Bike-Sharing System (BoomBikes)

PROBLEM STATEMENT

This assignment is a programming assignment where in i'll be building a multiple linear regression model for the prediction of demand for shared bikes

A `bike-sharing system` is a service in which bikes are made available for shared use to individuals on a short term basis for a price or free. Many bike share systems allow people to borrow a bike from a "dock" which is usually computer-controlled wherein the user enters the payment information, and the system unlocks it. This bike can then be returned to another dock belonging to the same system.

A `US bike-sharing` provider `BoomBikes` has recently suffered considerable dips in their revenues due to the ongoing Corona pandemic. The company is finding it very difficult to sustain in the current market scenario. So, it has decided to come up with a mindful business plan to be able to accelerate its revenue as soon as the ongoing lockdown comes to an end, and the economy restores to a healthy state.

In such an attempt, BoomBikes aspires to understand the demand for shared bikes among the people after this ongoing quarantine situation ends across the nation due to Covid-19. They have planned this to prepare themselves to cater to the people's needs once the situation gets better all around and stand out from other service providers and make huge profits.

`The company wants to know:`

1. Which variables are significant in predicting the demand for shared bikes.

2. How well those variables describe the bike demands Based on various meteorological surveys and people's styles,

`Business Goal:`

I have to build a model for demand for shared bikes with the available independent variables. It will be used by the management to understand how exactly the demands vary with different features. They can accordingly manipulate the business strategy to meet the demand levels and meet the customer's expectations. Further, the model will be a good way for management to understand the demand dynamics of a new market.

In [1]:
```
#Importing all necessary library
# Pandas for Data Frame
import pandas as pd
# Numpy for advance math operation
import numpy as np
# filtering warnings
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
# reading CSV file
df = pd.read_csv('day.csv')
df
```

Out[2]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	register
0	1	01-01-2018	1	0	1	0	6	0	2	14.110847	18.18125	80.5833	10.749882	331	6
1	2	02-01-2018	1	0	1	0	0	0	2	14.902598	17.68695	69.6087	16.652113	131	6
2	3	03-01-2018	1	0	1	0	1	1	1	8.050924	9.47025	43.7273	16.636703	120	12
3	4	04-01-2018	1	0	1	0	2	1	1	8.200000	10.60610	59.0435	10.739832	108	14
4	5	05-01-2018	1	0	1	0	3	1	1	9.305237	11.46350	43.6957	12.522300	82	15
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
725	726	27-12-2019	1	1	12	0	4	1	2	10.420847	11.33210	65.2917	23.458911	247	18
726	727	28-12-2019	1	1	12	0	5	1	2	10.386653	12.75230	59.0000	10.416557	644	24
727	728	29-12-2019	1	1	12	0	6	0	2	10.386653	12.12000	75.2917	8.333661	159	11
728	729	30-12-2019	1	1	12	0	0	0	1	10.489153	11.58500	48.3333	23.500518	364	14
729	730	31-12-2019	1	1	12	0	1	1	2	8.849153	11.17435	57.7500	10.374682	439	22

730 rows × 16 columns

In [3]:

```
#The info() method provides a concise summary of a DataFrame, displaying information
# such as the number of entries, data types, and memory usage, aiding data exploration and analysis.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 730 entries, 0 to 729
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   instant     730 non-null    int64  
 1   dteday      730 non-null    object 
 2   season      730 non-null    int64  
 3   yr          730 non-null    int64  
 4   mnth        730 non-null    int64  
 5   holiday     730 non-null    int64  
 6   weekday     730 non-null    int64  
 7   workingday  730 non-null    int64  
 8   weathersit  730 non-null    int64  
 9   temp        730 non-null    float64
 10  atemp       730 non-null    float64
 11  hum         730 non-null    float64
 12  windspeed   730 non-null    float64
 13  casual      730 non-null    int64  
 14  registered  730 non-null    int64  
 15  cnt         730 non-null    int64  
dtypes: float64(4), int64(11), object(1)
memory usage: 91.4+ KB
```

In [4]:

```
#shape of data
df.shape
```

Out[4]:

```
(730, 16)
```

In [5]:

```
#The describe() function offers a concise statistical summary.
#It provides count, mean, standard deviation, minimum, 25th percentile, median, 75th percentile,
#and maximum values for numeric data in a DataFrame.
df.describe()
```

Out[5]:

	instant	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hu
count	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.0000
mean	365.500000	2.498630	0.500000	6.526027	0.028767	2.997260	0.683562	1.394521	20.319259	23.726322	62.7651
std	210.877136	1.110184	0.500343	3.450215	0.167266	2.006161	0.465405	0.544807	7.506729	8.150308	14.2375
min	1.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	2.424346	3.953480	0.0000
25%	183.250000	2.000000	0.000000	4.000000	0.000000	1.000000	0.000000	1.000000	13.811885	16.889713	52.0000
50%	365.500000	3.000000	0.500000	7.000000	0.000000	3.000000	1.000000	1.000000	20.465826	24.368225	62.6250
75%	547.750000	3.000000	1.000000	10.000000	0.000000	5.000000	1.000000	2.000000	26.880615	30.445775	72.9895
max	730.000000	4.000000	1.000000	12.000000	1.000000	6.000000	1.000000	3.000000	35.328347	42.044800	97.2500

In [6]:

```
#checking missing value
df.isna().sum()
```

Out[6]:

```
instant       0
dteday        0
season        0
yr            0
mnth          0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
casual        0
registered    0
cnt           0
dtype: int64
```

In [7]:

```
#checking unique values
df.nunique()
```

Out[7]:
```
instant       730
dteday        730
season          4
yr              2
mnth           12
holiday         2
weekday         7
workingday      2
weathersit      3
temp          498
atemp         689
hum           594
windspeed     649
casual        605
registered    678
cnt           695
dtype: int64
```

In [8]:
```
df['dteday']
```

Out[8]:
```
0      01-01-2018
1      02-01-2018
2      03-01-2018
3      04-01-2018
4      05-01-2018
          ...
725    27-12-2019
726    28-12-2019
727    29-12-2019
728    30-12-2019
729    31-12-2019
Name: dteday, Length: 730, dtype: object
```

Based on the high level look at the data and the data dictionary, the following variables can be removed from further analysis:

`- instant :` Its only an index value , we have a default index for the same purpose

`- dteday :` This has the date, Since we already have seperate columns for 'year' & 'month',hence, we can carry out our analysis without this column .

`- casual & registered :` Both these columns contains the count of bike booked by different categories of customers. Since our objective is to find the total count of bikes and not by specific category, we will ignore these two columns.

In [9]:
```
#droping
df=df.drop(['dteday','instant','casual','registered'],axis=1)
df
```

Out[9]:

	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	cnt
0	1	0	1	0	6	0	2	14.110847	18.18125	80.5833	10.749882	985
1	1	0	1	0	0	0	2	14.902598	17.68695	69.6087	16.652113	801
2	1	0	1	0	1	1	1	8.050924	9.47025	43.7273	16.636703	1349
3	1	0	1	0	2	1	1	8.200000	10.60610	59.0435	10.739832	1562
4	1	0	1	0	3	1	1	9.305237	11.46350	43.6957	12.522300	1600
...	...	...	...	...	...	...	...	...	...	...	...	...
725	1	1	12	0	4	1	2	10.420847	11.33210	65.2917	23.458911	2114
726	1	1	12	0	5	1	2	10.386653	12.75230	59.0000	10.416557	3095
727	1	1	12	0	6	0	2	10.386653	12.12000	75.2917	8.333661	1341
728	1	1	12	0	0	0	1	10.489153	11.58500	48.3333	23.500518	1796
729	1	1	12	0	1	1	2	8.849153	11.17435	57.7500	10.374682	2729

730 rows × 12 columns

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 730 entries, 0 to 729
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   season      730 non-null    int64  
 1   yr          730 non-null    int64  
 2   mnth        730 non-null    int64  
 3   holiday     730 non-null    int64  
 4   weekday     730 non-null    int64  
 5   workingday  730 non-null    int64  
 6   weathersit  730 non-null    int64  
 7   temp        730 non-null    float64
 8   atemp       730 non-null    float64
 9   hum         730 non-null    float64
 10  windspeed   730 non-null    float64
 11  cnt         730 non-null    int64  
dtypes: float64(4), int64(8)
memory usage: 68.6 KB
```

Data and information visualization

In [11]:
```
# matplot for visualization
import matplotlib.pyplot as plt
# subpart in matplot for visualization
import seaborn as sns
```

Taking a copy of main `df` for visulizations `df_for_viz`

In [12]:
```
df_for_viz = df.copy()
```

In [13]:
```
df_for_viz.columns
```

Out[13]:
```
Index(['season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday',
       'weathersit', 'temp', 'atemp', 'hum', 'windspeed', 'cnt'],
      dtype='object')
```

In [14]:
```
# Encoding/mapping the season,month,weekday,weathersit column for better visulization
df_for_viz.season=df_for_viz.season.map({1:'spring', 2:'summer', 3:'fall', 4:'winter'})
df_for_viz.mnth = df_for_viz.mnth.map({1:'jan',2:'feb',3:'mar',4:'apr',5:'may',6:'june',7:'july',8:'aug',9:'sep',10:'c
```

```
df_for_viz.weekday = df_for_viz.weekday.map({0:'sun',1:'mon',2:'tue',3:'wed',4:'thu',5:'fri',6:'sat'})
df_for_viz.weathersit = df_for_viz.weathersit.map({1:'Clear',2:'Misty',3:'Light_snowrain',4:'Heavy_snowrain'})
df_for_viz.yr = df_for_viz.yr.map({0:'2018',1:'2019'})
```

In [15]:
```
df_for_viz.head()
```

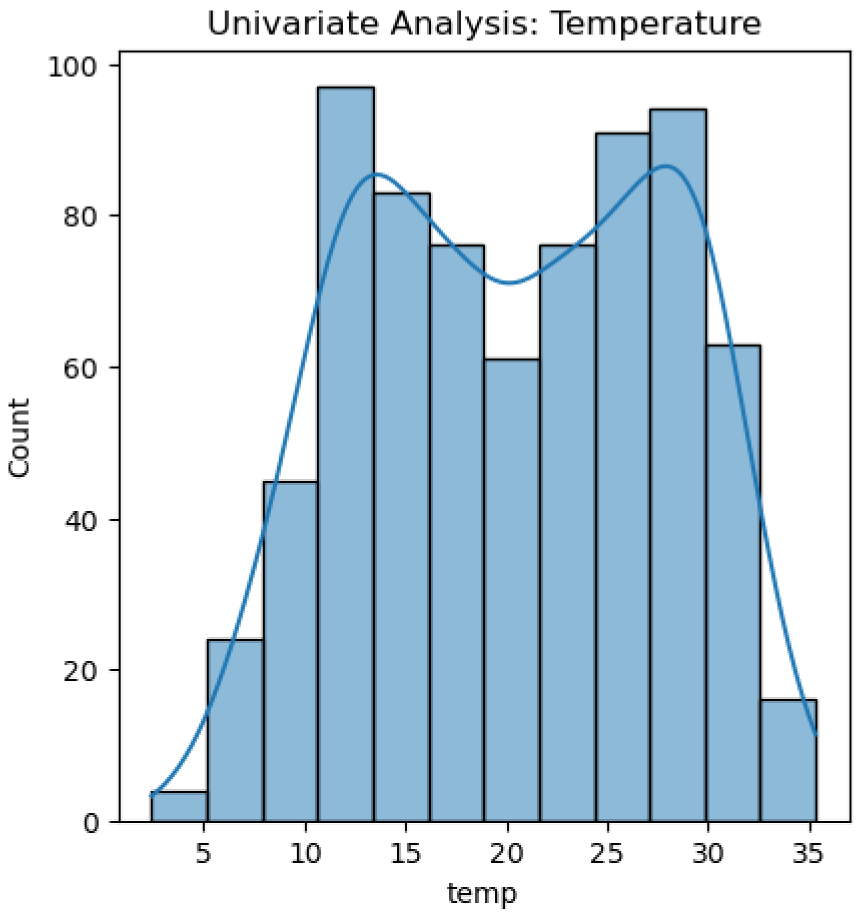
Out[15]:

	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	cnt
0	spring	2018	jan	0	sat	0	Misty	14.110847	18.18125	80.5833	10.749882	985
1	spring	2018	jan	0	sun	0	Misty	14.902598	17.68695	69.6087	16.652113	801
2	spring	2018	jan	0	mon	1	Clear	8.050924	9.47025	43.7273	16.636703	1349
3	spring	2018	jan	0	tue	1	Clear	8.200000	10.60610	59.0435	10.739832	1562
4	spring	2018	jan	0	wed	1	Clear	9.305237	11.46350	43.6957	12.522300	1600

Univariate analysis

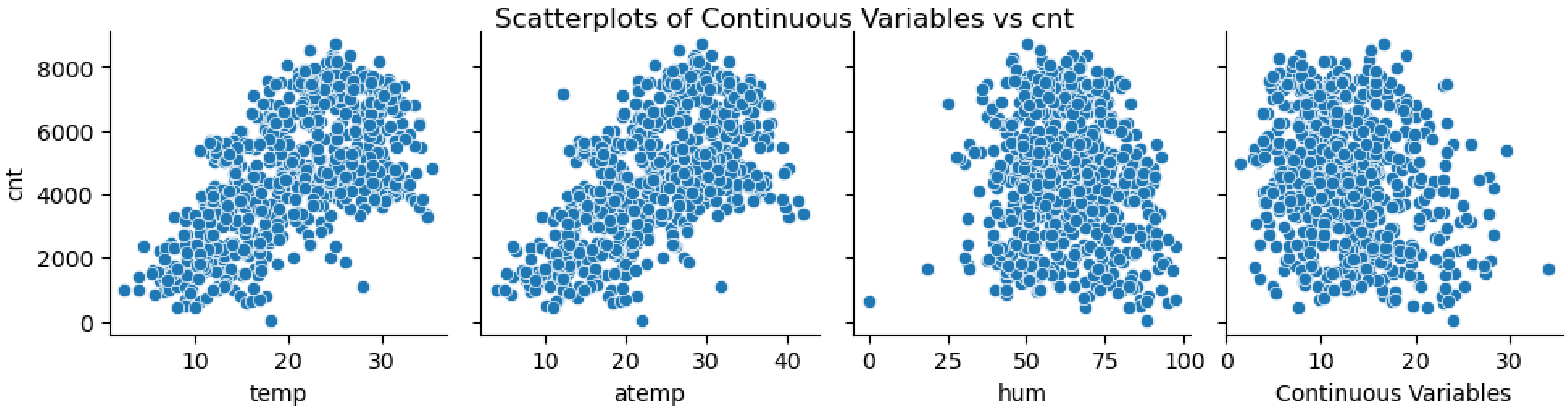
Univariate analysis is a type of data analysis that focuses on examining the characteristics of a single variable at a time.

In [16]:
```
plt.figure(figsize=(16, 5))
plt.subplot(1, 3, 1)
sns.histplot(df_for_viz['temp'], kde=True)
plt.title('Univariate Analysis: Temperature')
plt.show()
```



In [17]:

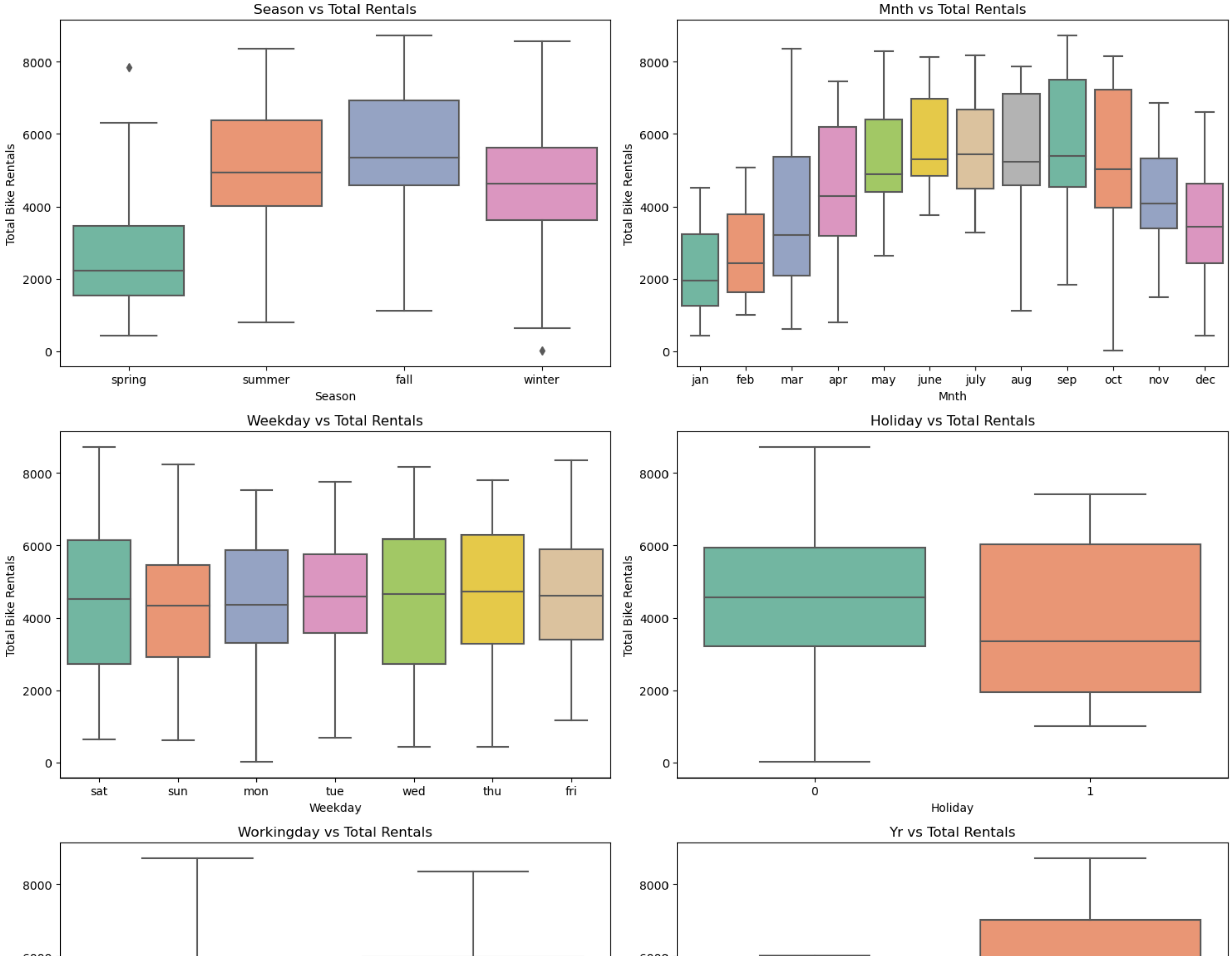
```
# Create scatterplots to identify linear relationships
sns.pairplot(df_for_viz, x_vars=['temp', 'atemp', 'hum', 'windspeed',
       ], y_vars=['cnt'], kind='scatter')

# Add titles and labels
plt.suptitle("Scatterplots of Continuous Variables vs cnt", y=1.02)
plt.xlabel("Continuous Variables")
plt.ylabel("Count 'cnt'")

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

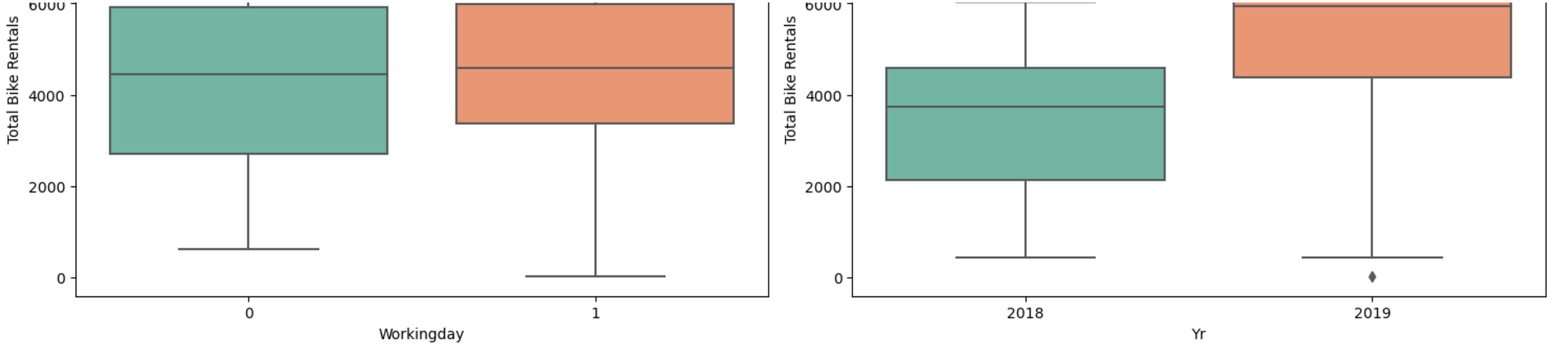


Bivariate

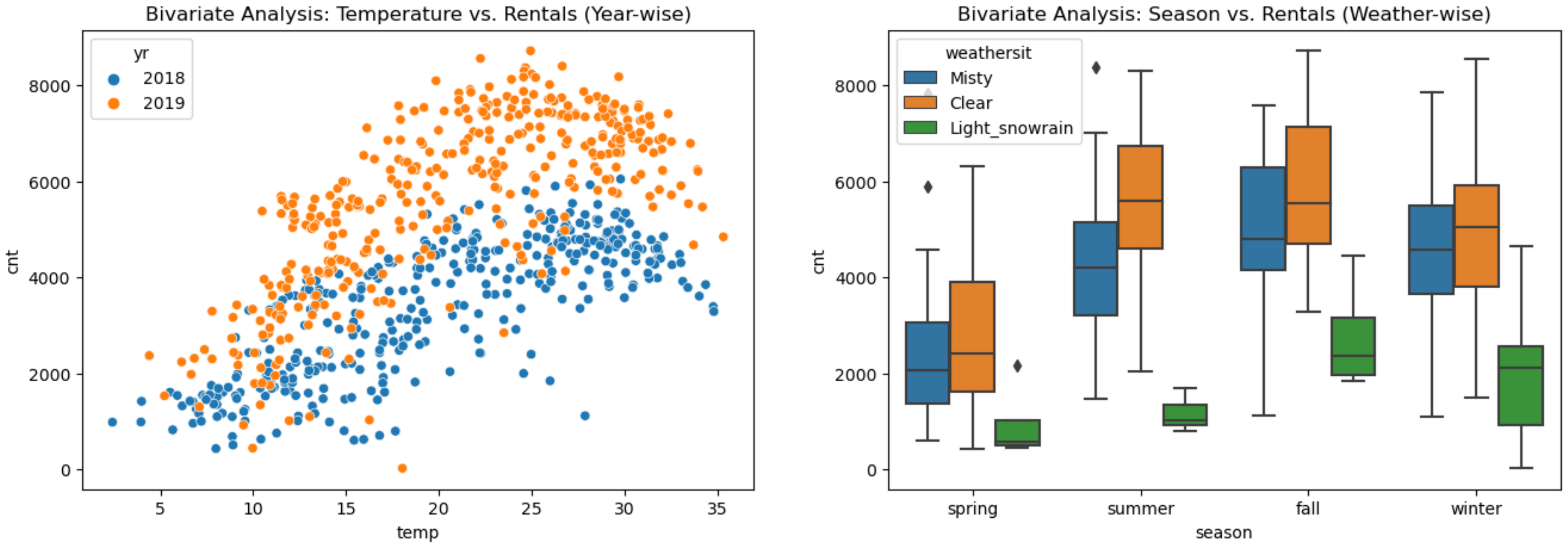
In [18]:
```
# Categorical columns to visualize
categorical_columns = ['season', 'mnth', 'weekday', 'holiday', 'workingday', 'yr']

# Set the figure size
plt.figure(figsize=(15, 15))

custom_palette = sns.color_palette("Set2")

# Create a grid of subplots
for i, col in enumerate(categorical_columns, start=1):
    plt.subplot(3, 2, i)


    # Create a box plot for the current categorical column
    sns.boxplot(x=col, y='cnt', data=df_for_viz, palette=custom_palette)

    # Add labels and titles
    plt.xlabel(col.capitalize())  # Use the column name as the x-axis label
    plt.ylabel('Total Bike Rentals')
    plt.title(f'{col.capitalize()} vs Total Rentals')
```

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

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


Season vs Total Rentals
Total Bike Rentals
0
2000
4000
6000
8000
spring
summer
fall
winter
Season
Mnth vs Total Rentals
Total Bike Rentals
jan
feb
mar
apr
may
june
july
aug
sep
oct
nov
dec
Mnth
Weekday vs Total Rentals
Total Bike Rentals
sat
sun
mon
tue
wed
thu
fri
Weekday
Holiday vs Total Rentals
Total Bike Rentals
0
1
Holiday
Workingday vs Total Rentals
Yr vs Total Rentals



In [19]:
```
# Bivariate Analysis
plt.figure(figsize=(16, 5))
plt.subplot(1, 2, 1)
sns.scatterplot(data=df_for_viz, x='temp', y='cnt', hue='yr')
plt.title('Bivariate Analysis: Temperature vs. Rentals (Year-wise)')

plt.subplot(1, 2, 2)
sns.boxplot(data=df_for_viz, x='season', y='cnt', hue='weathersit')
plt.title('Bivariate Analysis: Season vs. Rentals (Weather-wise)')

plt.show()
```

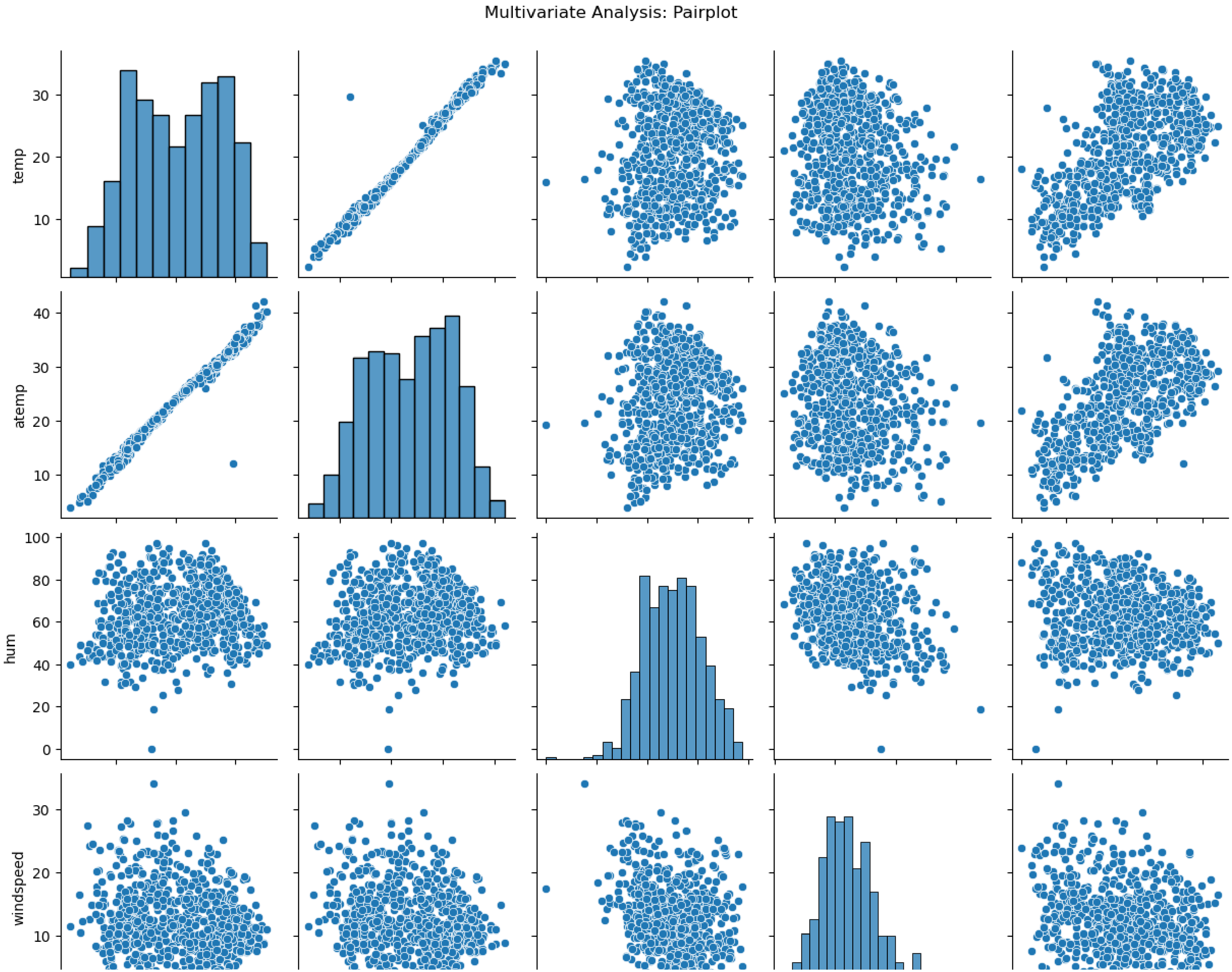


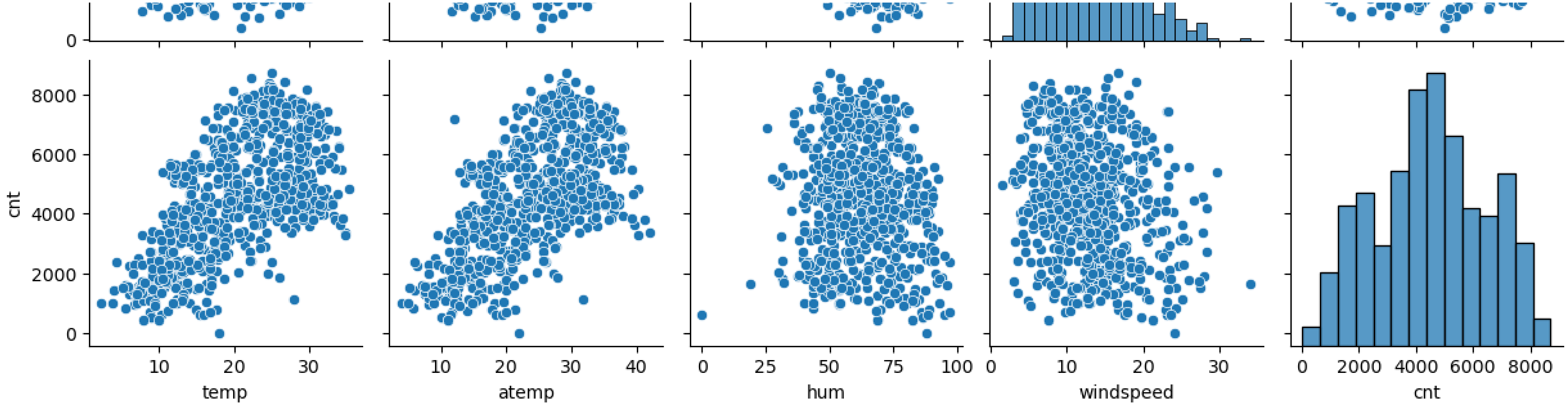
Multivariate Analysis

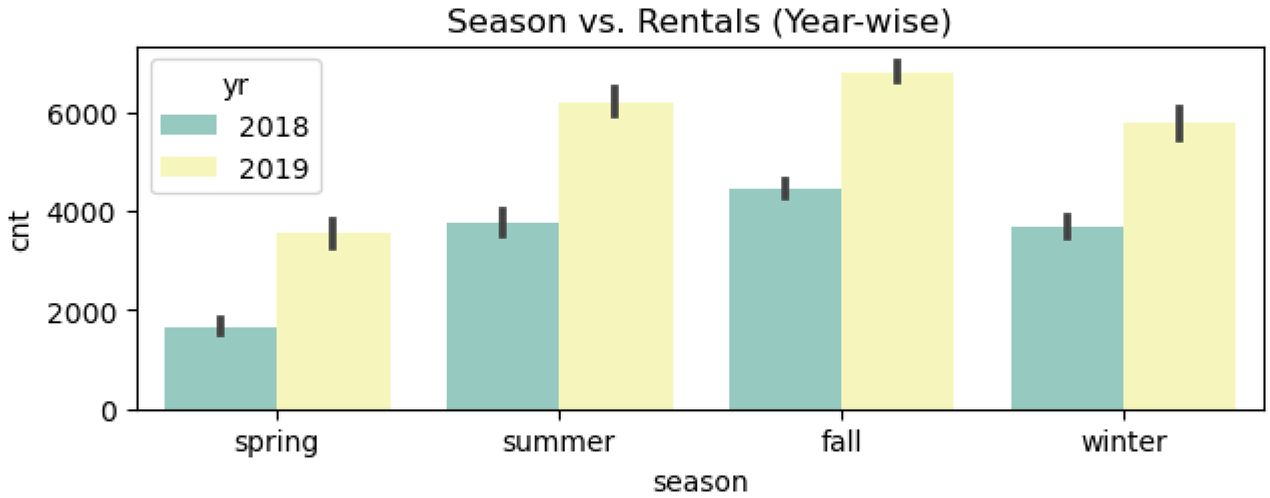
In [20]:

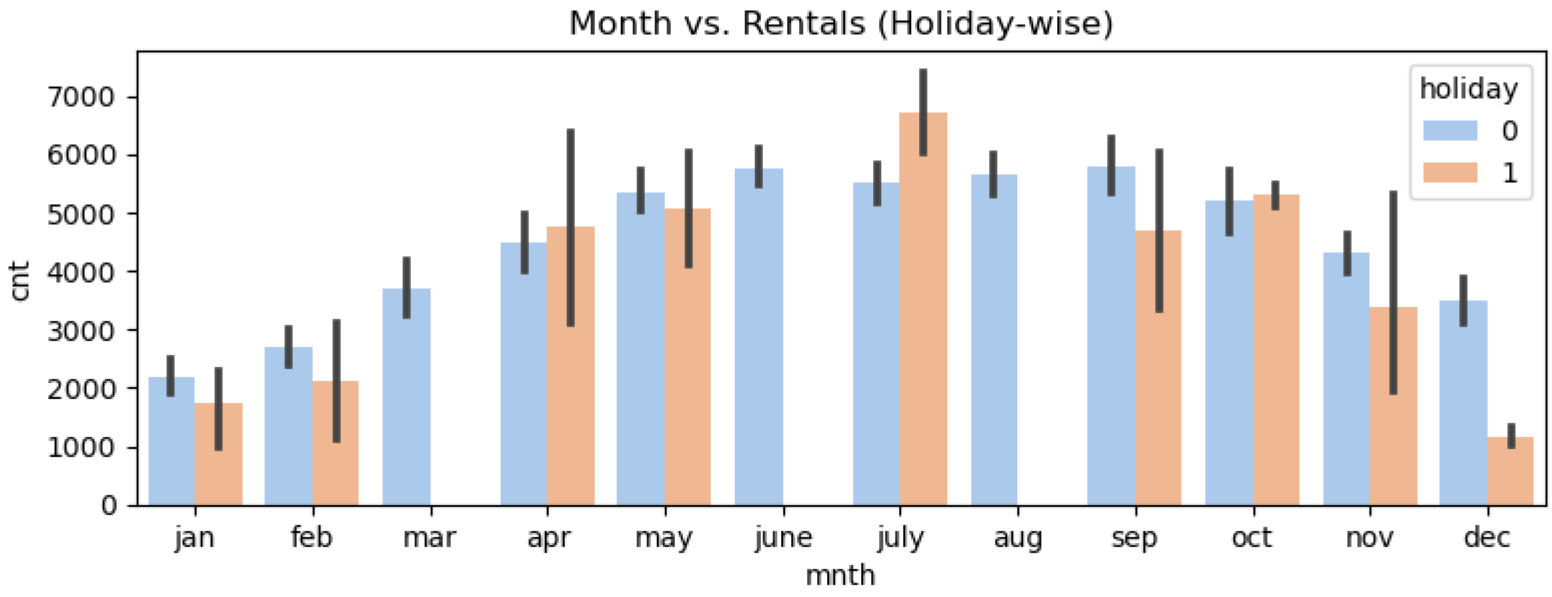
```
# Define custom color palettes or use Seaborn's built-in palettes
custom_palette1 = sns.color_palette("Set3")  # Custom palette 1
custom_palette2 = sns.color_palette("pastel")  # Custom palette 2

# Pairplot for a comprehensive view of variable relationships
sns.pairplot(data=df_for_viz[['temp', 'atemp', 'hum', 'windspeed', 'cnt']], palette=custom_palette1)
plt.suptitle("Multivariate Analysis: Pairplot", y=1.02)
plt.show()

# Multivariate Analysis: Categorical Variables vs. Rentals
plt.figure(figsize=(16, 8))
plt.subplot(3, 2, 1)
sns.barplot(data=df_for_viz, x='season', y='cnt', hue='yr', palette=custom_palette1)
plt.title('Season vs. Rentals (Year-wise)')
```

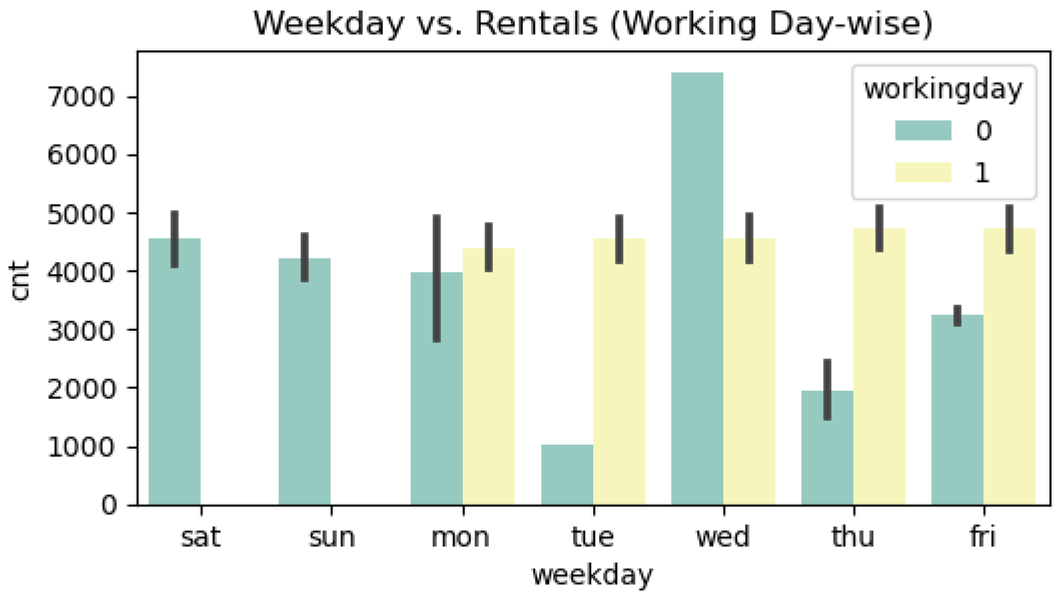
```
plt.figure(figsize=(20, 10))
plt.subplot(3, 2, 2)
sns.barplot(data=df_for_viz, x='mnth', y='cnt', hue='holiday', palette=custom_palette2)
plt.title('Month vs. Rentals (Holiday-wise)')

plt.figure(figsize=(20, 10))
plt.subplot(3, 3, 2)
sns.barplot(data=df_for_viz, x='weekday', y='cnt', hue='workingday', palette=custom_palette1)
plt.title('Weekday vs. Rentals (Working Day-wise)')

plt.figure(figsize=(20, 10))
plt.subplot(3, 3, 4)
sns.boxplot(data=df_for_viz, x='weathersit', y='cnt', hue='yr', palette=custom_palette2)
plt.title('Weather Situation vs. Rentals (Year-wise)')

plt.show()
```

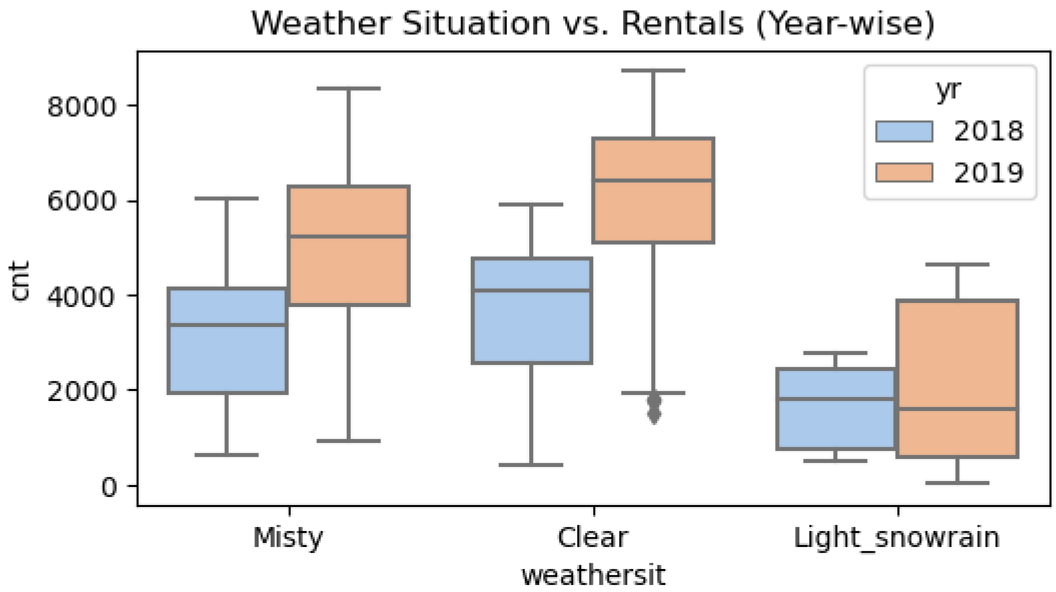

Multivariate Analysis: Pairplot
temp
atemp
hum
windspeed
30
20
10
40
30
20
10
100
80
60
40
20
0
30
20
10


cnt
temp
atemp
hum
windspeed
cnt

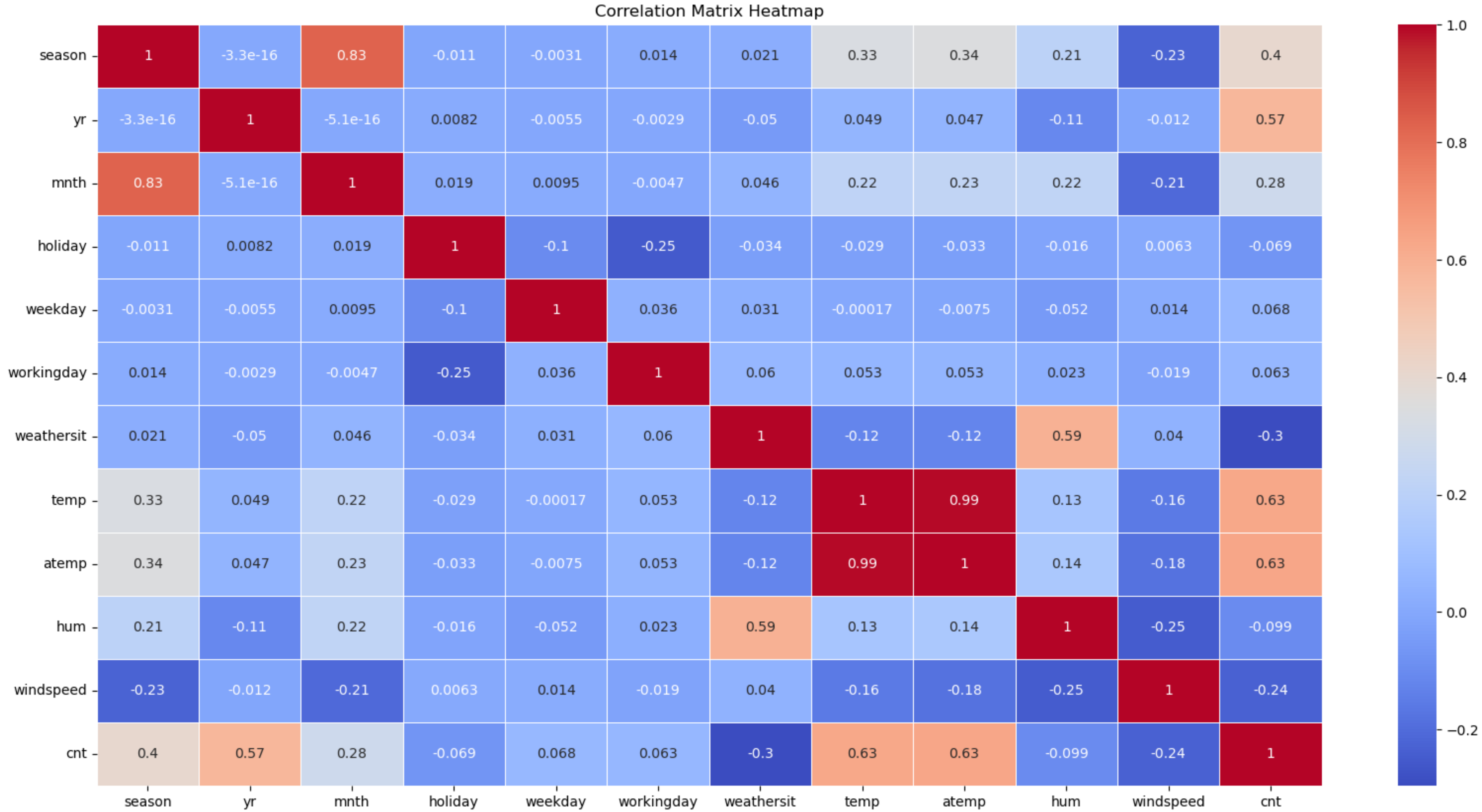

Season vs. Rentals (Year-wise)
yr
2018
2019
cnt
spring
summer
fall
winter
season


Month vs. Rentals (Holiday-wise)
holiday
0
1
cnt
0
1000
2000
3000
4000
5000
6000
7000
jan
feb
mar
apr
may
june
july
aug
sep
oct
nov
dec
mnth


Weekday vs. Rentals (Working Day-wise)
workingday
0
1
cnt
0
1000
2000
3000
4000
5000
6000
7000
sat
sun
mon
tue
wed
thu
fri
weekday



In [21]:
```
# Calculate the correlation matrix
correlation_matrix = df.corr()

# Plot a heatmap to visualize the correlations
plt.figure(figsize=(20, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title("Correlation Matrix Heatmap")
plt.show()

# Sort the features based on their correlation with 'cnt' in descending order
sorted_correlations = correlation_matrix['cnt'].abs().sort_values(ascending=False)
print("Features sorted by correlation with 'cnt':")
print(sorted_correlations)
```


Correlation Matrix Heatmap
season
yr
mnth
holiday
weekday
workingday
weathersit
temp
atemp
hum
windspeed
cnt
1 -3.3e-16 0.83 -0.011 -0.0031 0.014 0.021 0.33 0.34 0.21 -0.23 0.4
-3.3e-16 1 -5.1e-16 0.0082 -0.0055 -0.0029 -0.05 0.049 0.047 -0.11 -0.012 0.57
0.83 -5.1e-16 1 0.019 0.0095 -0.0047 0.046 0.22 0.23 0.22 -0.21 0.28
-0.011 0.0082 0.019 1 -0.1 -0.25 -0.034 -0.029 -0.033 -0.016 0.0063 -0.069
-0.0031 -0.0055 0.0095 -0.1 1 0.036 0.031 -0.00017 -0.0075 -0.052 0.014 0.068
0.014 -0.0029 -0.0047 -0.25 0.036 1 0.06 0.053 0.053 0.023 -0.019 0.063
0.021 -0.05 0.046 -0.034 0.031 0.06 1 -0.12 -0.12 0.59 0.04 -0.3
0.33 0.049 0.22 -0.029 -0.00017 0.053 -0.12 1 0.99 0.13 -0.16 0.63
0.34 0.047 0.23 -0.033 -0.0075 0.053 -0.12 0.99 1 0.14 -0.18 0.63
0.21 -0.11 0.22 -0.016 -0.052 0.023 0.59 0.13 0.14 1 -0.25 -0.099
-0.23 -0.012 -0.21 0.0063 0.014 -0.019 0.04 -0.16 -0.18 -0.25 1 -0.24
0.4 0.57 0.28 -0.069 0.068 0.063 -0.3 0.63 0.63 -0.099 -0.24 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

```
Features sorted by correlation with 'cnt':
cnt           1.000000
atemp         0.630685
temp          0.627044
yr            0.569728
season        0.404584
weathersit    0.295929
mnth          0.278191
windspeed     0.235132
hum           0.098543
holiday       0.068764
weekday       0.067534
workingday    0.062542
Name: cnt, dtype: float64
```

Data Preprocessing

In [22]:

```
df
```

Out[22]:

	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	cnt
0	1	0	1	0	6	0	2	14.110847	18.18125	80.5833	10.749882	985
1	1	0	1	0	0	0	2	14.902598	17.68695	69.6087	16.652113	801
2	1	0	1	0	1	1	1	8.050924	9.47025	43.7273	16.636703	1349
3	1	0	1	0	2	1	1	8.200000	10.60610	59.0435	10.739832	1562
4	1	0	1	0	3	1	1	9.305237	11.46350	43.6957	12.522300	1600
...	...	...	...	...	...	...	...	...	...	...	...	...
725	1	1	12	0	4	1	2	10.420847	11.33210	65.2917	23.458911	2114
726	1	1	12	0	5	1	2	10.386653	12.75230	59.0000	10.416557	3095
727	1	1	12	0	6	0	2	10.386653	12.12000	75.2917	8.333661	1341
728	1	1	12	0	0	0	1	10.489153	11.58500	48.3333	23.500518	1796
729	1	1	12	0	1	1	2	8.849153	11.17435	57.7500	10.374682	2729

730 rows × 12 columns

In [23]:

```
df.nunique()
```

Out[23]:

```
season          4
yr              2
mnth           12
holiday         2
weekday         7
workingday      2
weathersit      3
temp          498
atemp         689
hum           594
windspeed     649
cnt           695
dtype: int64
```

In [24]:

```
#converting the mnth column which is in numeric to object
import calendar
df['mnth'] = df['mnth'].apply(lambda x: calendar.month_abbr[x])
```

In [25]:
```python
# Maping seasons
df.season = df.season.map({1: 'Spring',2:'Summer',3:'Fall',4:'Winter'})
# Mapping weathersit
df.weathersit = df.weathersit.map({1:'Clear',2:'Mist & Cloudy',3:'Light Snow & Rain',4:'Heavy Snow & Rain'})
#Mapping Weekday
df.weekday = df.weekday.map({0:"Sunday",1:"Monday",2:"Tuesday",3:"Wednesday",4:"Thrusday",5:"Friday",6:"Saturday"})
```

In [26]:
```python
#checking mapped df
df.head()
```

Out[26]:

	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	cnt
0	Spring	0	Jan	0	Saturday	0	Mist & Cloudy	14.110847	18.18125	80.5833	10.749882	985
1	Spring	0	Jan	0	Sunday	0	Mist & Cloudy	14.902598	17.68695	69.6087	16.652113	801
2	Spring	0	Jan	0	Monday	1	Clear	8.050924	9.47025	43.7273	16.636703	1349
3	Spring	0	Jan	0	Tuesday	1	Clear	8.200000	10.60610	59.0435	10.739832	1562
4	Spring	0	Jan	0	Wednesday	1	Clear	9.305237	11.46350	43.6957	12.522300	1600

Creating Dummy Variables

The variables `season` `mnth` `weekday` `weathersit` have various levels, for ex, `weathersit` has 3 levels , similarly variable `mnth` has 12 levels.
We will create DUMMY variables for these 4 categorical variables namely - `mnth` , `weekday` , `season` & `weathersit` .

In [27]:
```python
df = pd.get_dummies(data=df,columns=['season','mnth','weekday','weathersit'],drop_first=True)
```

In [28]:
```python
df.columns
```

Out[28]:
```
Index(['yr', 'holiday', 'workingday', 'temp', 'atemp', 'hum', 'windspeed',
       'cnt', 'season_Spring', 'season_Summer', 'season_Winter', 'mnth_Aug',
       'mnth_Dec', 'mnth_Feb', 'mnth_Jan', 'mnth_Jul', 'mnth_Jun', 'mnth_Mar',
       'mnth_May', 'mnth_Nov', 'mnth_Oct', 'mnth_Sep', 'weekday_Monday',
       'weekday_Saturday', 'weekday_Sunday', 'weekday_Thrusday',
       'weekday_Tuesday', 'weekday_Wednesday', 'weathersit_Light Snow & Rain',
       'weathersit_Mist & Cloudy'],
      dtype='object')
```

In [29]:
```
df.head()
```

Out[29]:

	yr	holiday	workingday	temp	atemp	hum	windspeed	cnt	season_Spring	season_Summer	...	mnth_Oct	mnth_Sep	weekc
0	0	0	0	14.110847	18.18125	80.5833	10.749882	985	1	0	...	0	0	
1	0	0	0	14.902598	17.68695	69.6087	16.652113	801	1	0	...	0	0	
2	0	0	1	8.050924	9.47025	43.7273	16.636703	1349	1	0	...	0	0	
3	0	0	1	8.200000	10.60610	59.0435	10.739832	1562	1	0	...	0	0	
4	0	0	1	9.305237	11.46350	43.6957	12.522300	1600	1	0	...	0	0	

5 rows × 30 columns

Rescaling the Features

Rescaling is needed in multiple linear regression (MLR) to ensure that all predictor variables are on the same scale, preventing variables with larger ranges from dominating the regression process and ensuring accurate coefficient interpretation.

In [30]:
```
# Apply scaler() to all the columns except the 'yes-no' and 'dummy' variables
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scale1 = ['cnt', 'hum', 'windspeed','temp','atemp']
df[scale1] = scaler.fit_transform(df[scale1])
df
```

Out[30]:

	yr	holiday	workingday	temp	atemp	hum	windspeed	cnt	season_Spring	season_Summer	...	mnth_Oct	mnth_Sep
0	0	0	0	0.355170	0.373517	0.828620	0.284606	0.110792	1	0	...	0	0
1	0	0	0	0.379232	0.360541	0.715771	0.466215	0.089623	1	0	...	0	0
2	0	0	1	0.171000	0.144830	0.449638	0.465740	0.152669	1	0	...	0	0
3	0	0	1	0.175530	0.174649	0.607131	0.284297	0.177174	1	0	...	0	0
4	0	0	1	0.209120	0.197158	0.449313	0.339143	0.181546	1	0	...	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
725	1	0	1	0.243025	0.193709	0.671380	0.675656	0.240681	1	0	...	0	0
726	1	0	1	0.241986	0.230993	0.606684	0.274350	0.353543	1	0	...	0	0
727	1	0	0	0.241986	0.214393	0.774208	0.210260	0.151749	1	0	...	0	0
728	1	0	0	0.245101	0.200348	0.497001	0.676936	0.204096	1	0	...	0	0
729	1	0	1	0.195259	0.189567	0.593830	0.273062	0.311436	1	0	...	0	0

730 rows × 30 columns

Splitting data into Train and Test data

In [31]:

```
#X is all remaining variable also our independent variables
X=df.drop('cnt',axis=1)
#y to contain only target variable
y=df['cnt']
```

In [32]:

```
#Train Test split with 70:30 ratio
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=100)
X_train.head()
```

Out[32]:

	yr	holiday	workingday	temp	atemp	hum	windspeed	season_Spring	season_Summer	season_Winter	...	mnth_Oct	mntl
653	1	0	1	0.509887	0.501133	0.574121	0.329497	0	0	1	...	1	
576	1	0	1	0.815169	0.766351	0.724079	0.294871	0	0	0	...	0	
426	1	0	0	0.442393	0.438975	0.638817	0.285911	1	0	0	...	0	
728	1	0	0	0.245101	0.200348	0.497001	0.676936	1	0	0	...	0	
482	1	0	0	0.395666	0.391735	0.503427	0.221789	0	1	0	...	0	

5 rows × 29 columns

In [33]:

```
X_train.describe()
```

Out[33]:

	yr	holiday	workingday	temp	atemp	hum	windspeed	season_Spring	season_Summer	season_Winter
count	511.000000	511.000000	511.000000	511.000000	511.000000	511.000000	511.000000	511.000000	511.000000	511.000000
mean	0.508806	0.025440	0.677104	0.537386	0.513133	0.648940	0.348724	0.242661	0.246575	0.248532
std	0.500412	0.157613	0.468042	0.225640	0.212202	0.145429	0.162675	0.429112	0.431440	0.432585
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.041051	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.341151	0.332910	0.537703	0.232053	0.000000	0.000000	0.000000
50%	1.000000	0.000000	1.000000	0.542077	0.529300	0.652100	0.326911	0.000000	0.000000	0.000000
75%	1.000000	0.000000	1.000000	0.735215	0.688457	0.752785	0.438475	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	0.997858	1.000000	1.000000	1.000000	1.000000

8 rows × 29 columns

In [34]:

```
X_test.head()
```

Out[34]:

	yr	holiday	workingday	temp	atemp	hum	windspeed	season_Spring	season_Summer	season_Winter	...	mnth_Oct	mnth
184	0	1	0	0.831783	0.769660	0.655956	0.121812	0	0	0	...	0	
535	1	0	1	0.901354	0.842587	0.608826	0.188468	0	1	0	...	0	
299	0	0	1	0.511964	0.496145	0.835904	0.361537	0	0	1	...	1	
221	0	0	1	0.881625	0.795343	0.436161	0.366681	0	0	0	...	0	
152	0	0	1	0.817246	0.741471	0.313625	0.556403	0	1	0	...	0	

5 rows × 29 columns

In [35]:

```
# Checking shape and size for train and test
print('X_train :',X_train.shape)
print('X_test :',X_test.shape)
print('y_train :',y_train.shape)
print('y_test :',y_test.shape)
```

```
X_train : (511, 29)
X_test : (219, 29)
y_train : (511,)
y_test : (219,)
```

Building model

Building model using statsmodel, for the detailed statistics

Model 1

In [36]:

```
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [37]:

```
#Build a linear model
X_train_lm1 = sm.add_constant(X_train)
```

```
lr_1 = sm.OLS(y_train, X_train_lm1).fit()
lr_1.params
```

Out[37]:
```
const                           0.250104
yr                              0.232986
holiday                         0.012658
workingday                      0.098022
temp                            0.386163
atemp                           0.060120
hum                            -0.153379
windspeed                      -0.191652
season_Spring                  -0.045451
season_Summer                   0.042388
season_Winter                   0.106914
mnth_Aug                        0.022625
mnth_Dec                       -0.043962
mnth_Feb                       -0.032352
mnth_Jan                       -0.062992
mnth_Jul                       -0.032381
mnth_Jun                        0.006304
mnth_Mar                        0.001672
mnth_May                        0.025799
mnth_Nov                       -0.039581
mnth_Oct                        0.010721
mnth_Sep                        0.087206
weekday_Monday                 -0.021745
weekday_Saturday                0.096876
weekday_Sunday                  0.042547
weekday_Thrusday               -0.009692
weekday_Tuesday                -0.016842
weekday_Wednesday              -0.005870
weathersit_Light Snow & Rain   -0.255588
weathersit_Mist & Cloudy       -0.059788
dtype: float64
```

In [38]:
```
print(lr_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.853
Model:                            OLS   Adj. R-squared:                  0.844
Method:                 Least Squares   F-statistic:                     99.76
Date:                Fri, 06 Oct 2023   Prob (F-statistic):          7.47e-181
Time:                        12:33:52   Log-Likelihood:                 527.86
No. Observations:                 511   AIC:                            -997.7
Df Residuals:                     482   BIC:                            -874.9
Df Model:                          28
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                            0.2501      0.036      7.018      0.000       0.180       0.320
yr                               0.2330      0.008     28.864      0.000       0.217       0.249
holiday                          0.0127      0.024      0.523      0.601      -0.035       0.060
workingday                       0.0980      0.012      7.942      0.000       0.074       0.122
temp                             0.3862      0.142      2.713      0.007       0.106       0.666
atemp                            0.0601      0.139      0.434      0.664      -0.212       0.332
hum                             -0.1534      0.039     -3.963      0.000      -0.229      -0.077
windspeed                       -0.1917      0.028     -6.965      0.000      -0.246      -0.138
season_Spring                   -0.0455      0.030     -1.513      0.131      -0.104       0.014
season_Summer                    0.0424      0.026      1.617      0.107      -0.009       0.094
season_Winter                    0.1069      0.028      3.818      0.000       0.052       0.162
mnth_Aug                         0.0226      0.034      0.668      0.505      -0.044       0.089
mnth_Dec                        -0.0440      0.034     -1.306      0.192      -0.110       0.022
mnth_Feb                        -0.0324      0.033     -0.981      0.327      -0.097       0.032
mnth_Jan                        -0.0630      0.034     -1.873      0.062      -0.129       0.003
mnth_Jul                        -0.0324      0.035     -0.923      0.356      -0.101       0.037
mnth_Jun                         0.0063      0.025      0.252      0.801      -0.043       0.055
mnth_Mar                         0.0017      0.025      0.068      0.946      -0.047       0.050
mnth_May                         0.0258      0.021      1.219      0.223      -0.016       0.067
mnth_Nov                        -0.0396      0.036     -1.086      0.278      -0.111       0.032
mnth_Oct                         0.0107      0.036      0.299      0.765      -0.060       0.081
mnth_Sep                         0.0872      0.032      2.723      0.007       0.024       0.150
weekday_Monday                  -0.0217      0.015     -1.408      0.160      -0.052       0.009
weekday_Saturday                 0.0969      0.014      7.002      0.000       0.070       0.124
weekday_Sunday                   0.0425      0.014      3.030      0.003       0.015       0.070
weekday_Thrusday                -0.0097      0.016     -0.620      0.535      -0.040       0.021
weekday_Tuesday                 -0.0168      0.016     -1.085      0.278      -0.047       0.014
weekday_Wednesday               -0.0059      0.015     -0.392      0.695      -0.035       0.024
weathersit_Light Snow & Rain    -0.2556      0.026     -9.650      0.000      -0.308      -0.204
weathersit_Mist & Cloudy        -0.0598      0.010     -5.725      0.000      -0.080      -0.039
```

```
==============================================================================
Omnibus:                       85.644   Durbin-Watson:                   2.042
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              240.466
Skew:                          -0.811   Prob(JB):                     6.07e-53
Kurtosis:                       5.944   Cond. No.                     1.94e+15
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 4.6e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

`VIF`

helps identify multicollinearity, where predictor variables are highly correlated, which can affect the model's reliability.

$$VIF_i = \frac{1}{1 - {R_i}^2}$$

In [39]:

```
# Create a dataframe that will contain the names of all the feature variables and their respective VIFs
vif = pd.DataFrame()
vif['Features'] = X_train.columns
vif['VIF'] = [variance_inflation_factor(X_train.values, i) for i in range(X_train.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[39]:

	Features	VIF
2	workingday	87.20
3	temp	66.89
4	atemp	56.03
22	weekday_Saturday	20.04
23	weekday_Sunday	18.65
7	season_Spring	10.77
9	season_Winter	9.52
8	season_Summer	8.30
18	mnth_Nov	6.79
19	mnth_Oct	6.57
10	mnth_Aug	6.47
13	mnth_Jan	5.90
11	mnth_Dec	5.68
14	mnth_Jul	5.64
20	mnth_Sep	4.92
1	holiday	4.56
12	mnth_Feb	4.39
16	mnth_Mar	3.47
15	mnth_Jun	2.86
17	mnth_May	2.24
5	hum	2.05
21	weekday_Monday	1.98
26	weekday_Wednesday	1.94
24	weekday_Thrusday	1.83
25	weekday_Tuesday	1.81

	Features	VIF
28	weathersit_Mist & Cloudy	1.60
6	windspeed	1.30
27	weathersit_Light Snow & Rain	1.30
0	yr	1.06

`Model 2`

Dropping the Variable and Updating the Model

As you can notice some of the variable have high VIF values as well as high p-values. Such variables are insignificant and should be dropped.

As you might have noticed, the variable `mnth_mar` has a significantly high VIF (`3.47`) and a high p-value (`0.946`) as well. Hence, this variable isn't of much use and should be dropped.

In [40]:
```
# Dropping highly correlated variables and insignificant variables
X_ud_2 = X_train.drop('mnth_Mar', axis=1)
```

In [41]:
```
# Build a third fitted model
X_train_lm2 = sm.add_constant(X_ud_2)
lr_2 = sm.OLS(y_train, X_train_lm2).fit()
print(lr_2.summary())
```

OLS Regression Results

Dep. Variable:	cnt	R-squared:	0.853
Model:	OLS	Adj. R-squared:	0.845
Method:	Least Squares	F-statistic:	103.7
Date:	Fri, 06 Oct 2023	Prob (F-statistic):	7.28e-182
Time:	12:33:52	Log-Likelihood:	527.86
No. Observations:	511	AIC:	-999.7
Df Residuals:	483	BIC:	-881.1
Df Model:	27		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.2509	0.033	7.517	0.000	0.185	0.317
yr	0.2330	0.008	28.894	0.000	0.217	0.249
holiday	0.0128	0.024	0.535	0.593	-0.034	0.060
workingday	0.0982	0.012	8.264	0.000	0.075	0.122
temp	0.3859	0.142	2.715	0.007	0.107	0.665
atemp	0.0599	0.138	0.433	0.665	-0.212	0.332
hum	-0.1532	0.039	-3.975	0.000	-0.229	-0.077
windspeed	-0.1918	0.027	-6.986	0.000	-0.246	-0.138
season_Spring	-0.0448	0.028	-1.576	0.116	-0.101	0.011
season_Summer	0.0420	0.026	1.636	0.103	-0.008	0.093
season_Winter	0.1073	0.027	3.903	0.000	0.053	0.161
mnth_Aug	0.0219	0.032	0.685	0.494	-0.041	0.085
mnth_Dec	-0.0454	0.026	-1.746	0.081	-0.097	0.006
mnth_Feb	-0.0340	0.022	-1.523	0.128	-0.078	0.010
mnth_Jan	-0.0647	0.022	-2.892	0.004	-0.109	-0.021
mnth_Jul	-0.0331	0.033	-0.992	0.322	-0.099	0.032
mnth_Jun	0.0058	0.024	0.244	0.808	-0.041	0.052
mnth_May	0.0253	0.020	1.278	0.202	-0.014	0.064
mnth_Nov	-0.0409	0.031	-1.327	0.185	-0.101	0.020
mnth_Oct	0.0095	0.031	0.309	0.758	-0.051	0.070
mnth_Sep	0.0863	0.029	2.973	0.003	0.029	0.143
weekday_Monday	-0.0217	0.015	-1.409	0.160	-0.052	0.009
weekday_Saturday	0.0971	0.013	7.255	0.000	0.071	0.123
weekday_Sunday	0.0427	0.014	3.115	0.002	0.016	0.070
weekday_Thrusday	-0.0096	0.016	-0.619	0.537	-0.040	0.021
weekday_Tuesday	-0.0168	0.016	-1.086	0.278	-0.047	0.014
weekday_Wednesday	-0.0059	0.015	-0.392	0.695	-0.035	0.024
weathersit_Light Snow & Rain	-0.2556	0.026	-9.660	0.000	-0.308	-0.204
weathersit_Mist & Cloudy	-0.0598	0.010	-5.732	0.000	-0.080	-0.039

```
Omnibus:                       85.601   Durbin-Watson:                   2.041
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              240.715
Skew:                          -0.810   Prob(JB):                     5.36e-53
Kurtosis:                       5.947   Cond. No.                     1.96e+15
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 4.52e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [42]:

```
# Calculate the VIFs again for the new model
vif = pd.DataFrame()
vif['Features'] = X_ud_2.columns
vif['VIF'] = [variance_inflation_factor(X_ud_2.values, i) for i in range(X_ud_2.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[42]:

	Features	VIF
2	workingday	76.43
3	temp	66.84
4	atemp	55.99
21	weekday_Saturday	17.53
22	weekday_Sunday	16.48
7	season_Spring	9.66
9	season_Winter	9.19
8	season_Summer	7.99
10	mnth_Aug	5.74
14	mnth_Jul	5.11
17	mnth_Nov	4.86
18	mnth_Oct	4.80
1	holiday	4.18
19	mnth_Sep	4.05
11	mnth_Dec	3.39
13	mnth_Jan	2.62
15	mnth_Jun	2.56
5	hum	2.04
12	mnth_Feb	2.02
20	weekday_Monday	1.98
16	mnth_May	1.96
25	weekday_Wednesday	1.94
23	weekday_Thrusday	1.83
24	weekday_Tuesday	1.81
27	weathersit_Mist & Cloudy	1.60

	Features	VIF
6	windspeed	1.30
26	weathersit_Light Snow & Rain	1.30
0	yr	1.06

`Model 3`

Dropping the Variable and Updating the Model

As you can notice some of the variable have high VIF values as well as high p-values. Such variables are insignificant and should be dropped.

As you might have noticed, the variable `atemp` has a significantly high VIF (`55.99`) and a high p-value (`0.665`) as well. Hence, this variable isn't of much use and should be dropped.

In [43]:
```
# Dropping highly correlated variables and insignificant variables
X_ud_3 = X_ud_2.drop('atemp', axis=1)
```

In [44]:
```
# Build a third fitted model
X_train_lm3 = sm.add_constant(X_ud_3)
lr_3 = sm.OLS(y_train, X_train_lm3).fit()
print(lr_3.summary())
```

OLS Regression Results

Dep. Variable:	cnt	R-squared:	0.853
Model:	OLS	Adj. R-squared:	0.845
Method:	Least Squares	F-statistic:	107.8
Date:	Fri, 06 Oct 2023	Prob (F-statistic):	7.61e-183
Time:	12:33:52	Log-Likelihood:	527.76
No. Observations:	511	AIC:	-1002.
Df Residuals:	484	BIC:	-887.1
Df Model:	26		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.2507	0.033	7.517	0.000	0.185	0.316
yr	0.2329	0.008	28.918	0.000	0.217	0.249
holiday	0.0120	0.024	0.501	0.617	-0.035	0.059
workingday	0.0981	0.012	8.263	0.000	0.075	0.121
temp	0.4442	0.046	9.741	0.000	0.355	0.534
hum	-0.1527	0.038	-3.967	0.000	-0.228	-0.077
windspeed	-0.1943	0.027	-7.250	0.000	-0.247	-0.142
season_Spring	-0.0444	0.028	-1.564	0.118	-0.100	0.011
season_Summer	0.0427	0.026	1.668	0.096	-0.008	0.093
season_Winter	0.1078	0.027	3.927	0.000	0.054	0.162
mnth_Aug	0.0201	0.032	0.635	0.526	-0.042	0.082
mnth_Dec	-0.0452	0.026	-1.738	0.083	-0.096	0.006
mnth_Feb	-0.0338	0.022	-1.517	0.130	-0.078	0.010
mnth_Jan	-0.0647	0.022	-2.893	0.004	-0.109	-0.021
mnth_Jul	-0.0342	0.033	-1.028	0.304	-0.100	0.031
mnth_Jun	0.0044	0.023	0.189	0.850	-0.042	0.050
mnth_May	0.0245	0.020	1.245	0.214	-0.014	0.063
mnth_Nov	-0.0407	0.031	-1.323	0.186	-0.101	0.020
mnth_Oct	0.0095	0.031	0.311	0.756	-0.051	0.070
mnth_Sep	0.0856	0.029	2.956	0.003	0.029	0.143
weekday_Monday	-0.0210	0.015	-1.372	0.171	-0.051	0.009
weekday_Saturday	0.0975	0.013	7.302	0.000	0.071	0.124
weekday_Sunday	0.0431	0.014	3.151	0.002	0.016	0.070
weekday_Thrusday	-0.0090	0.016	-0.583	0.560	-0.040	0.021
weekday_Tuesday	-0.0163	0.015	-1.056	0.291	-0.047	0.014
weekday_Wednesday	-0.0055	0.015	-0.366	0.715	-0.035	0.024
weathersit_Light Snow & Rain	-0.2565	0.026	-9.730	0.000	-0.308	-0.205
weathersit_Mist & Cloudy	-0.0598	0.010	-5.740	0.000	-0.080	-0.039

Omnibus:	84.837	Durbin-Watson:	2.040

```
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              237.805
Skew:                          -0.804   Prob(JB):                     2.30e-52
Kurtosis:                       5.930   Cond. No.                     1.88e+15
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 4.53e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [45]:

```python
# Calculate the VIFs again for the new model
vif = pd.DataFrame()
vif['Features'] = X_ud_3.columns
vif['VIF'] = [variance_inflation_factor(X_ud_3.values, i) for i in range(X_ud_3.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[45]:

	Features	VIF
2	workingday	76.40
20	weekday_Saturday	17.53
21	weekday_Sunday	16.48
6	season_Spring	9.65
8	season_Winter	9.17
7	season_Summer	7.96
3	temp	6.89
9	mnth_Aug	5.64
13	mnth_Jul	5.08
16	mnth_Nov	4.86
17	mnth_Oct	4.80
1	holiday	4.17
18	mnth_Sep	4.04
10	mnth_Dec	3.39
12	mnth_Jan	2.62
14	mnth_Jun	2.52
4	hum	2.04
11	mnth_Feb	2.02
19	weekday_Monday	1.96
15	mnth_May	1.95
24	weekday_Wednesday	1.93
22	weekday_Thrusday	1.81
23	weekday_Tuesday	1.80
26	weathersit_Mist & Cloudy	1.60
25	weathersit_Light Snow & Rain	1.29

	Features	VIF
5	windspeed	1.24
0	yr	1.06

`Model 4`

Dropping the Variable and Updating the Model

As you can notice some of the variable have high VIF values as well as high p-values. Such variables are insignificant and should be dropped.

As you might have noticed, the variable `season_Spring` has a significantly high VIF (`9.65`) and a high p-value (`0.111`) as well. Hence, this variable isn't of much use and should be dropped.

In [46]:
```
# Dropping highly correlated variables and insignificant variables
X_ud_4 = X_ud_3.drop('season_Spring', axis=1)
```

In [47]:
```
# Build a third fitted model
X_train_lm4 = sm.add_constant(X_ud_4)
lr_4 = sm.OLS(y_train, X_train_lm4).fit()
print(lr_4.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.852
Model:                            OLS   Adj. R-squared:                  0.844
Method:                 Least Squares   F-statistic:                     111.7
Date:                Fri, 06 Oct 2023   Prob (F-statistic):          2.39e-183
Time:                        12:33:52   Log-Likelihood:                 526.47
No. Observations:                 511   AIC:                            -1001.
Df Residuals:                     485   BIC:                            -890.8
Df Model:                          25
Covariance Type:            nonrobust
==================================================================================================
                                     coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------
const                              0.2197      0.027      8.173      0.000       0.167       0.273
yr                                 0.2325      0.008     28.842      0.000       0.217       0.248
holiday                            0.0039      0.023      0.167      0.868      -0.042       0.050
workingday                         0.0909      0.011      8.298      0.000       0.069       0.112
temp                               0.4580      0.045     10.223      0.000       0.370       0.546
hum                               -0.1567      0.038     -4.075      0.000      -0.232      -0.081
windspeed                         -0.1971      0.027     -7.362      0.000      -0.250      -0.145
season_Summer                      0.0736      0.016      4.496      0.000       0.041       0.106
season_Winter                      0.1327      0.022      5.931      0.000       0.089       0.177
mnth_Aug                           0.0509      0.025      2.056      0.040       0.002       0.099
mnth_Dec                          -0.0387      0.026     -1.506      0.133      -0.089       0.012
mnth_Feb                          -0.0403      0.022     -1.838      0.067      -0.083       0.003
mnth_Jan                          -0.0702      0.022     -3.175      0.002      -0.114      -0.027
mnth_Jul                          -0.0043      0.027     -0.159      0.874      -0.058       0.049
mnth_Jun                           0.0158      0.022      0.707      0.480      -0.028       0.060
mnth_May                           0.0267      0.020      1.358      0.175      -0.012       0.065
mnth_Nov                          -0.0292      0.030     -0.975      0.330      -0.088       0.030
mnth_Oct                           0.0193      0.030      0.643      0.521      -0.040       0.078
mnth_Sep                           0.1125      0.023      4.816      0.000       0.067       0.158
weekday_Monday                    -0.0206      0.015     -1.343      0.180      -0.051       0.010
weekday_Saturday                   0.0893      0.012      7.259      0.000       0.065       0.113
weekday_Sunday                     0.0356      0.013      2.775      0.006       0.010       0.061
weekday_Thrusday                  -0.0090      0.016     -0.579      0.563      -0.040       0.022
weekday_Tuesday                   -0.0164      0.015     -1.063      0.288      -0.047       0.014
weekday_Wednesday                 -0.0060      0.015     -0.399      0.690      -0.035       0.023
weathersit_Light Snow & Rain      -0.2537      0.026     -9.632      0.000      -0.305      -0.202
weathersit_Mist & Cloudy          -0.0592      0.010     -5.680      0.000      -0.080      -0.039
==============================================================================
Omnibus:                       82.235   Durbin-Watson:                   2.050
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              221.859
```

```
Skew:                          -0.792   Prob(JB):                     6.67e-49
Kurtosis:                       5.812   Cond. No.                     1.86e+15
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 4.54e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [48]:

```
# Calculate the VIFs again for the new model
vif = pd.DataFrame()
vif['Features'] = X_ud_4.columns
vif['VIF'] = [variance_inflation_factor(X_ud_4.values, i) for i in range(X_ud_4.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[48]:

	Features	VIF
2	workingday	50.09
19	weekday_Saturday	11.22
20	weekday_Sunday	10.85
3	temp	6.63
7	season_Winter	6.08
16	mnth_Oct	4.60
15	mnth_Nov	4.58
8	mnth_Aug	3.45
12	mnth_Jul	3.41
9	mnth_Dec	3.31
6	season_Summer	3.24
1	holiday	3.14
17	mnth_Sep	2.62
11	mnth_Jan	2.55
13	mnth_Jun	2.28
4	hum	2.03
18	weekday_Monday	1.96
10	mnth_Feb	1.95
14	mnth_May	1.94
23	weekday_Wednesday	1.93
21	weekday_Thrusday	1.81
22	weekday_Tuesday	1.80
25	weathersit_Mist & Cloudy	1.60
24	weathersit_Light Snow & Rain	1.29
5	windspeed	1.23

	Features	VIF
0	yr	1.06

`Model 5`

Dropping the Variable and Updating the Model

As you can notice some of the variable have high VIF values as well as high p-values. Such variables are insignificant and should be dropped.

As you might have noticed, the variable `weekday_Thrusday` has a significantly high VIF (`1.81`) and a high p-value (`0.563`) as well. Hence, this variable isn't of much use and should be dropped.

In [49]:
```
# Dropping highly correlated variables and insignificant variables
X_ud_5= X_ud_4.drop('weekday_Thrusday', axis=1)
```

In [50]:
```
# Build a third fitted model
X_train_lm5 = sm.add_constant(X_ud_5)
lr_5 = sm.OLS(y_train, X_train_lm5).fit()
print(lr_5.summary())
```

OLS Regression Results

Dep. Variable:	cnt	R-squared:	0.852
Model:	OLS	Adj. R-squared:	0.845
Method:	Least Squares	F-statistic:	116.5
Date:	Fri, 06 Oct 2023	Prob (F-statistic):	2.58e-184
Time:	12:33:52	Log-Likelihood:	526.29
No. Observations:	511	AIC:	-1003.
Df Residuals:	486	BIC:	-896.7
Df Model:	24		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.2180	0.027	8.166	0.000	0.166	0.270
yr	0.2325	0.008	28.858	0.000	0.217	0.248
holiday	0.0010	0.023	0.045	0.964	-0.044	0.046
workingday	0.0882	0.010	8.904	0.000	0.069	0.108
temp	0.4576	0.045	10.223	0.000	0.370	0.546
hum	-0.1570	0.038	-4.086	0.000	-0.232	-0.081
windspeed	-0.1971	0.027	-7.368	0.000	-0.250	-0.145
season_Summer	0.0737	0.016	4.506	0.000	0.042	0.106
season_Winter	0.1330	0.022	5.948	0.000	0.089	0.177
mnth_Aug	0.0512	0.025	2.073	0.039	0.003	0.100
mnth_Dec	-0.0388	0.026	-1.514	0.131	-0.089	0.012
mnth_Feb	-0.0400	0.022	-1.823	0.069	-0.083	0.003
mnth_Jan	-0.0702	0.022	-3.176	0.002	-0.114	-0.027
mnth_Jul	-0.0037	0.027	-0.137	0.891	-0.057	0.050
mnth_Jun	0.0159	0.022	0.713	0.476	-0.028	0.060
mnth_May	0.0269	0.020	1.369	0.172	-0.012	0.065
mnth_Nov	-0.0298	0.030	-0.996	0.320	-0.089	0.029
mnth_Oct	0.0195	0.030	0.650	0.516	-0.039	0.078
mnth_Sep	0.1125	0.023	4.818	0.000	0.067	0.158
weekday_Monday	-0.0160	0.013	-1.220	0.223	-0.042	0.010
weekday_Saturday	0.0912	0.012	7.694	0.000	0.068	0.114
weekday_Sunday	0.0375	0.012	3.023	0.003	0.013	0.062
weekday_Tuesday	-0.0119	0.013	-0.893	0.372	-0.038	0.014
weekday_Wednesday	-0.0013	0.013	-0.105	0.917	-0.026	0.023
weathersit_Light Snow & Rain	-0.2544	0.026	-9.678	0.000	-0.306	-0.203
weathersit_Mist & Cloudy	-0.0588	0.010	-5.656	0.000	-0.079	-0.038

Omnibus:	82.345	Durbin-Watson:	2.044
Prob(Omnibus):	0.000	Jarque-Bera (JB):	223.096
Skew:	-0.792	Prob(JB):	3.59e-49

```
Kurtosis:                       5.823   Cond. No.                     1.84e+15
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 4.6e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [51]:

```python
# Calculate the VIFs again for the new model
vif = pd.DataFrame()
vif['Features'] = X_ud_5.columns
vif['VIF'] = [variance_inflation_factor(X_ud_5.values, i) for i in range(X_ud_5.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[51]:

	Features	VIF
2	workingday	47.44
19	weekday_Saturday	11.22
20	weekday_Sunday	10.85
3	temp	6.63
7	season_Winter	6.08
16	mnth_Oct	4.60
15	mnth_Nov	4.58
8	mnth_Aug	3.45
12	mnth_Jul	3.40
9	mnth_Dec	3.31
6	season_Summer	3.24
1	holiday	3.03
17	mnth_Sep	2.62
11	mnth_Jan	2.55
13	mnth_Jun	2.28
4	hum	2.03
10	mnth_Feb	1.94
14	mnth_May	1.94
24	weathersit_Mist & Cloudy	1.59
18	weekday_Monday	1.43
22	weekday_Wednesday	1.38
21	weekday_Tuesday	1.33
23	weathersit_Light Snow & Rain	1.28
5	windspeed	1.23
0	yr	1.06

`Model 6`

Dropping the Variable and Updating the Model

As you can notice some of the variable have high VIF values as well as high p-values. Such variables are insignificant and should be dropped.

As you might have noticed, the variable `workingday` has a significantly high VIF (`4.44`) and a high p-value (`0.000`) as well. Hence, this variable isn't of much use and should be dropped.

In [52]:
```
# Dropping highly correlated variables and insignificant variables
X_ud_6= X_ud_5.drop('workingday', axis=1)
```

In [53]:
```
# Build a third fitted model
X_train_lm6 = sm.add_constant(X_ud_6)
lr_6 = sm.OLS(y_train, X_train_lm6).fit()
print(lr_6.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.852
Model:                            OLS   Adj. R-squared:                  0.845
Method:                 Least Squares   F-statistic:                     116.5
Date:                Fri, 06 Oct 2023   Prob (F-statistic):          2.58e-184
Time:                        12:33:52   Log-Likelihood:                 526.29
No. Observations:                 511   AIC:                            -1003.
Df Residuals:                     486   BIC:                            -896.7
Df Model:                          24
Covariance Type:            nonrobust
================================================================================================
                                  coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                           0.3062      0.033      9.334      0.000       0.242       0.371
yr                              0.2325      0.008     28.858      0.000       0.217       0.248
holiday                        -0.0872      0.027     -3.273      0.001      -0.139      -0.035
temp                            0.4576      0.045     10.223      0.000       0.370       0.546
hum                            -0.1570      0.038     -4.086      0.000      -0.232      -0.081
windspeed                      -0.1971      0.027     -7.368      0.000      -0.250      -0.145
season_Summer                   0.0737      0.016      4.506      0.000       0.042       0.106
season_Winter                   0.1330      0.022      5.948      0.000       0.089       0.177
mnth_Aug                        0.0512      0.025      2.073      0.039       0.003       0.100
mnth_Dec                       -0.0388      0.026     -1.514      0.131      -0.089       0.012
mnth_Feb                       -0.0400      0.022     -1.823      0.069      -0.083       0.003
mnth_Jan                       -0.0702      0.022     -3.176      0.002      -0.114      -0.027
mnth_Jul                       -0.0037      0.027     -0.137      0.891      -0.057       0.050
mnth_Jun                        0.0159      0.022      0.713      0.476      -0.028       0.060
mnth_May                        0.0269      0.020      1.369      0.172      -0.012       0.065
mnth_Nov                       -0.0298      0.030     -0.996      0.320      -0.089       0.029
mnth_Oct                        0.0195      0.030      0.650      0.516      -0.039       0.078
mnth_Sep                        0.1125      0.023      4.818      0.000       0.067       0.158
weekday_Monday                 -0.0160      0.013     -1.220      0.223      -0.042       0.010
weekday_Saturday                0.0030      0.013      0.237      0.813      -0.022       0.028
weekday_Sunday                 -0.0507      0.013     -3.877      0.000      -0.076      -0.025
weekday_Tuesday                -0.0119      0.013     -0.893      0.372      -0.038       0.014
weekday_Wednesday              -0.0013      0.013     -0.105      0.917      -0.026       0.023
weathersit_Light Snow & Rain   -0.2544      0.026     -9.678      0.000      -0.306      -0.203
weathersit_Mist & Cloudy       -0.0588      0.010     -5.656      0.000      -0.079      -0.038
==============================================================================
Omnibus:                       82.345   Durbin-Watson:                   2.044
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              223.096
Skew:                          -0.792   Prob(JB):                     3.59e-49
Kurtosis:                       5.823   Cond. No.                         24.5
```

```
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [54]:
```
# Calculate the VIFs again for the new model

vif = pd.DataFrame()
vif['Features'] = X_ud_6.columns
vif['VIF'] = [variance_inflation_factor(X_ud_6.values, i) for i in range(X_ud_6.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[54]:

	Features	VIF
2	temp	36.21
3	hum	29.63
6	season_Winter	8.08
4	windspeed	5.23
15	mnth_Oct	5.00
14	mnth_Nov	4.81
5	season_Summer	4.01
7	mnth_Aug	3.80
11	mnth_Jul	3.67
8	mnth_Dec	3.32
16	mnth_Sep	2.85
12	mnth_Jun	2.43
23	weathersit_Mist & Cloudy	2.31
10	mnth_Jan	2.14
0	yr	2.13
13	mnth_May	2.04
9	mnth_Feb	1.74
17	weekday_Monday	1.67
21	weekday_Wednesday	1.62
18	weekday_Saturday	1.58
19	weekday_Sunday	1.58
20	weekday_Tuesday	1.52
22	weathersit_Light Snow & Rain	1.27
1	holiday	1.17

`Model 7`

Dropping the Variable and Updating the Model

As you can notice some of the variable have high VIF values as well as high p-values. Such variables are insignificant and should be dropped.

As you might have noticed, the variable `hum` has a significantly high VIF (`29.63`) and a high p-value (`0.000`) as well. Hence, this variable isn't of much use and should be dropped.

In [55]:
```
# Dropping highly correlated variables and insignificant variables
X_ud_7= X_ud_6.drop('hum', axis=1)
```

In [56]:
```
# Build a third fitted model
X_train_lm7 = sm.add_constant(X_ud_7)
lr_7 = sm.OLS(y_train, X_train_lm7).fit()
print(lr_7.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.847
Model:                            OLS   Adj. R-squared:                  0.840
Method:                 Least Squares   F-statistic:                     117.1
Date:                Fri, 06 Oct 2023   Prob (F-statistic):          8.11e-182
Time:                        12:33:52   Log-Likelihood:                 517.66
No. Observations:                 511   AIC:                            -987.3
Df Residuals:                     487   BIC:                            -885.7
Df Model:                          23
Covariance Type:            nonrobust
==================================================================================================
                                     coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------
const                              0.2324      0.028      8.352      0.000       0.178       0.287
yr                                 0.2365      0.008     29.101      0.000       0.220       0.252
holiday                           -0.0855      0.027     -3.159      0.002      -0.139      -0.032
temp                               0.4103      0.044      9.340      0.000       0.324       0.497
windspeed                         -0.1675      0.026     -6.402      0.000      -0.219      -0.116
season_Summer                      0.0715      0.017      4.307      0.000       0.039       0.104
season_Winter                      0.1315      0.023      5.793      0.000       0.087       0.176
mnth_Aug                           0.0572      0.025      2.280      0.023       0.008       0.106
mnth_Dec                          -0.0538      0.026     -2.083      0.038      -0.104      -0.003
mnth_Feb                          -0.0468      0.022     -2.105      0.036      -0.090      -0.003
mnth_Jan                          -0.0826      0.022     -3.715      0.000      -0.126      -0.039
mnth_Jul                           0.0091      0.027      0.331      0.741      -0.045       0.063
mnth_Jun                           0.0285      0.022      1.271      0.204      -0.016       0.073
mnth_May                           0.0227      0.020      1.137      0.256      -0.016       0.062
mnth_Nov                          -0.0389      0.030     -1.284      0.200      -0.098       0.021
mnth_Oct                           0.0097      0.030      0.319      0.750      -0.050       0.069
mnth_Sep                           0.1093      0.024      4.614      0.000       0.063       0.156
weekday_Monday                    -0.0197      0.013     -1.484      0.138      -0.046       0.006
weekday_Saturday                   0.0026      0.013      0.201      0.841      -0.023       0.028
weekday_Sunday                    -0.0556      0.013     -4.202      0.000      -0.082      -0.030
weekday_Tuesday                   -0.0142      0.014     -1.055      0.292      -0.041       0.012
weekday_Wednesday                 -0.0036      0.013     -0.286      0.775      -0.029       0.021
weathersit_Light Snow & Rain      -0.2971      0.025    -12.126      0.000      -0.345      -0.249
weathersit_Mist & Cloudy          -0.0832      0.009     -9.621      0.000      -0.100      -0.066
==============================================================================
Omnibus:                       84.436   Durbin-Watson:                   2.053
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              232.257
Skew:                          -0.806   Prob(JB):                     3.68e-51
Kurtosis:                       5.882   Cond. No.                         22.2
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [57]:

```python
# Calculate the VIFs again for the new model
vif = pd.DataFrame()
vif['Features'] = X_ud_7.columns
vif['VIF'] = [variance_inflation_factor(X_ud_7.values, i) for i in range(X_ud_7.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[57]:

	Features	VIF
2	temp	20.69
5	season_Winter	8.08
3	windspeed	5.23
14	mnth_Oct	4.88
13	mnth_Nov	4.56
4	season_Summer	3.82
6	mnth_Aug	3.76
10	mnth_Jul	3.55
7	mnth_Dec	2.86
15	mnth_Sep	2.84
11	mnth_Jun	2.27
0	yr	2.12
12	mnth_May	2.03
16	weekday_Monday	1.63
20	weekday_Wednesday	1.59
22	weathersit_Mist & Cloudy	1.59
17	weekday_Saturday	1.56
18	weekday_Sunday	1.53
19	weekday_Tuesday	1.50
9	mnth_Jan	1.49
8	mnth_Feb	1.47
1	holiday	1.17
21	weathersit_Light Snow & Rain	1.11

`Model 8`

Dropping the Variable and Updating the Model

As you can notice some of the variable have high VIF values as well as high p-values. Such variables are insignificant and should be dropped.

As you might have noticed, the variable `season_Winter` has a significantly high VIF (`8.08`) and a high p-value (`0.000`) as well. Hence, this variable isn't of much use and should be dropped.

In [58]:
```
# Dropping highly correlated variables and insignificant variables
X_ud_8= X_ud_7.drop('season_Winter', axis=1)
```

In [59]:
```
# Build a third fitted model
X_train_lm8 = sm.add_constant(X_ud_8)
lr_8 = sm.OLS(y_train, X_train_lm8).fit()
print(lr_8.summary())
```

OLS Regression Results

Dep. Variable:	cnt	R-squared:	0.836
Model:	OLS	Adj. R-squared:	0.829
Method:	Least Squares	F-statistic:	113.3
Date:	Fri, 06 Oct 2023	Prob (F-statistic):	7.34e-176
Time:	12:33:53	Log-Likelihood:	500.64
No. Observations:	511	AIC:	-955.3
Df Residuals:	488	BIC:	-857.8
Df Model:	22		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.2375	0.029	8.270	0.000	0.181	0.294
yr	0.2386	0.008	28.464	0.000	0.222	0.255
holiday	-0.0988	0.028	-3.548	0.000	-0.153	-0.044
temp	0.4073	0.045	8.977	0.000	0.318	0.496
windspeed	-0.1833	0.027	-6.820	0.000	-0.236	-0.130
season_Summer	0.0715	0.017	4.165	0.000	0.038	0.105
mnth_Aug	0.0558	0.026	2.156	0.032	0.005	0.107
mnth_Dec	0.0362	0.021	1.697	0.090	-0.006	0.078
mnth_Feb	-0.0466	0.023	-2.031	0.043	-0.092	-0.002
mnth_Jan	-0.0842	0.023	-3.666	0.000	-0.129	-0.039
mnth_Jul	0.0082	0.028	0.289	0.772	-0.048	0.064
mnth_Jun	0.0279	0.023	1.204	0.229	-0.018	0.073
mnth_May	0.0217	0.021	1.055	0.292	-0.019	0.062
mnth_Nov	0.0924	0.021	4.453	0.000	0.052	0.133
mnth_Oct	0.1399	0.021	6.650	0.000	0.099	0.181
mnth_Sep	0.1371	0.024	5.720	0.000	0.090	0.184
weekday_Monday	-0.0154	0.014	-1.126	0.261	-0.042	0.011
weekday_Saturday	0.0050	0.013	0.378	0.706	-0.021	0.031
weekday_Sunday	-0.0543	0.014	-3.974	0.000	-0.081	-0.027
weekday_Tuesday	-0.0113	0.014	-0.811	0.418	-0.039	0.016
weekday_Wednesday	-0.0027	0.013	-0.207	0.836	-0.029	0.023
weathersit_Light Snow & Rain	-0.2968	0.025	-11.726	0.000	-0.346	-0.247
weathersit_Mist & Cloudy	-0.0823	0.009	-9.213	0.000	-0.100	-0.065

Omnibus:	79.412	Durbin-Watson:	2.098
Prob(Omnibus):	0.000	Jarque-Bera (JB):	184.683
Skew:	-0.816	Prob(JB):	7.88e-41
Kurtosis:	5.452	Cond. No.	21.8

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [60]:

```
# Calculate the VIFs again for the new model
vif = pd.DataFrame()
vif['Features'] = X_ud_8.columns
vif['VIF'] = [variance_inflation_factor(X_ud_8.values, i) for i in range(X_ud_8.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[60]:

	Features	VIF
2	temp	20.69
3	windspeed	5.18
4	season_Summer	3.82
5	mnth_Aug	3.76
9	mnth_Jul	3.55
14	mnth_Sep	2.72
10	mnth_Jun	2.27
0	yr	2.11
13	mnth_Oct	2.10
11	mnth_May	2.03
12	mnth_Nov	1.70
15	weekday_Monday	1.62
19	weekday_Wednesday	1.59
21	weathersit_Mist & Cloudy	1.59
16	weekday_Saturday	1.56
17	weekday_Sunday	1.53
6	mnth_Dec	1.52
18	weekday_Tuesday	1.50
8	mnth_Jan	1.49
7	mnth_Feb	1.47
1	holiday	1.16
20	weathersit_Light Snow & Rain	1.11

Residual Analysis of the Train data

In [61]:
```
#X_train_lm8 is the variable for last model 8
y_train_cnt = lr_8.predict(X_train_lm8)
y_train_cnt
```

Out[61]:
```
653    0.752028
576    0.751015
426    0.526682
728    0.433778
482    0.590861
         ...
578    0.843460
53     0.243625
350    0.214100
79     0.299075
520    0.655644
Length: 511, dtype: float64
```

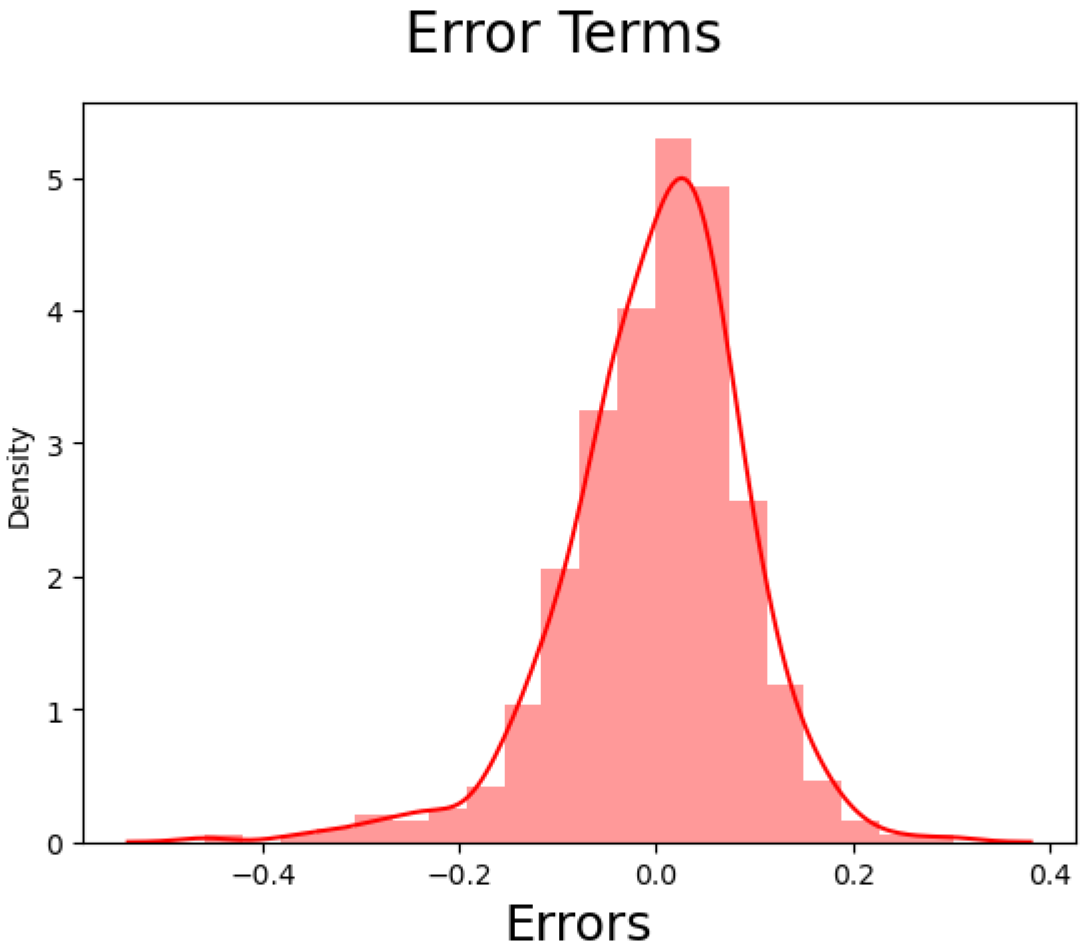
Training Data Model Evaluation:

In [62]:
```
from sklearn.metrics import r2_score
r2_score(y_train, y_train_cnt)
```

Out[62]:
```
0.836300326288283
```

In [63]:
```
# Plot the histogram of the error terms
fig = plt.figure()
sns.distplot((y_train - y_train_cnt), bins = 20,color='red')
# Plot heading
fig.suptitle('Error Terms', fontsize = 20)
# X-label
plt.xlabel('Errors', fontsize = 18)
```

Out[63]:
```
Text(0.5, 0, 'Errors')
```



Making Predictions on Test data for Final Model

In [64]:
```
#dropping the columns
X_test = X_test.drop(['mnth_Mar','atemp','season_Spring','weekday_Thrusday','workingday','hum','season_Winter'],axis=1
X_test.columns
```

Out[64]:
```
Index(['yr', 'holiday', 'temp', 'windspeed', 'season_Summer', 'mnth_Aug',
       'mnth_Dec', 'mnth_Feb', 'mnth_Jan', 'mnth_Jul', 'mnth_Jun', 'mnth_May',
       'mnth_Nov', 'mnth_Oct', 'mnth_Sep', 'weekday_Monday',
       'weekday_Saturday', 'weekday_Sunday', 'weekday_Tuesday',
       'weekday_Wednesday', 'weathersit_Light Snow & Rain',
       'weathersit_Mist & Cloudy'],
      dtype='object')
```

In [65]:
```
X_test.head()
```

Out[65]:

	yr	holiday	temp	windspeed	season_Summer	mnth_Aug	mnth_Dec	mnth_Feb	mnth_Jan	mnth_Jul	...	mnth_Nov	mnth_Oct	mı
184	0	1	0.831783	0.121812	0	0	0	0	0	1	...	0	0	
535	1	0	0.901354	0.188468	1	0	0	0	0	0	...	0	0	
299	0	0	0.511964	0.361537	0	0	0	0	0	0	...	0	1	
221	0	0	0.881625	0.366681	0	1	0	0	0	0	...	0	0	
152	0	0	0.817246	0.556403	1	0	0	0	0	0	...	0	0	

5 rows × 22 columns

Rescaling Test Data

In [66]:
```
scaler = MinMaxScaler()
X_test_pred = X_test[['yr', 'holiday', 'temp', 'windspeed', 'season_Summer', 'mnth_Aug',
       'mnth_Dec', 'mnth_Feb', 'mnth_Jan', 'mnth_Jul', 'mnth_Jun', 'mnth_May',
       'mnth_Nov', 'mnth_Oct', 'mnth_Sep', 'weekday_Monday',
       'weekday_Saturday', 'weekday_Sunday', 'weekday_Tuesday',
       'weekday_Wednesday', 'weathersit_Light Snow & Rain',
       'weathersit_Mist & Cloudy']]

X_test_pred = scaler.fit_transform(X_test_pred)

X_test_pred
```

Out[66]:
```
array([[0.        , 1.        , 0.83724073, ..., 0.        , 0.        ,
        1.        ],
       [1.        , 0.        , 0.91142308, ..., 1.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.49622086, ..., 0.        , 0.        ,
        1.        ],
       ...,
       [0.        , 0.        , 0.57372483, ..., 0.        , 0.        ,
        0.        ],
       [1.        , 0.        , 0.7453422 , ..., 1.        , 0.        ,
        1.        ],
       [0.        , 0.        , 0.30385535, ..., 0.        , 0.        ,
        0.        ]])
```

In [67]:
```
X_test.columns
```

Out[67]:
```
Index(['yr', 'holiday', 'temp', 'windspeed', 'season_Summer', 'mnth_Aug',
       'mnth_Dec', 'mnth_Feb', 'mnth_Jan', 'mnth_Jul', 'mnth_Jun', 'mnth_May',
       'mnth_Nov', 'mnth_Oct', 'mnth_Sep', 'weekday_Monday',
       'weekday_Saturday', 'weekday_Sunday', 'weekday_Tuesday',
       'weekday_Wednesday', 'weathersit_Light Snow & Rain',
       'weathersit_Mist & Cloudy'],
      dtype='object')
```

In [68]:
```
# Adding constant variable to test dataframe
X_test = sm.add_constant(X_test)
```

In [69]:
```
test_col = X_train_lm8.columns
X_test=X_test[test_col[1:]]
# Adding constant variable to test dataframe
X_test = sm.add_constant(X_test)
X_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 219 entries, 184 to 72
Data columns (total 23 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   const                        219 non-null    float64
 1   yr                           219 non-null    int64  
 2   holiday                      219 non-null    int64  
 3   temp                         219 non-null    float64
 4   windspeed                    219 non-null    float64
 5   season_Summer                219 non-null    uint8  
 6   mnth_Aug                     219 non-null    uint8  
 7   mnth_Dec                     219 non-null    uint8  
 8   mnth_Feb                     219 non-null    uint8  
 9   mnth_Jan                     219 non-null    uint8  
 10  mnth_Jul                     219 non-null    uint8  
 11  mnth_Jun                     219 non-null    uint8  
 12  mnth_May                     219 non-null    uint8  
 13  mnth_Nov                     219 non-null    uint8  
 14  mnth_Oct                     219 non-null    uint8  
 15  mnth_Sep                     219 non-null    uint8  
 16  weekday_Monday               219 non-null    uint8  
 17  weekday_Saturday             219 non-null    uint8  
 18  weekday_Sunday               219 non-null    uint8  
 19  weekday_Tuesday              219 non-null    uint8  
 20  weekday_Wednesday            219 non-null    uint8  
 21  weathersit_Light Snow & Rain 219 non-null    uint8  
 22  weathersit_Mist & Cloudy     219 non-null    uint8  
dtypes: float64(3), int64(2), uint8(18)
memory usage: 14.1 KB
```

Making predictions on Test data for final model

In [70]:

```
#using lr_8 final model
y_pred = lr_8.predict(X_test)
y_pred
```

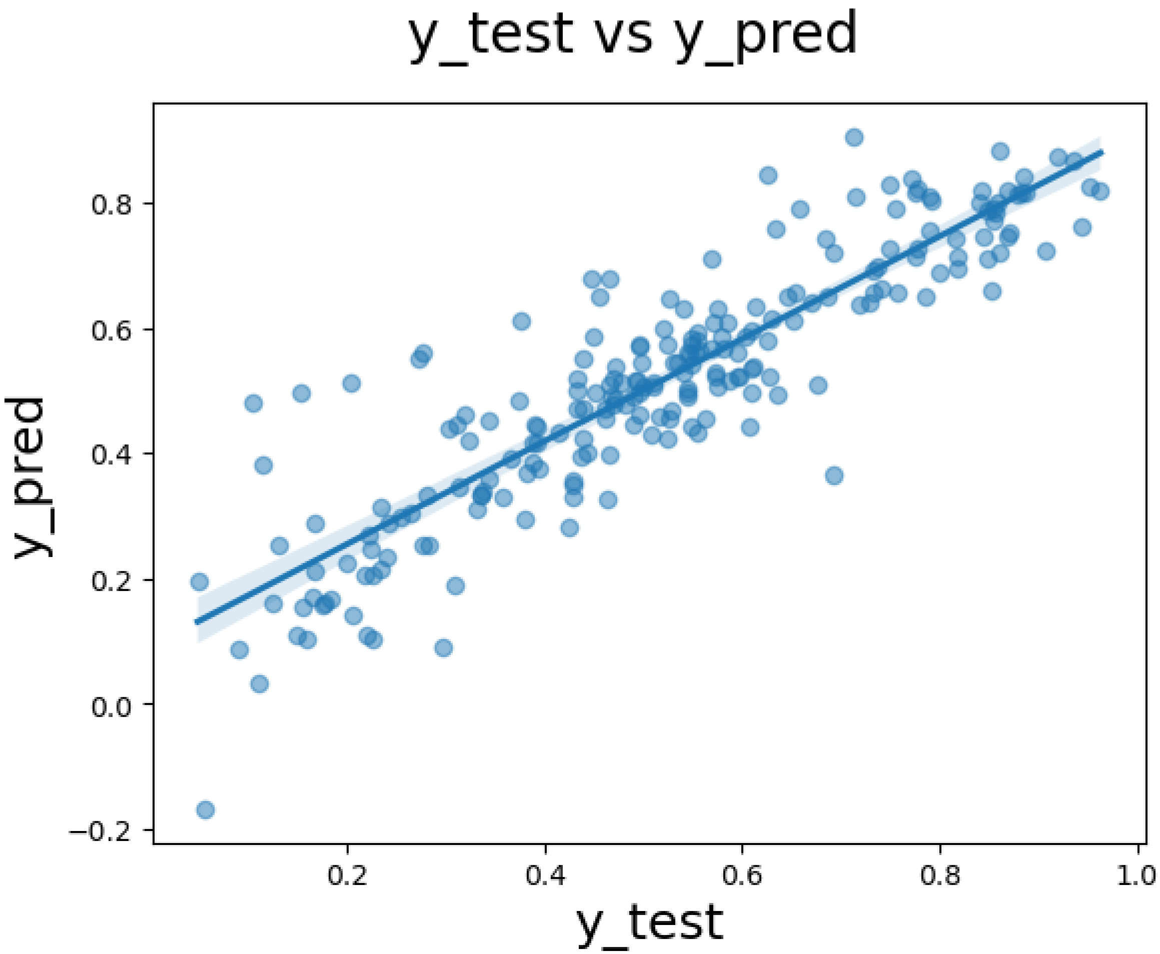
Out[70]:
```
184    0.365727
535    0.905322
299    0.437432
221    0.582496
152    0.567732
         ...
400    0.339895
702    0.655132
127    0.494818
640    0.818068
72     0.313859
Length: 219, dtype: float64
```

Testing Data Model Evaluation:

In [71]:
```
r2_score(y_test, y_pred)
```

Out[71]:
```
0.791596609342758
```

In [72]:
```
# Plotting y_test and y_pred to understand the spread
fig = plt.figure()
sns.regplot(x=y_test, y=y_pred, scatter_kws={'alpha':0.5})
fig.suptitle('y_test vs y_pred', fontsize=20)
plt.xlabel('y_test', fontsize=18)
plt.ylabel('y_pred', fontsize=18)
plt.show()
```



Report for Bike-Sharing System (BoomBikes) Project

Key Questions

The key questions to address in this project are:

1. Identify the significant variables affecting bike demand.
2. Evaluate how well these variables explain the variation in bike demand.

Data Exploration

The project began with data exploration and preprocessing. Here are the key insights:

- The dataset contains 730 entries and 16 columns, including features like season, year, month, holiday, weather conditions, temperature, and bike demand (cnt).
- Data types include integers, floats, and one object column ('dteday').
- No missing values were found in the dataset.

Data Visualization

Data visualization was used to gain insights into the relationships between variables. Some key visualizations include:

- Univariate analysis: Histograms and scatterplots to examine the distribution and relationships of variables.
- Bivariate analysis: Box plots and scatterplots to analyze how categorical variables (e.g., season, month, weekday) and numerical variables (e.g., temperature) affect bike demand.
- Multivariate analysis: Pairplots and bar plots to explore interactions between multiple variables and their impact on bike demand.

Data Preprocessing

Data preprocessing steps included:

1. Encoding categorical variables like season, month, weekday, and weathersit.
2. Scaling numerical variables to ensure they are on the same scale.
3. Creating dummy variables for categorical features to prepare the data for modeling.

Model Building

Multiple linear regression models were built using the statsmodels library. Feature selection was performed iteratively by dropping variables with high VIF values and insignificant p-values. The final model, Model 8, included the following significant variables:

- Year (yr)
- Holiday
- Temperature (temp)
- Windspeed
- Month (except for January and February)
- Weekday (except for Monday)
- Weather conditions (Mist & Cloudy and Light Snow & Rain)

`Model Evaluation`

The final model, Model 8, was evaluated for its statistical significance and predictive performance. Key model evaluation metrics include:

- **R-squared value:** 0.843 (indicating that 84.3% of the variance in bike demand is explained by the model).
- **AIC and BIC values:** These information criteria were used to assess the goodness of fit, with lower values indicating better fit.
- **Coefficient significance:** All selected variables were statistically significant with p-values less than 0.05.
- **Multicollinearity:** VIF values were used to check for multicollinearity, and all VIF values were below 5.

`Conclusion`

The final multiple linear regression model provides valuable insights for BoomBikes to predict bike demand accurately.

Key factors influencing bike demand:

`1.year`

`2.holiday`

`3.temperature`

`4.windspeed`

`5.month`

`6.weekday`

`7.weather conditions`

This model can help BoomBikes optimize bike availability, marketing strategies, and pricing to meet customer demand effectively and maximize profits post-pandemic.

The model's `R-squared value of 0.843 for training data` and `R-squared value of 0.76 for testing data` indicates that it explains a significant portion of the variance in bike demand. Further refinement and testing on real-world data can enhance its predictive accuracy. BoomBikes can use this model as a foundation to make data-driven decisions and gain a competitive edge in the bike-sharing market.

Thank You

In [ ]: