

Linear Regression Assignment (Bike Share)

Reading and Understanding the Data

In [1]:

```
# Import all the required libraries

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [2]:

```
# Import the data

bike_data = pd.read_csv('day.csv')
bike_data.head()
```

Out[2]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp
0	1	01-01-2018	1	0	1	0	6	0	2	14.110847
1	2	02-01-2018	1	0	1	0	0	0	2	14.902598
2	3	03-01-2018	1	0	1	0	1	1	1	8.050924
3	4	04-01-2018	1	0	1	0	2	1	1	8.200000
4	5	05-01-2018	1	0	1	0	3	1	1	9.305237

In [3]:

```
# Check the shape

bike_data.shape
```

Out[3]:

(730, 16)

In [4]:

```
# Check data-info and other info regarding the null data
bike_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
```

Observation

- We see that there are *no null datapoint* available in any columns.
- Total *16 features* are available to study the data.

In [5]:

```
# Describe the numeric columns of the dataset

bike_data.describe()
```

Out[5]:

	instant	season	yr	mnth	holiday	weekday	workingday
count	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000
mean	365.500000	2.498630	0.500000	6.526027	0.028767	2.997260	0.683562
std	210.877136	1.110184	0.500343	3.450215	0.167266	2.006161	0.465405
min	1.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000
25%	183.250000	2.000000	0.000000	4.000000	0.000000	1.000000	0.000000
50%	365.500000	3.000000	0.500000	7.000000	0.000000	3.000000	1.000000
75%	547.750000	3.000000	1.000000	10.000000	0.000000	5.000000	1.000000
max	730.000000	4.000000	1.000000	12.000000	1.000000	6.000000	1.000000

In [6]:

```
# Rename few columns name for better understanding and interpretation

bike_data.rename(columns={"yr":"year",
                          "mnth":"month",
                          "atemp":"abs-temp",
                          "hum":"humidity"}, inplace=True)

bike_data.head()
```

Out[6]:

	instant	dteday	season	year	month	holiday	weekday	workingday	weathersit	ten
0	1	01-01-2018	1	0	1	0	6	0	2	14.1108
1	2	02-01-2018	1	0	1	0	0	0	2	14.9025
2	3	03-01-2018	1	0	1	0	1	1	1	8.0509
3	4	04-01-2018	1	0	1	0	2	1	1	8.2000
4	5	05-01-2018	1	0	1	0	3	1	1	9.3052

Drop the following columns

- instant : record index
- dteday : date
- casual and registered : Due to a feature column cnt , which is a sum of these two columns.

In [7]:

```
# Drop some columns

bike_data.drop(['instant', 'dteday', 'casual', 'registered'], axis=1, inplace=True)
```

In [8]:

```
# We, see that after dropping some columns, the remaining features are 12.

bike_data.shape
```

Out[8]:

(730, 12)

Visualising the Data

Let's now spend some time doing what is arguably the most important step - **understanding the data**.

- If there is some obvious multicollinearity going on, this is the first place to catch it

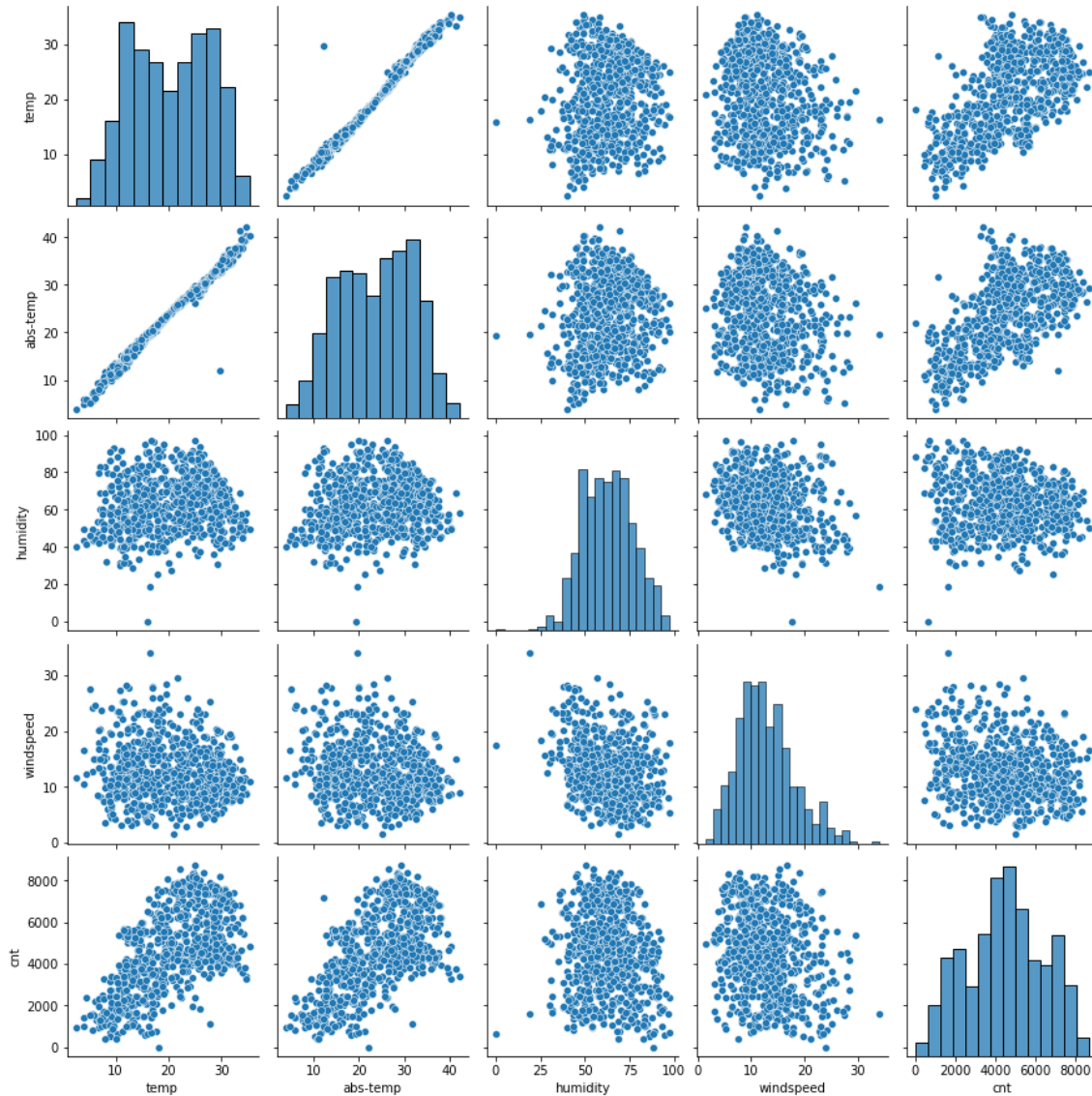
- Here's where you'll also identify if some predictors directly have a strong association with the outcome variable

We'll visualise our data using `matplotlib` and `seaborn`.

In [9]:

```
# Lets see pair plot to understand the behaviour of one feature w.r.t to other feature
```

```
sns.pairplot(data=bike_data,vars=['temp','abs-temp','humidity','windspeed','cnt'])  
plt.show()
```



Observation

- We see some very strong linear relation between `temp`, `abs-temp` and `cnt`.
- We see `temp` and `abs-temp` are very strongly correlated with each other.

In [10]:

```
# Lets map the data based on the data dictionary in order to create dummy columns

bike_data['season'] = bike_data['season'].map({1:"spring", 2:"summer", 3:"fall", 4:"winter"})
bike_data['month'] = bike_data['month'].map({1:"Jan", 2:"Feb", 3:"Mar", 4:"Apr", 5:"May", 6:"Jun", 7:"Jul", 8:"Aug", 9:"Sep", 10:"Oct", 11:"Nov", 12:"Dec"})

bike_data['weekday'] = bike_data['weekday'].map({0:"Sun", 1:"Mon", 2:"Tue", 3:"Wed", 4:"Thu", 5:"Fri", 6:"Sat"})

bike_data['weathersit'] = bike_data['weathersit'].map({1:"Clear", 2:"Mist", 3:"Light Snow", 4:"Heavy Rain"})

bike_data.head()
```

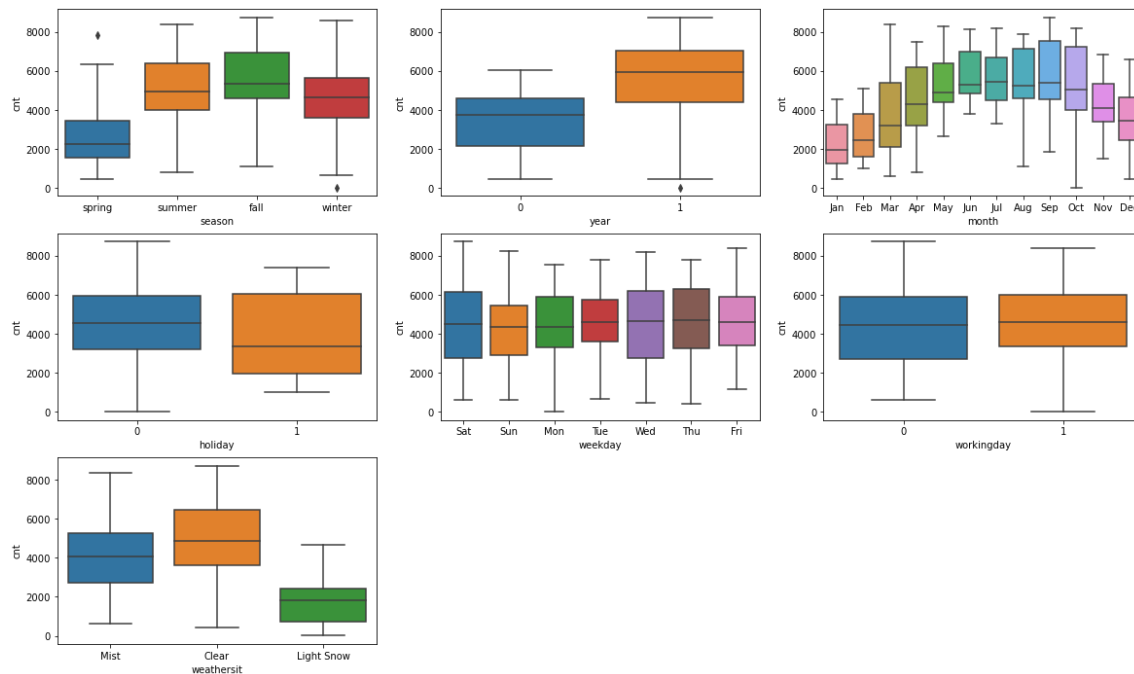
Out[10]:

	season	year	month	holiday	weekday	workingday	weathersit	temp	abs-temp	humid
0	spring	0	Jan	0	Sat	0	Mist	14.110847	18.18125	86
1	spring	0	Jan	0	Sun	0	Mist	14.902598	17.68695	66
2	spring	0	Jan	0	Mon	1	Clear	8.050924	9.47025	46
3	spring	0	Jan	0	Tue	1	Clear	8.200000	10.60610	56
4	spring	0	Jan	0	Wed	1	Clear	9.305237	11.46350	46

In [11]:

```
# Lets understand the behaviour of some categorical data
```

```
plt.figure(figsize=(20, 12))
plt.subplot(3,3,1)
sns.boxplot(x = 'season', y = 'cnt', data = bike_data)
plt.subplot(3,3,2)
sns.boxplot(x = 'year', y = 'cnt', data = bike_data)
plt.subplot(3,3,3)
sns.boxplot(x = 'month', y = 'cnt', data = bike_data)
plt.subplot(3,3,4)
sns.boxplot(x = 'holiday', y = 'cnt', data = bike_data)
plt.subplot(3,3,5)
sns.boxplot(x = 'weekday', y = 'cnt', data = bike_data)
plt.subplot(3,3,6)
sns.boxplot(x = 'workingday', y = 'cnt', data = bike_data)
plt.subplot(3,3,7)
sns.boxplot(x = 'weathersit', y = 'cnt', data = bike_data)
plt.show()
```



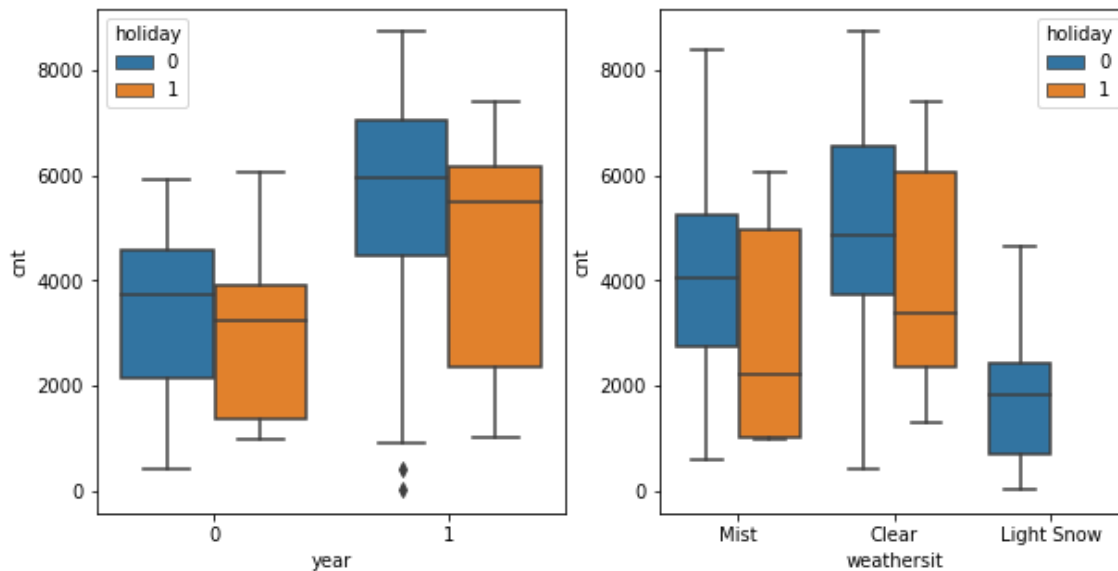
Observation

- Fall season has the most bike share count, whereas Spring has the minimum count.
- Number of counts in 2019 are significantly (33.33%) more than that of 2018.
- Number of counts in the month of Jun, July, Aug, and Sep are the most.
- Count is more when the weather is clear.

In [12]:

Bivariate analysis

```
plt.figure(figsize = (10, 5))
plt.subplot(1,2,1)
sns.boxplot(x = 'year', y = 'cnt', hue = 'holiday', data = bike_data)
plt.subplot(1,2,2)
sns.boxplot(x = 'weathersit', y = 'cnt', hue = 'holiday', data = bike_data)
plt.show()
```



Observation

- More people use the bike during the non-holiday time and clear weather.

In []:

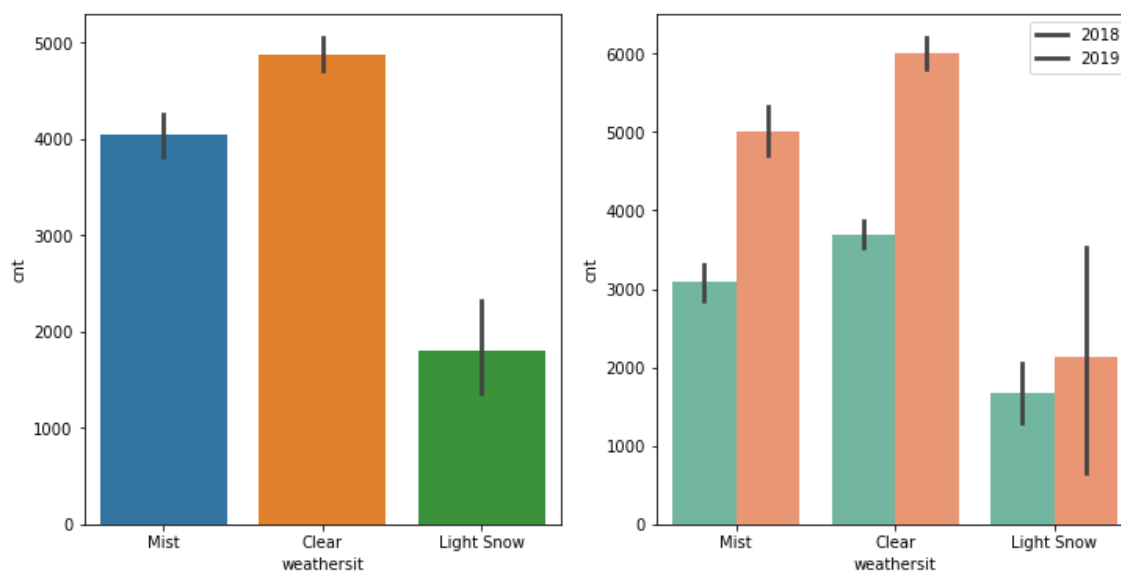
In [13]:

Define a function to get the bar plots

```
def get_bar_plot(column):
    plt.figure(figsize = (12,6))
    plt.subplot(1,2,1)
    sns.barplot(column, 'cnt', data=bike_data)
    plt.subplot(1,2,2)
    sns.barplot(column, 'cnt', data=bike_data, hue='year', palette='Set2')
    plt.legend(labels=['2018', '2019'])
    plt.show()
```

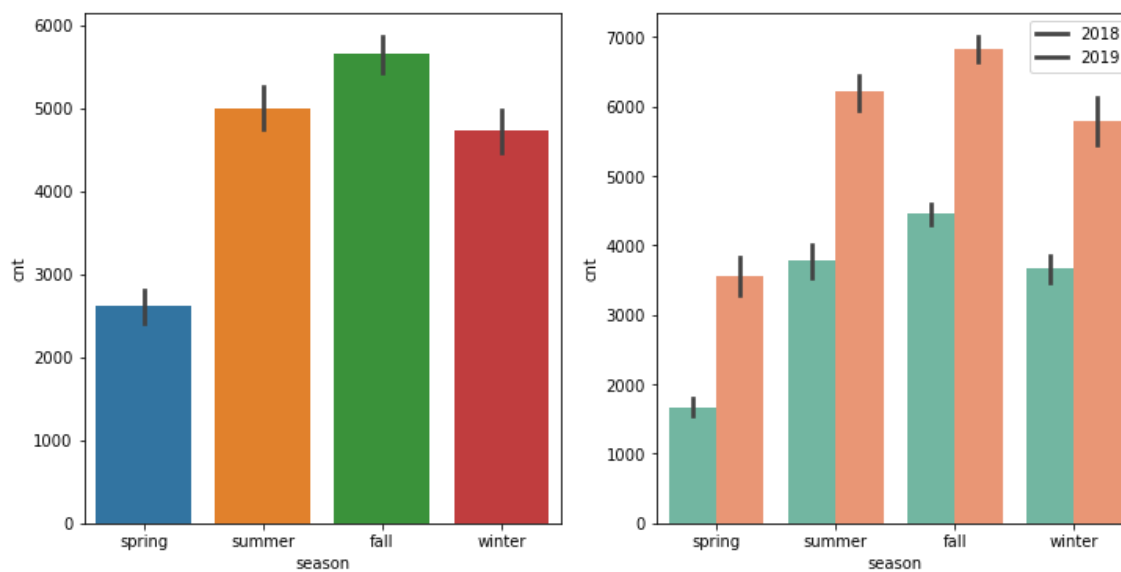
In [14]:

```
get_bar_plot('weathersit')
```



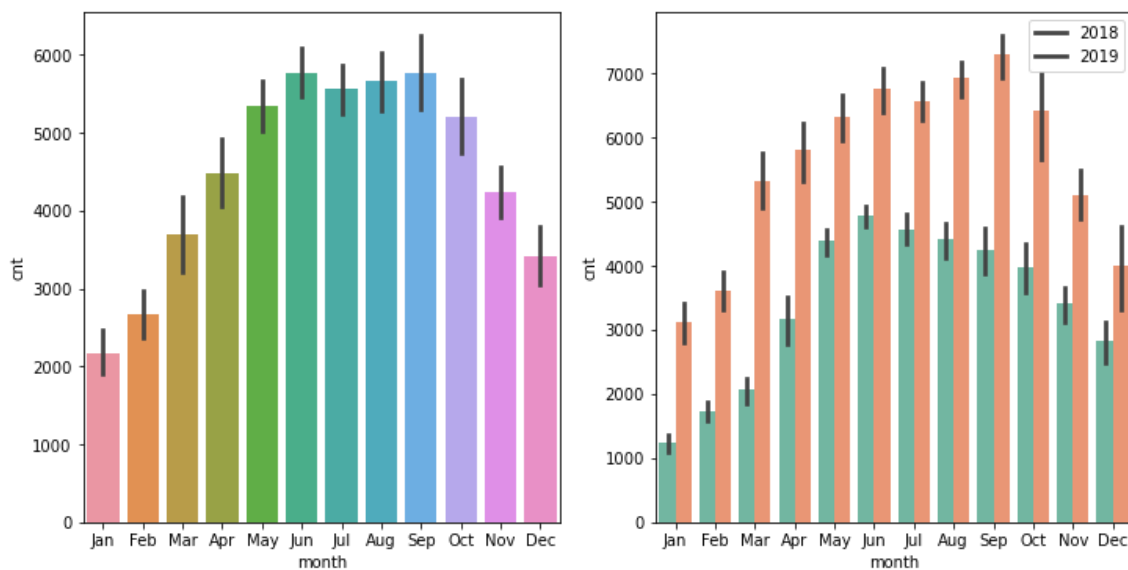
In [15]:

```
get_bar_plot('season')
```



In [16]:

```
get_bar_plot('month')
```



Observation

- More people use the bike during the non-holiday time and clear weather.
- Fall season has the most bike share count, where as the Spring has minimum count.
- Number of count in 2019 are significantly (33.33%) more than that of 2018.
- Number of count in the month of Jun, July, Aug and Sep are the most.
- Count is more when the weather is clear.

Create dummy variable

In [17]:

```
# Create a function to get the dummy variable dataframe

def get_dummy_dataframe(column_name: list[str]):
    output = pd.DataFrame()
    for column in column_name:
        status = pd.get_dummies(bike_data[column], drop_first=True)
        output = pd.concat([output, status], axis=1) # Concatenate the status DataFrame
    return output
```

In [18]:

```
# Dummy dataframe from the below categorical columns

dummy_df = get_dummy_dataframe(column_name=['season', 'month', 'weekday', 'weathersit'])
```

In [19]:

Dummy variable column name

dummy_df.columns

Out[19]:

```
Index(['spring', 'summer', 'winter', 'Aug', 'Dec', 'Feb', 'Jan', 'Jul', 'Jun',
      'Mar', 'May', 'Nov', 'Oct', 'Sep', 'Mon', 'Sat', 'Sun', 'Thu', 'Tue',
      'Wed', 'Light Snow', 'Mist'],
      dtype='object')
```

In [20]:

Dummy variable dataframe

dummy_df

Out[20]:

	spring	summer	winter	Aug	Dec	Feb	Jan	Jul	Jun	Mar	...	Oct	Sep	Mon	Sat
0	1	0	0	0	0	0	1	0	0	0	...	0	0	0	1
1	1	0	0	0	0	0	1	0	0	0	...	0	0	0	0
2	1	0	0	0	0	0	1	0	0	0	...	0	0	1	0
3	1	0	0	0	0	0	1	0	0	0	...	0	0	0	0
4	1	0	0	0	0	0	1	0	0	0	...	0	0	0	0
...
725	1	0	0	0	1	0	0	0	0	0	...	0	0	0	0
726	1	0	0	0	1	0	0	0	0	0	...	0	0	0	0
727	1	0	0	0	1	0	0	0	0	0	...	0	0	0	1
728	1	0	0	0	1	0	0	0	0	0	...	0	0	0	0
729	1	0	0	0	1	0	0	0	0	0	...	0	0	1	0

730 rows × 22 columns

In [21]:

Concatenate dummy dataframe with the original dataframe

bike_data = pd.concat([bike_data, dummy_df], axis=1)

In [22]:

```
# Concatinated dataframe

bike_data
```

Out[22]:

	season	year	month	holiday	weekday	workingday	weathersit	temp	abs-temp	i
0	spring	0	Jan	0	Sat	0	Mist	14.110847	18.18125	
1	spring	0	Jan	0	Sun	0	Mist	14.902598	17.68695	
2	spring	0	Jan	0	Mon	1	Clear	8.050924	9.47025	
3	spring	0	Jan	0	Tue	1	Clear	8.200000	10.60610	
4	spring	0	Jan	0	Wed	1	Clear	9.305237	11.46350	
...	
725	spring	1	Dec	0	Thu	1	Mist	10.420847	11.33210	
726	spring	1	Dec	0	Fri	1	Mist	10.386653	12.75230	
727	spring	1	Dec	0	Sat	0	Mist	10.386653	12.12000	
728	spring	1	Dec	0	Sun	0	Clear	10.489153	11.58500	
729	spring	1	Dec	0	Mon	1	Mist	8.849153	11.17435	

730 rows × 34 columns



In [23]:

```
# Data info
```

```
bike_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 730 entries, 0 to 729
Data columns (total 34 columns):
 #   Column                Non-Null Count  Dtype  
---  -
 0   season                730 non-null   object 
 1   year                  730 non-null   int64  
 2   month                 730 non-null   object 
 3   holiday                730 non-null   int64  
 4   weekday                730 non-null   object 
 5   workingday            730 non-null   int64  
 6   weathersit             730 non-null   object 
 7   temp                  730 non-null   float64 
 8   abs-temp              730 non-null   float64 
 9   humidity              730 non-null   float64 
10  windspeed             730 non-null   float64 
11  cnt                   730 non-null   int64  
12  spring                730 non-null   uint8  
13  summer                730 non-null   uint8  
14  winter                730 non-null   uint8  
15  Aug                   730 non-null   uint8  
16  Dec                   730 non-null   uint8  
17  Feb                   730 non-null   uint8  
18  Jan                   730 non-null   uint8  
19  Jul                   730 non-null   uint8  
20  Jun                   730 non-null   uint8  
21  Mar                   730 non-null   uint8  
22  May                   730 non-null   uint8  
23  Nov                   730 non-null   uint8  
24  Oct                   730 non-null   uint8  
25  Sep                   730 non-null   uint8  
26  Mon                   730 non-null   uint8  
27  Sat                   730 non-null   uint8  
28  Sun                   730 non-null   uint8  
29  Thu                   730 non-null   uint8  
30  Tue                   730 non-null   uint8  
31  Wed                   730 non-null   uint8  
32  Light Snow            730 non-null   uint8  
33  Mist                  730 non-null   uint8  
dtypes: float64(4), int64(4), object(4), uint8(22)
memory usage: 84.2+ KB
```

In [24]:

```
# Drop columns for which dummy df created

bike_data.drop(['season', 'month', 'weekday', 'weathersit'],
               axis=1, inplace=True)

bike_data.head()
```

Out[24]:

	year	holiday	workingday	temp	abs-temp	humidity	windspeed	cnt	spring	sumr
0	0	0	0	14.110847	18.18125	80.5833	10.749882	985	1	
1	0	0	0	14.902598	17.68695	69.6087	16.652113	801	1	
2	0	0	1	8.050924	9.47025	43.7273	16.636703	1349	1	
3	0	0	1	8.200000	10.60610	59.0435	10.739832	1562	1	
4	0	0	1	9.305237	11.46350	43.6957	12.522300	1600	1	

5 rows × 30 columns



In [25]:

```
# Final dataframe

bike_data.columns
```

Out[25]:

```
Index(['year', 'holiday', 'workingday', 'temp', 'abs-temp', 'humidity',
      'windspeed', 'cnt', 'spring', 'summer', 'winter', 'Aug', 'Dec', 'Feb',
      'Jan', 'Jul', 'Jun', 'Mar', 'May', 'Nov', 'Oct', 'Sep', 'Mon', 'Sat',
      'Sun', 'Thu', 'Tue', 'Wed', 'Light Snow', 'Mist'],
      dtype='object')
```

In [26]:

```
# We specify this so that the train and test data set always have the same rows, respect
# Create a train test split

np.random.seed(0)
df_train, df_test = train_test_split(bike_data, train_size = 0.7, test_size = 0.3, random_state=0)
```

In [27]:

```
# Test data

df_test
```

Out[27]:

	year	holiday	workingday	temp	abs-temp	humidity	windspeed	cnt	spring	su
184	0	1	0	29.793347	33.27085	63.7917	5.459106	6043	0	
535	1	0	1	32.082500	36.04875	59.2083	7.625404	6211	0	
299	0	0	1	19.270000	22.85230	81.2917	13.250121	2659	0	
221	0	0	1	31.433347	34.24915	42.4167	13.417286	4780	0	
152	0	0	1	29.315000	32.19710	30.5000	19.583229	4968	0	
...
400	1	0	0	10.899153	13.22605	68.7917	11.791732	2947	1	
702	1	0	1	19.509153	23.45270	73.3750	11.666643	6606	0	
127	0	0	0	21.661653	25.94665	63.1667	5.000712	4333	0	
640	1	0	1	26.957500	29.95665	79.3750	4.458569	7572	0	
72	0	0	1	13.333897	16.60000	49.6957	9.174042	2046	1	

219 rows × 30 columns



In [28]:

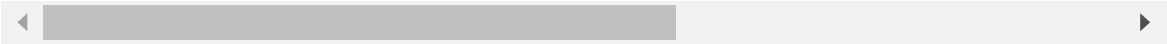
```
# Train data

df_train
```

Out[28]:

	year	holiday	workingday	temp	abs-temp	humidity	windspeed	cnt	spring	su
653	1	0	1	19.201653	23.04230	55.8333	12.208807	7534	0	
576	1	0	1	29.246653	33.14480	70.4167	11.083475	7216	0	
426	1	0	0	16.980847	20.67460	62.1250	10.792293	4066	1	
728	1	0	0	10.489153	11.58500	48.3333	23.500518	1796	1	
482	1	0	0	15.443347	18.87520	48.9583	8.708325	4220	0	
...
526	1	0	1	29.554153	32.98605	58.7917	13.916771	6664	0	
578	1	0	1	30.852500	35.35440	65.9583	8.666718	7261	0	
53	0	0	1	9.091299	12.28585	42.3043	6.305571	1917	1	
350	0	0	0	10.591653	12.46855	56.0833	16.292189	2739	0	
79	0	0	1	17.647835	20.48675	73.7391	19.348461	2077	0	

510 rows × 30 columns



In [29]:

```
# Use scalling for some numeric feature
```

```
scaler = MinMaxScaler()
num_var = ['temp', 'abs-temp', 'humidity', 'windspeed', 'cnt']
df_train[num_var] = scaler.fit_transform(df_train[num_var])
df_train
```

Out[29]:

	year	holiday	workingday	temp	abs-temp	humidity	windspeed	cnt	spring
653	1	0	1	0.509887	0.501133	0.575354	0.300794	0.864243	0
576	1	0	1	0.815169	0.766351	0.725633	0.264686	0.827658	0
426	1	0	0	0.442393	0.438975	0.640189	0.255342	0.465255	1
728	1	0	0	0.245101	0.200348	0.498067	0.663106	0.204096	1
482	1	0	0	0.395666	0.391735	0.504508	0.188475	0.482973	0
...
526	1	0	1	0.824514	0.762183	0.605840	0.355596	0.764151	0
578	1	0	1	0.863973	0.824359	0.679690	0.187140	0.832835	0
53	0	0	1	0.202618	0.218747	0.435939	0.111379	0.218017	1
350	0	0	0	0.248216	0.223544	0.577930	0.431816	0.312586	0
79	0	0	1	0.462664	0.434043	0.759870	0.529881	0.236424	0

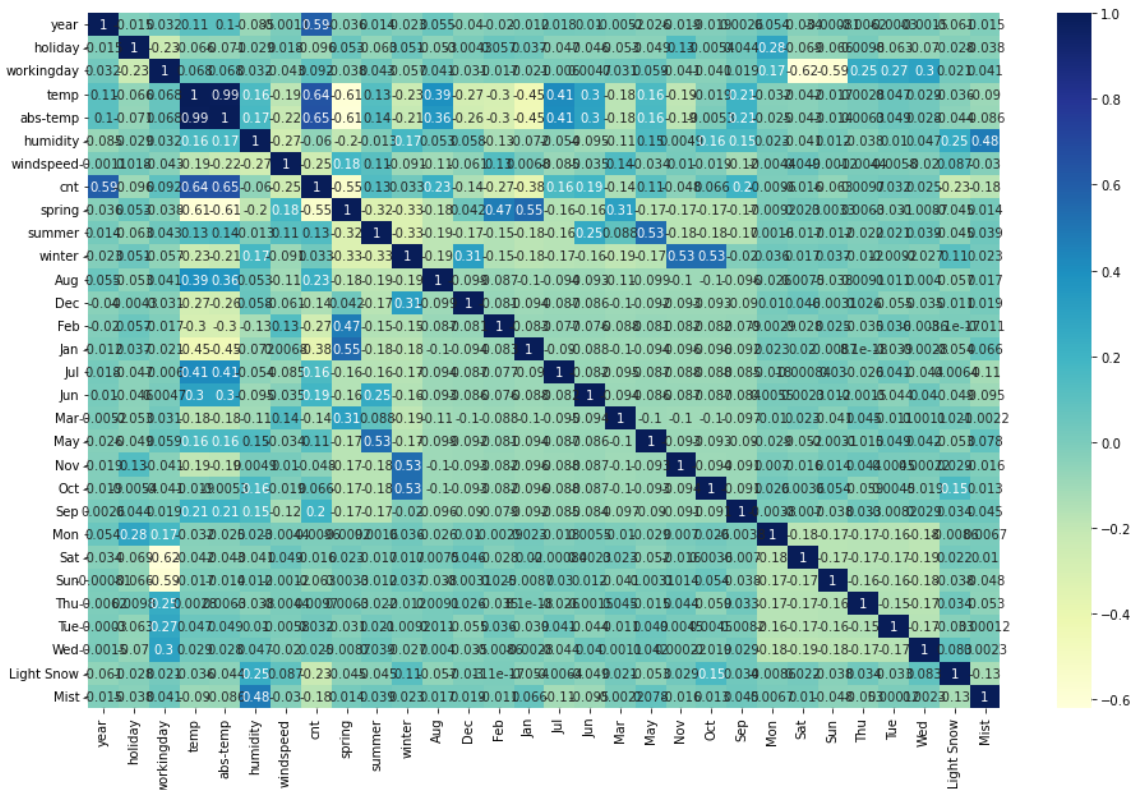
510 rows × 30 columns



In [30]:

Correlation plot for train dataset

```
plt.figure(figsize = (16, 10))
sns.heatmap(df_train.corr(), annot = True, cmap="YlGnBu")
plt.show()
```



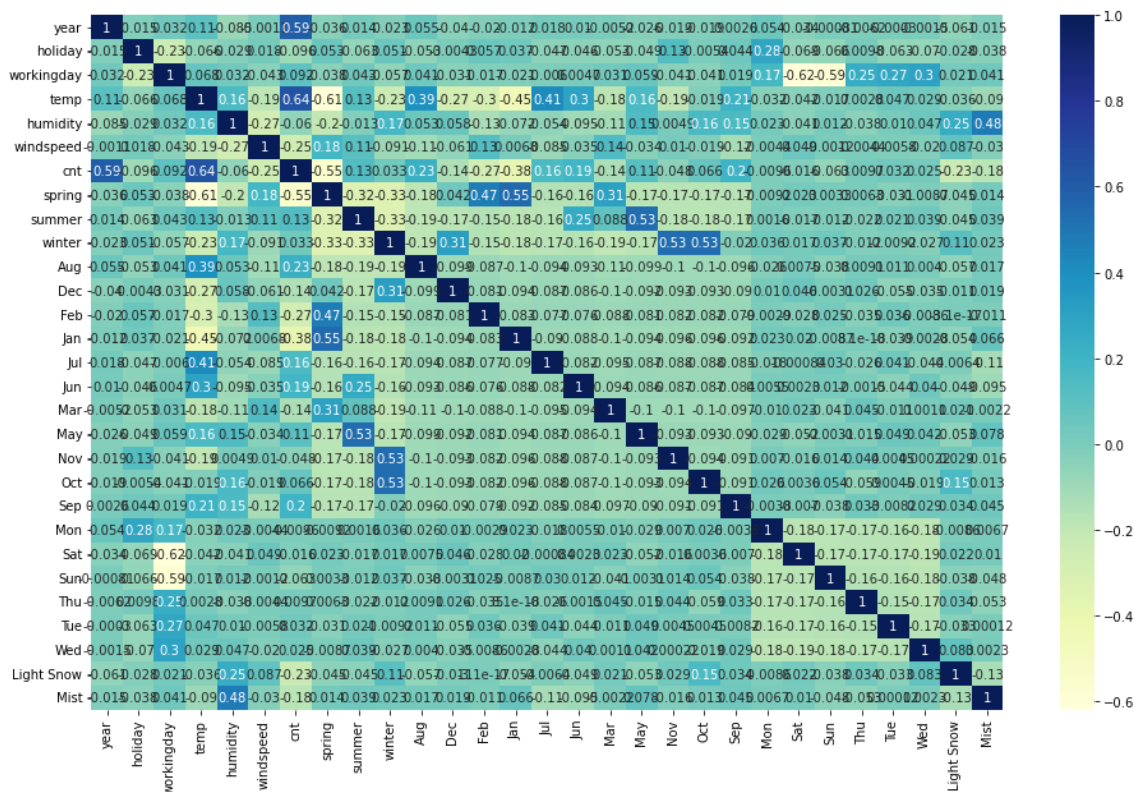
Observation

- We see that temp and abs_temp are highly correlated, therefore we will remove one of these feature.

In [31]:

Drop 'abs-temp' columns

```
df_train.drop(['abs-temp'], axis=1, inplace=True)
plt.figure(figsize = (16, 10))
sns.heatmap(df_train.corr(), annot = True, cmap="YlGnBu")
plt.show()
```



Observation

- Now, the strong correlation effect has been removed.

Fit the model and predict using best model

In [32]:

```
# Select target feature and apply model
# Use RFE to select the best features
```

```
y_train = df_train.pop('cnt')
X_train = df_train
```

```
lm = LinearRegression()
lm.fit(X_train, y_train)
```

```
rfe = RFE(lm, 20)
rfe = rfe.fit(X_train, y_train)
```

In [33]:

```
# List of features with their names and ranking  
  
list(zip(X_train.columns, rfe.support_, rfe.ranking_))
```

Out[33]:

```
[('year', True, 1),  
 ('holiday', True, 1),  
 ('workingday', True, 1),  
 ('temp', True, 1),  
 ('humidity', True, 1),  
 ('windspeed', True, 1),  
 ('spring', True, 1),  
 ('summer', True, 1),  
 ('winter', True, 1),  
 ('Aug', False, 2),  
 ('Dec', True, 1),  
 ('Feb', True, 1),  
 ('Jan', True, 1),  
 ('Jul', True, 1),  
 ('Jun', False, 8),  
 ('Mar', False, 9),  
 ('May', True, 1),  
 ('Nov', True, 1),  
 ('Oct', False, 5),  
 ('Sep', True, 1),  
 ('Mon', False, 3),  
 ('Sat', True, 1),  
 ('Sun', True, 1),  
 ('Thu', False, 6),  
 ('Tue', False, 4),  
 ('Wed', False, 7),  
 ('Light Snow', True, 1),  
 ('Mist', True, 1)]
```

In [34]:

```
# Cols which supports the model  
  
col = X_train.columns[rfe.support_]  
col
```

Out[34]:

```
Index(['year', 'holiday', 'workingday', 'temp', 'humidity', 'windspeed',  
      'spring', 'summer', 'winter', 'Dec', 'Feb', 'Jan', 'Jul', 'May', 'Nov',  
      'Sep', 'Sat', 'Sun', 'Light Snow', 'Mist'],  
      dtype='object')
```

In [35]:

Column which does not support the model`X_train.columns[~rfe.support_]`

Out[35]:

`Index(['Aug', 'Jun', 'Mar', 'Oct', 'Mon', 'Thu', 'Tue', 'Wed'], dtype='object')`

In [36]:

Build a model WITH required feature which supports the model`X_train_rfe_1 = X_train[col]`

In [37]:

Add constant to the model

```
import statsmodels.api as sm
X_train_lm_1 = sm.add_constant(X_train_rfe_1)
```

In [38]:

Fit the model`lm_1 = sm.OLS(y_train, X_train_lm_1).fit()`

In [39]:

Print the model summary`print(lm_1.summary())`

OLS Regression Results

```
=====
=====
Dep. Variable:          cnt    R-squared:
0.852
Model:                OLS    Adj. R-squared:
0.847
Method:             Least Squares    F-statistic:
148.8
Date:                Wed, 16 Aug 2023    Prob (F-statistic):          1.
59e-189
Time:                16:53:37    Log-Likelihood:
526.24
No. Observations:          510    AIC:
-1012.
Df Residuals:              490    BIC:
-927.8
Df Model:                  19
Covariance Type:          nonrobust
```

In [40]:

```
# Write a function to create a VIF dataframe

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

In [41]:

```
# Get VIF for 1st set of data

get_vif_dataframe(X_train_rfe_1)
```

Out[41]:

	Features	VIF
2	workingday	60.93
16	Sat	14.59
17	Sun	13.91
6	spring	5.79
3	temp	4.88
8	winter	3.88
7	summer	3.49
1	holiday	3.43
11	Jan	2.38
4	humidity	1.97
10	Feb	1.87
14	Nov	1.81
9	Dec	1.65
19	Mist	1.58
13	May	1.52
12	Jul	1.49
15	Sep	1.34
18	Light Snow	1.28
5	windspeed	1.22
0	year	1.04

In [42]:

```
# Perform the second iteration (As the feature 'holiday' is not significant), so will re  
X_train_rfe_2 = X_train_rfe_1.drop(['holiday'], axis=1)
```

In [43]:

```
# Fit the model based on feature selecte in 2nd iteration  
  
X_train_lm_2 = sm.add_constant(X_train_rfe_2)  
lm_2 = sm.OLS(y_train, X_train_lm_2).fit()  
print(lm_2.summary())
```

OLS Regression Results

```

=====
====
Dep. Variable:          cnt    R-squared:
0.852
Model:                  OLS    Adj. R-squared:
0.847
Method:                 Least Squares    F-statistic:          1
48.8
Date:                   Wed, 16 Aug 2023    Prob (F-statistic):      1.59e
-189
Time:                   16:53:37    Log-Likelihood:          52
6.24
No. Observations:       510    AIC:                      -1
012.
Df Residuals:           490    BIC:                      -9
27.8
Df Model:               19
Covariance Type:        nonrobust
=====
=====

```

```

=====
====
              coef    std err          t      P>|t|      [0.025    0.
975]
-----
----
const         0.2494     0.045     5.516     0.000     0.161
0.338
year          0.2317     0.008    29.150     0.000     0.216
0.247
workingday     0.0933     0.025     3.685     0.000     0.044
0.143
temp          0.4500     0.038    11.796     0.000     0.375
0.525
humidity     -0.1521     0.038    -4.055     0.000    -0.226     -
0.078
windspeed    -0.1868     0.025    -7.365     0.000    -0.237     -
0.137
spring       -0.0560     0.022    -2.563     0.011    -0.099     -
0.013
summer        0.0269     0.017     1.593     0.112    -0.006
0.060
winter        0.1012     0.018     5.705     0.000     0.066
0.136
Dec          -0.0506     0.018    -2.807     0.005    -0.086     -
0.015
Feb          -0.0355     0.021    -1.661     0.097    -0.077
0.006
Jan          -0.0658     0.021    -3.110     0.002    -0.107     -
0.024
Jul          -0.0512     0.018    -2.858     0.004    -0.086     -
0.016
May           0.0250     0.017     1.449     0.148    -0.009
0.059
Nov          -0.0483     0.019    -2.592     0.010    -0.085     -
0.012
Sep           0.0718     0.017     4.324     0.000     0.039
0.104
Sat           0.1038     0.027     3.876     0.000     0.051
0.156
Sun           0.0491     0.027     1.827     0.068    -0.004
0.102

```


Light Snow	-0.2564	0.026	-9.847	0.000	-0.308	-
0.205						
Mist	-0.0599	0.010	-5.807	0.000	-0.080	-
0.040						

```
=====
=====
Omnibus:                84.215   Durbin-Watson:
2.035
Prob(Omnibus):          0.000   Jarque-Bera (JB):        24
1.321
Skew:                   -0.792   Prob(JB):                3.96
e-53
Kurtosis:               5.974   Cond. No.
27.5
=====
=====
```

Notes:

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

In [44]:

```
# VIF for 2nd set of data

get_vif_dataframe(X_train_rfe_2)
```

Out[44]:

	Features	VIF
3	humidity	34.79
2	temp	22.48
1	workingday	19.10
5	spring	6.06
15	Sat	5.11
4	windspeed	5.06
16	Sun	4.89
7	winter	4.40
6	summer	3.71
10	Jan	2.42
18	Mist	2.35
0	year	2.11
9	Feb	1.92
13	Nov	1.83
8	Dec	1.70
11	Jul	1.61
12	May	1.60
14	Sep	1.41
17	Light Snow	1.29

In [45]:

```
# 3rd iteration  
# As the feature 'May' is not significant), so will remove that  
  
X_train_rfe_3 = X_train_rfe_2.drop(['May'], axis=1)  
X_train_lm_3 = sm.add_constant(X_train_rfe_3)  
lm_3 = sm.OLS(y_train, X_train_lm_3).fit()  
print(lm_3.summary())
```

OLS Regression Results

```
=====
====
Dep. Variable:          cnt    R-squared:
0.852
Model:                  OLS    Adj. R-squared:
0.846
Method:                 Least Squares    F-statistic:          1
56.6
Date:                   Wed, 16 Aug 2023    Prob (F-statistic):      3.55e
-190
Time:                   16:53:37    Log-Likelihood:          52
5.15
No. Observations:       510    AIC:                      -1
012.
Df Residuals:           491    BIC:                      -9
31.8
Df Model:               18
Covariance Type:        nonrobust
=====
=====
```

```
=====
====
              coef    std err          t      P>|t|      [0.025    0.
975]
-----
----
const         0.2374     0.044     5.335     0.000     0.150
0.325
year          0.2311     0.008    29.082     0.000     0.215
0.247
workingday     0.0943     0.025     3.722     0.000     0.045
0.144
temp          0.4598     0.038    12.233     0.000     0.386
0.534
humidity     -0.1456     0.037    -3.904     0.000    -0.219    -
0.072
windspeed    -0.1887     0.025    -7.440     0.000    -0.239    -
0.139
spring       -0.0518     0.022    -2.390     0.017    -0.094    -
0.009
summer        0.0377     0.015     2.483     0.013     0.008
0.068
winter        0.1035     0.018     5.852     0.000     0.069
0.138
Dec          -0.0491     0.018    -2.727     0.007    -0.085    -
0.014
Feb          -0.0339     0.021    -1.591     0.112    -0.076
0.008
Jan          -0.0640     0.021    -3.025     0.003    -0.106    -
0.022
Jul          -0.0517     0.018    -2.883     0.004    -0.087    -
0.016
Nov          -0.0465     0.019    -2.499     0.013    -0.083    -
0.010
Sep           0.0718     0.017     4.319     0.000     0.039
0.104
Sat           0.1042     0.027     3.886     0.000     0.051
0.157
Sun           0.0499     0.027     1.857     0.064    -0.003
0.103
Light Snow   -0.2583     0.026    -9.924     0.000    -0.309    -
0.207
```

```

Mist          -0.0600    0.010   -5.813    0.000   -0.080   -
0.040
=====
====
Omnibus:                81.478   Durbin-Watson:
2.041
Prob(Omnibus):          0.000   Jarque-Bera (JB):          21
9.245
Skew:                   -0.787   Prob(JB):                2.46
e-48
Kurtosis:               5.800   Cond. No.
27.0
=====
=====

```

Notes:

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

In [46]:

```

# VIF dataframe

get_vif_dataframe(X_train_rfe_3)

```

Out[46]:

	Features	VIF
3	humidity	34.69
2	temp	22.29
1	workingday	18.92
5	spring	6.05
14	Sat	5.04
4	windspeed	4.99
15	Sun	4.85
7	winter	4.40
6	summer	3.09
10	Jan	2.42
17	Mist	2.35
0	year	2.10
9	Feb	1.92
12	Nov	1.83
8	Dec	1.70
11	Jul	1.61
13	Sep	1.41
16	Light Snow	1.29

In [47]:

```
# 4th iteration for the model fitting  
# As the feature 'Feb' is not significant), so will remove that  
X_train_rfe_4 = X_train_rfe_3.drop(['Feb'], axis=1)  
X_train_lm_4 = sm.add_constant(X_train_rfe_4)  
lm_4 = sm.OLS(y_train, X_train_lm_4).fit()  
print(lm_4.summary())
```

OLS Regression Results

```

=====
====
Dep. Variable:          cnt    R-squared:
0.851
Model:                  OLS    Adj. R-squared:
0.846
Method:                 Least Squares    F-statistic:          1
65.2
Date:                   Wed, 16 Aug 2023    Prob (F-statistic):      9.55e
-191
Time:                   16:53:37    Log-Likelihood:          52
3.84
No. Observations:      510    AIC:                      -1
012.
Df Residuals:          492    BIC:                      -9
35.5
Df Model:              17
Covariance Type:       nonrobust
=====
=====

```

```

=====
====
              coef    std err          t      P>|t|      [0.025    0.
975]
-----
----
const         0.2239     0.044     5.118     0.000     0.138
0.310
year          0.2308     0.008    29.005     0.000     0.215
0.246
workingday     0.0966     0.025     3.812     0.000     0.047
0.146
temp          0.4750     0.036    13.038     0.000     0.403
0.547
humidity     -0.1474     0.037    -3.949     0.000    -0.221    -
0.074
windspeed    -0.1876     0.025    -7.388     0.000    -0.237    -
0.138
spring       -0.0611     0.021    -2.925     0.004    -0.102    -
0.020
summer        0.0409     0.015     2.708     0.007     0.011
0.071
winter        0.1052     0.018     5.949     0.000     0.070
0.140
Dec          -0.0405     0.017    -2.351     0.019    -0.074    -
0.007
Jan          -0.0458     0.018    -2.570     0.010    -0.081    -
0.011
Jul          -0.0526     0.018    -2.932     0.004    -0.088    -
0.017
Nov          -0.0420     0.018    -2.277     0.023    -0.078    -
0.006
Sep           0.0732     0.017     4.404     0.000     0.041
0.106
Sat           0.1070     0.027     3.997     0.000     0.054
0.160
Sun           0.0520     0.027     1.933     0.054    -0.001
0.105
Light Snow   -0.2571     0.026   -9.865     0.000    -0.308    -
0.206
Mist         -0.0598     0.010    -5.781     0.000    -0.080    -
0.039

```

```
=====
====
Omnibus:                76.965    Durbin-Watson:
2.041
Prob(Omnibus):          0.000    Jarque-Bera (JB):        20
5.539
Skew:                   -0.747    Prob(JB):                2.33
e-45
Kurtosis:               5.727    Cond. No.
26.4
=====
=====
```

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

In []:

In [48]:

```
# VIF dataframe for 4th iteration

get_vif_dataframe(X_train_rfe_4)
```

Out[48]:

	Features	VIF
3	humidity	34.24
2	temp	21.56
1	workingday	18.83
5	spring	5.06
13	Sat	5.03
4	windspeed	4.98
14	Sun	4.83
7	winter	4.40
6	summer	3.08
16	Mist	2.34
0	year	2.10
11	Nov	1.81
9	Jan	1.77
10	Jul	1.61
8	Dec	1.58
12	Sep	1.41
15	Light Snow	1.29

In [49]:

```
# 5th iteration for the model fitting  
# As the feature 'Nov' is not significant), so will remove that  
  
X_train_rfe_5 = X_train_rfe_4.drop(['Nov'], axis=1)  
X_train_lm_5 = sm.add_constant(X_train_rfe_5)  
lm_5 = sm.OLS(y_train, X_train_lm_5).fit()  
print(lm_5.summary())
```


OLS Regression Results

```

=====
====
Dep. Variable:          cnt    R-squared:
0.849
Model:                  OLS    Adj. R-squared:
0.844
Method:                 Least Squares    F-statistic:          1
73.7
Date:                   Wed, 16 Aug 2023    Prob (F-statistic):      9.27e
-191
Time:                   16:53:38    Log-Likelihood:          52
1.17
No. Observations:      510    AIC:                      -1
008.
Df Residuals:          493    BIC:                      -9
36.3
Df Model:               16
Covariance Type:       nonrobust
=====
====

```

	coef	std err	t	P> t	[0.025	0.
975]						

const	0.1997	0.043	4.686	0.000	0.116	
0.283						
year	0.2306	0.008	28.866	0.000	0.215	
0.246						
workingday	0.1033	0.025	4.088	0.000	0.054	
0.153						
temp	0.4959	0.035	14.010	0.000	0.426	
0.565						
humidity	-0.1484	0.037	-3.959	0.000	-0.222	-
0.075						
windspeed	-0.1888	0.025	-7.405	0.000	-0.239	-
0.139						
spring	-0.0525	0.021	-2.543	0.011	-0.093	-
0.012						
summer	0.0467	0.015	3.130	0.002	0.017	
0.076						
winter	0.0954	0.017	5.538	0.000	0.062	
0.129						
Dec	-0.0251	0.016	-1.579	0.115	-0.056	
0.006						
Jan	-0.0406	0.018	-2.287	0.023	-0.076	-
0.006						
Jul	-0.0526	0.018	-2.916	0.004	-0.088	-
0.017						
Sep	0.0790	0.016	4.791	0.000	0.047	
0.111						
Sat	0.1142	0.027	4.274	0.000	0.062	
0.167						
Sun	0.0593	0.027	2.208	0.028	0.007	
0.112						
Light Snow	-0.2527	0.026	-9.682	0.000	-0.304	-
0.201						
Mist	-0.0589	0.010	-5.671	0.000	-0.079	-
0.038						

```

=====
====

```

Omnibus:
2.062
Prob(Omnibus):
7.450
Skew:
e-39
Kurtosis:
25.5

70.775
0.000
-0.709
5.518

Durbin-Watson:

Jarque-Bera (JB):

Prob(JB):
Cond. No.

17

2.93

=====

=====

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

In []:

In [50]:

```
# VIF dataframe for 5th iteration

get_vif_dataframe(X_train_rfe_5)
```

Out[50]:

	Features	VIF
3	humidity	33.73
2	temp	21.06
1	workingday	18.81
5	spring	5.04
12	Sat	5.03
4	windspeed	4.93
13	Sun	4.83
7	winter	3.79
6	summer	3.06
15	Mist	2.33
0	year	2.10
9	Jan	1.76
10	Jul	1.61
11	Sep	1.39
8	Dec	1.37
14	Light Snow	1.27

In [51]:

```
# 6th iteration for model fitting  
# As the feature 'Dec' is not significant), so will remove that  
  
X_train_rfe_6 = X_train_rfe_5.drop(['Dec'], axis=1)  
X_train_lm_6 = sm.add_constant(X_train_rfe_6)  
lm_6 = sm.OLS(y_train, X_train_lm_6).fit()  
print(lm_6.summary())
```

OLS Regression Results

=====

=====

Dep. Variable: cnt R-squared: 0.849

Model: OLS Adj. R-squared: 0.844

Method: Least Squares F-statistic: 84.6 1

Date: Wed, 16 Aug 2023 Prob (F-statistic): -191 2.31e

Time: 16:53:38 Log-Likelihood: 9.88 51

No. Observations: 510 AIC: -1

Df Residuals: 494 BIC: -9

Df Model: 15

Covariance Type: nonrobust

=====

=====

coef std err t P>|t| [0.025 0.975]

const 0.1925 0.042 4.537 0.000 0.109 0.276

year 0.2305 0.008 28.807 0.000 0.215 0.246

workingday 0.1021 0.025 4.035 0.000 0.052 0.152

temp 0.5098 0.034 14.851 0.000 0.442 0.577

humidity -0.1547 0.037 -4.145 0.000 -0.228 -0.081

windspeed -0.1860 0.025 -7.301 0.000 -0.236 -0.136

spring -0.0508 0.021 -2.461 0.014 -0.091 -0.010

summer 0.0497 0.015 3.346 0.001 0.020 0.079

winter 0.0944 0.017 5.479 0.000 0.061 0.128

Jan -0.0343 0.017 -1.980 0.048 -0.068 -0.000

Jul -0.0532 0.018 -2.947 0.003 -0.089 -0.018

Sep 0.0812 0.016 4.934 0.000 0.049 0.114

Sat 0.1123 0.027 4.200 0.000 0.060 0.165

Sun 0.0585 0.027 2.178 0.030 0.006 0.111

Light Snow -0.2488 0.026 -9.560 0.000 -0.300 -0.198

Mist -0.0578 0.010 -5.574 0.000 -0.078 -0.037

=====

=====

Omnibus: 66.344 Durbin-Watson: 2.071

localhost:8888/notebooks/Downloads/LinearRegression_Nitin_Jain.ipynb#

36/61

Prob(Omnibus):0.000Jarque-Bera (JB):161.565

Skew:-0.676Prob(JB):8.25e-36

Kurtosis:5.403Cond. No.25.4

=====

=====

Notes:

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

In [52]:

```
# VIF for 6th iteration

get_vif_dataframe(X_train_rfe_6)
```

Out[52]:

	Features	VIF
3	humidity	32.81
2	temp	19.97
1	workingday	18.57
5	spring	5.04
11	Sat	4.95
4	windspeed	4.92
12	Sun	4.79
7	winter	3.76
6	summer	3.03
14	Mist	2.31
0	year	2.10
8	Jan	1.68
9	Jul	1.60
10	Sep	1.38
13	Light Snow	1.25

In [53]:

```
# 7th iteration for the model fitting  
# As the feature 'May' is not significant due to high VIF), so will remove that  
  
X_train_rfe_7 = X_train_rfe_6.drop(['humidity'], axis=1)  
X_train_lm_7 = sm.add_constant(X_train_rfe_7)  
lm_7 = sm.OLS(y_train, X_train_lm_7).fit()  
print(lm_7.summary())
```

OLS Regression Results

```

=====
====
Dep. Variable:          cnt    R-squared:
0.843
Model:                  OLS    Adj. R-squared:
0.839
Method:                 Least Squares    F-statistic:          1
90.3
Date:                   Wed, 16 Aug 2023    Prob (F-statistic):      7.33e
-189
Time:                   16:53:38    Log-Likelihood:          51
1.16
No. Observations:       510    AIC:                      -9
92.3
Df Residuals:           495    BIC:                      -9
28.8
Df Model:               14
Covariance Type:        nonrobust
=====
=====

```

```

=====
====
          coef    std err          t      P>|t|      [0.025      0.
975]
-----
-----

```

```

----
const      0.1175    0.039    3.012    0.003    0.041
0.194
year       0.2344    0.008   29.019    0.000    0.218
0.250
workingday  0.1027    0.026    3.996    0.000    0.052
0.153
temp       0.4728    0.034   14.037    0.000    0.407
0.539
windspeed  -0.1563    0.025   -6.292    0.000   -0.205    -
0.107
spring     -0.0597    0.021   -2.861    0.004   -0.101    -
0.019
summer     0.0434    0.015    2.890    0.004    0.014
0.073
winter     0.0797    0.017    4.650    0.000    0.046
0.113
Jan        -0.0389    0.018   -2.215    0.027   -0.073    -
0.004
Jul        -0.0482    0.018   -2.635    0.009   -0.084    -
0.012
Sep        0.0753    0.017    4.522    0.000    0.043
0.108
Sat        0.1146    0.027    4.222    0.000    0.061
0.168
Sun        0.0562    0.027    2.058    0.040    0.003
0.110
Light Snow -0.2917    0.024  -12.027    0.000   -0.339    -
0.244
Mist       -0.0826    0.009   -9.592    0.000   -0.100    -
0.066
=====
=====

```

```

====
Omnibus:          67.959    Durbin-Watson:
2.066
Prob(Omnibus):    0.000    Jarque-Bera (JB):      16
6.078

```

Skew: -0.690 Prob(JB): 8.64

e-37

Kurtosis: 5.431 Cond. No.

23.0

=====

=====

Notes:

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

In [54]:

```
# VIF dataframe for 7th iteration

get_vif_dataframe(X_train_rfe_7)
```

Out[54]:

	Features	VIF
1	workingday	16.57
2	temp	13.12
3	windspeed	4.79
10	Sat	4.54
11	Sun	4.28
4	spring	4.22
6	winter	2.80
5	summer	2.75
0	year	2.08
7	Jan	1.65
8	Jul	1.60
13	Mist	1.59
9	Sep	1.35
12	Light Snow	1.09

In [55]:

```
# 8th iteration for the model fitting  
# As the feature 'workingday' is not significant), so will remove that  
  
X_train_rfe_8 = X_train_rfe_7.drop(['workingday'], axis=1)  
X_train_lm_8 = sm.add_constant(X_train_rfe_8)  
lm_8 = sm.OLS(y_train, X_train_lm_8).fit()  
print(lm_8.summary())
```

OLS Regression Results

```

=====
====
Dep. Variable:          cnt    R-squared:
0.838
Model:                  OLS    Adj. R-squared:
0.834
Method:                 Least Squares    F-statistic:          1
97.7
Date:                  Wed, 16 Aug 2023    Prob (F-statistic):      1.27e
-186
Time:                  16:53:38    Log-Likelihood:          50
3.07
No. Observations:      510    AIC:                      -9
78.1
Df Residuals:          496    BIC:                      -9
18.9
Df Model:              13
Covariance Type:       nonrobust
=====
====

```

	coef	std err	t	P> t	[0.025	0.
975]						

const	0.2172	0.030	7.138	0.000	0.157	
0.277						
year	0.2349	0.008	28.657	0.000	0.219	
0.251						
temp	0.4737	0.034	13.856	0.000	0.407	
0.541						
windspeed	-0.1586	0.025	-6.293	0.000	-0.208	-
0.109						
spring	-0.0622	0.021	-2.936	0.003	-0.104	-
0.021						
summer	0.0437	0.015	2.868	0.004	0.014	
0.074						
winter	0.0767	0.017	4.411	0.000	0.043	
0.111						
Jan	-0.0398	0.018	-2.231	0.026	-0.075	-
0.005						
Jul	-0.0474	0.019	-2.551	0.011	-0.084	-
0.011						
Sep	0.0717	0.017	4.247	0.000	0.039	
0.105						
Sat	0.0159	0.011	1.391	0.165	-0.007	
0.038						
Sun	-0.0425	0.012	-3.600	0.000	-0.066	-
0.019						
Light Snow	-0.2871	0.025	-11.675	0.000	-0.335	-
0.239						
Mist	-0.0807	0.009	-9.247	0.000	-0.098	-
0.064						
=====						
====						
Omnibus:	82.611		Durbin-Watson:			
2.013						
Prob(Omnibus):	0.000		Jarque-Bera (JB):		21	
7.073						
Skew:	-0.806		Prob(JB):		7.30	
e-48						

Kurtosis: 5.760 Cond. No.
17.9

=====

=====

Notes:

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

In [56]:

```
# VIF dataframe for 8th iteration  
  
get_vif_dataframe(X_train_rfe_8)
```

Out[56]:

	Features	VIF
1	temp	5.22
2	windspeed	4.64
3	spring	2.78
4	summer	2.24
0	year	2.07
5	winter	1.83
6	Jan	1.61
7	Jul	1.59
12	Mist	1.56
8	Sep	1.33
9	Sat	1.22
10	Sun	1.21
11	Light Snow	1.08

In [57]:

```
# Model fitting for the 9th iteration  
# As the feature 'temp' is not significant), so will remove that  
  
X_train_rfe_9 = X_train_rfe_8.drop(['temp'], axis=1)  
X_train_lm_9 = sm.add_constant(X_train_rfe_9)  
lm_9 = sm.OLS(y_train, X_train_lm_9).fit()  
print(lm_9.summary())
```

OLS Regression Results

```

=====
====
Dep. Variable:          cnt    R-squared:
0.776
Model:                  OLS    Adj. R-squared:
0.770
Method:                 Least Squares    F-statistic:          1
43.2
Date:                   Wed, 16 Aug 2023    Prob (F-statistic):      1.19e
-152
Time:                   16:53:38    Log-Likelihood:          41
9.63
No. Observations:       510    AIC:                      -8
13.3
Df Residuals:           497    BIC:                      -7
58.2
Df Model:                12
Covariance Type:        nonrobust
=====

```

```

=====
====
              coef    std err          t      P>|t|      [0.025    0.
975]
-----
----
const          0.5905     0.017    35.515     0.000     0.558
0.623
year           0.2483     0.010    25.933     0.000     0.230
0.267
windspeed     -0.1903     0.030     -6.444     0.000    -0.248    -
0.132
spring        -0.2632     0.018   -14.507     0.000    -0.299    -
0.228
summer        -0.0439     0.016     -2.690     0.007    -0.076    -
0.012
winter        -0.0783     0.016     -5.004     0.000    -0.109    -
0.048
Jan           -0.1035     0.020     -5.099     0.000    -0.143    -
0.064
Jul           -0.0089     0.022     -0.412     0.680    -0.051
0.034
Sep            0.0671     0.020     3.380     0.001     0.028
0.106
Sat            0.0124     0.013     0.923     0.356    -0.014
0.039
Sun           -0.0439     0.014     -3.162     0.002    -0.071    -
0.017
Light Snow    -0.2997     0.029   -10.367     0.000    -0.357    -
0.243
Mist          -0.0877     0.010     -8.552     0.000    -0.108    -
0.068
=====

```

```

=====
====
Omnibus:              46.366    Durbin-Watson:
1.987
Prob(Omnibus):         0.000    Jarque-Bera (JB):          9
9.978
Skew:                  -0.514    Prob(JB):                  1.95
e-22
Kurtosis:              4.910    Cond. No.
9.29

```

=====

=====

Notes:

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

In [58]:

```
# VIF dataframe for 9th iteration

get_vif_dataframe(X_train_rfe_9)
```

Out[58]:

	Features	VIF
1	windspeed	3.91
2	spring	2.78
3	summer	2.04
0	year	1.84
4	winter	1.79
5	Jan	1.60
11	Mist	1.52
6	Jul	1.22
8	Sat	1.21
9	Sun	1.20
7	Sep	1.16
10	Light Snow	1.08

In [59]:

```
# Model fitting for 10th iteration  
# As the feature 'JUL' is not significant), so will remove that  
  
X_train_rfe_10 = X_train_rfe_9.drop(['Jul'], axis=1)  
X_train_lm_10 = sm.add_constant(X_train_rfe_10)  
lm_10 = sm.OLS(y_train, X_train_lm_10).fit()  
print(lm_10.summary())
```

OLS Regression Results

```

=====
====
Dep. Variable:          cnt    R-squared:
0.776
Model:                  OLS    Adj. R-squared:
0.771
Method:                  Least Squares    F-statistic:          1
56.5
Date:                    Wed, 16 Aug 2023    Prob (F-statistic):      1.01e
-153
Time:                    16:53:38    Log-Likelihood:          41
9.54
No. Observations:        510    AIC:                      -8
15.1
Df Residuals:            498    BIC:                      -7
64.3
Df Model:                11
Covariance Type:         nonrobust
=====
=====

```

```

=====
====
              coef      std err          t      P>|t|      [0.025      0.
975]
-----
-----

```

```

----
const          0.5872      0.015      40.182      0.000      0.559
0.616
year           0.2483      0.010      25.959      0.000      0.230
0.267
windspeed      -0.1902      0.030      -6.447      0.000      -0.248      -
0.132
spring         -0.2600      0.016     -15.849      0.000      -0.292      -
0.228
summer         -0.0407      0.014      -2.830      0.005      -0.069      -
0.012
winter         -0.0753      0.014      -5.442      0.000      -0.102      -
0.048
Jan            -0.1035      0.020      -5.106      0.000      -0.143      -
0.064
Sep            0.0696      0.019       3.683      0.000      0.032
0.107
Sat            0.0123      0.013       0.916      0.360      -0.014
0.039
Sun           -0.0442      0.014      -3.184      0.002      -0.071      -
0.017
Light Snow     -0.2999      0.029     -10.380      0.000      -0.357      -
0.243
Mist           -0.0874      0.010      -8.551      0.000      -0.107      -
0.067
=====
=====

```

```

====
Omnibus:          46.458    Durbin-Watson:
1.991
Prob(Omnibus):    0.000    Jarque-Bera (JB):          9
9.734
Skew:             -0.516    Prob(JB):                  2.20
e-22
Kurtosis:         4.905    Cond. No.
8.92
=====
=====

```


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

In [60]:

```
# VIF for the 10th iteration

get_vif_dataframe(X_train_rfe_10)
```

Out[60]:

	Features	VIF
1	windspeed	3.60
2	spring	2.58
3	summer	1.87
0	year	1.78
4	winter	1.66
5	Jan	1.60
10	Mist	1.51
7	Sat	1.20
8	Sun	1.18
6	Sep	1.14
9	Light Snow	1.08

In [61]:

```
# Model fitting for 11th iteration  
# As the feature 'Sat' is not significant), so will remove that  
  
X_train_rfe_11 = X_train_rfe_10.drop(['Sat'], axis=1)  
X_train_lm_11 = sm.add_constant(X_train_rfe_11)  
lm_11 = sm.OLS(y_train, X_train_lm_11).fit()  
print(lm_11.summary())
```

OLS Regression Results						
=====						
=====						
Dep. Variable:	cnt		R-squared:			
0.775						
Model:	OLS		Adj. R-squared:			
0.771						
Method:	Least Squares		F-statistic:		1	
72.1						
Date:	Wed, 16 Aug 2023		Prob (F-statistic):		1.14e	
-154						
Time:	16:53:38		Log-Likelihood:		41	
9.11						
No. Observations:	510		AIC:		-8	
16.2						
Df Residuals:	499		BIC:		-7	
69.6						
Df Model:	10					
Covariance Type:	nonrobust					
=====						
=====						
	coef	std err	t	P> t	[0.025	0.
975]						

const	0.5891	0.014	40.705	0.000	0.561	
0.617						
year	0.2481	0.010	25.947	0.000	0.229	
0.267						
windspeed	-0.1889	0.029	-6.411	0.000	-0.247	-
0.131						
spring	-0.2599	0.016	-15.845	0.000	-0.292	-
0.228						
summer	-0.0408	0.014	-2.837	0.005	-0.069	-
0.013						
winter	-0.0750	0.014	-5.423	0.000	-0.102	-
0.048						
Jan	-0.1033	0.020	-5.095	0.000	-0.143	-
0.063						
Sep	0.0695	0.019	3.678	0.000	0.032	
0.107						
Sun	-0.0464	0.014	-3.399	0.001	-0.073	-
0.020						
Light Snow	-0.2997	0.029	-10.376	0.000	-0.356	-
0.243						
Mist	-0.0874	0.010	-8.550	0.000	-0.107	-
0.067						
=====						
=====						
Omnibus:	44.678		Durbin-Watson:			
1.989						
Prob(Omnibus):	0.000		Jarque-Bera (JB):		10	
0.042						
Skew:	-0.483		Prob(JB):		1.89	
e-22						
Kurtosis:	4.943		Cond. No.			
8.86						
=====						
=====						

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

In [62]:

```
# VIF for 11th iteration

get_vif_dataframe(X_train_rfe_11)
```

Out[62]:

	Features	VIF
1	windspeed	3.52
2	spring	2.58
3	summer	1.86
0	year	1.78
4	winter	1.64
5	Jan	1.59
9	Mist	1.51
7	Sun	1.15
6	Sep	1.13
8	Light Snow	1.08

Observation

- We see, model lm_11, is having all the significant feature and having VIF <5. So, we will proceed with this model.
- R square and the adjusted R square values are: 0.775 and 0.771, which are pretty good to show the variation.

In [63]:

```
# Get the prediction on X_train_lm_11

y_train_cnt = lm_11.predict(X_train_lm_11)
y_train_cnt
```

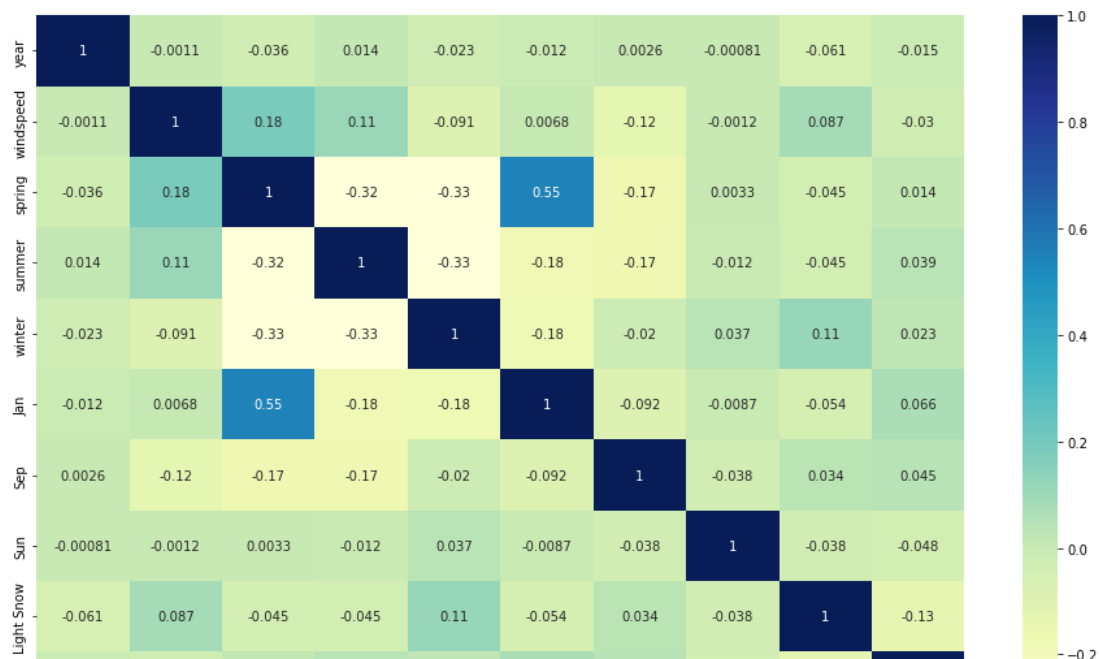
Out[63]:

```
653    0.705298
576    0.787127
426    0.441624
728    0.405578
482    0.673354
...
526    0.641783
578    0.801776
53     0.308135
350    0.345105
79     0.360795
Length: 510, dtype: float64
```

In [64]:

```
# Plot heatmap for final train data after feature removal
```

```
plt.figure(figsize = (16, 10))
sns.heatmap(X_train_rfe_11.corr(), annot = True, cmap="YlGnBu")
plt.show()
```



Observation

- We see all the feature, which are relavent are having no multi-collinearity.

Plot histogram for Error terms

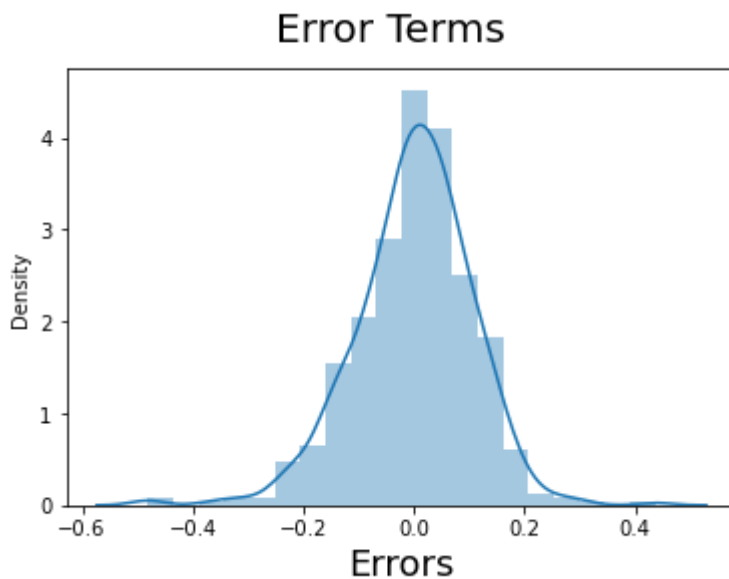
In [65]:

```
# Plot the histogram of the error terms
# We see, mean is almost zero and this should be the case.

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

Out[65]:

Text(0.5, 0, 'Errors')



Prepare the test set based on the train set

In [66]:

```
# Select the columns to be transformed in the test set

num_var = ['temp', 'abs-temp', 'humidity', 'windspeed', 'cnt']

df_test[num_var] = scaler.transform(df_test[num_var])
```

In [67]:

```
# Drop target and abs-temp from the test set

X_test = df_test.drop(['cnt', 'abs-temp'], axis=1)
y_test = df_test[['cnt']]
```

In [68]:

Add a constant for test dataset`X_test_lm_11 = sm.add_constant(X_test)`

In [69]:

Select required columns for X_test_lm_11 WHICH are available in X_train_lm_11`X_test_lm_11 = X_test_lm_11[X_train_lm_11.columns]``X_test_lm_11`

Out[69]:

	const	year	windspeed	spring	summer	winter	Jan	Sep	Sun	Light	Snow	Mist
184	1.0	0	0.084219	0	0	0	0	0	0		0	1
535	1.0	1	0.153728	0	1	0	0	0	0		0	0
299	1.0	0	0.334206	0	0	1	0	0	0		0	1
221	1.0	0	0.339570	0	0	0	0	0	0		0	0
152	1.0	0	0.537414	0	1	0	0	0	0		0	0
...
400	1.0	1	0.287411	1	0	0	0	0	1		0	1
702	1.0	1	0.283397	0	0	1	0	0	0		0	0
127	1.0	0	0.069510	0	1	0	0	0	1		0	0
640	1.0	1	0.052115	0	0	1	0	0	0		0	1
72	1.0	0	0.203418	1	0	0	0	0	0		0	0

219 rows × 11 columns

Get the prediction for test set

In [70]:

*# we will use lm_11 model as this was the best model.**# Get the prediction*`y_pred = lm_11.predict(X_test_lm_11)`

In [71]:

```
# Predicted dataframe
```

```
y_pred
```

Out[71]:

```
184    0.485778
535    0.767296
299    0.363545
221    0.524917
152    0.446750
...
400    0.389174
702    0.708584
127    0.488750
640    0.664898
72     0.290748
Length: 219, dtype: float64
```

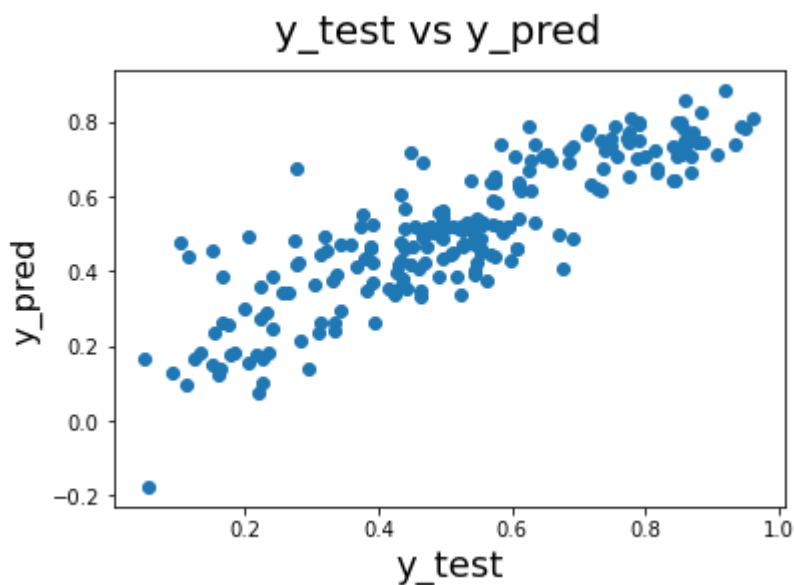
In [72]:

```
# Plot test and pred datapoints
```

```
fig = plt.figure()
plt.scatter(y_test, y_pred)
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[72]:

```
Text(0, 0.5, 'y_pred')
```



In [73]:

```
# Get the ccpr plots for some feature

sm.graphics.plot_ccpr(lm_11, 'windspeed')
plt.show()

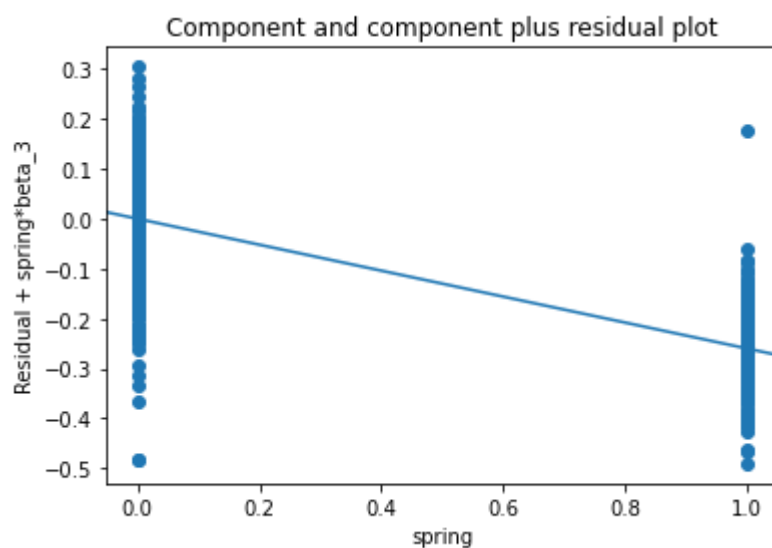
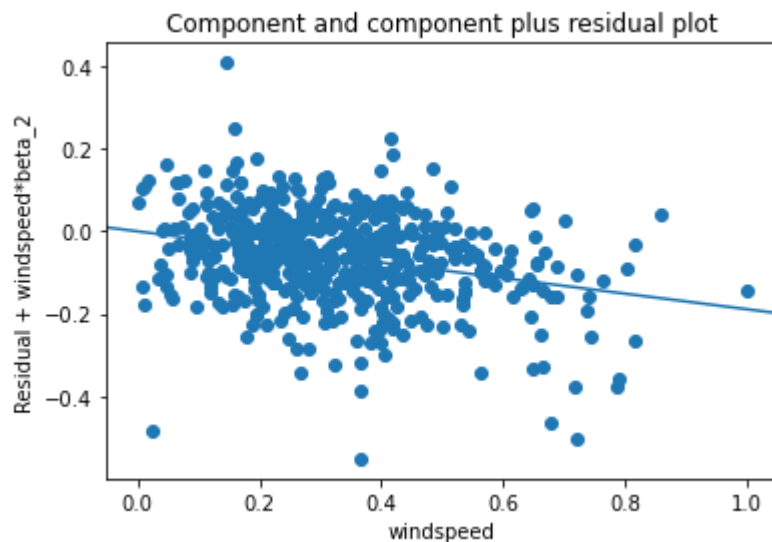
sm.graphics.plot_ccpr(lm_11, 'spring')
plt.show()

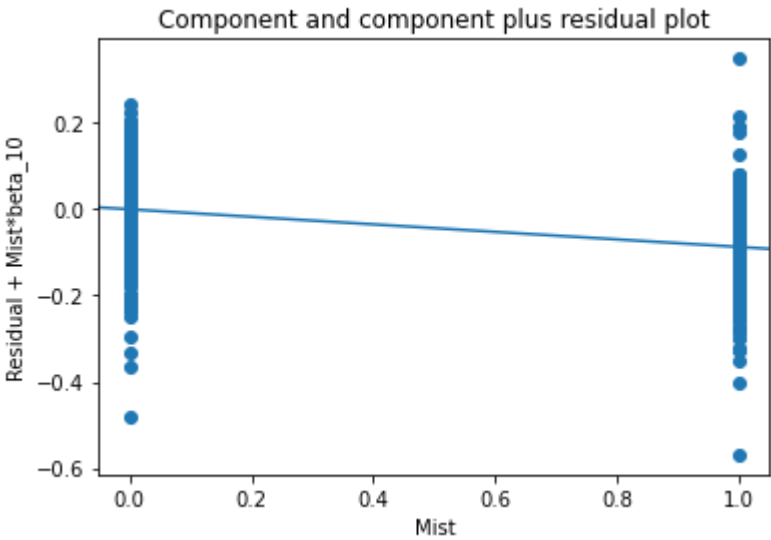
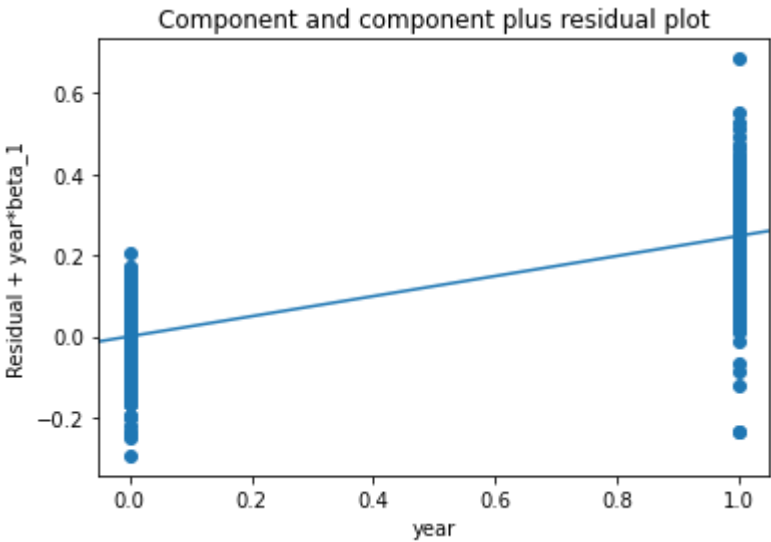
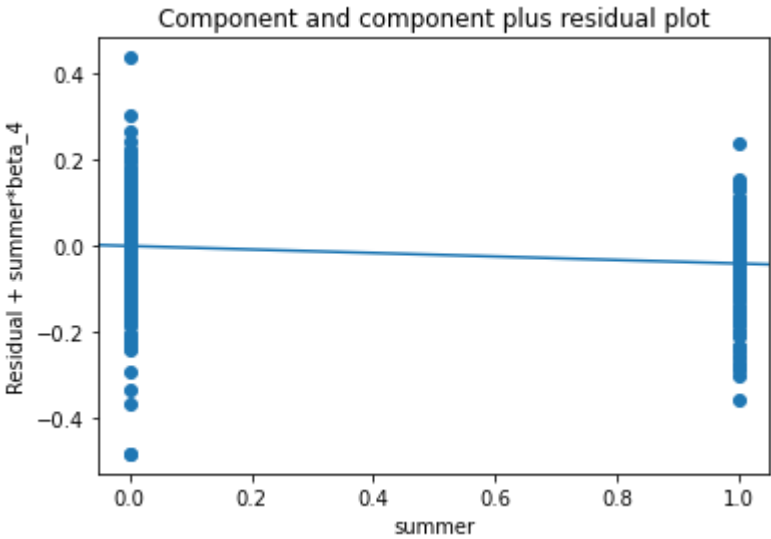
sm.graphics.plot_ccpr(lm_11, 'summer')
plt.show()

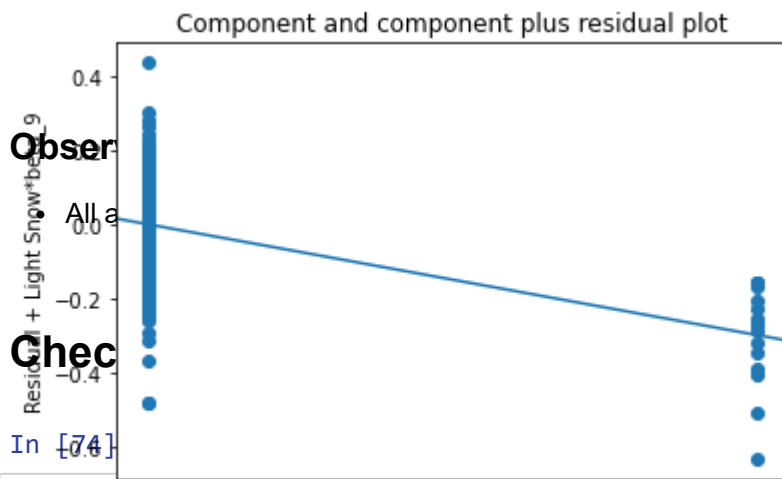
sm.graphics.plot_ccpr(lm_11, 'year')
plt.show()

sm.graphics.plot_ccpr(lm_11, 'Mist')
plt.show()

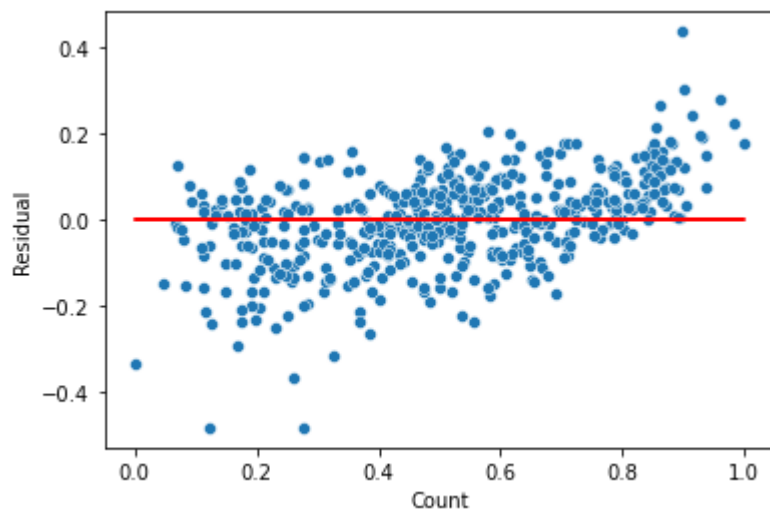
sm.graphics.plot_ccpr(lm_11, 'Light Snow')
plt.show()
```







```
residual = y_train - y_train_cnt*beta_1
sns.scatterplot(y_train, residual)
plt.plot(y_train, (y_train - y_train), '-r')
plt.xlabel('Count')
plt.ylabel('Residual')
plt.show()
```



Observation

- We see that residual is almost zero and constant.

In [75]:

```
## Get some derived metrics for test and train dataset
# Get r2 score for test
from sklearn.metrics import r2_score
r2_test = r2_score(y_test, y_pred)
print(r2_test)
```

0.7395825489781629

In [76]:

```
# Get r2 score for test
from sklearn.metrics import r2_score
r2_train = r2_score(y_train, y_train_cnt)
print(r2_train)
```

0.7752016324072822

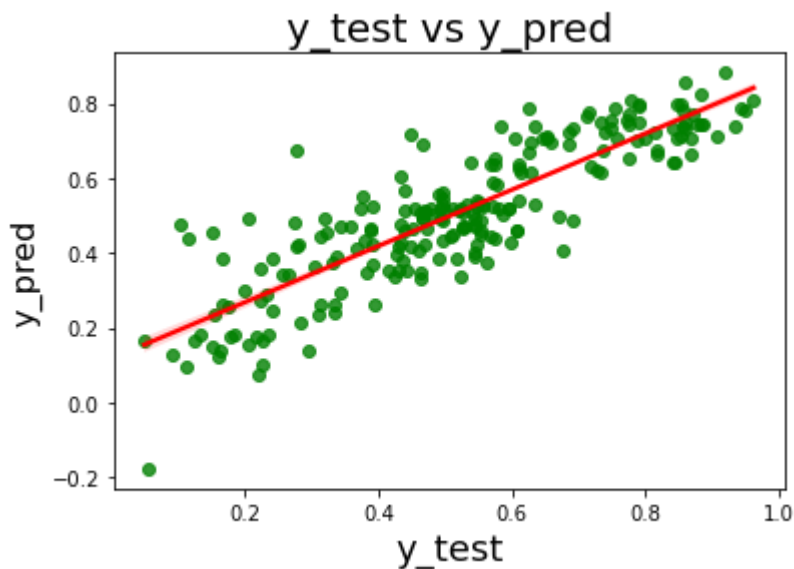
Observation

- We see that the variation in test and train dataset is almost uniform w.r.t prediction.

In [77]:

```
# Plot y_test vs y_pred with the best fit line

plt.figure()
sns.regplot(x=y_test, y=y_pred, ci=68, fit_reg=True,
            scatter_kws={"color": "green"}, line_kws={"color": "red"})
plt.title('y_test vs y_pred', fontsize=20)
plt.xlabel('y_test', fontsize=18)
plt.ylabel('y_pred', fontsize=16)
plt.show()
```



In [78]:

```
# Final model coefficients
```

```
print(lm_11.params)
```

```
const      0.589066
year       0.248063
windspeed  -0.188912
spring     -0.259890
summer     -0.040792
winter     -0.075008
Jan        -0.103274
Sep        0.069520
Sun        -0.046392
Light Snow -0.299689
Mist       -0.087378
dtype: float64
```

Final Model Equation

We can see that the equation of our best fitted line is:

$$cnt = 0.5891 + 0.2481 \times year - 0.1889 \times windspeed - 0.2599 \times spring - 0.0408 \times summer - 0.07 \times Sep - 0.0464 \times Sun - 0.2997 \times LightSnow - 0.0874 \times Mist$$