Boom Bikes

Problem Statment:

A US bike-sharing provider BoomBikes has recently suffered considerable dips in their revenues. 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.

The company wants to know :

- Which variables are significant in predicting the demands for shred bikes.
- How well those variables describe the bike demands.

Importing required modules

In [41]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Importing statsmodel to get the detailed statistics summary of the trained model

In [42]:
```
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

Importing sklearn required modules for performing Linear Regression

In [43]:
```
import sklearn
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import r2_score, mean_squared_error,mean_absolute_error
```

In [44]:
```
#  Reading Dataset
df = pd.read_csv('day.csv')
df.head()
```

Out[44]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	regis
0	1	01-01-2018	1	0	1	0	6	0	2	14.110847	18.18125	80.5833	10.749882	331	
1	2	02-01-2018	1	0	1	0	0	0	2	14.902598	17.68695	69.6087	16.652113	131	
2	3	03-01-2018	1	0	1	0	1	1	1	8.050924	9.47025	43.7273	16.636703	120	
3	4	04-01-2018	1	0	1	0	2	1	1	8.200000	10.60610	59.0435	10.739832	108	
4	5	05-01-2018	1	0	1	0	3	1	1	9.305237	11.46350	43.6957	12.522300	82	

In [45]:
```
df.shape
```

Out[45]:
```
(730, 16)
```

In [46]:
```
df.info()
```

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

Data Cleaning

- All the columns contains 0 null values
- dteday columns needs to get converted into datatime format as well as the name of the column should also be standardised
- instant columns cantains the pre-populated index, we can drop this column as it is of no use
- Season and year need to be converted into its actuall categorical value at first

In [47]:
```
df.drop('instant', axis=1, inplace=True)
```

In [48]:
```
df.dteday = pd.to_datetime(df.dteday)
```

Maping the numerical values to its respective description

- season : season (1:spring, 2:summer, 3:fall, 4:winter)
- yr : year (0: 2018, 1:2019)
- mnth : (1:Jan,2:Feb,3:Mar,...,12:Dec)
- weekdays : days(1:'Monday',2:'Tuesday',3:'Wednesday',....,0:'Sunday')
- weathersit : climate(1:'Clear', 2:'Cloudy', 3:'LightSnow Rain', 4:'HeavysSnow Rain')

In [49]:
```
# Year
df.yr = df.yr.map({1: 2019, 0 : 2018})
# Season
df.season = df.season.map({1:'spring', 2:'summer', 3:'fall', 4:'winter'})
# Month
df.mnth = df.mnth.map({1:'Jan',2:'Feb',3:'March',4:'April',
                             5:'May',6:'June',7:'July',8:'Aug',9:'Sep',
                             10:'Oct',11:'Nov',12:'Dec'})
# weekday
df.weekday =df.weekday.map({1:'Monday',2:'Tuesday',3:'Wednesday',4:'Thrusday',
                                  5:'Friday',6:'Saturday',0:'Sunday',})
```

weathersit :

- 1: Clear, Few clouds, Partly cloudy, Partly cloudy
- 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
- 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
- 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

In [50]:
```
df.weathersit =df.weathersit.map({1:'Clear', 2:'Cloudy', 3:'LightSnow Rain', 4:'HeavysSnow Rain'})
```

In [51]:
```
print("Season :- {}".format(df.season.unique()))
print("Years :- {}".format(df.yr.unique()))
print("Months :- {}".format(df.mnth.unique()))
print("Weekday :- {}".format(df.weekday.unique()))
print("Weather Situation :- {}".format(df.weathersit.unique()))
```

```
Season :- ['spring' 'summer' 'fall' 'winter']
Years :- [2018 2019]
Months :- ['Jan' 'Feb' 'March' 'April' 'May' 'June' 'July' 'Aug' 'Sep' 'Oct' 'Nov'
 'Dec']
Weekday :- ['Saturday' 'Sunday' 'Monday' 'Tuesday' 'Wednesday' 'Thrusday' 'Friday']
Weather Situation :- ['Cloudy' 'Clear' 'LightSnow Rain']
```

Looking to the weather situation we can say that the rider never rented a boom bike during a heavy rain or during storm which was expected

In [52]:
```
df.head()
```

Out[52]:

	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registere
0	2018-01-01	spring	2018	Jan	0	Saturday	0	Cloudy	14.110847	18.18125	80.5833	10.749882	331	65
1	2018-02-01	spring	2018	Jan	0	Sunday	0	Cloudy	14.902598	17.68695	69.6087	16.652113	131	67
2	2018-03-01	spring	2018	Jan	0	Monday	1	Clear	8.050924	9.47025	43.7273	16.636703	120	122
3	2018-04-01	spring	2018	Jan	0	Tuesday	1	Clear	8.200000	10.60610	59.0435	10.739832	108	145
4	2018-05-01	spring	2018	Jan	0	Wednesday	1	Clear	9.305237	11.46350	43.6957	12.522300	82	151

Changing name of the columns as desired

In [53]:
```
df.rename(columns = {'dteday':'date','yr':'year','mnth':'month','hum':'humidity','cnt':'count'}, inplac
e = True)
```

Let see the dataset information

In [54]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 730 entries, 0 to 729
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   date        730 non-null    datetime64[ns]
 1   season      730 non-null    object        
 2   year        730 non-null    int64         
 3   month       730 non-null    object        
 4   holiday     730 non-null    int64         
 5   weekday     730 non-null    object        
 6   workingday  730 non-null    int64         
 7   weathersit  730 non-null    object        
 8   temp        730 non-null    float64       
 9   atemp       730 non-null    float64       
 10  humidity    730 non-null    float64       
 11  windspeed   730 non-null    float64       
 12  casual      730 non-null    int64         
 13  registered  730 non-null    int64         
 14  count       730 non-null    int64         
dtypes: datetime64[ns](1), float64(4), int64(6), object(4)
memory usage: 85.7+ KB
```

As we can see that the categorical columns have been changed from integer type to object type as well as the date data type has been changed to Datetime data type

In [55]:
```
# Statistical Summary
df.describe()
```

Out[55]:

	year	holiday	workingday	temp	atemp	humidity	windspeed	casual	registered	count
count	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000
mean	2018.500000	0.028767	0.683562	20.319259	23.726322	62.765175	12.763620	849.249315	3658.757534	4508.006849
std	0.500343	0.167266	0.465405	7.506729	8.150308	14.237589	5.195841	686.479875	1559.758728	1936.011647
min	2018.000000	0.000000	0.000000	2.424346	3.953480	0.000000	1.500244	2.000000	20.000000	22.000000
25%	2018.000000	0.000000	0.000000	13.811885	16.889713	52.000000	9.041650	316.250000	2502.250000	3169.750000
50%	2018.500000	0.000000	1.000000	20.465826	24.368225	62.625000	12.125325	717.000000	3664.500000	4548.500000
75%	2019.000000	0.000000	1.000000	26.880615	30.445775	72.989575	15.625589	1096.500000	4783.250000	5966.000000
max	2019.000000	1.000000	1.000000	35.328347	42.044800	97.250000	34.000021	3410.000000	6946.000000	8714.000000

Data Exploration

Do we have any repetative date in the Dataset?

In [56]:
```
df.groupby('date').agg({'date':'count'}).shape
```

Out[56]:
```
(730, 1)
```

So, the total no's of rows comes out to be 730 after grouping on the date column which is same as the nos of rows in the actual dataset so we don't have any repetative dates in our dataset

Summarizing the date column in the dataset

In [57]:
```
print('dataset starting date = {}'.format(df.date.min().date()))
print('dataset ending date = {}'.format(df.date.max().date()))
s = df.date.min().date()
e = df.date.max().date()
print("No's of days between max and min date = {}".format((e-s).days))
```

```
dataset starting date = 2018-01-01
dataset ending date = 2019-12-31
No's of days between max and min date = 729
```

In [58]:
```
total_days = df.groupby(['weathersit','workingday']).agg({'workingday':'count'})
total_days.rename(columns={'workingday':'count of days'}, inplace = True)
```

In [59]:
```
total_days.pivot_table(index=['weathersit','workingday'],
               margins=True,
               margins_name='total',  # defaults to 'All'
               aggfunc=sum)
# total_days
```

Out[59]:

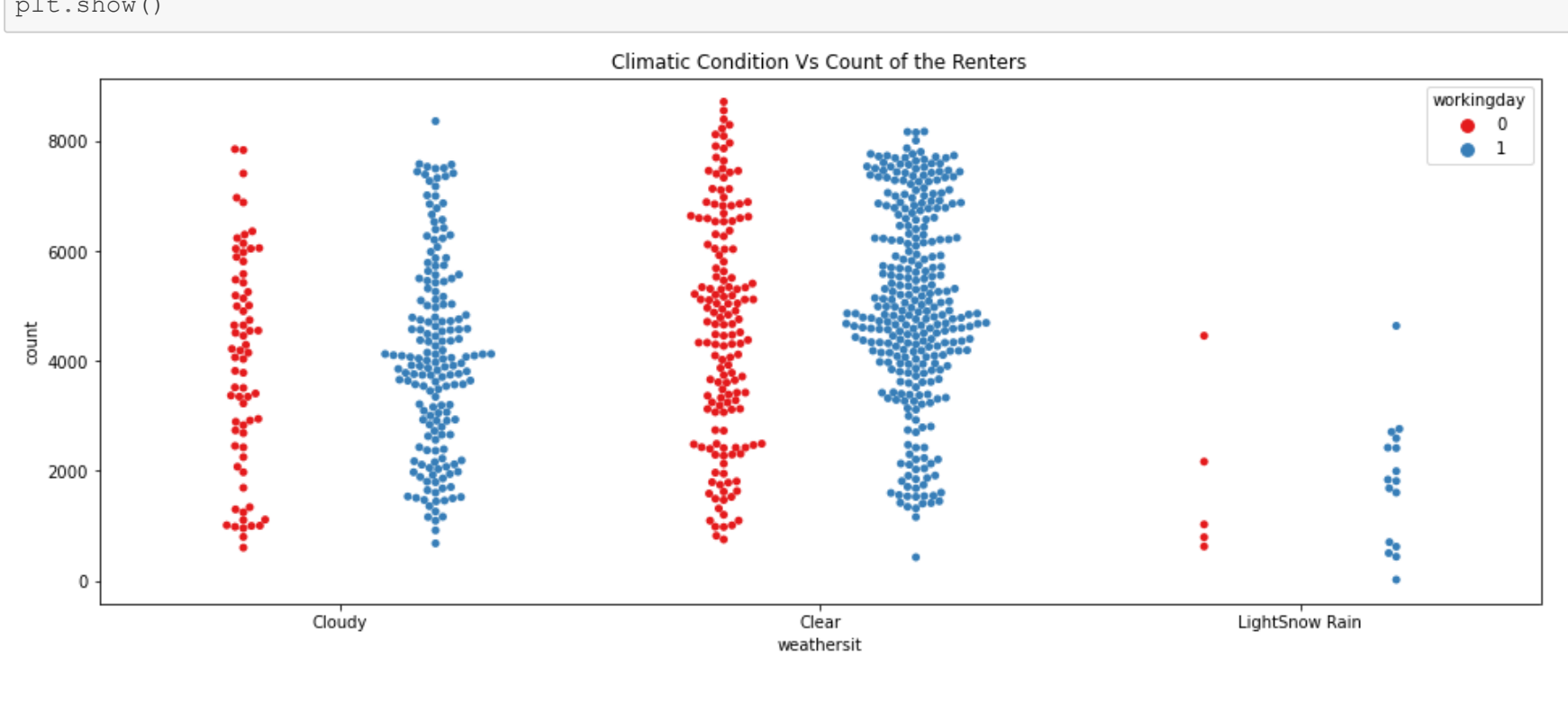
weathersit	workingday	count of days
Clear	0	156
	1	307
Cloudy	0	70
	1	176
LightSnow Rain	0	5
	1	16
total		730

Earlier while looking into the unique values of the weather situation we stated that the riders don't use the bike rental service during storm or heavy Rain which seems to be right as expected but on looking to the total no's of days on which the bike has been rented we can say that the firm never faced the Heavy Rain or storm situation

So, we don't have enough data for the heavy rain situation to conclude any statement but ideally the providers should avoid renting out their bikes during this situation also take an additional steps to face this situation in the future

lets see the spread of the renters data across different weather situation using bee swarm plot

In [60]:
```
plt.figure(figsize=(16,6))
sns.swarmplot(x="weathersit", y="count", hue="workingday",
              data=df, palette="Set1", dodge=True)
plt.title('Climatic Condition Vs Count of the Renters')
plt.show()
```



From above plot we can state that during clear weather the numbers of riders renting the Bikes are more as compare to the Mist/Cloudy weather

- we can see that the riders use to access the services more during Mist/cloudy or Clear weather
- during light snow/light rain and the user won't prefer to avail the services if it's non-working day but during working days they do have rented the bike a very few nos of times

Now let's analysed on which week day the sevices has been used more with in combination of weathear

In [61]:
```
#  Though we have seen the spread of the data lets also explore its density
df.weathersit.value_counts(normalize=True)
```

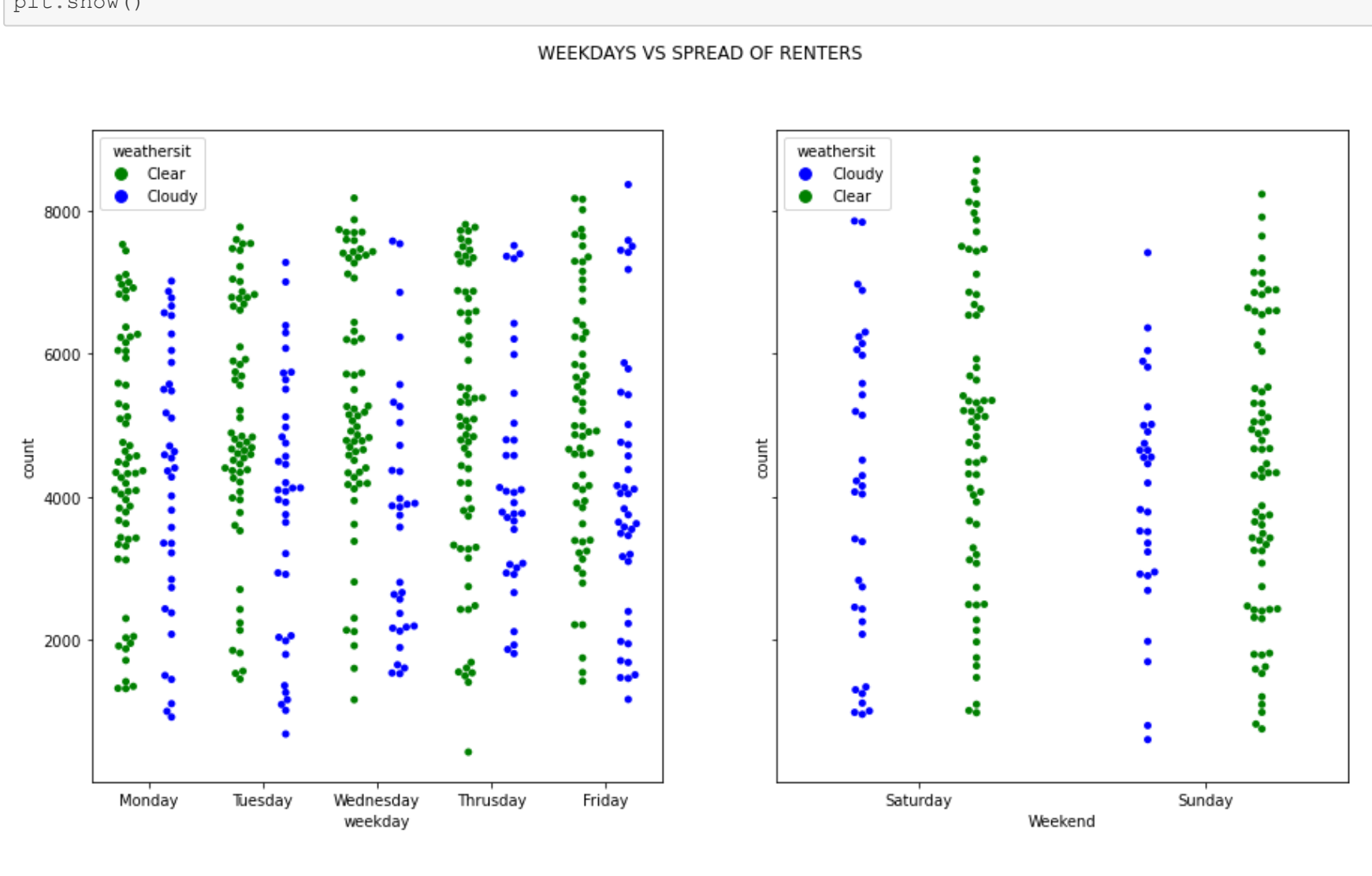
Out[61]:
```
Clear             0.634247
Cloudy            0.336986
LightSnow Rain    0.028767
Name: weathersit, dtype: float64
```

Though the spread across the weekdays was seemed to be the same but the density and user preference lies towards Clear weather

In [62]:
```
df.weekday.value_counts(normalize=True).apply(lambda x:str(round(x*100,2))+'%')
```

Out[62]:
```
Monday       14.38%
Sunday       14.38%
Saturday     14.38%
Friday       14.25%
Thrusday     14.25%
Tuesday      14.25%
Wednesday    14.11%
Name: weekday, dtype: object
```

In [63]:
```
%matplotlib inline

fig, axes = plt.subplots(1, 2,sharey=True,figsize=(15,8))
plt.suptitle('WEEKDAYS VS SPREAD OF RENTERS')


sns.swarmplot(x="weekday", y="count", hue='weathersit',
              data=df[(df['weathersit']!= 'LightSnow Rain') & (~df['weekday'].isin(['Saturday', 'S
unday']))],
           palette=['Green','Blue'], dodge=True,ax=axes[0])


sns.swarmplot(x="weekday", y="count", hue='weathersit',
              data=df[(df['weathersit']!= 'LightSnow Rain') & (df.weekday.isin (['Saturday', 'Sund
ay']))],
           palette=['Blue','Green'], dodge=True ,ax=axes[1])
plt.xlabel('Weekend')

plt.show()
```



There is not significant increase in the numbers but if we compare the weekends as compare to any days of the weekdays excluding friday(day=5) the numbers of the user renting the bikes have been increase and among weekdays the friday is the most populous day irrespective of the weather

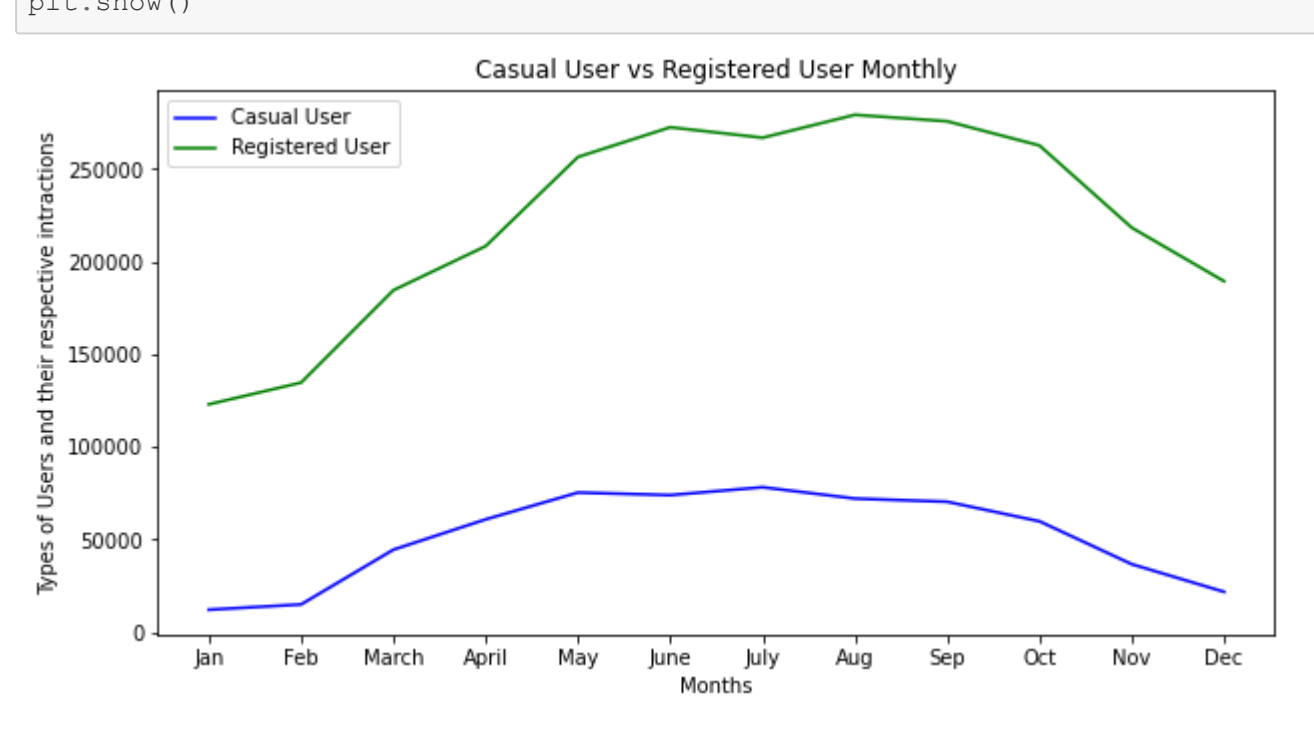
The count variable has been derived by summing up the registered users and the casual users. So, lets see the contribution of each class of users

In [64]:
```
rented_user = df.groupby('month', as_index =False,sort=False).agg({'count':'sum','registered':'sum','ca
sual':'sum'})
rented_user
```

Out[64]:

	month	count	registered	casual
0	Jan	134933	122891	12042
1	Feb	149518	134620	14898
2	March	228920	184476	44444
3	April	269094	208292	60802
4	May	331686	256401	75285
5	June	346342	272436	73906
6	July	344948	266791	78157
7	Aug	351194	279155	72039
8	Sep	345991	275668	70323
9	Oct	322352	262592	59760
10	Nov	254831	218228	36603
11	Dec	211036	189343	21693

In [65]:
```
plt.figure(figsize=(10,5))
sns.lineplot(rented_user.month,rented_user.casual,color='b',label='Casual User',sort=False)
sns.lineplot(rented_user.month,rented_user.registered,color='g',dashes='-',label='Registered User',sort
=False)
plt.ylabel('Types of Users and their respective intractions')
plt.xlabel('Months')
plt.title('Casual User vs Registered User Monthly')
plt.show()
```



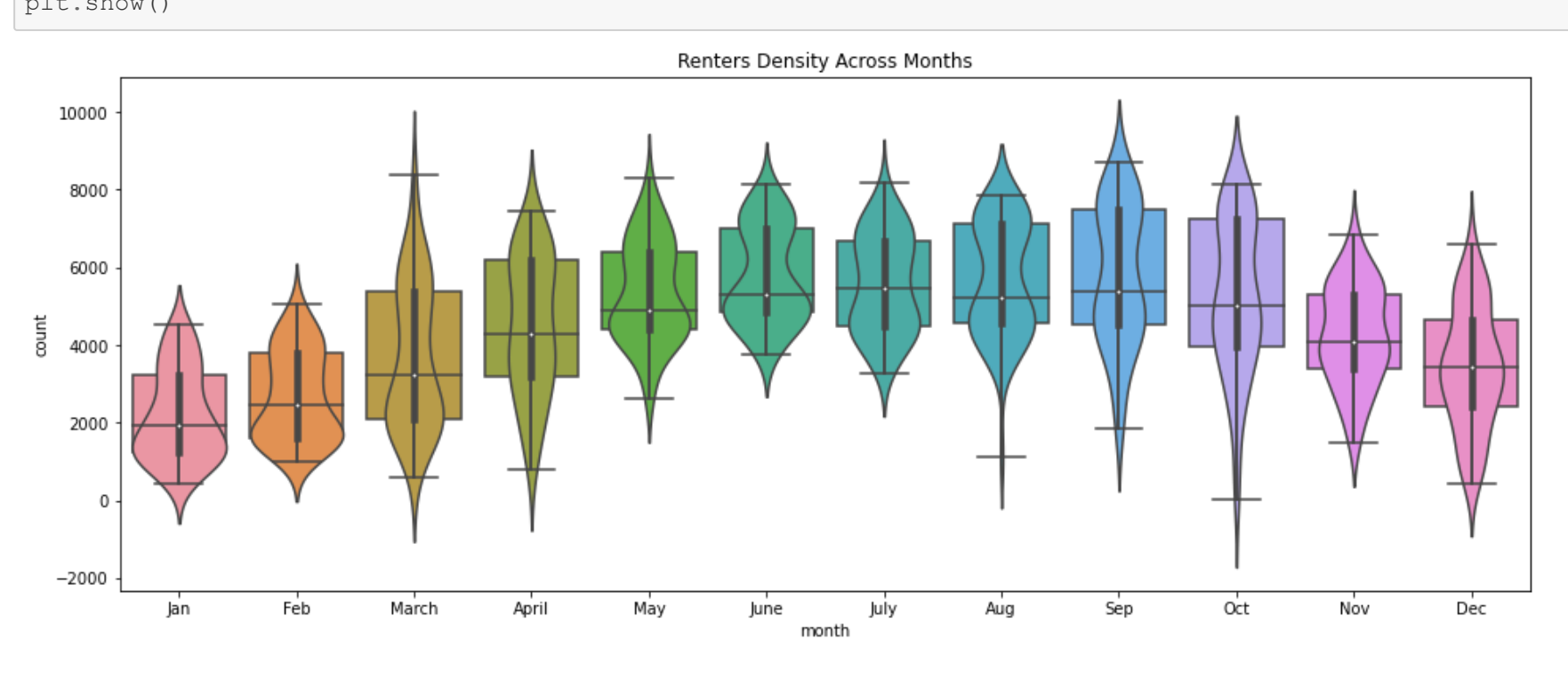
Clearly the the Booms Bike get rented by the registered user more frequently as compared to the their casual users on the same note, the cnt of casual user is also good in numbers and can contribute or affect the firms revenue directly.

Also, the firm can increase the margin of price for the non-registered user to earn the extra penny

The Box plot unlike the Boxes

On which month the wind is blowing and in what direction my bees are flying?

In [66]:
```
plt.figure(figsize=(16,6))
# sns.lineplot(x="mnth", y="cnt", data=df, color='red')
sns.violinplot(x="month", y="count", data=df)
sns.boxplot(x="month", y="count", data=df)
plt.title("Renters Density Across Months")
plt.show()
```



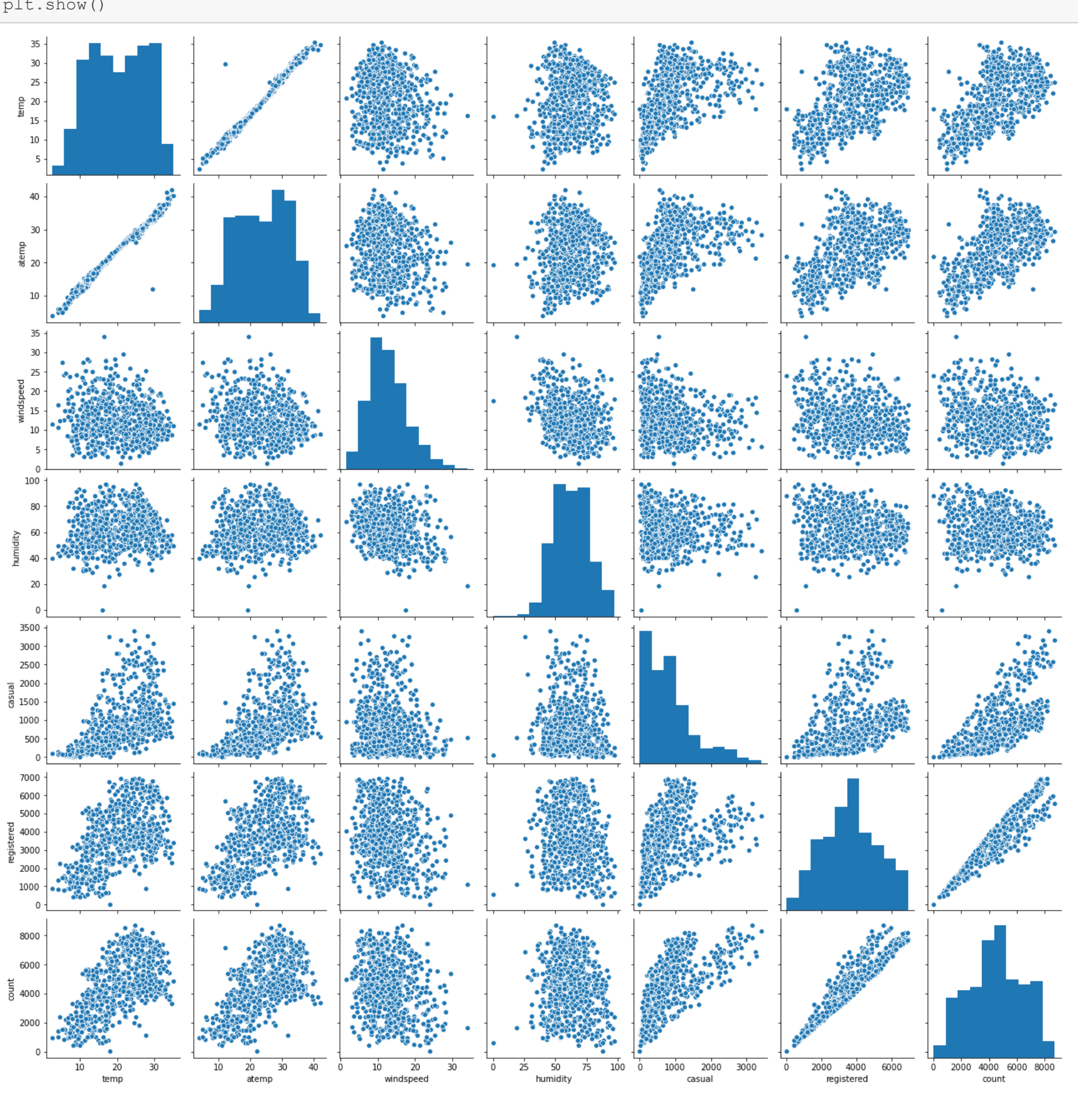
`Consider the the renter as bees and tail representing the lower density of the renters and the head showing the higher density of the renters and more stable on the flowers bed(June, July)....`

So, basically the users to take off in the month of Jan and attains the few heights in Feb and in March the firm suffers from bad weather situation and observed the varied spread of the renters but still able to manage the higher heights than its previous months and takes an entire april to stablised the count of renters and to change its direction and finally keeps on moving into the upward direction till the month

May to Oct has high median while in Jan it is lowest then it is gradually increasing till july then again going down in winter months. It means cnt increase in higher temperature months

Exploring the correlation between different numerical features

In [67]:
```
#Pairplot for numeric variables
sns.pairplot(df, vars=["temp",'atemp', 'windspeed',"humidity",'casual','registered','count'])
plt.show()
```

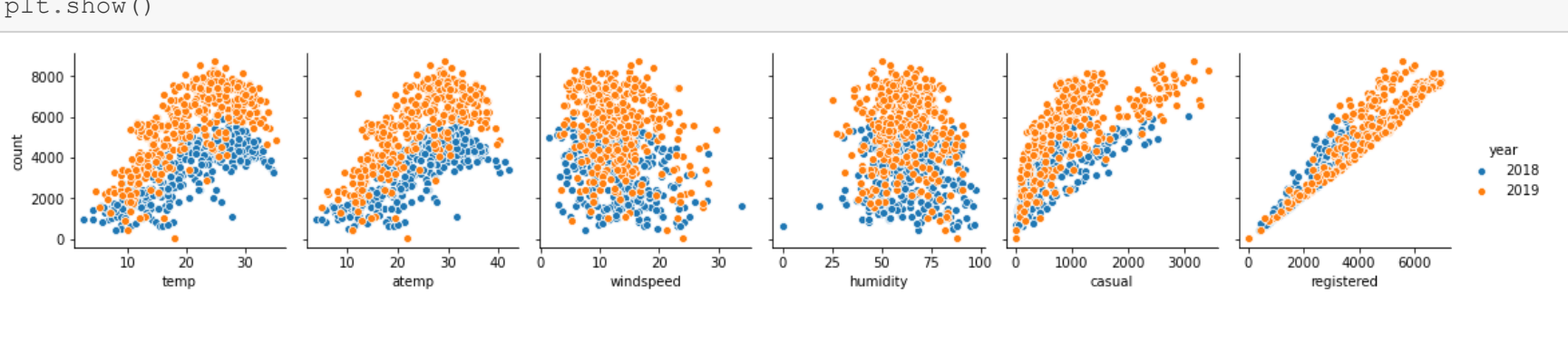


temp and atemp are highly correlated Casual, registered have good correlation with count variable as the count gets derived on summing the casual and registered Users value

Let's analyzed for the Year variable and how the behaviour gets changed across it

Pairplot for numeric variables

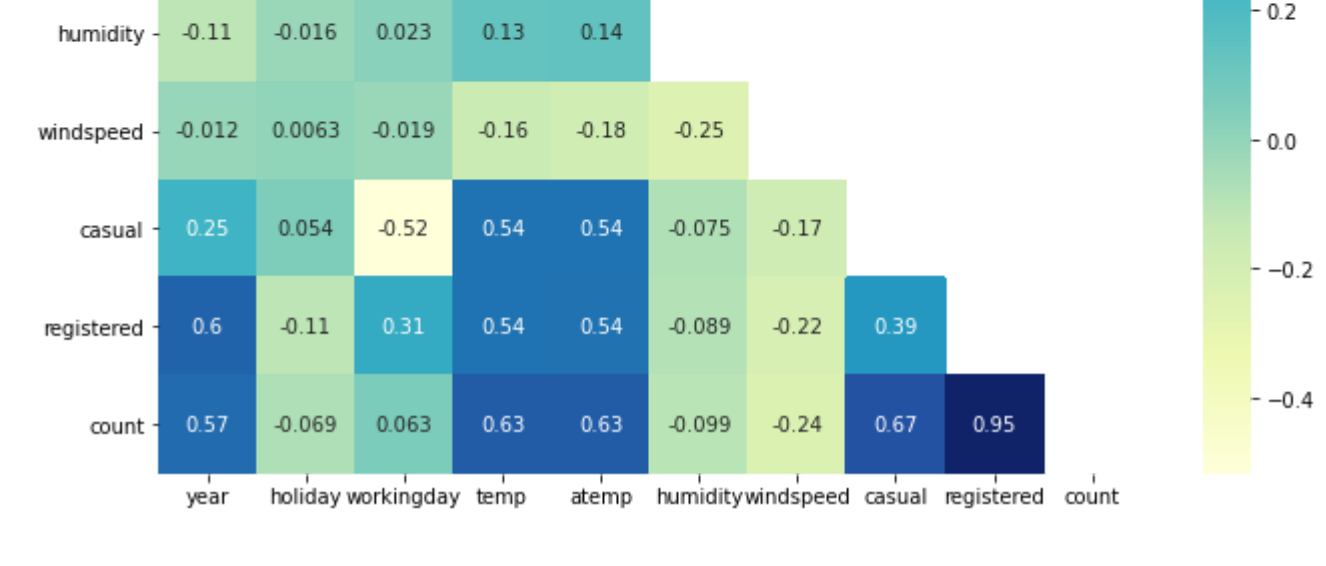
In [68]:
```
sns.pairplot(df, x_vars=["temp",'atemp', 'windspeed',"humidity",'casual','registered'],y_vars=['count'
],hue='year')
plt.show()
```



From above pairplot we can infer that the no's of casual users are getting increased as compared to the previous year while the registered user show slightly downward trend as compared to the previous year

Correlation Heat Map

In [69]:
```
corr = df.corr()
# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap='YlGnBu', annot = True)

plt.show()
```



Inference

- `temp and atemp` both have the same correlation coefficient as well as they are `highly correlated` to each other. So, it need to analyzed while building model and need to drop one variable while building the model to avoid `multicolinearity` using `VIF and p-value`
- On the other hand, `year and casual users, and registered users are positively correlated to the count variable`
- Also, the target variable gets derived from the casual and registered feature so moving forward we will not consider these columns

Drop the unnecessary variables from the dataset

We can see the dataset has some variables that are not required. We can drop `date, casual, registered, atemp`

In [70]:
```
#drop unnecessary columns
df=df.drop(['date','casual', 'registered','atemp'], axis=1)
df.head()
```

Out[70]:

	season	year	month	holiday	weekday	workingday	weathersit	temp	humidity	windspeed	count
0	spring	2018	Jan	0	Saturday	0	Cloudy	14.110847	80.5833	10.749882	985
1	spring	2018	Jan	0	Sunday	0	Cloudy	14.902598	69.6087	16.652113	801
2	spring	2018	Jan	0	Monday	1	Clear	8.050924	43.7273	16.636703	1349
3	spring	2018	Jan	0	Tuesday	1	Clear	8.200000	59.0435	10.739832	1562
4	spring	2018	Jan	0	Wednesday	1	Clear	9.305237	43.6957	12.522300	1600

Model Building

1. Data Preparation

In [71]:
```
print('The columns which we have are')
print(list(df.columns))
```

```
The columns which we have are
['season', 'year', 'month', 'holiday', 'weekday', 'workingday', 'weathersit', 'temp', 'humidity', 'wi
ndspeed', 'count']
```

1.1 Introduction of Dummy Variables

- columns **['season', 'Year', 'month', 'weekday','weathersit']** contains a categorical string variable
- We need to convert these variables into integers according to their levels
- For this, we will use something called `dummy variables`.

In [72]:
```
print('Checking the levels of each columns')
print(df.select_dtypes(include='object').nunique().sort_values())
```

```
Checking the levels of each columns
weathersit     3
season         4
weekday        7
month         12
dtype: int64
```

Total Dummy Variables will be created as : ((3-1)+(4-1)+(7-1)+(12-1)) = 22

In [73]:
```
# Season
print('storing season dummy variables into cat1')
cat1 = pd.get_dummies(df.season, drop_first = True)
cat1.head()
```

```
storing season dummy variables into cat1
```

Out[73]:

	spring	summer	winter
0	1	0	0
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0

In [74]:
```
# month
print('storing month dummy variables into cat2')
cat2 = pd.get_dummies(df.month,drop_first=True)
cat2.head()
```

```
storing month dummy variables into cat2
```

Out[74]:

	Aug	Dec	Feb	Jan	July	June	March	May	Nov	Oct	Sep
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0

In [75]:
```
# weekday
print('    storing weekday dummy variables into cat3')
cat3 = pd.get_dummies(df.weekday,drop_first=True)
cat3.head()
```

```
    storing weekday dummy variables into cat3
```

Out[75]:

	Monday	Saturday	Sunday	Thrusday	Tuesday	Wednesday
0	0	1	0	0	0	0
1	0	0	1	0	0	0
2	1	0	0	0	0	0
3	0	0	0	0	1	0
4	0	0	0	0	0	1

In [76]:
```
# weathersit
print('storing weathersit dummy variables into cat4')
cat4 = pd.get_dummies(df.weathersit,drop_first=True)
cat4.head()
```

```
storing weathersit dummy variables into cat4
```

Out[76]:

	Cloudy	LightSnow Rain
0	1	0
1	1	0
2	0	0
3	0	0
4	0	0

In [77]:
```
print('original dataset')
```

```
original dataset
```

In [78]:
```
#year
df.year = df.year.map({2018:0,2019:1})
df.head()
```

Out[78]:

	season	year	month	holiday	weekday	workingday	weathersit	temp	humidity	windspeed	count
0	spring	0	Jan	0	Saturday	0	Cloudy	14.110847	80.5833	10.749882	985
1	spring	0	Jan	0	Sunday	0	Cloudy	14.902598	69.6087	16.652113	801
2	spring	0	Jan	0	Monday	1	Clear	8.050924	43.7273	16.636703	1349
3	spring	0	Jan	0	Tuesday	1	Clear	8.200000	59.0435	10.739832	1562
4	spring	0	Jan	0	Wednesday	1	Clear	9.305237	43.6957	12.522300	1600

```
In [293]: feature.append('temp')
          feature.remove('July')
          get_model(6,feature,X_train)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  count   R-squared:                       0.839
Model:                            OLS   Adj. R-squared:                  0.836
Method:                 Least Squares   F-statistic:                     216.5
Date:                Tue, 29 Sep 2020   Prob (F-statistic):         1.48e-188
Time:                        23:35:17   Log-Likelihood:                -4120.9
No. Observations:                 510   AIC:                             8268.
Df Residuals:                     497   BIC:                             8323.
Df Model:                          12
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const          1312.2548    267.028      4.914      0.000     787.612    1836.897
year           2035.3014     70.846     28.729      0.000    1896.107    2174.496
holiday        -478.8258    239.645     -1.998      0.046    -949.669      -7.983
workingday      413.2153    102.447      4.033      0.000     211.933     614.498
windspeed     -1297.0616    216.282     -5.997      0.000   -1722.001    -872.122
spring         -469.1358    178.254     -2.632      0.009    -819.360    -118.911
summer          534.2226    122.036      4.378      0.000     294.452     773.993
winter          853.3356    143.152      5.961      0.000     572.079    1134.592
Sep             776.3950    139.660      5.559      0.000     501.998    1050.792
Saturday        509.6778    128.903      3.954      0.000     256.416     762.939
Cloudy         -714.4683     75.442     -9.470      0.000    -862.693    -566.243
LightSnow Rain -2532.6169    212.941    -11.894      0.000   -2950.993   -2114.241
temp           4165.8605    282.924     14.724      0.000    3609.987    4721.734
==============================================================================
Omnibus:                       71.158   Durbin-Watson:                   2.092
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              170.059
Skew:                          -0.729   Prob(JB):                     1.18e-37
Kurtosis:                       5.424   Cond. No.                         19.5
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
-------------------------------------------------------------------------------
                                      VIF
===============================================================================
         features   VIF
0            temp  5.70
1      workingday  5.20
2       windspeed  4.65
3          spring  2.40
4            year  2.07
5          summer  2.00
6        Saturday  1.96
7          winter  1.83
8          Cloudy  1.56
9             Sep  1.24
10        holiday  1.17
11  LightSnow Rain  1.08
```

Adjusted R2_score changes by 0.002 (0.2%), and wont affect our model

Now, we will eliminate the holiday column as all other variables except holiday has p-value of 0

Model7

```
In [294]: feature.remove('holiday')
          get_model(7,feature,X_train)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  count   R-squared:                       0.838
Model:                            OLS   Adj. R-squared:                  0.835
Method:                 Least Squares   F-statistic:                     234.4
Date:                Tue, 29 Sep 2020   Prob (F-statistic):         6.86e-189
Time:                        23:35:17   Log-Likelihood:                -4123.0
No. Observations:                 510   AIC:                             8270.
Df Residuals:                     498   BIC:                             8321.
Df Model:                          11
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const          1243.5563    265.599      4.682      0.000     721.723    1765.390
year           2036.7262     71.055     28.664      0.000    1897.122    2176.330
workingday      484.0640     96.403      5.021      0.000     294.657     673.471
windspeed     -1304.3717    216.899     -6.014      0.000   -1730.522    -878.221
spring         -478.9565    178.721     -2.680      0.008    -830.096    -127.817
summer          534.8955    122.402      4.370      0.000     294.408     775.383
winter          844.6349    143.515      5.885      0.000     562.666    1126.603
Sep             757.7232    139.765      5.421      0.000     483.121    1032.325
Saturday        581.4210    124.172      4.682      0.000     337.455     825.387
Cloudy         -710.3641     75.640     -9.391      0.000    -858.978    -561.750
LightSnow Rain -2522.3554    213.518    -11.813      0.000   -2941.862   -2102.849
temp           4171.9183    283.756     14.702      0.000    3614.412    4729.425
==============================================================================
Omnibus:                       76.386   Durbin-Watson:                   2.073
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              190.041
Skew:                          -0.765   Prob(JB):                     5.41e-42
Kurtosis:                       5.569   Cond. No.                         19.4
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
-------------------------------------------------------------------------------
                                      VIF
===============================================================================
         features   VIF
0            temp  5.48
1      workingday  4.65
2       windspeed  4.63
3          spring  2.30
4            year  2.07
5          summer  1.99
6        Saturday  1.83
7          winter  1.77
8          Cloudy  1.56
9             Sep  1.23
10  LightSnow Rain  1.08
```

The Adjusted R-Square gets reduced by 0.001 or 0.1% which again won't affect our model also, all other variable have high statistical significance

Now, if we look to correlation coefficient we find a negative coefficient among all the features windspeed and LightSnow Rain have high negative coefficient and also th VIF value for the windspeed is high.

So. let's try dropping windspeed an evaluate the model

Model8

```
In [295]: feature.remove('windspeed')
          get_model(8,feature,X_train)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  count   R-squared:                       0.826
Model:                            OLS   Adj. R-squared:                  0.823
Method:                 Least Squares   F-statistic:                     237.5
Date:                Tue, 29 Sep 2020   Prob (F-statistic):         1.55e-182
Time:                        23:35:18   Log-Likelihood:                -4140.8
No. Observations:                 510   AIC:                             8304.
Df Residuals:                     499   BIC:                             8350.
Df Model:                          10
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const           805.3959    264.256      3.048      0.002     286.203    1324.588
year           2027.6600     73.499     27.587      0.000    1883.254    2172.066
workingday      493.4235     99.729      4.948      0.000     297.483     689.364
spring         -562.0888    184.357     -3.049      0.002    -924.300    -199.877
summer          448.6179    125.768      3.567      0.000     201.517     695.719
winter          854.9460    148.475      5.758      0.000     563.234    1146.658
Sep             795.5240    144.460      5.507      0.000     511.700    1079.348
Saturday        562.5839    128.432      4.380      0.000     310.250     814.918
Cloudy         -695.0068     78.216     -8.886      0.000    -848.679    -541.334
LightSnow Rain -2651.4976    219.793    -12.064      0.000   -3083.331   -2219.664
temp           4274.7202    293.051     14.587      0.000    3698.955    4850.486
==============================================================================
Omnibus:                       77.818   Durbin-Watson:                   2.057
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              209.021
Skew:                          -0.753   Prob(JB):                     4.09e-46
Kurtosis:                       5.751   Cond. No.                         18.9
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
-------------------------------------------------------------------------------
                                      VIF
===============================================================================
        features   VIF
0           temp  4.72
1     workingday  4.63
2           year  2.07
3       Saturday  1.81
4         summer  1.74
5         spring  1.70
6         winter  1.59
7         Cloudy  1.56
8            Sep  1.23
9  LightSnow Rain  1.07
```

On removing Windspeed

`R-square get change from 0.838 to 0.826 with the difference of 0.012 (1.2%)

Adj. R-square get change from 0.835 to 0.823 with the difference of 0.012(1.2%)`

So, now the p-value and the VIF of the features are within the acceptable range

- A low p-value (< 0.05) indicates that we can reject the null hypothesis.
- With VIF values less than 5, we are in good shape, and can proceed with our regression without having multicolinearity
- We have the R-square value of 0.826 or 82.6%
- The adjusted R-squared came to be 0.823 or 82.3%. The low differences between R-square and the adj. R-square signifies that all the variables in the model are describing our data points

Also, let's check by removing temp and keeping windspeed into the model and

Model9

```
In [296]: feature.append('windspeed')
          feature.remove('temp')
          get_model(9,feature,X_train)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  count   R-squared:                       0.768
Model:                            OLS   Adj. R-squared:                  0.763
Method:                 Least Squares   F-statistic:                     165.0
Date:                Tue, 29 Sep 2020   Prob (F-statistic):         3.40e-151
Time:                        23:35:19   Log-Likelihood:                -4214.9
No. Observations:                 510   AIC:                             8452.
Df Residuals:                     499   BIC:                             8498.
Df Model:                          10
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const          4630.4923    158.146     29.280      0.000    4319.779    4941.206
year           2150.9308     84.495     25.456      0.000    1984.921    2316.941
workingday      491.7071    115.328      4.264      0.000     265.120     718.295
spring        -2580.2806    128.370    -20.100      0.000   -2832.494   -2328.068
summer         -360.9779    126.998     -2.842      0.005    -610.495    -111.461
winter         -634.8625    122.413     -5.186      0.000    -875.370    -394.355
Sep             627.5042    166.868      3.760      0.000     299.653     955.355
Saturday        556.3684    148.536      3.746      0.000     264.535     848.201
Cloudy         -800.7040     90.191     -8.878      0.000    -977.906    -623.502
LightSnow Rain -2617.7535    255.318    -10.253      0.000   -3119.385   -2116.122
windspeed     -1496.4872    259.011     -5.778      0.000   -2005.373    -987.602
==============================================================================
Omnibus:                       32.745   Durbin-Watson:                   2.016
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               70.965
Skew:                          -0.356   Prob(JB):                     3.89e-16
Kurtosis:                       4.683   Cond. No.                         10.0
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
-------------------------------------------------------------------------------
                                      VIF
===============================================================================
        features   VIF
0      windspeed  3.99
1     workingday  3.29
2         spring  2.00
3         summer  1.99
4           year  1.88
5         winter  1.73
6       Saturday  1.56
7         Cloudy  1.55
8            Sep  1.18
9  LightSnow Rain  1.08
```

As we can see, for the above model on removing temp variable and keeping windspped leds to huge drop in R-Square and adj. R-square which is about (0.5.8%) and (10 %) respectively.

```
In [297]: feature.append('temp')
          feature.remove('windspeed')
          get_model(8,feature,X_train)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  count   R-squared:                       0.826
Model:                            OLS   Adj. R-squared:                  0.823
Method:                 Least Squares   F-statistic:                     237.5
Date:                Tue, 29 Sep 2020   Prob (F-statistic):         1.55e-182
Time:                        23:35:19   Log-Likelihood:                -4140.8
No. Observations:                 510   AIC:                             8304.
Df Residuals:                     499   BIC:                             8350.
Df Model:                          10
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const           805.3959    264.256      3.048      0.002     286.203    1324.588
year           2027.6600     73.499     27.587      0.000    1883.254    2172.066
workingday      493.4235     99.729      4.948      0.000     297.483     689.364
spring         -562.0888    184.357     -3.049      0.002    -924.300    -199.877
summer          448.6179    125.768      3.567      0.000     201.517     695.719
winter          854.9460    148.475      5.758      0.000     563.234    1146.658
Sep             795.5240    144.460      5.507      0.000     511.700    1079.348
Saturday        562.5839    128.432      4.380      0.000     310.250     814.918
Cloudy         -695.0068     78.216     -8.886      0.000    -848.679    -541.334
LightSnow Rain -2651.4976    219.793    -12.064      0.000   -3083.331   -2219.664
temp           4274.7202    293.051     14.587      0.000    3698.955    4850.486
==============================================================================
Omnibus:                       77.818   Durbin-Watson:                   2.057
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              209.021
Skew:                          -0.753   Prob(JB):                     4.09e-46
Kurtosis:                       5.751   Cond. No.                         18.9
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
-------------------------------------------------------------------------------
                                      VIF
===============================================================================
        features   VIF
0           temp  4.72
1     workingday  4.63
2           year  2.07
3       Saturday  1.81
4         summer  1.74
5         spring  1.70
6         winter  1.59
7         Cloudy  1.56
8            Sep  1.23
9  LightSnow Rain  1.07
```

We will move with the model8 as our final model

y_pred = 0.0901 + year *(0.2333)* + *workingday* (0.0568) + spring *(-0.0647)* + *summer* (0.0516) + winter *(0.0984)* + *Sep* (0.0915) + Saturday *(0.0647)* + *Cloudy* (-0.0800) + LightSnow Rain *(-0.3051)* + *temp* (0.4918)

R-Square: 0.826

Adj. R-Square: 0.823

Residual Analysis of the train data

So, now to check if the error terms are also normally distributed (which is infact, one of the major assumptions of linear regression), let's plot the histogram of the error terms and see what it looks like.

```
In [298]: # residual = y_pred - y_train
          # y_train = y_train.reshape(y_train.shape[0])
```

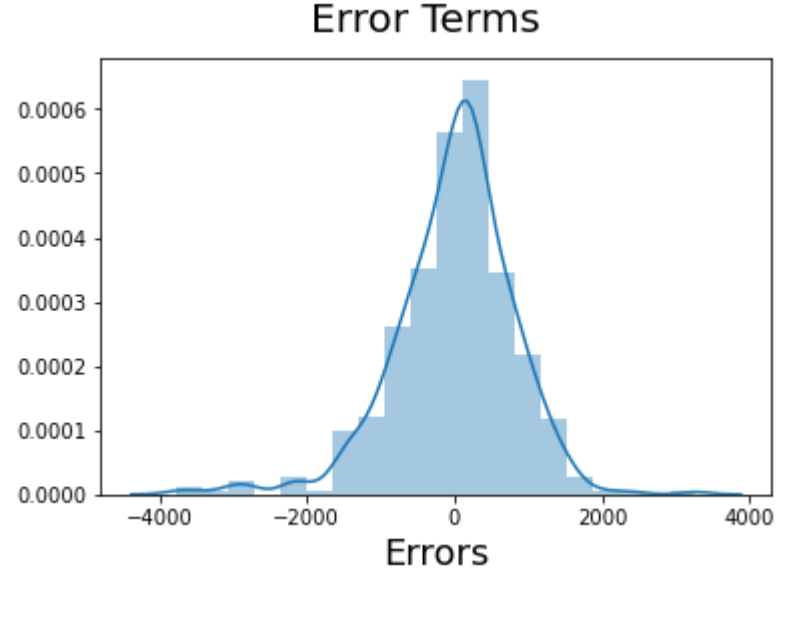
```
In [299]: y_train.shape
Out[299]: (510,)
```

```
In [300]: X_train_sm1.head(3)
Out[300]:
```

	const	year	workingday	spring	summer	winter	Sep	Saturday	Cloudy	LightSnow Rain	temp
653	1.0	1	1	0	0	1	0	0	0	0	0.509887
576	1.0	1	1	0	0	0	0	0	0	0	0.815169
426	1.0	1	0	1	0	0	0	1	1	0	0.442393

Normal distribution of the Error term around mean 0

```
In [301]: y_pred = lr.predict(X_train_sm1)

          fig = plt.figure()
          sns.distplot((y_train - y_pred), bins = 20)
          fig.suptitle('Error Terms', fontsize = 20)                  # Plot heading
          plt.xlabel('Errors', fontsize = 18)
          plt.show()
```

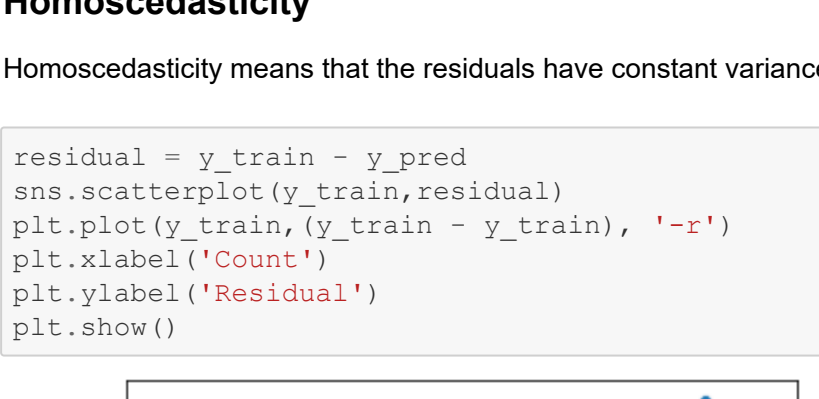


The errors terms are normally Distributed with mean around 0

Homoscedasticity

Homoscedasticity means that the residuals have constant variance no matter the level of the dependent variable.

```
In [302]: residual = y_train - y_pred
          sns.scatterplot(y_train,residual)
          plt.plot(y_train,(y_train - y_train), '-r')
          plt.xlabel('Count')
          plt.ylabel('Residual')
          plt.show()
```

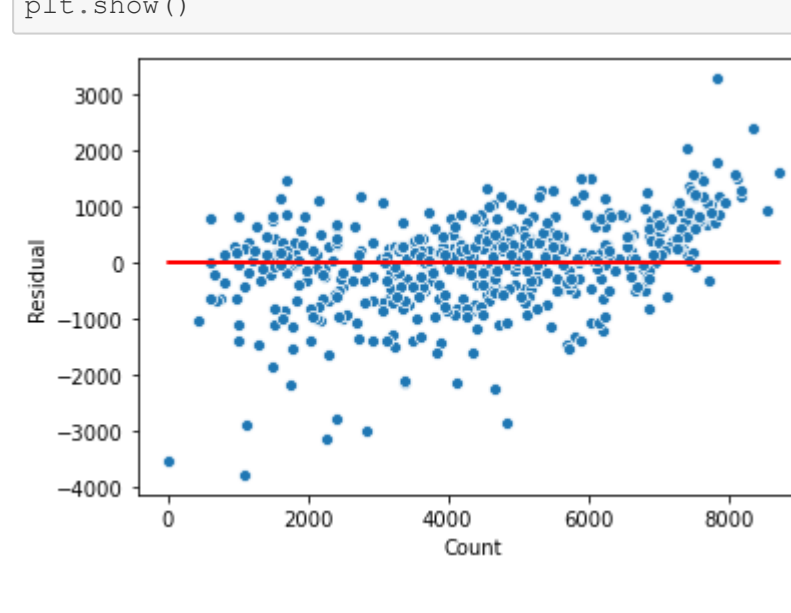


Insight:

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

Normality of Error terms

```
In [303]: sm.qqplot((y_train - y_pred), fit=True, line='45')
          plt.title('Q-Q Plot')
          plt.show()
```



If the errors are normally distributed, the plot would show fairly straight line. Absence of normality in the errors can be seen with deviation in the straight line.

Infrence:

The error terms are normally distributed

Independence of residuals (DW Statistics)

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 [304]: DW_value = round(sm.stats.stattools.durbin_watson((y_train - y_pred)),4)
          print('The Durbin-Watson value for Final Model8 is',DW_value)

          The Durbin-Watson value for Final Model8 is 2.0567
```

Insight: There is almost no autocorrelation.

Multicolinearity Check

> VIF value <= 4 suggests no multicollinearity
> VIF value of >= 10 implies serious multicollinearity

```
In [305]: # Calling VIF from the function for the final model define within the function
          vif
Out[305]:
```

	features	VIF
0	temp	4.72
1	workingday	4.63
2	year	2.07
3	Saturday	1.81
4	summer	1.74
5	spring	1.70
6	winter	1.59
7	Cloudy	1.56
8	Sep	1.23
9	LightSnow Rain	1.07

Infrence:

All the predictor variables have VIF value less than 5. So we can say that there is insignificant multicolinearity among the predictor variables.

Thus, the model stands on all the assumption of the Linear regresion

Making Predictions

```
In [306]: df_model_test.head()
Out[306]:
```

	year	holiday	workingday	temp	humidity	windspeed	count	spring	summer	winter	...	Oct	Sep	Monday	Saturday	Sunc
184	0	1	0	29.793347	63.7917	5.459106	6043	0	0	0	...	0	0	1	0	
535	1	0	1	32.082500	59.2083	7.625404	6211	0	1	0	...	0	0	0	0	
299	0	0	1	19.270000	81.2917	13.250121	2659	0	0	1	...	1	0	0	0	
221	0	0	1	31.433347	42.4167	13.417286	4780	0	0	0	...	0	0	0	0	
152	0	0	1	29.315000	30.5000	19.583229	4968	0	1	0	...	0	0	0	0	

5 rows × 29 columns

Scaling

Apply scaler() to all numeric variables in test dataset.

> Note: we will only use **scaler.transform**, as we want to use the metrics that the model learned from the training data to be applied on the test data.

In other words, we want to prevent the information leak from train to test dataset.

```
In [307]: df_model_test.columns
Out[307]: Index(['year', 'holiday', 'workingday', 'temp', 'humidity', 'windspeed',
                 'count', 'spring', 'summer', 'winter', 'Aug', 'Dec', 'Feb', 'Jan',
                 'July', 'June', 'March', 'May', 'Nov', 'Oct', 'Sep', 'Monday',
                 'Saturday', 'Sunday', 'Thrusday', 'Tuesday', 'Wednesday', 'Cloudy',
                 'LightSnow Rain'],
                dtype='object')
```

```
In [308]: df_model_test[variables] = scaler.transform(df_model_test[variables])
```

```
In [309]: df_model_test.head()
Out[309]:
```

	year	holiday	workingday	temp	humidity	windspeed	count	spring	summer	winter	...	Oct	Sep	Monday	Saturday	Sunda
184	0	1	0	0.831783	0.657364	0.084219	6043	0	0	0	...	0	0	1	0	
535	1	0	1	0.901354	0.610133	0.153728	6211	0	1	0	...	0	0	0	0	
299	0	0	1	0.511964	0.837699	0.334206	2659	0	0	1	...	1	0	0	0	
221	0	0	1	0.881625	0.437098	0.339570	4780	0	0	0	...	0	0	0	0	
152	0	0	1	0.817246	0.314298	0.537414	4968	0	1	0	...	0	0	0	0	

5 rows × 29 columns

Creating X_test and y_test dataset

```
In [310]: y_test = df_model_test.pop('count')
          X_test = df_model_test
```

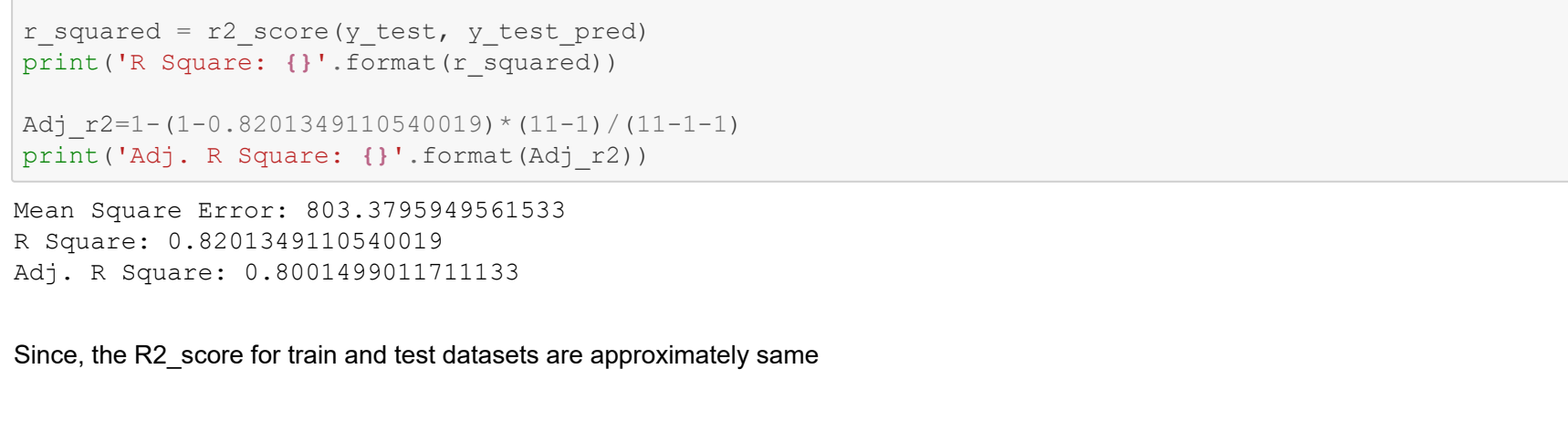
```
In [311]: # Add Constant
          X_test_sm = sm.add_constant(X_test[feature])
          # create a fitted model in one line
          lr_test = sm.OLS(y_test,X_test_sm).fit()
          y_test_pred = lr_test.predict(X_test_sm)
```

```
In [312]: #MSE
          print('Mean Square Error: {}'.format(np.sqrt(mean_squared_error(y_test, y_test_pred))))

          r_squared = r2_score(y_test, y_test_pred)
          print('R Square: {}'.format(r_squared))

          Adj_r2=1-(1-0.8201349110540019)*(11-1)/(11-1-1)
          print('Adj. R Square: {}'.format(Adj_r2))

          Mean Square Error: 803.3795949561533
          R Square: 0.8201349110540019
          Adj. R Square: 0.8001499011711133
```

Since, the R2_score for train and test datasets are approximately same

Model Evaluation

```
In [313]: # Plotting y_test and y_pred to understand the spread.
          plt.figure(figsize=(15,6))
          plt.scatter(y_test,y_test_pred,color='blue')
          plt.title('y_test vs y_pred', fontsize=15)
          plt.xlabel('y_test', fontsize=18)
          plt.ylabel('y_pred', fontsize=16)
          plt.show()
```




```
In [314]: from math import sqrt
          round(sqrt(mean_squared_error(y_test, y_test_pred)),4)
Out[314]: 803.3796
```

```
In [315]: r_squared = r2_score(y_test, y_test_pred)
          print('R Square: {}'.format(r_squared))

          Adj_r2=1-(1-0.8201349110540019)*(11-1)/(11-1-1)
          print('Adj. R Square: {}'.format(Adj_r2))

          R Square: 0.8201349110540019
          Adj. R Square: 0.8001499011711133
```

```
In [316]: pd.DataFrame({"Measurement":['R Square','Adj. R Square'],"Train":[0.826,0.823],"Test":[0.82, 0.80]})
Out[316]:
```

	Measurement	Train	Test
0	R Square	0.826	0.82
1	Adj. R Square	0.823	0.80

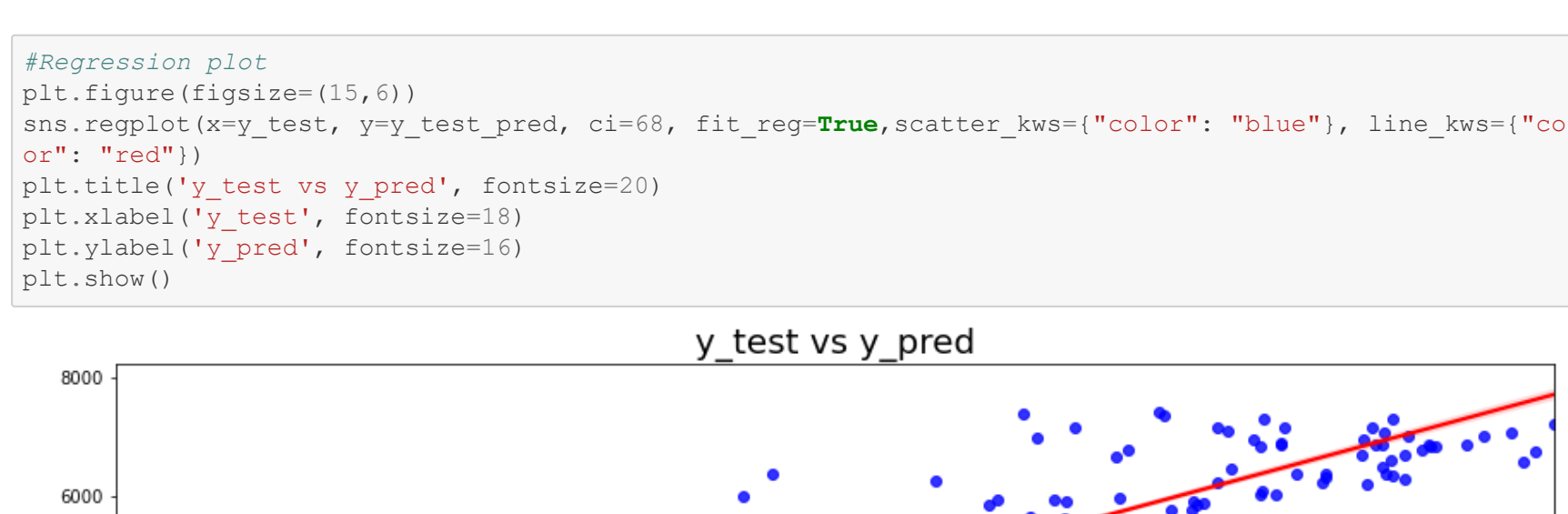
The R Square and Adj. R Square values for train and Test datasets are very close

Hence, it seems to be good model with 82.6% of R-Square and free from overfiting as well generalize the various datasets

```
In [317]: #RMSE
          print('Root Mean Square Error: {}'.format(np.sqrt(mean_squared_error(y_test, y_test_pred))))
          #AMSE
          print('Mean Absolute Error: {}'.format(np.sqrt(mean_absolute_error(y_test, y_test_pred))))

          Root Mean Square Error: 803.3795949561533
          Mean Absolute Error: 24.748135947167846
```

The Root Mean Squared Error value for the test dataset based on final model is 0.0924 and Mean Absolute Error is 0.26544, which indicates that the model is really good.

```
In [318]: #Regression plot
          plt.figure(figsize=(15,6))
          sns.regplot(x=y_test, y=y_test_pred, ci=68, fit_reg=True,scatter_kws={"color": "blue"}, line_kws={"col
          or": "red"})
          plt.title('y_test vs y_pred', fontsize=20)
          plt.xlabel('y_test', fontsize=18)
          plt.ylabel('y_pred', fontsize=16)
          plt.show()
```



Hence, We reach to the decent model for the the demand for shared bikes with the significant variables

Best Fit Line (y_pred) = 0.0901 + year *(0.2333)* + *workingday* (0.0568) + spring *(-0.0647)* + *summer* (0.0516) + winter *(0.0984)* + *Sep* (0.0915) + Saturday *(0.0647)* + *Cloudy* (-0.0800) + LightSnow Rain *(-0.3051)* + *temp* (0.4918)

R-Square: 82.6%

Adj. R-Square: 82.3%

F-Statistics: 237.5

Important & Significant Features(predictors) which effect demeand for shared bikes:

- Temperature has a high coefficient of 0.4918, which means that if the temperature increases by 1 unit the demand goes up by 0.4918 units

- During Heavy Rain the user strictly won't prefer to rent a bike while during cloudy and light snow rain climate the demand for renting a shared bike goes down as these variables are negatively corealted with the negative coefficient of -0.0800 and -0.3051 respectively

- The demand of renting BIKE Usually high on Saturday as compared to usuall working days so the firm need to stock up the bike till weekend

- Similarly, During Summer and winter the user prefer to rent a bike so the provider can introduce some offer or increase the promotion to boost the sales and during spring the demand goes down and for boosting the sales the providers can introduce some small budget plan

- **Another Important point to infer that yearly demand is going up and the rental number is increasing by the positive coefficient of 0.2333**