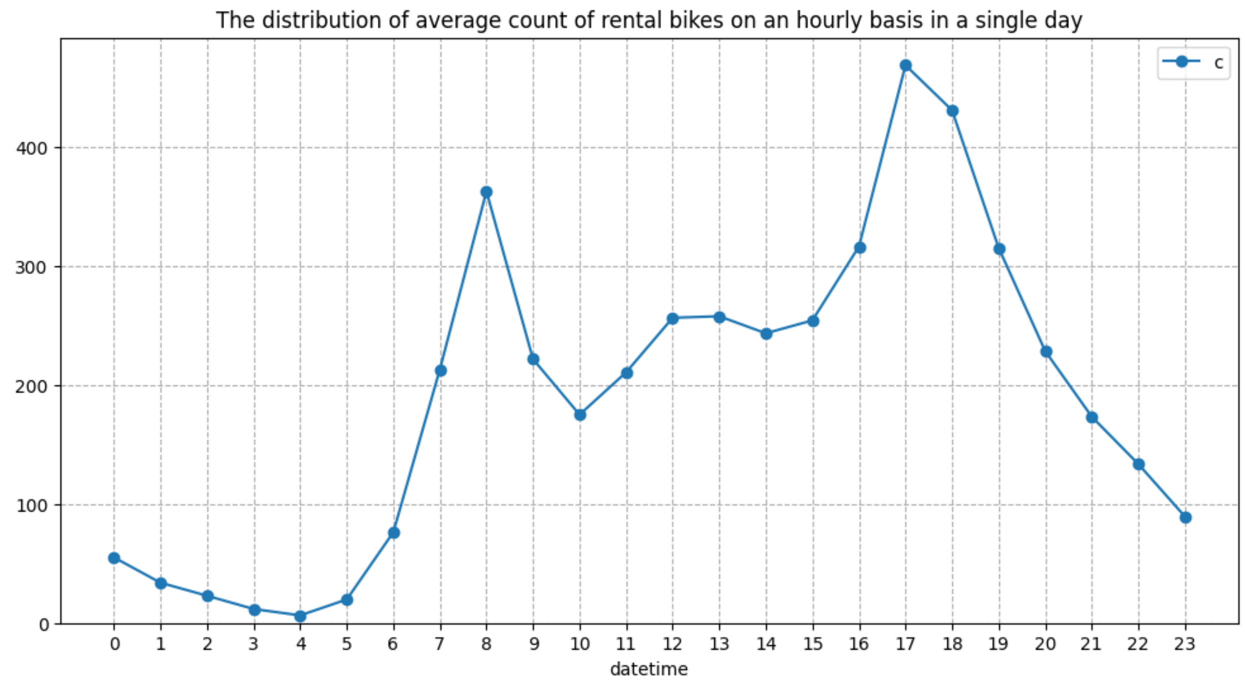
During the early morning hours (hours 0 to 5), there is a significant decrease in the count, with negative growth percentages ranging from -38.59% to -48.66%.

However, starting from hour 5, there is a sudden increase in count, with a sharp positive growth percentage of 208.52% observed from hour 4 to hour 5.

The count continues to rise significantly until reaching its peak at hour 17, with a growth percentage of 48.17% compared to the previous hour.

After hour 17, there is a gradual decrease in count, with negative growth percentages ranging from -8.08% to -32.99% during the late evening and nighttime hours.

```
plt.figure(figsize = (12, 6))
plt.title("The distribution of average count of rental bikes on an
hourly basis in a single day")
df.groupby(by = df['datetime'].dt.hour)['count'].mean().plot(kind =
'line', marker = 'o')
plt.ylim(0,)
plt.xticks(np.arange(0, 24))
plt.legend('count')
plt.grid(axis = 'both', linestyle = '--')
plt.plot()

[]
```



The average count of rental bikes is the highest at 5 PM followed by 6 PM and 8 AM of the day.

The average count of rental bikes is the lowest at 4 AM followed by 3 AM and 5 AM of the day.

These patterns indicate that there is a distinct fluctuation in count throughout the day, with low counts during early morning hours, a sudden increase in the morning, a peak count in the afternoon, and a gradual decline in the evening and nighttime.

Basic Information about the Dataset

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    10886 non-null  datetime64[ns]
 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
 12  day         10886 non-null  object
```

```
dtypes: datetime64[ns](1), float64(3), int64(8), object(1)
memory usage: 1.1+ MB
```

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

Optimizing Memory Usage of the Dataframe

Updating dtype of season column

```
print('Memory usage of season column : ', df['season'].memory_usage())
# Since the dtype of season column is object, we can convert the dtype
to category to save memory
df['season'] = df['season'].astype('category')
print('Updated Memory usage of season column : ',
df['season'].memory_usage())
```

```
Memory usage of season column :  87216
Updated Memory usage of season column :  11218
```

```
print('Max value entry in holiday column : ', df['holiday'].max())
print('Memory usage of holiday column : ',
df['holiday'].memory_usage())
# Since the maximum entry in holiday column is 1 and the dtype is
int64, we can convert the dtype to category to save memory
df['holiday'] = df['holiday'].astype('category')
print('Updated Memory usage of holiday column : ',
df['holiday'].memory_usage())
```

```
Max value entry in holiday column :  1
Memory usage of holiday column :  87216
Updated Memory usage of holiday column :  11138
```

Updating dtype of workingday column

```
print('Max value entry in workingday column : ',
df['workingday'].max())
print('Memory usage of workingday column : ',
df['workingday'].memory_usage())
# Since the maximum entry in workingday column is 1 and the dtype is
int64, we can convert the dtype to category to save memory
df['workingday'] = df['workingday'].astype('category')
```

```
print('Updated Memory usage of workingday column : ',
df['workingday'].memory_usage())
```

```
Max value entry in workingday column :  1
Memory usage of workingday column :  87216
Updated Memory usage of workingday column :  11138
```

Updating dtype of weather column

```
print('Max value entry in weather column : ', df['weather'].max())
print('Memory usage of weather column : ',
df['weather'].memory_usage())
# Since the maximum entry in weather column is 4 and the dtype is
int64, we can convert the dtype to category to save memory
df['weather'] = df['weather'].astype('category')
print('Updated Memory usage of weather column : ',
df['weather'].memory_usage())
```

```
Max value entry in weather column :  4
Memory usage of weather column :  87216
Updated Memory usage of weather column :  11218
```

Updating dtype of temp column

```
print('Max value entry in temp column : ', df['temp'].max())
print('Memory usage of temp column : ', df['temp'].memory_usage())
# Since the maximum entry in temp column is 41.0 and the dtype is
float64, we can convert the dtype to float32 to save memory
df['temp'] = df['temp'].astype('float32')
print('Updated Memory usage of temp column : ',
df['temp'].memory_usage())
```

```
Max value entry in temp column :  41.0
Memory usage of temp column :  87216
Updated Memory usage of temp column :  43672
```

Updating dtype of atemp column

```
print('Max value entry in atemp column : ', df['atemp'].max())
print('Memory usage of atemp column : ', df['atemp'].memory_usage())
# Since the maximum entry in atemp column is 45.455 and the dtype is
float64, we can convert the dtype to float32 to save memory
df['atemp'] = df['atemp'].astype('float32')
print('Updated Memory usage of atemp column : ',
df['atemp'].memory_usage())
```

```
Max value entry in atemp column :  45.455
Memory usage of atemp column :  87216
Updated Memory usage of atemp column :  43672
```

Updating dtype of humidity column

```
print('Max value entry in humidity column : ', df['humidity'].max())
print('Memory usage of humidity column : ', df['temp'].memory_usage())
# Since the maximum entry in humidity column is 100 and the dtype is
int64, we can convert the dtype to int8 to save memory
df['humidity'] = df['humidity'].astype('int8')
print('Updated Memory usage of humidity column : ',
df['humidity'].memory_usage())
```

```
Max value entry in humidity column :  100
Memory usage of humidity column :  43672
Updated Memory usage of humidity column :  11014
```

Updating dtype of windspeed column

```
print('Max value entry in windspeed column : ', df['windspeed'].max())
print('Memory usage of windspeed column : ',
df['windspeed'].memory_usage())
# Since the maximum entry in windspeed column is 56.9969 and the dtype
is float64, we can convert the dtype to float32 to save memory
df['windspeed'] = df['windspeed'].astype('float32')
print('Updated Memory usage of windspeed column : ',
df['windspeed'].memory_usage())
```

```
Max value entry in windspeed column :  56.9969
Memory usage of windspeed column :  87216
Updated Memory usage of windspeed column :  43672
```

Updating dtype of casual column

```
print('Max value entry in casual column : ', df['casual'].max())
print('Memory usage of casual column : ', df['casual'].memory_usage())
# Since the maximum entry in casual column is 367 and the dtype is
int64, we can convert the dtype to int16 to save memory
df['casual'] = df['casual'].astype('int16')
print('Updated Memory usage of casual column : ',
df['casual'].memory_usage())
```

```
Max value entry in casual column :  367
Memory usage of casual column :  87216
Updated Memory usage of casual column :  21900
```

Updating dtype of registered column

```
print('Max value entry in registered column : ',
df['registered'].max())
print('Memory usage of registered column : ',
df['registered'].memory_usage())
```

```
# Since the maximum entry in registered column is 886 and the dtype is
int64, we can convert the dtype to int16 to save memory
df['registered'] = df['registered'].astype('int16')
print('Updated Memory usage of registered column : ',
df['registered'].memory_usage())
```

```
Max value entry in registered column :  886
Memory usage of registered column :  87216
Updated Memory usage of registered column :  21900
```

Updating dtype of count column

```
print('Max value entry in count column : ', df['count'].max())
print('Memory usage of count column : ', df['count'].memory_usage())
# Since the maximum entry in count column is 977 and the dtype is
int64, we can convert the dtype to int16 to save memory
df['count'] = df['count'].astype('int16')
print('Updated Memory usage of count column : ',
df['count'].memory_usage())
```

```
Max value entry in count column :  977
Memory usage of count column :  87216
Updated Memory usage of count column :  21900
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    10886 non-null  datetime64[ns]
 1   season      10886 non-null  category
 2   holiday     10886 non-null  category
 3   workingday  10886 non-null  category
 4   weather     10886 non-null  category
 5   temp        10886 non-null  float32
 6   atemp       10886 non-null  float32
 7   humidity    10886 non-null  int8
 8   windspeed   10886 non-null  float32
 9   casual      10886 non-null  int16
 10  registered  10886 non-null  int16
 11  count       10886 non-null  int16
 12  day         10886 non-null  object
dtypes: category(4), datetime64[ns](1), float32(3), int16(3), int8(1),
object(1)
memory usage: 415.4+ KB
```

Earlier the dataset was using 1.1+ MB of memory but now it has been reduced to 415.2+ KB.
Around 63.17 % reduction in the memory usage.

```
df.describe()
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 8,\n  \"fields\": [\n
{\n      \"column\": \"temp\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 3842.2088125886876,\n
\"min\": 0.8199999928474426,\n        \"max\": 10886.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
20.23086166381836,\n          20.5,\n          10886.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"atemp\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 3841.214608885923,\n
\"min\": 0.7599999904632568,\n        \"max\": 10886.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
23.65508460998535,\n          24.239999771118164,\n          10886.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"humidity\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3830.3684503021896,\n        \"min\": 0.0,\n        \"max\": 10886.0,\
n        \"num_unique_values\": 8,\n        \"samples\": [\n
61.88645967297446,\n          62.0,\n          10886.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"windspeed\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3843.014939484328,\n        \"min\": 0.0,\n        \"max\": 10886.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
12.799395561218262,\n          12.998000144958496,\n          10886.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"casual\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 3824.2753676913135,\n        \"min\": 0.0,\n        \"max\":
10886.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n
36.02195480433584,\n          17.0,\n          10886.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"registered\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3779.869612125704,\n        \"min\": 0.0,\n        \"max\": 10886.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
155.5521771082124,\n          118.0,\n          10886.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"count\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 3769.174237043881,\n
\"min\": 1.0,\n        \"max\": 10886.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
191.57413191254824,\n          145.0,\n          10886.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```

```
np.round(df['season'].value_counts(normalize = True) * 100, 2)
```

```
winter    25.11
fall      25.11
summer    25.11
spring    24.67
Name: season, dtype: float64
```

```
np.round(df['holiday'].value_counts(normalize = True) * 100, 2)
```

```
0    97.14
1     2.86
Name: holiday, dtype: float64
```

```
np.round(df['workingday'].value_counts(normalize = True) * 100, 2)
```

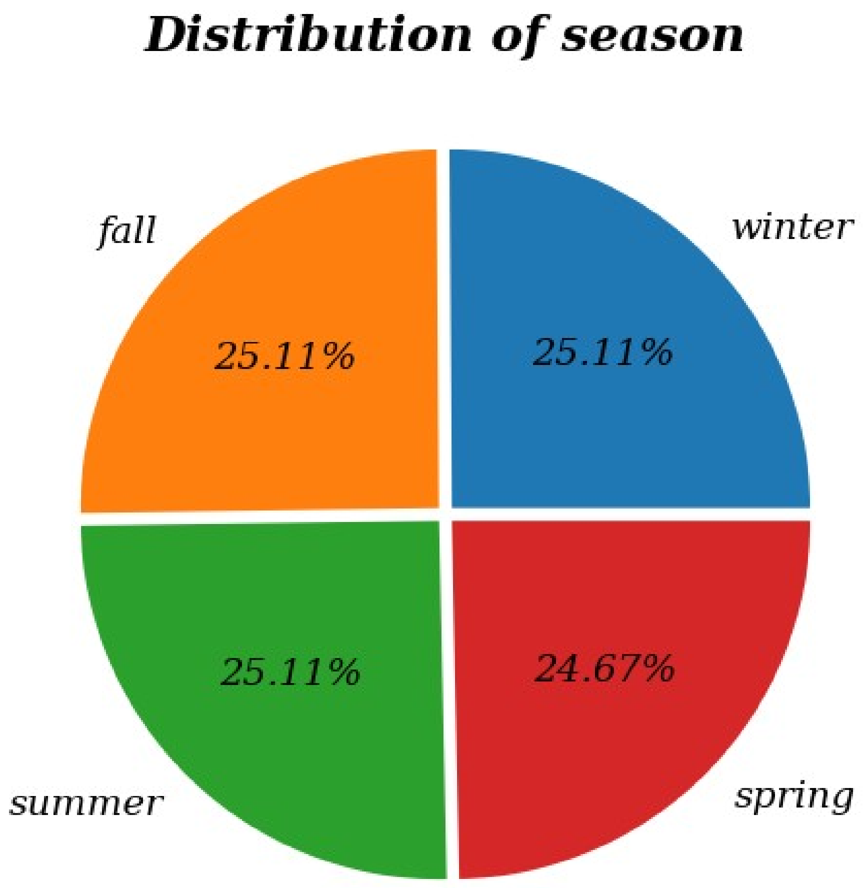
```
1    68.09
0    31.91
Name: workingday, dtype: float64
```

```
np.round(df['weather'].value_counts(normalize = True) * 100, 2)
```

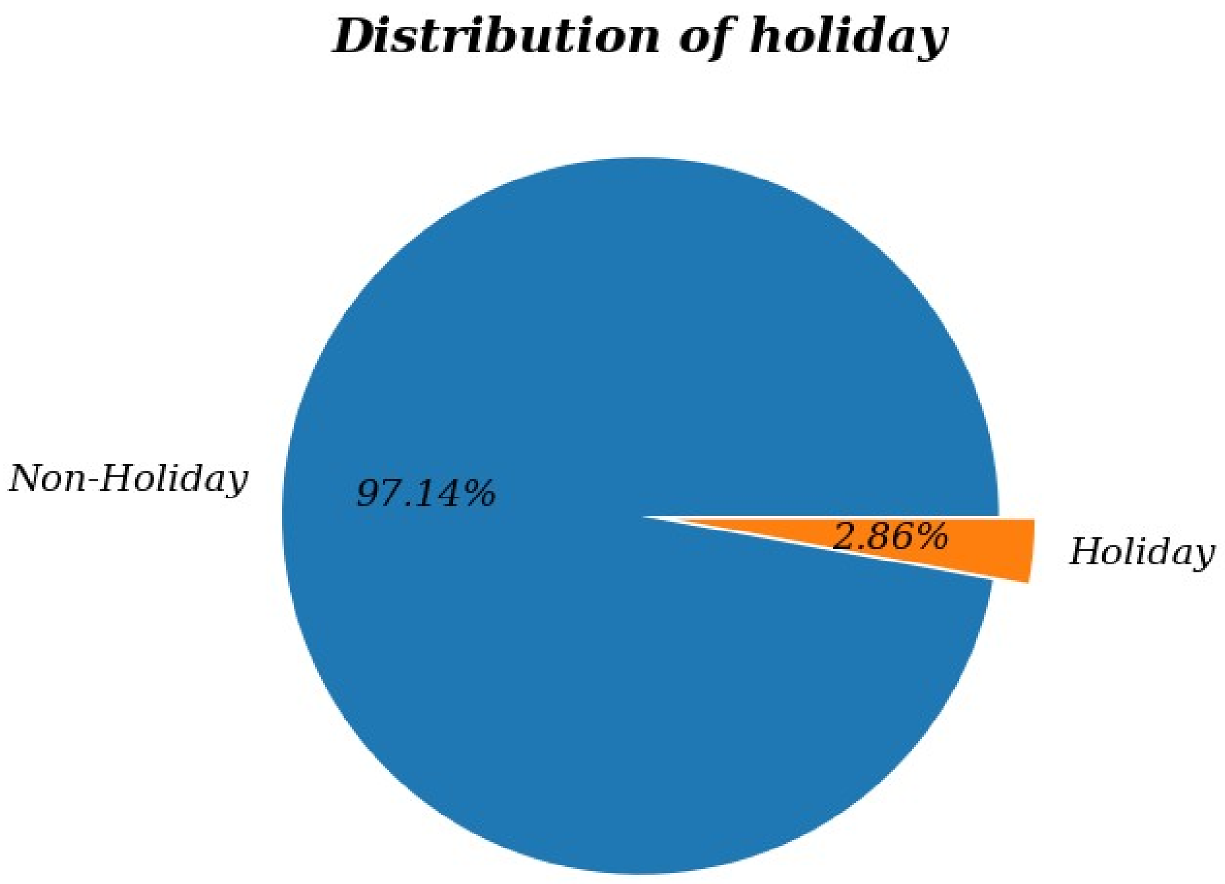
```
1    66.07
2    26.03
3     7.89
4     0.01
Name: weather, dtype: float64
```

```
plt.figure(figsize = (6, 6))

# setting the title of the plot
plt.title('Distribution of season', fontdict = {'fontsize' : 18,
                                                'fontweight' : 600,
                                                'fontstyle' :
'oblique',
                                                'fontfamily' :
'serif'})

df_season = np.round(df['season'].value_counts(normalize = True) *
100, 2).to_frame()

# Creating the pie-chart
plt.pie(x = df_season['season'],
        explode = [0.025, 0.025, 0.025, 0.025],
        labels = df_season.index,
        autopct = '%.2f%%',
        textprops = {'fontsize' : 14,
                     'fontstyle' : 'oblique',
                     'fontfamily' : 'serif',
                     'fontweight' : 500})

plt.plot()     # displaying the plot
```

```
[]
```

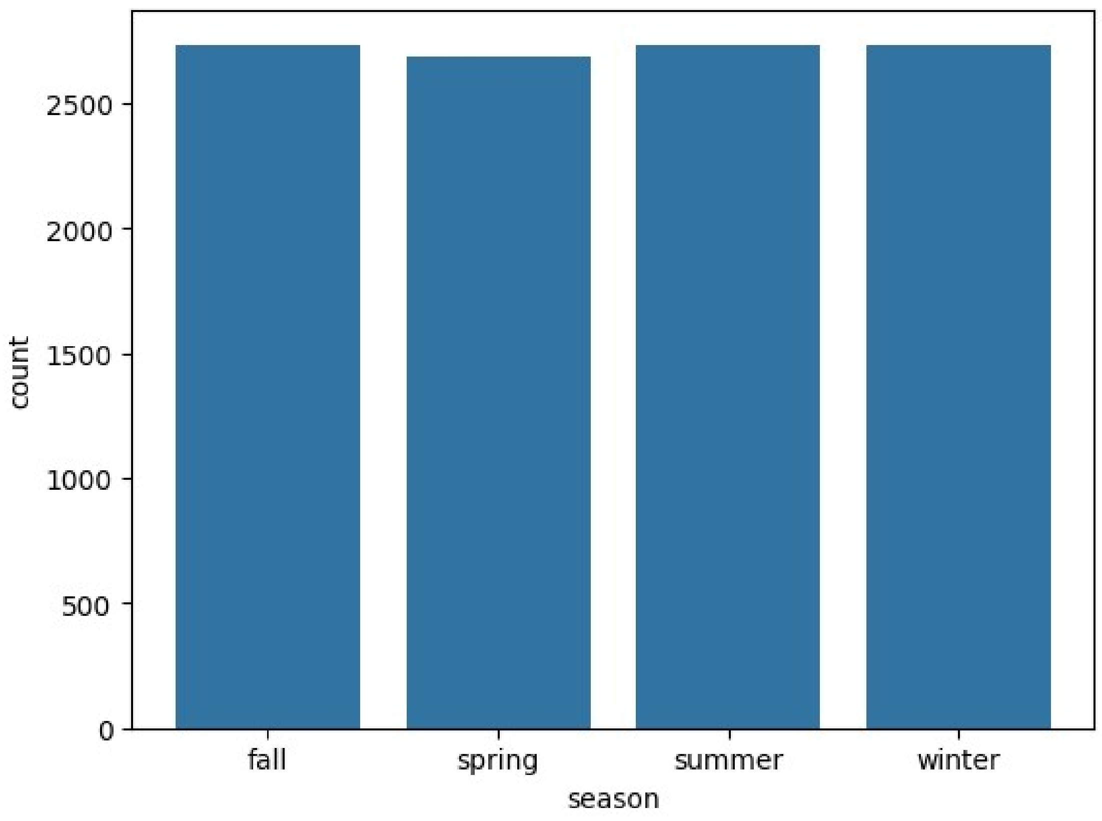


```
plt.figure(figsize = (6, 6))     # setting the figure size to 6*6

# setting the title of the plot
plt.title('Distribution of holiday', fontdict = {'fontsize' : 18,
                                                 'fontweight' : 600,
                                                 'fontstyle' :
'oblique',
                                                 'fontfamily' :
'serif'})

df_holiday = np.round(df['holiday'].value_counts(normalize = True) *
100, 2).to_frame()

# Creating the pie-chart
plt.pie(x = df_holiday['holiday'],
        explode = [0, 0.1],
        labels = ['Non-Holiday', 'Holiday'],
        autopct = '%.2f%%',
        textprops = {'fontsize' : 14,
                   'fontstyle' : 'oblique',
                   'fontfamily' : 'serif',
                   'fontweight' : 500})

plt.plot()        # displaying the plot

[]
```

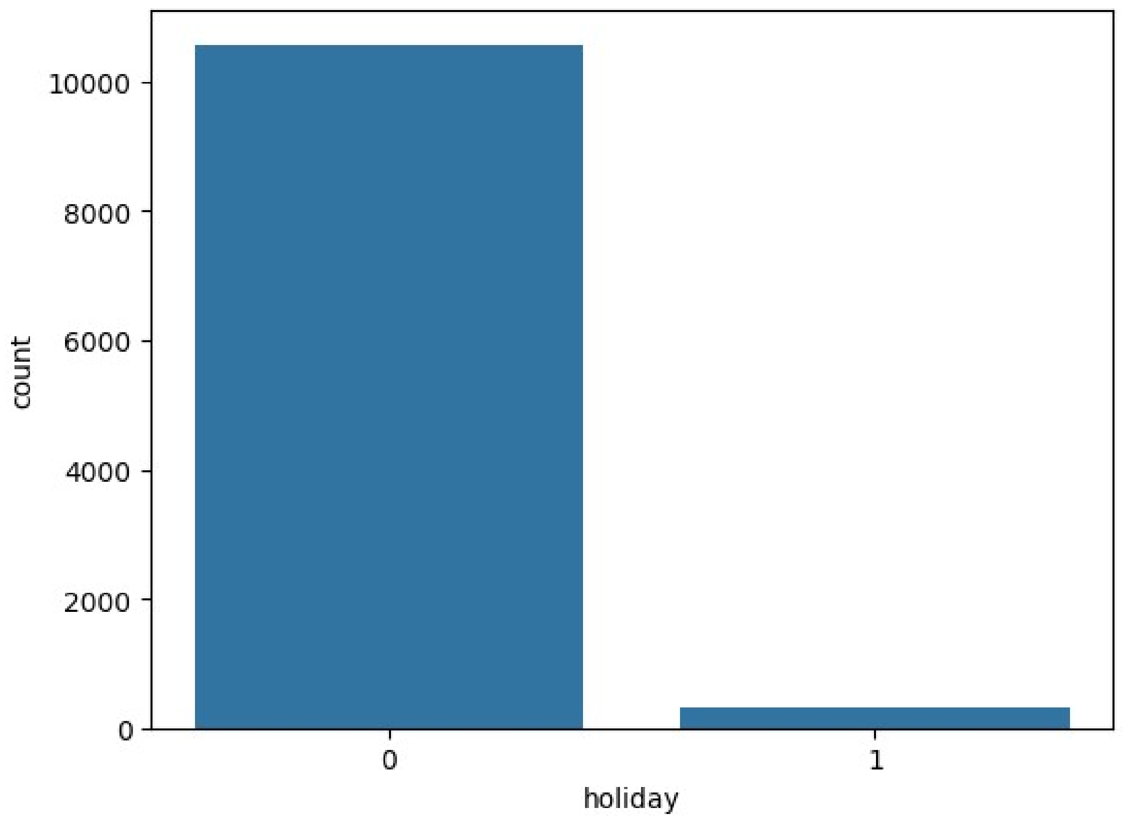


Univariate Analysis

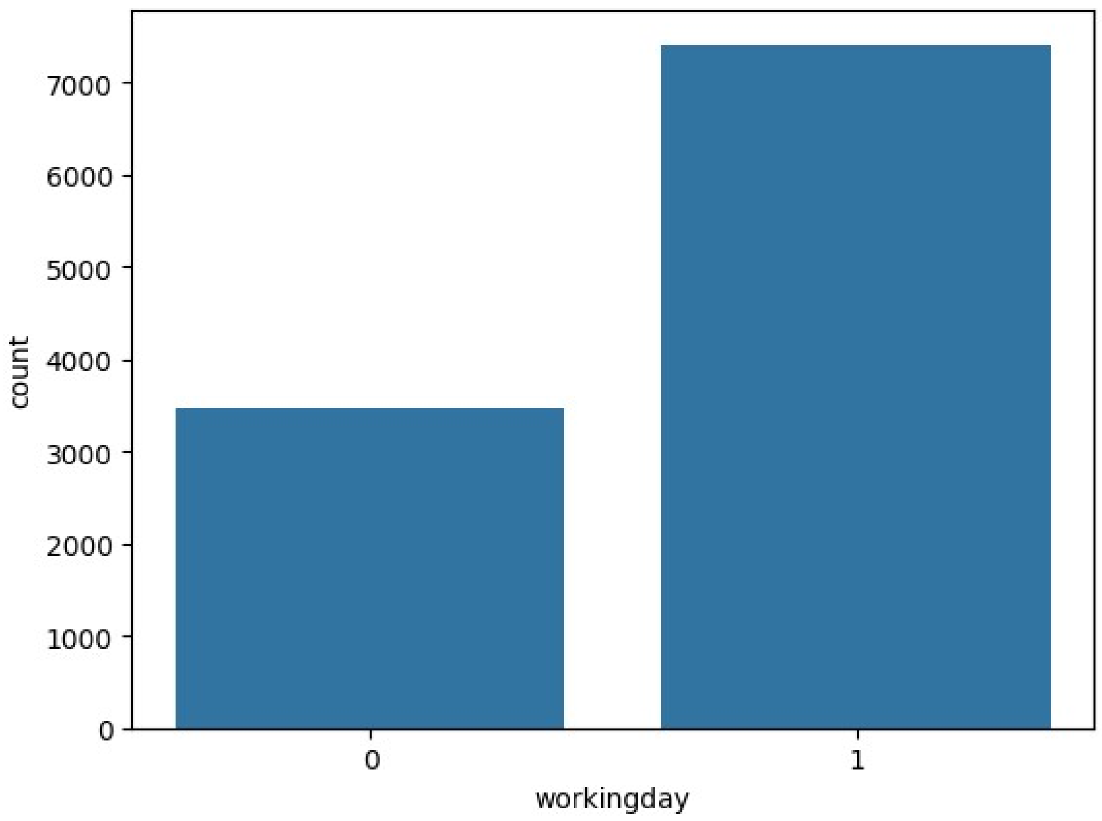
```
sns.countplot(data = df, x = 'season')
plt.plot()   # displaying the plot

[]
```



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

[]
```

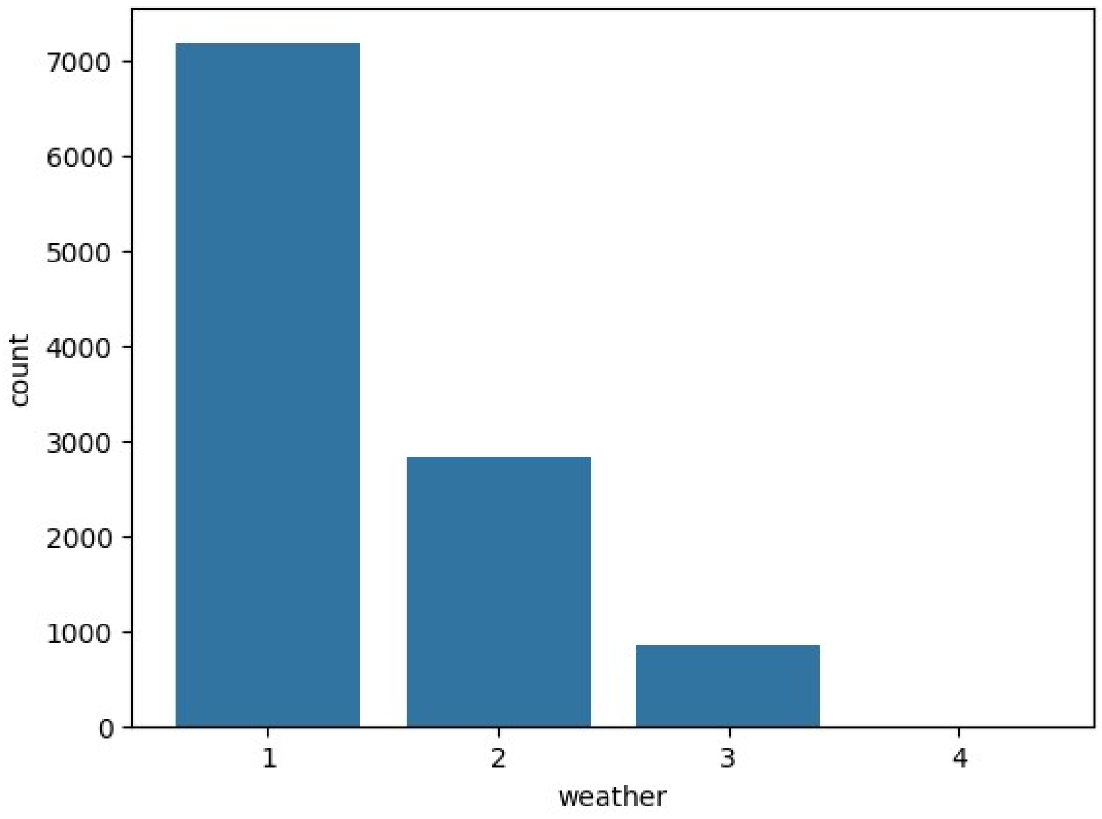


```
sns.countplot(data = df, x = 'workingday')
plt.plot()      # displaying the chart

[]
```

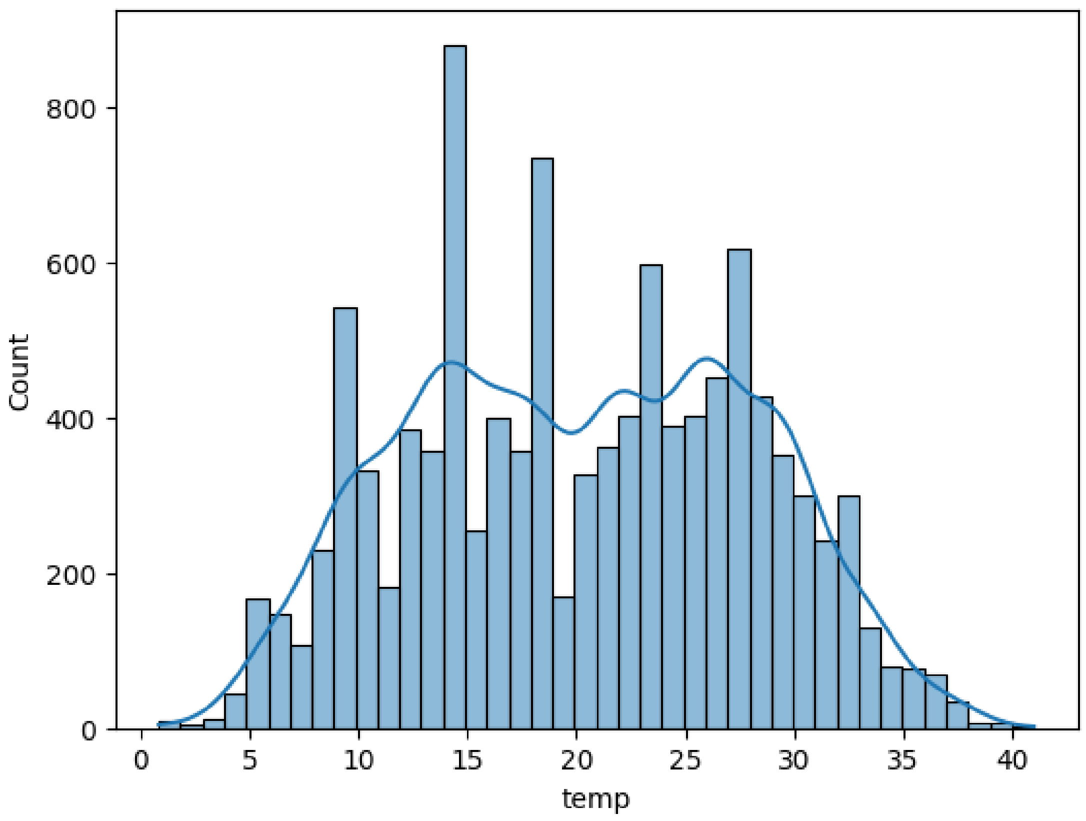


```
sns.countplot(data = df, x = 'weather')
plt.plot()        # displaying the chart
```

```
[]
```



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

[]
```

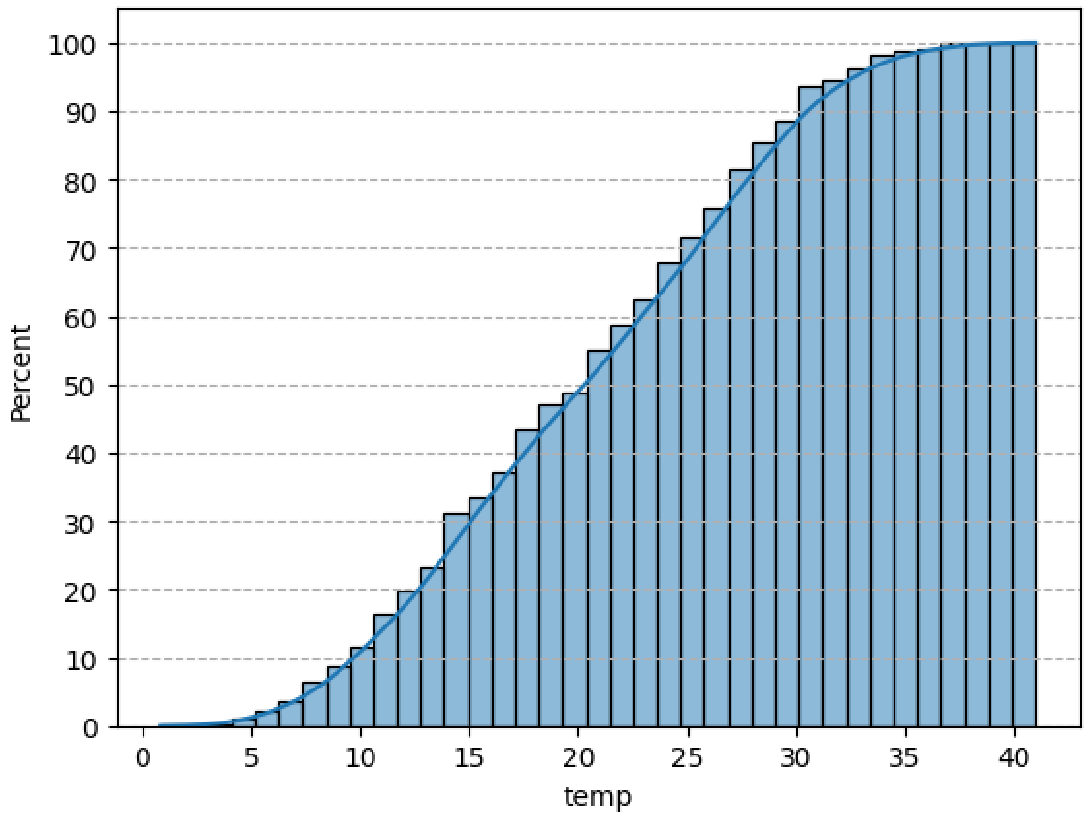


```
temp_mean = np.round(df['temp'].mean(), 2)
temp_std = np.round(df['temp'].std(), 2)
temp_mean, temp_std
```

```
(20.23, 7.79)
```

```
sns.histplot(data = df, x = 'temp', kde = True, cumulative = True,
stat = 'percent')
plt.grid(axis = 'y', linestyle = '--')
plt.yticks(np.arange(0, 101, 10))
plt.plot()
```

```
[]
```

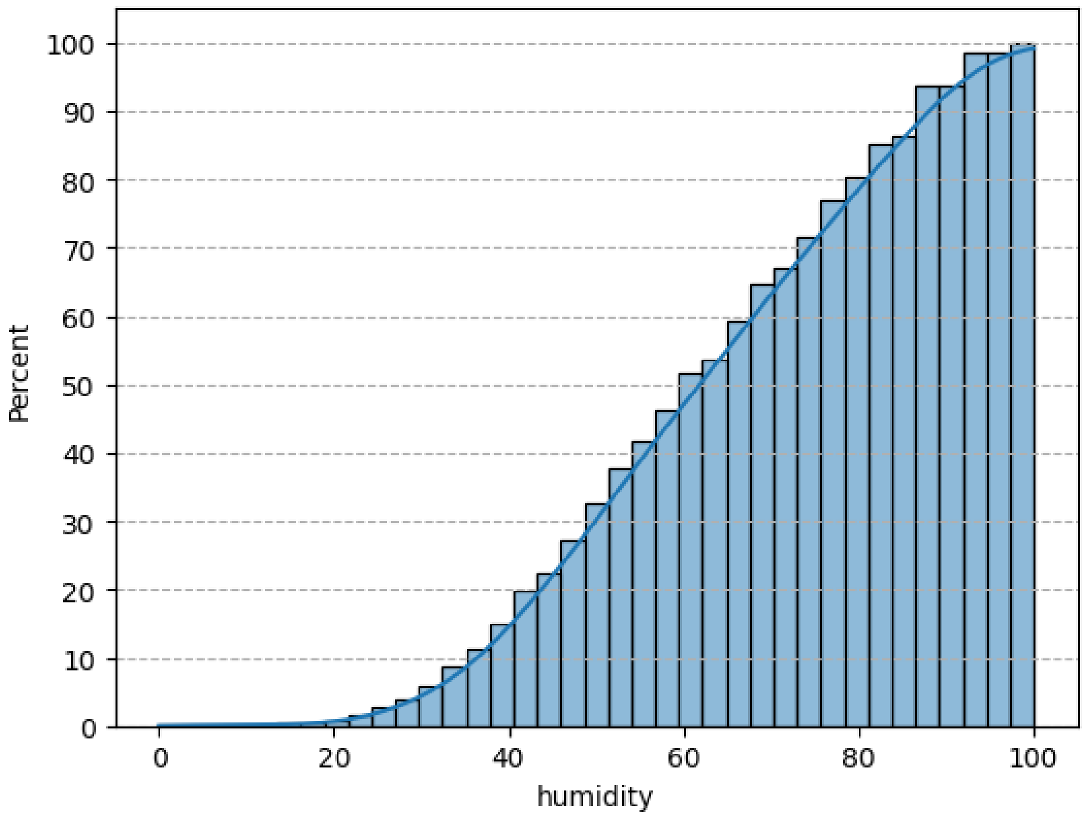


```
humidity_mean = np.round(df['humidity'].mean(), 2)
humidity_std = np.round(df['humidity'].std(), 2)
humidity_mean, humidity_std
```

```
(61.89, 19.25)
```

```
sns.histplot(data = df, x = 'humidity', kde = True, cumulative = True,
stat = 'percent')
plt.grid(axis = 'y', linestyle = '--')    # setting the gridlines
along y axis
plt.yticks(np.arange(0, 101, 10))
plt.plot()
```

```
[]
```

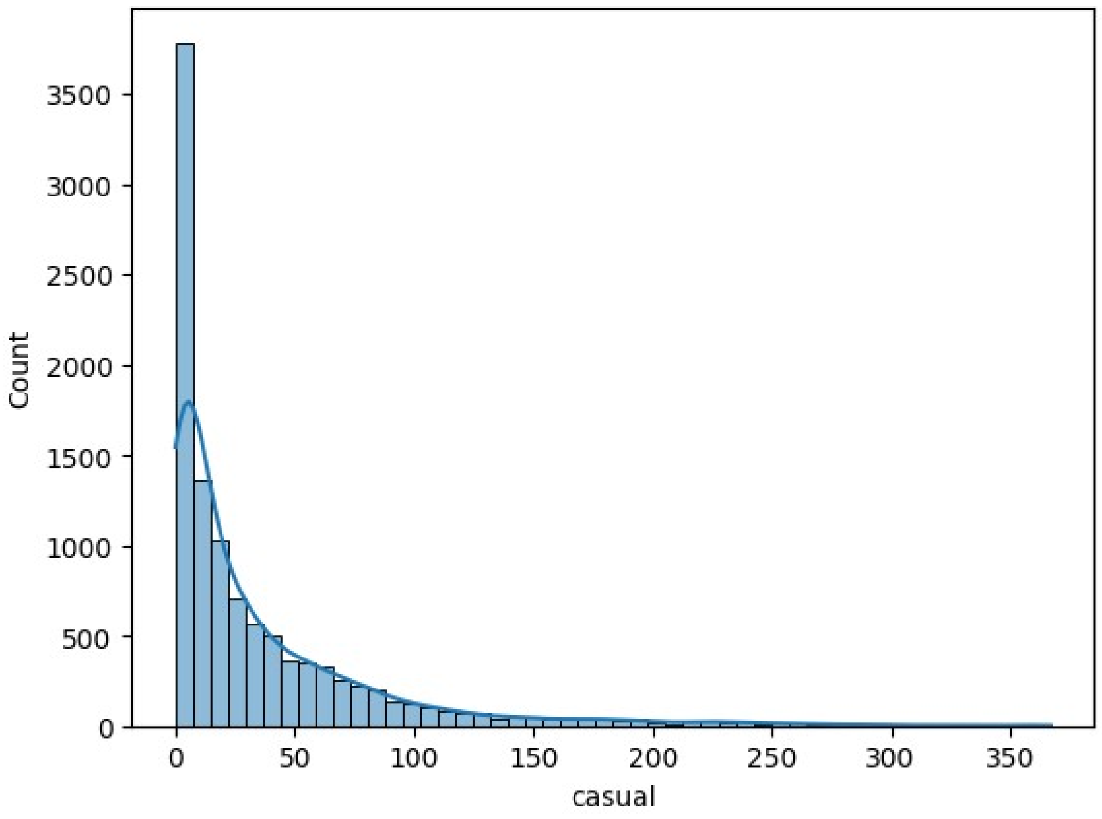


```
len(df[df['windspeed'] < 20]) / len(df)
```

```
0.8626676465184641
```

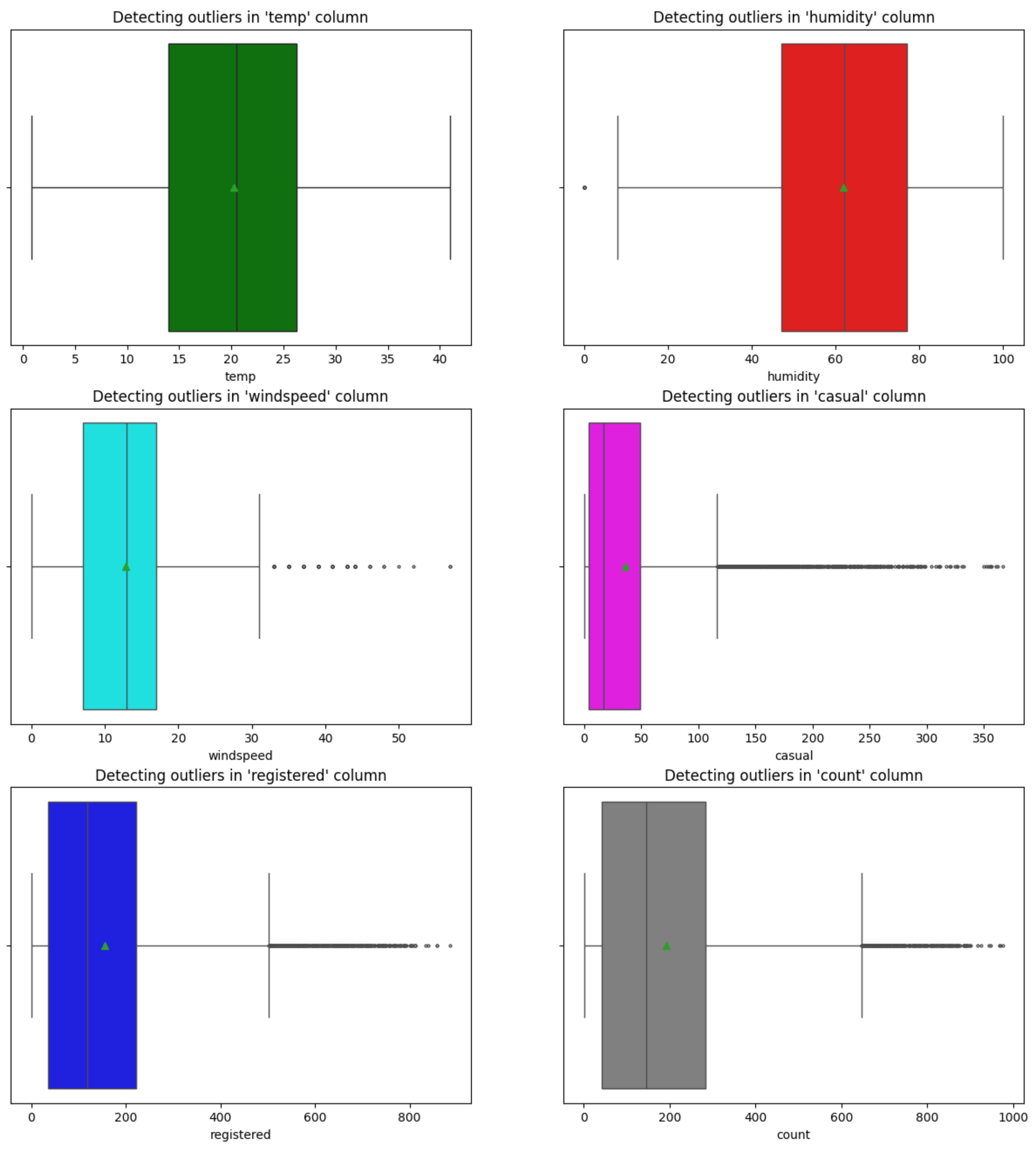
```
sns.histplot(data = df, x = 'casual', kde = True, bins = 50)
plt.plot()
```

```
[]
```



Outliers Detection

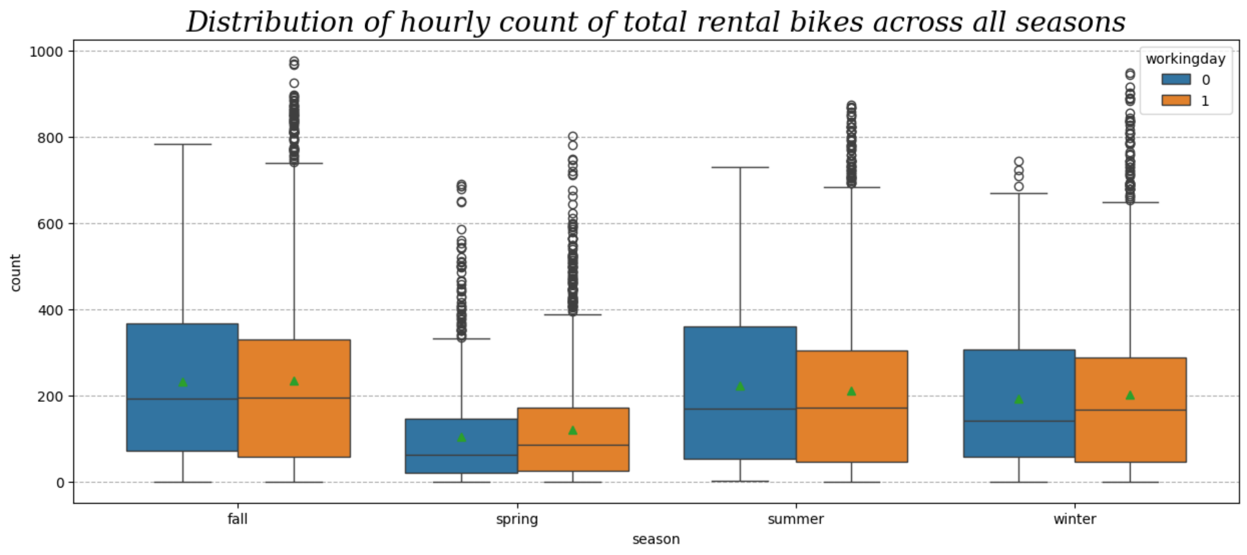
```
columns = ['temp', 'humidity', 'windspeed', 'casual', 'registered',
'count']
colors = np.random.permutation(['red', 'blue', 'green', 'magenta',
'cyan', 'gray'])
count = 1
plt.figure(figsize = (15, 16))
for i in columns:
    plt.subplot(3, 2, count)
    plt.title(f"Detecting outliers in '{i}' column")
    sns.boxplot(data = df, x = df[i], color = colors[count - 1],
showmeans = True, fliersize = 2)
    plt.plot()
    count += 1
```



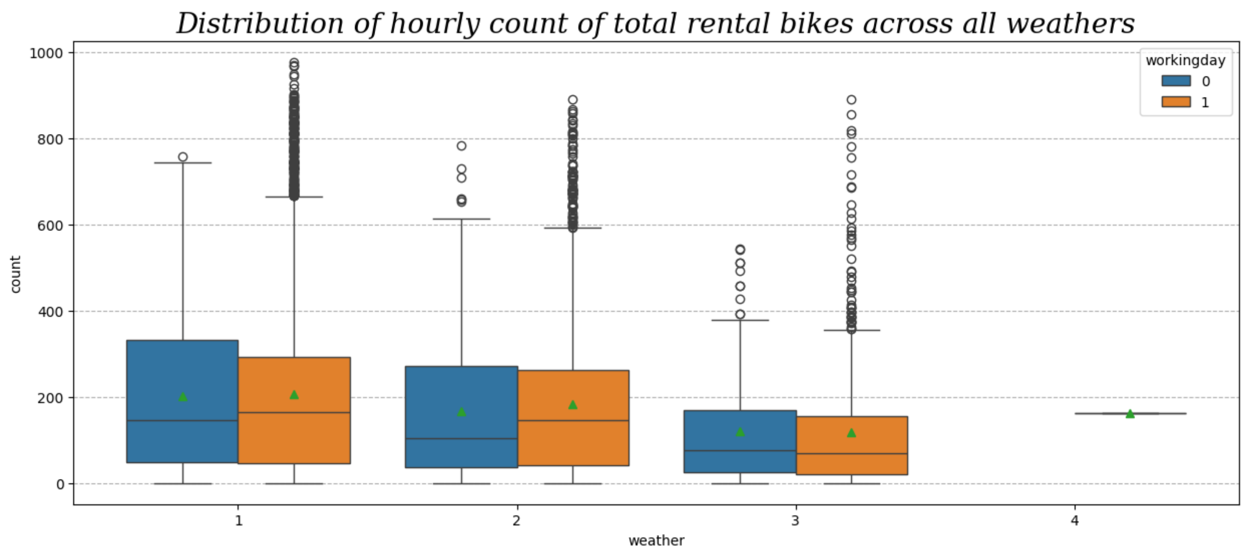
Bivariate Analysis

```
plt.figure(figsize = (15, 6))
plt.title('Distribution of hourly count of total rental bikes across
all seasons',
         fontdict = {'size' : 20,
                     'style' : 'oblique',
                     'family' : 'serif'})
```

```
sns.boxplot(data = df, x = 'season', y = 'count', hue = 'workingday',
showmeans = True)
plt.grid(axis = 'y', linestyle = '--')
plt.plot()

[]
```



```
plt.figure(figsize = (15, 6))
plt.title('Distribution of hourly count of total rental bikes across
all weathers',
         fontdict = {'size' : 20,
                     'style' : 'oblique',
                     'family' : 'serif'})
sns.boxplot(data = df, x = 'weather', y = 'count', hue = 'workingday',
showmeans = True)
plt.grid(axis = 'y', linestyle = '--')
plt.plot()

[]
```



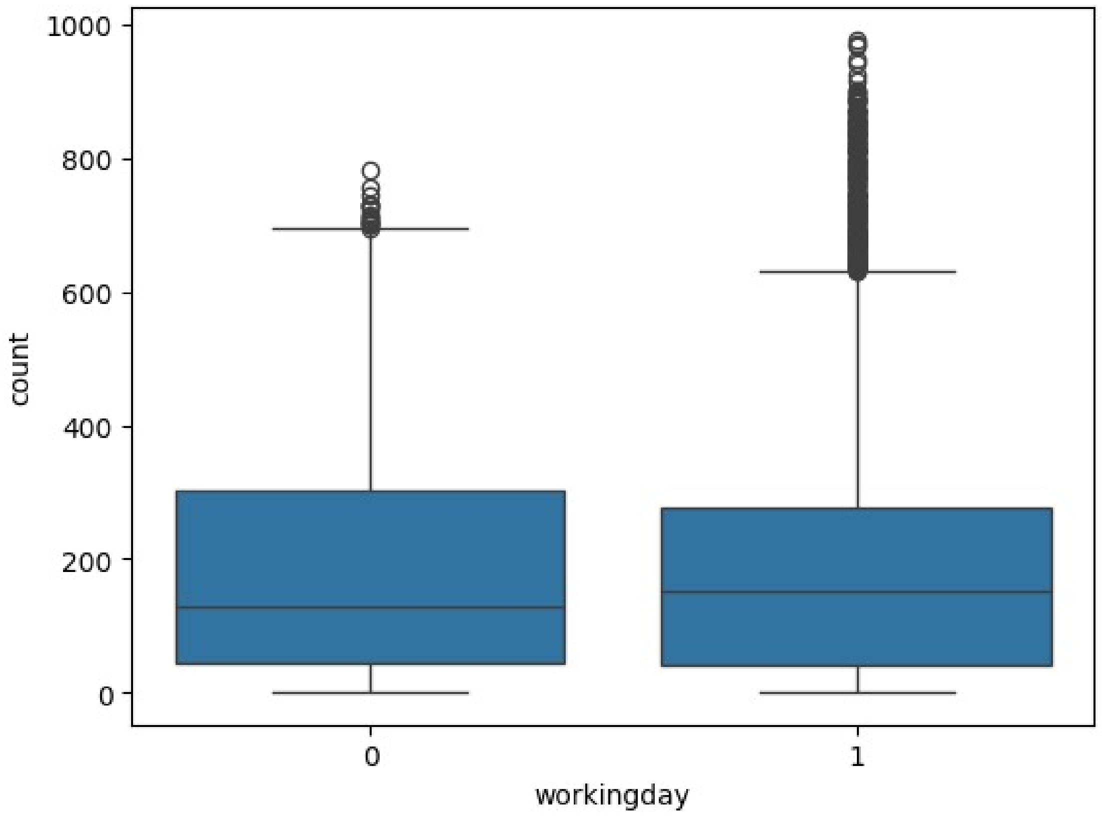
Is there any effect of Working Day on the number of electric cycles rented ?

```
df.groupby(by = 'workingday')['count'].describe()
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 2,\n  \"fields\": [\n
{\n      \"column\": \"count\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 2784.586504312624,\n
\"min\": 3474.0,\n        \"max\": 7412.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          7412.0,\n
3474.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"mean\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 3.1856942604597354,\n        \"min\": 188.50662061024755,\n
\"max\": 193.01187263896384,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          193.01187263896384,\n
188.50662061024755\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"std\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 7.629430238982044,\n        \"min\": 173.7240153250003,\n
\"max\": 184.5136590421481,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          184.5136590421481,\n
173.7240153250003\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"min\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.0,\n        \"min\": 1.0,\n        \"max\": 1.0,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          1.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"25%\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 2.1213203435596424,\
n        \"min\": 41.0,\n        \"max\": 44.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          41.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"50%\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 16.263455967290593,\
n        \"min\": 128.0,\n        \"max\": 151.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          151.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"75%\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 19.091883092036785,\
n        \"min\": 277.0,\n        \"max\": 304.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          277.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"max\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 137.17871555019022,\
n        \"min\": 783.0,\n        \"max\": 977.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          977.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```

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

```
[]
```



Null Hypothesis (H0) - Working Day does not have any effect on the number of electric cycles rented.

Alternate Hypothesis (HA) - Working Day has some effect on the number of electric cycles rented

Distribution check using QQ Plot Homogeneity of Variances using Levene's test

If the assumptions of T Test are met then we can proceed performing T Test for independent samples else we will perform the non parametric test equivalent to T Test for independent sample i.e., Mann-Whitney U rank test for two independent samples.

We set our alpha to be 0.05

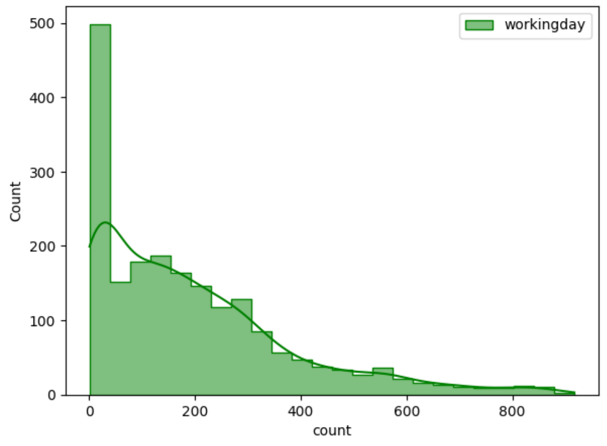
Based on p-value, we will accept or reject H0.

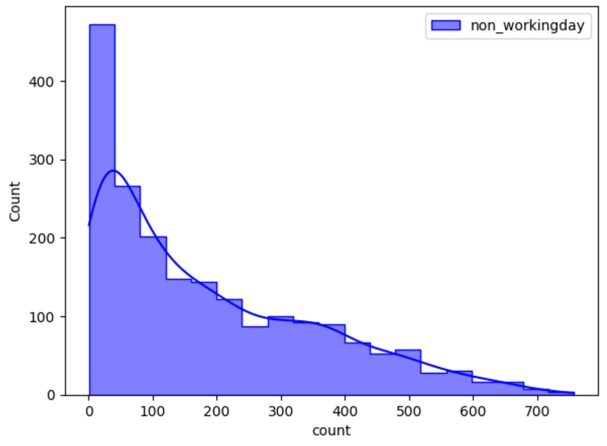
p-val > alpha : Accept H0 p-val < alpha : Reject H0

Visual Tests to know if the samples follow normal distribution

```
plt.figure(figsize = (15, 5))
plt.subplot(1, 2, 1)
sns.histplot(df.loc[df['workingday'] == 1, 'count'].sample(2000),
             element = 'step', color = 'green', kde = True, label =
'workingday')
plt.legend()
plt.subplot(1, 2, 2)
sns.histplot(df.loc[df['workingday'] == 0, 'count'].sample(2000),
             element = 'step', color = 'blue', kde = True, label =
'non_workingday')
plt.legend()
plt.plot()
```

```
[]
```

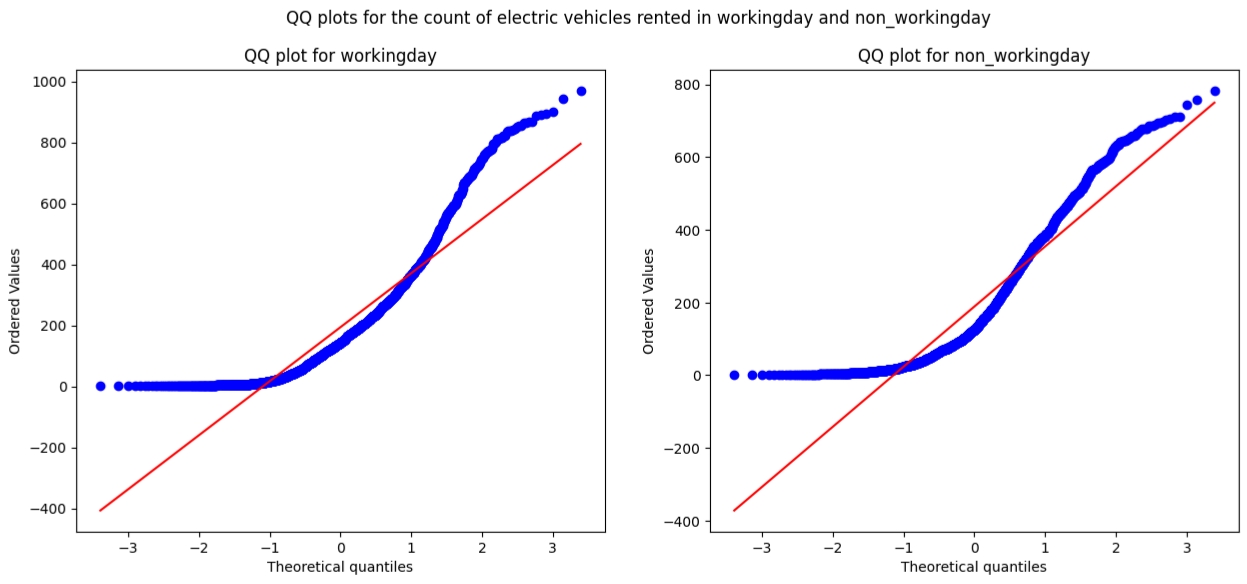




Distribution check using QQ Plot

```
plt.figure(figsize = (15, 6))
plt.subplot(1, 2, 1)
plt.suptitle('QQ plots for the count of electric vehicles rented in
workingday and non_workingday')
spy.probplot(df.loc[df['workingday'] == 1, 'count'].sample(2000), plot
= plt, dist = 'norm')
plt.title('QQ plot for workingday')
plt.subplot(1, 2, 2)
spy.probplot(df.loc[df['workingday'] == 0, 'count'].sample(2000), plot
= plt, dist = 'norm')
plt.title('QQ plot for non_workingday')
plt.plot()
```

```
[]
```



It can be inferred from the above plot that the distributions do not follow normal distribution.

It can be seen from the above plots that the samples do not come from normal distribution.

Applying Shapiro-Wilk test for normality

: The sample follows normal distribution : The sample does not follow normal distribution

alpha = 0.05

Test Statistics : Shapiro-Wilk test for normality

```
test_stat, p_value = spy.shapiro(df.loc[df['workingday'] == 1,
'count'].sample(2000))
print('p-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')
```

```
p-value 1.6363908044384308e-38
The sample does not follow normal distribution
```

```python
test_stat, p_value = spy.shapiro(df.loc[df['workingday'] == 0,
'count'].sample(2000))
print('p-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')
```

```
p-value 4.330088844132551e-36
The sample does not follow normal distribution
```

Transforming the data using boxcox transformation and checking if the transformed data follows normal distribution.

```python
transformed_workingday = spy.boxcox(df.loc[df['workingday'] == 1,
'count'])[0]
test_stat, p_value = spy.shapiro(transformed_workingday)
print('p-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')
```

```
p-value 1.6136246052607705e-33
The sample does not follow normal distribution
```

```
/usr/local/lib/python3.10/dist-packages/scipy/stats/
_morestats.py:1882: UserWarning: p-value may not be accurate for N >
5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

```python
transformed_non_workingday = spy.boxcox(df.loc[df['workingday'] == 1,
'count'])[0]
test_stat, p_value = spy.shapiro(transformed_non_workingday)
print('p-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')
```

```
p-value 1.6136246052607705e-33
The sample does not follow normal distribution
```

```python
# Null Hypothesis(H0) - Homogenous Variance

# Alternate Hypothesis(HA) - Non Homogenous Variance

test_stat, p_value = spy.levene(df.loc[df['workingday'] == 1,
'count'].sample(2000),
                                df.loc[df['workingday'] == 0,
'count'].sample(2000))
print('p-value', p_value)
if p_value < 0.05:
    print('The samples do not have  Homogenous Variance')
else:
    print('The samples have Homogenous Variance ')

p-value 0.4680749226042915
The samples have Homogenous Variance
```

Since the samples are not normally distributed, T-Test cannot be applied here, we can perform its non parametric equivalent test i.e., Mann-Whitney U rank test for two independent samples.

```
# Ho : Mean no.of electric cycles rented is same for working and non-
working days
# Ha : Mean no.of electric cycles rented is not same for working and
non-working days
# Assuming significance Level to be 0.05
# Test statistics : Mann-Whitney U rank test for two independent
samples

test_stat, p_value = spy.mannwhitneyu(df.loc[df['workingday'] == 1,
'count'],
                                      df.loc[df['workingday'] == 0,
'count'])
print('P-value :',p_value)
if p_value < 0.05:
    print('Mean no.of electric cycles rented is not same for working
and non-working days')
else:
    print('Mean no.of electric cycles rented is same for working and
non-working days')

P-value : 0.9679139953914079
Mean no.of electric cycles rented is same for working and non-working
days
```

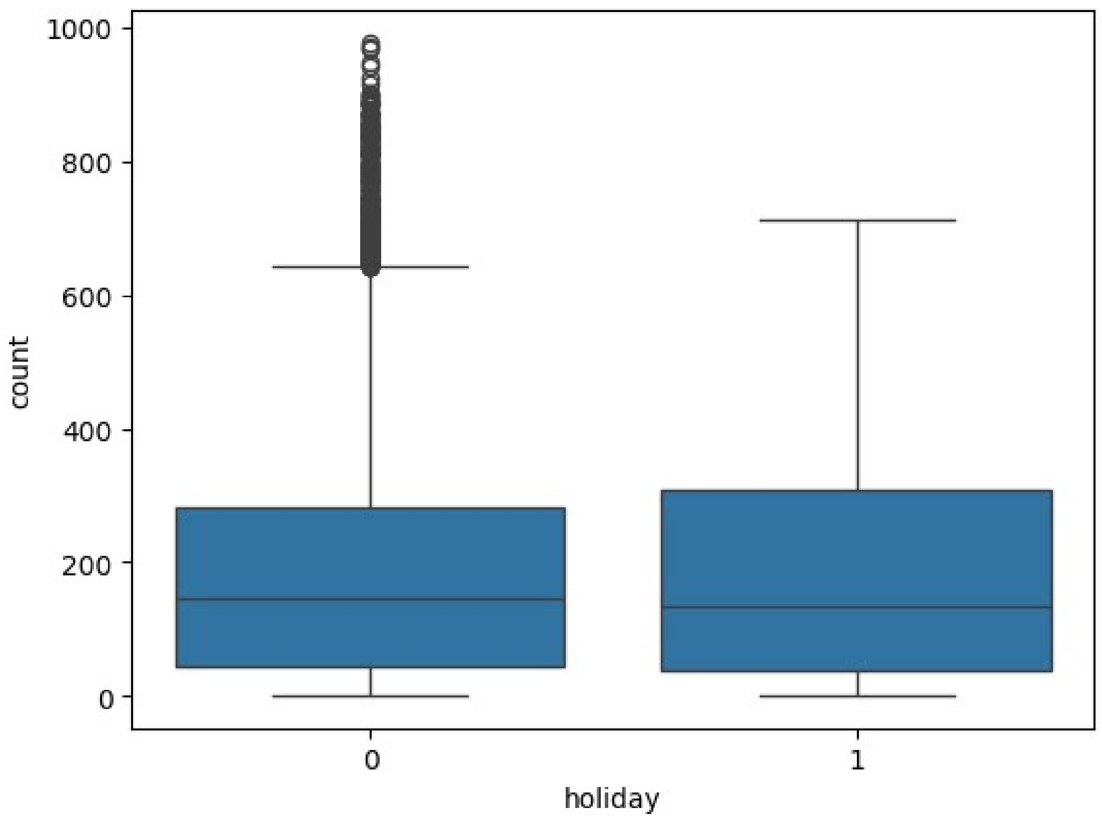
Is there any effect of holidays on the number of electric cycles rented ?

```
df.groupby(by = 'holiday')['count'].describe()
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 2,\n  \"fields\": [\n
{\n      \"column\": \"count\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 7257.744002098724,\n
\"min\": 311.0,\n        \"max\": 10575.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          311.0,\n
10575.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"mean\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 4.146361976385559,\n        \"min\": 185.87781350482314,\n
\"max\": 191.7416548463357,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          185.87781350482314,\n
191.7416548463357\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"std\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 9.342718603983695,\n        \"min\": 168.30053147054628,\n
\"max\": 181.51313082973545,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          168.30053147054628,\n
181.51313082973545\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"min\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.0,\n        \"min\": 1.0,\n        \"max\": 1.0,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          1.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"25%\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 3.181980515339464,\n
\"min\": 38.5,\n        \"max\": 43.0,\n        \"num_unique_values\":
2,\n        \"samples\": [\n          38.5\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"50%\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 8.48528137423857,\n
\"min\": 133.0,\n        \"max\": 145.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          133.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"75%\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 17.67766952966369,\n
\"min\": 283.0,\n        \"max\": 308.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          308.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"max\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 187.3832970144351,\n
\"min\": 712.0,\n        \"max\": 977.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          712.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```

```
sns.boxplot(data = df, x = 'holiday', y = 'count')
plt.plot()
```

```
[]
```



STEP-1 : Set up Null Hypothesis

Null Hypothesis (H0) - Holidays have no effect on the number of electric vehicles rented

Alternate Hypothesis (HA) - Holidays has some effect on the number of electric vehicles rented

STEP-2 : Checking for basic assumpitons for the hypothesis

Distribution check using QQ Plot Homogeneity of Variances using Levene's test STEP-3: Define Test statistics; Distribution of T under H0.

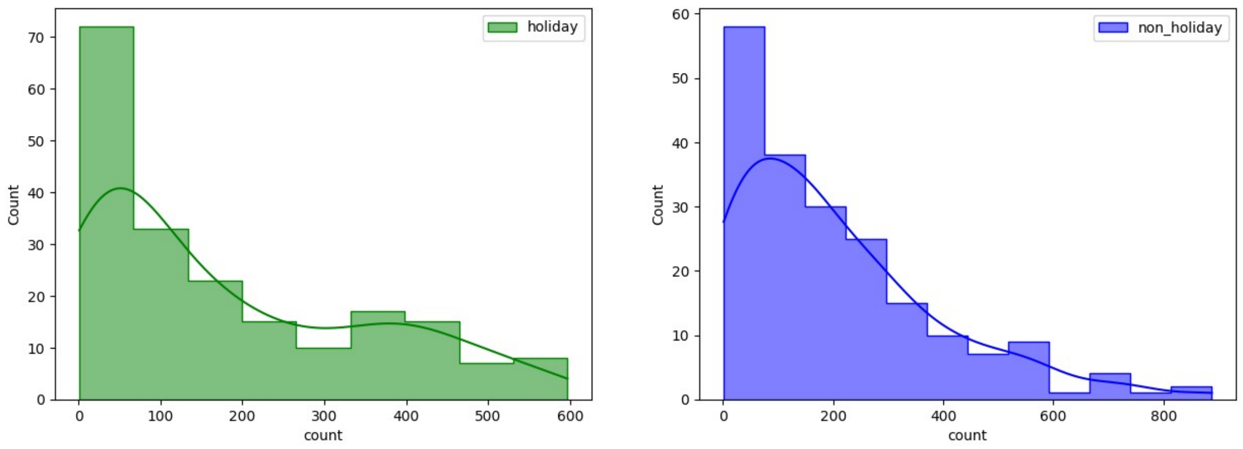
If the assumptions of T Test are met then we can proceed performing T Test for independent samples else we will perform the non parametric test equivalent to T Test for independent sample i.e., Mann-Whitney U rank test for two independent samples. STEP-4: Compute the p-value and fix value of alpha.

We set our alpha to be 0.05 STEP-5: Compare p-value and alpha.

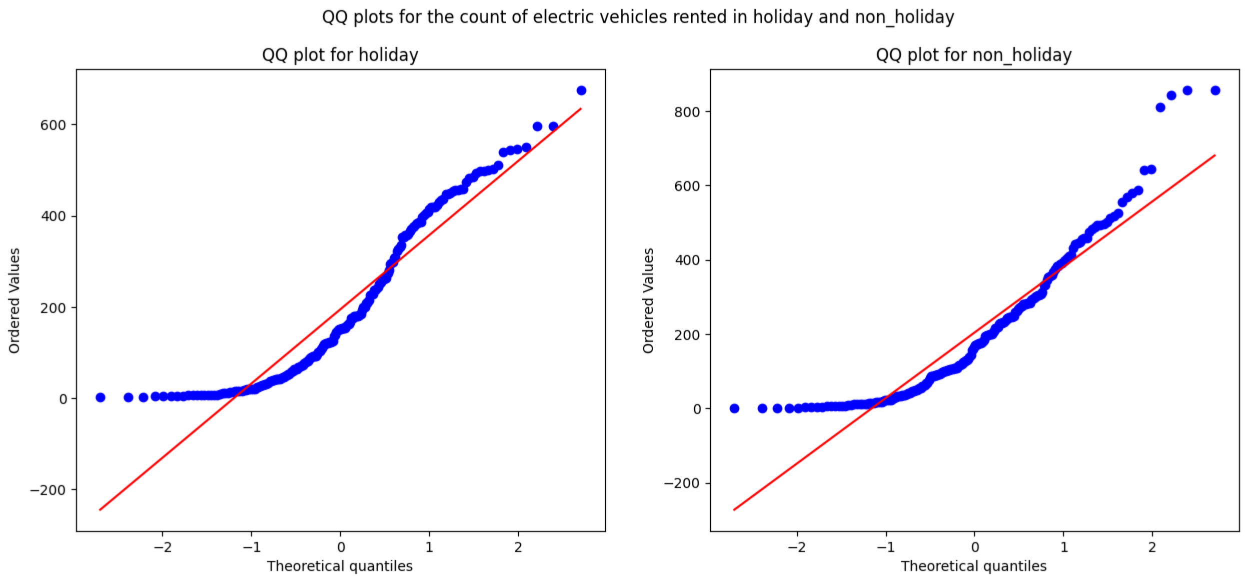
Based on p-value, we will accept or reject H0.

p-val > alpha : Accept H0 p-val < alpha : Reject H0

```
plt.figure(figsize = (15, 5))
plt.subplot(1, 2, 1)
sns.histplot(df.loc[df['holiday'] == 1, 'count'].sample(200),
             element = 'step', color = 'green', kde = True, label =
'holiday')
plt.legend()
plt.subplot(1, 2, 2)
sns.histplot(df.loc[df['holiday'] == 0, 'count'].sample(200),
             element = 'step', color = 'blue', kde = True, label =
'non_holiday')
plt.legend()
plt.plot()

[]
```



```
plt.figure(figsize = (15, 6))
plt.subplot(1, 2, 1)
plt.suptitle('QQ plots for the count of electric vehicles rented in
holiday and non_holiday')
spy.probplot(df.loc[df['holiday'] == 1, 'count'].sample(200), plot =
plt, dist = 'norm')
plt.title('QQ plot for holiday')
plt.subplot(1, 2, 2)
spy.probplot(df.loc[df['holiday'] == 0, 'count'].sample(200), plot =
plt, dist = 'norm')
plt.title('QQ plot for non_holiday')
plt.plot()

[]
```



Insights

```
The data is given from Timestamp('2011-01-01 00:00:00') to
Timestamp('2012-12-19 23:00:00'). The total time period for which the
data is given is '718 days 23:00:00'.

Out of every 100 users, around 19 are casual users and 81 are
registered users.

The mean total hourly count of rental bikes is 144 for the year 2011
and 239 for the year 2012. An annual growth rate of 65.41 % can be
seen in the demand of electric vehicles on an hourly basis.

There is a seasonal pattern in the count of rental bikes, with higher
demand during the spring and summer months, a slight decline in the
fall, and a further decrease in the winter months.

The average hourly count of rental bikes is the lowest in the month of
January followed by February and March.

There is a distinct fluctuation in count throughout the day, with low
counts during early morning hours, a sudden increase in the morning, a
peak count in the afternoon, and a gradual decline in the evening and
nighttime.
More than 80 % of the time, the temperature is less than 28 degrees
celcius.

More than 80 % of the time, the humidity value is greater than 40.
Thus for most of the time, humidity level varies from optimum to too
moist.

More than 85 % of the total, windspeed data has a value of less than
```

```
20.

The hourly count of total rental bikes is the highest in the clear and
cloudy weather, followed by the misty weather and rainy weather. There
are very few records for extreme weather conditions.

The mean hourly count of the total rental bikes is statistically
similar for both working and non- working days.

There is statistically significant dependency of weather and season
based on the hourly total number of bikes rented.

The hourly total number of rental bikes is statistically different for
different weathers.

There is no statistically significant dependency of weather 1, 2, 3 on
season based on the average hourly total number of bikes rented.

The hourly total number of rental bikes is statistically different for
different seasons.
```

Recommendations

Seasonal Marketing: Since there is a clear seasonal pattern in the count of rental bikes, Yulu can adjust its marketing strategies accordingly. Focus on promoting bike rentals during the spring and summer months when there is higher demand. Offer seasonal discounts or special packages to attract more customers during these periods.

Time-based Pricing: Take advantage of the hourly fluctuation in bike rental counts throughout the day. Consider implementing time-based pricing where rental rates are lower during off-peak hours and higher during peak hours. This can encourage customers to rent bikes during less busy times, balancing out the demand and optimizing the resources.

Weather-based Promotions: Recognize the impact of weather on bike rentals. Create weather-based promotions that target customers during clear and cloudy weather, as these conditions show the highest rental counts. Yulu can offer weather-specific discounts to attract more customers during these favorable weather conditions.

User Segmentation: Given that around 81% of users are registered, and the remaining 19% are casual, Yulu can tailor its marketing and communication strategies accordingly. Provide loyalty programs, exclusive offers, or personalized recommendations for registered users to encourage repeat business. For casual users, focus on providing a seamless rental experience and promoting the benefits of bike rentals for occasional use.

Optimize Inventory: Analyze the demand patterns during different months and adjust the inventory accordingly. During months with lower rental counts such as January, February, and March, Yulu can optimize its inventory levels to avoid excess bikes. On the other hand, during peak months, ensure having sufficient bikes available to meet the higher demand.

Improve Weather Data Collection: Given the lack of records for extreme weather conditions, consider improving the data collection process for such scenarios. Having more data on extreme weather conditions can help to understand customer behavior and adjust the operations accordingly, such as offering specialized bike models for different weather conditions or implementing safety measures during extreme weather.

Collaborations with Weather Services: Consider collaborating with weather services to provide real-time weather updates and forecasts to potential customers. Incorporate weather information into your marketing campaigns or rental app to showcase the ideal biking conditions and attract users who prefer certain weather conditions.

Seasonal Bike Maintenance: Allocate resources for seasonal bike maintenance. Before the peak seasons, conduct thorough maintenance checks on the bike fleet to ensure they are in top condition. Regularly inspect and service bikes throughout the year to prevent breakdowns and maximize customer satisfaction.

Customer Feedback and Reviews: Encourage customers to provide feedback and reviews on their biking experience. Collecting feedback can help identify areas for improvement, understand customer preferences, and tailor the services to better meet customer expectations.

Social Media Marketing: Leverage social media platforms to promote the electric bike rental services. Share captivating visuals of biking experiences in different weather conditions, highlight customer testimonials, and engage with potential customers through interactive posts and contests. Utilize targeted advertising campaigns to reach specific customer segments and drive more bookings.

Special Occasion Discounts: Since Yulu focusses on providing a sustainable solution for vehicular pollution, it should give special discounts on the occassions like Zero Emissions Day (21st September), Earth day (22nd April), World Environment Day (5th June) etc in order to attract new users.