ml(linear regression)

September 15, 2025

0.0.1 1.Reading and Understanding the Data

[1]:
```python
import warnings
warnings.filterwarnings('ignore')
```

Import the libraries

[3]:
```python
import numpy as np
import pandas as pd
import matplotlib as plt
from sklearn import linear_model
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Read and Understanding the Data

[10]:
```python
# Read the given CSV file, and view some sample records
df=pd.read_csv('day.csv')
df
```

[10]:
```
     instant      dteday  season  yr  mnth  holiday  weekday  workingday  \
0          1  01-01-2018       1   0     1        0        1           1
1          2  02-01-2018       1   0     1        0        2           1
2          3  03-01-2018       1   0     1        0        3           1
3          4  04-01-2018       1   0     1        0        4           1
4          5  05-01-2018       1   0     1        0        5           1
..       …         …       …  ..    …      …        …         …
725      726  27-12-2019       1   1    12        0        5           1
726      727  28-12-2019       1   1    12        0        6           0
727      728  29-12-2019       1   1    12        0        0           0
728      729  30-12-2019       1   1    12        0        1           1
729      730  31-12-2019       1   1    12        0        2           1

     weathersit       temp    atemp      hum  windspeed  casual  registered  \
0             2  14.110847  18.18125  80.5833  10.749882     331         654
1             2  14.902598  17.68695  69.6087  16.652113     131         670
2             1   8.050924   9.47025  43.7273  16.636703     120        1229
3             1   8.200000  10.60610  59.0435  10.739832     108        1454
```

```
4              1   9.305237  11.46350  43.6957  12.522300      82         1518
..           ...        ...       ...      ...        ...      ...          ...
725            2  10.420847  11.33210  65.2917  23.458911     247         1867
726            2  10.386653  12.75230  59.0000  10.416557     644         2451
727            2  10.386653  12.12000  75.2917   8.333661     159         1182
728            1  10.489153  11.58500  48.3333  23.500518     364         1432
729            2   8.849153  11.17435  57.7500  10.374682     439         2290

      cnt
0     985
1     801
2    1349
3    1562
4    1600
..    ...
725  2114
726  3095
727  1341
728  1796
729  2729

[730 rows x 16 columns]
```

[9]:
```python
df.columns
```

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

[11]:
```python
#Determining the number of rows and columns
df.shape
```

[11]:
```
(730, 16)
```

[13]:
```python
#summary of all the numeric columns in the dataset
df.describe()
```

[13]:
```
          instant      season          yr        mnth     holiday     weekday  \
count  730.000000  730.000000  730.000000  730.000000  730.000000  730.000000
mean   365.500000    2.498630    0.500000    6.526027    0.028767    2.995890
std    210.877136    1.110184    0.500343    3.450215    0.167266    2.000339
min      1.000000    1.000000    0.000000    1.000000    0.000000    0.000000
25%    183.250000    2.000000    0.000000    4.000000    0.000000    1.000000
50%    365.500000    3.000000    0.500000    7.000000    0.000000    3.000000
75%    547.750000    3.000000    1.000000   10.000000    0.000000    5.000000
max    730.000000    4.000000    1.000000   12.000000    1.000000    6.000000
```

```
       workingday  weathersit        temp       atemp         hum   windspeed  \
count  730.000000  730.000000  730.000000  730.000000  730.000000  730.000000
mean     0.690411    1.394521   20.319259   23.726322   62.765175   12.763620
std      0.462641    0.544807    7.506729    8.150308   14.237589    5.195841
min      0.000000    1.000000    2.424346    3.953480    0.000000    1.500244
25%      0.000000    1.000000   13.811885   16.889713   52.000000    9.041650
50%      1.000000    1.000000   20.465826   24.368225   62.625000   12.125325
75%      1.000000    2.000000   26.880615   30.445775   72.989575   15.625589
max      1.000000    3.000000   35.328347   42.044800   97.250000   34.000021

            casual   registered          cnt
count   730.000000   730.000000   730.000000
mean    849.249315  3658.757534  4508.006849
std     686.479875  1559.758728  1936.011647
min       2.000000    20.000000    22.000000
25%     316.250000  2502.250000  3169.750000
50%     717.000000  3664.500000  4548.500000
75%    1096.500000  4783.250000  5966.000000
max    3410.000000  6946.000000  8714.000000
```

[15]:

```
#Datatypes of each column
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
```

[17]:
```
#Checking missing values
df.isnull().sum()
```

[17]:
```
instant       0
dteday        0
season        0
yr            0
mnth          0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
casual        0
registered    0
cnt           0
dtype: int64
```

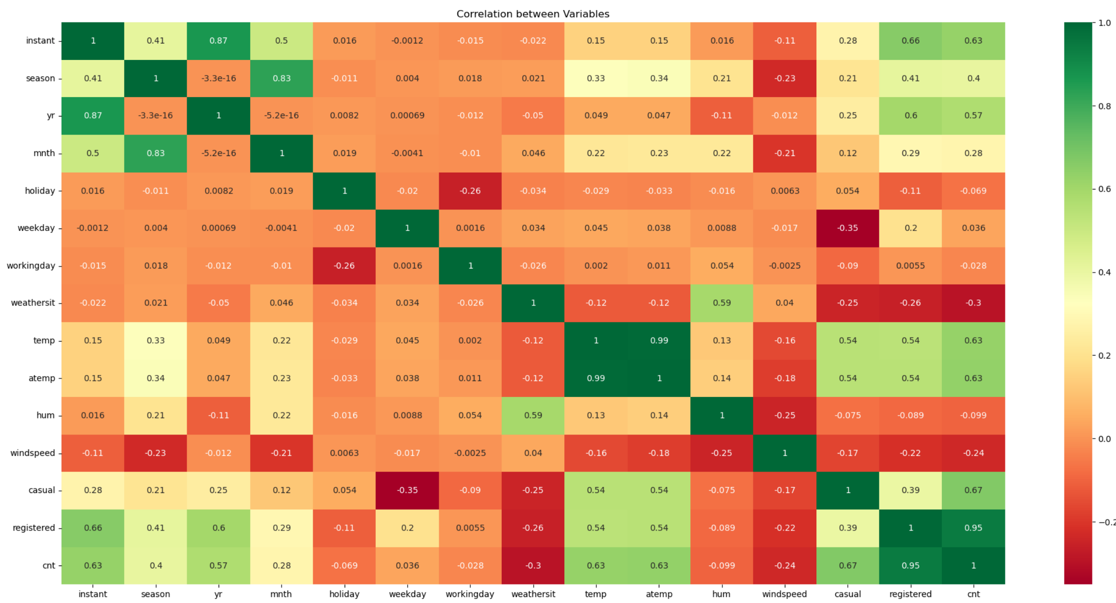
1 2.Data Visualisation

[21]:
```
df.drop(columns=['dteday'],inplace=True)
df
```

[21]:
```
     instant  season  yr  mnth  holiday  weekday  workingday  weathersit  \
0          1       1   0     1        0        1           1           2
1          2       1   0     1        0        2           1           2
2          3       1   0     1        0        3           1           1
3          4       1   0     1        0        4           1           1
4          5       1   0     1        0        5           1           1
..         …      …   ..    …        …        …           …
725      726       1   1    12        0        5           1           2
726      727       1   1    12        0        6           0           2
727      728       1   1    12        0        0           0           2
728      729       1   1    12        0        1           1           1
729      730       1   1    12        0        2           1           2

          temp     atemp      hum  windspeed  casual  registered   cnt
0    14.110847  18.18125  80.5833  10.749882     331         654   985
1    14.902598  17.68695  69.6087  16.652113     131         670   801
2     8.050924   9.47025  43.7273  16.636703     120        1229  1349
3     8.200000  10.60610  59.0435  10.739832     108        1454  1562
4     9.305237  11.46350  43.6957  12.522300      82        1518  1600
..          …         …        …          …       …           …     …
725  10.420847  11.33210  65.2917  23.458911     247        1867  2114
```

```
726  10.386653  12.75230  59.0000  10.416557     644         2451  3095
727  10.386653  12.12000  75.2917   8.333661     159         1182  1341
728  10.489153  11.58500  48.3333  23.500518     364         1432  1796
729   8.849153  11.17435  57.7500  10.374682     439         2290  2729

[730 rows x 15 columns]
```

[23]:
```python
plt.figure(figsize=(25, 12))
sns.heatmap(df.corr(), cmap='RdYlGn', annot = True)
plt.title("Correlation between Variables")
plt.show()
```

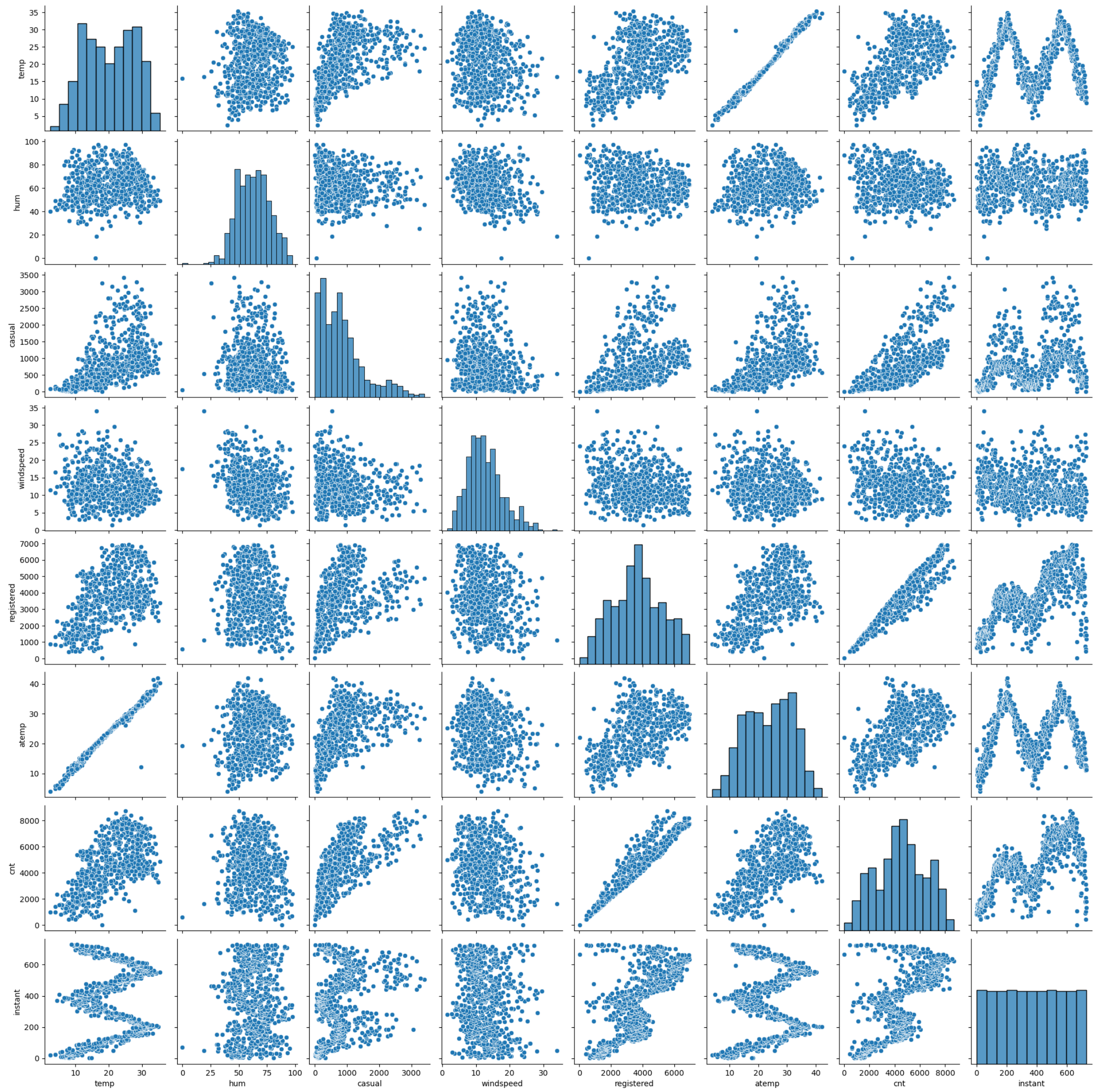


[25]:
```python
df['season']=df.season.map({1: 'spring', 2: 'summer',3:'fall', 4:'winter' })
df['mnth']=df.mnth.map({1:'Jan',2:'Feb',3:'Mar',4:'Apr',5:'May',6:'June',7:
 ↪'July',8:'Aug',9:'Sep',10:'Oct',11:'Nov',12:'Dec'})
df['weathersit']=df.weathersit.map({1: 'Clear',2:'Mist + Cloudy',3:'Light␣
 ↪Snow',4:'Snow + Fog'})
df['weekday']=df.weekday.map({0:'Sun',1:'Mon',2:'Tue',3:'Wed',4:'Thu',5:'Fri',6:
 ↪'Sat'})
df
```

[25]:
```
   instant  season  yr mnth  holiday weekday  workingday     weathersit  \
0        1  spring   0  Jan        0     Mon           1  Mist + Cloudy
1        2  spring   0  Jan        0     Tue           1  Mist + Cloudy
2        3  spring   0  Jan        0     Wed           1          Clear
3        4  spring   0  Jan        0     Thu           1          Clear
4        5  spring   0  Jan        0     Fri           1          Clear
```

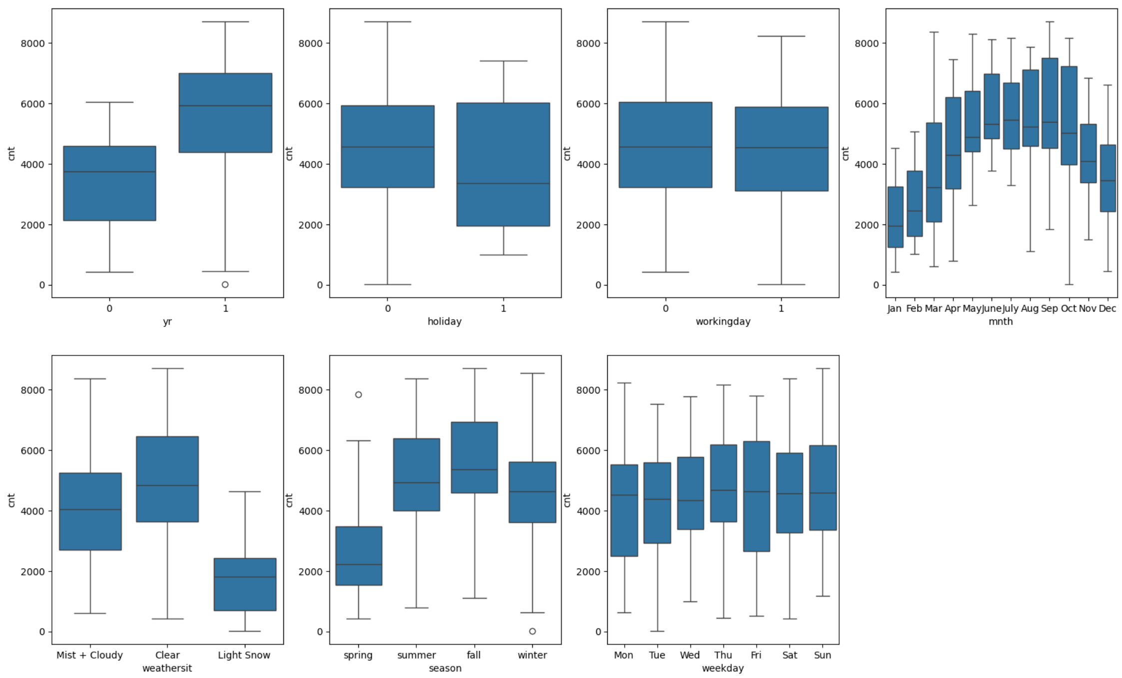
```
..        …     …  ..  …      …     …      …           …
725      726  spring   1  Dec        0     Fri           1  Mist + Cloudy
726      727  spring   1  Dec        0     Sat           0  Mist + Cloudy
727      728  spring   1  Dec        0     Sun           0  Mist + Cloudy
728      729  spring   1  Dec        0     Mon           1          Clear
729      730  spring   1  Dec        0     Tue           1  Mist + Cloudy

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

[730 rows x 15 columns]
```

[27]:
```
#Pairplot for numeric variables
sns.
 ↪pairplot(df,vars=['temp','hum','casual','windspeed','registered','atemp','cnt','instant'])
plt.show()
```

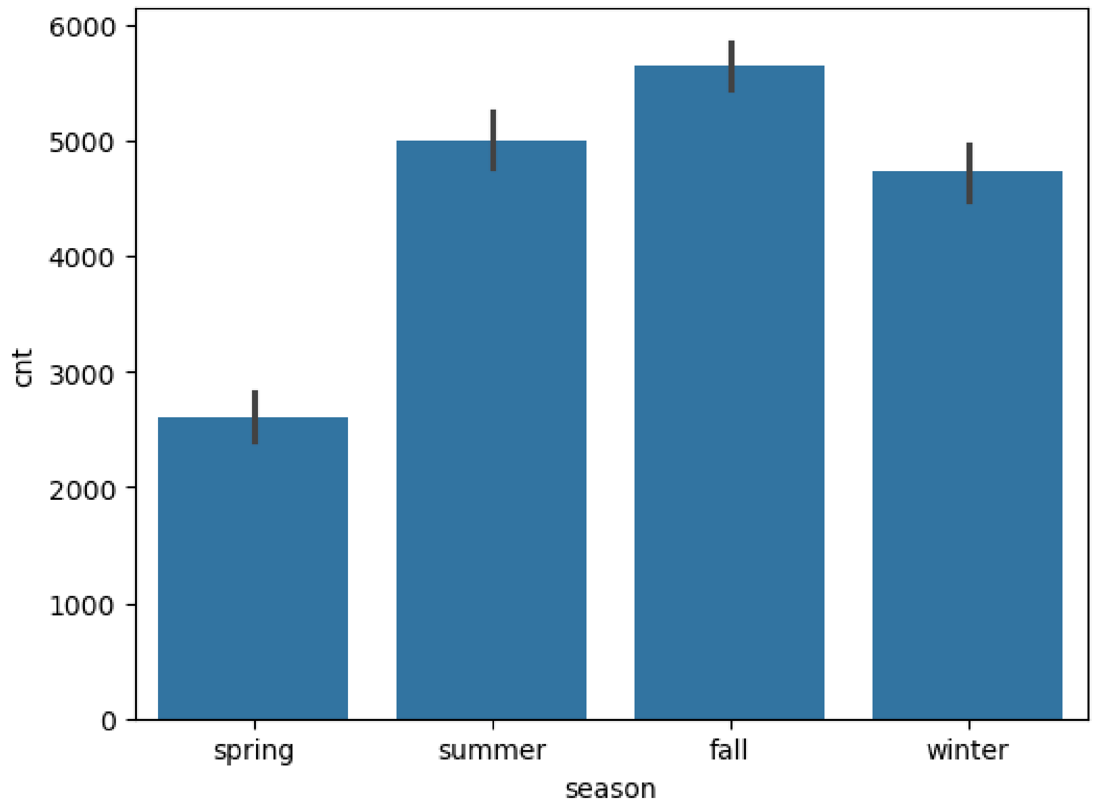



```
[28]: ##Relationship between categorical and continuous variable
plt.figure(figsize=(20, 12))
plt.subplot(2,4,1)
sns.boxplot(x = 'yr', y = 'cnt', data =df)
plt.subplot(2,4,2)
sns.boxplot(x = 'holiday', y = 'cnt', data = df)
plt.subplot(2,4,3)
sns.boxplot(x = 'workingday', y = 'cnt', data = df)
plt.subplot(2,4,4)
sns.boxplot(x = 'mnth', y = 'cnt', data = df)
plt.subplot(2,4,5)
sns.boxplot(x = 'weathersit', y = 'cnt', data = df)
plt.subplot(2,4,6)
```

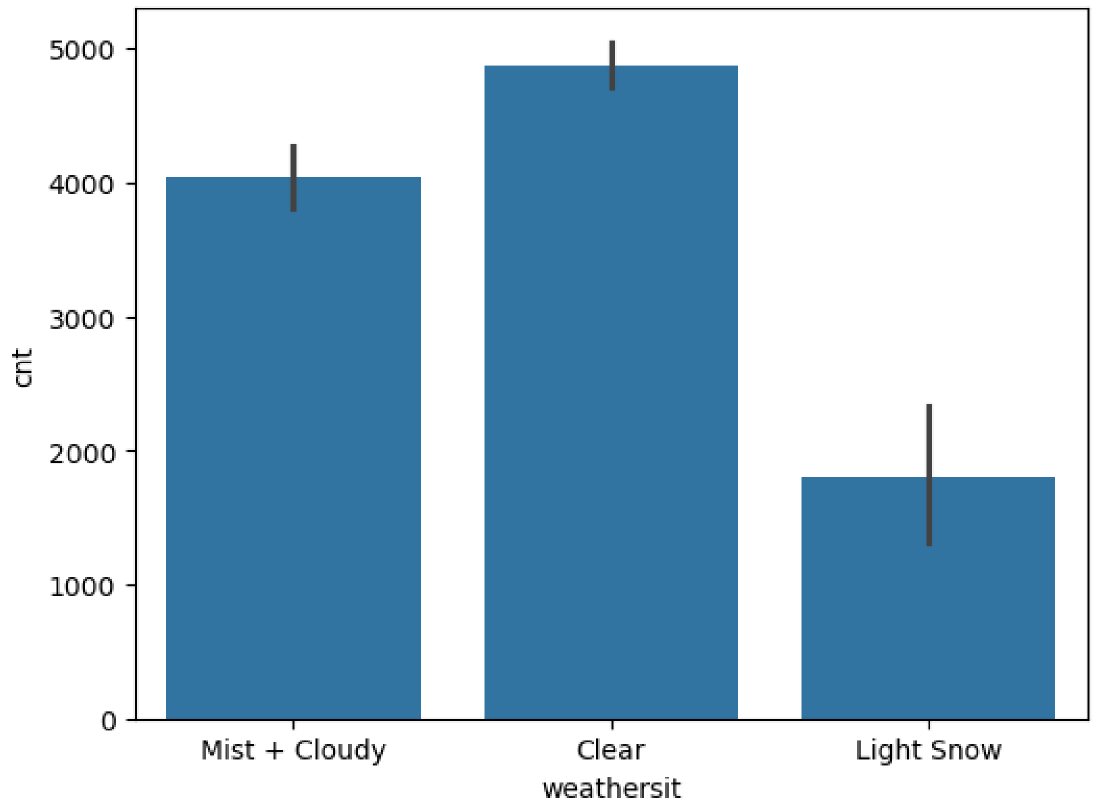
```
sns.boxplot(x = 'season', y = 'cnt', data = df)
plt.subplot(2,4,7)
sns.boxplot(x = 'weekday', y = 'cnt', data = df)
plt.show()
```



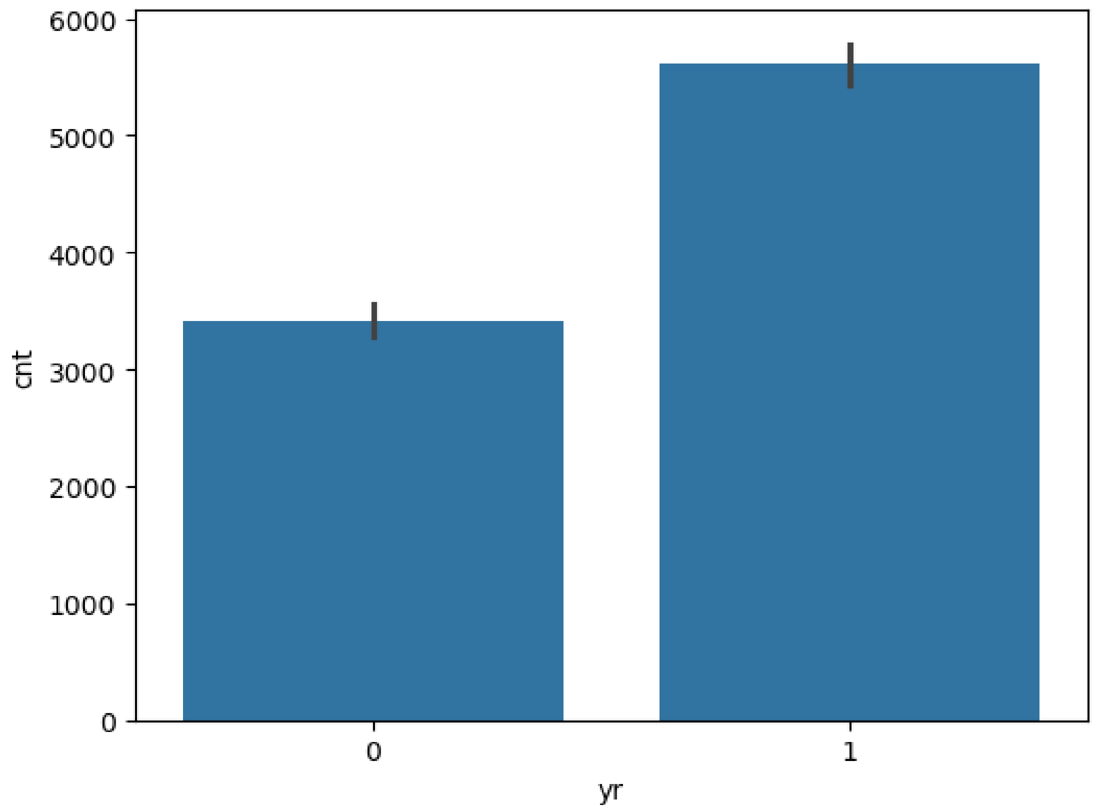
[29]:
```
#Barplot to see relation between season and count of bike rentals
sns.barplot(x='season',y='cnt',data=df)
plt.show()
```



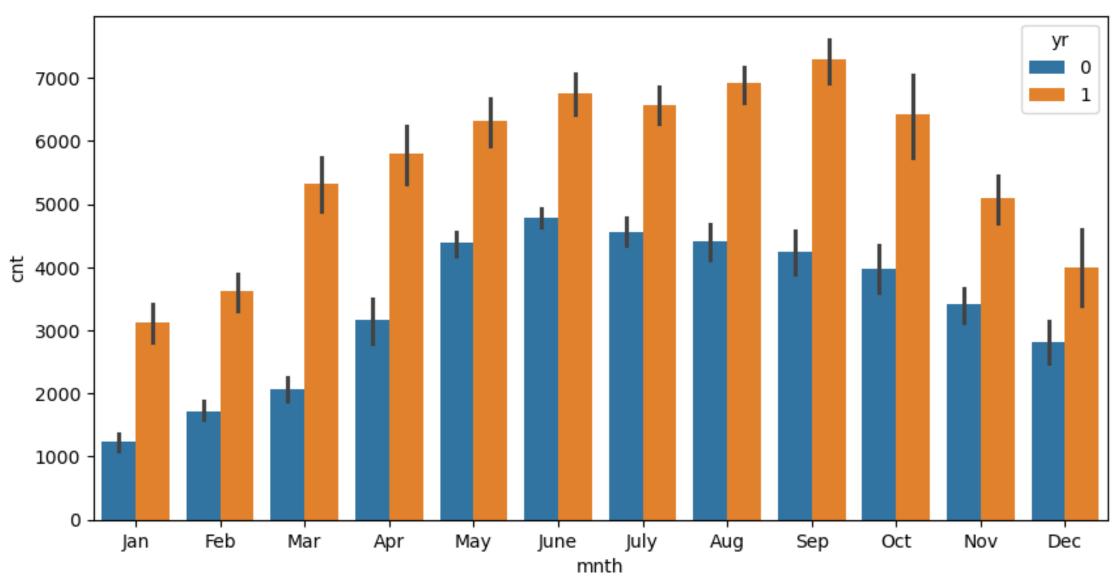
[33]:
```
#Relation between weather and count of bike rentals
sns.barplot(x='weathersit',y='cnt',data=df)
plt.show()
```



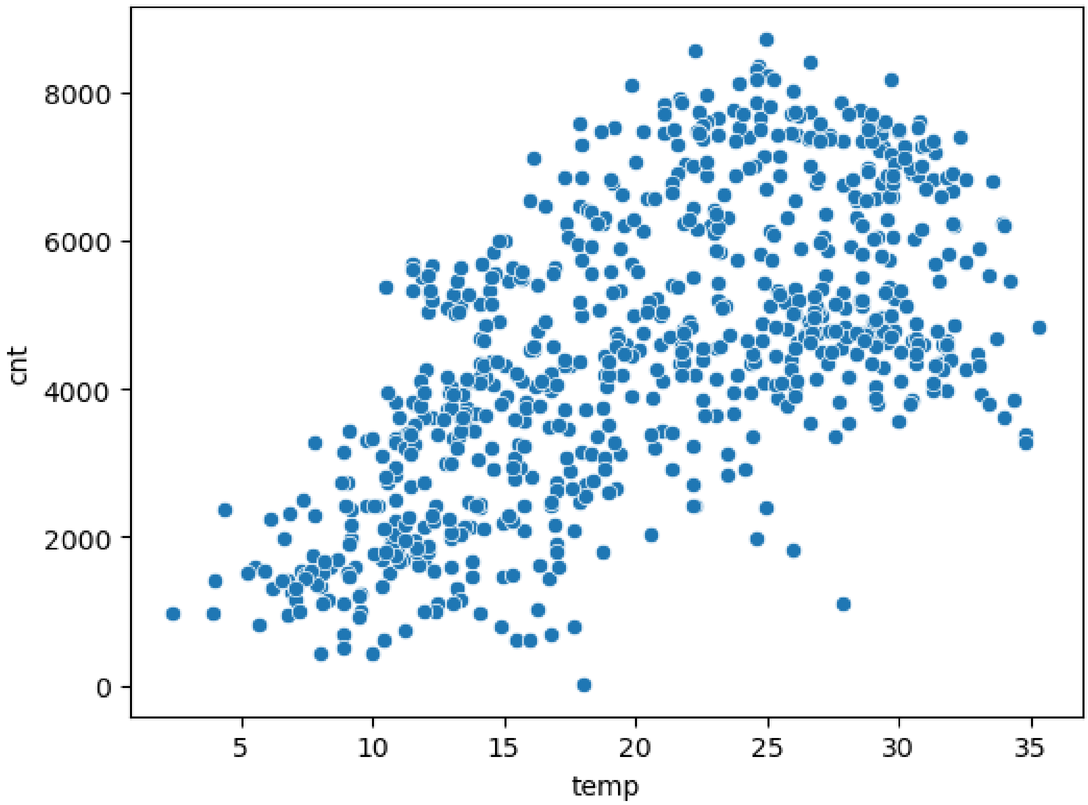
[21]:
```
#Relation between Year and count of bike rentals
sns.barplot(x='yr',y='cnt',data=df)
plt.show()
```



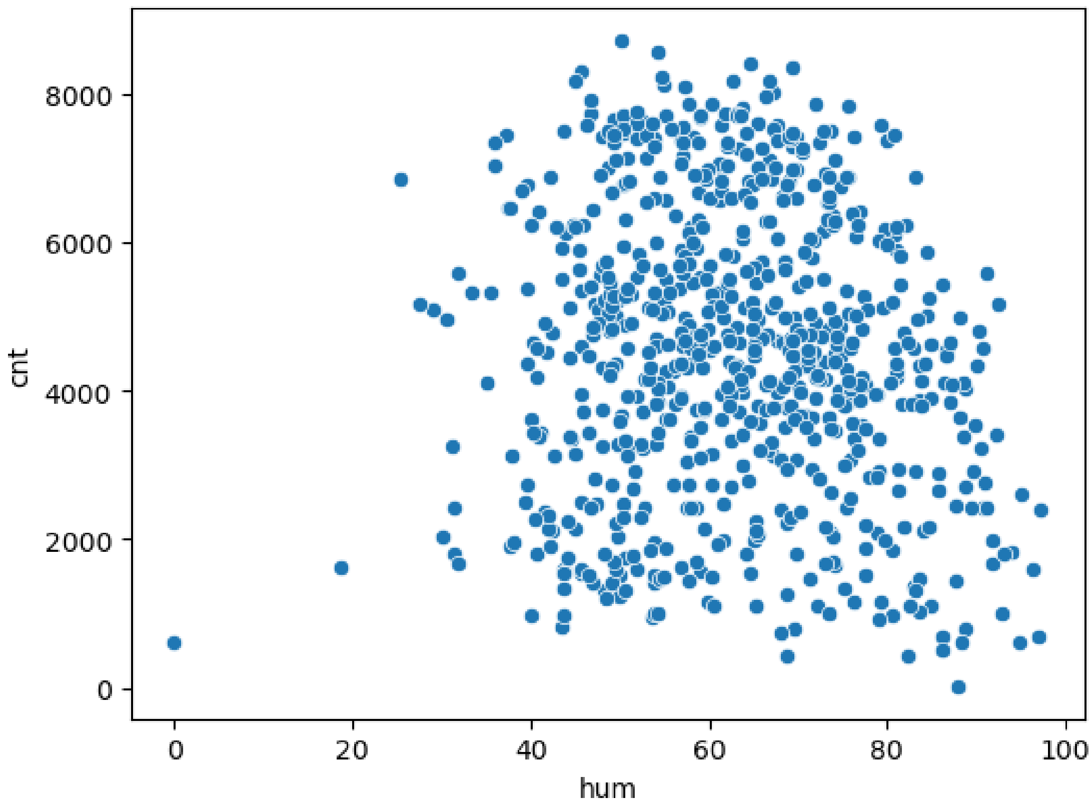
[35]:
```
plt.figure(figsize=(10,5))
sns.barplot(x='mnth',y='cnt',hue='yr',data=df)
plt.show()
```



[37]:
```
#scatter plot for temperature to count
sns.scatterplot(x='temp',y='cnt' ,data=df)
plt.show()
```



[24]:
```
sns.scatterplot(x='hum', y='cnt',data=df)
plt.show()
```



[39]:
```
#drop unnecessary columns
df.drop(['instant','atemp','casual','registered'],axis=1,inplace=True)
df
```

[39]:
```
     season  yr mnth  holiday weekday  workingday      weathersit       temp  \
0    spring   0  Jan        0     Mon           1  Mist + Cloudy  14.110847
1    spring   0  Jan        0     Tue           1  Mist + Cloudy  14.902598
2    spring   0  Jan        0     Wed           1          Clear   8.050924
3    spring   0  Jan        0     Thu           1          Clear   8.200000
4    spring   0  Jan        0     Fri           1          Clear   9.305237
..      …  ..  …        …       …           …              …         …
725  spring   1  Dec        0     Fri           1  Mist + Cloudy  10.420847
726  spring   1  Dec        0     Sat           0  Mist + Cloudy  10.386653
727  spring   1  Dec        0     Sun           0  Mist + Cloudy  10.386653
728  spring   1  Dec        0     Mon           1          Clear  10.489153
729  spring   1  Dec        0     Tue           1  Mist + Cloudy   8.849153

         hum  windspeed   cnt
0    80.5833  10.749882   985
1    69.6087  16.652113   801
2    43.7273  16.636703  1349
```

```
3    59.0435  10.739832  1562
4    43.6957  12.522300  1600
..       …         …    …
725  65.2917  23.458911  2114
726  59.0000  10.416557  3095
727  75.2917   8.333661  1341
728  48.3333  23.500518  1796
729  57.7500  10.374682  2729

[730 rows x 11 columns]
```

[41]:
```python
df.dtypes
```

[41]:
```
season         object
yr              int64
mnth           object
holiday         int64
weekday        object
workingday      int64
weathersit     object
temp          float64
hum           float64
windspeed     float64
cnt             int64
dtype: object
```

2 3.Data Preparation

[43]:
```python
# Get the dummy variables for month, season, weathersit, weekday and Let's␣
↪drop the first column from  using 'drop_first = True'pd.get_dummies(bike.
↪season,drop_first=True)
mnths=pd.get_dummies(df.mnth,drop_first=True,dtype=int)
weekdays=pd.get_dummies(df.weekday,drop_first=True,dtype=int)
weathersit=pd.get_dummies(df.weathersit,drop_first=True,dtype=int)
seasons=pd.get_dummies(df.season,drop_first=True,dtype=int)
```

[29]:
```python
# Add the results to the original bike dataframe
df=pd.concat([mnths,weekdays,weathersit,seasons,df],axis=1)
df
```

[29]:
```
     Aug  Dec  Feb  Jan  July  June  Mar  May  Nov  Oct  …  yr  mnth  \
0      0    0    0    1     0     0    0    0    0    0  …   0   Jan
1      0    0    0    1     0     0    0    0    0    0  …   0   Jan
2      0    0    0    1     0     0    0    0    0    0  …   0   Jan
3      0    0    0    1     0     0    0    0    0    0  …   0   Jan
4      0    0    0    1     0     0    0    0    0    0  …   0   Jan
```

```
..   …  …  …  …   …   …  …  …  …  …  …  ..   …
725    0    1    0    0    0    0    0    0    0    0  …  1   Dec
726    0    1    0    0    0    0    0    0    0    0  …  1   Dec
727    0    1    0    0    0    0    0    0    0    0  …  1   Dec
728    0    1    0    0    0    0    0    0    0    0  …  1   Dec
729    0    1    0    0    0    0    0    0    0    0  …  1   Dec

     holiday weekday  workingday    weathersit       temp      hum  \
0          0     Mon           1  Mist + Cloudy  14.110847  80.5833
1          0     Tue           1  Mist + Cloudy  14.902598  69.6087
2          0     Wed           1          Clear   8.050924  43.7273
3          0     Thu           1          Clear   8.200000  59.0435
4          0     Fri           1          Clear   9.305237  43.6957
..       …       …           …              …          …        …
725        0     Fri           1  Mist + Cloudy  10.420847  65.2917
726        0     Sat           0  Mist + Cloudy  10.386653  59.0000
727        0     Sun           0  Mist + Cloudy  10.386653  75.2917
728        0     Mon           1          Clear  10.489153  48.3333
729        0     Tue           1  Mist + Cloudy   8.849153  57.7500

     windspeed   cnt
0    10.749882   985
1    16.652113   801
2    16.636703  1349
3    10.739832  1562
4    12.522300  1600
..         …     …
725  23.458911  2114
726  10.416557  3095
727   8.333661  1341
728  23.500518  1796
729  10.374682  2729

[730 rows x 33 columns]
```

[45]:
```python
# Drop 'season','month','weekday','weathersit' as we have created the dummies␣
↪for it
df.drop(['season','mnth','weekday','weathersit'], axis = 1, inplace = True)
df
```

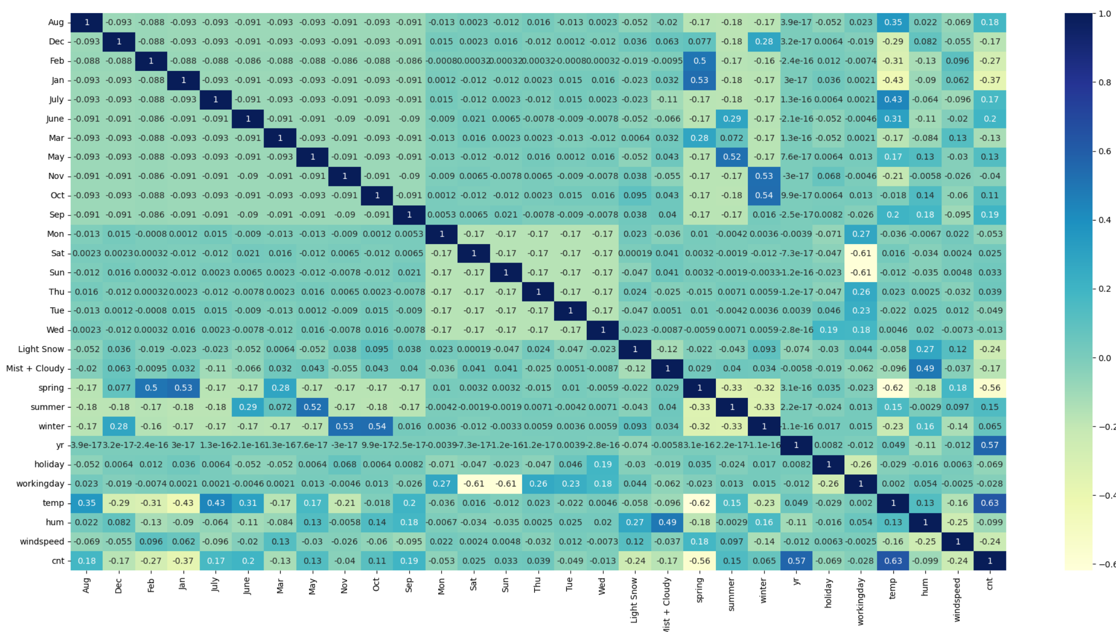
[45]:
```
     yr  holiday  workingday       temp      hum  windspeed   cnt
0     0        0           1  14.110847  80.5833  10.749882   985
1     0        0           1  14.902598  69.6087  16.652113   801
2     0        0           1   8.050924  43.7273  16.636703  1349
3     0        0           1   8.200000  59.0435  10.739832  1562
4     0        0           1   9.305237  43.6957  12.522300  1600
..   ..      …           …          …        …          …     …
```

```
725    1          0             1  10.420847  65.2917  23.458911  2114
726    1          0             0  10.386653  59.0000  10.416557  3095
727    1          0             0  10.386653  75.2917   8.333661  1341
728    1          0             1  10.489153  48.3333  23.500518  1796
729    1          0             1   8.849153  57.7500  10.374682  2729

[730 rows x 7 columns]
```

[47]:
```python
#Number of rows and columns
df.shape
```

[47]:
```
(730, 7)
```

[32]:
```python
#Now lets check the correlation between variables again
#Heatmap to see correlation between variables
plt.figure(figsize=(25, 12))
sns.heatmap(df.corr(), cmap='YlGnBu', annot = True)
plt.show()
```



3 4.Splitting the Data into Training and Testing Sets

[34]:
```python
from sklearn.model_selection import train_test_split
# We specify this so that the train and test data set always have the same␣
↪rows, respectively
df_train, df_test = train_test_split(df, train_size = 0.7, random_state = 100)
```

[35]:
```python
#Rows and columns after split
print(df_train.shape)
print(df_test.shape)
```

```
(510, 29)
(220, 29)
```

Rescaling the Features(min-max scaling)

[37]:
```python
from sklearn.preprocessing import MinMaxScaler
```

[38]:
```python
#Instantiate an object
scaler = MinMaxScaler()

#Create a list of numeric variables
num_vars=['temp','hum','windspeed','cnt']

#Fit on data
df_train[num_vars] = scaler.fit_transform(df_train[num_vars])
df_train
```

[38]:
```
     Aug  Dec  Feb  Jan  July  June  Mar  May  Nov  Oct  …  spring  summer  \
576    0    0    0    0     1     0    0    0    0    0  …       0       0
426    0    0    0    0     0     0    1    0    0    0  …       1       0
728    0    1    0    0     0     0    0    0    0    0  …       1       0
482    0    0    0    0     0     0    0    0    0    0  …       0       1
111    0    0    0    0     0     0    0    0    0    0  …       0       1
..   …  …  …  …   …   …  …  …  …  …  …      …     …
578    1    0    0    0     0     0    0    0    0    0  …       0       0
53     0    0    1    0     0     0    0    0    0    0  …       1       0
350    0    1    0    0     0     0    0    0    0    0  …       0       0
79     0    0    0    0     0     0    1    0    0    0  …       0       1
520    0    0    0    0     0     1    0    0    0    0  …       0       1

     winter  yr  holiday  workingday      temp       hum  windspeed       cnt
576       0   1        0           1  0.815169  0.725633   0.264686  0.827658
426       0   1        0           0  0.442393  0.640189   0.255342  0.465255
728       0   1        0           1  0.245101  0.498067   0.663106  0.204096
482       0   1        0           0  0.395666  0.504508   0.188475  0.482973
111       0   0        0           0  0.345824  0.751824   0.380981  0.191095
..      …  ..      …         …       …       …        …       …
578       0   1        0           1  0.863973  0.679690   0.187140  0.832835
53        0   0        0           1  0.202618  0.435939   0.111379  0.218017
350       1   0        0           1  0.248216  0.577930   0.431816  0.312586
79        0   0        0           1  0.462664  0.759870   0.529881  0.236424
520       0   1        0           1  0.600225  0.632030   0.359599  0.802922

[510 rows x 29 columns]
```

[39]:
```python
#Checking numeric variables(min and max) after scaling
df_train.describe()
```

[39]:

	Aug	Dec	Feb	Jan	July	June
count	510.000000	510.000000	510.000000	510.000000	510.000000	510.000000
mean	0.096078	0.084314	0.066667	0.088235	0.076471	0.076471
std	0.294988	0.278131	0.249689	0.283915	0.266010	0.266010
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

	Mar	May	Nov	Oct	…	spring
count	510.000000	510.000000	510.000000	510.000000	…	510.000000
mean	0.098039	0.084314	0.086275	0.084314	…	0.243137
std	0.297660	0.278131	0.281045	0.278131	…	0.429398
min	0.000000	0.000000	0.000000	0.000000	…	0.000000
25%	0.000000	0.000000	0.000000	0.000000	…	0.000000
50%	0.000000	0.000000	0.000000	0.000000	…	0.000000
75%	0.000000	0.000000	0.000000	0.000000	…	0.000000
max	1.000000	1.000000	1.000000	1.000000	…	1.000000

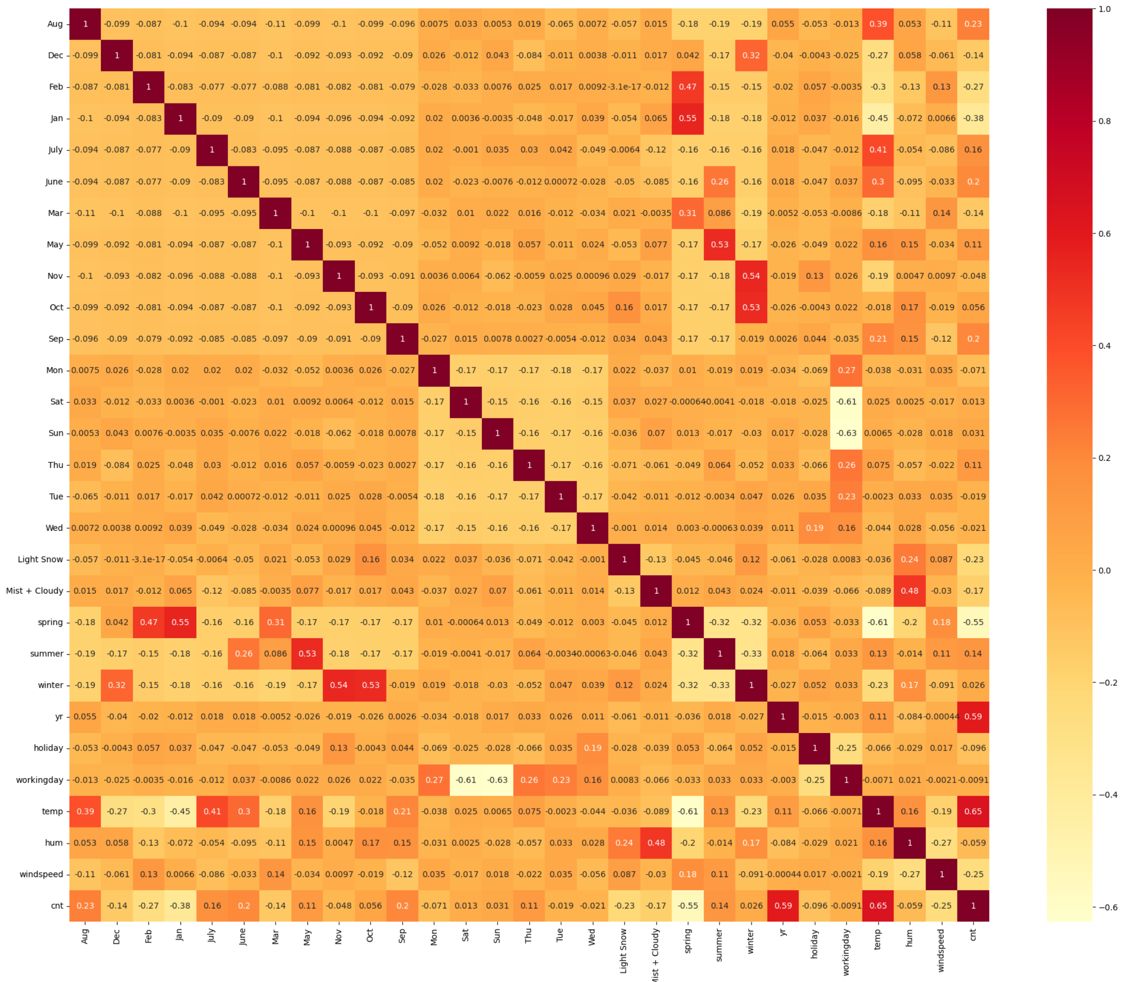
	summer	winter	yr	holiday	workingday	temp
count	510.000000	510.000000	510.000000	510.000000	510.000000	510.000000
mean	0.247059	0.247059	0.507843	0.025490	0.711765	0.537440
std	0.431725	0.431725	0.500429	0.157763	0.453386	0.225858
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.339853
50%	0.000000	0.000000	1.000000	0.000000	1.000000	0.542596
75%	0.000000	0.000000	1.000000	0.000000	1.000000	0.735215
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

	hum	windspeed	cnt
count	510.000000	510.000000	510.000000
mean	0.650480	0.320883	0.513499
std	0.145846	0.169803	0.224421
min	0.000000	0.000000	0.000000
25%	0.538643	0.199179	0.356420
50%	0.653714	0.296763	0.518638
75%	0.754830	0.414447	0.684710
max	1.000000	1.000000	1.000000

[8 rows x 29 columns]

[40]:
```python
# Let's check the correlation coefficients to see which variables are highly
correlated after scaling
```

```python
#Little to no multicollinearity among predictors

plt.figure(figsize=(25, 20))
sns.heatmap(df_train.corr(),cmap='YlOrRd',annot = True)
plt.show()
```



[41]:
```python
#Divide the data into X and y
y_train =df_train.pop('cnt')
x_train = df_train
```

4 5.Building a linear model

RFE(Recursive feature elimination)

[44]:
```python
from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression
```

[45]:
```python
# Running RFE with the output number of the variable equal to 15
lm = LinearRegression()
lm = LinearRegression()
rfe = RFE(lm,n_features_to_select=15)
rfe = rfe.fit(x_train, y_train)
```

[46]:
```python
#List of variables selected
list(zip(x_train.columns,rfe.support_,rfe.ranking_))
```

[46]:
```
[('Aug', False, 9),
 ('Dec', True, 1),
 ('Feb', False, 2),
 ('Jan', True, 1),
 ('July', True, 1),
 ('June', False, 11),
 ('Mar', False, 14),
 ('May', False, 8),
 ('Nov', True, 1),
 ('Oct', False, 12),
 ('Sep', True, 1),
 ('Mon', False, 7),
 ('Sat', False, 4),
 ('Sun', False, 5),
 ('Thu', False, 13),
 ('Tue', False, 6),
 ('Wed', False, 10),
 ('Light Snow', True, 1),
 ('Mist + Cloudy', True, 1),
 ('spring', True, 1),
 ('summer', True, 1),
 ('winter', True, 1),
 ('yr', True, 1),
 ('holiday', True, 1),
 ('workingday', False, 3),
 ('temp', True, 1),
 ('hum', True, 1),
 ('windspeed', True, 1)]
```

[47]:
```python
#Columns where RFE support is True
col = x_train.columns[rfe.support_]
col
```

[47]:
```
Index(['Dec', 'Jan', 'July', 'Nov', 'Sep', 'Light Snow', 'Mist + Cloudy',
       'spring', 'summer', 'winter', 'yr', 'holiday', 'temp', 'hum',
       'windspeed'],
      dtype='object')
```

```
[48]: #Columns where RFE support is False
x_train.columns[~rfe.support_]
```

```
[48]: Index(['Aug', 'Feb', 'June', 'Mar', 'May', 'Oct', 'Mon', 'Sat', 'Sun', 'Thu',
       'Tue', 'Wed', 'workingday'],
      dtype='object')
```

```
[49]: # Creating X_test dataframe with RFE selected variables
x_train_rfe = x_train[col]
```

```
[50]: # Adding a constant variable
import statsmodels.api as sm
x_train_rfe = sm.add_constant(x_train_rfe)
```

```
[51]: # Running the linear model
lm = sm.OLS(y_train,x_train_rfe).fit()
```

```
[52]: print(lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.845
Model:                            OLS   Adj. R-squared:                  0.840
Method:                 Least Squares   F-statistic:                     179.4
Date:                Sat, 06 Jul 2024   Prob (F-statistic):          8.15e-189
Time:                        23:08:44   Log-Likelihood:                 514.19
No. Observations:                 510   AIC:                            -996.4
Df Residuals:                     494   BIC:                            -928.6
Df Model:                          15
Covariance Type:            nonrobust
================================================================================
=
                    coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-
const             0.3197      0.036      8.859      0.000       0.249
0.391
Dec              -0.0355      0.018     -2.024      0.043      -0.070
-0.001
Jan              -0.0434      0.018     -2.393      0.017      -0.079
-0.008
July             -0.0553      0.018     -3.030      0.003      -0.091
-0.019
Nov              -0.0387      0.019     -2.057      0.040      -0.076
-0.002
Sep               0.0755      0.017      4.466      0.000       0.042
0.109
```

```
Light Snow       -0.2465      0.026     -9.331      0.000      -0.298
-0.195
Mist + Cloudy    -0.0543      0.010     -5.194      0.000      -0.075
-0.034
spring           -0.0613      0.021     -2.881      0.004      -0.103
-0.019
summer            0.0423      0.015      2.761      0.006       0.012
0.072
winter            0.1019      0.018      5.656      0.000       0.067
0.137
yr                0.2304      0.008     28.487      0.000       0.215
0.246
holiday          -0.0911      0.026     -3.557      0.000      -0.141
-0.041
temp              0.4815      0.037     13.005      0.000       0.409
0.554
hum              -0.1622      0.038     -4.291      0.000      -0.236
-0.088
windspeed        -0.1887      0.026     -7.315      0.000      -0.239
-0.138
==============================================================================
Omnibus:                       66.656   Durbin-Watson:                   2.025
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              161.040
Skew:                          -0.682   Prob(JB):                     1.07e-35
Kurtosis:                       5.392   Cond. No.                         20.8
==============================================================================

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

[53]:

```
#Drop the constant term B0
x_train_rfe = x_train_rfe.drop(['const'], axis=1)
```

[54]:

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

[54]:

	Features	VIF
13	hum	30.89
12	temp	17.79

```
14       windspeed   4.72
7           spring   4.37
9           winter   4.06
8           summer   2.82
6    Mist + Cloudy   2.32
10              yr   2.09
3              Nov   1.85
1              Jan   1.75
2             July   1.59
0              Dec   1.56
4              Sep   1.41
5       Light Snow   1.28
11         holiday   1.06
```

[55]:

```
#drop July
x_train_new1 = x_train_rfe.drop(["July"], axis = 1)
```

[56]:

```
#Build a model
x_train_lm1 = sm.add_constant(x_train_new1)
lm1 = sm.OLS(y_train,x_train_lm1).fit()
print(lm1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.842
Model:                            OLS   Adj. R-squared:                  0.838
Method:                 Least Squares   F-statistic:                     188.5
Date:                Sat, 06 Jul 2024   Prob (F-statistic):         5.38e-188
Time:                        23:08:45   Log-Likelihood:                 509.49
No. Observations:                 510   AIC:                            -989.0
Df Residuals:                     495   BIC:                            -925.5
Df Model:                          14
Covariance Type:            nonrobust
================================================================================
=
                    coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-
const             0.3083      0.036      8.520      0.000       0.237
0.379
Dec              -0.0364      0.018     -2.062      0.040      -0.071
-0.002
Jan              -0.0461      0.018     -2.526      0.012      -0.082
-0.010
Nov              -0.0385      0.019     -2.029      0.043      -0.076
-0.001
Sep               0.0900      0.016      5.512      0.000       0.058
```

```
0.122
Light Snow       -0.2496      0.027     -9.378      0.000      -0.302
-0.197
Mist + Cloudy    -0.0540      0.011     -5.119      0.000      -0.075
-0.033
spring           -0.0490      0.021     -2.328      0.020      -0.090
-0.008
summer            0.0582      0.015      4.006      0.000       0.030
0.087
winter            0.1143      0.018      6.457      0.000       0.079
0.149
yr                0.2312      0.008     28.372      0.000       0.215
0.247
holiday          -0.0899      0.026     -3.484      0.001      -0.141
-0.039
temp              0.4642      0.037     12.586      0.000       0.392
0.537
hum              -0.1546      0.038     -4.066      0.000      -0.229
-0.080
windspeed        -0.1880      0.026     -7.229      0.000      -0.239
-0.137
==============================================================================
Omnibus:                       73.786   Durbin-Watson:                   2.063
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              177.909
Skew:                          -0.751   Prob(JB):                     2.33e-39
Kurtosis:                       5.473   Cond. No.                         20.7
==============================================================================

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

[57]:

```python
#Drop the constant term B0
x_train_lm1 = x_train_lm1.drop(['const'], axis=1)
```

[58]:

```python
# Calculate the VIFs for the new model
vif = pd.DataFrame()
x = x_train_new1
vif['Features'] = x.columns
vif['VIF'] = [variance_inflation_factor(x.values, i) for i in range(x.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

[58]:

	Features	VIF
12	hum	30.88
11	temp	16.35

```
13      windspeed   4.72
6          spring   4.25
8          winter   3.90
7          summer   2.50
5    Mist + Cloudy  2.32
9              yr   2.09
2             Nov   1.84
1             Jan   1.74
0             Dec   1.55
3             Sep   1.31
4      Light Snow   1.28
10        holiday   1.06
```

[59]:

```python
#Drop humidity
x_train_new2 = x_train_lm1.drop(["hum"], axis = 1)
```

[60]:

```python
#Build a model
x_train_lm2 = sm.add_constant(x_train_new2)
lm2 = sm.OLS(y_train,x_train_lm2).fit()
print(lm2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.837
Model:                            OLS   Adj. R-squared:                  0.832
Method:                 Least Squares   F-statistic:                     195.6
Date:                Sat, 06 Jul 2024   Prob (F-statistic):          1.23e-185
Time:                        23:08:45   Log-Likelihood:                 501.12
No. Observations:                 510   AIC:                            -974.2
Df Residuals:                     496   BIC:                            -915.0
Df Model:                          13
Covariance Type:            nonrobust
================================================================================
=
                    coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-
const             0.2379      0.032      7.373      0.000       0.174
0.301
Dec              -0.0434      0.018     -2.429      0.015      -0.078
-0.008
Jan              -0.0522      0.018     -2.824      0.005      -0.089
-0.016
Nov              -0.0393      0.019     -2.040      0.042      -0.077
-0.001
Sep               0.0823      0.016      4.994      0.000       0.050
0.115
```

```
Light Snow        -0.2926      0.025    -11.803      0.000      -0.341
-0.244
Mist + Cloudy     -0.0787      0.009     -8.994      0.000      -0.096
-0.061
spring            -0.0597      0.021     -2.814      0.005      -0.101
-0.018
summer             0.0496      0.015      3.400      0.001       0.021
0.078
winter             0.0988      0.018      5.628      0.000       0.064
0.133
yr                 0.2350      0.008     28.586      0.000       0.219
0.251
holiday           -0.0907      0.026     -3.460      0.001      -0.142
-0.039
temp               0.4248      0.036     11.755      0.000       0.354
0.496
windspeed         -0.1591      0.025     -6.263      0.000      -0.209
-0.109
==============================================================================
Omnibus:                       73.264   Durbin-Watson:                   2.057
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              179.059
Skew:                          -0.742   Prob(JB):                     1.31e-39
Kurtosis:                       5.495   Cond. No.                         18.8
==============================================================================

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

[61]:

```
#Drop the constant
x_train_lm2=x_train_lm2.drop(['const'],axis=1)
```

[62]:

```
# Calculate the VIFs for the new model
vif = pd.DataFrame()
x = x_train_new2
vif['Features'] = x.columns
vif['VIF'] = [variance_inflation_factor(x.values, i) for i in range(x.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

[62]:

	Features	VIF
12	windspeed	4.66
11	temp	3.88
8	winter	2.77
6	spring	2.76
9	yr	2.07

```
7          summer  1.91
2             Nov  1.81
1             Jan  1.65
5   Mist + Cloudy  1.56
0             Dec  1.46
3             Sep  1.25
4      Light Snow  1.09
10        holiday  1.06
```

[63]:

```python
#Drop the windspeed column
x_train_new3=x_train_lm2.drop(['windspeed'],axis=1)
```

[64]:

```python
# Adding a constant variable
x_train_lm3 = sm.add_constant(x_train_new3)
lm3 = sm.OLS(y_train,x_train_lm3).fit()
print(lm3.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.824
Model:                            OLS   Adj. R-squared:                  0.820
Method:                 Least Squares   F-statistic:                     193.7
Date:                Sat, 06 Jul 2024   Prob (F-statistic):          1.25e-178
Time:                        23:08:45   Log-Likelihood:                 481.71
No. Observations:                 510   AIC:                            -937.4
Df Residuals:                     497   BIC:                            -882.4
Df Model:                          12
Covariance Type:            nonrobust
================================================================================
=
                    coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-
const             0.1761      0.032      5.525      0.000       0.114
0.239
Dec              -0.0337      0.018     -1.827      0.068      -0.070
0.003
Jan              -0.0355      0.019     -1.870      0.062      -0.073
0.002
Nov              -0.0413      0.020     -2.068      0.039      -0.081
-0.002
Sep               0.0878      0.017      5.144      0.000       0.054
0.121
Light Snow       -0.3070      0.026    -11.980      0.000      -0.357
-0.257
Mist + Cloudy    -0.0772      0.009     -8.510      0.000      -0.095
-0.059
```

```
spring          -0.0712      0.022     -3.248      0.001      -0.114
-0.028
summer           0.0417      0.015      2.761      0.006       0.012
0.071
winter           0.1024      0.018      5.623      0.000       0.067
0.138
yr               0.2338      0.009     27.407      0.000       0.217
0.251
holiday         -0.0930      0.027     -3.419      0.001      -0.146
-0.040
temp             0.4482      0.037     12.018      0.000       0.375
0.522
==============================================================================
Omnibus:                       78.127   Durbin-Watson:                   2.049
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              204.916
Skew:                          -0.764   Prob(JB):                     3.18e-45
Kurtosis:                       5.703   Cond. No.                         18.1
==============================================================================

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

[65]:

```
#Drop constant
x_train_lm3=x_train_lm3.drop(['const'],axis=1)
```

[66]:

```
# Calculate the VIFs for the new model
vif = pd.DataFrame()
x = x_train_new3
vif['Features'] = x.columns
vif['VIF'] = [variance_inflation_factor(x.values, i) for i in range(x.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

[66]:

	Features	VIF
11	temp	2.93
8	winter	2.64
9	yr	2.07
6	spring	2.01
2	Nov	1.79
1	Jan	1.64
7	summer	1.63
5	Mist + Cloudy	1.56
0	Dec	1.46
3	Sep	1.25
4	Light Snow	1.07

```
10        holiday  1.06
```

[67]:

```python
#Drop Nov
x_train_new4= x_train_lm3.drop(['Nov'], axis=1)
```

[68]:

```python
#Build a model
x_train_lm4=sm.add_constant(x_train_new4)
lm4=sm.OLS(y_train,x_train_lm4).fit()
print(lm4.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.822
Model:                            OLS   Adj. R-squared:                  0.818
Method:                 Least Squares   F-statistic:                     209.6
Date:                Sat, 06 Jul 2024   Prob (F-statistic):          7.02e-179
Time:                        23:08:45   Log-Likelihood:                 479.53
No. Observations:                 510   AIC:                            -935.1
Df Residuals:                     498   BIC:                            -884.2
Df Model:                          11
Covariance Type:            nonrobust
================================================================================
=
                    coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-
const             0.1583      0.031      5.141      0.000       0.098
0.219
Dec              -0.0185      0.017     -1.088      0.277      -0.052
0.015
Jan              -0.0303      0.019     -1.605      0.109      -0.067
0.007
Sep               0.0935      0.017      5.532      0.000       0.060
0.127
Light Snow       -0.3029      0.026    -11.819      0.000      -0.353
-0.253
Mist + Cloudy    -0.0764      0.009     -8.402      0.000      -0.094
-0.059
spring           -0.0629      0.022     -2.909      0.004      -0.105
-0.020
summer            0.0473      0.015      3.177      0.002       0.018
0.077
winter            0.0924      0.018      5.247      0.000       0.058
0.127
yr                0.2336      0.009     27.296      0.000       0.217
0.250
holiday          -0.0997      0.027     -3.681      0.000      -0.153
```

```
-0.047
temp                0.4687      0.036     12.991      0.000       0.398
0.540
==============================================================================
Omnibus:                       71.781   Durbin-Watson:                   2.066
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              175.328
Skew:                          -0.728   Prob(JB):                     8.47e-39
Kurtosis:                       5.477   Cond. No.                         17.4
==============================================================================

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

[69]:
```
#Drop constant
x_train_lm4= x_train_lm4.drop(['const'], axis=1)
```

[70]:
```
# Calculate the VIFs for the new model
vif = pd.DataFrame()
x =x_train_new4
vif['Features'] = x.columns
vif['VIF'] = [variance_inflation_factor(x.values, i) for i in range(x.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

[70]:

	Features	VIF
10	temp	2.93
8	yr	2.07
5	spring	2.00
7	winter	1.67
1	Jan	1.64
6	summer	1.63
4	Mist + Cloudy	1.56
0	Dec	1.29
2	Sep	1.23
3	Light Snow	1.07
9	holiday	1.04

[71]:
```
#Drop holiday
x_train_new5=x_train_lm4.drop(['holiday'], axis=1)
```

[72]:
```
#Building a model
x_train_lm5= sm.add_constant(x_train_new5)
lm5=sm.OLS(y_train,x_train_lm5).fit()
print(lm5.summary())
```

```
                            OLS Regression Results
```

```
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.818
Model:                            OLS   Adj. R-squared:                  0.814
Method:                 Least Squares   F-statistic:                     223.6
Date:                Sat, 06 Jul 2024   Prob (F-statistic):         3.56e-177
Time:                        23:08:45   Log-Likelihood:                 472.68
No. Observations:                 510   AIC:                            -923.4
Df Residuals:                     499   BIC:                            -876.8
Df Model:                          10
Covariance Type:            nonrobust
================================================================================
=
                    coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-
const             0.1552      0.031      4.980      0.000       0.094
0.216
Dec              -0.0163      0.017     -0.951      0.342      -0.050
0.017
Jan              -0.0305      0.019     -1.594      0.112      -0.068
0.007
Sep               0.0898      0.017      5.259      0.000       0.056
0.123
Light Snow       -0.2987      0.026    -11.520      0.000      -0.350
-0.248
Mist + Cloudy    -0.0747      0.009     -8.120      0.000      -0.093
-0.057
spring           -0.0654      0.022     -2.992      0.003      -0.108
-0.022
summer            0.0474      0.015      3.147      0.002       0.018
0.077
winter            0.0893      0.018      5.013      0.000       0.054
0.124
yr                0.2339      0.009     27.004      0.000       0.217
0.251
temp              0.4708      0.037     12.890      0.000       0.399
0.543
==============================================================================
Omnibus:                       77.027   Durbin-Watson:                   2.024
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              188.651
Skew:                          -0.777   Prob(JB):                     1.08e-41
Kurtosis:                       5.542   Cond. No.                         17.4
==============================================================================

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

[73]:
```
#Drop the constant
x_train_lm7=x_train_lm5.drop(['const'],axis=1)
```

[74]:
```
# Calculate the VIFs for the new model
vif = pd.DataFrame()
x = x_train_new5
vif['Features'] = x.columns
vif['VIF'] = [variance_inflation_factor(x.values, i) for i in range(x.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

[74]:

	Features	VIF
9	temp	2.93
8	yr	2.07
5	spring	1.98
7	winter	1.65
1	Jan	1.64
6	summer	1.63
4	Mist + Cloudy	1.56
0	Dec	1.29
2	Sep	1.23
3	Light Snow	1.07

[75]:
```
#Drop temp for the lm4 model
x_train_new6=x_train_lm4.drop(['temp'], axis=1)
```

[76]:
```
#Building a model
x_train_lm6= sm.add_constant(x_train_new6)
lm6=sm.OLS(y_train,x_train_lm6).fit()
print(lm6.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.762
Model:                            OLS   Adj. R-squared:                  0.757
Method:                 Least Squares   F-statistic:                     159.9
Date:                Sat, 06 Jul 2024   Prob (F-statistic):          1.40e-148
Time:                        23:08:45   Log-Likelihood:                 405.10
No. Observations:                 510   AIC:                            -788.2
Df Residuals:                     499   BIC:                            -741.6
Df Model:                          10
Covariance Type:            nonrobust
================================================================================
=
                    coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
```

```
-
const               0.5337      0.012     43.522      0.000       0.510
0.558
Dec                -0.0673      0.019     -3.521      0.000      -0.105
-0.030
Jan                -0.0989      0.021     -4.720      0.000      -0.140
-0.058
Sep                 0.0763      0.019      3.917      0.000       0.038
0.115
Light Snow         -0.3223      0.030    -10.896      0.000      -0.380
-0.264
Mist + Cloudy      -0.0839      0.010     -7.994      0.000      -0.104
-0.063
spring             -0.2722      0.017    -16.342      0.000      -0.305
-0.239
summer             -0.0562      0.015     -3.863      0.000      -0.085
-0.028
winter             -0.0632      0.015     -4.235      0.000      -0.093
-0.034
yr                  0.2459      0.010     25.009      0.000       0.227
0.265
holiday            -0.1053      0.031     -3.363      0.001      -0.167
-0.044
==============================================================================
Omnibus:                       49.619   Durbin-Watson:                   2.006
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              119.252
Skew:                          -0.512   Prob(JB):                     1.27e-26
Kurtosis:                       5.136   Cond. No.                         8.36
==============================================================================

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

5 6.Residual Analysis of the train data

[77]:
```python
#X_train_lm5=sm.add_constant(X_train_lm5)
#X_train_lm5.columns
x_train_lm5
```

[77]:
```
     const  Dec  Jan  Sep  Light Snow  Mist + Cloudy  spring  summer  winter  \
576    1.0    0    0    0           0              0       0       0       0
426    1.0    0    0    0           0              1       1       0       0
728    1.0    1    0    0           0              0       1       0       0
482    1.0    0    0    0           0              1       0       1       0
111    1.0    0    0    0           0              1       0       1       0
..     …    …    …    …           …              …       …       …       …
```

```
578     1.0     0     0     0           0            0       0       0       0
53      1.0     0     0     0           0            0       1       0       0
350     1.0     1     0     0           0            1       0       0       1
79      1.0     0     0     0           0            1       0       1       0
520     1.0     0     0     0           0            1       0       1       0

     yr      temp
576   1  0.815169
426   1  0.442393
728   1  0.245101
482   1  0.395666
111   0  0.345824
..   ..       …
578   1  0.863973
53    0  0.202618
350   0  0.248216
79    0  0.462664
520   1  0.600225

[510 rows x 11 columns]
```

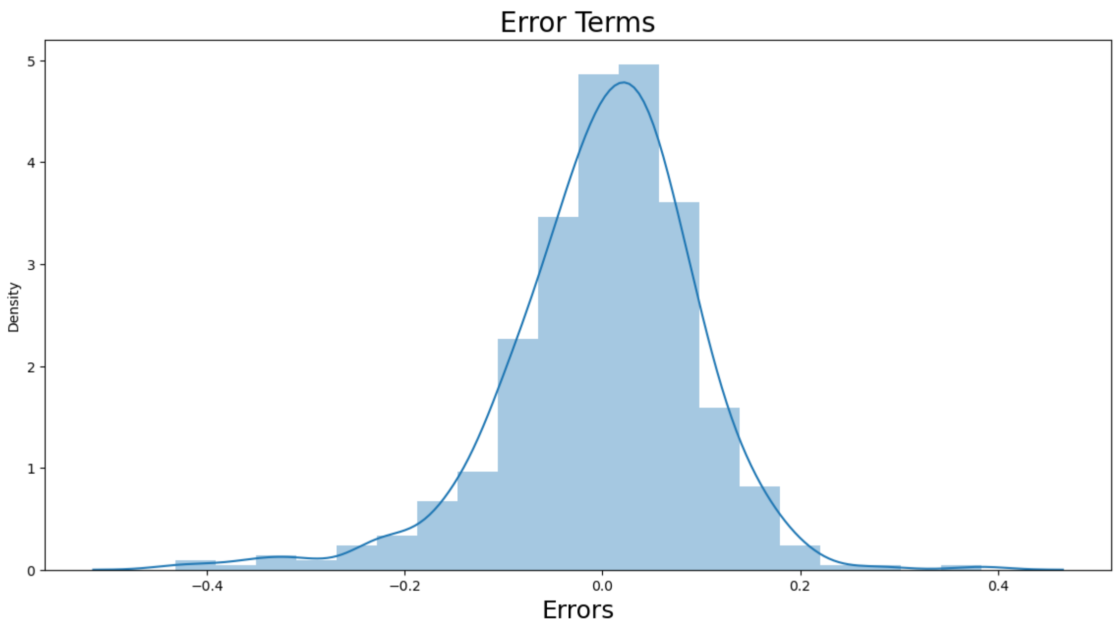
[78]:

```
#y train predicted
y_train_pred = lm5.predict(x_train_lm5)
```

[79]:

```
# Plot the histogram of the error terms

fig = plt.figure()
plt.figure(figsize=(14,7))
sns.distplot((y_train - y_train_pred), bins = 20)
plt.title('Error Terms', fontsize = 20)                  # Plot heading
plt.xlabel('Errors', fontsize = 18)  # X-label
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```



6 7.Making Predictions

[81]:
```
#Create a list of numeric variables
num_vars=['temp','hum','windspeed','cnt']

#Fit on data
df_test[num_vars] = scaler.transform(df_test[num_vars])
df_test.head()
```

[81]:

	Aug	Dec	Feb	Jan	July	June	Mar	May	Nov	Oct	…	spring	summer	\
184	0	0	0	0	1	0	0	0	0	0	…	0	0	
535	0	0	0	0	0	1	0	0	0	0	…	0	1	
299	0	0	0	0	0	0	0	0	0	1	…	0	0	
221	1	0	0	0	0	0	0	0	0	0	…	0	0	
152	0	0	0	0	0	1	0	0	0	0	…	0	1	

	winter	yr	holiday	workingday	temp	hum	windspeed	cnt
184	0	0	1	0	0.831783	0.657364	0.084219	0.692706
535	0	1	0	1	0.901354	0.610133	0.153728	0.712034
299	1	0	0	0	0.511964	0.837699	0.334206	0.303382
221	0	0	0	1	0.881625	0.437098	0.339570	0.547400
152	0	0	0	0	0.817246	0.314298	0.537414	0.569029

[5 rows x 29 columns]

[82]:
```
#Dividing into X_test and y_test
y_test =df_test.pop('cnt')
x_test =df_test
x_test.describe()
```

[82]:
```
              Aug         Dec         Feb         Jan        July        June  \
count  220.000000  220.000000  220.000000  220.000000  220.000000  220.000000
mean     0.059091    0.086364    0.100000    0.077273    0.104545    0.095455
std      0.236333    0.281541    0.300684    0.267633    0.306665    0.294512
min      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
50%      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
75%      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
max      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000

              Mar         May         Nov         Oct  …  Mist + Cloudy  \
count  220.000000  220.000000  220.000000  220.000000  …     220.000000
mean     0.054545    0.086364    0.072727    0.086364  …       0.318182
std      0.227609    0.281541    0.260281    0.281541  …       0.466833
min      0.000000    0.000000    0.000000    0.000000  …       0.000000
25%      0.000000    0.000000    0.000000    0.000000  …       0.000000
50%      0.000000    0.000000    0.000000    0.000000  …       0.000000
75%      0.000000    0.000000    0.000000    0.000000  …       1.000000
max      1.000000    1.000000    1.000000    1.000000  …       1.000000

           spring      summer      winter          yr     holiday  workingday  \
count  220.000000  220.000000  220.000000  220.000000  220.000000  220.000000
mean     0.254545    0.263636    0.236364    0.481818    0.036364    0.640909
std      0.436599    0.441609    0.425817    0.500809    0.187620    0.480828
min      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
50%      0.000000    0.000000    0.000000    0.000000    0.000000    1.000000
75%      1.000000    1.000000    0.000000    1.000000    0.000000    1.000000
max      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000

             temp         hum   windspeed
count  220.000000  220.000000  220.000000
mean     0.558718    0.638221    0.313293
std      0.233187    0.148694    0.159584
min      0.046591    0.261915   -0.042808
25%      0.355429    0.529197    0.198843
50%      0.558172    0.625590    0.300126
75%      0.755981    0.743798    0.402718
max      0.984424    1.002146    0.807474

[8 rows x 28 columns]
```

[83]:
```
#Columns
x_train_new5.columns
```

[83]:
```
Index(['Dec', 'Jan', 'Sep', 'Light Snow', 'Mist + Cloudy', 'spring', 'summer',
       'winter', 'yr', 'temp'],
      dtype='object')
```

[84]:
```
# Now let's use our model to make predictions.

# Creating X_test_new dataframe by dropping variables from X_test
x_test_new = x_test[x_train_new5.columns]

# Adding a constant variable
x_test_new1 = sm.add_constant(x_test_new)
x_test_new1.head()
```

[84]:
```
     const  Dec  Jan  Sep  Light Snow  Mist + Cloudy  spring  summer  winter  \
184    1.0    0    0    0           0              1       0       0       0
535    1.0    0    0    0           0              0       0       1       0
299    1.0    0    0    0           0              1       0       0       1
221    1.0    0    0    0           0              0       0       0       0
152    1.0    0    0    0           0              0       0       1       0

     yr      temp
184   0  0.831783
535   1  0.901354
299   0  0.511964
221   0  0.881625
152   0  0.817246
```

[85]:
```
# Making predictions
y_pred = lm5.predict(x_test_new1)
```

Finding R-squared and Adjusted R-Squared for Test set

[86]:
```
#Evaluate R-square for test
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

[86]: 0.8099204382008233

[87]:
```
#Adjusted R^2
#adj r2=1-(1-R2)*(n-1)/(n-p-1)

#n =sample size , p = number of independent variables

Adj_r2=1-(1-0.8115083)*(11-1)/(11-1-1)
print(Adj_r2)
```

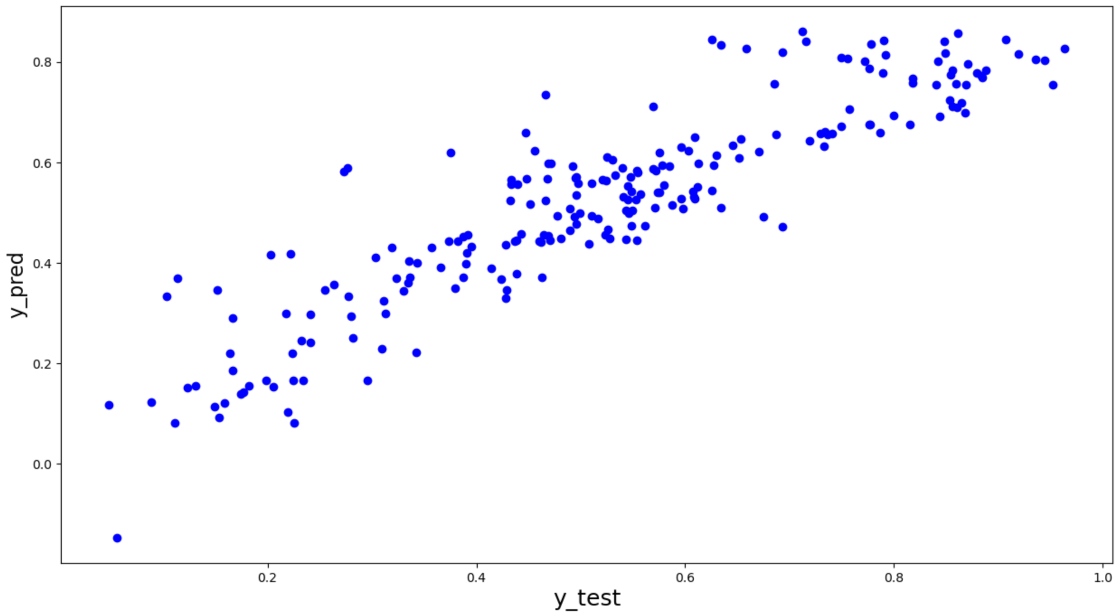
```
0.7905647777777778
```

7 8.Model Evaluation

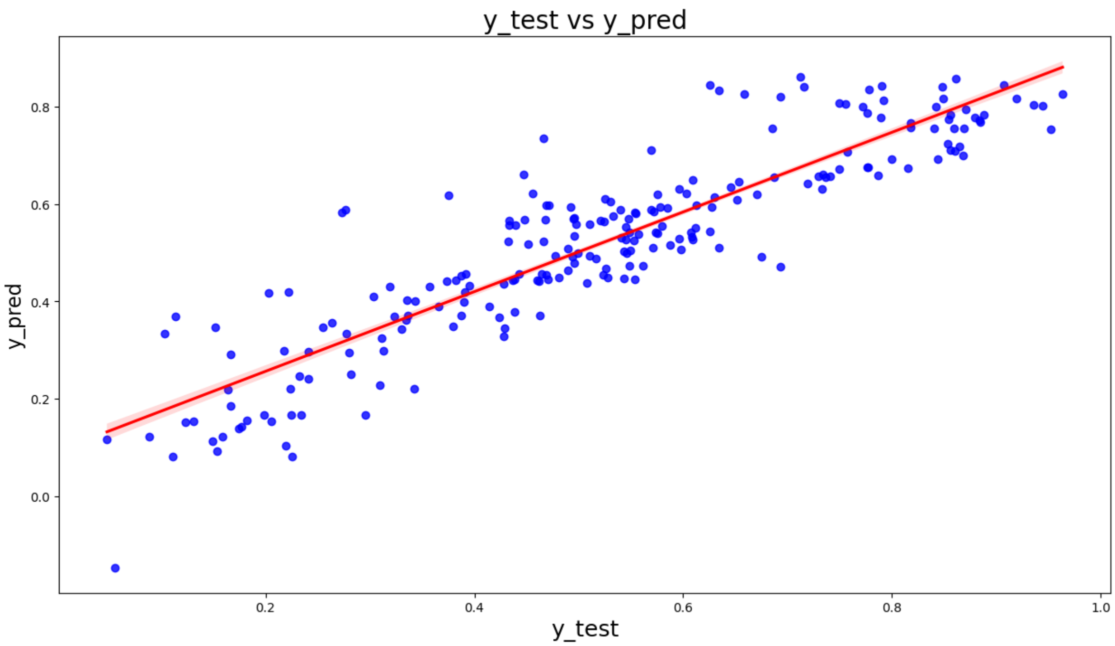
[88]:
```python
# Plotting y_test and y_pred to understand the spread.
fig = plt.figure()
plt.figure(figsize=(15,8))
plt.scatter(y_test,y_pred,color='blue')
fig.suptitle('y_test vs y_pred', fontsize=20)              # Plot heading
plt.xlabel('y_test', fontsize=18)                          # X-label
plt.ylabel('y_pred', fontsize=16)                          # Y-label
```

[88]:
```
Text(0, 0.5, 'y_pred')
```

```
<Figure size 640x480 with 0 Axes>
```



[89]:
```python
#Regression plot
plt.figure(figsize=(15,8))
sns.regplot(x=y_test, y=y_pred, ci=68, fit_reg=True,scatter_kws={"color":
 ↪"blue"}, line_kws={"color": "red"})

plt.title('y_test vs y_pred', fontsize=20)              # Plot heading
plt.xlabel('y_test', fontsize=18)                          # X-label
plt.ylabel('y_pred', fontsize=16)                          # Y-label
plt.show()
```


y_test vs y_pred
y_pred
y_test
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0