Introduction of 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.

Problem Statement :- 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.

Project by :- Om Mishra

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

Dataset description

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

data.head()
```

{"summary":"{\n \"name\": \"data\",\n \"rows\": 10886,\n \"fields\": [\n   {\n   \"column\": \"datetime\",\n   \"properties\": {\n   \"dtype\": \"object\",\n   \"num_unique_values\": 10886,\n   \"samples\": [\n   \"2011-07-19 11:00:00\",\n   \"2012-01-16 06:00:00\",\n   \"2011-12-11 18:00:00\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"season\",\n   \"properties\": {\n   \"dtype\":

```
\"number\",\n        \"std\": 1,\n        \"min\": 1,\n
\"max\": 4,\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          2,\n          4,\n          1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"holiday\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 0,\n
\"min\": 0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1,\n          0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"workingday\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1,\n
0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"weather\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 0,\n        \"min\": 1,\n        \"max\": 4,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          2,\n
4\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"temp\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 7.791589843987567,\n        \"min\": 0.82,\n        \"max\":
41.0,\n        \"num_unique_values\": 49,\n        \"samples\": [\n
6.56,\n          1.64\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"atemp\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 8.474600626484948,\n        \"min\": 0.76,\n        \"max\":
45.455,\n        \"num_unique_values\": 60,\n        \"samples\": [\n
14.395,\n          16.665\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"humidity\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 19,\n        \"min\": 0,\n
\"max\": 100,\n        \"num_unique_values\": 89,\n
\"samples\": [\n          29,\n          61\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"windspeed\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
8.164537326838689,\n        \"min\": 0.0,\n        \"max\": 56.9969,\n
\"num_unique_values\": 28,\n        \"samples\": [\n
22.0028,\n          43.0006\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"casual\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 49,\n        \"min\": 0,\n
\"max\": 367,\n        \"num_unique_values\": 309,\n
\"samples\": [\n          287,\n          47\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"registered\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
151,\n        \"min\": 0,\n        \"max\": 886,\n
```

\"num_unique_values\": 731,\n   \"samples\": [\n   566,\n
9\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"count\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 181,\n   \"min\": 1,\n   \"max\": 977,\n
\"num_unique_values\": 822,\n   \"samples\": [\n   626,\n
256\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"data"}

```
# Structure of the given data
print(data.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
None
```

```
# Number of entries in dataframe
print(len(data))
```

```
10886
```

```
#Summary of DataFrame
print(data.describe())
```

```
             season       holiday    workingday       weather
temp  \
count  10886.000000  10886.000000  10886.000000  10886.000000
10886.00000
mean       2.506614      0.028569      0.680875      1.418427
20.23086
std        1.116174      0.166599      0.466159      0.633839
7.79159
min        1.000000      0.000000      0.000000      1.000000
```

```
0.82000
25%        2.000000     0.000000     0.000000     1.000000
13.94000
50%        3.000000     0.000000     1.000000     1.000000
20.50000
75%        4.000000     0.000000     1.000000     2.000000
26.24000
max        4.000000     1.000000     1.000000     4.000000
41.00000

              atemp      humidity     windspeed        casual
registered  \
count  10886.000000  10886.000000  10886.000000  10886.000000
10886.000000
mean      23.655084     61.886460     12.799395     36.021955
155.552177
std        8.474601     19.245033      8.164537     49.960477
151.039033
min        0.760000      0.000000      0.000000      0.000000
0.000000
25%       16.665000     47.000000      7.001500      4.000000
36.000000
50%       24.240000     62.000000     12.998000     17.000000
118.000000
75%       31.060000     77.000000     16.997900     49.000000
222.000000
max       45.455000    100.000000     56.996900    367.000000
886.000000

              count
count  10886.000000
mean     191.574132
std      181.144454
min        1.000000
25%       42.000000
50%      145.000000
75%      284.000000
max      977.000000
```

```python
# Checking for missing values
print(data.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
```

```
# Checking the timeframe of all values taken
data['datetime'] = pd.to_datetime(data['datetime'])
print("The earliest entry is from", min(data['datetime']))
print("The latest entry is from", max(data['datetime']))
data['datetime'].max() - data['datetime'].min()
```

```
The earliest entry is from 2011-01-01 00:00:00
The latest entry is from 2012-12-19 23:00:00
```

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

```
# Converting categorical variables to 'category' to optimize memory
usage and enable smoother analysis
data['season'] = data['season'].astype('category')
data['holiday'] = data['holiday'].astype('category')
data['workingday'] = data['workingday'].astype('category')
data['weather'] = data['weather'].astype('category')
```

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

```
season
4    25.11
2    25.11
3    25.11
1    24.67
Name: proportion, dtype: float64
```

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

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

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

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

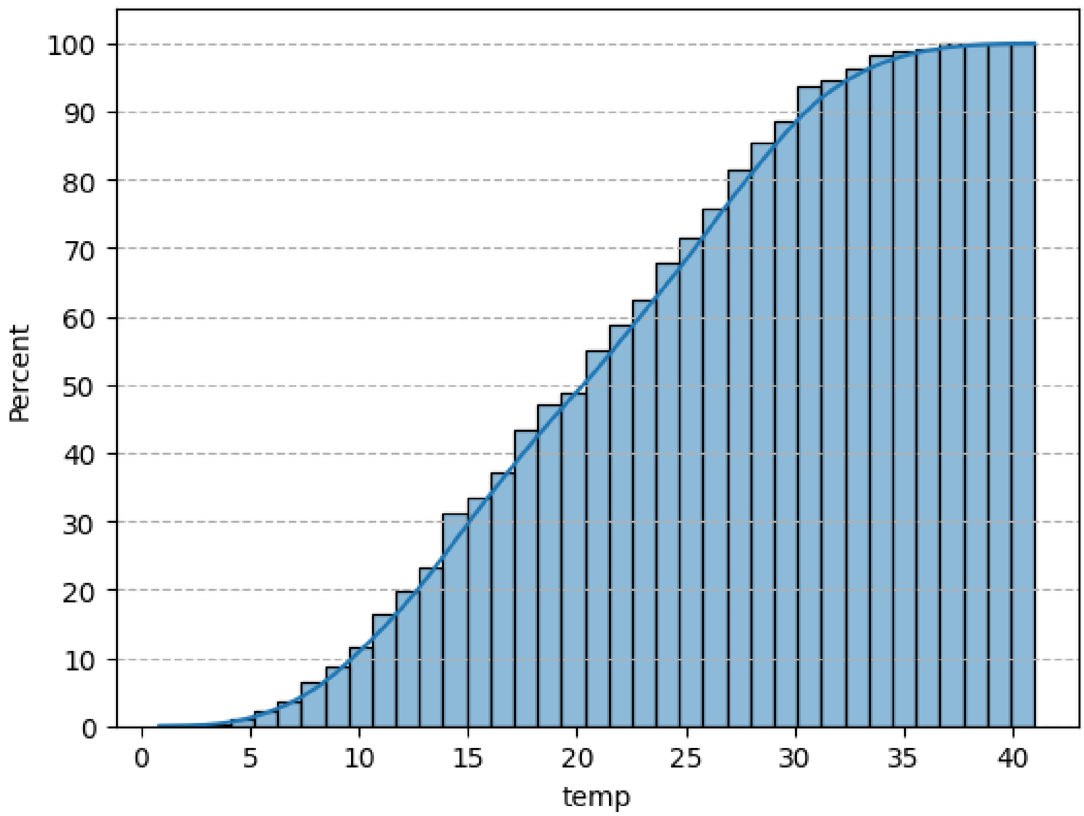
```
np.round(data['weather'].value_counts(normalize = True) * 100, 2)
```

```
weather
1    66.07
2    26.03
```

```
3      7.89
4      0.01
Name: proportion, dtype: float64
```

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

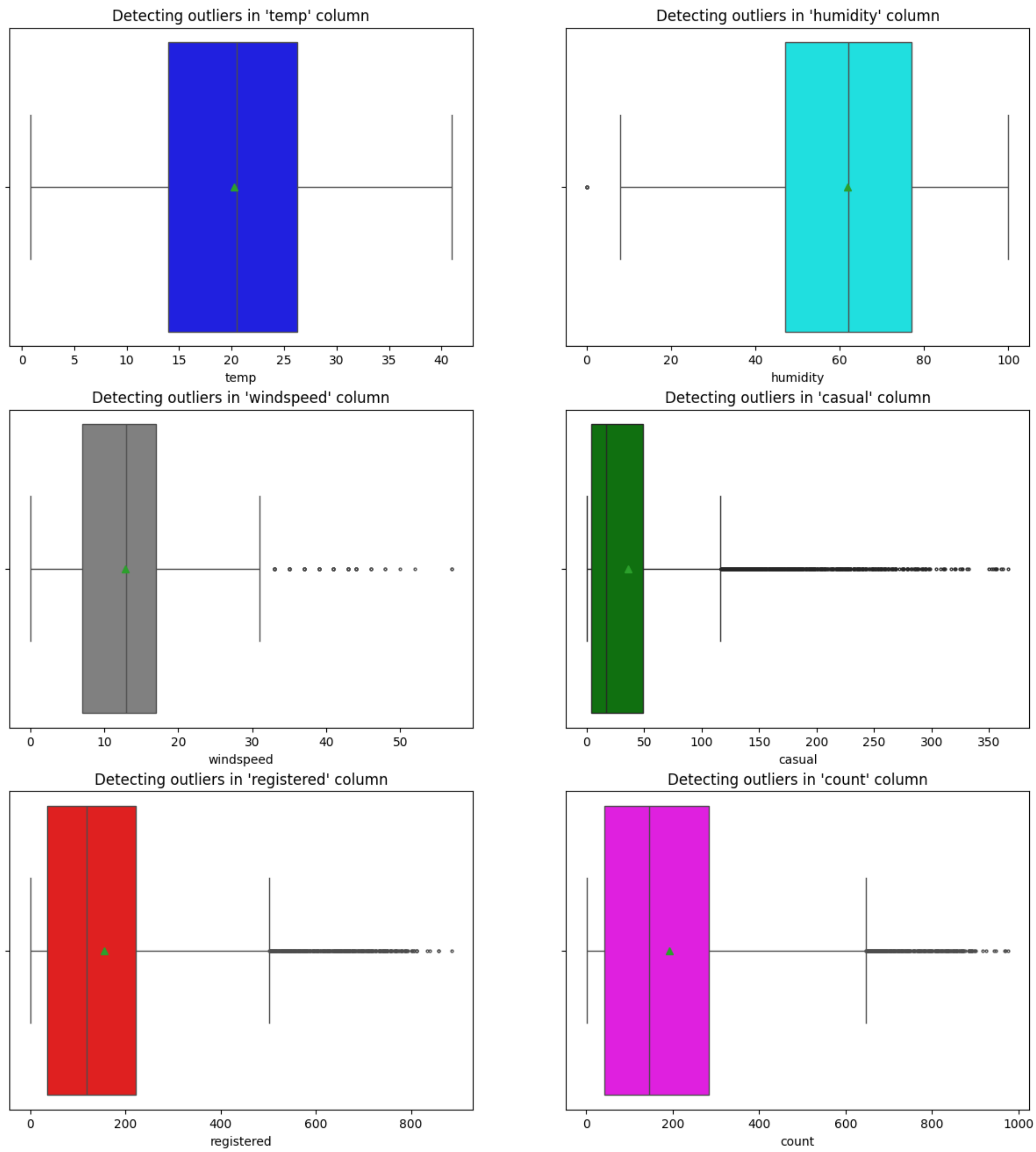
```
[]
```



Checking for Outliars

```
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")
```

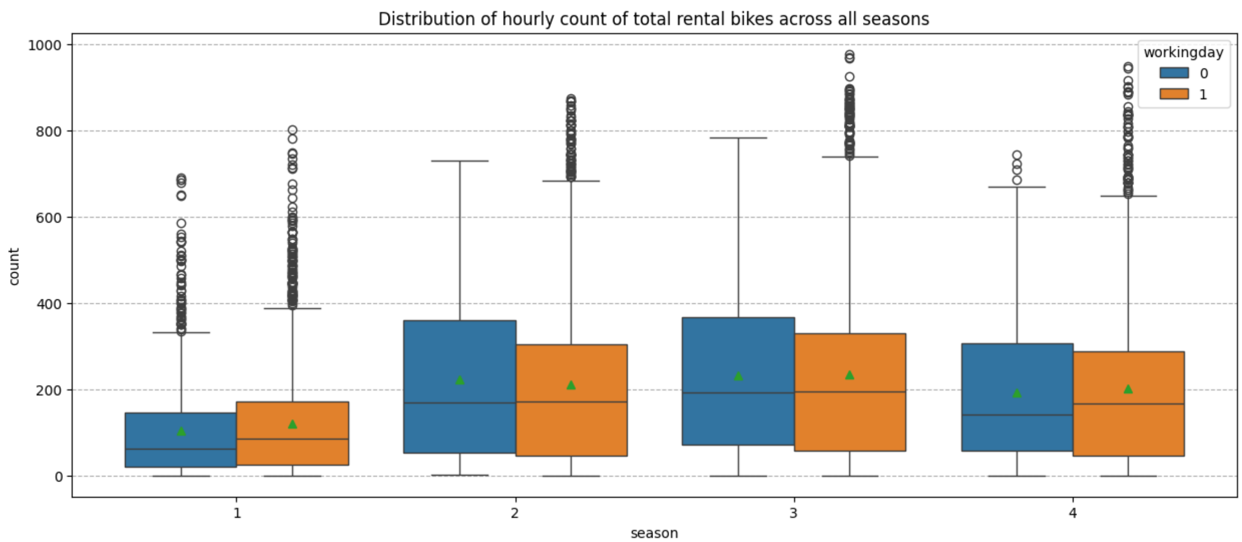
```
 sns.boxplot(data = data, x = data[i], color = colors[count - 1],
showmeans = True, fliersize = 2)
 plt.plot()
 count += 1
```




```
plt.figure(figsize = (15, 6))
plt.title('Distribution of hourly count of total rental bikes across
all seasons')
```

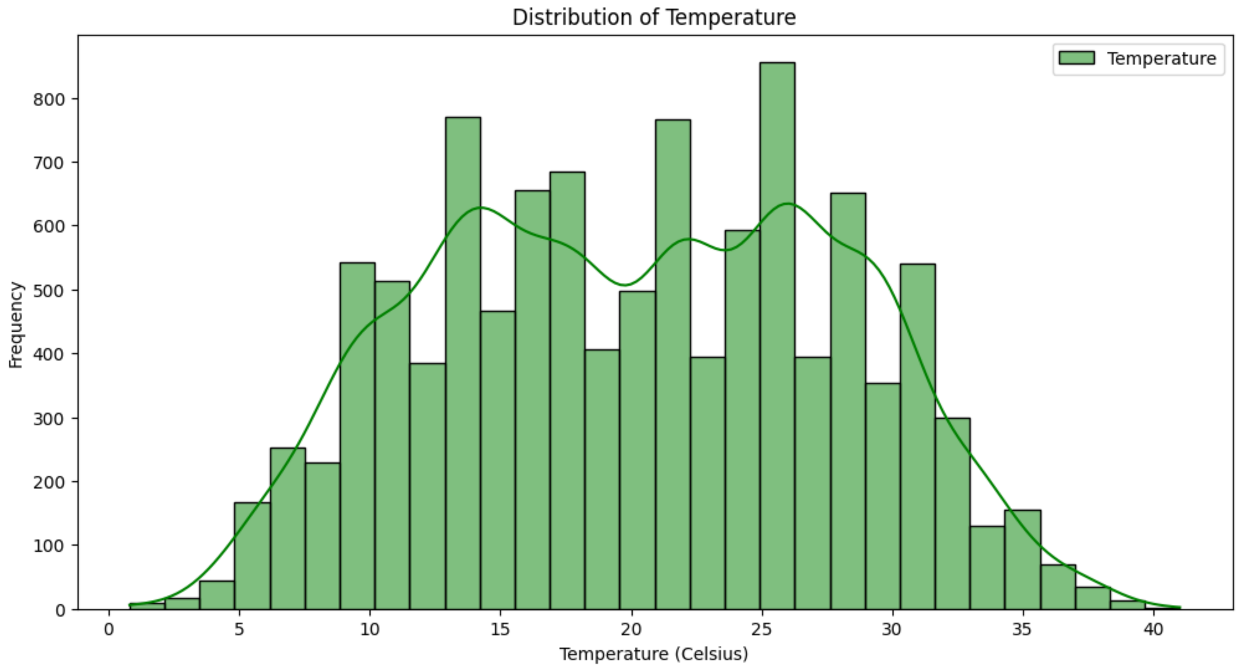
```
sns.boxplot(data = data, x = 'season', y = 'count', hue =
'workingday', showmeans = True)
plt.grid(axis = 'y', linestyle = '--')
plt.plot()

[]
```



Univariate Analysis

```
# Distribution plots of continuous variables
plt.figure(figsize=(12, 6))
sns.histplot(data['temp'], kde=True, bins=30, color='green',
label='Temperature')
plt.title('Distribution of Temperature')
plt.xlabel('Temperature (Celsius)')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```



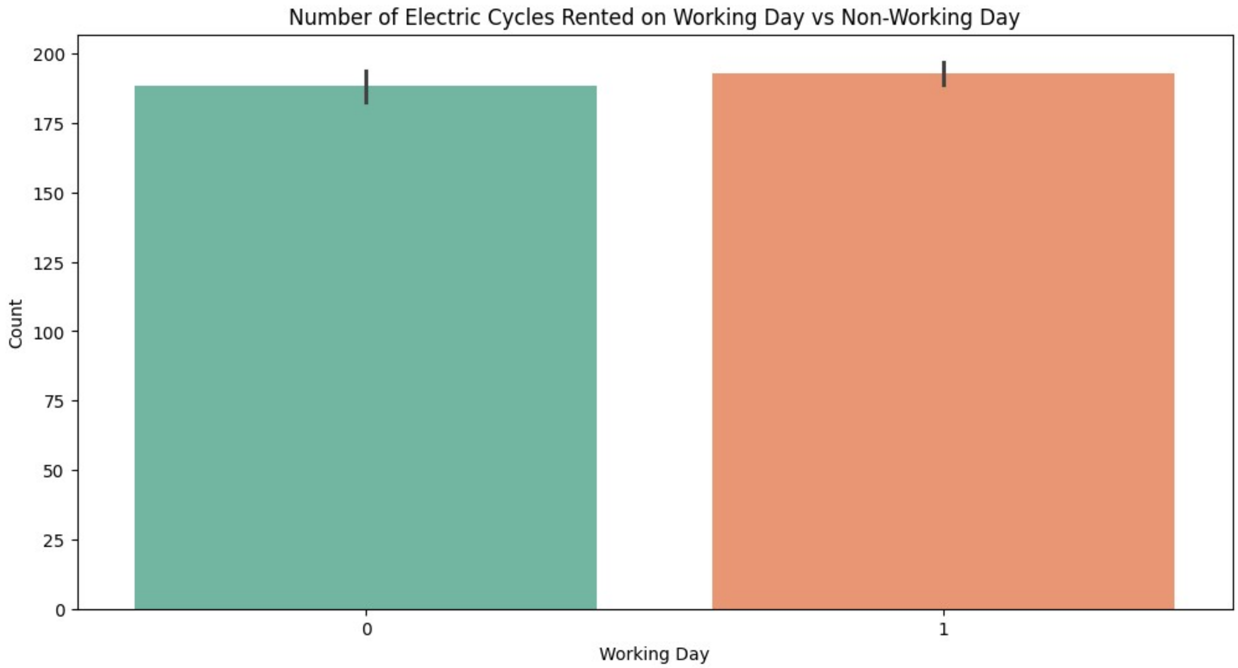
Bivariate Analysis

```
# Relationships between important variables
plt.figure(figsize=(12, 6))
sns.barplot(x='workingday', y='count', data=data, palette='Set2')
plt.title('Number of Electric Cycles Rented on Working Day vs Non-Working Day')
plt.xlabel('Working Day')
plt.ylabel('Count')
plt.show()
```

```
<ipython-input-27-6a8f1403449e>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='workingday', y='count', data=data, palette='Set2')
```

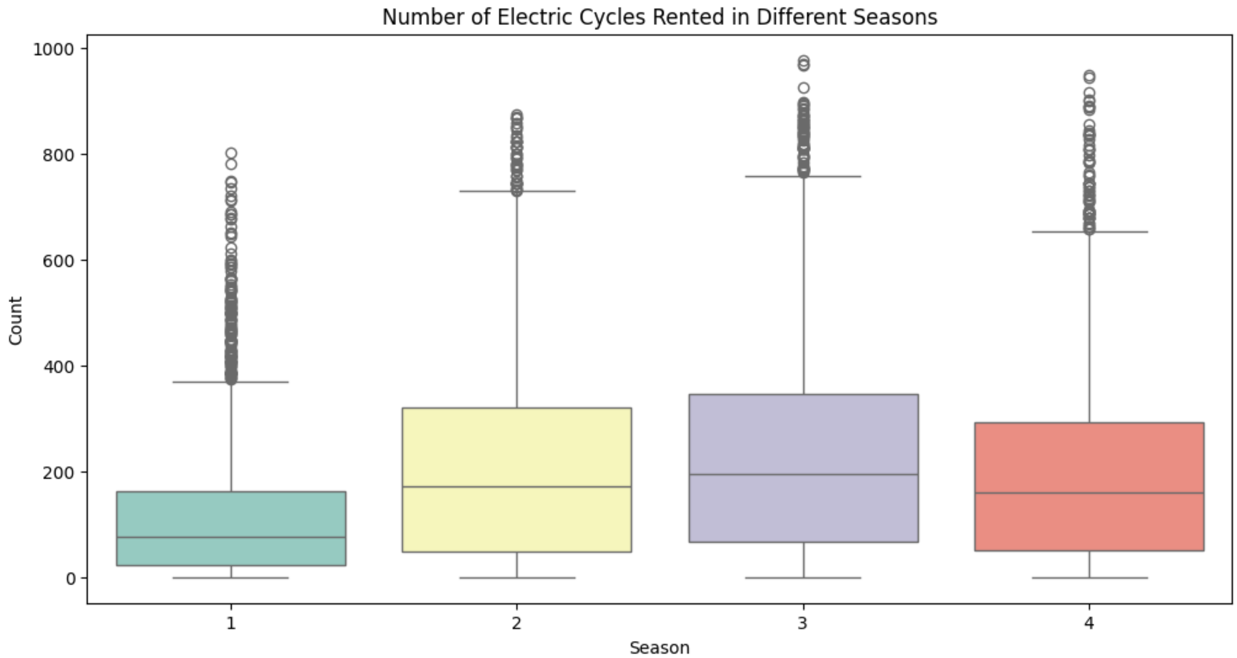



```
plt.figure(figsize=(12, 6))
sns.boxplot(x='season', y='count', data=data, palette='Set3')
plt.title('Number of Electric Cycles Rented in Different Seasons')
plt.xlabel('Season')
plt.ylabel('Count')
plt.show()
```

```
<ipython-input-28-19dabbea4587>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x='season', y='count', data=data, palette='Set3')
```

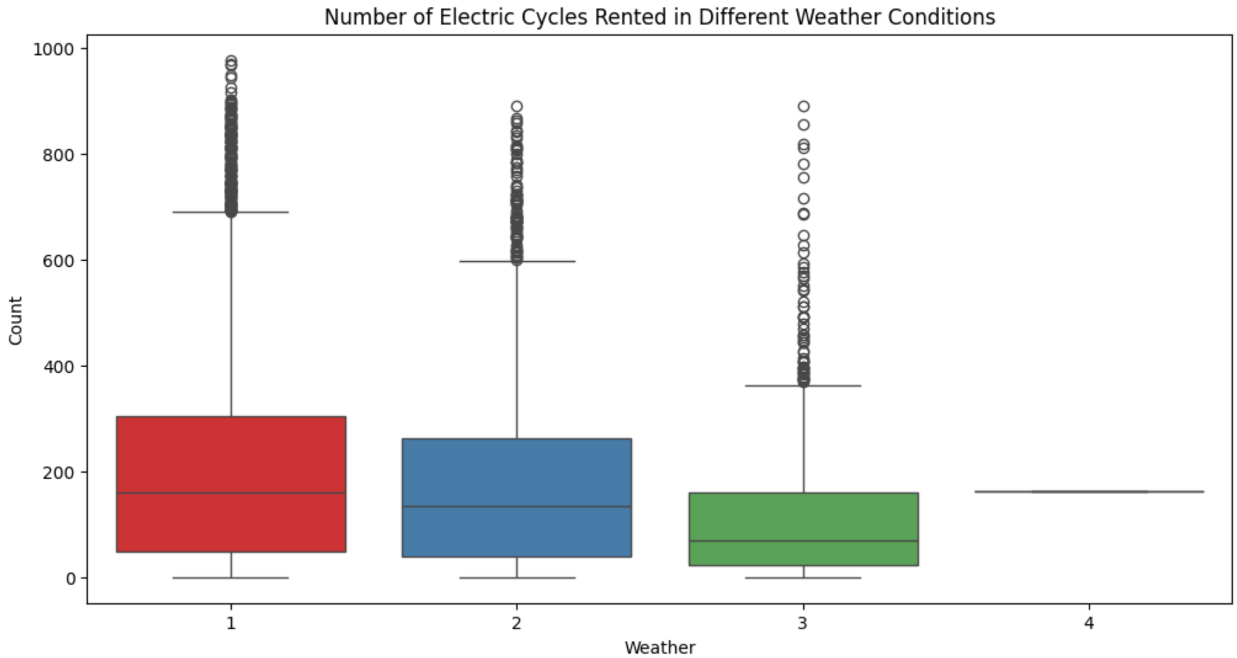



```
plt.figure(figsize=(12, 6))
sns.boxplot(x='weather', y='count', data=data, palette='Set1')
plt.title('Number of Electric Cycles Rented in Different Weather
Conditions')
plt.xlabel('Weather')
plt.ylabel('Count')
plt.show()

<ipython-input-29-8f167b7ea03b>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x='weather', y='count', data=data, palette='Set1')
```



Hypothesis Testing

```
# 2 sample T-Test
# Spliting data into working day and non-working day
working_day = data[data['workingday'] == 1]['count']
non_working_day = data[data['workingday'] == 0]['count']
# Performing 2-sample t-test
t_stat, p_value = ttest_ind(working_day, non_working_day)
# Printing p-value
print("P-value for 2-Sample T-Test:", p_value)

P-value for 2-Sample T-Test: 0.22644804226361348
```

Null Hypothesis (H0): There is no significant difference in the number of electric cycles rented on working days compared to non-working days.

Alternate Hypothesis (H1): There is a significant difference in the number of electric cycles rented on working days compared to non-working days.

Since P-value > 0.05, we fail to reject null hypothesis

ANOVA

```
# Performing ANOVA for weather
weather_groups = [data[data['weather'] == i]['count'] for i in
data['weather'].unique()]
weather_anova = f_oneway(*weather_groups)
```

```
# Printing p-value for weather ANOVA
print("P-value for ANOVA (Weather):", weather_anova.pvalue)

P-value for ANOVA (Weather): 5.482069475935669e-42
```

Null Hypothesis (H0): The number of cycles rented is similar across different weather conditions.

Alternate Hypothesis (H1): The number of cycles rented differs across different weather conditions.

Since P-value > 0.05, we fail to reject the null hypothesis.

```
# Performing ANOVA for season
season_groups = [data[data['season'] == i]['count'] for i in 
data['season'].unique()]
season_anova = f_oneway(*season_groups)
# Printing p-value for season ANOVA
print("P-value for ANOVA (Season):", season_anova.pvalue)

P-value for ANOVA (Season): 6.164843386499654e-149
```

Null Hypothesis (H0): The number of cycles rented is similar across different seasons.

Alternate Hypothesis (H1): The number of cycles rented differs across different seasons.

Since P-value > 0.05, we fail to reject the null hypothesis.

Chi-Square Test

```
# Creating contingency table for weather and season
weather_season_table = pd.crosstab(data['weather'], data['season'])
# Performing Chi-square test
chi2, p, dof, expected = chi2_contingency(weather_season_table)
# Printing p-value for Chi-square test
print("P-value for Chi-square Test:", p)

P-value for Chi-square Test: 1.5499250736864862e-07
```

Null Hypothesis (H0): Weather is independent of the season.

Alternate Hypothesis (H1): Weather is dependent on the season.

Since P-value > 0.05, we fail to reject the null hypothesis.

Data Insights

1. There's a notable annual growth rate of 65.41% in the hourly demand for rental bikes, increasing from 144 in 2011 to 239 in 2012.

2. Temperature is mostly below 28°C, humidity is generally above 40%, and windspeeds are mostly below 20.
3. Out of every 100 users, approximately 19 are casual and 81 are registered.
4. Seasonal trends show higher demand during spring and summer, followed by a decline in fall and winter.
5. January, February, and March exhibit the lowest average hourly bike counts.
6. Throughout the day, there's a distinct fluctuation in bike counts, peaking in the afternoon.
7. Data spans from January 1, 2011, to December 19, 2012, totaling 718 days and 23 hours.
8. Weather and season significantly impact bike rentals, with different weather conditions leading to varied rental counts.
9. The mean hourly bike count remains statistically similar for both working and non-working days.
10. Clear and cloudy weather conditions witness the highest bike rentals, with few records for extreme weather.
11. Bike rentals vary significantly across different seasons.
12. There's no significant dependency of weather on season for certain weather conditions.

Recommendations

1. Optimize inventory based on seasonal demand patterns.
2. Launch weather-based promotions targeting favorable weather conditions.
3. Implement time-based pricing to balance demand throughout the day.
4. Offer special discounts on environmental awareness days to attract new users.
5. Improve data collection for extreme weather conditions.
6. Prioritize seasonal bike maintenance to ensure optimal fleet condition.
7. Capitalize on seasonal trends by adjusting marketing strategies and offering seasonal discounts.
8. Tailor marketing approaches for registered and casual users.
9. Solicit customer feedback to drive service improvements.
10. Utilize social media for targeted marketing and engagement.
11. Enhance customer comfort with amenities and collaborations with weather services.