YULU

November 25, 2024

1 Problem Statement:

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 wan

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

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

[ ]:
```
data.head()
```

[ ]:
```
              datetime  season  holiday  workingday  weather  temp   atemp  \
0  2011-01-01 00:00:00       1        0           0        1  9.84  14.395
1  2011-01-01 01:00:00       1        0           0        1  9.02  13.635
2  2011-01-01 02:00:00       1        0           0        1  9.02  13.635
3  2011-01-01 03:00:00       1        0           0        1  9.84  14.395
4  2011-01-01 04:00:00       1        0           0        1  9.84  14.395

   humidity  windspeed  casual  registered  count
0        81        0.0       3          13     16
1        80        0.0       8          32     40
2        80        0.0       5          27     32
3        75        0.0       3          10     13
4        75        0.0       0           1      1
```

[ ]:
```
data.shape
```

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

[ ]:
```
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
```

[ ]:
```

```

[ ]:
```
data.isna().sum()
```

[ ]:
```
datetime      0
season        0
holiday       0
workingday    0
weather       0
temp          0
atemp         0
humidity      0
windspeed     0
casual        0
registered    0
count         0
dtype: int64
```

[ ]:
```
data.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
```

[ ]:
```
data.duplicated().sum()
```

[ ]: 0

1.1 Dataset Overview

- **Number of Rows:** 10,886
- **Number of Columns:** 12 ### Key Insights

1. **No Missing Values**: All columns have complete data.
2. **No Duplicate Records**: The dataset does not contain duplicates.
3. **Data Types**: season, holiday, workingday, and weather columns were of type int64

1.1.1 Data Types

1.1.2 1. Integer (`int64`)

- `season`
- `holiday`
- `workingday`
- `weather`
- `humidity`
- `casual`
- `registered`
- `count` ### 2. Float (`float64`)
- `temp`
- `atemp`
- `windspeed`

1.1.3 3. Object (`object`)

- `datetime`

2 PROCESSING

[ ]:

```

```

[ ]:

```
processdata=data.copy()
```

[ ]:

```
processdata["datetime"]=pd.to_datetime(processdata["datetime"])
processdata["season"]=pd.Categorical(processdata["season"])
processdata["weather"]=pd.Categorical(processdata["weather"])
processdata["holiday"]=pd.Categorical(processdata["holiday"])
processdata["workingday"]=pd.Categorical(processdata["workingday"])
```

[ ]:

```
processdata.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  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  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: category(4), datetime64[ns](1), float64(3), int64(4)
memory usage: 723.7 KB
```

[ ]:

```
processdata.shape
```

[ ]: (10886, 12)

[ ]:

```
preprocessed_data=processdata.copy()
```

[ ]:

```
dataOutliers=processdata.copy().select_dtypes(include=np.number)
dataOutliers.head()
```

[ ]:

	temp	atemp	humidity	windspeed	casual	registered	count
0	9.84	14.395	81	0.0	3	13	16
1	9.02	13.635	80	0.0	8	32	40
2	9.02	13.635	80	0.0	5	27	32

```
3  9.84  14.395        75         0.0        3            10       13
4  9.84  14.395        75         0.0        0             1        1
```

[ ]:

```
```

[ ]:

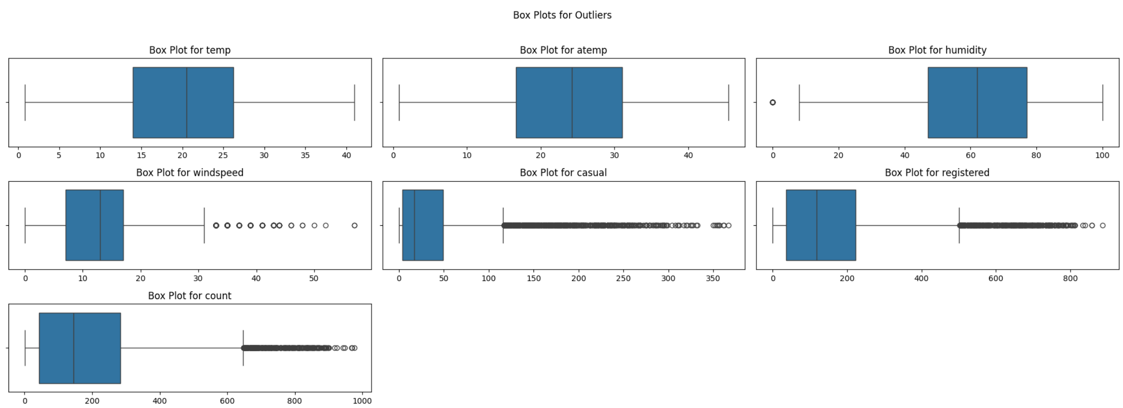
```
dataOutliers.columns
```

[ ]:

```
Index(['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered',
       'count'],
      dtype='object')
```

[ ]:

```
# @title
fig, axes = plt.subplots(3, 3, figsize=(20, 7))  # Enough space for 7 columns
axes = axes.flatten()
# Plot boxplots for each column
for i, col in enumerate(dataOutliers.columns):
    sns.boxplot(data=dataOutliers, x=col, ax=axes[i])
    axes[i].set_title(f'Box Plot for {col}')
    axes[i].set_xlabel('')

for j in range(len(dataOutliers.columns), len(axes)):
    axes[j].set_visible(False)
plt.tight_layout()
plt.suptitle('Box Plots for Outliers', y=1.02)

# Adjust layout
plt.tight_layout()
plt.show()
```



[ ]:

```
q1=dataOutliers.quantile(0.25)
q3=dataOutliers.quantile(0.75)
iqr=q3-q1
upper_bound=q3+1.5*iqr
```

```
lower_bound=q1-1.5*iqr
```

[ ]:
```
iqr
```

[ ]:
```
temp           12.3000
atemp          14.3950
humidity       30.0000
windspeed       9.9964
casual         45.0000
registered    186.0000
count         242.0000
dtype: float64
```

[ ]:
```
lower_bound
```

[ ]:
```
temp           -4.5100
atemp          -4.9275
humidity        2.0000
windspeed      -7.9931
casual        -63.5000
registered   -243.0000
count        -321.0000
dtype: float64
```

[ ]:
```
upper_bound
```

[ ]:
```
temp           44.6900
atemp          52.6525
humidity      122.0000
windspeed      31.9925
casual        116.5000
registered    501.0000
count         647.0000
dtype: float64
```

[ ]:
```
dataOutliers.shape
```

[ ]:
```
(10886, 7)
```

[ ]:
```
preprocessed_data.shape
```

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

[ ]:
```
mask= ~((dataOutliers < lower_bound) | (dataOutliers > upper_bound)).any(axis=1)
```

[ ]:
```
preprocessed_data = preprocessed_data[mask]
```

[ ]:
```
preprocessed_data.head()
```

[ ]:
```
             datetime  season  holiday  workingday  weather  temp   atemp  \
0 2011-01-01 00:00:00       1        0           0        1  9.84  14.395
1 2011-01-01 01:00:00       1        0           0        1  9.02  13.635
2 2011-01-01 02:00:00       1        0           0        1  9.02  13.635
3 2011-01-01 03:00:00       1        0           0        1  9.84  14.395
4 2011-01-01 04:00:00       1        0           0        1  9.84  14.395

   humidity  windspeed  casual  registered  count
0        81        0.0       3          13     16
1        80        0.0       8          32     40
2        80        0.0       5          27     32
3        75        0.0       3          10     13
4        75        0.0       0           1      1
```

[ ]:
```
preprocessed_data.shape
```

[ ]:
```
(9518, 12)
```

[ ]:
```
preprocessed_data.columns
```

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

[ ]:
```
preprocessed_data["season"]=preprocessed_data["season"].map({1:"spring",2:
 ↪"summer",3:"fall",4:"winter"})
```

[ ]:
```
preprocessed_data["season"].value_counts()
```

[ ]:
```
season
winter    2475
spring    2463
summer    2292
fall      2288
Name: count, dtype: int64
```

2.1 Insights

These Converted to Categorical (`category`) - `season` - `holiday` - `workingday` - `weather`

2.2 Data Cleaning

- **Outliers Removed:** Based on IQR (Interquartile Range), resulting in **9,518 rows** after cleaning. all columns outliers are removed

[ ]:
```

```

[ ]:

```

```

3 NON-GRAPHIC ANALYSIS

[ ]:

```
dataw=preprocessed_data.copy()
```

[ ]:

```
dataw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9518 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    9518 non-null   datetime64[ns]
 1   season      9518 non-null   category
 2   holiday     9518 non-null   category
 3   workingday  9518 non-null   category
 4   weather     9518 non-null   category
 5   temp        9518 non-null   float64
 6   atemp       9518 non-null   float64
 7   humidity    9518 non-null   int64
 8   windspeed   9518 non-null   float64
 9   casual      9518 non-null   int64
 10  registered  9518 non-null   int64
 11  count       9518 non-null   int64
dtypes: category(4), datetime64[ns](1), float64(3), int64(4)
memory usage: 707.1 KB
```

[ ]:

```
dataw.describe()
```

[ ]:

```
                            datetime         temp        atemp     humidity  \
count                           9518  9518.000000  9518.000000  9518.000000
mean   2011-12-17 13:43:08.611052800    19.589971    22.987399    63.737025
min              2011-01-01 00:00:00     0.820000     0.760000     8.000000
25%              2011-06-15 06:15:00    13.120000    15.910000    49.000000
50%              2011-12-10 19:30:00    18.860000    22.725000    64.500000
75%              2012-06-13 06:45:00    26.240000    30.305000    79.000000
max              2012-12-19 23:00:00    41.000000    45.455000   100.000000
std                              NaN     7.686871     8.361526    18.693175

         windspeed       casual   registered        count
count  9518.000000  9518.000000  9518.000000  9518.000000
mean     12.133336    23.955033   126.181025   150.136058
min       0.000000     0.000000     0.000000     1.000000
25%       7.001500     3.000000    28.250000    34.000000
50%      11.001400    13.000000   101.000000   122.000000
```

```
75%      16.997900   37.000000   187.000000   231.000000
max      31.000900  116.000000   501.000000   590.000000
std       7.437481   26.956046   114.116911   131.586548
```

`temp` Average recorded temperature (`mean`: 19.59°C) ranges from 0.82°C to 41.0°C.

`atemp` temperature (`mean`: 22.99°C) ranges from 0.76°C to 45.46°C.

`humidity` Average humidity is 63.74% and ranges from 8% to 100%.

`windspeed` Wind speed (`mean`: 12.13 m/s) ranges from 0 to 31 m/s.

[ ]:
```python
dataw.describe(include="category")
```

[ ]:

	season	holiday	workingday	weather
count	9518	9518	9518	9518
unique	4	2	2	4
top	winter	0	1	1
freq	2475	9264	6790	6176

[ ]:
```python

```

[ ]:
```python
dataw.groupby("season",observed=False)["count"].agg(["mean","sum"])
```

[ ]:

season	mean	sum
spring	103.164028	254093
summer	160.360820	367547
fall	177.151661	405323
winter	162.437172	402032

3.0.1 Key Insights:

1. **Fall (Season 3)**:
   - **Highest demand** for bike rentals with an average of **177.15 rentals/day**.
   - Total rentals in Fall: **405,323**, the **highest among all seasons**.
2. **Spring (Season 1)**:
   - **Lowest demand** with an average of **103.16 rentals/day**.
   - Total rentals in Spring: **254,093**, the **lowest among all seasons**.
3. **Summer (Season 2)**:
   - Bike rentals increase significantly compared to Spring, with a mean of **160.36 rentals/day**.
   - Total rentals in Summer: **367,547**.
4. **Winter (Season 4)**:
   - Winter also sees high demand with an average of **162.44 rentals/day**.
   - Total rentals in Winter: **402,032**, slightly less than Fall.

- Bike rentals peak in **Fall** (Season 3), followed closely by **Winter** (Season 4) and **Summer** (Season 2).

- Rentals are **lowest in Spring (Season 1)**, suggesting reduced demand, possibly due to weather conditions or other seasonal factors.

```
[ ]: dataw.groupby("workingday",observed=False)["count"].agg(["mean","sum"])
```

```
[ ]:                   mean      sum
     workingday
     0           120.681085   329218
     1           161.970103  1099777
```

3.0.2 Key Insights:

1. **Working Days (1)**:
   - **Higher demand** for bike rentals with an average of **161.97 rentals/day**.
   - Total rentals on working days: **1,099,777**, which is significantly higher than on non-working days.
2. **Non-working Days (0)**:
   - Bike rentals are lower on non-working days, with an average of **120.68 rentals/day**.
   - Total rentals on non-working days: **329,218**.

```
[ ]: data.columns
```

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

```
[ ]: dataw.groupby("weather",observed=False)["count"].agg(["mean","sum"])
```

```
[ ]:                mean     sum
     weather
     1        157.522021  972856
     2        146.805685  376997
     3        102.170763   78978
     4        164.000000     164
```

3.0.3 Key Insights:

1. **Weather Condition 1 (Clear, Few Clouds, Partly Cloudy)**:
   - The highest total rentals at **972,856**.
   - Mean rentals: **157.52/day**, indicating favorable weather drives demand.
2. **Weather Condition 2 (Mist, Cloudy)**:
   - Mean rentals: **146.81/day**, slightly lower than clear weather.
   - Total rentals: **376,997**, suggesting moderate demand in misty/cloudy conditions.
3. **Weather Condition 3 (Light Rain/Snow)**:
   - The lowest demand among significant categories, with mean rentals at **102.17/day**.
   - Total rentals: **78,978**, showing a sharp drop in usage during light rain or snow.
4. **Weather Condition 4 (Heavy Rain/Snow, Severe)**:
   - Rare weather condition with very few data points (Total: **164 rentals**).

- Mean rentals: **164/day**,

[ ]:
```python
dataw.columns
```

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

[ ]:
```python
dataw.groupby(["workingday", "season"], observed=False).agg(
    Total_Ev_Count=("count", "sum"),
    Average_Count=("count", "mean"),
    Average_Temp=("temp", "mean"),
    Average_aTemp=("atemp", "mean"),
    Average_Humidity=("humidity", "mean"),
    Average_Wind_Speed=("windspeed", "mean")
)
```

[ ]:
```
                    Total_Ev_Count  Average_Count  Average_Temp  Average_aTemp  \
workingday season
0          spring            63321      82.772549     11.823007      14.368039
           summer            71974     121.372681     21.230118      24.968027
           fall              82933     136.402961     27.513158      31.543372
           winter           110990     145.656168     14.719108      18.009403
1          spring           190772     112.351001     12.363746      15.154988
           summer           295573     173.968805     22.685862      26.525853
           fall             322390     191.898810     28.716595      32.349048
           winter           291042     169.901926     16.986877      20.401602

                    Average_Humidity  Average_Wind_Speed
workingday season
0          spring          58.771242           14.555930
           summer          67.669477           10.511294
           fall            72.003289           10.638498
           winter          67.492126           10.862485
1          spring          57.289753           13.052239
           summer          62.284285           13.497531
           fall            64.040476           11.039546
           winter          67.523059           11.517656
```

3.0.4 Key Insights:

1. **Seasonal Trends (Non-Working Days)**:
   - Highest rentals in **Winter** with an average of **145.66 rentals/day**.
   - Rentals gradually increase from Spring to Winter.
   - Fall has the highest **average temperature** (27.51°C) and humidity (72%).
2. **Seasonal Trends (Working Days)**:
   - Highest rentals in **Fall** with an average of **191.90 rentals/day**, closely followed by Summer.

- Winter and Spring have slightly lower averages compared to Fall and Summer.

[ ]:
```
dataw.groupby(["workingday", "weather"], observed=False).agg(
    Total_Count=("count", "sum"),
    Average_Count=("count", "mean"),
    Average_Temp=("temp", "mean"),
    Average_aTemp=("atemp", "mean"),
    Average_Humidity=("humidity", "mean"),
    Average_Wind_Speed=("windspeed", "mean")
)
```

[ ]:
```
                    Total_Count  Average_Count  Average_Temp  Average_aTemp  \
workingday weather
0          1             220488     124.148649     18.388592      21.755954
           2              89694     118.642857     17.836085      21.159914
           3              19036      97.122449     17.529592      20.732934
           4                  0            NaN           NaN            NaN
1          1             752368     170.992727     20.405886      23.836408
           2             287303     158.555740     19.637914      23.113678
           3              59942     103.885615     19.932964      23.087340
           4                164     164.000000      8.200000      11.365000

                    Average_Humidity  Average_Wind_Speed
workingday weather
0          1               61.278153           11.715984
           2               72.854497           11.688132
           3               83.607143           12.602495
           4                     NaN                 NaN
1          1               57.398864           12.356020
           2               69.022075           11.830329
           3               84.306759           13.105966
           4               86.000000            6.003200
```

[ ]:
```
dataw.
↪pivot_table(values="count",index=["workingday","weather"],aggfunc=["sum","mean"],observed=F
```

[ ]:
```
                       sum        mean
                     count       count
workingday weather
0          1        220488  124.148649
           2         89694  118.642857
           3         19036   97.122449
           4             0         NaN
1          1        752368  170.992727
           2        287303  158.555740
           3         59942  103.885615
           4           164  164.000000
```

[ ]:
```
pd.crosstab(
    dataw["weather"],
    dataw["season"],
    values=dataw["count"],
    aggfunc="sum",
    margins=True
)
```

[ ]:

```
season   spring  summer    fall  winter      All
weather
1        175925  255490  290261  251180   972856
2         66525   89774   94779  125919   376997
3         11479   22283   20283   24933    78978
4           164       0       0       0      164
All      254093  367547  405323  402032  1428995
```

[ ]:
```
pd.crosstab(dataw["workingday"],dataw["season"])
```

[ ]:

```
season      spring  summer  fall  winter
workingday
0              765     593   608     762
1             1698    1699  1680    1713
```

[ ]:
```
dataw.
 ↪pivot_table(values="count",index=["weather"],columns="season",aggfunc=["sum","mean"],observ
```

[ ]:

```
            sum                                mean                          \
season   spring  summer    fall  winter      spring      summer        fall
weather
1        175925  255490  290261  251180  110.297806  173.448744  181.640175
2         66525   89774   94779  125919   97.401171  146.211726  183.324952
3         11479   22283   20283   24933   62.385870  108.697561  117.242775
4           164       0       0       0  164.000000         NaN         NaN


season       winter
weather
1        166.344371
2        167.001326
3        118.165877
4               NaN
```

weather: 1: Clear, Few clouds, partly cloudy, partly cloudy

2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist

3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds

4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

3.1 1. Weather Condition Insights

- **Weather 1:**
  - **Highest rentals** across all seasons, both in total (`sum`) and average (`mean`).
  - Mean rentals peak during **Summer (173.45)** and **Fall (181.64)** under clear weather.
  - **Winter** also shows strong demand with an average of **166.34 rentals/day** under Weather 1.
- **Weather 2:**
  - Rentals are moderate, with higher averages in **Fall (183.32)** and **Winter (167.00)** compared to other seasons.
  - Consistent demand under misty/cloudy weather highlights its acceptability for users.
- **Weather 3:**
  - Significant drop in rentals compared to clear or misty conditions.
  - Mean rentals are highest in **Summer (108.69)** and lowest in **Spring (62.39)**.
  - Total rentals remain low under light rain/snow, but there is still a notable user base.
- **Weather 4**
  - **Minimal demand**, with only **164 rentals recorded in Spring**.
  - No rentals observed in Summer, Fall, or Winter for this condition, showing its strong negative impact on demand.

[ ]:
```
dataw.shape
```

[ ]:
```
(9518, 12)
```

Univariate Analysis

[ ]:
```
dataw.columns
```

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

[ ]:
```
dataw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9518 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    9518 non-null   datetime64[ns]
 1   season      9518 non-null   category
 2   holiday     9518 non-null   category
 3   workingday  9518 non-null   category
 4   weather     9518 non-null   category
 5   temp        9518 non-null   float64
 6   atemp       9518 non-null   float64
 7   humidity    9518 non-null   int64
```

 8   windspeed   9518 non-null   float64
 9   casual   9518 non-null   int64
 10 registered 9518 non-null   int64
 11 count   9518 non-null   int64
dtypes: category(4), datetime64[ns](1), float64(3), int64(4)
memory usage: 707.1 KB

sns.histplot(dataw["casual"],kde=True)

[ ]:
```
dataw.columns
```

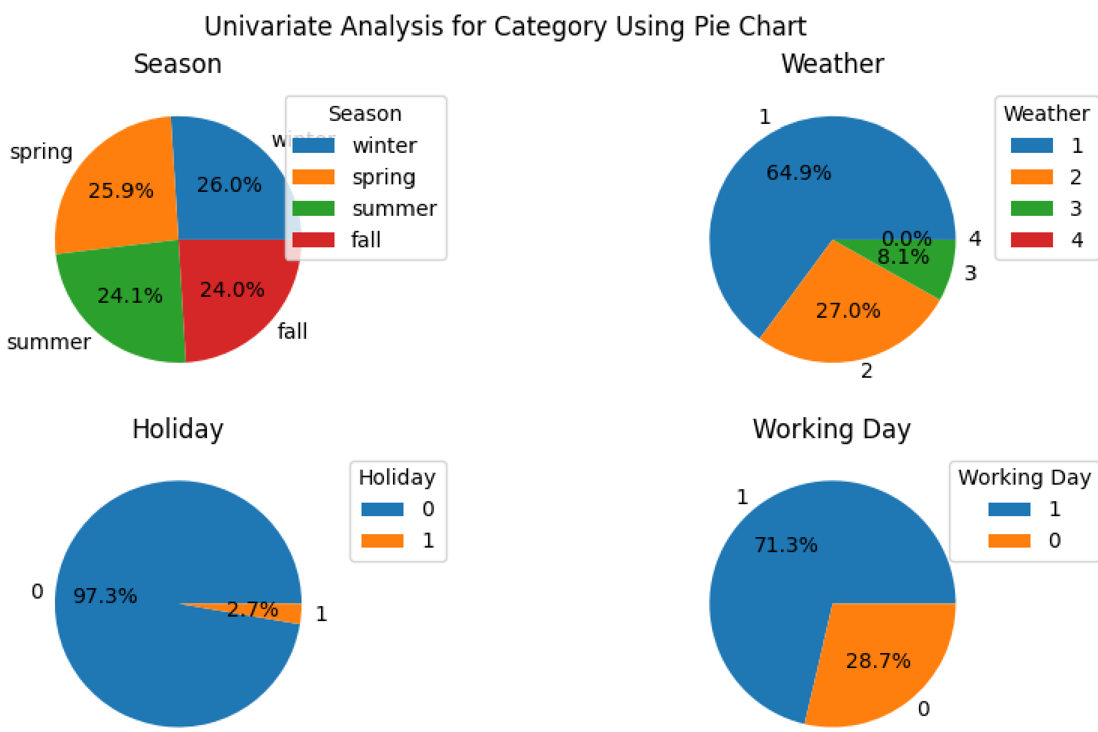
[ ]:
```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count'],
      dtype='object')
```

pie chart cat var

4 Graphic Analysis

[ ]:
```
fig, axes = plt.subplots(2, 2, figsize=(10, 5))

# Pie chart for "season"
axes[0, 0].pie(dataw["season"].value_counts(), labels=dataw["season"].
 ↪value_counts().index, autopct="%1.1f%%")
axes[0, 0].set_title("Season")
axes[0, 0].legend(dataw["season"].value_counts().index, title="Season",␣
 ↪loc="upper right", bbox_to_anchor=(1.4, 1))

# Pie chart for "weather"
axes[0, 1].pie(dataw["weather"].value_counts(), labels=dataw["weather"].
 ↪value_counts().index, autopct="%1.1f%%")
axes[0, 1].set_title("Weather")
axes[0, 1].legend(dataw["weather"].value_counts().index, title="Weather",␣
 ↪loc="upper right", bbox_to_anchor=(1.4, 1))

# Pie chart for "holiday"
axes[1, 0].pie(dataw["holiday"].value_counts(), labels=dataw["holiday"].
 ↪value_counts().index, autopct="%1.1f%%")
axes[1, 0].set_title("Holiday")
axes[1, 0].legend(dataw["holiday"].value_counts().index, title="Holiday",␣
 ↪loc="upper right", bbox_to_anchor=(1.4, 1))

# Pie chart for "workingday"
axes[1, 1].pie(dataw["workingday"].value_counts(), labels=dataw["workingday"].
 ↪value_counts().index, autopct="%1.1f%%")
axes[1, 1].set_title("Working Day")
```

```
axes[1, 1].legend(dataw["workingday"].value_counts().index, title="Working␣
↪Day", loc="upper right", bbox_to_anchor=(1.4, 1))

fig.tight_layout()
plt.suptitle("Univariate Analysis for Category Using Pie Chart", y=1.02)
plt.show()
```



4.1 Insights from Pie Chart Analysis

4.2 1. Season

- **Winter (26%)** and **Spring (25.9%)** are the most represented seasons, followed closely by **Summer (24.1%)** and **Fall (24%)**.

4.3 2. Weather

- **Weather Type 1 (64.9%)** dominates the dataset, possibly representing clear or mild conditions.
- **Weather Type 2 (27%)** is significant but less frequent.
- **Weather Types 3 (8.1%)** and **4 (0%)** are rare

4.4 3. Holiday

- **97.3%** of the data represents **Non-Holiday (0)** days, while only **2.7%** corresponds to **Holiday (1)** days.

4.5 4. Working Day

- **71.3%** of the data represents **Working Days (1)**, while **28.7%** is for non-working days.

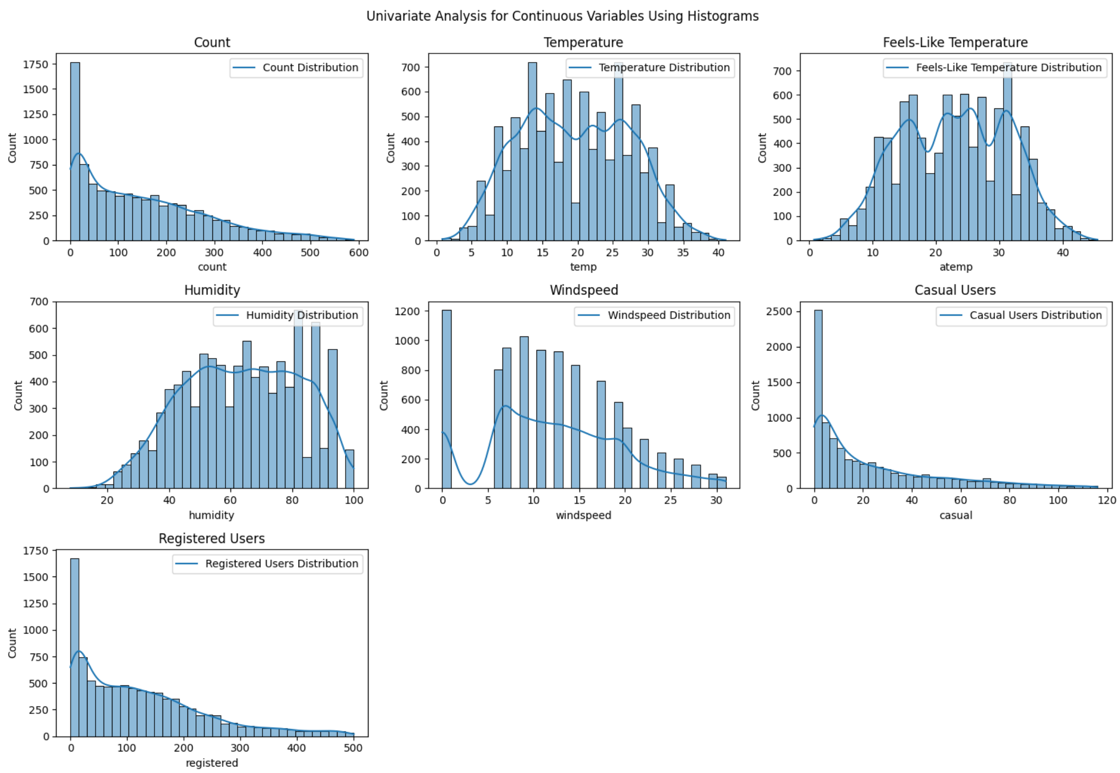
[ ]:

```

```

[ ]:

```python
fig, axes = plt.subplots(3, 3, figsize=(15, 10))

# Histogram for "count"
sns.histplot(dataw["count"], kde=True, ax=axes[0, 0])
axes[0, 0].set_title("Count")
axes[0, 0].legend(["Count Distribution"], loc="upper right")

# Histogram for "temp"
sns.histplot(dataw["temp"], kde=True, ax=axes[0, 1])
axes[0, 1].set_title("Temperature")
axes[0, 1].legend(["Temperature Distribution"], loc="upper right")

# Histogram for "atemp"
sns.histplot(dataw["atemp"], kde=True, ax=axes[0, 2])
axes[0, 2].set_title("Feels-Like Temperature")
axes[0, 2].legend(["Feels-Like Temperature Distribution"], loc="upper right")

# Histogram for "humidity"
sns.histplot(dataw["humidity"], kde=True, ax=axes[1, 0])
axes[1, 0].set_title("Humidity")
axes[1, 0].legend(["Humidity Distribution"], loc="upper right")

# Histogram for "windspeed"
sns.histplot(dataw["windspeed"], kde=True, ax=axes[1, 1])
axes[1, 1].set_title("Windspeed")
axes[1, 1].legend(["Windspeed Distribution"], loc="upper right")

# Histogram for "casual"
sns.histplot(dataw["casual"], kde=True, ax=axes[1, 2])
axes[1, 2].set_title("Casual Users")
axes[1, 2].legend(["Casual Users Distribution"], loc="upper right")

# Histogram for "registered"
sns.histplot(dataw["registered"], kde=True, ax=axes[2, 0])
axes[2, 0].set_title("Registered Users")
axes[2, 0].legend(["Registered Users Distribution"], loc="upper right")
```

```python
# Turn off the empty subplots
axes[2, 1].axis("off")
axes[2, 2].axis("off")

# Adjust layout
fig.tight_layout()
plt.suptitle("Univariate Analysis for Continuous Variables Using Histograms", y=1.02)
plt.show()
```



4.6 Insights from Histograms

4.7 1. Count

- The distribution of total user count is heavily **right-skewed**.
- A significant portion of user counts are concentrated at lower values, suggesting many instances with a low number of users.

4.8 2. Temperature

- The temperature shows a **normal distribution**, with most values concentrated between **15°C and 30°C**.
- There are fewer data points at extremely low or high temperatures.

4.9 3. Feels-Like Temperature (atemp)

- The "Feels-Like Temperature" follows a distribution similar to actual temperature.
- Most values are concentrated between **15°C and 30°C**, with fewer data points at extremes.

4.10 4. Humidity

- The humidity distribution is **right-skewed**, with most values between **60% and 80%**.
- Very low humidity values are rare.

4.11 5. Windspeed

- Windspeed is also **right-skewed**, with a large proportion of values below **10 m/s**.
- Higher windspeed values are much less frequent.

4.12 6. Casual Users

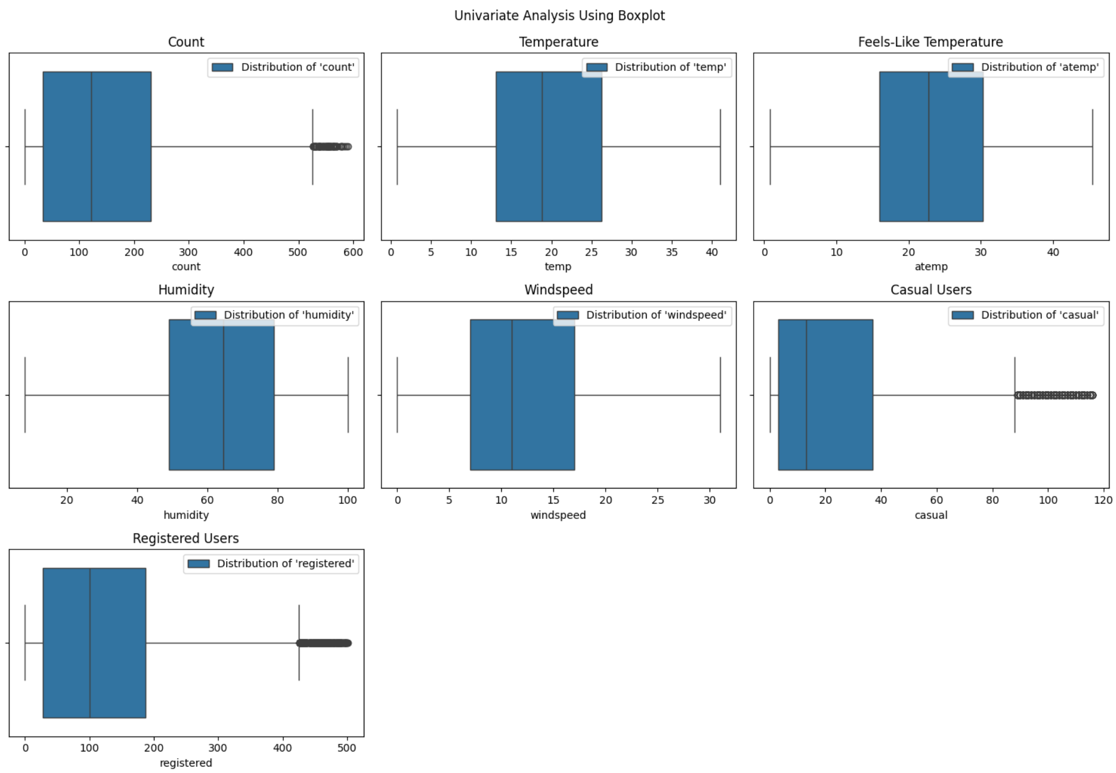
- The distribution of casual users is **highly right-skewed**, with most values concentrated near **0–10 users**.
- A small number of instances have a large number of casual users.

4.13 7. Registered Users

- Similar to casual users, registered users' data is **right-skewed**, but with a broader spread.
- Most registered user counts are concentrated below **200**, with a long tail for higher values.

[ ]:
```
fig, axes = plt.subplots(3, 3, figsize=(15, 10))

# Boxplot for "count"
sns.boxplot(data=dataw, x="count", ax=axes[0, 0])
axes[0, 0].set_title("Count")
axes[0, 0].legend(["Distribution of 'count'"], loc="upper right")

# Boxplot for "temp"
sns.boxplot(data=dataw, x="temp", ax=axes[0, 1])
axes[0, 1].set_title("Temperature")
axes[0, 1].legend(["Distribution of 'temp'"], loc="upper right")

# Boxplot for "atemp"
sns.boxplot(data=dataw, x="atemp", ax=axes[0, 2])
axes[0, 2].set_title("Feels-Like Temperature")
axes[0, 2].legend(["Distribution of 'atemp'"], loc="upper right")

# Boxplot for "humidity"
sns.boxplot(data=dataw, x="humidity", ax=axes[1, 0])
axes[1, 0].set_title("Humidity")
axes[1, 0].legend(["Distribution of 'humidity'"], loc="upper right")

# Boxplot for "windspeed"
```

```
sns.boxplot(data=dataw, x="windspeed", ax=axes[1, 1])
axes[1, 1].set_title("Windspeed")
axes[1, 1].legend(["Distribution of 'windspeed'"], loc="upper right")

# Boxplot for "casual"
sns.boxplot(data=dataw, x="casual", ax=axes[1, 2])
axes[1, 2].set_title("Casual Users")
axes[1, 2].legend(["Distribution of 'casual'"], loc="upper right")

# Boxplot for "registered"
sns.boxplot(data=dataw, x="registered", ax=axes[2, 0])
axes[2, 0].set_title("Registered Users")
axes[2, 0].legend(["Distribution of 'registered'"], loc="upper right")

# Turn off empty subplots
axes[2, 1].axis("off")
axes[2, 2].axis("off")

# Adjust layout
fig.tight_layout()
plt.suptitle("Univariate Analysis Using Boxplot", y=1.02)
plt.show()
```



4.14 Insights from Boxplot Analysis

4.15 1. Count

- The median value for the total user count lies near **200**.
- There are multiple outliers with user counts exceeding **500**, indicating occasional high-demand days.

4.16 2. Temperature

- The temperature distribution is symmetric, with most values between **15°C and 25°C**.
- No significant outliers are observed, showing a consistent range.

4.17 3. Feels-Like Temperature (atemp)

- The "Feels-Like Temperature" mirrors the actual temperature, with most values between **15°C and 25°C**.

4.18 4. Humidity

- The humidity is concentrated between **60% and 80%**, with the median near **70%**.

4.19 5. Windspeed

- Most windspeed values lie between **5 m/s and 20 m/s**, with the median around **12 m/s**.

4.20 6. Casual Users

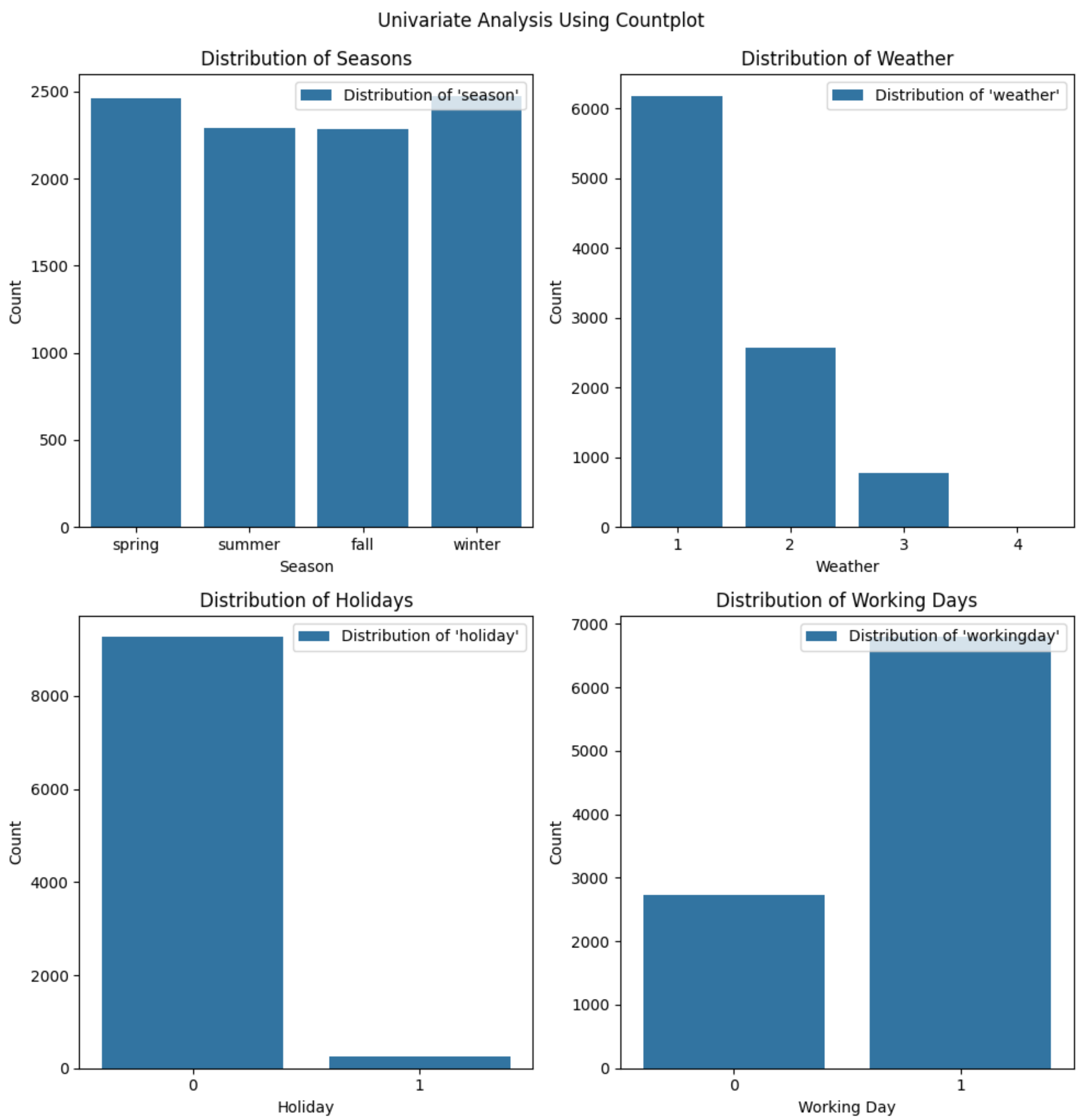
- The casual user data is heavily skewed, with the majority below **20 users**.
- There are numerous outliers above **60 users**

4.21 7. Registered Users

- Registered users mostly fall below **300**, with the median around **200**.
- There are multiple outliers above **400**

[ ]:
```
fig, axes = plt.subplots(2, 2, figsize=(10, 10))

# Countplot for "season"
sns.countplot(data=dataw, x="season", ax=axes[0, 0])
axes[0, 0].set_title("Distribution of Seasons")
axes[0, 0].set_xlabel("Season")
axes[0, 0].set_ylabel("Count")
axes[0, 0].legend(["Distribution of 'season'"], loc="upper right")
# Countplot for "weather"
sns.countplot(data=dataw, x="weather", ax=axes[0, 1])
axes[0, 1].set_title("Distribution of Weather")
axes[0, 1].set_xlabel("Weather")
axes[0, 1].set_ylabel("Count")
axes[0, 1].legend(["Distribution of 'weather'"], loc="upper right")
# Countplot for "holiday"
```

```python
sns.countplot(data=dataw, x="holiday", ax=axes[1, 0])
axes[1, 0].set_title("Distribution of Holidays")
axes[1, 0].set_xlabel("Holiday")
axes[1, 0].set_ylabel("Count")
axes[1, 0].legend(["Distribution of 'holiday'"], loc="upper right")

# Countplot for "workingday"
sns.countplot(data=dataw, x="workingday", ax=axes[1, 1])
axes[1, 1].set_title("Distribution of Working Days")
axes[1, 1].set_xlabel("Working Day")
axes[1, 1].set_ylabel("Count")
axes[1, 1].legend(["Distribution of 'workingday'"], loc="upper right")


# Adjust layout
fig.tight_layout()
plt.suptitle("Univariate Analysis Using Countplot", y=1.02)
plt.show()
```



4.22 Insights from Univariate Analysis (Countplot)

4.23 1. Distribution of Seasons

- The data is almost evenly distributed across the seasons, with **Spring** having slightly more occurrences than others.

4.24 2. Distribution of Weather

- **Weather Type 1** dominates the dataset (clear or mild weather).
- Other weather types are less frequent, with **Type 4 (extreme weather)** being rare.

4.25 3. Distribution of Holidays

- Most records (majority of the dataset) correspond to **Non-Holiday (0)** days.

4.26 4. Distribution of Working Days

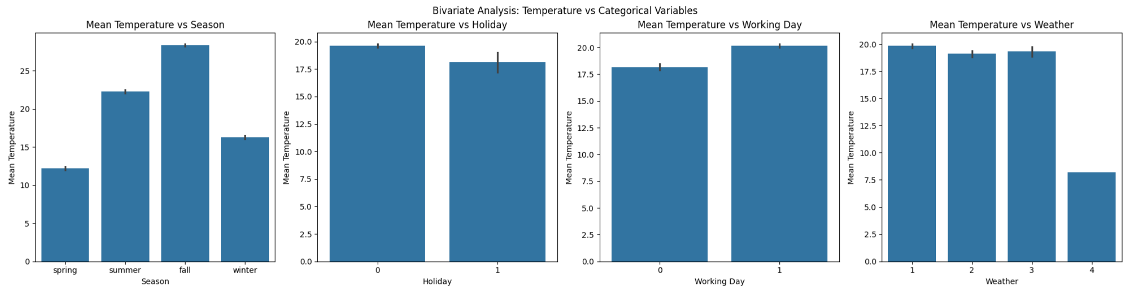
- The majority of the data corresponds to **Working Days (1)**, indicating the dataset focuses more on weekdays than weekends.

4.27 Bivariate Analysis:

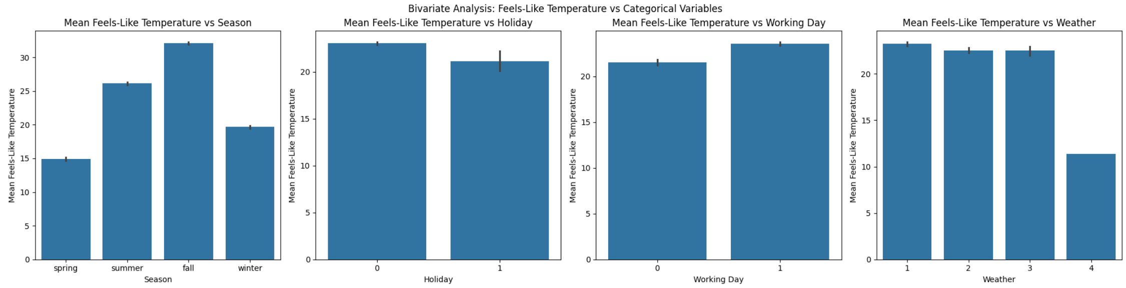
[ ]:
```
dataw.columns
```

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

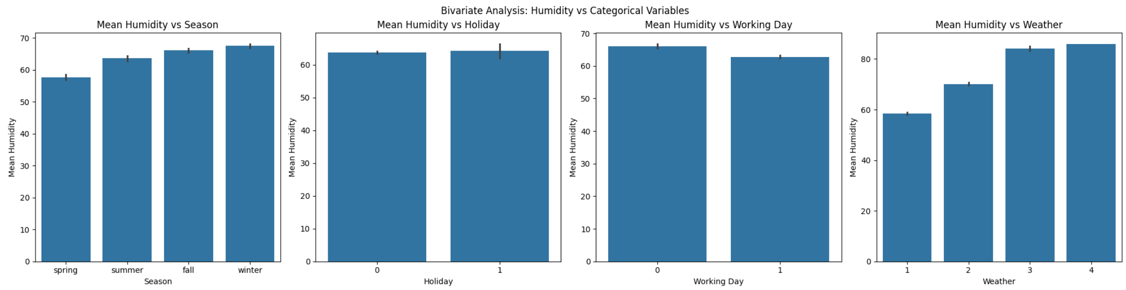
[ ]:
```
fig, axes = plt.subplots(1, 4, figsize=(20, 5))

# Temperature (temp) vs Season
sns.barplot(data=dataw, x="season", y="temp", ax=axes[0])
axes[0].set_title("Mean Temperature vs Season")
axes[0].set_xlabel("Season")
axes[0].set_ylabel("Mean Temperature")

# Temperature (temp) vs Holiday
sns.barplot(data=dataw, x="holiday", y="temp", ax=axes[1])
axes[1].set_title("Mean Temperature vs Holiday")
axes[1].set_xlabel("Holiday")
axes[1].set_ylabel("Mean Temperature")

# Temperature (temp) vs Working Day
sns.barplot(data=dataw, x="workingday", y="temp", ax=axes[2])
axes[2].set_title("Mean Temperature vs Working Day")
axes[2].set_xlabel("Working Day")
axes[2].set_ylabel("Mean Temperature")

# Temperature (temp) vs Weather
sns.barplot(data=dataw, x="weather", y="temp", ax=axes[3])
axes[3].set_title("Mean Temperature vs Weather")
axes[3].set_xlabel("Weather")
axes[3].set_ylabel("Mean Temperature")

# Adjust layout and add a title
fig.tight_layout()
plt.suptitle("Bivariate Analysis: Temperature vs Categorical Variables", y=1.02)
plt.show()
```



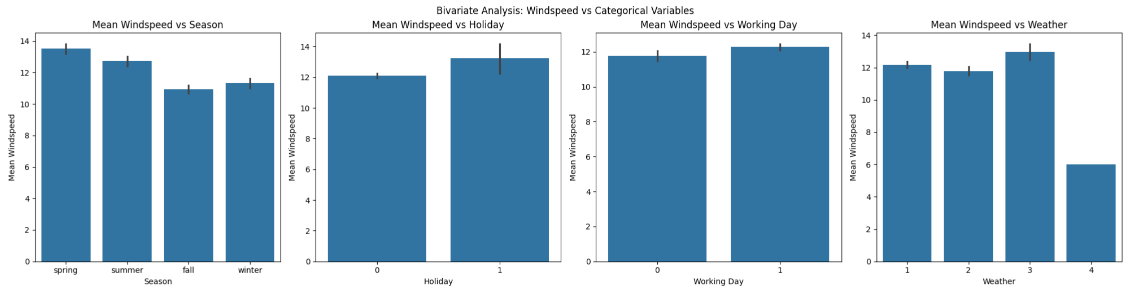
[ ]:
```
fig, axes = plt.subplots(1, 4, figsize=(20, 5))

# Feels-Like Temperature (atemp) vs Season
sns.barplot(data=dataw, x="season", y="atemp", ax=axes[0])
axes[0].set_title("Mean Feels-Like Temperature vs Season")
axes[0].set_xlabel("Season")
axes[0].set_ylabel("Mean Feels-Like Temperature")

# Feels-Like Temperature (atemp) vs Holiday
sns.barplot(data=dataw, x="holiday", y="atemp", ax=axes[1])
axes[1].set_title("Mean Feels-Like Temperature vs Holiday")
axes[1].set_xlabel("Holiday")
axes[1].set_ylabel("Mean Feels-Like Temperature")

# Feels-Like Temperature (atemp) vs Working Day
sns.barplot(data=dataw, x="workingday", y="atemp", ax=axes[2])
axes[2].set_title("Mean Feels-Like Temperature vs Working Day")
axes[2].set_xlabel("Working Day")
axes[2].set_ylabel("Mean Feels-Like Temperature")

# Feels-Like Temperature (atemp) vs Weather
sns.barplot(data=dataw, x="weather", y="atemp", ax=axes[3])
axes[3].set_title("Mean Feels-Like Temperature vs Weather")
axes[3].set_xlabel("Weather")
axes[3].set_ylabel("Mean Feels-Like Temperature")

# Adjust layout and add a main title
fig.tight_layout()
plt.suptitle("Bivariate Analysis: Feels-Like Temperature vs Categorical Variables", y=1.02)
plt.show()
```



[ ]:
```
# Bar charts for Humidity
fig, axes = plt.subplots(1, 4, figsize=(20, 5))

# Humidity vs Season
sns.barplot(data=dataw, x="season", y="humidity", ax=axes[0])
axes[0].set_title("Mean Humidity vs Season")
axes[0].set_xlabel("Season")
axes[0].set_ylabel("Mean Humidity")

# Humidity vs Holiday
sns.barplot(data=dataw, x="holiday", y="humidity", ax=axes[1])
axes[1].set_title("Mean Humidity vs Holiday")
axes[1].set_xlabel("Holiday")
axes[1].set_ylabel("Mean Humidity")

# Humidity vs Working Day
sns.barplot(data=dataw, x="workingday", y="humidity", ax=axes[2])
axes[2].set_title("Mean Humidity vs Working Day")
axes[2].set_xlabel("Working Day")
axes[2].set_ylabel("Mean Humidity")

# Humidity vs Weather
sns.barplot(data=dataw, x="weather", y="humidity", ax=axes[3])
axes[3].set_title("Mean Humidity vs Weather")
axes[3].set_xlabel("Weather")
axes[3].set_ylabel("Mean Humidity")

# Adjust layout and add a title
fig.tight_layout()
plt.suptitle("Bivariate Analysis: Humidity vs Categorical Variables", y=1.02)
plt.show()
```

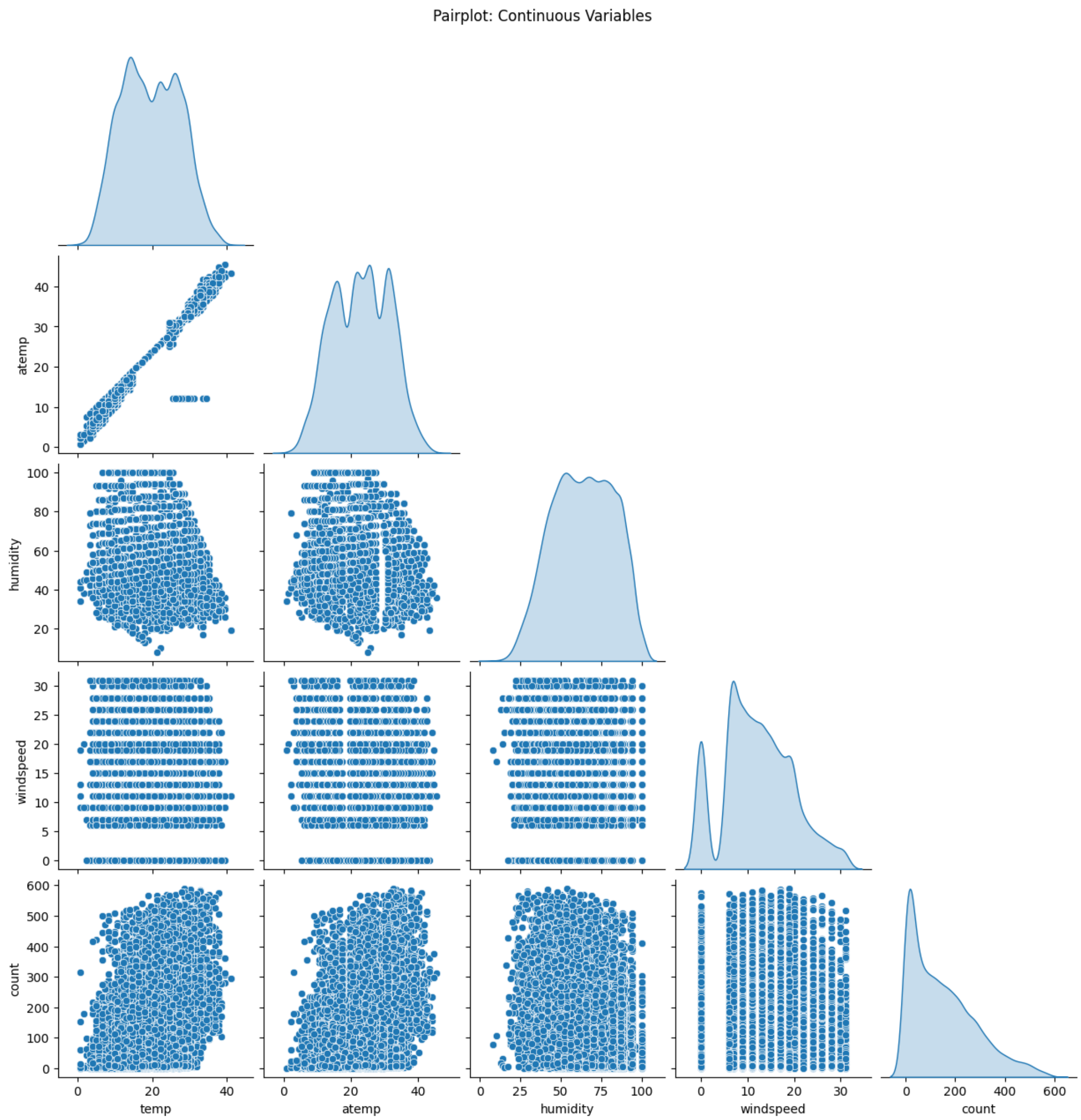


[ ]:
```
fig, axes = plt.subplots(1, 4, figsize=(20,5))

# Windspeed vs Season
sns.barplot(data=dataw, x="season", y="windspeed", ax=axes[0])
axes[0].set_title("Mean Windspeed vs Season")
axes[0].set_xlabel("Season")
axes[0].set_ylabel("Mean Windspeed")

# Windspeed vs Holiday
sns.barplot(data=dataw, x="holiday", y="windspeed", ax=axes[1])
axes[1].set_title("Mean Windspeed vs Holiday")
axes[1].set_xlabel("Holiday")
axes[1].set_ylabel("Mean Windspeed")

# Windspeed vs Working Day
sns.barplot(data=dataw, x="workingday", y="windspeed", ax=axes[2])
axes[2].set_title("Mean Windspeed vs Working Day")
axes[2].set_xlabel("Working Day")
axes[2].set_ylabel("Mean Windspeed")

# Windspeed vs Weather
sns.barplot(data=dataw, x="weather", y="windspeed", ax=axes[3])
axes[3].set_title("Mean Windspeed vs Weather")
axes[3].set_xlabel("Weather")
axes[3].set_ylabel("Mean Windspeed")

# Adjust layout and add a title
fig.tight_layout()
plt.suptitle("Bivariate Analysis: Windspeed vs Categorical Variables", y=1.02)
plt.show()
```



[ ]:

```
continuous_vars = ["temp", "atemp", "humidity", "windspeed", "count"]

pairplot = sns.pairplot(data=dataw[continuous_vars], diag_kind="kde",corner=True)
pairplot.fig.suptitle("Pairplot: Continuous Variables", y=1.02)
plt.show()
```



4.28 Insights from Pairplot Analysis

- **Temp vs Atemp**: There is a strong positive correlation between actual temperature and feels-like temperature.
- **Count vs Temp**: Total user count increases with temperature but levels off after a certain point.
- **Count vs Humidity**: Slight negative correlation; count decreases as humidity increases.
- **Count vs Windspeed**: No strong correlation is observed between count and windspeed.

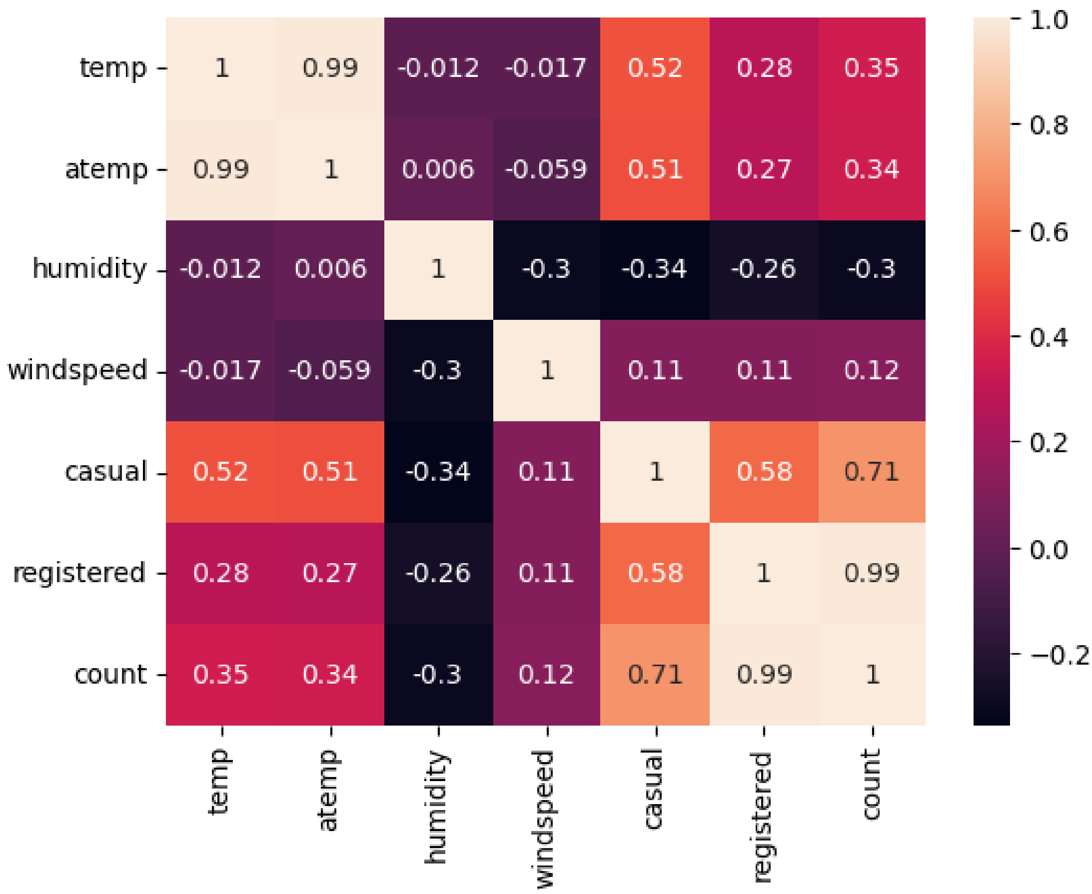
[ ]:

```

```

[ ]:

```
sns.heatmap(dataw.corr(numeric_only=True),annot=True)
```

```
<Axes: >
```



2- Sample T-Test to check if Working Day has an effect on the number of electric cycles rented (10 points)

4.29 Insights from Correlation Heatmap

- **Temp and Atemp**: Strong positive correlation (0.99).
- **Count and Casual**: Strong positive correlation (0.71).
- **Count and Registered**: Strong positive correlation (0.99).
- **Count and Temp**: Moderate positive correlation (0.35).
- **Count and Humidity**: Weak negative correlation (-0.30).

```
dataw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9518 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
```

```
 0   datetime    9518 non-null   datetime64[ns]
 1   season      9518 non-null   category
 2   holiday     9518 non-null   category
 3   workingday  9518 non-null   category
 4   weather     9518 non-null   category
 5   temp        9518 non-null   float64
 6   atemp       9518 non-null   float64
 7   humidity    9518 non-null   int64
 8   windspeed   9518 non-null   float64
 9   casual      9518 non-null   int64
 10  registered  9518 non-null   int64
 11  count       9518 non-null   int64
dtypes: category(4), datetime64[ns](1), float64(3), int64(4)
memory usage: 707.1 KB
```

5 Working Day has effect on number of electric cycles rented

[ ]:
```
dataw.to_csv("dataw.csv",index=False)
```

[ ]:
```
from scipy.stats import ttest_ind
from scipy.stats import norm
from scipy.stats import shapiro
from scipy.stats import levene
from scipy.stats import kstest
import seaborn as sns
```

Assumptions

```
1   Normaility of data
2   Independent Observations
3   Homogeneou variances
```

[ ]:
```
workingday=dataw[dataw["workingday"]==1]["count"]
notworkingday=dataw[dataw["workingday"]==0]["count"]
```

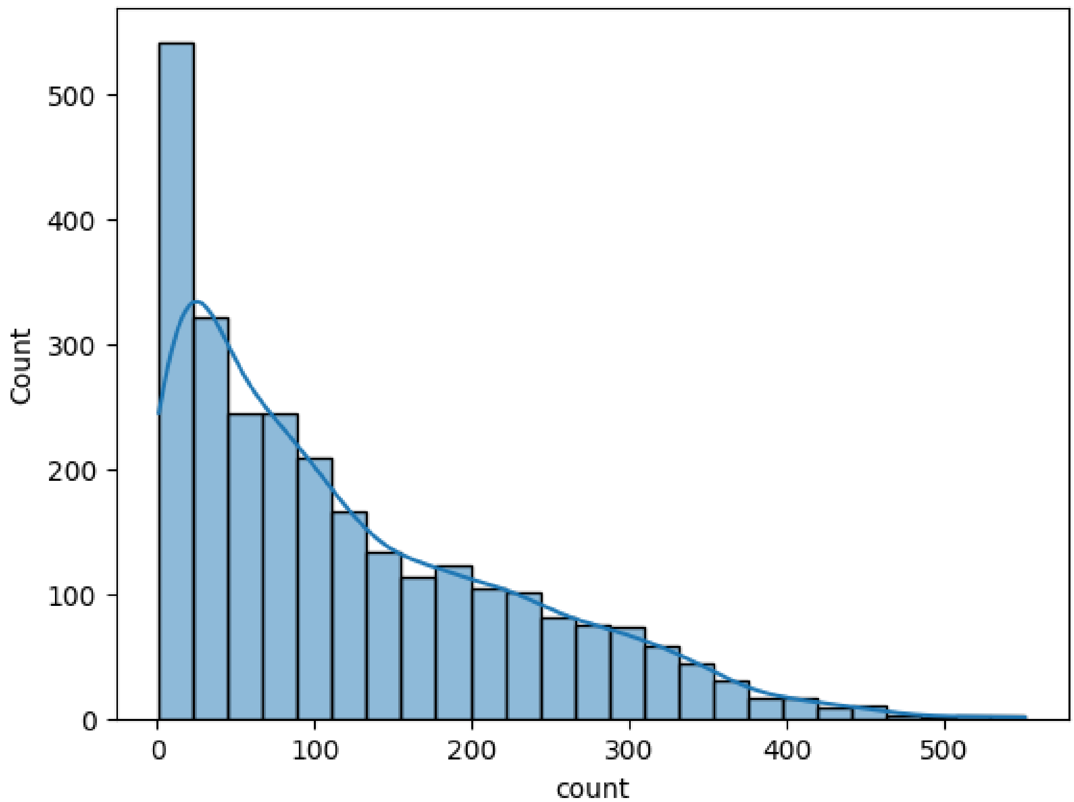
[ ]:
```
notworkingday.shape,workingday.shape
```

[ ]:
```
((2728,), (6790,))
```

Checkng normality using hiostogram

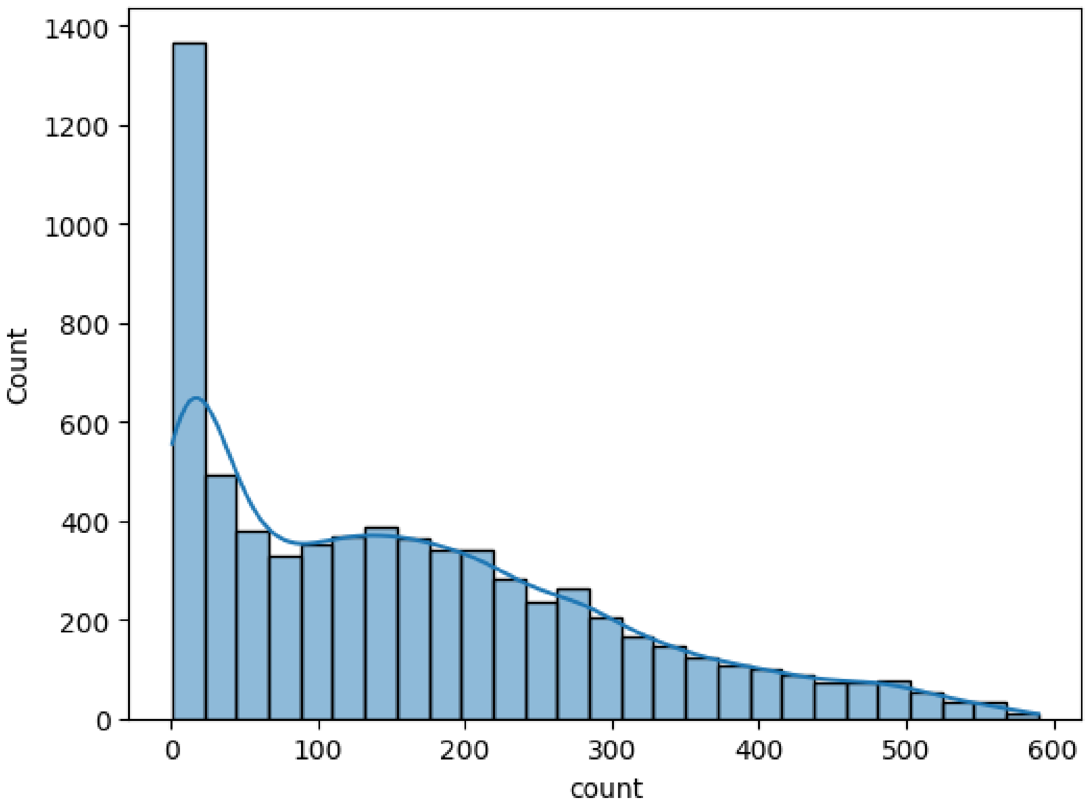
[ ]:
```
sns.histplot(notworkingday,kde=True)
```

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



[ ]:
```
sns.histplot(workingday,kde=True)
```

[ ]: <Axes: xlabel='count', ylabel='Count'>



Step1

` Null Hypothesis : There is no significant difference between the mean of electric cycles rent`

` Alternate Hypothesis : There is significant difference between the mean of electric cycles ren`

Step2

`Normal Distribution`

checking for normal distribution using shapiro

[ ]:

```

```

[ ]:

```
from scipy.stats import shapiro, levene, kstest


from scipy.stats import shapiro

def test_normality(group, group_name, alpha=0.05, max_sample_size=100):

    sample_size = min(len(group), max_sample_size)

```

```
    shapiro_result = shapiro(group.sample(sample_size, random_state=42))
    print(f"Shapiro-Wilk Test for {group_name} (Sample Size: {sample_size}):")
    print(shapiro_result)

    if shapiro_result.pvalue < alpha:
        print(f"Conclusion: Reject Null Hypothesis (p = {shapiro_result.pvalue:.
↪5f}). Data is not normally distributed.\n")
    else:
        print(f"Conclusion: Fail to Reject Null Hypothesis (p = {shapiro_result.
↪pvalue:.5f}). Data is normally distributed.\n")


def test_variance_equality(group1, group2, alpha=0.05):
    """
    Perform Levene's Test for equality of variances and interpret the p-value.
    """
    levene_result = levene(group1, group2)
    print("Levene's Test (Equality of Variance):")
    print(levene_result)

    # P-value interpretation
    if levene_result.pvalue < alpha:
        print(f"Conclusion: Reject Null Hypothesis (p = {levene_result.pvalue:.
↪5f}). Variances are significantly different.\n")
    else:
        print(f"Conclusion: Fail to Reject Null Hypothesis (p = {levene_result.
↪pvalue:.5f}). Variances are equal.\n")
```

[ ]:
```
test_normality(notworkingday,"Not Working Day")
```

```
Shapiro-Wilk Test for Not Working Day (Sample Size: 100):
ShapiroResult(statistic=0.9021842833805265, pvalue=1.7936127824652382e-06)
Conclusion: Reject Null Hypothesis (p = 0.00000). Data is not normally
distributed.
```

[ ]:
```
test_normality(workingday,"Working Day")
```

```
Shapiro-Wilk Test for Working Day (Sample Size: 100):
ShapiroResult(statistic=0.8933011623952262, pvalue=6.937217420684444e-07)
Conclusion: Reject Null Hypothesis (p = 0.00000). Data is not normally
distributed.
```

[ ]:
```
test_variance_equality(notworkingday,workingday)
```

```
Levene's Test (Equality of Variance):
```

```
LeveneResult(statistic=232.23283443276156, pvalue=7.927650077398614e-52)
Conclusion: Reject Null Hypothesis (p = 0.00000). Variances are significantly
different.
```

5.1 After testing it results that the data is not normal. There is no Homogeneous variance between them

5.2 Even if we test the using with equal varainces thsi results such that t-statistic assumes equal group variances, so it over- or underestimates the true effect size.

[ ]:
```
res=ttest_ind(notworkingday,workingday)
if res.pvalue<0.05:
    print("Reject Null Hypothesis",res)
else:
    print("Fail to reject Null Hypothesis",res)
```

```
Reject Null Hypothesis TtestResult(statistic=-13.983019373271851,
pvalue=5.384896180235767e-44, df=9516.0)
```

5.3 There is significant difference between the mean of electric cycles rented on working and non working days

5.4 PERFORMING CENTRAL NORMAL DISTRIBUTION

Under the assumption the data violates the normality and homogeneous varaince done test under sahprio,levene

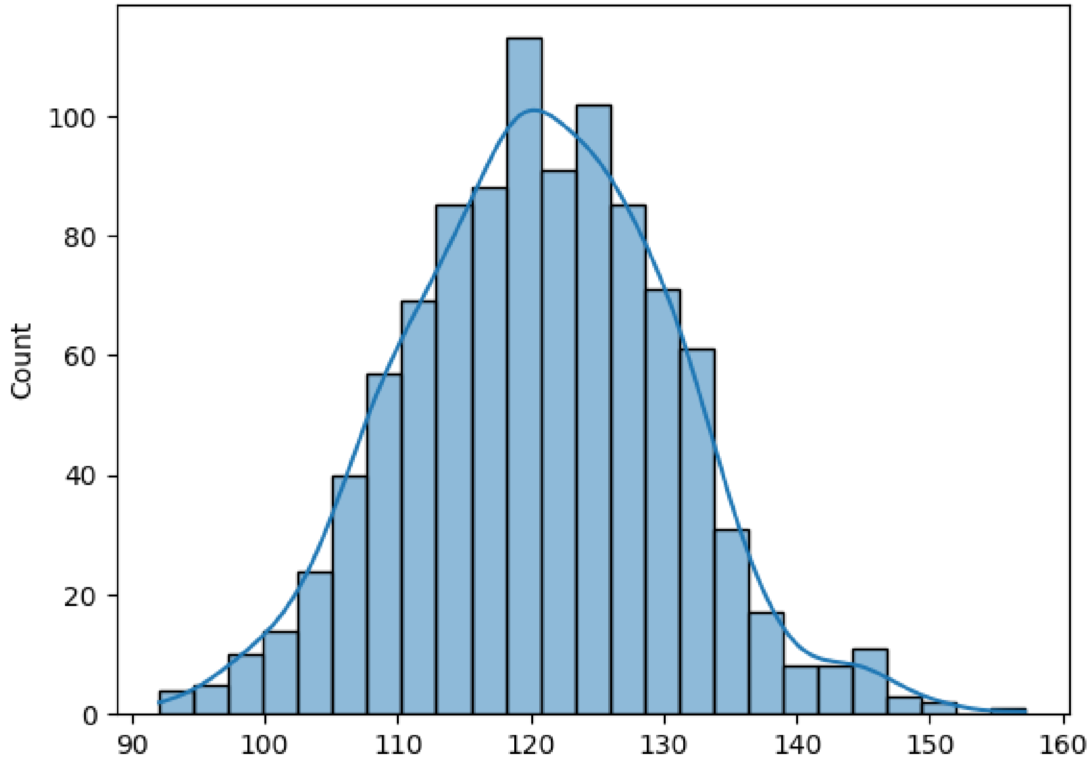
[ ]:
```
sample_size=100
```

[ ]:
```
notworkingday_samples=np.array([np.mean(np.random.choice(notworkingday,␣
↪sample_size, replace=False)) for _ in range(1000)])
```

[ ]:
```
workingday_samples=np.array([np.mean(np.random.choice(workingday, sample_size,␣
↪replace=False)) for _ in range(1000)])
```

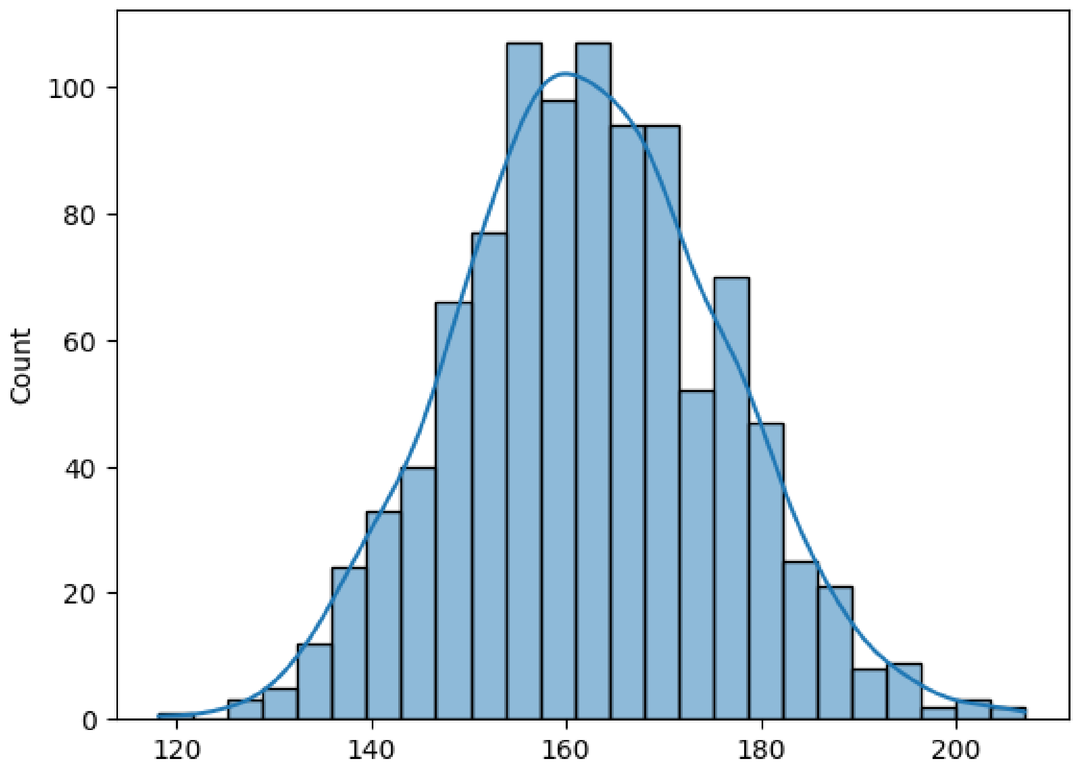
[ ]:
```
sns.histplot(notworkingday_samples,kde=True)
```

[ ]:
```
<Axes: ylabel='Count'>
```



[ ]:
```
sns.histplot(workingday_samples,kde=True)
```

[ ]: <Axes: ylabel='Count'>




```
[ ]: test_normality(pd.DataFrame(notworkingday_samples),"Not Working Day")
```

```
Shapiro-Wilk Test for Not Working Day (Sample Size: 100):
ShapiroResult(statistic=0.9803109641393678, pvalue=0.1405469790239932)
Conclusion: Fail to Reject Null Hypothesis (p = 0.14055). Data is normally
distributed.
```

```
[ ]: test_normality(pd.DataFrame(workingday_samples),"Working Day")
```

```
Shapiro-Wilk Test for Working Day (Sample Size: 100):
ShapiroResult(statistic=0.9926458069160509, pvalue=0.8658030425539214)
Conclusion: Fail to Reject Null Hypothesis (p = 0.86580). Data is normally
distributed.
```

```
[ ]: test_variance_equality(notworkingday_samples,workingday_samples)
```

```
Levene's Test (Equality of Variance):
LeveneResult(statistic=79.22110190800404, pvalue=1.2168589613667966e-18)
Conclusion: Reject Null Hypothesis (p = 0.00000). Variances are significantly
different.
```

Step2

```
Normal Distribution
```

Step3

```
two-tail test
```

Step4

```
Calculate test statistics and p-value
```

Step5

```
 Decide  "Reject Null Hypothesis" or Fail to reject Null Hypothesis" based on p-value and alpha
```

[ ]:

```

```

Null Hypothesis (H): The means of the two groups are equal.

Alternative Hypothesis (H): The means of the two groups are not equal (i.e., there is a difference in either direction).

[ ]:

```
tstat,p_val=ttest_ind(notworkingday_samples,workingday_samples,alternative='two-sided',equal_v
print("test statistics ",tstat)
print("p value ",p_val)
if p_val<0.05:
    print("Reject Null Hypothesis")
else:
    print("Fail to reject Null Hypothesis")
```

```
test statistics  -78.39197634670285
p value  0.0
Reject Null Hypothesis
```

5.5 There is significant difference between the mean of electric cycles rented on working and non working days

5.6 Checking for Right Tail Test

Null Hypothesis (H): Mean of notworkingday_samples <= Mean of workingday_samples

Alternative Hypothesis (H): Mean of notworkingday_samples > Mean of workingday_samples

[ ]:

```
tstat,p_val=ttest_ind(notworkingday_samples,workingday_samples,alternative='greater',equal_var
print("test statistics ",tstat)
print("p value ",p_val)
if p_val<0.05:
    print("Reject Null Hypothesis")
else:
    print("Fail to reject Null Hypothesis ")
```

```
test statistics  -78.39197634670285
p value  1.0
Fail to reject Null Hypothesis
```

5.7 INFERENCE

```
Mean of notworkingday_samples <= Mean of workingday_samples
```

5.8 There is significant difference between the mean of electric cycles rented on working and non working days

6 No. of cycles rented similar or different in different seasons using ANOVA

Step1

```
Null Hypothesis :  The mean number of cycles rented is the same across all seasons.

Alternate Hypothesis: The mean number of cycles rented is different across seasons.
```

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

[ ]:
```
dataw["season"].value_counts()
```

[ ]:
```
season
winter    2475
spring    2463
summer    2292
fall      2288
Name: count, dtype: int64
```

[ ]:
```
season1=dataw[dataw["season"]=="spring"]["count"]
season2=dataw[dataw["season"]=="summer"]["count"]
season3=dataw[dataw["season"]=="fall"]["count"]
season4=dataw[dataw["season"]=="winter"]["count"]
```

[ ]:
```
test_normality(season1,"season1")
test_normality(season2,"season2")
test_normality(season3,"season3")
test_normality(season4,"season4")
```

```
Shapiro-Wilk Test for season1 (Sample Size: 100):
ShapiroResult(statistic=0.8312206959902055, pvalue=2.5464949098718397e-09)
Conclusion: Reject Null Hypothesis (p = 0.00000). Data is not normally
distributed.

Shapiro-Wilk Test for season2 (Sample Size: 100):
ShapiroResult(statistic=0.9089865460041465, pvalue=3.837189241769641e-06)
Conclusion: Reject Null Hypothesis (p = 0.00000). Data is not normally
```

```
distributed.

Shapiro-Wilk Test for season3 (Sample Size: 100):
ShapiroResult(statistic=0.9048534390340699, pvalue=2.4086305338135233e-06)
Conclusion: Reject Null Hypothesis (p = 0.00000). Data is not normally
distributed.

Shapiro-Wilk Test for season4 (Sample Size: 100):
ShapiroResult(statistic=0.9168088605470425, pvalue=9.569101337826996e-06)
Conclusion: Reject Null Hypothesis (p = 0.00001). Data is not normally
distributed.
```

[ ]:
```
f_oneway(season1,season2,season3,season4)
```

[ ]:
```
F_onewayResult(statistic=155.83821650550502, pvalue=1.328514170995064e-98)
```

[ ]:
```
test_variance_equality(season1,season2)
test_variance_equality(season1,season3)
test_variance_equality(season1,season4)
```

```
Levene's Test (Equality of Variance):
LeveneResult(statistic=267.6644088334988, pvalue=1.4237919808988396e-58)
Conclusion: Reject Null Hypothesis (p = 0.00000). Variances are significantly
different.

Levene's Test (Equality of Variance):
LeveneResult(statistic=330.11741316931074, pvalue=2.2569243678362363e-71)
Conclusion: Reject Null Hypothesis (p = 0.00000). Variances are significantly
different.

Levene's Test (Equality of Variance):
LeveneResult(statistic=238.68612835697152, pvalue=1.2752288855572295e-52)
Conclusion: Reject Null Hypothesis (p = 0.00000). Variances are significantly
different.
```

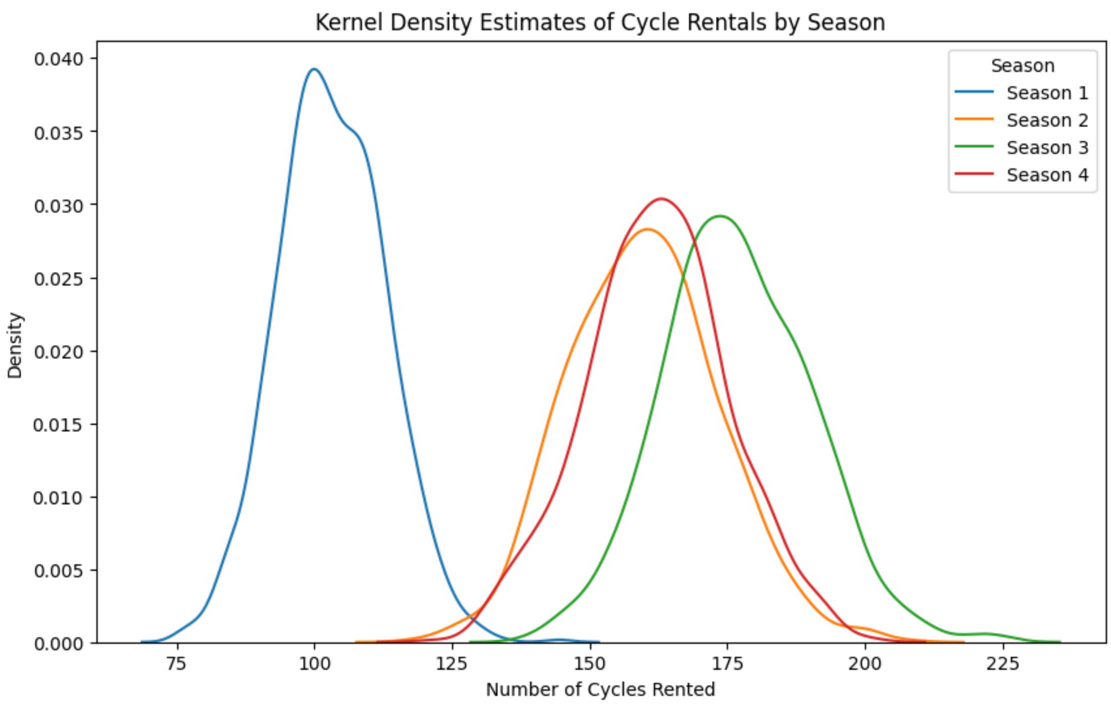
6.1 PERFORMING CENTRAL NORMAL DISTRIBUTION

Under the assumption the data violates the normality

[ ]:
```
season1_samples=np.array([np.mean(np.random.choice(season1, sample_size, replace=False)) for _ in range(1000)])
season2_samples=np.array([np.mean(np.random.choice(season2, sample_size, replace=False)) for _ in range(1000)])
season3_samples=np.array([np.mean(np.random.choice(season3, sample_size, replace=False)) for _ in range(1000)])
```

```
season4_samples=np.array([np.mean(np.random.choice(season4, sample_size,␣
↪replace=False)) for _ in range(1000)])
```

[ ]:
```
plt.figure(figsize=(10, 6))
sns.kdeplot(season1_samples, label='Season 1')
sns.kdeplot(season2_samples, label='Season 2' )
sns.kdeplot(season3_samples, label='Season 3')
sns.kdeplot(season4_samples, label='Season 4')

plt.title('Kernel Density Estimates of Cycle Rentals by Season')
plt.xlabel('Number of Cycles Rented')
plt.ylabel('Density')
plt.legend(title='Season')
plt.show()
```



[ ]:
```
test_normality(pd.DataFrame(season1_samples),"Season1")
test_normality(pd.DataFrame(season2_samples),"Season2")
test_normality(pd.DataFrame(season3_samples),"Season3")
test_normality(pd.DataFrame(season4_samples),"Season4")
```

```
Shapiro-Wilk Test for Season1 (Sample Size: 100):
ShapiroResult(statistic=0.9915266090514493, pvalue=0.7859694306524077)
Conclusion: Fail to Reject Null Hypothesis (p = 0.78597). Data is normally
distributed.
```

```
Shapiro-Wilk Test for Season2 (Sample Size: 100):
ShapiroResult(statistic=0.9887011149286913, pvalue=0.5612446089701296)
Conclusion: Fail to Reject Null Hypothesis (p = 0.56124). Data is normally
distributed.

Shapiro-Wilk Test for Season3 (Sample Size: 100):
ShapiroResult(statistic=0.9913377681366985, pvalue=0.7714459719496329)
Conclusion: Fail to Reject Null Hypothesis (p = 0.77145). Data is normally
distributed.

Shapiro-Wilk Test for Season4 (Sample Size: 100):
ShapiroResult(statistic=0.9839198855563129, pvalue=0.26477526631759135)
Conclusion: Fail to Reject Null Hypothesis (p = 0.26478). Data is normally
distributed.
```

[ ]:
```
test_variance_equality(season1_samples,season2_samples)
test_variance_equality(season1_samples,season3_samples)
test_variance_equality(season1_samples,season4_samples)
```

```
Levene's Test (Equality of Variance):
LeveneResult(statistic=95.23809295628078, pvalue=5.190719458645026e-22)
Conclusion: Reject Null Hypothesis (p = 0.00000). Variances are significantly
different.

Levene's Test (Equality of Variance):
LeveneResult(statistic=85.8826521023895, pvalue=4.782571463864029e-20)
Conclusion: Reject Null Hypothesis (p = 0.00000). Variances are significantly
different.

Levene's Test (Equality of Variance):
LeveneResult(statistic=66.96123202240709, pvalue=4.872771144867655e-16)
Conclusion: Reject Null Hypothesis (p = 0.00000). Variances are significantly
different.
```

[ ]:
```
res=f_oneway(season1_samples,season2_samples,season3_samples,season4_samples)
res
```

[ ]:
```
F_onewayResult(statistic=6649.486227497695, pvalue=0.0)
```

[ ]:
```
res=f_oneway(season1_samples,season2_samples,season3_samples,season4_samples)
if res.pvalue<0.05:
    print("Reject Null Hypothesis",)
else:
    print("Fail to reject Null Hypothesis")
```

```
Reject Null Hypothesis
```

6.2 The mean number of cycles rented is different across seasons.

[ ]:

```

```

7 No. of cycles rented similar or different in different weather using ANOVA

Step1

```
Null Hypothesis :  The mean number of cycles rented is the same across all weather conditions

Alternate Hypothesis: The mean number of cycles rented is different across weather conditions
```

[ ]:

```
weather1=dataw[dataw["weather"]==1]["count"]
weather2=dataw[dataw["weather"]==2]["count"]
weather3=dataw[dataw["weather"]==3]["count"]
weather4=dataw[dataw["weather"]==4]["count"]
```

[ ]:

```
test_normality(weather1,"weather1")
test_normality(weather2,"weather2")
test_normality(weather3,"weather3")
```

```
Shapiro-Wilk Test for weather1 (Sample Size: 100):
ShapiroResult(statistic=0.8910609930555164, pvalue=5.498787531742089e-07)
Conclusion: Reject Null Hypothesis (p = 0.00000). Data is not normally
distributed.

Shapiro-Wilk Test for weather2 (Sample Size: 100):
ShapiroResult(statistic=0.8872511550289661, pvalue=3.7269393330915137e-07)
Conclusion: Reject Null Hypothesis (p = 0.00000). Data is not normally
distributed.

Shapiro-Wilk Test for weather3 (Sample Size: 100):
ShapiroResult(statistic=0.8756795380530384, pvalue=1.1965297818662179e-07)
Conclusion: Reject Null Hypothesis (p = 0.00000). Data is not normally
distributed.
```

[ ]:

```
res=f_oneway(weather1,weather2,weather3,weather4)
if res.pvalue<0.05:
    print("Reject Null Hypothesis",)
else:
    print("Fail to reject Null Hypothesis")
```

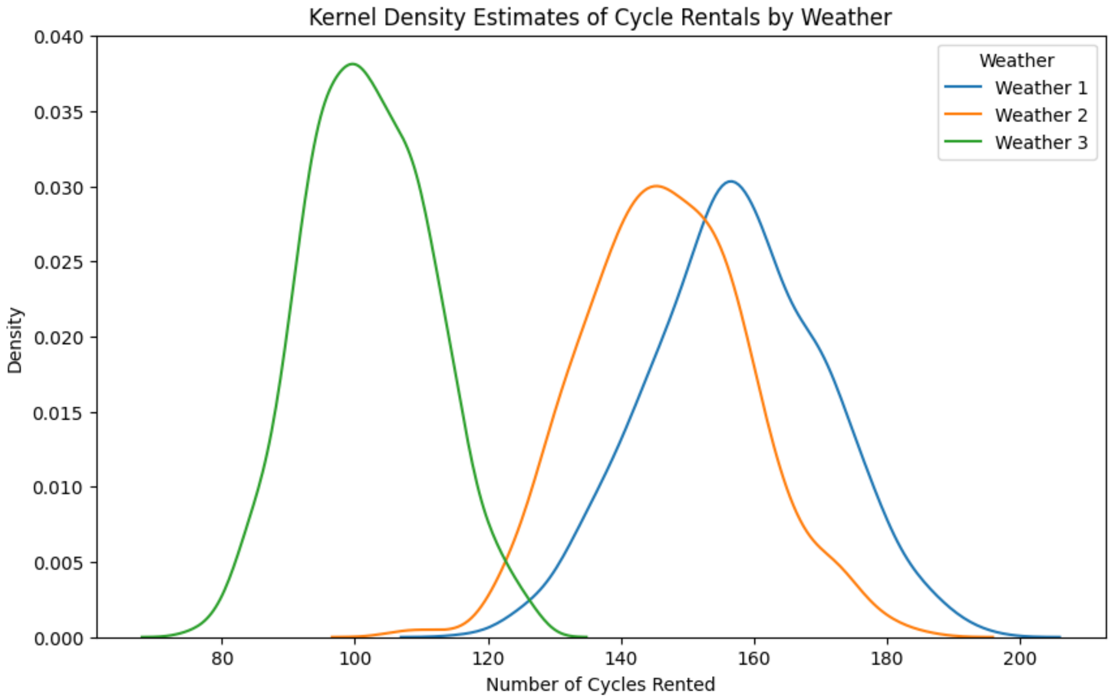
```
Reject Null Hypothesis
```

7.1 The mean number of cycles rented is different across weather conditions.

```
[ ]: weather1_samples=np.array([np.mean(np.random.choice(weather1, sample_size,␣
      ↪replace=False)) for _ in range(1000)])
     weather2_samples=np.array([np.mean(np.random.choice(weather2, sample_size,␣
      ↪replace=False)) for _ in range(1000)])
     weather3_samples=np.array([np.mean(np.random.choice(weather3, sample_size,␣
      ↪replace=False)) for _ in range(1000)])
```

```
[ ]: plt.figure(figsize=(10, 6))
     sns.kdeplot(weather1_samples, label='Weather 1')
     sns.kdeplot(weather2_samples, label='Weather 2' )
     sns.kdeplot(weather3_samples, label='Weather 3')

     plt.title('Kernel Density Estimates of Cycle Rentals by Weather')
     plt.xlabel('Number of Cycles Rented')
     plt.ylabel('Density')
     plt.legend(title='Weather')
```

```
[ ]: <matplotlib.legend.Legend at 0x7b4e0463acb0>
```




```
[ ]: test_normality(pd.DataFrame(weather1_samples),"Weather1")
     test_normality(pd.DataFrame(weather2_samples),"Weather2")
     test_normality(pd.DataFrame(weather3_samples),"Weather3")
```

```
Shapiro-Wilk Test for Weather1 (Sample Size: 100):
ShapiroResult(statistic=0.9930718868554078, pvalue=0.8923583912924385)
Conclusion: Fail to Reject Null Hypothesis (p = 0.89236). Data is normally
distributed.

Shapiro-Wilk Test for Weather2 (Sample Size: 100):
ShapiroResult(statistic=0.9766353186468227, pvalue=0.07247958565086454)
Conclusion: Fail to Reject Null Hypothesis (p = 0.07248). Data is normally
distributed.

Shapiro-Wilk Test for Weather3 (Sample Size: 100):
ShapiroResult(statistic=0.9822993063215206, pvalue=0.19997755701229386)
Conclusion: Fail to Reject Null Hypothesis (p = 0.19998). Data is normally
distributed.
```

[ ]:
```
res=f_oneway(weather1,weather2,weather3,weather4)
if res.pvalue<0.05:
    print("Reject Null Hypothesis",)
else:
    print("Fail to reject Null Hypothesis")
```

```
Reject Null Hypothesis
```

7.2 The mean number of cycles rented is different across weather conditions.

8 Is weather dependent on season (relationship between two predictors)?

Step1

```
Null Hypothesis :  weather and season are not associated
Alternate Hypothesis: weather and season are associated
```

Weather: 1: Clear, Few clouds, partly cloudy, partly cloudy

2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist

3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds

4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

[ ]:
```
pd.crosstab(dataw["season"],dataw["weather"],margins=True)
```

[ ]:

weather	1	2	3	4	All
season					
spring	1595	683	184	1	2463
summer	1473	614	205	0	2292
fall	1598	517	173	0	2288
winter	1510	754	211	0	2475

```
All      6176  2568  773  1  9518
```

- weather 1 consistently accounts for the **highest rentals** across all seasons, totaling **972,856**.
- Fall and Summer lead with **290,261** and **255,490** rentals, respectively, under clear weather.

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

[ ]:
```
res=chi2_contingency(pd.crosstab(dataw["season"],dataw["weather"]))
if res.pvalue<0.05:
    print("Reject Null Hypothesis",)
else:
    print("Fail to reject Null Hypothesis")
```

```
Reject Null Hypothesis
```

8.1 weather and season are associated

9 Insights

1. Rentals are significantly higher on working days compared to non-working days.
2. Non-working days and holidays see lower rentals compared to working days
3. Fall has the highest demand, followed by Winter and Summer, while Spring shows the lowest rentals.
4. Clear weather (Weather 1) drives the highest rentals, while extreme weather conditions (Weather 4) lead to the lowest demand.
5. Spring has the lowest demand, potentially due to less favorable weather or other factors.
6. Weather conditions vary significantly across seasons. Clear weather dominates Fall and Summer, contributing to high rentals.
7. Rentals remain stable at moderate windspeed levels, but extreme winds may reduce demand.
8. "Feels-like" temperature (atemp) closely correlates with actual temperature and shows similar effects on rentals.
9. Each season has distinct rental behavior influenced by temperature and weather.
10. Registered users account for a significant portion of rentals, indicating a loyal customer base.

#END

[229]:
```
!jupyter nbconvert --to pdf '/content/drive/My Drive/Colab Notebooks/YULU'
↪--output-dir='/content/drive/My Drive/Colab Notebooks/'
```

```
[NbConvertApp] Converting notebook /content/drive/My Drive/Colab Notebooks/YULU
to pdf
[NbConvertApp] Support files will be in YUL_files/
[NbConvertApp] Making directory ./YUL_files
[NbConvertApp] Writing 188711 bytes to notebook.tex
```

```
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 1250953 bytes to /content/drive/My Drive/Colab
Notebooks/YUL.pdf
```

[229]:

```

```