Assignment

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.

They have contracted a consulting company to understand the factors on which the demand for these shared bikes depends. Specifically, they want to understand the factors affecting the demand for these shared bikes in the American market.

Business Goal:

You are required to model the 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.

Objective

This assignment is a programming assignment wherein I've have to build a multiple linear regression model for the prediction of demand for shared bikes.

Problem Statements

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

2 - How well those variables describe the bike demands

Importing necessary liabraries for analyzing the Dataset

In [1]:
```
# importing warnings

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# importing numpy and pandas

import numpy as np
import pandas as pd
```

In [3]:
```
# importing libraries for data visualization

import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
# importing machine learning libraries
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFE
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
```

Data Understanding and Exploration

In [5]:
```
# importing dataset

data = pd.read_csv('Bike Sharing Dataset.csv')
```

In [6]:
```
data.head() #To display all first 5 rows
```

Out[6]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	01-01-2018	1	0	1	0	1	1	2	14.110847	18.18125	80.5833	10.749882	331	654	985
1	2	02-01-2018	1	0	1	0	2	1	2	14.902598	17.68695	69.6087	16.652113	131	670	801
2	3	03-01-2018	1	0	1	0	3	1	1	8.050924	9.47025	43.7273	16.636703	120	1229	1349
3	4	04-01-2018	1	0	1	0	4	1	1	8.200000	10.60610	59.0435	10.739832	108	1454	1562
4	5	05-01-2018	1	0	1	0	5	1	1	9.305237	11.46350	43.6957	12.522300	82	1518	1600

In [7]:
```
data.tail() #To display all last 5 rows
```

Out[7]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cr
725	726	27-12-2019	1	1	12	0	5	1	2	10.420847	11.33210	65.2917	23.458911	247	1867	211
726	727	28-12-2019	1	1	12	0	6	0	2	10.386653	12.75230	59.0000	10.416557	644	2451	309
727	728	29-12-2019	1	1	12	0	0	0	2	10.386653	12.12000	75.2917	8.333661	159	1182	134
728	729	30-12-2019	1	1	12	0	1	1	1	10.489153	11.58500	48.3333	23.500518	364	1432	179
729	730	31-12-2019	1	1	12	0	2	1	2	8.849153	11.17435	57.7500	10.374682	439	2290	272

In [8]:
```
# Checking the shape

data.shape
```

Out[8]:
```
(730, 16)
```

There are total of 730 rows and 16 columns in the raw dataset

In [9]:
```
# Checking the name of columns

data.columns
```

Out[9]:
```
Index(['instant', 'dteday', 'season', 'yr', 'mnth', 'holiday', 'weekday',
       'workingday', 'weathersit', 'temp', 'atemp', 'hum', 'windspeed',
       'casual', 'registered', 'cnt'],
      dtype='object')
```

In [10]:
```
# Checking the numerical columns and data distribution statistics

data.describe()
```

Out[10]:

	instant	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspe
count	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000
mean	365.500000	2.498630	0.500000	6.526027	0.028767	2.995890	0.690411	1.394521	20.319259	23.726322	62.765175	12.763
std	210.877136	1.110184	0.500343	3.450215	0.167266	2.000339	0.462641	0.544807	7.506729	8.150308	14.237589	5.195
min	1.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	2.424346	3.953480	0.000000	1.500
25%	183.250000	2.000000	0.000000	4.000000	0.000000	1.000000	0.000000	1.000000	13.811885	16.889713	52.000000	9.041
50%	365.500000	3.000000	0.500000	7.000000	0.000000	3.000000	1.000000	1.000000	20.465826	24.368225	62.625000	12.125
75%	547.750000	3.000000	1.000000	10.000000	0.000000	5.000000	1.000000	2.000000	26.880615	30.445775	72.989575	15.625
max	730.000000	4.000000	1.000000	12.000000	1.000000	6.000000	1.000000	3.000000	35.328347	42.044800	97.250000	34.000

In [11]:
```
# Checking for Null values and Datatypes

data.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 [12]:
```
# Counting the sum of Null values in the dataset

data.isnull().sum()
```

Out[12]:
```
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
```

We can see that there are no null values in the dataset

In [13]:
```
# Checking for duplicates

data.drop_duplicates(subset = None, inplace=True)

# Subset is used to perform operations on any specific or filtered variables.
# Since, we are checking for duplicates in the entire dataset, so I've used subset = None( which means no subset)

# Inplace is used to perform operations on same dataset. So, I've used inplace=True
```

In [14]:
```
data.shape
```

Out[14]:
```
(730, 16)
```

We can see that after droping the duplicate values from entire dataset, the rows and columns does not changed. So, we can say that there are no duplicates in the dataset.

In [15]:
```
# Checking the size of Data

data.size
```

Out[15]:
```
11680
```

In [16]:
```
# Checking the axes of Data

data.axes
```

Out[16]:
```
[Int64Index([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,
             ...
             720, 721, 722, 723, 724, 725, 726, 727, 728, 729],
            dtype='int64', length=730),
 Index(['instant', 'dteday', 'season', 'yr', 'mnth', 'holiday', 'weekday',
        'workingday', 'weathersit', 'temp', 'atemp', 'hum', 'windspeed',
        'casual', 'registered', 'cnt'],
       dtype='object')]
```

In [17]:
```
# Checking the dimensions of Data

data.ndim
```

Out[17]:
```
2
```

In [18]:
```
# Checking the values of Data

data.values
```

Out[18]:
```
array([[1, '01-01-2018', 1, ..., 331, 654, 985],
       [2, '02-01-2018', 1, ..., 131, 670, 801],
       [3, '03-01-2018', 1, ..., 120, 1229, 1349],
       ...,
       [728, '29-12-2019', 1, ..., 159, 1182, 1341],
       [729, '30-12-2019', 1, ..., 364, 1432, 1796],
       [730, '31-12-2019', 1, ..., 439, 2290, 2729]], dtype=object)
```

Checking for Continuous and Categorical Data

In [19]:
```
# Checking for Continuous Data

data.select_dtypes(include=[np.number])
```

Out[19]:

	instant	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	1	0	1	0	1	1	2	14.110847	18.18125	80.5833	10.749882	331	654	985
1	2	1	0	1	0	2	1	2	14.902598	17.68695	69.6087	16.652113	131	670	801
2	3	1	0	1	0	3	1	1	8.050924	9.47025	43.7273	16.636703	120	1229	1349
3	4	1	0	1	0	4	1	1	8.200000	10.60610	59.0435	10.739832	108	1454	1562
4	5	1	0	1	0	5	1	1	9.305237	11.46350	43.6957	12.522300	82	1518	1600
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
725	726	1	1	12	0	5	1	2	10.420847	11.33210	65.2917	23.458911	247	1867	2114
726	727	1	1	12	0	6	0	2	10.386653	12.75230	59.0000	10.416557	644	2451	3095
727	728	1	1	12	0	0	0	2	10.386653	12.12000	75.2917	8.333661	159	1182	1341
728	729	1	1	12	0	1	1	1	10.489153	11.58500	48.3333	23.500518	364	1432	1796
729	730	1	1	12	0	2	1	2	8.849153	11.17435	57.7500	10.374682	439	2290	2729

In [20]:
```
# Checking for Categoical Data

data.select_dtypes(exclude=[np.number])
```

Out[20]:

	dteday
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

730 rows × 1 columns

Data Preparation

Casual and Registered variables will describe that the target variable which is cnt in a very trivial way . So target = casual + registered, which leads to data leakage. . clearlly saying that sum of the both casual + registered is the target column/variable . Also to avoid multicollinearity we delete the columns 'casual' and 'registered'

In [21]:
```
#droping the columns
data.drop(['casual'], axis = 1, inplace = True)
data.drop(['registered'], axis = 1, inplace = True)
```

Instant is just a row instance identifier or series. dteday is removed as we have some of date features like mnth and year and weekday already in other columns . And also for this analysis we will not consider day to day trend in demand for bikes. So, we can delete Instant and dteday columns from the dataframe

In [22]:
```
#droping the columns
data.drop(['instant','dteday'],axis = 1,inplace = True)
```

In [23]:
```
data.shape
```

Out[23]:
```
(730, 12)
```

In [24]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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: 74.1 KB
```

Converting some numeric data to Categorical data

In [25]:
```
#season
data.season = data.season.map({1: 'Spring',2:'Summer',3:'Fall',4:'Winter'})
data['season'].astype('category').value_counts()
```

Out[25]:
```
Fall      188
Summer    184
Spring    180
Winter    178
Name: season, dtype: int64
```

In [26]:
```
#mnth
data.mnth.replace({1: 'jan',2: 'feb',3: 'mar',4: 'apr',5: 'may',6: 'jun',
                   7: 'jul',8: 'aug',9: 'sept',10: 'oct',11: 'nov',12: 'dec'},inplace=True)
```

In [27]:
```
#Checking whether the conversion is done properly or not
data['mnth'].astype('category').value_counts()
```

Out[27]:
```
oct     62
may     62
mar     62
jul     62
jan     62
dec     62
aug     62
sept    60
nov     60
jun     60
apr     60
feb     56
Name: mnth, dtype: int64
```

In [28]:
```
#weathersit
data.weathersit.replace({1:'good',2:'moderate',3:'bad',4:'severe'},inplace = True)
```

In [29]:
```
#Checking whether the conversion is done properly or not
data['weathersit'].astype('category').value_counts()
```

Out[29]:
```
good        463
moderate    246
bad          21
Name: weathersit, dtype: int64
```

In [30]:
```
#weekday
data.weekday.replace({0: 'sun',1: 'mon',2: 'tue',3: 'wed',4: 'thu',5: 'fri',6: 'sat'},inplace=True)
```

In [31]:
```
#Checking whether the conversion is done properly or not
data['weekday'].astype('category').value_counts()
```

Out[31]:
```
tue    105
mon    105
wed    104
thu    104
sun    104
sat    104
fri    104
Name: weekday, dtype: int64
```

In [32]:
```
# Check the dataframe again
data.head()
```

Out[32]:

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

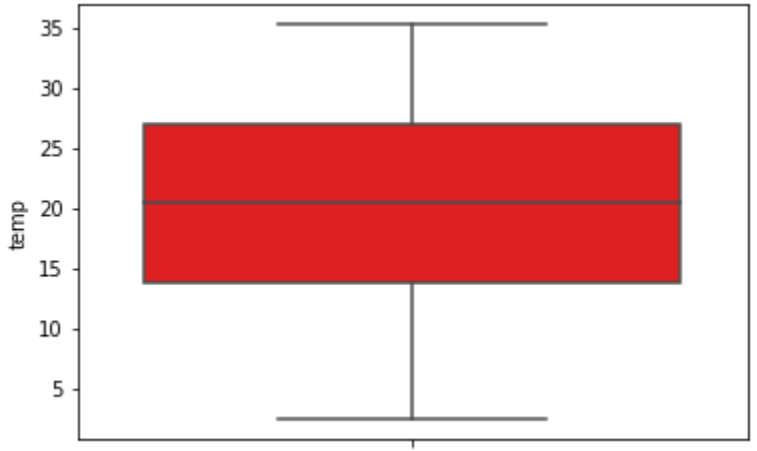
Data Visualization

Finding out the outliers in the continuous and Categorical data

In [33]:
```
# Below columns are all numberical columns in dataset.
cols = ['season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday', 'weathersit', 'temp','atemp', 'hum', 'windspeed', 'cnt
# We are just checking outliers on some specific columns that may have some values that unexpected such as:
#     temp : temperature in Celsius
#   - atemp: feeling temperature in Celsius
#   - hum: humidity
#   - windspeed: wind speed
```

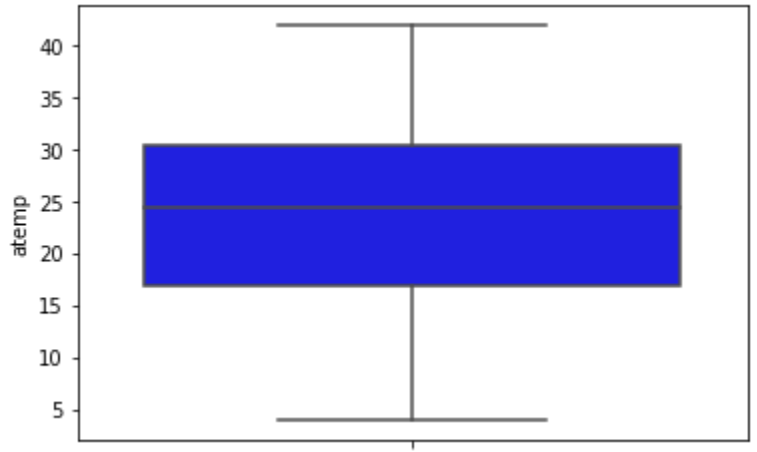
In [34]:
```
sns.boxplot(y='temp', data=data, color='red')
```

Out[34]: <AxesSubplot:ylabel='temp'>



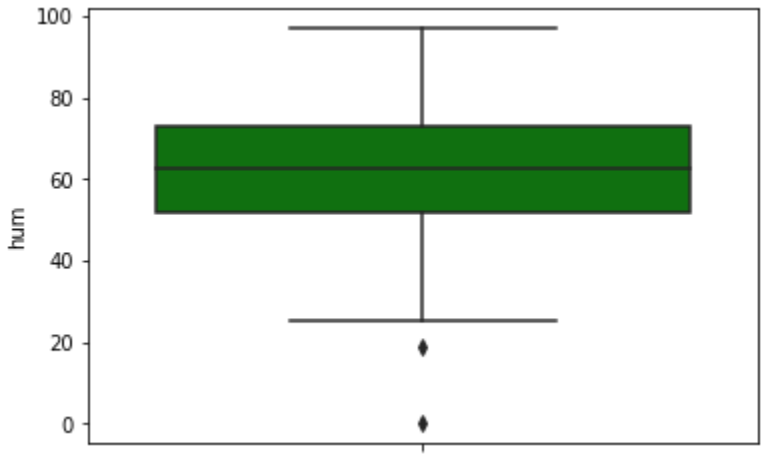
In [35]:
```
sns.boxplot(y='atemp', data=data, color='blue')
```

Out[35]: <AxesSubplot:ylabel='atemp'>



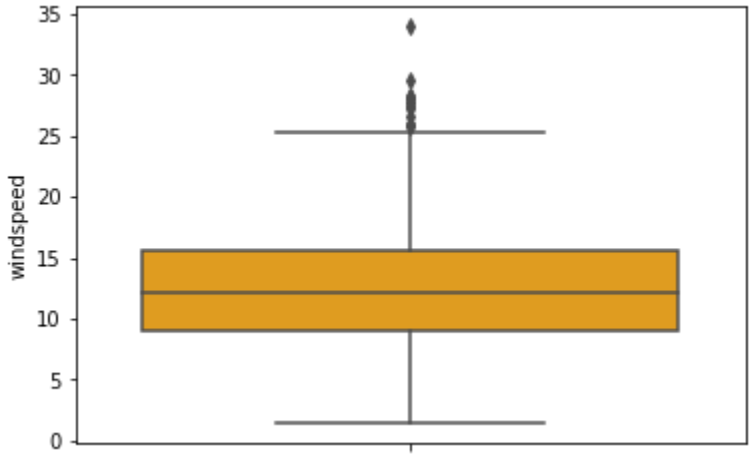
In [36]:
```
sns.boxplot(y='hum', data=data, color='green')
```

Out[36]: <AxesSubplot:ylabel='hum'>



In [37]:
```
sns.boxplot(y='windspeed', data=data, color='orange')
```

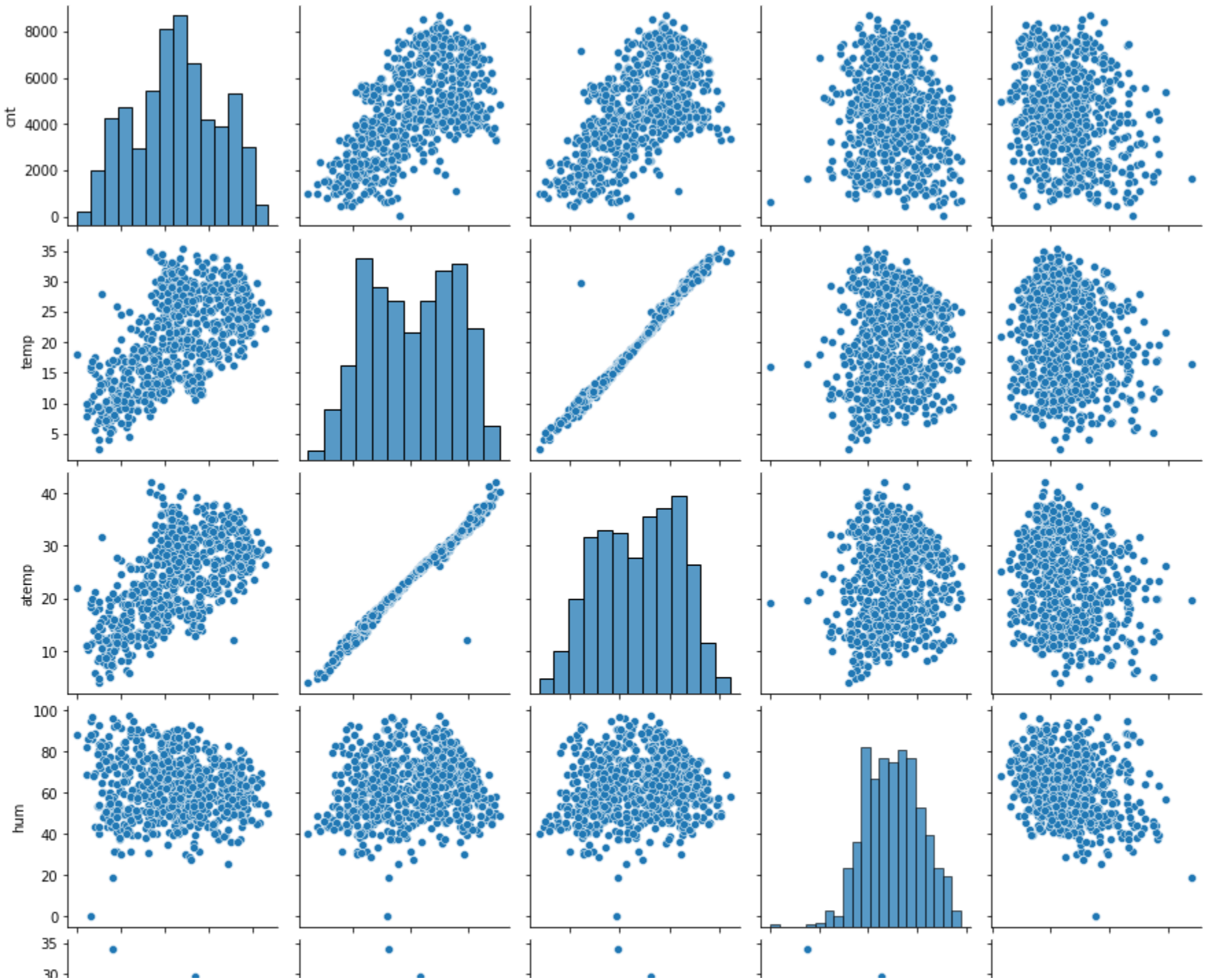
Out[37]: <AxesSubplot:ylabel='windspeed'>

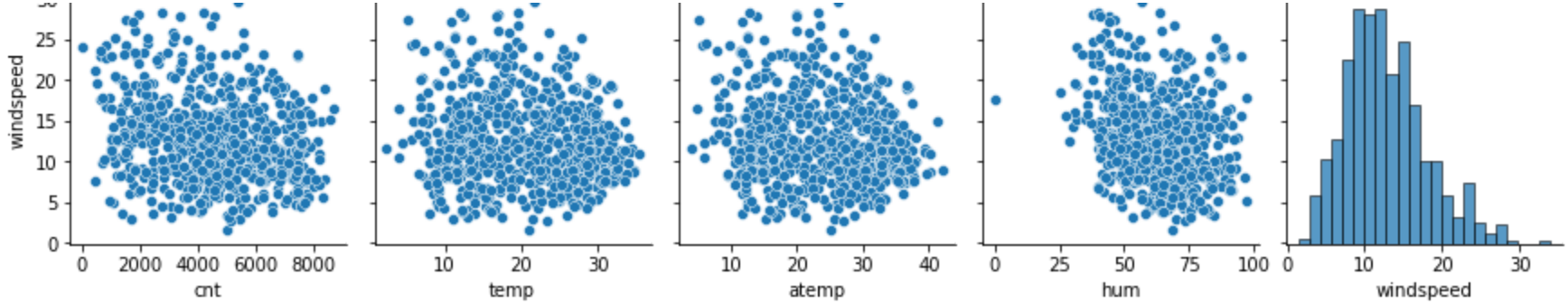


By looking at above plots we can say that there are no outliers.

Visualising Numeric Variables

In [38]:
```
sns.pairplot(data=data,vars=['cnt', 'temp', 'atemp', 'hum','windspeed'])
plt.show()
```

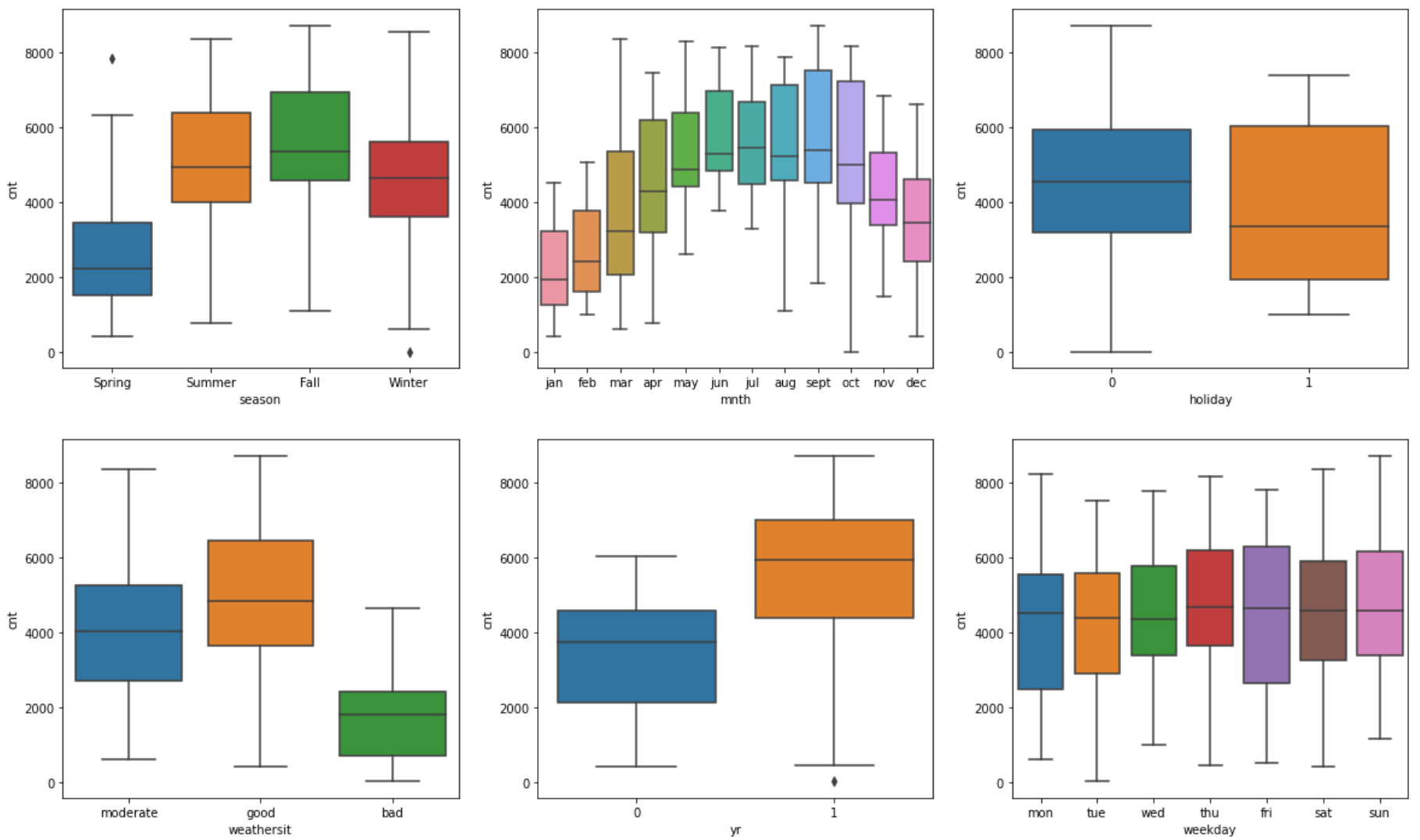




Made a pairplot to visualize all the numeric variables and found that temp and atemp has the highest corelation with the target variable cnt . temp and atemp are highly co-related with each other

Visualising Categorical Variables

In [39]:
```
plt.figure(figsize=(20, 12))
plt.subplot(2,3,1)
sns.boxplot(x = 'season', y = 'cnt', data = data)
plt.subplot(2,3,2)
sns.boxplot(x = 'mnth', y = 'cnt', data =data)
plt.subplot(2,3,3)
sns.boxplot(x = 'holiday', y = 'cnt', data = data)
plt.subplot(2,3,4)
sns.boxplot(x = 'weathersit', y = 'cnt', data = data)
plt.subplot(2,3,5)
sns.boxplot(x = 'yr', y = 'cnt', data = data)
plt.subplot(2,3,6)
sns.boxplot(x = 'weekday', y = 'cnt', data = data)
plt.show()
```



from the above boxplot diagram, we have observed that bike sharing count is very less in spring season compared to all other seasons which are higher than 4000 cnt. The Bike sharing count is highest on the month of July and is being increased in summer months and customers are less likely to use bikes on holidays to be more precise, count of shared bikes are highest on thursday and slightly low on Friday. Weathersit shows that customers shared bikes on good weather. The count of shared bikes are higher in 2019 in comparison of 2018.

Creating Dummy variables for all categorical variables

In [40]:
```
dummy = data[['season','mnth','weekday','weathersit']]
dummy = pd.get_dummies(dummy,drop_first=True)
```

Adding Dummy variables to the raw dataset

In [41]:
```
data = pd.concat([dummy,data],axis = 1)
```

In [42]:
```
# Checking if the dummy variables are added properly or not

data.head()
```

Out[42]:

	season_Spring	season_Summer	season_Winter	mnth_aug	mnth_dec	mnth_feb	mnth_jan	mnth_jul	mnth_jun	mnth_mar	...	mnth	holiday	we
0	1	0	0	0	0	0	1	0	0	0	...	jan	0	
1	1	0	0	0	0	0	1	0	0	0	...	jan	0	
2	1	0	0	0	0	0	1	0	0	0	...	jan	0	
3	1	0	0	0	0	0	1	0	0	0	...	jan	0	
4	1	0	0	0	0	0	1	0	0	0	...	jan	0	

5 rows × 34 columns

In [43]:
```
# Checking the shape of dataset

data.shape
```

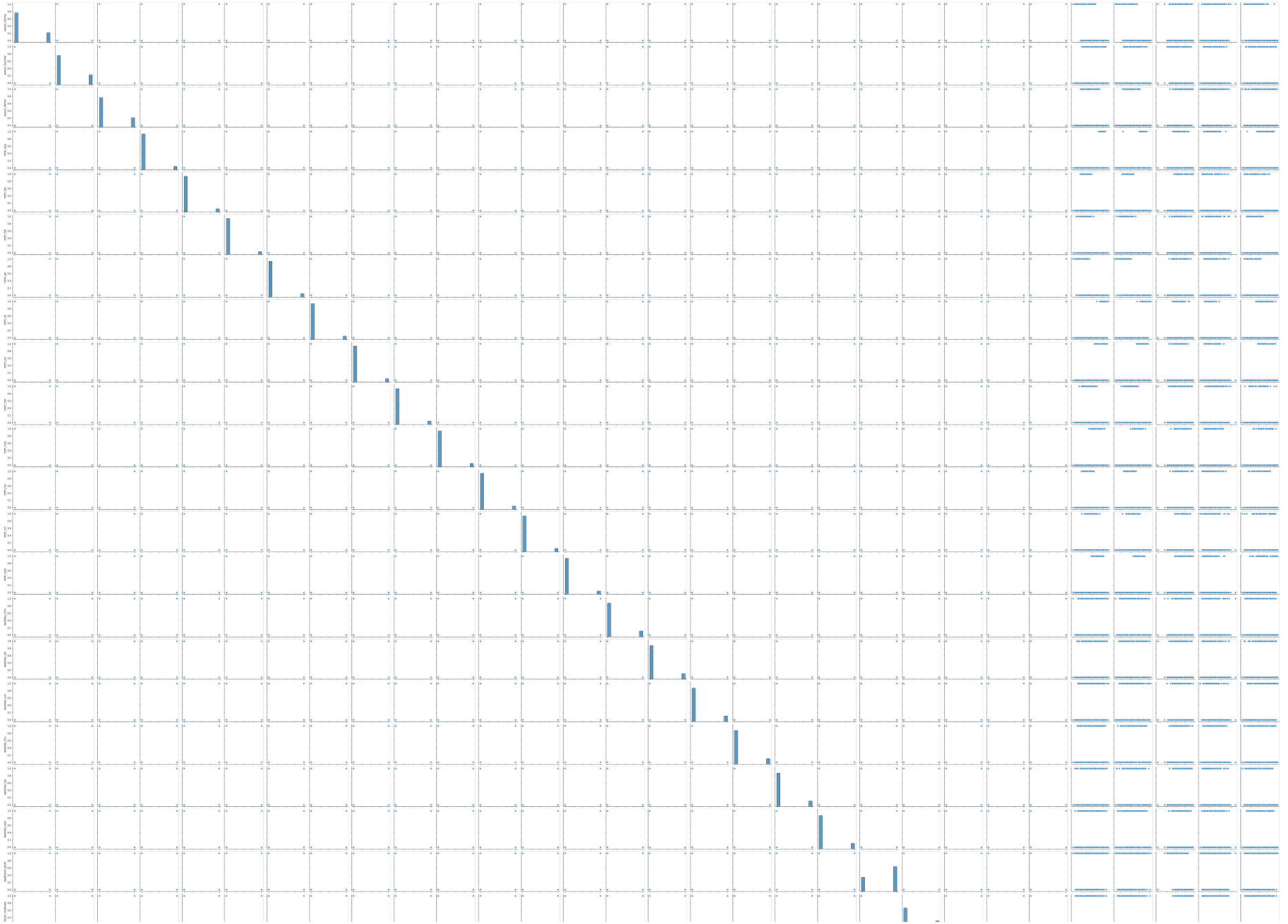
Out[43]:
```
(730, 34)
```

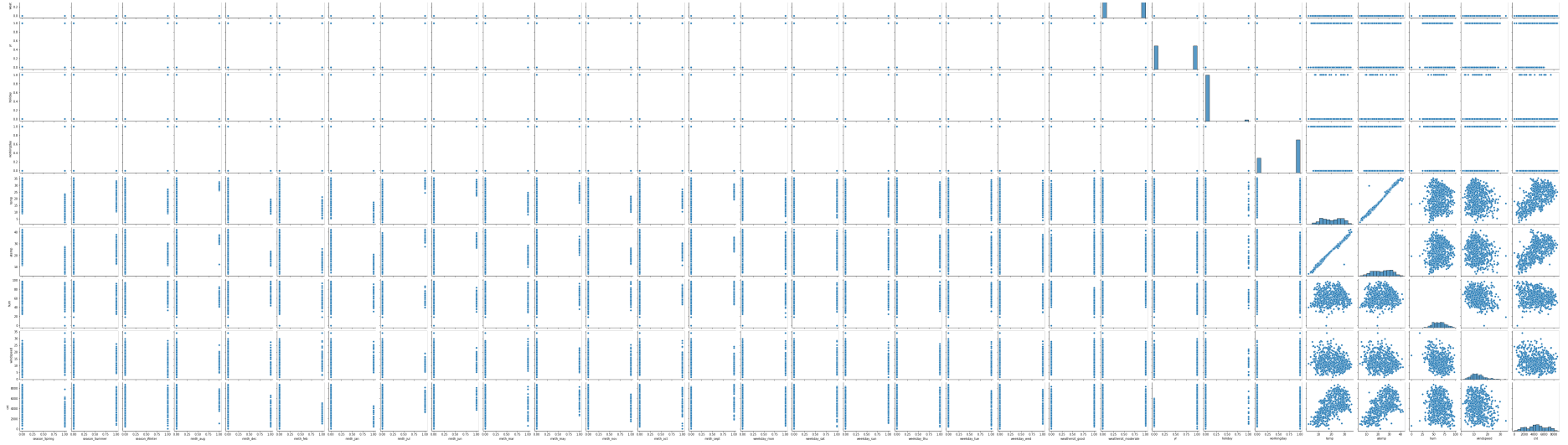
Using univariate,Bivariate Analysis for the numerical and categorical variables

Visualising the Dataframe to Find the Correlation between the Numerical Variable

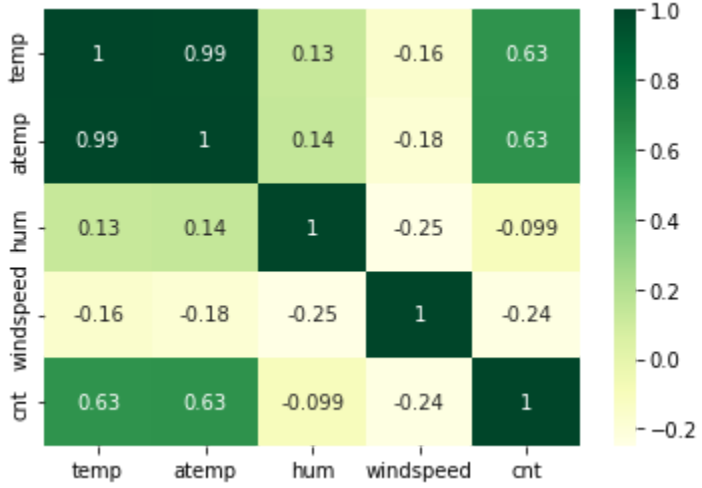
In [44]:
```
plt.figure(figsize=(20,15))
sns.pairplot(data)
plt.show()
```

```
<Figure size 1440x1080 with 0 Axes>
```



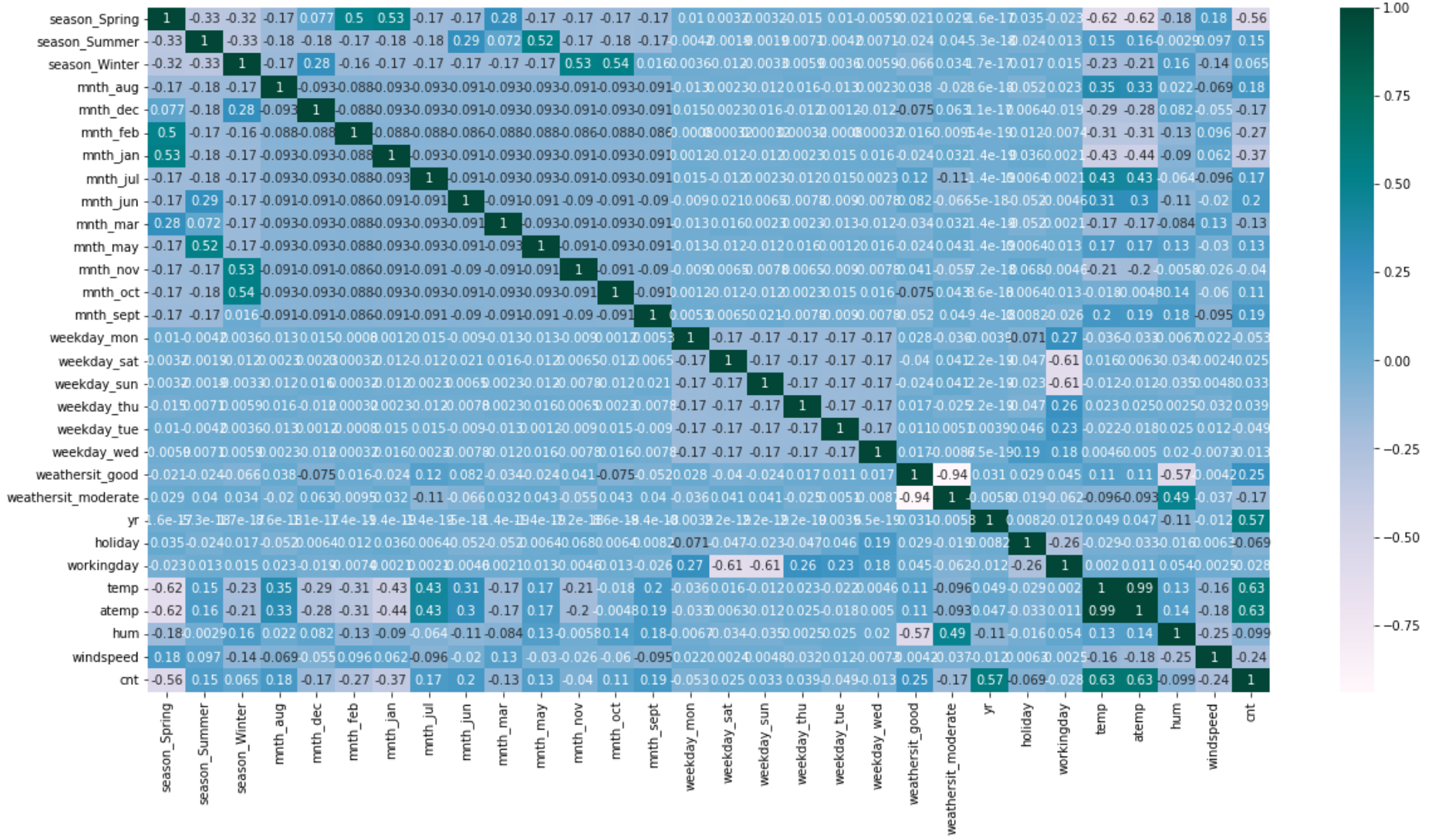


In [45]:
```
# Checking continuous variables Correlation relationship with each other
sns.heatmap(data[['temp','atemp','hum','windspeed','cnt']].corr(), cmap='YlGn', annot = True)
plt.show()
```



Here we see that temp and atemp has correlation 0.99,that means almost 1 (highly correlated) and atemp seems to be derived from temp so atemp field can be dropped here

In [46]:
```
plt.figure(figsize = (20, 10))
cor=data.corr()
sns.heatmap(cor, cmap="PuBuGn", annot = True)
plt.show()
```

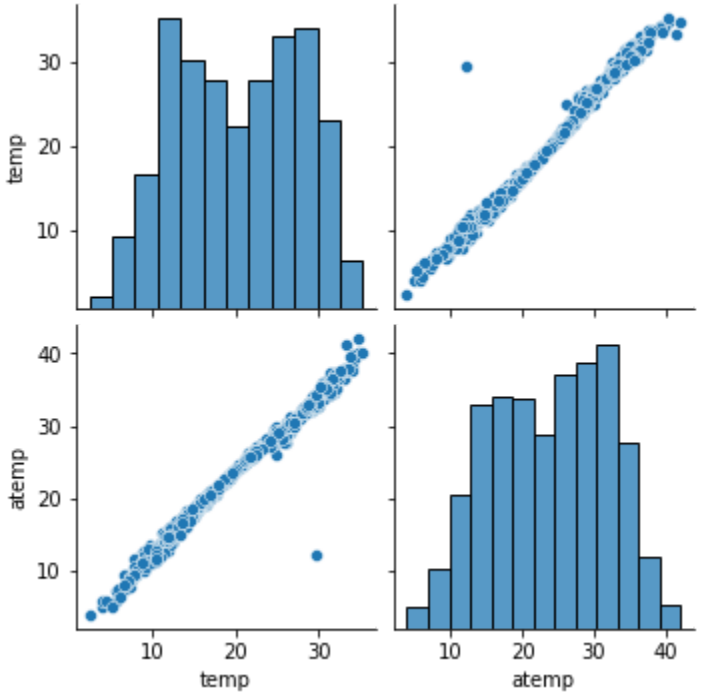


From the correlation map, season_winter and mnth_oct seems to be highly correlated with 0.54 and only should variable can be considered for the

model. . However let us elminate it based on the Variance Inflation Factor later during the model building. . We also see Target variable has a linear relationship with some of the indeptendent variables. so that we can drop the atemp variable .temp variable also highlly corelated with target variable with 0.63

Bivariate Analysis

In [47]:
```
sns.pairplot(data[['temp','atemp']])
plt.show()
```



Predictor columns temp and atemp are very much correlated to each other, . so we can drop any one of them, here we are dropping atemp variable

In [ ]:
```
#Droping the atemp variable:
data.drop(['atemp'],axis=1,inplace=True)
```

In [50]:
```
# check the dataframe again
data.head()
```

Out[50]:

	season_Spring	season_Summer	season_Winter	mnth_aug	mnth_dec	mnth_feb	mnth_jan	mnth_jul	mnth_jun	mnth_mar	...	yr	mnth	holiday
0	1	0	0	0	0	0	1	0	0	0	...	0	jan	0
1	1	0	0	0	0	0	1	0	0	0	...	0	jan	0
2	1	0	0	0	0	0	1	0	0	0	...	0	jan	0
3	1	0	0	0	0	0	1	0	0	0	...	0	jan	0
4	1	0	0	0	0	0	1	0	0	0	...	0	jan	0

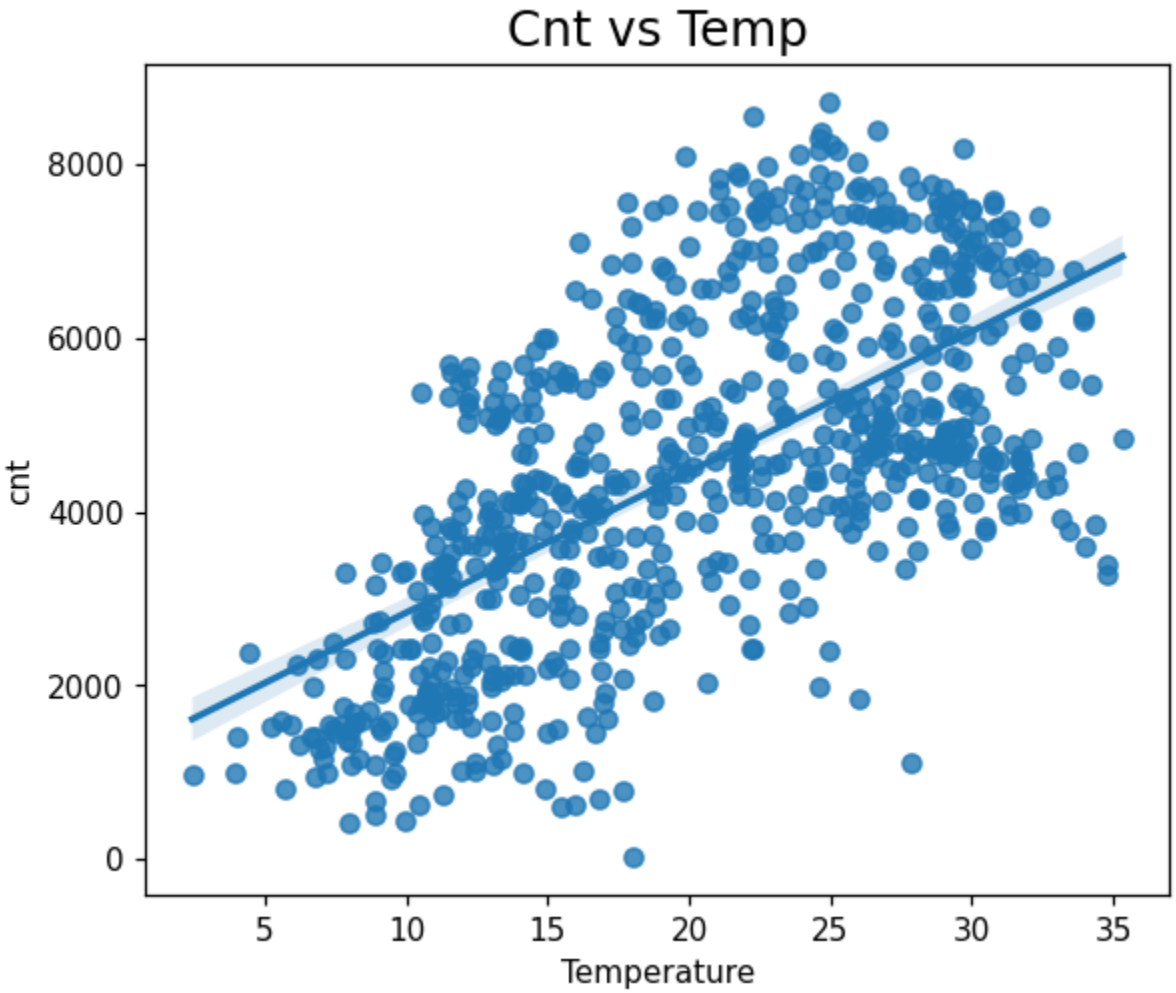
5 rows × 33 columns

In [51]:
```
#Checking the dataframe again
data.shape
```

Out[51]:
```
(730, 33)
```

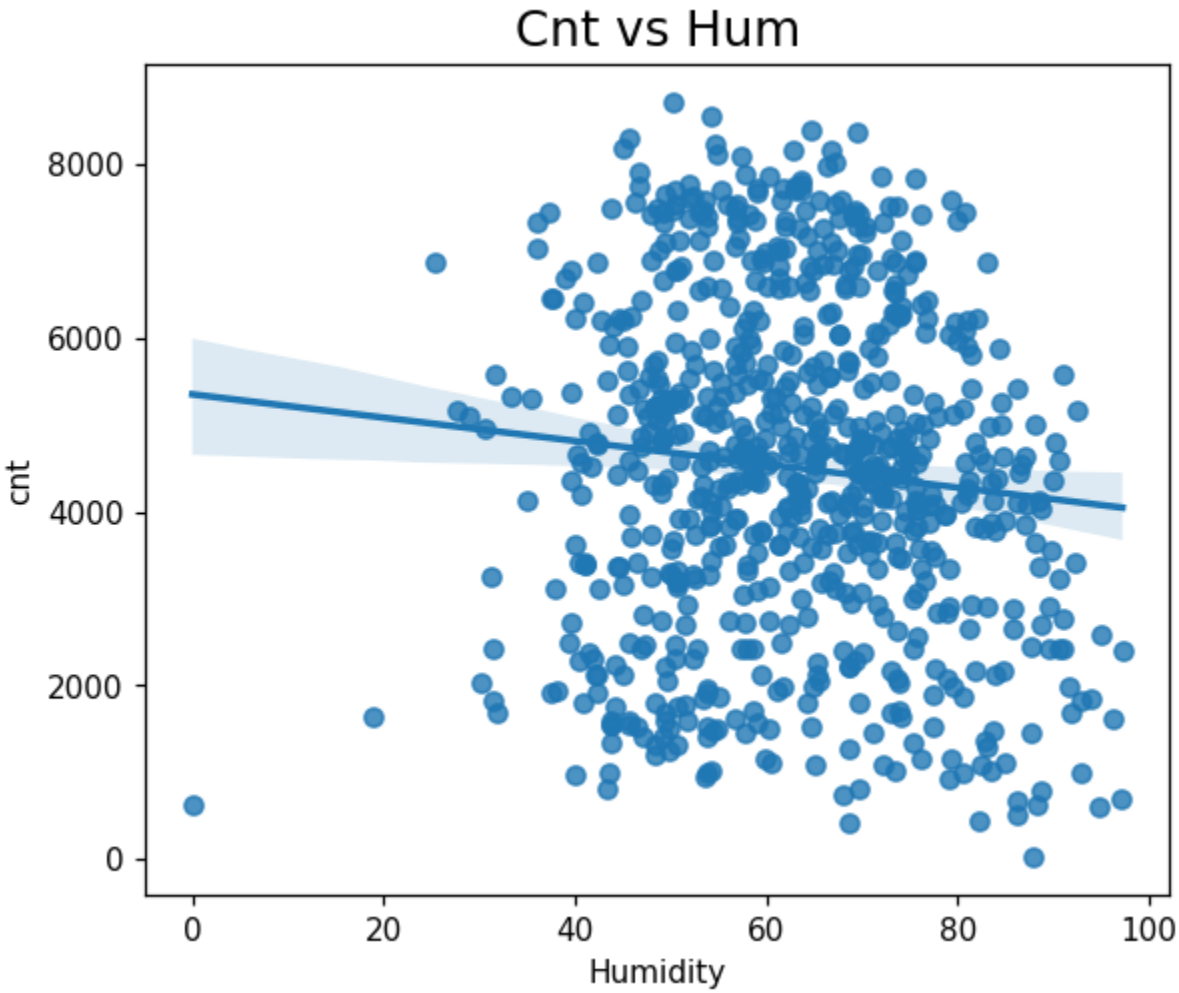
Univariate Analysis

In [52]:
```
# univariate Analysis for Cnt vs Temp
plt.figure(figsize=(6,5),dpi=110)
plt.title("Cnt vs Temp",fontsize=16)
sns.regplot(data=data,y="cnt",x="temp")
plt.xlabel("Temperature")
plt.show()
```



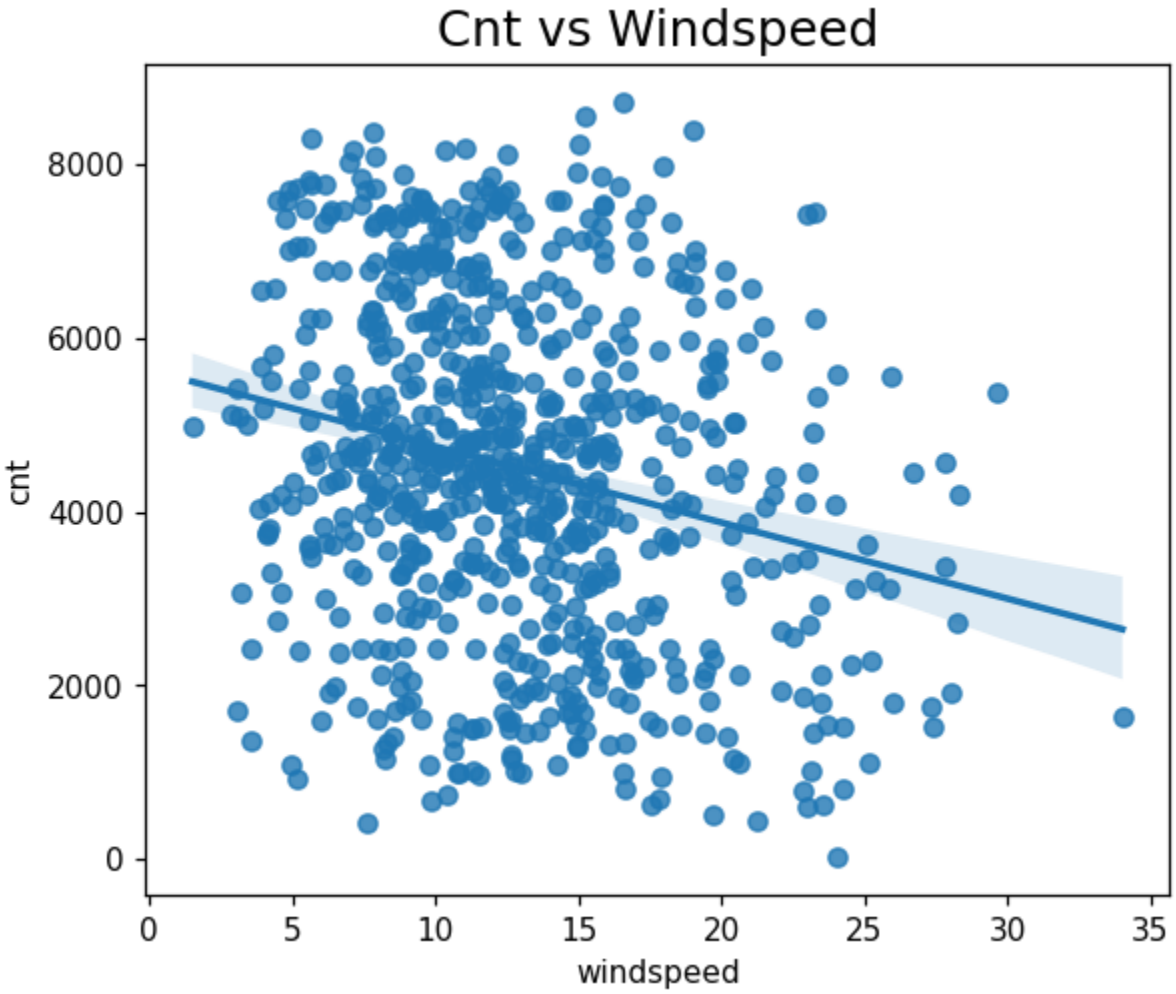
Demand for bikes is positively correlated to temp. . We can see that cnt is linearly increasing with temp indicating linear relation.

In [53]:
```
# univariate Analysis for Cnt vs Hum
plt.figure(figsize=(6,5),dpi=110)
plt.title("Cnt vs Hum",fontsize=16)
sns.regplot(data=data,y="cnt",x="hum")
plt.xlabel("Humidity")
plt.show()
```



Hum is values are more scattered around. . Although we can see cnt decreasing with increase in humidity.

In [54]:
```
# univariate Analysis for Cnt vs Windspeed
plt.figure(figsize=(6,5),dpi=110)
plt.title("Cnt vs Windspeed",fontsize=16)
sns.regplot(data=data,y="cnt",x="windspeed")
plt.show()
```



Windspeed values are more scattered around. . Although we can see cnt decreasing with increase in windspeed.

Model building:

Divide the data in to train and test . perform scaling . divide the data into X and Y . perform linear regression . use the mixed appreach(RFE and manual)

In [56]:
```
data.shape
```

Out[56]:
```
(730, 33)
```

In [58]:
```
#using train_test_split we are dividing the daraset inti train and test
np.random.seed(0)
data_train, data_test = train_test_split(data, train_size=0.7, test_size=0.3, random_state=100)
```

In [70]:
```
#checking shape of train and test dataframes
print("Train dataframe : ",data_train.shape)
print("Test dataframe : ",data_test.shape)
```

```
Train dataframe :  (510, 33)
Test dataframe :  (219, 33)
```

In [71]:
```
data_train.head()
```

Out[71]:

	season_Spring	season_Summer	season_Winter	mnth_aug	mnth_dec	mnth_feb	mnth_jan	mnth_jul	mnth_jun	mnth_mar	...	yr	mnth	holida
653	0	0	1	0	0	0	0	0	0	0	...	1	oct	
576	0	0	0	0	0	0	0	1	0	0	...	1	jul	
426	1	0	0	0	0	0	0	0	0	1	...	1	mar	
728	1	0	0	0	1	0	0	0	0	0	...	1	dec	
482	0	1	0	0	0	0	0	0	0	0	...	1	apr	

5 rows × 33 columns

In [72]:
```
# Train data set statistical values
data_train.describe()
```

Out[72]:

	season_Spring	season_Summer	season_Winter	mnth_aug	mnth_dec	mnth_feb	mnth_jan	mnth_jul	mnth_jun	mnth_mar	...	wee
count	510.000000	510.000000	510.00000	510.000000	510.000000	510.000000	510.000000	510.000000	510.000000	510.000000	...	5
mean	0.243137	0.245098	0.24902	0.096078	0.084314	0.066667	0.088235	0.076471	0.074510	0.098039	...	
std	0.429398	0.430568	0.43287	0.294988	0.278131	0.249689	0.283915	0.266010	0.262857	0.297660	...	
min	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
25%	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
50%	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
75%	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
max	1.000000	1.000000	1.00000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	

8 rows × 29 columns

In [73]:
```
# Test data set shape
data_test.shape
```

Out[73]: (219, 33)

In [74]:
```
## Test data set statistical values
data_test.describe()
```

Out[74]:

	season_Spring	season_Summer	season_Winter	mnth_aug	mnth_dec	mnth_feb	mnth_jan	mnth_jul	mnth_jun	mnth_mar	...	wee
count	219.000000	219.000000	219.000000	219.000000	219.000000	219.000000	219.000000	219.000000	219.000000	219.000000	...	2
mean	0.255708	0.264840	0.232877	0.059361	0.086758	0.100457	0.077626	0.105023	0.095890	0.054795	...	
std	0.437258	0.442259	0.423633	0.236840	0.282125	0.301297	0.268194	0.307285	0.295115	0.228100	...	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
75%	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	

8 rows × 29 columns

In [75]:
```
## Train data set first 5 rows
data_test.head()
```

Out[75]:

	season_Spring	season_Summer	season_Winter	mnth_aug	mnth_dec	mnth_feb	mnth_jan	mnth_jul	mnth_jun	mnth_mar	...	yr	mnth	holid
184	0	0	0	0	0	0	0	1	0	0	...	0	jul	
535	0	1	0	0	0	0	0	0	1	0	...	1	jun	
299	0	0	1	0	0	0	0	0	0	0	...	0	oct	
221	0	0	0	1	0	0	0	0	0	0	...	0	aug	
152	0	1	0	0	0	0	0	0	1	0	...	0	jun	

5 rows × 33 columns

Performing scaling

In [76]:
```
#Use Normalized scaler to scale
scaler = MinMaxScaler()
```

Performing scaling on train set

In [77]:
```
#We can apply scaller to all columns except dummy variables and target variable
scale_columns=['temp','hum','windspeed']
data_train[scale_columns] = scaler.fit_transform(data_train[scale_columns])
```

In [78]:
```
# again checking for values are changed or not
data_train.head()
```

Out[78]:

	season_Spring	season_Summer	season_Winter	mnth_aug	mnth_dec	mnth_feb	mnth_jan	mnth_jul	mnth_jun	mnth_mar	...	yr	mnth	holida
653	0	0	1	0	0	0	0	0	0	0	...	1	oct	
576	0	0	0	0	0	0	0	1	0	0	...	1	jul	
426	1	0	0	0	0	0	0	0	0	1	...	1	mar	
728	1	0	0	0	1	0	0	0	0	0	...	1	dec	
482	0	1	0	0	0	0	0	0	0	0	...	1	apr	

5 rows × 33 columns

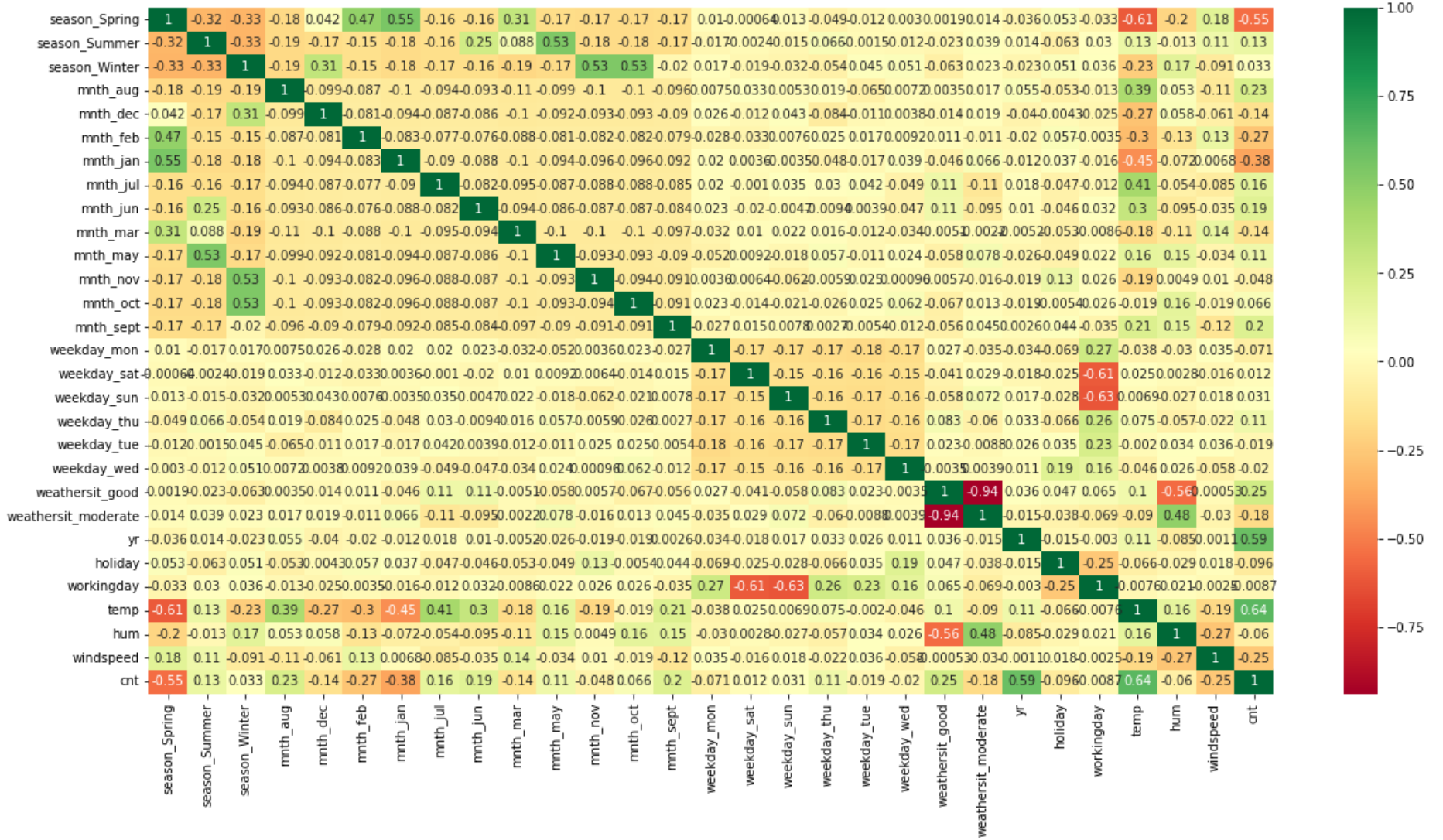
In [79]:
```
## checking data set statistical values after scaling
data_train.describe()
```

Out[79]:

	season_Spring	season_Summer	season_Winter	mnth_aug	mnth_dec	mnth_feb	mnth_jan	mnth_jul	mnth_jun	mnth_mar	...	wee
count	510.000000	510.000000	510.00000	510.000000	510.000000	510.000000	510.000000	510.000000	510.000000	510.000000	...	5
mean	0.243137	0.245098	0.24902	0.096078	0.084314	0.066667	0.088235	0.076471	0.074510	0.098039	...	
std	0.429398	0.430568	0.43287	0.294988	0.278131	0.249689	0.283915	0.266010	0.262857	0.297660	...	
min	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
25%	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
50%	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
75%	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
max	1.000000	1.000000	1.00000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	

8 rows × 29 columns

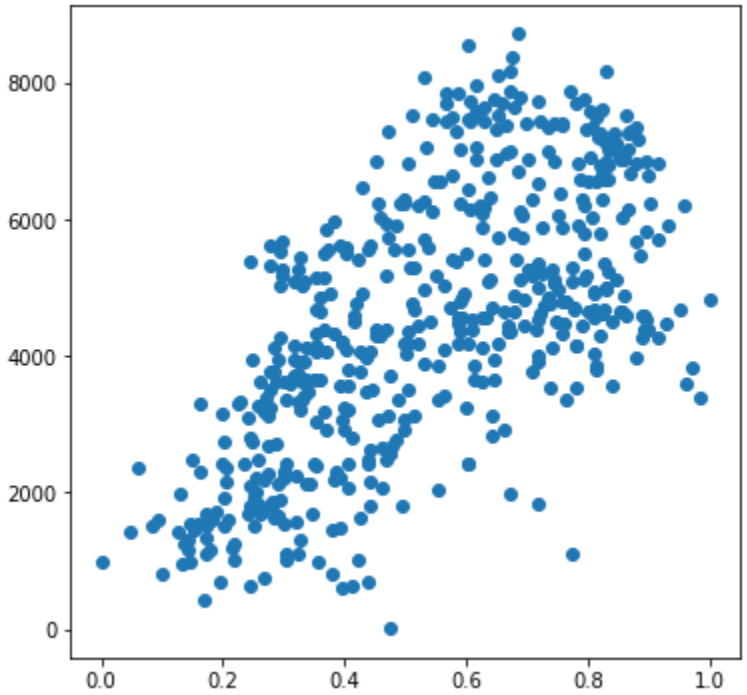
In [80]:
```
# Let's check the correlation coefficients to see which variables are highly correlated for the train set
plt.figure(figsize = (20, 10))
sns.heatmap(data_train.corr(), annot = True, cmap="RdYlGn")
plt.show()
```



There is multi-colinearity between the variables. . We need to consider the factors when developing the model. . workingday variable has high negative correlation with Sat & Sun (where workingday =0) . Spring is negatively correlated with temp . weathersit_moderate and humidity (hum)has correlation . various months and corresponding weather has correlation

In [81]:
```
##Let's see a scatterplot for temp vs cnt.

plt.figure(figsize=[6,6])
plt.scatter(data_train.temp, data_train.cnt)
plt.show()
```



We pick temp as the first variable and we'll try to fit a regression line to that.

Dividing the data into X train and Ytrain

In [82]:
```
#y is contain only target variable
y_train = data_train.pop("cnt")
#X is all remainign variable also our independent variables
X_train = data_train
```

In [83]:
```
#Inspect independent variables
X_train.head()
```

Out[83]:

›n_Winter	mnth_aug	mnth_dec	mnth_feb	mnth_jan	mnth_jul	mnth_jun	mnth_mar	...	season	yr	mnth	holiday	weekday	workingday	weathersit
1	0	0	0	0	0	0	0	...	Winter	1	oct	0	wed	1	good
0	0	0	0	0	1	0	0	...	Fall	1	jul	0	wed	1	good
0	0	0	0	0	0	0	1	...	Spring	1	mar	0	sun	0	moderate
0	0	1	0	0	0	0	0	...	Spring	1	dec	0	mon	1	good
0	0	0	0	0	0	0	0	...	Summer	1	apr	0	sun	0	moderate

In [106]:
```
X_train.drop(['season'], axis=1, inplace=True)
```

In [107]:
```
X_train.columns
```

Out[107]:
```
Index(['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_sept', 'weekday_mon',
       'weekday_sat', 'weekday_sun', 'weekday_thu', 'weekday_tue',
       'weekday_wed', 'weathersit_good', 'weathersit_moderate', 'yr', 'mnth',
       'holiday', 'weekday', 'workingday', 'weathersit', 'temp', 'hum',
       'windspeed'],
      dtype='object')
```

In [101]:
```
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 510 entries, 653 to 79
Data columns (total 32 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   season_Spring        510 non-null    uint8
 1   season_Summer        510 non-null    uint8
 2   season_Winter        510 non-null    uint8
 3   mnth_aug             510 non-null    uint8
 4   mnth_dec             510 non-null    uint8
 5   mnth_feb             510 non-null    uint8
 6   mnth_jan             510 non-null    uint8
 7   mnth_jul             510 non-null    uint8
 8   mnth_jun             510 non-null    uint8
 9   mnth_mar             510 non-null    uint8
 10  mnth_may             510 non-null    uint8
 11  mnth_nov             510 non-null    uint8
 12  mnth_oct             510 non-null    uint8
 13  mnth_sept            510 non-null    uint8
 14  weekday_mon          510 non-null    uint8
 15  weekday_sat          510 non-null    uint8
 16  weekday_sun          510 non-null    uint8
 17  weekday_thu          510 non-null    uint8
 18  weekday_tue          510 non-null    uint8
 19  weekday_wed          510 non-null    uint8
 20  weathersit_good      510 non-null    uint8
 21  weathersit_moderate  510 non-null    uint8
 22  season               510 non-null    object
 23  yr                   510 non-null    int64
 24  mnth                 510 non-null    object
 25  holiday              510 non-null    int64
 26  weekday              510 non-null    object
 27  workingday           510 non-null    int64
 28  weathersit           510 non-null    object
 29  temp                 510 non-null    float64
 30  hum                  510 non-null    float64
 31  windspeed            510 non-null    float64
dtypes: float64(3), int64(3), object(4), uint8(22)
memory usage: 54.8+ KB
```

In [113]:
```
X_train.drop(['mnth'], axis=1, inplace=True)
```

In [115]:
```
X_train.drop(['weekday'], axis=1, inplace=True)
```

In [117]:
```
X_train.drop(['weathersit'], axis=1, inplace=True)
```

As we have already expanded season, weathersit, mnth to seperate columns, so, we do not need specific columns. Hence, we can drop those columns from the dataset to perform linear regression

Performing Linear Regression

Fit a regression line through the training dataset using statsmodels. . In statsmodels, you need to explicitly fit a constant using sm.add_constant(X) because if we don't perform this step, statsmodels fits a regression line passing through the origin, by default.

RFE is a Automated approach

In [118]:
```
# Build a Lienar Regression model using SKLearn for RFE
lr = LinearRegression()
lr.fit(X_train,y_train)
```

Out[118]:
```
LinearRegression()
```

In [123]:
```
rfe = RFE(lr)
rfe.fit(X_train,y_train)
```

Out[123]:
```
RFE(estimator=LinearRegression())
```

In [124]:
```
# Checking which parameters have been selected in that list of 15
list(zip(X_train.columns,rfe.support_,rfe.ranking_))
```

Out[124]:
```
[('season_Spring', True, 1),
 ('season_Summer', True, 1),
 ('season_Winter', True, 1),
 ('mnth_aug', False, 10),
 ('mnth_dec', False, 2),
 ('mnth_feb', False, 3),
 ('mnth_jan', True, 1),
 ('mnth_jul', True, 1),
 ('mnth_jun', False, 14),
 ('mnth_mar', False, 15),
 ('mnth_may', False, 7),
 ('mnth_nov', True, 1),
 ('mnth_oct', False, 12),
 ('mnth_sept', True, 1),
 ('weekday_mon', False, 9),
 ('weekday_sat', False, 5),
 ('weekday_sun', False, 6),
 ('weekday_thu', False, 13),
 ('weekday_tue', False, 8),
 ('weekday_wed', False, 11),
 ('weathersit_good', True, 1),
 ('weathersit_moderate', True, 1),
 ('yr', True, 1),
 ('holiday', True, 1),
 ('workingday', False, 4),
 ('temp', True, 1),
 ('hum', True, 1),
 ('windspeed', True, 1)]
```

In [125]:
```
#Print Columns selected by RFE. We will start with these columns for manual elimination
col = X_train.columns[rfe.support_]
col
```

Out[125]:
```
Index(['season_Spring', 'season_Summer', 'season_Winter', 'mnth_jan',
       'mnth_jul', 'mnth_nov', 'mnth_sept', 'weathersit_good',
       'weathersit_moderate', 'yr', 'holiday', 'temp', 'hum', 'windspeed'],
      dtype='object')
```

In [126]:
```
# rfe not selected variables
X_train.columns[~rfe.support_]
```

Out[126]:
```
Index(['mnth_aug', 'mnth_dec', 'mnth_feb', 'mnth_jun', 'mnth_mar', 'mnth_may',
       'mnth_oct', 'weekday_mon', 'weekday_sat', 'weekday_sun', 'weekday_thu',
       'weekday_tue', 'weekday_wed', 'workingday'],
      dtype='object')
```

In [127]:
```
# Creating X_test dataframe with RFE selected variables
X_train_rfe = X_train[col]
```

Manual Model Development using statsmodel

Function to build a model using statsmodel api

In [128]:
```
#Function to build a model using statsmodel api - Takes the columns to be selected for model as a parameter
def build_model(cols):
    X_train_sm = sm.add_constant(X_train[cols])
    lm = sm.OLS(y_train, X_train_sm).fit()
    print(lm.summary())
    return lm
```

Function to calculate VIFs and print them

In [129]:
```
#Function to calculate VIFs and print them -Takes the columns for which VIF to be calcualted as a parameter
def get_vif(cols):
    df1 = X_train[cols]
    vif = pd.DataFrame()
    vif['Features'] = df1.columns
    vif['VIF'] = [variance_inflation_factor(df1.values, i) for i in range(df1.shape[1])]
    vif['VIF'] = round(vif['VIF'],2)
    print(vif.sort_values(by='VIF',ascending=False))
```

Model 1

In [130]:
```
# Add a constant
X_train_lm = sm.add_constant(X_train)

# Create a first fitted model
lr = sm.OLS(y_train, X_train_lm).fit()
```

In [131]:
```
# Check the parameters obtained
lr.params
```

Out[131]:
```
const                 1265.234105
season_Spring         -379.881612
season_Summer          367.175174
season_Winter          963.916145
mnth_aug               186.939629
mnth_dec              -396.956902
mnth_feb              -324.027005
mnth_jan              -552.913637
mnth_jul              -303.434913
mnth_jun                 8.470414
mnth_mar                 5.284566
mnth_may               209.165719
mnth_nov              -361.193045
mnth_oct                50.326183
mnth_sept              747.990361
weekday_mon           -214.109834
weekday_sat           -489.525791
weekday_sun           -434.264536
weekday_thu             33.579662
weekday_tue           -228.592742
weekday_wed            -97.705077
weathersit_good       2186.132333
weathersit_moderate   1670.217437
yr                    2006.607467
holiday              -1177.104431
workingday            -526.076655
temp                  3871.920691
hum                  -1367.379594
windspeed            -1591.036910
dtype: float64
```

In [132]:
```
# Print a summary of the linear regression model obtained
print(lr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.851
Model:                            OLS   Adj. R-squared:                  0.842
Method:                 Least Squares   F-statistic:                     98.04
Date:                Tue, 06 Sep 2022   Prob (F-statistic):          4.33e-179
Time:                        15:00:02   Log-Likelihood:                -4102.0
No. Observations:                 510   AIC:                             8262.
Df Residuals:                     481   BIC:                             8385.
Df Model:                          28
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                   1265.2341    783.541      1.615      0.107    -274.351    2804.819
season_Spring           -379.8816    262.991     -1.444      0.149    -896.634     136.871
season_Summer            367.1752    229.817      1.598      0.111     -84.395     818.745
season_Winter            963.9161    244.804      3.938      0.000     482.899    1444.933
mnth_aug                 186.9396    295.632      0.632      0.527    -393.951     767.830
mnth_dec                -396.9569    294.437     -1.348      0.178    -975.499     181.585
mnth_feb                -324.0270    289.131     -1.121      0.263    -892.142     244.088
mnth_jan                -552.9136    294.157     -1.880      0.061   -1130.906      25.079
mnth_jul                -303.4349    308.030     -0.985      0.325    -908.685     301.815
mnth_jun                   8.4704    220.118      0.038      0.969    -424.042     440.983
mnth_mar                   5.2846    215.164      0.025      0.980    -417.493     428.062
mnth_may                 209.1657    184.402      1.134      0.257    -153.167     571.499
mnth_nov                -361.1930    318.828     -1.133      0.258    -987.661     265.275
mnth_oct                  50.3262    314.042      0.160      0.873    -566.737     667.389
mnth_sept                747.9904    280.460      2.667      0.008     196.913    1299.068
weekday_mon             -214.1098    126.673     -1.690      0.092    -463.011      34.792
weekday_sat             -489.5258    622.151     -0.787      0.432   -1711.995     732.944
weekday_sun             -434.2645    621.379     -0.699      0.485   -1655.217     786.688
weekday_thu               33.5797    129.996      0.258      0.796    -221.851     289.010
weekday_tue             -228.5927    127.532     -1.792      0.074    -479.181      21.996
weekday_wed              -97.7051    132.754     -0.736      0.462    -358.555     163.145
weathersit_good         2186.1323    231.202      9.455      0.000    1731.841    2640.424
weathersit_moderate     1670.2174    218.772      7.634      0.000    1240.350    2100.085
yr                      2006.6075     70.688     28.387      0.000    1867.711    2145.504
holiday                -1177.1044    568.017     -2.072      0.039   -2293.207     -61.002
workingday              -526.0767    617.341     -0.852      0.395   -1739.095     686.942
temp                    3871.9207    407.000      9.513      0.000    3072.203    4671.639
hum                    -1367.3796    337.720     -4.049      0.000   -2030.968    -703.791
```

```
windspeed           -1591.0369    226.131     -7.036      0.000   -2035.364   -1146.710
==============================================================================
Omnibus:                       78.889   Durbin-Watson:                   1.998
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              209.035
Skew:                          -0.768   Prob(JB):                     4.06e-46
Kurtosis:                       5.735   Cond. No.                         74.9
==============================================================================

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

In [133]:
```
# We have high P value for mnth_March feature, so we can drop the column and fit the model again.
X_train=X_train.drop('mnth_mar', 1)
```

Model 2

In [ ]:
```
```

In [134]:
```
X_train_lm = sm.add_constant(X_train)

# Creating a first fitted model
lr_1 = sm.OLS(y_train, X_train_lm).fit()
```

In [135]:
```
# Print the summary of the model
print(lr_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.851
Model:                            OLS   Adj. R-squared:                  0.843
Method:                 Least Squares   F-statistic:                     101.9
Date:                Tue, 06 Sep 2022   Prob (F-statistic):          4.24e-180
Time:                        15:02:38   Log-Likelihood:                -4102.0
No. Observations:                 510   AIC:                             8260.
Df Residuals:                     482   BIC:                             8379.
Df Model:                          27
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                   1268.5969    770.685      1.646      0.100    -245.721    2782.915
season_Spring           -377.8068    248.795     -1.519      0.130    -866.664     111.051
season_Summer            366.0922    225.314      1.625      0.105     -76.626     808.811
season_Winter            965.0316    240.304      4.016      0.000     492.858    1437.205
mnth_aug                 184.5402    278.734      0.662      0.508    -363.143     732.224
mnth_dec                -401.5310    227.820     -1.762      0.079    -849.173      46.111
mnth_feb                -329.2613    195.187     -1.687      0.092    -712.784      54.262
mnth_jan                -558.3017    195.761     -2.852      0.005    -942.952    -173.652
mnth_jul                -305.7050    293.530     -1.041      0.298    -882.462     271.052
mnth_jun                   6.7664    208.682      0.032      0.974    -403.272     416.804
mnth_may                 207.5716    172.423      1.204      0.229    -131.223     546.366
mnth_nov                -365.3521    269.859     -1.354      0.176    -895.598     164.894
mnth_oct                  46.3318    268.371      0.173      0.863    -480.989     573.653
mnth_sept                745.1279    254.833      2.924      0.004     244.408    1245.848
weekday_mon              -214.2622    126.390     -1.695      0.091    -462.606      34.081
weekday_sat              -489.4700    621.502     -0.788      0.431   -1710.657     731.717
weekday_sun              -434.1562    620.719     -0.699      0.485   -1653.805     785.493
weekday_thu                33.6045    129.857      0.259      0.796    -221.552     288.761
weekday_tue              -228.6873    127.342     -1.796      0.073    -478.901      21.526
weekday_wed               -97.7580    132.599     -0.737      0.461    -358.302     162.786
weathersit_good          2186.0901    230.956      9.465      0.000    1732.285    2639.895
weathersit_moderate      1670.1435    218.525      7.643      0.000    1240.765    2099.522
yr                       2006.5985     70.614     28.416      0.000    1867.849    2145.348
holiday                 -1177.0900    567.428     -2.074      0.039   -2292.028     -62.152
workingday               -525.9774    616.687     -0.853      0.394   -1737.705     685.750
temp                     3870.2610    400.935      9.653      0.000    3082.464    4658.058
hum                     -1366.7201    336.301     -4.064      0.000   -2027.518    -705.922
windspeed               -1591.3007    225.642     -7.052      0.000   -2034.664   -1147.938
```

```
==============================================================================
Omnibus:                       78.874   Durbin-Watson:                   1.997
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              209.108
Skew:                          -0.768   Prob(JB):                     3.92e-46
Kurtosis:                       5.735   Cond. No.                         74.6
==============================================================================

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

In [136]:
```
# We have high P value for mnth_March feature, so we can drop the column and fit the model again
X_train=X_train.drop('mnth_jun', 1)
```

Model 3

In [137]:
```
# Add a constant
X_train_lm = sm.add_constant(X_train)

# Create a first fitted model
lr_2 = sm.OLS(y_train, X_train_lm).fit()
```

In [138]:
```
# Print the summary of the model
print(lr_2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.851
Model:                            OLS   Adj. R-squared:                  0.843
Method:                 Least Squares   F-statistic:                     106.0
Date:                Tue, 06 Sep 2022   Prob (F-statistic):          4.08e-181
Time:                        15:03:19   Log-Likelihood:                -4102.0
No. Observations:                 510   AIC:                             8258.
Df Residuals:                     483   BIC:                             8372.
Df Model:                          26
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                   1269.2404    769.633      1.649      0.100    -243.001    2781.482
season_Spring           -380.3562    235.801     -1.613      0.107    -843.678      82.966
season_Summer            364.2125    217.503      1.675      0.095     -63.156     791.581
season_Winter            963.5965    235.948      4.084      0.000     499.984    1427.209
mnth_aug                 178.9242    218.162      0.820      0.413    -249.740     607.588
mnth_dec                -402.1413    226.806     -1.773      0.077    -847.789      43.507
mnth_feb                -328.7339    194.307     -1.692      0.091    -710.525      53.057
mnth_jan                -557.3123    193.168     -2.885      0.004    -936.866    -177.759
mnth_jul                -311.7603    226.228     -1.378      0.169    -756.272     132.752
mnth_may                 204.9552    152.218      1.346      0.179     -94.137     504.047
mnth_nov                -366.6801    266.457     -1.376      0.169    -890.239     156.879
mnth_oct                  44.1610    259.617      0.170      0.865    -465.957     554.279
mnth_sept                740.6363    213.671      3.466      0.001     320.796    1160.477
weekday_mon             -214.0863    126.143     -1.697      0.090    -461.943      33.770
weekday_sat             -489.0099    620.697     -0.788      0.431   -1708.609     730.589
weekday_sun             -433.6540    619.884     -0.700      0.485   -1651.656     784.348
weekday_thu               33.3876    129.551      0.258      0.797    -221.165     287.940
weekday_tue             -228.6985    127.209     -1.798      0.073    -478.651      21.254
weekday_wed              -97.8181    132.449     -0.739      0.461    -358.065     162.429
weathersit_good         2186.0070    230.703      9.475      0.000    1732.702    2639.312
weathersit_moderate     1670.1710    218.297      7.651      0.000    1241.242    2099.100
yr                      2006.3688     70.185     28.587      0.000    1868.462    2144.275
holiday                -1176.9770    566.830     -2.076      0.038   -2290.735     -63.219
workingday              -525.4078    615.799     -0.853      0.394   -1735.384     684.569
temp                    3877.2022    338.651     11.449      0.000    3211.791    4542.613
hum                    -1367.8741    334.067     -4.095      0.000   -2024.278    -711.471
windspeed              -1592.0138    224.335     -7.097      0.000   -2032.807   -1151.220
==============================================================================
```

```
Omnibus:                       78.804   Durbin-Watson:                   1.997
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              208.608
Skew:                          -0.768   Prob(JB):                     5.03e-46
Kurtosis:                       5.731   Cond. No.                         74.6
==============================================================================

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

In [139]:
```
# We have high P value for mnth_October feature, so we can drop the column and fit the model again
X_train=X_train.drop('mnth_oct', 1)
```

Model 4

In [140]:
```
# Add a constant
X_train_lm = sm.add_constant(X_train)

# Create a first fitted model
lr_3 = sm.OLS(y_train, X_train_lm).fit()
```

In [141]:
```
# Print the summary of the model
print(lr_3.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.851
Model:                            OLS   Adj. R-squared:                  0.843
Method:                 Least Squares   F-statistic:                     110.5
Date:                Tue, 06 Sep 2022   Prob (F-statistic):          3.90e-182
Time:                        15:04:08   Log-Likelihood:                -4102.0
No. Observations:                 510   AIC:                             8256.
Df Residuals:                     484   BIC:                             8366.
Df Model:                          25
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                   1283.4914    764.291      1.679      0.094    -218.247    2785.230
season_Spring           -385.9873    233.231     -1.655      0.099    -844.258      72.283
season_Summer            351.5112    204.079      1.722      0.086     -49.479     752.501
season_Winter            985.1946    198.671      4.959      0.000     594.830    1375.559
mnth_aug                 167.3497    207.071      0.808      0.419    -239.519     574.219
mnth_dec                -429.5721    159.325     -2.696      0.007    -742.625    -116.519
mnth_feb                -337.1201    187.760     -1.795      0.073    -706.045      31.805
mnth_jan                -566.0317    186.056     -3.042      0.002    -931.608    -200.455
mnth_jul                -322.8524    216.408     -1.492      0.136    -748.068     102.363
mnth_may                 204.9992    152.065      1.348      0.178     -93.791     503.789
mnth_nov                -402.1002    166.085     -2.421      0.016    -728.437     -75.764
mnth_sept                723.2986    187.606      3.855      0.000     354.677    1091.921
weekday_mon             -213.0844    125.879     -1.693      0.091    -460.421      34.252
weekday_sat             -491.7492    619.865     -0.793      0.428   -1709.708     726.209
weekday_sun             -436.8795    618.972     -0.706      0.481   -1653.083     779.324
weekday_thu               33.8718    129.389      0.262      0.794    -220.363     288.106
weekday_tue             -228.7047    127.082     -1.800      0.073    -478.405      20.995
weekday_wed              -97.2537    132.274     -0.735      0.463    -357.157     162.649
weathersit_good         2185.2494    230.428      9.483      0.000    1732.486    2638.013
weathersit_moderate     1668.7591    217.920      7.658      0.000    1240.573    2096.946
yr                      2006.1059     70.098     28.619      0.000    1868.372    2143.840
holiday                -1176.6479    566.258     -2.078      0.038   -2289.276     -64.020
workingday              -528.9778    614.824     -0.860      0.390   -1737.031     679.076
temp                    3872.2829    337.075     11.488      0.000    3209.971    4534.594
hum                    -1360.6950    331.057     -4.110      0.000   -2011.182    -710.208
windspeed              -1589.4964    223.622     -7.108      0.000   -2028.886   -1150.107
==============================================================================
Omnibus:                       78.293   Durbin-Watson:                   1.997
```

```
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              206.411
Skew:                          -0.764   Prob(JB):                    1.51e-45
Kurtosis:                       5.716   Cond. No.                        74.4
==============================================================================

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

In [142]:
```
# We have high P value for weekday_Wednesday feature, so we can drop the column and fit the model again.
X_train=X_train.drop('weekday_thu', 1)
```

Model 5

In [143]:
```
# Add a constant
X_train_lm = sm.add_constant(X_train)

# Create a first fitted model
lr_4 = sm.OLS(y_train, X_train_lm).fit()
```

In [144]:
```
# Print the summary of the model
print(lr_4.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.851
Model:                            OLS   Adj. R-squared:                  0.843
Method:                 Least Squares   F-statistic:                     115.3
Date:                Tue, 06 Sep 2022   Prob (F-statistic):          3.71e-183
Time:                        15:05:26   Log-Likelihood:                -4102.0
No. Observations:                 510   AIC:                             8254.
Df Residuals:                     485   BIC:                             8360.
Df Model:                          24
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                   1294.7762    762.341      1.698      0.090    -203.123    2792.676
season_Spring           -386.1316    233.006     -1.657      0.098    -843.958      71.695
season_Summer            353.4036    203.755      1.734      0.083     -46.948     753.755
season_Winter            987.2630    198.323      4.978      0.000     597.584    1376.942
mnth_aug                 169.3430    206.732      0.819      0.413    -236.858     575.544
mnth_dec                -432.3147    158.827     -2.722      0.007    -744.389    -120.241
mnth_feb                -335.6226    187.492     -1.790      0.074    -704.020      32.775
mnth_jan                -566.4229    185.871     -3.047      0.002    -931.634    -201.211
mnth_jul                -318.9945    215.698     -1.479      0.140    -742.813     104.824
mnth_may                 205.9379    151.877      1.356      0.176     -92.480     504.356
mnth_nov                -403.7243    165.810     -2.435      0.015    -729.518     -77.931
mnth_sept                724.4056    187.378      3.866      0.000     356.233    1092.578
weekday_mon             -229.7570    108.474     -2.118      0.035    -442.893     -16.621
weekday_sat             -507.0516    616.510     -0.822      0.411   -1718.411     704.308
weekday_sun             -452.3675    615.546     -0.735      0.463   -1661.833     757.098
weekday_tue             -245.5902    109.390     -2.245      0.025    -460.527     -30.653
weekday_wed             -113.8943    115.888     -0.983      0.326    -341.598     113.810
weathersit_good         2190.9226    229.187      9.560      0.000    1740.601    2641.244
weathersit_moderate     1673.7038    216.891      7.717      0.000    1247.541    2099.867
yr                      2006.6862     69.996     28.669      0.000    1869.154    2144.218
holiday                -1176.5472    565.714     -2.080      0.038   -2288.100     -64.995
workingday              -527.6995    614.214     -0.859      0.391   -1734.548     679.149
temp                    3870.3627    336.672     11.496      0.000    3208.847    4531.878
hum                    -1362.8657    330.636     -4.122      0.000   -2012.521    -713.211
windspeed              -1590.2025    223.391     -7.118      0.000   -2029.136   -1151.269
==============================================================================
Omnibus:                       78.598   Durbin-Watson:                   1.998
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              207.793
```

```
Skew:                          -0.766   Prob(JB):                     7.56e-46
Kurtosis:                       5.726   Cond. No.                         74.0
==============================================================================

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

In [145]:
```
# We have high P value for weekday_Sunday feature, so we can drop the column and fit the model again¶
X_train=X_train.drop('weekday_sun', 1)
```

Model 6

In [149]:
```
# Add a constant
X_train_lm = sm.add_constant(X_train)

# Create a first fitted model
lr_5 = sm.OLS(y_train, X_train_lm).fit()
```

In [153]:
```
# Print the summary of the model
print(lr_5.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.851
Model:                            OLS   Adj. R-squared:                  0.844
Method:                 Least Squares   F-statistic:                     120.4
Date:                Tue, 06 Sep 2022   Prob (F-statistic):          4.37e-184
Time:                        15:06:08   Log-Likelihood:                -4102.3
No. Observations:                 510   AIC:                             8253.
Df Residuals:                     486   BIC:                             8354.
Df Model:                          23
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                    838.3605    441.895      1.897      0.058     -29.900    1706.621
season_Spring           -383.3899    232.866     -1.646      0.100    -840.939      74.159
season_Summer            354.1287    203.656      1.739      0.083     -46.027     754.284
season_Winter            985.1335    198.208      4.970      0.000     595.683    1374.584
mnth_aug                 166.2607    206.592      0.805      0.421    -239.662     572.184
mnth_dec                -430.0622    158.722     -2.710      0.007    -741.929    -118.196
mnth_feb                -332.2463    187.347     -1.773      0.077    -700.357      35.864
mnth_jan                -564.4085    185.763     -3.038      0.003    -929.406    -199.411
mnth_jul                -322.3897    215.547     -1.496      0.135    -745.909     101.129
mnth_may                 203.2468    151.761      1.339      0.181     -94.942     501.436
mnth_nov                -414.6825    165.060     -2.512      0.012    -739.001     -90.364
mnth_sept                725.5359    187.283      3.874      0.000     357.552    1093.520
weekday_mon             -224.9140    108.222     -2.078      0.038    -437.555     -12.273
weekday_sat              -64.5682    132.468     -0.487      0.626    -324.849     195.712
weekday_tue             -238.3622    108.896     -2.189      0.029    -452.326     -24.398
weekday_wed             -102.4548    114.783     -0.893      0.373    -327.987     123.078
weathersit_good         2189.4179    229.069      9.558      0.000    1739.330    2639.506
weathersit_moderate     1672.3159    216.781      7.714      0.000    1246.373    2098.259
yr                      2009.6982     69.842     28.775      0.000    1872.468    2146.928
holiday                 -802.7302    247.465     -3.244      0.001   -1288.963    -316.498
workingday               -84.0146    112.939     -0.744      0.457    -305.924     137.895
temp                    3877.7777    336.361     11.529      0.000    3216.876    4538.679
hum                    -1353.0297    330.208     -4.098      0.000   -2001.842    -704.218
windspeed              -1594.6527    223.203     -7.144      0.000   -2033.215   -1156.091
==============================================================================
Omnibus:                       78.549   Durbin-Watson:                   1.998
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              204.045
Skew:                          -0.771   Prob(JB):                     4.92e-45
```

```
Kurtosis:                       5.687   Cond. No.                         31.1
==============================================================================

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

In [154]:
```
# Calculate the VIFs again for the model
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[154]:

	Features	VIF
21	hum	29.27
20	temp	28.06
15	weathersit_good	17.77
16	weathersit_moderate	10.02
0	season_Spring	9.21
19	workingday	7.56
1	season_Summer	7.15
2	season_Winter	7.01
22	windspeed	4.84
3	mnth_aug	3.32
7	mnth_jul	2.88
6	mnth_jan	2.53
10	mnth_sept	2.27
17	yr	2.12
9	mnth_nov	1.97
5	mnth_feb	1.97
12	weekday_sat	1.90
4	mnth_dec	1.77
8	mnth_may	1.61
11	weekday_mon	1.55
13	weekday_tue	1.54
14	weekday_wed	1.53

	Features	VIF
18	holiday	1.33

Choosing the variable with high p-value and high VIF value . The variable workingday has a significantly high VIF (7.56) and a high p-value (0.457) as well. Hence, this variable isn't much use and it should be dropped.

In [155]:
```
# We have high P value for workingday feature, so we can drop the column and fit the model again¶
X_train=X_train.drop('workingday', 1)
```

Model 7

In [156]:
```
# Add a constant
X_train_lm = sm.add_constant(X_train)

# Create a first fitted model
lr_6 = sm.OLS(y_train, X_train_lm).fit()
```

In [157]:
```
# Print the summary of the model
print(lr_6.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.851
Model:                            OLS   Adj. R-squared:                  0.844
Method:                 Least Squares   F-statistic:                     126.0
Date:                Tue, 06 Sep 2022   Prob (F-statistic):          5.06e-185
Time:                        15:07:29   Log-Likelihood:                -4102.6
No. Observations:                 510   AIC:                             8251.
Df Residuals:                     487   BIC:                             8349.
Df Model:                          22
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                    784.6589    435.758      1.801      0.072     -71.540    1640.857
season_Spring           -383.4162    232.759     -1.647      0.100    -840.753      73.921
season_Summer            353.6117    203.562      1.737      0.083     -46.356     753.579
season_Winter            986.9756    198.102      4.982      0.000     597.736    1376.215
mnth_aug                 167.5819    206.489      0.812      0.417    -238.138     573.302
mnth_dec                -426.8122    158.589     -2.691      0.007    -738.416    -115.208
mnth_feb                -333.2490    187.256     -1.780      0.076    -701.179      34.681
mnth_jan                -564.6508    185.677     -3.041      0.002    -929.478    -199.824
mnth_jul                -315.9335    215.273     -1.468      0.143    -738.913     107.046
mnth_may                 201.7779    151.679      1.330      0.184     -96.247     499.803
mnth_nov                -424.1003    164.498     -2.578      0.010    -747.313    -100.887
mnth_sept                726.8056    187.189      3.883      0.000     359.008    1094.604
weekday_mon             -251.6609    102.028     -2.467      0.014    -452.130     -51.192
weekday_sat               -8.3838    108.777     -0.077      0.939    -222.114     205.346
weekday_tue             -264.5946    102.980     -2.569      0.010    -466.935     -62.254
weekday_wed             -128.0455    109.457     -1.170      0.243    -343.111      87.020
weathersit_good         2190.4504    228.960      9.567      0.000    1740.579    2640.321
weathersit_moderate     1679.5136    216.465      7.759      0.000    1254.193    2104.835
yr                      2010.8310     69.794     28.811      0.000    1873.697    2147.965
holiday                 -727.4553    225.726     -3.223      0.001   -1170.973    -283.937
temp                    3874.5628    336.179     11.525      0.000    3214.022    4535.103
hum                    -1363.2361    329.772     -4.134      0.000   -2011.187    -715.285
windspeed              -1592.4539    223.081     -7.138      0.000   -2030.774   -1154.134
==============================================================================
Omnibus:                       75.972   Durbin-Watson:                   1.994
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              199.058
Skew:                          -0.744   Prob(JB):                     5.96e-44
Kurtosis:                       5.674   Cond. No.                         28.4
```

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

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

In [ ]:
```
# We have high P value for weekday_Saturday feature, so we can drop the column and fit the model again¶
X_train=X_train.drop('weekday_sat', 1)
```

Model 8

In [164]:
```
# Add a constant
X_train_lm = sm.add_constant(X_train)

# Create a first fitted model
lr_7 = sm.OLS(y_train, X_train_lm).fit()
```

In [165]:
```
# Print the summary of the model
print(lr_7.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.851
Model:                            OLS   Adj. R-squared:                  0.844
Method:                 Least Squares   F-statistic:                     132.2
Date:                Tue, 06 Sep 2022   Prob (F-statistic):          4.36e-186
Time:                        15:08:34   Log-Likelihood:                -4102.6
No. Observations:                 510   AIC:                             8249.
Df Residuals:                     488   BIC:                             8342.
Df Model:                          21
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                   782.1187    434.067      1.802      0.072     -70.753    1634.990
season_Spring          -384.1525    232.326     -1.654      0.099    -840.636      72.331
season_Summer           353.1493    203.266      1.737      0.083     -46.235     752.533
season_Winter           986.4677    197.790      4.987      0.000     597.842    1375.093
mnth_aug                166.9326    206.107      0.810      0.418    -238.034     571.899
mnth_dec               -426.7732    158.427     -2.694      0.007    -738.056    -115.490
mnth_feb               -332.7548    186.956     -1.780      0.076    -700.093      34.583
mnth_jan               -564.6571    185.488     -3.044      0.002    -929.111    -200.204
mnth_jul               -316.3974    214.970     -1.472      0.142    -738.778     105.983
mnth_may                201.7163    151.522      1.331      0.184     -96.000     499.432
mnth_nov               -424.2760    164.314     -2.582      0.010    -747.127    -101.425
mnth_sept               726.4097    186.928      3.886      0.000     359.127    1093.692
weekday_mon            -249.7089     98.734     -2.529      0.012    -443.705     -55.713
weekday_tue            -262.6862     99.857     -2.631      0.009    -458.889     -66.483
weekday_wed            -126.0827    106.344     -1.186      0.236    -335.032      82.866
weathersit_good        2191.2706    228.479      9.591      0.000    1742.346    2640.195
weathersit_moderate    1680.0975    216.112      7.774      0.000    1255.472    2104.723
yr                     2010.9424     69.708     28.848      0.000    1873.978    2147.907
holiday                -727.7074    225.473     -3.227      0.001   -1170.725    -284.690
temp                   3874.2091    335.805     11.537      0.000    3214.406    4534.012
hum                   -1362.5439    329.313     -4.138      0.000   -2009.591    -715.497
windspeed             -1592.1317    222.815     -7.146      0.000   -2029.926   -1154.337
==============================================================================
Omnibus:                       75.945   Durbin-Watson:                   1.993
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              198.838
Skew:                          -0.744   Prob(JB):                     6.65e-44
Kurtosis:                       5.672   Cond. No.                         28.3
==============================================================================
```

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

In [166]:
```
# We have high P value for mnth_August feature, so we can drop the column and fit the model again
X_train=X_train.drop('mnth_aug', 1)
```

Model 9

In [167]:
```
# Add a constant
X_train_lm = sm.add_constant(X_train)

# Create a first fitted model
lr_8 = sm.OLS(y_train, X_train_lm).fit()
```

In [168]:
```
# Print the summary of the model
print(lr_8.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.850
Model:                            OLS   Adj. R-squared:                  0.844
Method:                 Least Squares   F-statistic:                     138.9
Date:                Tue, 06 Sep 2022   Prob (F-statistic):          5.05e-187
Time:                        15:09:19   Log-Likelihood:                -4102.9
No. Observations:                 510   AIC:                             8248.
Df Residuals:                     489   BIC:                             8337.
Df Model:                          20
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                    865.9740    421.391      2.055      0.040      38.014    1693.934
season_Spring           -490.5808    191.527     -2.561      0.011    -866.899    -114.263
season_Summer            240.5659    148.255      1.623      0.105     -50.729     531.861
season_Winter            887.5486    155.524      5.707      0.000     581.971    1193.126
mnth_dec                -434.6621    158.072     -2.750      0.006    -745.245    -124.079
mnth_feb                -332.6082    186.890     -1.780      0.076    -699.815      34.599
mnth_jan                -564.5136    185.423     -3.044      0.002    -928.837    -200.190
mnth_jul                -435.0615    157.255     -2.767      0.006    -744.040    -126.083
mnth_may                 197.8044    151.392      1.307      0.192     -99.654     495.263
mnth_nov                -434.5645    163.765     -2.654      0.008    -756.334    -112.794
mnth_sept                631.3645    145.452      4.341      0.000     345.577     917.152
weekday_mon             -247.0870     98.646     -2.505      0.013    -440.909     -53.265
weekday_tue             -264.0195     99.809     -2.645      0.008    -460.126     -67.913
weekday_wed             -124.2752    106.283     -1.169      0.243    -333.104      84.553
weathersit_good         2200.1820    228.134      9.644      0.000    1751.938    2648.426
weathersit_moderate     1691.9290    215.542      7.850      0.000    1268.426    2115.432
yr                      2012.5326     69.655     28.893      0.000    1875.672    2149.394
holiday                 -732.4101    225.319     -3.251      0.001   -1175.122    -289.698
temp                    3894.7081    334.732     11.635      0.000    3237.017    4552.399
hum                    -1351.0031    328.889     -4.108      0.000   -1997.214    -704.793
windspeed              -1593.5122    222.730     -7.154      0.000   -2031.137   -1155.887
==============================================================================
Omnibus:                       77.067   Durbin-Watson:                   1.997
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              201.505
Skew:                          -0.755   Prob(JB):                     1.75e-44
Kurtosis:                       5.683   Cond. No.                         27.2
==============================================================================
```

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

In [169]:
```
# We have high P value for weekday_wednesday  feature, so we can drop the column and fit the model again.
X_train=X_train.drop('weekday_wed', 1)
```

Model 10

In [170]:
```
# Add a constant
X_train_lm = sm.add_constant(X_train)

# Create a first fitted model
lr_9 = sm.OLS(y_train, X_train_lm).fit()
```

In [171]:
```
# Print the summary of the model
print(lr_9.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.850
Model:                            OLS   Adj. R-squared:                  0.844
Method:                 Least Squares   F-statistic:                     146.0
Date:                Tue, 06 Sep 2022   Prob (F-statistic):          8.12e-188
Time:                        15:10:08   Log-Likelihood:                -4103.7
No. Observations:                 510   AIC:                             8247.
Df Residuals:                     490   BIC:                             8332.
Df Model:                          19
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                    838.0492    420.871      1.991      0.047      11.114    1664.984
season_Spring           -481.9480    191.456     -2.517      0.012    -858.125    -105.771
season_Summer            244.9803    148.262      1.652      0.099     -46.328     536.289
season_Winter            882.3133    155.517      5.673      0.000     576.750    1187.877
mnth_dec                -426.2741    157.968     -2.698      0.007    -736.652    -115.896
mnth_feb                -333.5600    186.958     -1.784      0.075    -700.899      33.779
mnth_jan                -567.7376    185.472     -3.061      0.002    -932.155    -203.320
mnth_jul                -433.2519    157.306     -2.754      0.006    -742.330    -124.174
mnth_may                 192.4466    151.379      1.271      0.204    -104.985     489.879
mnth_nov                -420.3701    163.376     -2.573      0.010    -741.373     -99.367
mnth_sept                637.9778    145.396      4.388      0.000     352.301     923.655
weekday_mon             -224.1985     96.721     -2.318      0.021    -414.237     -34.160
weekday_tue             -238.8507     97.496     -2.450      0.015    -430.413     -47.288
weathersit_good         2195.2255    228.180      9.621      0.000    1746.894    2643.557
weathersit_moderate     1689.9652    215.616      7.838      0.000    1266.319    2113.612
yr                      2010.6007     69.662     28.862      0.000    1873.728    2147.474
holiday                 -783.7819    221.077     -3.545      0.000   -1218.157    -349.407
temp                    3915.4167    334.389     11.709      0.000    3258.404    4572.429
hum                    -1363.3692    328.842     -4.146      0.000   -2009.484    -717.254
windspeed              -1584.0730    222.667     -7.114      0.000   -2021.572   -1146.574
==============================================================================
Omnibus:                       75.657   Durbin-Watson:                   1.998
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              202.065
Skew:                          -0.735   Prob(JB):                     1.32e-44
Kurtosis:                       5.711   Cond. No.                         27.1
==============================================================================
```

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

In [172]:
```
# We have high P value for weekday_wednesday  feature, so we can drop the column and fit the model again.
X_train=X_train.drop('mnth_may', 1)
```

Model 11

In [173]:
```
# Add a constant
X_train_lm = sm.add_constant(X_train)

# Create a first fitted model
lr_10 = sm.OLS(y_train, X_train_lm).fit()
```

In [174]:
```
# Print the summary of the model
print(lr_10.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.849
Model:                            OLS   Adj. R-squared:                  0.844
Method:                 Least Squares   F-statistic:                     153.9
Date:                Tue, 06 Sep 2022   Prob (F-statistic):          1.44e-188
Time:                        15:11:01   Log-Likelihood:                -4104.5
No. Observations:                 510   AIC:                             8247.
Df Residuals:                     491   BIC:                             8327.
Df Model:                          18
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                   737.5032    413.632      1.783      0.075     -75.205    1550.211
season_Spring          -449.9637    189.915     -2.369      0.018    -823.111     -76.817
season_Summer           327.9712    133.200      2.462      0.014      66.258     589.684
season_Winter           899.9979    154.991      5.807      0.000     595.470    1204.526
mnth_dec               -415.1232    157.823     -2.630      0.009    -725.215    -105.031
mnth_feb               -321.9131    186.851     -1.723      0.086    -689.039      45.213
mnth_jan               -553.7623    185.262     -2.989      0.003    -917.766    -189.759
mnth_jul               -436.6643    157.382     -2.775      0.006    -745.889    -127.439
mnth_nov               -406.5559    163.116     -2.492      0.013    -727.048     -86.064
mnth_sept               637.9512    145.487      4.385      0.000     352.097     923.806
weekday_mon            -229.4445     96.693     -2.373      0.018    -419.428     -39.461
weekday_tue            -242.2737     97.520     -2.484      0.013    -433.882     -50.665
weathersit_good        2210.8806    227.990      9.697      0.000    1762.924    2658.838
weathersit_moderate    1704.3101    215.456      7.910      0.000    1280.981    2127.639
yr                     2006.0367     69.613     28.817      0.000    1869.260    2142.813
holiday                -791.1663    221.139     -3.578      0.000   -1225.662    -356.671
temp                   3990.1438    329.388     12.114      0.000    3342.959    4637.328
hum                   -1311.4136    326.498     -4.017      0.000   -1952.918    -669.909
windspeed             -1598.0056    222.536     -7.181      0.000   -2035.247   -1160.765
==============================================================================
Omnibus:                       74.167   Durbin-Watson:                   2.003
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              188.272
Skew:                          -0.738   Prob(JB):                     1.31e-41
Kurtosis:                       5.585   Cond. No.                         26.5
==============================================================================

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

In [175]:
```
# We have high P value for mnth_Feb feature, so we can drop the column and fit the model again.
X_train=X_train.drop('mnth_feb', 1)
```

Model 12

In [176]:
```
# Add a constant
X_train_lm = sm.add_constant(X_train)

# Create a first fitted model
lr_11 = sm.OLS(y_train, X_train_lm).fit()
```

In [177]:
```
# Print the summary of the model
print(lr_11.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.849
Model:                            OLS   Adj. R-squared:                  0.843
Method:                 Least Squares   F-statistic:                     162.1
Date:                Tue, 06 Sep 2022   Prob (F-statistic):          4.84e-189
Time:                        15:11:41   Log-Likelihood:                -4106.0
No. Observations:                 510   AIC:                             8248.
Df Residuals:                     492   BIC:                             8324.
Df Model:                          17
Covariance Type:            nonrobust
=========================================================================================
                            coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------
const                   643.3589    410.826      1.566      0.118    -163.831    1450.549
season_Spring          -538.1045    183.260     -2.936      0.003    -898.173    -178.036
season_Summer           357.9569    132.322      2.705      0.007      97.970     617.943
season_Winter           916.3103    155.011      5.911      0.000     611.745    1220.875
mnth_dec               -332.4373    150.649     -2.207      0.028    -628.432     -36.443
mnth_jan               -381.5611    156.297     -2.441      0.015    -688.653     -74.469
mnth_jul               -445.5612    157.611     -2.827      0.005    -755.236    -135.887
mnth_nov               -363.3408    161.498     -2.250      0.025    -680.652     -46.030
mnth_sept               651.6894    145.559      4.477      0.000     365.696     937.683
weekday_mon            -224.1120     96.837     -2.314      0.021    -414.376     -33.848
weekday_tue            -243.8379     97.711     -2.496      0.013    -435.820     -51.856
weathersit_good        2198.6246    228.335      9.629      0.000    1749.993    2647.256
weathersit_moderate    1694.7429    215.815      7.853      0.000    1270.711    2118.775
yr                     2002.8899     69.728     28.724      0.000    1865.888    2139.892
holiday                -812.3045    221.239     -3.672      0.000   -1246.995    -377.614
temp                   4134.2143    319.233     12.950      0.000    3506.986    4761.442
hum                   -1329.7425    326.976     -4.067      0.000   -1972.185    -687.300
windspeed             -1587.3315    222.895     -7.121      0.000   -2025.274   -1149.389
==============================================================================
Omnibus:                       69.224   Durbin-Watson:                   2.004
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              174.219
Skew:                          -0.693   Prob(JB):                     1.47e-38
Kurtosis:                       5.506   Cond. No.                         26.3
==============================================================================

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

In [228]:
```
# Calculate the VIFs again for the model
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[228]:

	Features	VIF
15	hum	26.50
14	temp	25.17
10	weathersit_good	15.78
11	weathersit_moderate	9.26
0	season_Spring	5.75
16	windspeed	4.70
2	season_Winter	4.54
1	season_Summer	3.28
12	yr	2.10
6	mnth_nov	1.89
4	mnth_jan	1.81
3	mnth_dec	1.62
5	mnth_jul	1.60
7	mnth_sept	1.43
9	weekday_tue	1.24
8	weekday_mon	1.24
13	holiday	1.06

In [260]:
```
# Checking the parameters obtained
lr_11.params
```

Out[260]:
```
const                   643.358872
season_Spring          -538.104533
season_Summer           357.956866
season_Winter           916.310285
mnth_dec               -332.437289
mnth_jan               -381.561098
mnth_jul               -445.561247
mnth_nov               -363.340785
mnth_sept               651.689354
weekday_mon            -224.112031
weekday_tue            -243.837871
weathersit_good        2198.624593
weathersit_moderate    1694.742869
yr                     2002.889895
holiday                -812.304513
temp                   4134.214260
hum                   -1329.742467
windspeed             -1587.331534
dtype: float64
```

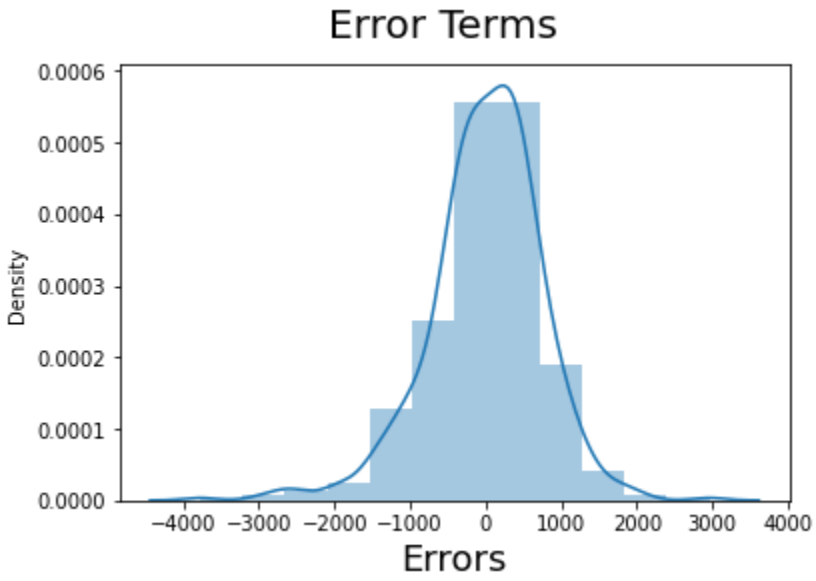
The model 12(lr_11) looks good because there are no features which is >0.05 and all the features VIF is less than 3.

Residual Analysis of the train data

In [272]:
```
y_train_pred = lr_11.predict(X_train_lm)
```

In [276]:
```
# Plot the histogram of the error terms
fig = plt.figure()
sns.distplot((y_train - y_train_pred), bins = 12)
fig.suptitle('Error Terms', fontsize = 20)
plt.xlabel('Errors', fontsize = 18)
```

Out[276]:
```
Text(0.5, 0, 'Errors')
```



By looking at the plot it shows clearlly the distplot is like normalality curve

Making Predictions Using the Final Model

In [281]:
```
num_columns=['temp','hum','windspeed']
data_test[num_columns] = scaler.transform(data_test[num_columns])
```

In [286]:
```
data_test.describe()
```

Out[286]:

	season_Spring	season_Summer	season_Winter	mnth_aug	mnth_dec	mnth_feb	mnth_jan	mnth_jul	mnth_jun	mnth_mar	...	wee
count	219.000000	219.000000	219.000000	219.000000	219.000000	219.000000	219.000000	219.000000	219.000000	219.000000	...	21
mean	0.255708	0.264840	0.232877	0.059361	0.086758	0.100457	0.077626	0.105023	0.095890	0.054795	...	
std	0.437258	0.442259	0.423633	0.236840	0.282125	0.301297	0.268194	0.307285	0.295115	0.228100	...	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
75%	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	

8 rows × 28 columns

Dividing the data into X_test and y_test

In [ ]:
```
y_test = data_test.pop('cnt')
X_test = data_test
```

In [294]:
```
X_test.shape
```

Out[294]: (219, 17)

In [296]:
```
y_test.shape
```

Out[296]:
```
(219,)
```

In [297]:
```
X_test_m = sm.add_constant(X_test)
```

In [299]:
```
X_train.columns
```

Out[299]:
```
Index(['season_Spring', 'season_Summer', 'season_Winter', 'mnth_dec',
       'mnth_jan', 'mnth_jul', 'mnth_nov', 'mnth_sept', 'weekday_mon',
       'weekday_tue', 'weathersit_good', 'weathersit_moderate', 'yr',
       'holiday', 'temp', 'hum', 'windspeed'],
      dtype='object')
```

In [302]:
```
X_test_m.columns
```

Out[302]:
```
Index(['const', 'season_Spring', 'season_Summer', 'season_Winter', 'mnth_dec',
       'mnth_jan', 'mnth_jul', 'mnth_nov', 'mnth_sept', 'weekday_mon',
       'weekday_tue', 'weathersit_good', 'weathersit_moderate', 'yr',
       'holiday', 'temp', 'hum', 'windspeed'],
      dtype='object')
```

In [191]:
```
# Creating X_test_m dataframe by dropping variables from X_test_m:

drop_cols = ['mnth_mar','mnth_jun','mnth_oct','weekday_thu','weekday_sun','workingday','weekday_sat',
             'mnth_aug','weekday_wed','mnth_may','mnth_feb']
X_test_m = X_test_m.drop(drop_cols, axis = 1)
```

In [ ]:
```

```

Model evaluation:

checking the various assumptions . check the R square and adjusted R square . Report the final model

Validating the assumption of Linear Regression Model :

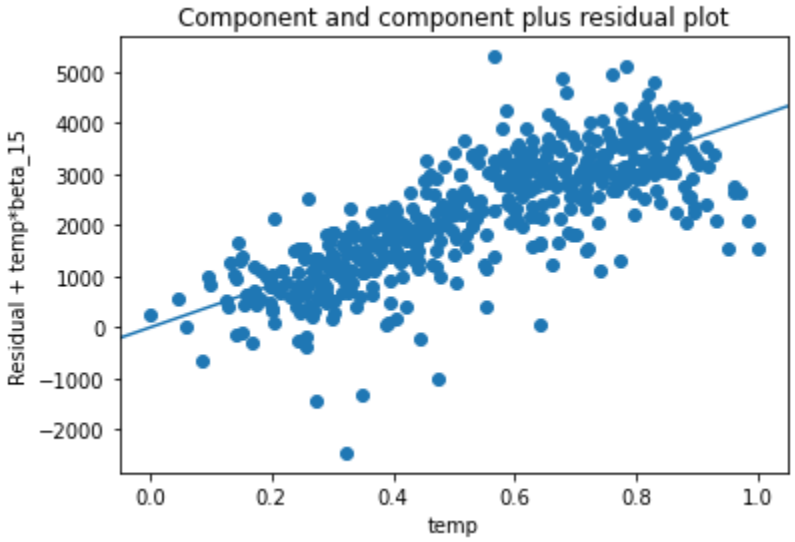
check the various assumptions

check the various assumptions

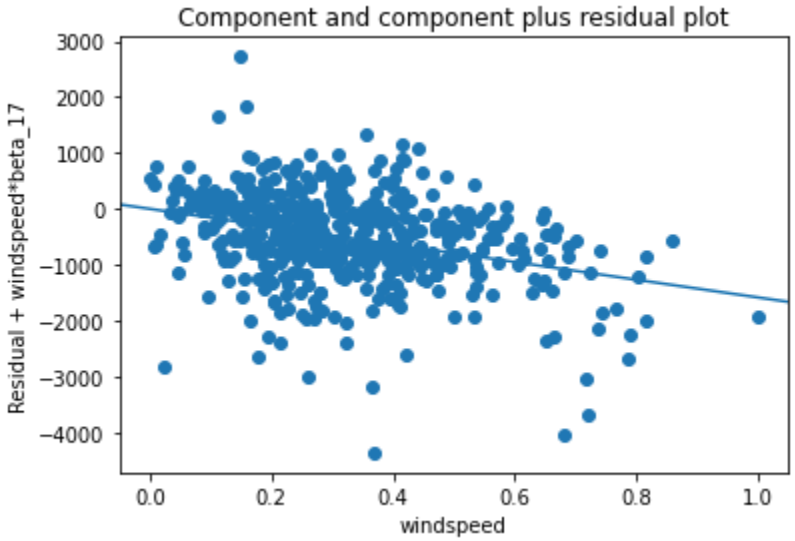
Linear Relationship . Homoscedasticity . Absence of Multicollinearity . Independence of residuals . Normality of Errors

Linear Relationship

In [304]:
```
sm.graphics.plot_ccpr(lr_11, 'temp')
plt.show()
```



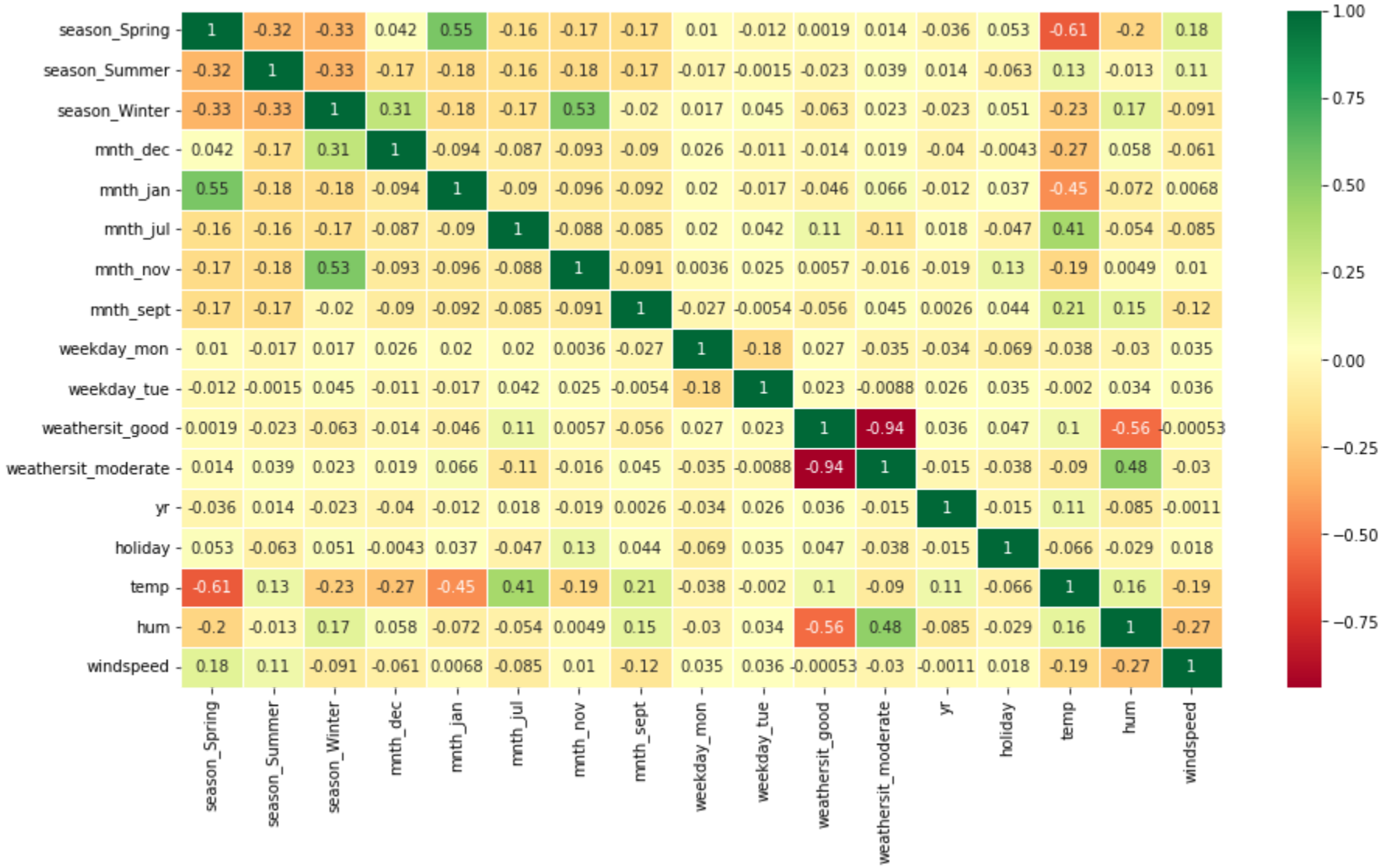
In [303]:
```
sm.graphics.plot_ccpr(lr_11, 'windspeed')
plt.show()
```



The above plots represents the relationship between the model and the predictor variables. As we can see, linearity is well preserved

Absence of Multicolinearity

In [309]:
```
# Validating Multi Colinearity
plt.figure(figsize=(15,8))
sns.heatmap(X_train.corr(),annot = True, cmap="RdYlGn",linewidth =1)
plt.show()
```



By observing the corelation matrix/heat map,we can notice some best points that are: . The season_spring and mnth_jan are positivelly high corelated with 0.55 . The season_winter and mnth_nov are positivelly corelated with 0.53

Independence of residuals

Autocorrelation refers to the fact that observations' errors are correlated. To verify that the observations are not auto-correlated, we can use the Durbin-Watson test. The test will output values between 0 and 4. The closer it is to 2, the less auto-correlation there is between the various variables.

0 – 2: positive auto-correlation 2 – 4: negative auto-correlation

In [310]:
```
print('The Durbin-Watson value for Final Model lr_11 is',round(sm.stats.stattools.durbin_watson((y_train - y_train_pred)),
```

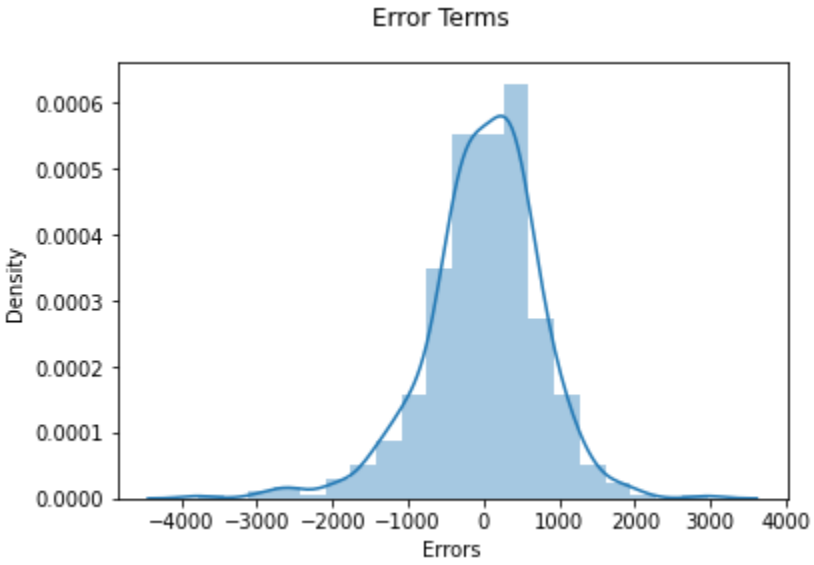
```
The Durbin-Watson value for Final Model lr_11 is 2.0045
```

There is almost no autocorrelation

Normality of error

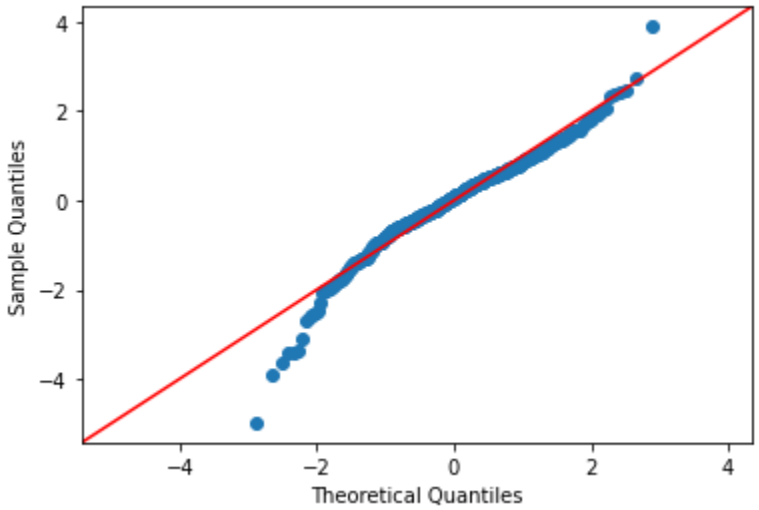
In [314]:
```
res = y_train-y_train_pred

# Plot the histogram of the error terms
fig = plt.figure()
sns.distplot((res), bins = 20)
fig.suptitle('Error Terms')
plt.xlabel('Errors')
plt.show()
```



The bins are normally distributed

In [315]:
```
sm.qqplot((y_train - y_train_pred), fit=True, line='45')
plt.show()
```



Based on the histogram, we can conclude that error terms are following a normal distribution

Making Predictions using final model

In [319]:
```
num_vars = ['temp','hum', 'windspeed']
data_test[num_vars] = scaler.transform(data_test[num_vars])
data_test.head()
```

Out[319]:

	season_Spring	season_Summer	season_Winter	mnth_aug	mnth_dec	mnth_feb	mnth_jan	mnth_jul	mnth_jun	mnth_mar	...	season	yr	mr
184	0	0	0	0	0	0	0	1	0	0	...	Fall	0	
535	0	1	0	0	0	0	0	0	1	0	...	Summer	1	j
299	0	0	1	0	0	0	0	0	0	0	...	Winter	0	
221	0	0	0	1	0	0	0	0	0	0	...	Fall	0	a
152	0	1	0	0	0	0	0	0	1	0	...	Summer	0	j

5 rows × 32 columns

In [322]:
```
data_test.describe()
```

Out[322]:

	season_Spring	season_Summer	season_Winter	mnth_aug	mnth_dec	mnth_feb	mnth_jan	mnth_jul	mnth_jun	mnth_mar	...	wee
count	219.000000	219.000000	219.000000	219.000000	219.000000	219.000000	219.000000	219.000000	219.000000	219.000000	...	21
mean	0.255708	0.264840	0.232877	0.059361	0.086758	0.100457	0.077626	0.105023	0.095890	0.054795	...	
std	0.437258	0.442259	0.423633	0.236840	0.282125	0.301297	0.268194	0.307285	0.295115	0.228100	...	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
75%	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	

8 rows × 28 columns

In [325]:
```
#Selecting the variables that were part of final model.
col1=X_train.columns

X_test=X_test[col1]

# Adding constant variable to test dataframe
X_test_lm_11 = sm.add_constant(X_test)

X_test_lm_11.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 219 entries, 184 to 72
Data columns (total 18 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   const                219 non-null    float64
 1   season_Spring        219 non-null    uint8  
 2   season_Summer        219 non-null    uint8  
 3   season_Winter        219 non-null    uint8  
 4   mnth_dec             219 non-null    uint8  
 5   mnth_jan             219 non-null    uint8  
 6   mnth_jul             219 non-null    uint8  
 7   mnth_nov             219 non-null    uint8  
 8   mnth_sept            219 non-null    uint8  
 9   weekday_mon          219 non-null    uint8  
 10  weekday_tue          219 non-null    uint8  
 11  weathersit_good      219 non-null    uint8  
 12  weathersit_moderate  219 non-null    uint8  
 13  yr                   219 non-null    int64  
 14  holiday              219 non-null    int64  
 15  temp                 219 non-null    float64
 16  hum                  219 non-null    float64
 17  windspeed            219 non-null    float64
dtypes: float64(4), int64(2), uint8(12)
memory usage: 14.5 KB
```

In [333]:
```
X_test_lm_11.columns
```

Out[333]:
```
Index(['const', 'season_Spring', 'season_Summer', 'season_Winter', 'mnth_dec',
       'mnth_jan', 'mnth_jul', 'mnth_nov', 'mnth_sept', 'weekday_mon',
       'weekday_tue', 'weathersit_good', 'weathersit_moderate', 'yr',
       'holiday', 'temp', 'hum', 'windspeed'],
      dtype='object')
```

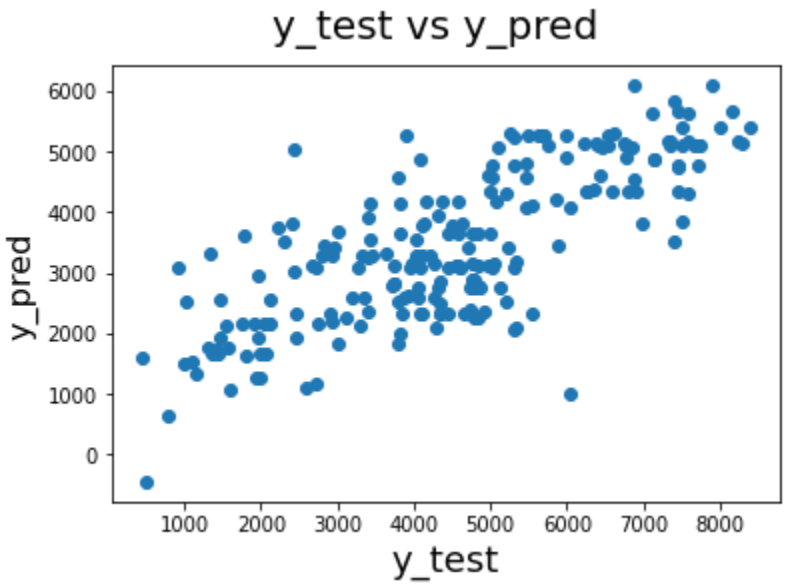
Homoscedasticity

In [ ]:

In [336]:
```
y_pred_m = lr_11.predict(X_test_m)
```

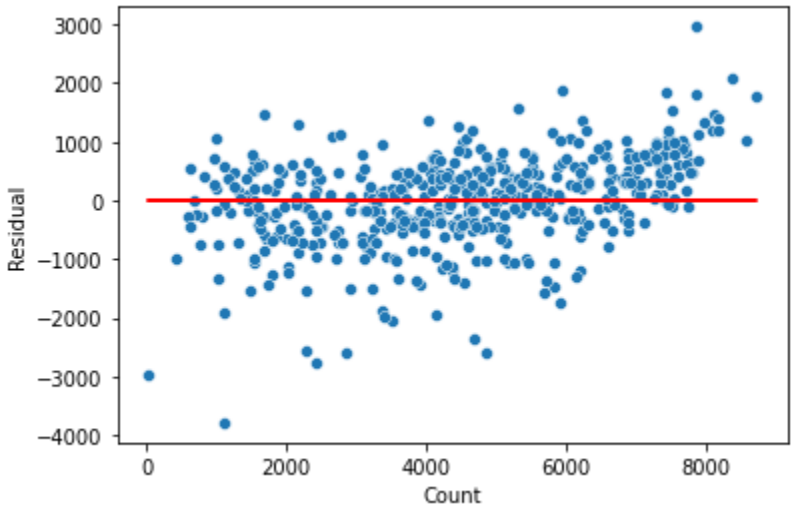
In [338]:
```
# Plotting y_test and y_pred to understand the spread
fig = plt.figure()
plt.scatter(y_test, y_pred_m)
fig.suptitle('y_test vs y_pred', fontsize = 20)              # Plot heading
plt.xlabel('y_test', fontsize = 18)                          # X-label
plt.ylabel('y_pred', fontsize = 16)
```

Out[338]: Text(0, 0.5, 'y_pred')



There is no visible pattern in residual values. . We can observe that variance of the residuals (error terms) is constant across predictions. i.e error term does not vary much as the value of the predictor variable changes.

In [339]:
```
y_train_pred = lr_11.predict(X_train_lm)
residual = y_train - y_train_pred
sns.scatterplot(y_train,residual)
plt.plot(y_train,(y_train - y_train), '-r')
plt.xlabel('Count')
plt.ylabel('Residual')
plt.show()
```



There is no visible pattern in residual values, thus homoscedacity is well preserved

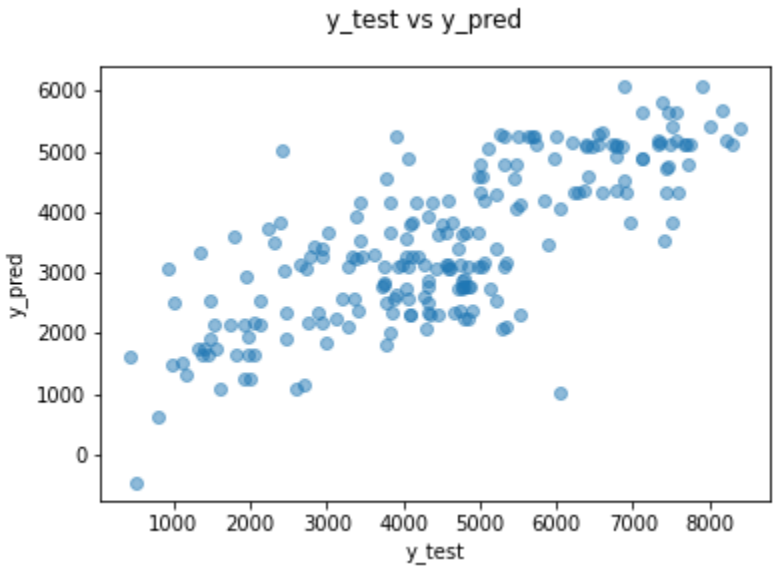
In [340]:
```
# Making predictions using the final model (lr_11)

y_pred = lr_11.predict(X_test_m)
y_pred
```

Out[340]:
```
184    1011.202684
535    5139.645067
299    3129.991601
221    2769.225243
152    3110.699225
          ...
400    3406.812944
702    5308.944804
127    2856.855469
640    5176.874548
72     2167.914159
Length: 219, dtype: float64
```

In [341]:
```
# plotting y_test and y_pred to understand the spread
fig = plt.figure()
plt.scatter(y_test, y_pred,alpha = 0.5)
fig.suptitle('y_test vs y_pred')
plt.xlabel('y_test')
plt.ylabel('y_pred')
```

Out[341]: Text(0, 0.5, 'y_pred')



R2 Value Calculation for data_test dataframe

In [342]:
```
r2 = round(r2_score(y_test, y_pred),4)
r2
```

Out[342]: 0.2583

R2 Value Calculation for data_train dataframe

In [343]:
```
print('Train R-Score: ' ,r2_score(y_train,y_train_pred))
```

Train R-Score: 0.8485060124356757

Adjusted R2 Value Calculation for data_test dataframe

In [345]:
```
# n is number of rows in test dataset
n = X_test.shape[0]

# Number of features (predictors, p) is the shape along axis 1
p = X_test.shape[1]

# We find the Adjusted R-squared using the formula
adjusted_r2 = round(1-(1-r2)*(n-1)/(n-p-1),4)
adjusted_r2
```

Out[345]: 0.1956

Adjusted R2 Value Calculation for data_train dataframe

In [346]:
```
# n is number of rows in train dataset
n = X_train.shape[0]

# Number of features (predictors, p) is the shape along axis 1
p = X_train.shape[1]

# We find the Adjusted R-squared using the formula
adjusted_r2 = round(1-(1-r2)*(n-1)/(n-p-1),4)
adjusted_r2
```

Out[346]: 0.2327

Adjusted R-squared more than 0.75 is a very good value for showing the accuracy. In some cases, Adjusted R-squared of 0.4 or more is acceptable as well

In [ ]:

Calculating Mean Absolute Error for the selected Model

In [354]:
```
MAE = round(mean_absolute_error(y_test, y_pred),4)
MAE
```

Out[354]:
```
1353.3965
```

Mean Absolute Error is 1353.3965, which indicates that the model is good. Note: . The MAE and RMSE can range from 0 to ∞. They are negatively-oriented scores: Lower values are better.

Intrepretting the Model

Reporting the final model

Let us rebuild the final model of manual + rfe approach using statsmodel to interpret it

In [358]:
```
cols = ['season_Spring', 'season_Summer', 'season_Winter', 'mnth_dec',
       'mnth_jan', 'mnth_jul', 'mnth_nov', 'mnth_sept', 'weekday_mon',
       'weekday_tue', 'weathersit_good', 'weathersit_moderate', 'yr',
       'holiday', 'temp', 'hum', 'windspeed']
lm = build_model(cols)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.849
Model:                            OLS   Adj. R-squared:                  0.843
Method:                 Least Squares   F-statistic:                     162.1
Date:                Tue, 06 Sep 2022   Prob (F-statistic):          4.84e-189
Time:                        22:34:21   Log-Likelihood:                -4106.0
No. Observations:                 510   AIC:                             8248.
Df Residuals:                     492   BIC:                             8324.
Df Model:                          17
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                   643.3589    410.826      1.566      0.118    -163.831    1450.549
season_Spring          -538.1045    183.260     -2.936      0.003    -898.173    -178.036
season_Summer           357.9569    132.322      2.705      0.007      97.970     617.943
season_Winter           916.3103    155.011      5.911      0.000     611.745    1220.875
mnth_dec               -332.4373    150.649     -2.207      0.028    -628.432     -36.443
mnth_jan               -381.5611    156.297     -2.441      0.015    -688.653     -74.469
mnth_jul               -445.5612    157.611     -2.827      0.005    -755.236    -135.887
mnth_nov               -363.3408    161.498     -2.250      0.025    -680.652     -46.030
mnth_sept               651.6894    145.559      4.477      0.000     365.696     937.683
weekday_mon            -224.1120     96.837     -2.314      0.021    -414.376     -33.848
weekday_tue            -243.8379     97.711     -2.496      0.013    -435.820     -51.856
weathersit_good        2198.6246    228.335      9.629      0.000    1749.993    2647.256
weathersit_moderate    1694.7429    215.815      7.853      0.000    1270.711    2118.775
yr                     2002.8899     69.728     28.724      0.000    1865.888    2139.892
holiday                -812.3045    221.239     -3.672      0.000   -1246.995    -377.614
temp                   4134.2143    319.233     12.950      0.000    3506.986    4761.442
hum                   -1329.7425    326.976     -4.067      0.000   -1972.185    -687.300
windspeed             -1587.3315    222.895     -7.121      0.000   -2025.274   -1149.389
==============================================================================
Omnibus:                       69.224   Durbin-Watson:                   2.004
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              174.219
Skew:                          -0.693   Prob(JB):                     1.47e-38
Kurtosis:                       5.506   Cond. No.                         26.3
==============================================================================
```

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

Model Outcome Summary

All the positive coefficients like temp,season_Summer indicate that an increase in these values will lead to an increase in the value of cnt.

Analysing the above model, the comapany should focus on the following features:

Company should focus on expanding business during Spring. . Company should focus on expanding business during September. . Based on previous data it is expected to have a boom in number of users once situation comes back to normal, compared to 2019. . There would be less bookings during Light Snow or Rain, they could probably use this time to serive the bikes without having business impact.

Significant variables to predict the demand for shared bikes

holiday . temp . hum . windspeed . Season(Spring,Summer,Winter) . months(January, July, September, November, December) . Year (2019) . Monday and Tuesday . weathersit(good,moderate)