# Reading and Understanding the Data

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

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [3]:
```
bike = pd.DataFrame(pd.read_csv(r"C:\Users\DELL\Downloads\day.csv"))
```

In [4]:
```
# Check the head of the dataset
bike.head()
```

Out[4]:

| | 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 [5]:
```
# Check the descriptive information
bike.info()
```

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

In [6]:
```
bike.describe()
```

Out[6]:

| | instant | season | yr | mnth | holiday | weekday | workingda |
|---|---|---|---|---|---|---|---|
| **count** | 730.000000 | 730.000000 | 730.000000 | 730.000000 | 730.000000 | 730.000000 | 730.00000 |
| **mean** | 365.500000 | 2.498630 | 0.500000 | 6.526027 | 0.028767 | 2.997260 | 0.68356 |
| **std** | 210.877136 | 1.110184 | 0.500343 | 3.450215 | 0.167266 | 2.006161 | 0.46540 |
| **min** | 1.000000 | 1.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 0.00000 |
| **25%** | 183.250000 | 2.000000 | 0.000000 | 4.000000 | 0.000000 | 1.000000 | 0.00000 |
| **50%** | 365.500000 | 3.000000 | 0.500000 | 7.000000 | 0.000000 | 3.000000 | 1.00000 |
| **75%** | 547.750000 | 3.000000 | 1.000000 | 10.000000 | 0.000000 | 5.000000 | 1.00000 |
| **max** | 730.000000 | 4.000000 | 1.000000 | 12.000000 | 1.000000 | 6.000000 | 1.00000 |

In [7]:
```
# Check the shape of df

print(bike.shape)
```

```
(730, 16)
```

# Finding :

Dataset has 730 rows and 16 columns.

# DATA QUALITY CHECK

## Checking for Null or Missing values

In [8]:
```
# percentage of missing values in each column
round(100*(bike.isnull().sum()/len(bike)), 2).sort_values(ascending=
```

Out[8]:
```
instant       0.0
dteday        0.0
season        0.0
yr            0.0
mnth          0.0
holiday       0.0
weekday       0.0
workingday    0.0
weathersit    0.0
temp          0.0
atemp         0.0
hum           0.0
windspeed     0.0
casual        0.0
registered    0.0
cnt           0.0
dtype: float64
```

In [9]:
```
# row-wise null count percentage
round((bike.isnull().sum(axis=1)/len(bike))*100,2).sort_values(ascer
```

Out[9]:
```
0      0.0
479    0.0
481    0.0
482    0.0
483    0.0
      ...
245    0.0
246    0.0
247    0.0
248    0.0
729    0.0
Length: 730, dtype: float64
```

## Finding

There are no missing / Null values either in columns or rows

# Duplicate Check

In [10]:
```
bike_dup = bike.copy()

# Checking for duplicates and dropping the entire duplicate row if d
bike_dup.drop_duplicates(subset=None, inplace=True)
```

In [11]:
```
bike_dup.shape
```

Out[11]: (730, 16)

In [12]:
```
bike.shape
```

Out[12]: (730, 16)

## Insights

The shape after running the drop duplicate command is same as the original dataframe.

Hence we can conclude that there were zero duplicate values in the dataset.

# Data Cleaning

In [14]:
```
bike_dummy=bike.iloc[:,1:16]
```

In [15]:
```
for col in bike_dummy:
    print(bike_dummy[col].value_counts(ascending=False), '\n\n\n')
```

```
01-01-2018    1
25-04-2019    1
27-04-2019    1
28-04-2019    1
29-04-2019    1
             ..
03-09-2018    1
04-09-2018    1
05-09-2018    1
06-09-2018    1
31-12-2019    1
Name: dteday, Length: 730, dtype: int64



3    188
2    184
1    180
4    178
```

## Insights

# Removing redundant & unwanted columns

In [16]:
```
1 bike.columns
```

Out[16]:
```
Index(['instant', 'dteday', 'season', 'yr', 'mnth', 'holiday', 'weekda
y',
       'workingday', 'weathersit', 'temp', 'atemp', 'hum', 'windspeed',
       'casual', 'registered', 'cnt'],
      dtype='object')
```

In [17]:
```
1 bike_new=bike[['season', 'yr', 'mnth', 'holiday', 'weekday',
2        'workingday', 'weathersit', 'temp', 'atemp', 'hum', 'windspee
3        'cnt']]
4
```

In [18]:
```
1 bike_new.info()
```

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

# Creating Dummy Variables

In [19]:
```
# Check the datatypes before convertion
bike_new.info()
```

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

In [20]:
```
# Convert to 'category' data type

bike_new['season']=bike_new['season'].astype('category')
bike_new['weathersit']=bike_new['weathersit'].astype('category')
bike_new['mnth']=bike_new['mnth'].astype('category')
bike_new['weekday']=bike_new['weekday'].astype('category')

```

In [21]:
```
bike_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 730 entries, 0 to 729
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype   
---  ------      --------------  -----   
 0   season      730 non-null    category
 1   yr          730 non-null    int64   
 2   mnth        730 non-null    category
 3   holiday     730 non-null    int64   
 4   weekday     730 non-null    category
 5   workingday  730 non-null    int64   
 6   weathersit  730 non-null    category
 7   temp        730 non-null    float64 
 8   atemp       730 non-null    float64 
 9   hum         730 non-null    float64 
 10  windspeed   730 non-null    float64 
 11  cnt         730 non-null    int64   
dtypes: category(4), float64(4), int64(4)
memory usage: 49.7 KB
```

In [22]:
```
bike_new = pd.get_dummies(bike_new, drop_first=True)
bike_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 730 entries, 0 to 729
Data columns (total 30 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   yr            730 non-null    int64  
 1   holiday       730 non-null    int64  
 2   workingday    730 non-null    int64  
 3   temp          730 non-null    float64
 4   atemp         730 non-null    float64
 5   hum           730 non-null    float64
 6   windspeed     730 non-null    float64
 7   cnt           730 non-null    int64  
 8   season_2      730 non-null    uint8  
 9   season_3      730 non-null    uint8  
 10  season_4      730 non-null    uint8  
 11  mnth_2        730 non-null    uint8  
 12  mnth_3        730 non-null    uint8  
 13  mnth_4        730 non-null    uint8  
 14  mnth_5        730 non-null    uint8  
 15  mnth_6        730 non-null    uint8  
 16  mnth_7        730 non-null    uint8  
 17  mnth_8        730 non-null    uint8  
 18  mnth_9        730 non-null    uint8  
 19  mnth_10       730 non-null    uint8  
 20  mnth_11       730 non-null    uint8  
 21  mnth_12       730 non-null    uint8  
 22  weekday_1     730 non-null    uint8  
 23  weekday_2     730 non-null    uint8  
 24  weekday_3     730 non-null    uint8  
 25  weekday_4     730 non-null    uint8  
 26  weekday_5     730 non-null    uint8  
 27  weekday_6     730 non-null    uint8  
 28  weathersit_2  730 non-null    uint8  
 29  weathersit_3  730 non-null    uint8  
dtypes: float64(4), int64(4), uint8(22)
memory usage: 61.4 KB
```

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

Out[23]: (730, 30)

# SPLITTING THE DATA

- Splitting the data to Train and Test
- We will use train_test_split method from sklearn package for this

In [24]:
```
# Check the shape before spliting

bike_new.shape
```

Out[24]: (730, 30)

In [25]:
```
# Check the info before spliting

bike_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 730 entries, 0 to 729
Data columns (total 30 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   yr            730 non-null    int64
 1   holiday       730 non-null    int64
 2   workingday    730 non-null    int64
 3   temp          730 non-null    float64
 4   atemp         730 non-null    float64
 5   hum           730 non-null    float64
 6   windspeed     730 non-null    float64
 7   cnt           730 non-null    int64
 8   season_2      730 non-null    uint8
 9   season_3      730 non-null    uint8
 10  season_4      730 non-null    uint8
 11  mnth_2        730 non-null    uint8
 12  mnth_3        730 non-null    uint8
 13  mnth_4        730 non-null    uint8
 14  mnth_5        730 non-null    uint8
 15  mnth_6        730 non-null    uint8
 16  mnth_7        730 non-null    uint8
 17  mnth_8        730 non-null    uint8
 18  mnth_9        730 non-null    uint8
 19  mnth_10       730 non-null    uint8
 20  mnth_11       730 non-null    uint8
 21  mnth_12       730 non-null    uint8
 22  weekday_1     730 non-null    uint8
 23  weekday_2     730 non-null    uint8
 24  weekday_3     730 non-null    uint8
 25  weekday_4     730 non-null    uint8
 26  weekday_5     730 non-null    uint8
 27  weekday_6     730 non-null    uint8
 28  weathersit_2  730 non-null    uint8
 29  weathersit_3  730 non-null    uint8
dtypes: float64(4), int64(4), uint8(22)
memory usage: 61.4 KB
```

In [26]:
```
from sklearn.model_selection import train_test_split
np.random.seed(0)
df_train, df_test = train_test_split(bike_new, train_size = 0.70, te
```

- Verify the info and shape of the dataframes after split:

In [27]:
```
1 df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 510 entries, 483 to 366
Data columns (total 30 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   yr            510 non-null    int64  
 1   holiday       510 non-null    int64  
 2   workingday    510 non-null    int64  
 3   temp          510 non-null    float64
 4   atemp         510 non-null    float64
 5   hum           510 non-null    float64
 6   windspeed     510 non-null    float64
 7   cnt           510 non-null    int64  
 8   season_2      510 non-null    uint8  
 9   season_3      510 non-null    uint8  
 10  season_4      510 non-null    uint8  
 11  mnth_2        510 non-null    uint8  
 12  mnth_3        510 non-null    uint8  
 13  mnth_4        510 non-null    uint8  
 14  mnth_5        510 non-null    uint8  
 15  mnth_6        510 non-null    uint8  
 16  mnth_7        510 non-null    uint8  
 17  mnth_8        510 non-null    uint8  
 18  mnth_9        510 non-null    uint8  
 19  mnth_10       510 non-null    uint8  
 20  mnth_11       510 non-null    uint8  
 21  mnth_12       510 non-null    uint8  
 22  weekday_1     510 non-null    uint8  
 23  weekday_2     510 non-null    uint8  
 24  weekday_3     510 non-null    uint8  
 25  weekday_4     510 non-null    uint8  
 26  weekday_5     510 non-null    uint8  
 27  weekday_6     510 non-null    uint8  
 28  weathersit_2  510 non-null    uint8  
 29  weathersit_3  510 non-null    uint8  
dtypes: float64(4), int64(4), uint8(22)
memory usage: 46.8 KB
```

In [28]:
```
1 df_train.shape
```

Out[28]: (510, 30)

In [29]:
```
1 df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 219 entries, 22 to 313
Data columns (total 30 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   yr            219 non-null    int64  
 1   holiday       219 non-null    int64  
 2   workingday    219 non-null    int64  
 3   temp          219 non-null    float64
 4   atemp         219 non-null    float64
 5   hum           219 non-null    float64
 6   windspeed     219 non-null    float64
 7   cnt           219 non-null    int64  
 8   season_2      219 non-null    uint8  
 9   season_3      219 non-null    uint8  
 10  season_4      219 non-null    uint8  
 11  mnth_2        219 non-null    uint8  
 12  mnth_3        219 non-null    uint8  
 13  mnth_4        219 non-null    uint8  
 14  mnth_5        219 non-null    uint8  
 15  mnth_6        219 non-null    uint8  
 16  mnth_7        219 non-null    uint8  
 17  mnth_8        219 non-null    uint8  
 18  mnth_9        219 non-null    uint8  
 19  mnth_10       219 non-null    uint8  
 20  mnth_11       219 non-null    uint8  
 21  mnth_12       219 non-null    uint8  
 22  weekday_1     219 non-null    uint8  
 23  weekday_2     219 non-null    uint8  
 24  weekday_3     219 non-null    uint8  
 25  weekday_4     219 non-null    uint8  
 26  weekday_5     219 non-null    uint8  
 27  weekday_6     219 non-null    uint8  
 28  weathersit_2  219 non-null    uint8  
 29  weathersit_3  219 non-null    uint8  
dtypes: float64(4), int64(4), uint8(22)
memory usage: 20.1 KB
```

In [30]:
```
1 df_test.shape
```

Out[30]:
```
(219, 30)
```

# EXPLORATORY DATA ANALYSIS

In [31]:
```
1 df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 510 entries, 483 to 366
Data columns (total 30 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   yr            510 non-null    int64  
 1   holiday       510 non-null    int64  
 2   workingday    510 non-null    int64  
 3   temp          510 non-null    float64
 4   atemp         510 non-null    float64
 5   hum           510 non-null    float64
 6   windspeed     510 non-null    float64
 7   cnt           510 non-null    int64  
 8   season_2      510 non-null    uint8  
 9   season_3      510 non-null    uint8  
 10  season_4      510 non-null    uint8  
 11  mnth_2        510 non-null    uint8  
 12  mnth_3        510 non-null    uint8  
 13  mnth_4        510 non-null    uint8  
 14  mnth_5        510 non-null    uint8  
 15  mnth_6        510 non-null    uint8  
 16  mnth_7        510 non-null    uint8  
 17  mnth_8        510 non-null    uint8  
 18  mnth_9        510 non-null    uint8  
 19  mnth_10       510 non-null    uint8  
 20  mnth_11       510 non-null    uint8  
 21  mnth_12       510 non-null    uint8  
 22  weekday_1     510 non-null    uint8  
 23  weekday_2     510 non-null    uint8  
 24  weekday_3     510 non-null    uint8  
 25  weekday_4     510 non-null    uint8  
 26  weekday_5     510 non-null    uint8  
 27  weekday_6     510 non-null    uint8  
 28  weathersit_2  510 non-null    uint8  
 29  weathersit_3  510 non-null    uint8  
dtypes: float64(4), int64(4), uint8(22)
memory usage: 46.8 KB
```

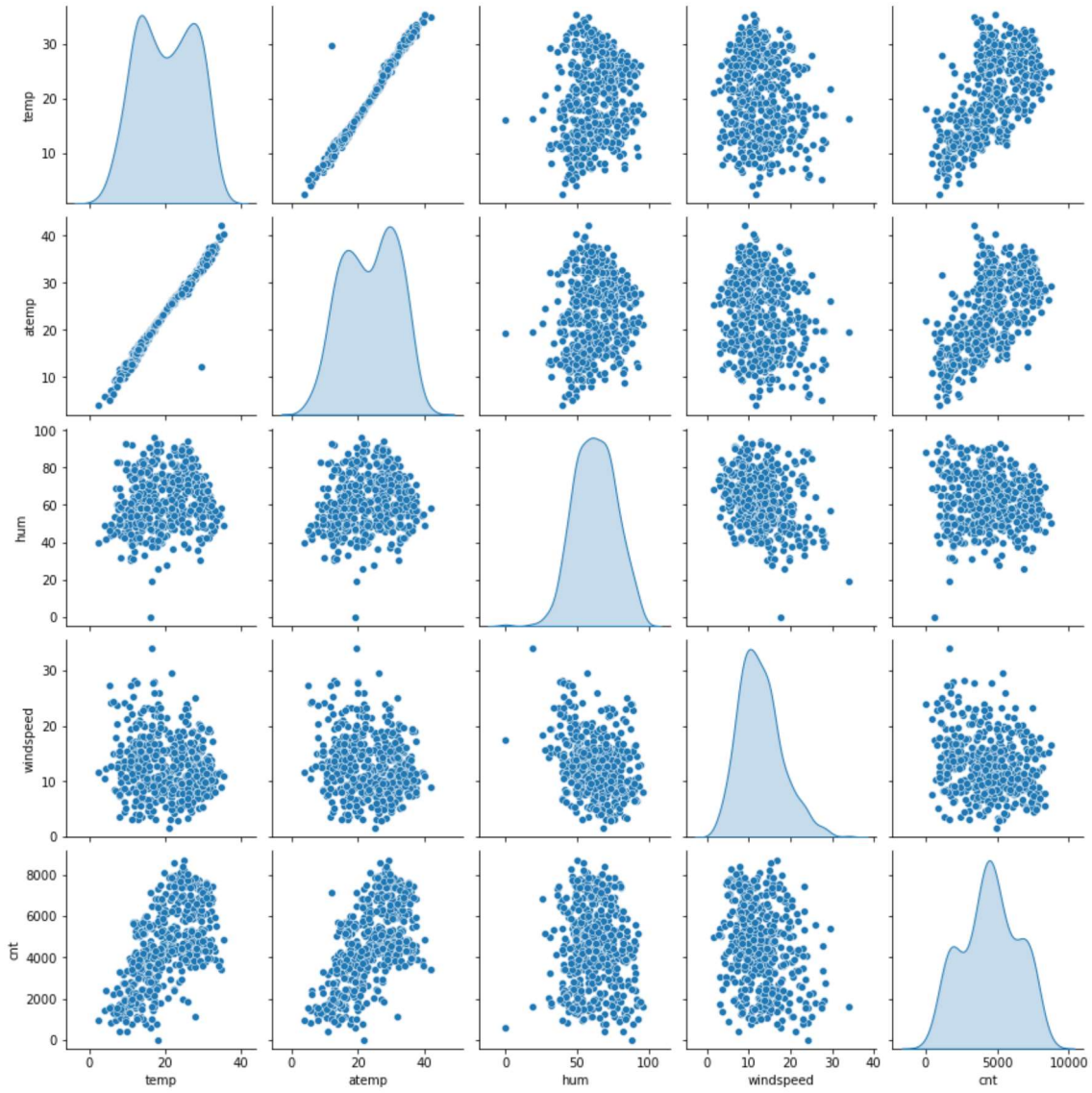
In [32]:
```
1 df_train.columns
```

Out[32]:
```
Index(['yr', 'holiday', 'workingday', 'temp', 'atemp', 'hum', 'windspeed',
       'cnt', 'season_2', 'season_3', 'season_4', 'mnth_2', 'mnth_3', 'mnth_4',
       'mnth_5', 'mnth_6', 'mnth_7', 'mnth_8', 'mnth_9', 'mnth_10', 'mnth_11',
       'mnth_12', 'weekday_1', 'weekday_2', 'weekday_3', 'weekday_4',
       'weekday_5', 'weekday_6', 'weathersit_2', 'weathersit_3'],
      dtype='object')
```

In [33]:
```
# Create a new dataframe of only numeric variables:

bike_num=df_train[[ 'temp', 'atemp', 'hum', 'windspeed','cnt']]

sns.pairplot(bike_num, diag_kind='kde')
plt.show()
```




## Insights

- The above Pair-Plot tells us that there is a LINEAR RELATION between 'temp'and'atemp'

# Visualising Catagorical Variables

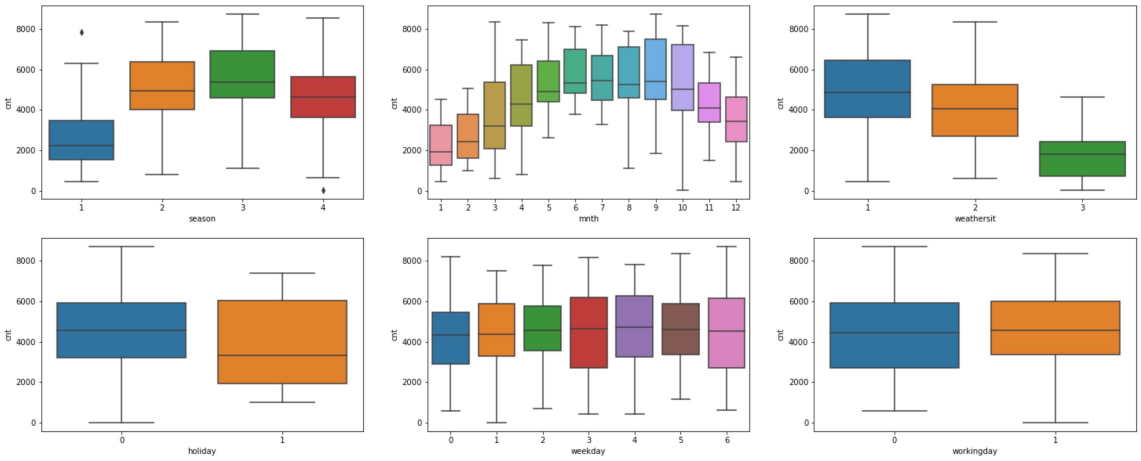
In [34]:
```
1 df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 510 entries, 483 to 366
Data columns (total 30 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   yr            510 non-null    int64
 1   holiday       510 non-null    int64
 2   workingday    510 non-null    int64
 3   temp          510 non-null    float64
 4   atemp         510 non-null    float64
 5   hum           510 non-null    float64
 6   windspeed     510 non-null    float64
 7   cnt           510 non-null    int64
 8   season_2      510 non-null    uint8
 9   season_3      510 non-null    uint8
 10  season_4      510 non-null    uint8
 11  mnth_2        510 non-null    uint8
 12  mnth_3        510 non-null    uint8
 13  mnth_4        510 non-null    uint8
 14  mnth_5        510 non-null    uint8
 15  mnth_6        510 non-null    uint8
 16  mnth_7        510 non-null    uint8
 17  mnth_8        510 non-null    uint8
 18  mnth_9        510 non-null    uint8
 19  mnth_10       510 non-null    uint8
 20  mnth_11       510 non-null    uint8
 21  mnth_12       510 non-null    uint8
 22  weekday_1     510 non-null    uint8
 23  weekday_2     510 non-null    uint8
 24  weekday_3     510 non-null    uint8
 25  weekday_4     510 non-null    uint8
 26  weekday_5     510 non-null    uint8
 27  weekday_6     510 non-null    uint8
 28  weathersit_2  510 non-null    uint8
 29  weathersit_3  510 non-null    uint8
dtypes: float64(4), int64(4), uint8(22)
memory usage: 63.0 KB
```

In [35]:

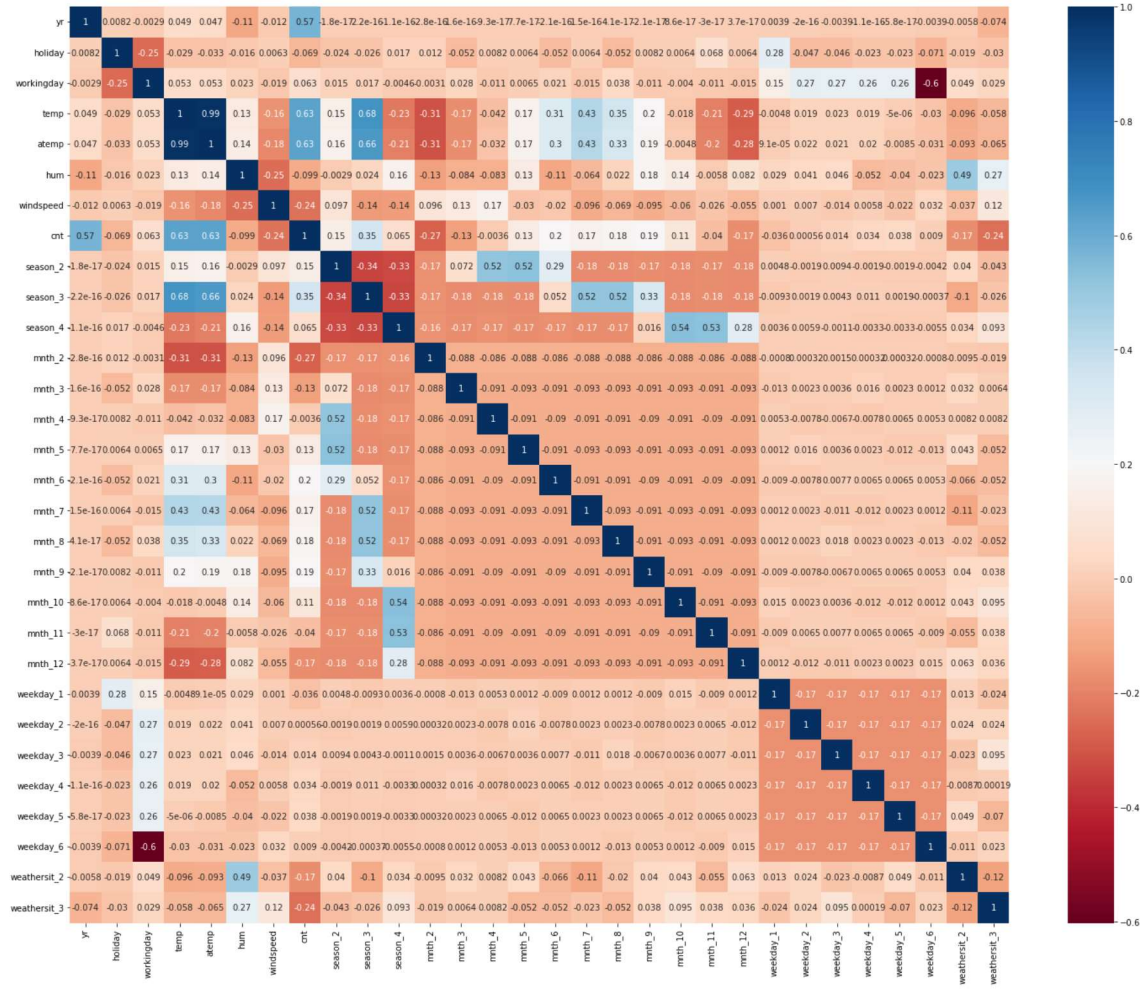
```
# Build boxplot of all categorical variables (before creating dummie
# to see how each of the predictor variable stackup against the targ

plt.figure(figsize=(25, 10))
plt.subplot(2,3,1)
sns.boxplot(x = 'season', y = 'cnt', data = bike)
plt.subplot(2,3,2)
sns.boxplot(x = 'mnth', y = 'cnt', data = bike)
plt.subplot(2,3,3)
sns.boxplot(x = 'weathersit', y = 'cnt', data = bike)
plt.subplot(2,3,4)
sns.boxplot(x = 'holiday', y = 'cnt', data = bike)
plt.subplot(2,3,5)
sns.boxplot(x = 'weekday', y = 'cnt', data = bike)
plt.subplot(2,3,6)
sns.boxplot(x = 'workingday', y = 'cnt', data = bike)
plt.show()
```



## Correlation Matrix

In [37]:
```
plt.figure(figsize = (25,20))
sns.heatmap(bike_new.corr(), annot = True, cmap="RdBu")
plt.show()
```




# RESCALING THE FEATURES

In [38]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [39]:
```
scaler = MinMaxScaler()
```

In [40]:
```
# Checking the values before scaling
df_train.head()
```

Out[40]:

| | yr | holiday | workingday | temp | atemp | hum | windspeed | cnt | season_2 | se |
|---|---|---|---|---|---|---|---|---|---|---|
| **483** | 1 | 0 | 0 | 18.791653 | 22.50605 | 58.7083 | 7.832836 | 6304 | 1 | |
| **650** | 1 | 0 | 0 | 16.126653 | 19.56980 | 49.4583 | 9.791514 | 7109 | 0 | |
| **212** | 0 | 0 | 1 | 31.638347 | 35.16460 | 55.0833 | 10.500039 | 4266 | 0 | |
| **714** | 1 | 0 | 0 | 14.862500 | 18.49690 | 83.8750 | 6.749714 | 3786 | 0 | |
| **8** | 0 | 0 | 0 | 5.671653 | 5.80875 | 43.4167 | 24.250650 | 822 | 0 | |

5 rows × 30 columns

In [41]:
```
df_train.columns
```

Out[41]:
```
Index(['yr', 'holiday', 'workingday', 'temp', 'atemp', 'hum', 'windspee
d',
       'cnt', 'season_2', 'season_3', 'season_4', 'mnth_2', 'mnth_3',
'mnth_4',
       'mnth_5', 'mnth_6', 'mnth_7', 'mnth_8', 'mnth_9', 'mnth_10', 'mn
th_11',
       'mnth_12', 'weekday_1', 'weekday_2', 'weekday_3', 'weekday_4',
       'weekday_5', 'weekday_6', 'weathersit_2', 'weathersit_3'],
      dtype='object')
```

In [42]:
```
# Apply scaler() to all the numeric variables

num_vars = ['temp', 'atemp', 'hum', 'windspeed','cnt']

df_train[num_vars] = scaler.fit_transform(df_train[num_vars])
```

In [43]:
```
# Checking values after scaling
df_train.head()
```

Out[43]:

| | yr | holiday | workingday | temp | atemp | hum | windspeed | cnt | season_2 |
|---|---|---|---|---|---|---|---|---|---|
| **483** | 1 | 0 | 0 | 0.497426 | 0.487055 | 0.609956 | 0.194850 | 0.722734 | 1 |
| **650** | 1 | 0 | 0 | 0.416433 | 0.409971 | 0.513852 | 0.255118 | 0.815347 | 0 |
| **212** | 0 | 0 | 1 | 0.887856 | 0.819376 | 0.572294 | 0.276919 | 0.488265 | 0 |
| **714** | 1 | 0 | 0 | 0.378013 | 0.381804 | 0.871429 | 0.161523 | 0.433042 | 0 |
| **8** | 0 | 0 | 0 | 0.098690 | 0.048706 | 0.451083 | 0.700017 | 0.092039 | 0 |

5 rows × 30 columns

In [44]:
```
1 df_train.describe()
```

Out[44]:

| | yr | holiday | workingday | temp | atemp | hum | windspee |
|---|---|---|---|---|---|---|---|
| **count** | 510.000000 | 510.000000 | 510.000000 | 510.000000 | 510.000000 | 510.000000 | 510.00000 |
| **mean** | 0.501961 | 0.023529 | 0.682353 | 0.540901 | 0.515631 | 0.647390 | 0.34631 |
| **std** | 0.500487 | 0.151726 | 0.466018 | 0.227898 | 0.213626 | 0.149722 | 0.16026 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.00000 |
| **25%** | 0.000000 | 0.000000 | 0.000000 | 0.343228 | 0.335807 | 0.536147 | 0.23078 |
| **50%** | 1.000000 | 0.000000 | 1.000000 | 0.540519 | 0.525578 | 0.646367 | 0.32563 |
| **75%** | 1.000000 | 0.000000 | 1.000000 | 0.740406 | 0.692378 | 0.757900 | 0.43428 |
| **max** | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.00000 |

8 rows × 30 columns

# BUILDING A LINEAR MODEL

## Dividing into X and Y sets for the model building

In [45]:
```
1 y_train = df_train.pop('cnt')
2 X_train = df_train
```

### RFE

Recursive feature elimination: We will be using the **LinearRegression function from SciKit Learn** for its compatibility with RFE (which is a utility from sklearn)

In [46]:
```
1 # Importing RFE and LinearRegression
2 from sklearn.feature_selection import RFE
3 from sklearn.linear_model import LinearRegression
```

In [47]:
```
1 # Running RFE with the output number of the variable equal to 15
2 lm = LinearRegression()
3 lm.fit(X_train, y_train)
4
5 rfe = RFE(lm, 15)             # running RFE
6 rfe = rfe.fit(X_train, y_train)
```

In [48]:
```
list(zip(X_train.columns,rfe.support_,rfe.ranking_))
```

Out[48]:
```
[('yr', True, 1),
 ('holiday', False, 13),
 ('workingday', True, 1),
 ('temp', True, 1),
 ('atemp', True, 1),
 ('hum', True, 1),
 ('windspeed', True, 1),
 ('season_2', True, 1),
 ('season_3', True, 1),
 ('season_4', True, 1),
 ('mnth_2', False, 7),
 ('mnth_3', True, 1),
 ('mnth_4', False, 3),
 ('mnth_5', False, 2),
 ('mnth_6', False, 4),
 ('mnth_7', False, 15),
 ('mnth_8', False, 5),
 ('mnth_9', True, 1),
 ('mnth_10', True, 1),
 ('mnth_11', False, 8),
 ('mnth_12', False, 14),
 ('weekday_1', False, 6),
 ('weekday_2', False, 12),
 ('weekday_3', False, 10),
 ('weekday_4', False, 11),
 ('weekday_5', False, 9),
 ('weekday_6', True, 1),
 ('weathersit_2', True, 1),
 ('weathersit_3', True, 1)]
```

In [49]:
```
col = X_train.columns[rfe.support_]
col
```

Out[49]:
```
Index(['yr', 'workingday', 'temp', 'atemp', 'hum', 'windspeed', 'season
_2',
       'season_3', 'season_4', 'mnth_3', 'mnth_9', 'mnth_10', 'weekday_
6',
       'weathersit_2', 'weathersit_3'],
      dtype='object')
```

In [50]:
```
X_train.columns[~rfe.support_]
```

Out[50]:
```
Index(['holiday', 'mnth_2', 'mnth_4', 'mnth_5', 'mnth_6', 'mnth_7', 'mn
th_8',
       'mnth_11', 'mnth_12', 'weekday_1', 'weekday_2', 'weekday_3',
       'weekday_4', 'weekday_5'],
      dtype='object')
```

In [51]:
```
# Creating X_test dataframe with RFE selected variables
X_train_rfe = X_train[col]
```

# Building Linear Model using 'STATS MODEL'

# Model 1

## VIF Check

In [52]:
```
# Check for the VIF values of the feature variables.
from statsmodels.stats.outliers_influence import variance_inflation_

# Create a dataframe that will contain the names of all the feature
vif = pd.DataFrame()
vif['Features'] = X_train_rfe.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe.values, i) for i
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[52]:

| | Features | VIF |
|---|---|---|
| 2 | temp | 384.22 |
| 3 | atemp | 363.12 |
| 4 | hum | 17.52 |
| 7 | season_3 | 7.09 |
| 5 | windspeed | 4.71 |
| 1 | workingday | 4.61 |
| 6 | season_2 | 3.54 |
| 8 | season_4 | 3.01 |
| 13 | weathersit_2 | 2.14 |
| 0 | yr | 2.02 |
| 12 | weekday_6 | 1.80 |
| 11 | mnth_10 | 1.66 |
| 10 | mnth_9 | 1.28 |
| 9 | mnth_3 | 1.20 |
| 14 | weathersit_3 | 1.17 |

In [53]:
```
import statsmodels.api as sm

# Add a constant
X_train_lm1 = sm.add_constant(X_train_rfe)

# Create a first fitted model
lr1 = sm.OLS(y_train, X_train_lm1).fit()
```

In [54]:
```
# Check the parameters obtained

lr1.params
```

Out[54]:
```
const           0.195340
yr              0.228741
workingday      0.040787
temp            0.433878
atemp           0.058635
hum            -0.178382
windspeed      -0.184925
season_2        0.130228
season_3        0.079599
season_4        0.153475
mnth_3          0.047149
mnth_9          0.100017
mnth_10         0.054370
weekday_6       0.054618
weathersit_2   -0.047472
weathersit_3   -0.271174
dtype: float64
```

In [55]:

```
# Print a summary of the linear regression model obtained
print(lr1.summary())
```

OLS Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | cnt | R-squared: | 0.842 |
| Model: | OLS | Adj. R-squared: | 0.837 |
| Method: | Least Squares | F-statistic: | 175.1 |
| Date: | Sun, 07 Jan 2024 | Prob (F-statistic): | 1.28e-186 |
| Time: | 23:12:47 | Log-Likelihood: | 509.26 |
| No. Observations: | 510 | AIC: | -986.5 |
| Df Residuals: | 494 | BIC: | -918.8 |
| Df Model: | 15 | | |
| Covariance Type: | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| const | 0.1953 | 0.030 | 6.576 | 0.000 | 0.137 | 0.254 |
| yr | 0.2287 | 0.008 | 28.013 | 0.000 | 0.213 | 0.245 |
| workingday | 0.0408 | 0.011 | 3.705 | 0.000 | 0.019 | 0.062 |
| temp | 0.4339 | 0.134 | 3.238 | 0.001 | 0.171 | 0.697 |
| atemp | 0.0586 | 0.137 | 0.427 | 0.670 | -0.211 | 0.328 |
| hum | -0.1784 | 0.037 | -4.777 | 0.000 | -0.252 | -0.105 |
| windspeed | -0.1849 | 0.028 | -6.612 | 0.000 | -0.240 | -0.130 |
| season_2 | 0.1302 | 0.015 | 8.575 | 0.000 | 0.100 | 0.160 |
| season_3 | 0.0796 | 0.021 | 3.818 | 0.000 | 0.039 | 0.121 |
| season_4 | 0.1535 | 0.014 | 10.765 | 0.000 | 0.125 | 0.181 |
| mnth_3 | 0.0471 | 0.016 | 2.958 | 0.003 | 0.016 | 0.078 |
| mnth_9 | 0.1000 | 0.016 | 6.303 | 0.000 | 0.069 | 0.131 |
| mnth_10 | 0.0544 | 0.018 | 3.046 | 0.002 | 0.019 | 0.089 |
| weekday_6 | 0.0546 | 0.014 | 3.818 | 0.000 | 0.027 | 0.083 |
| weathersit_2 | -0.0475 | 0.011 | -4.455 | 0.000 | -0.068 | -0.027 |
| weathersit_3 | -0.2712 | 0.028 | -9.542 | 0.000 | -0.327 | -0.215 |

| | | | |
|---|---|---|---|
| Omnibus: | 92.576 | Durbin-Watson: | 2.037 |

```
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
221.202
Skew:                          -0.933   Prob(JB):
9.26e-49
Kurtosis:                       5.632   Cond. No.
85.8
==============================================================================
=======

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

## Model 2

- Removing the variable 'atemp' based on its High p-value & High VIF

In [56]:
```
X_train_new = X_train_rfe.drop(["atemp"], axis = 1)
```

## VIF Check

In [57]:
```
# Check for the VIF values of the feature variables.
from statsmodels.stats.outliers_influence import variance_inflation_

# Create a dataframe that will contain the names of all the feature
vif = pd.DataFrame()
vif['Features'] = X_train_new.columns
vif['VIF'] = [variance_inflation_factor(X_train_new.values, i) for i
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[57]:

| | Features | VIF |
|---|---|---|
| 2 | temp | 23.21 |
| 3 | hum | 17.23 |
| 6 | season_3 | 7.01 |
| 1 | workingday | 4.60 |
| 4 | windspeed | 4.55 |
| 5 | season_2 | 3.54 |
| 7 | season_4 | 3.01 |
| 12 | weathersit_2 | 2.14 |
| 0 | yr | 2.02 |
| 11 | weekday_6 | 1.79 |
| 10 | mnth_10 | 1.66 |
| 9 | mnth_9 | 1.28 |
| 8 | mnth_3 | 1.20 |
| 13 | weathersit_3 | 1.17 |

In [58]:
```
# Add a constant
X_train_lm2 = sm.add_constant(X_train_new)

# Create a first fitted model
lr2 = sm.OLS(y_train, X_train_lm2).fit()
```

In [59]:
```
# Check the parameters obtained

lr2.params
```

Out[59]:
```
const           0.196221
yr              0.228723
workingday      0.040773
temp            0.489280
hum            -0.177805
windspeed      -0.187198
season_2        0.130352
season_3        0.078664
season_4        0.153732
mnth_3          0.047295
mnth_9          0.100029
mnth_10         0.054438
weekday_6       0.054705
weathersit_2   -0.047620
weathersit_3   -0.271535
dtype: float64
```

In [60]:

```
# Print a summary of the linear regression model obtained
print(lr2.summary())
```

```
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:                    cnt   R-squared:
0.842
Model:                            OLS   Adj. R-squared:
0.837
Method:                 Least Squares   F-statistic:
187.9
Date:                Sun, 07 Jan 2024   Prob (F-statistic):          1.
00e-187
Time:                        23:12:59   Log-Likelihood:
509.17
No. Observations:                 510   AIC:
-988.3
Df Residuals:                     495   BIC:
-924.8
Df Model:                          14
Covariance Type:            nonrobust
==============================================================================
=========
                   coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
---------
const            0.1962      0.030      6.627      0.000       0.138
0.254
yr               0.2287      0.008     28.034      0.000       0.213
0.245
workingday       0.0408      0.011      3.706      0.000       0.019
0.062
temp             0.4893      0.034     14.595      0.000       0.423
0.555
hum             -0.1778      0.037     -4.769      0.000      -0.251
-0.105
windspeed       -0.1872      0.027     -6.823      0.000      -0.241
-0.133
season_2         0.1304      0.015      8.592      0.000       0.101
0.160
season_3         0.0787      0.021      3.797      0.000       0.038
0.119
season_4         0.1537      0.014     10.802      0.000       0.126
0.182
mnth_3           0.0473      0.016      2.971      0.003       0.016
0.079
mnth_9           0.1000      0.016      6.309      0.000       0.069
0.131
mnth_10          0.0544      0.018      3.052      0.002       0.019
0.089
weekday_6        0.0547      0.014      3.828      0.000       0.027
0.083
weathersit_2    -0.0476      0.011     -4.475      0.000      -0.069
-0.027
weathersit_3    -0.2715      0.028     -9.567      0.000      -0.327
-0.216
==============================================================================
=======
Omnibus:                       92.002   Durbin-Watson:
2.038
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
219.387
```

```
Skew:                          -0.929   Prob(JB):
2.29e-48
Kurtosis:                       5.622   Cond. No.
21.2
==============================================================================
=======

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

## Model 3

- Removing the variable 'hum' based on its Very High 'VIF' value.
- Even though the VIF of hum is second highest, we decided to drop 'hum' and not 'temp' based on general knowledge that temperature can be an important factor for a business like bike rentals, and wanted to retain 'temp'.

In [61]:
```
X_train_new = X_train_new.drop(["hum"], axis = 1)
```

## VIF Check

In [62]:
```
# Check for the VIF values of the feature variables.
from statsmodels.stats.outliers_influence import variance_inflation_

# Create a dataframe that will contain the names of all the feature
vif = pd.DataFrame()
vif['Features'] = X_train_new.columns
vif['VIF'] = [variance_inflation_factor(X_train_new.values, i) for i
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[62]:

| | Features | VIF |
|---|---|---|
| 2 | temp | 16.81 |
| 5 | season_3 | 6.75 |
| 3 | windspeed | 4.27 |
| 1 | workingday | 4.11 |
| 4 | season_2 | 3.51 |
| 6 | season_4 | 2.89 |
| 0 | yr | 2.02 |
| 9 | mnth_10 | 1.66 |
| 10 | weekday_6 | 1.66 |
| 11 | weathersit_2 | 1.54 |
| 8 | mnth_9 | 1.27 |
| 7 | mnth_3 | 1.20 |
| 12 | weathersit_3 | 1.08 |

In [63]:
```
# Add a constant
X_train_lm3 = sm.add_constant(X_train_new)

# Create a first fitted model
lr3 = sm.OLS(y_train, X_train_lm3).fit()
```

In [64]:
```
1 lr3.params
```

Out[64]:
```
const           0.091594
yr              0.233129
workingday      0.042443
temp            0.456709
windspeed      -0.148815
season_2        0.131914
season_3        0.087922
season_4        0.150243
mnth_3          0.055303
mnth_9          0.091371
mnth_10         0.053320
weekday_6       0.055451
weathersit_2   -0.077149
weathersit_3   -0.324223
dtype: float64
```

In [65]:

```
# Print a summary of the linear regression model obtained
print(lr3.summary())
```

OLS Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | cnt | R-squared: | 0.834 |
| Model: | OLS | Adj. R-squared: | 0.830 |
| Method: | Least Squares | F-statistic: | 192.2 |
| Date: | Sun, 07 Jan 2024 | Prob (F-statistic): | 4.52e-184 |
| Time: | 23:13:05 | Log-Likelihood: | 497.71 |
| No. Observations: | 510 | AIC: | -967.4 |
| Df Residuals: | 496 | BIC: | -908.1 |
| Df Model: | 13 | | |
| Covariance Type: | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| const | 0.0916 | 0.020 | 4.509 | 0.000 | 0.052 | 0.132 |
| yr | 0.2331 | 0.008 | 28.149 | 0.000 | 0.217 | 0.249 |
| workingday | 0.0424 | 0.011 | 3.778 | 0.000 | 0.020 | 0.065 |
| temp | 0.4567 | 0.034 | 13.620 | 0.000 | 0.391 | 0.523 |
| windspeed | -0.1488 | 0.027 | -5.553 | 0.000 | -0.201 | -0.096 |
| season_2 | 0.1319 | 0.015 | 8.512 | 0.000 | 0.101 | 0.162 |
| season_3 | 0.0879 | 0.021 | 4.172 | 0.000 | 0.047 | 0.129 |
| season_4 | 0.1502 | 0.015 | 10.346 | 0.000 | 0.122 | 0.179 |
| mnth_3 | 0.0553 | 0.016 | 3.419 | 0.001 | 0.024 | 0.087 |
| mnth_9 | 0.0914 | 0.016 | 5.678 | 0.000 | 0.060 | 0.123 |
| mnth_10 | 0.0533 | 0.018 | 2.926 | 0.004 | 0.018 | 0.089 |
| weekday_6 | 0.0555 | 0.015 | 3.798 | 0.000 | 0.027 | 0.084 |
| weathersit_2 | -0.0771 | 0.009 | -8.727 | 0.000 | -0.095 | -0.060 |
| weathersit_3 | -0.3242 | 0.027 | -12.139 | 0.000 | -0.377 | -0.272 |

| | | | |
|---|---|---|---|
| Omnibus: | 87.519 | Durbin-Watson: | 2.013 |
| Prob(Omnibus): | 0.000 | Jarque-Bera (JB): | 205.489 |
| Skew: | -0.891 | Prob(JB): | 2.39e-45 |

```
Kurtosis:                       5.548   Cond. No.
16.3
==============================================================================
=======

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

# Model 4

- Removing the variable 'season3' based on its Very High 'VIF' value.
- Even though the VIF of season3 is second highest, we decided to drop 'season3' and not 'temp' based on general knowledge that temperature can be an important factor for a business like bike rentals, and wanted to retain 'temp'.

In [66]:
```
X_train_new = X_train_new.drop(["season_3"], axis = 1)
```

## VIF Check

In [67]:
```
# Check for the VIF values of the feature variables.
from statsmodels.stats.outliers_influence import variance_inflation_

# Create a dataframe that will contain the names of all the feature
vif = pd.DataFrame()
vif['Features'] = X_train_new.columns
vif['VIF'] = [variance_inflation_factor(X_train_new.values, i) for i
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[67]:

| | Features | VIF |
|---|---|---|
| 2 | temp | 4.92 |
| 3 | windspeed | 4.15 |
| 1 | workingday | 4.07 |
| 0 | yr | 2.01 |
| 5 | season_4 | 1.98 |
| 9 | weekday_6 | 1.66 |
| 8 | mnth_10 | 1.63 |
| 4 | season_2 | 1.56 |
| 10 | weathersit_2 | 1.54 |
| 7 | mnth_9 | 1.23 |
| 6 | mnth_3 | 1.15 |
| 11 | weathersit_3 | 1.08 |

In [68]:
```
# Add a constant
X_train_lm4 = sm.add_constant(X_train_new)

# Create a first fitted model
lr4 = sm.OLS(y_train, X_train_lm4).fit()
```

In [69]:
```
# Check the parameters obtained

lr4.params
```

Out[69]:
```
const           0.076726
yr              0.231340
workingday      0.042231
temp            0.568327
windspeed      -0.153306
season_2        0.083704
season_4        0.119733
mnth_3          0.044132
mnth_9          0.102838
mnth_10         0.041927
weekday_6       0.056942
weathersit_2   -0.077340
weathersit_3   -0.316638
dtype: float64
```

In [70]:

```
# Print a summary of the linear regression model obtained
print(lr4.summary())
```

```
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:                    cnt   R-squared:
0.829
Model:                            OLS   Adj. R-squared:
0.824
Method:                 Least Squares   F-statistic:
200.2
Date:                Sun, 07 Jan 2024   Prob (F-statistic):          1.
56e-181
Time:                        23:13:18   Log-Likelihood:
488.92
No. Observations:                 510   AIC:
-951.8
Df Residuals:                     497   BIC:
-896.8
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
=========
                   coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
---------
const            0.0767      0.020      3.775      0.000       0.037
0.117
yr               0.2313      0.008     27.520      0.000       0.215
0.248
workingday       0.0422      0.011      3.699      0.000       0.020
0.065
temp             0.5683      0.021     27.663      0.000       0.528
0.609
windspeed       -0.1533      0.027     -5.633      0.000      -0.207
-0.100
season_2         0.0837      0.010      7.976      0.000       0.063
0.104
season_4         0.1197      0.013      9.390      0.000       0.095
0.145
mnth_3           0.0441      0.016      2.722      0.007       0.012
0.076
mnth_9           0.1028      0.016      6.382      0.000       0.071
0.134
mnth_10          0.0419      0.018      2.290      0.022       0.006
0.078
weekday_6        0.0569      0.015      3.838      0.000       0.028
0.086
weathersit_2    -0.0773      0.009     -8.607      0.000      -0.095
-0.060
weathersit_3    -0.3166      0.027    -11.691      0.000      -0.370
-0.263
==============================================================================
=======
Omnibus:                       70.599   Durbin-Watson:
2.047
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
140.361
Skew:                          -0.787   Prob(JB):
3.32e-31
Kurtosis:                       5.031   Cond. No.
12.4
```

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

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

# Model 5

- Removing the variable 'mnth_10' based on its Very High p-value.

In [71]:
```
X_train_new = X_train_new.drop(["mnth_10"], axis = 1)
```

## VIF Check

In [72]:
```
# Check for the VIF values of the feature variables.
from statsmodels.stats.outliers_influence import variance_inflation_

# Create a dataframe that will contain the names of all the feature
vif = pd.DataFrame()
vif['Features'] = X_train_new.columns
vif['VIF'] = [variance_inflation_factor(X_train_new.values, i) for i
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[72]:

| | Features | VIF |
|---|---|---|
| 2 | temp | 4.80 |
| 3 | windspeed | 4.11 |
| 1 | workingday | 4.07 |
| 0 | yr | 2.00 |
| 8 | weekday_6 | 1.66 |
| 4 | season_2 | 1.56 |
| 9 | weathersit_2 | 1.53 |
| 5 | season_4 | 1.41 |
| 7 | mnth_9 | 1.20 |
| 6 | mnth_3 | 1.15 |
| 10 | weathersit_3 | 1.07 |

In [73]:
```
# Add a constant
X_train_lm5 = sm.add_constant(X_train_new)

# Create a first fitted model
lr5 = sm.OLS(y_train, X_train_lm5).fit()
```

In [74]:
```
# Check the parameters obtained

lr5.params
```

Out[74]:
```
const           0.074194
yr              0.230191
workingday      0.042294
temp            0.575607
windspeed      -0.156154
season_2        0.082553
season_4        0.134820
mnth_3          0.044834
mnth_9          0.096439
weekday_6       0.057422
weathersit_2   -0.075711
weathersit_3   -0.311216
dtype: float64
```

In [75]:

```
# Print a summary of the linear regression model obtained
print(lr5.summary())
```

```
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:                    cnt   R-squared:
0.827
Model:                            OLS   Adj. R-squared:
0.823
Method:                 Least Squares   F-statistic:
216.0
Date:                Sun, 07 Jan 2024   Prob (F-statistic):          1.
39e-181
Time:                        23:13:27   Log-Likelihood:
486.24
No. Observations:                 510   AIC:
-948.5
Df Residuals:                     498   BIC:
-897.7
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
=========
                   coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
---------
const            0.0742      0.020      3.640      0.000       0.034
0.114
yr               0.2302      0.008     27.316      0.000       0.214
0.247
workingday       0.0423      0.011      3.689      0.000       0.020
0.065
temp             0.5756      0.020     28.239      0.000       0.536
0.616
windspeed       -0.1562      0.027     -5.720      0.000      -0.210
-0.103
season_2         0.0826      0.011      7.842      0.000       0.062
0.103
season_4         0.1348      0.011     12.297      0.000       0.113
0.156
mnth_3           0.0448      0.016      2.754      0.006       0.013
0.077
mnth_9           0.0964      0.016      6.051      0.000       0.065
0.128
weekday_6        0.0574      0.015      3.855      0.000       0.028
0.087
weathersit_2    -0.0757      0.009     -8.417      0.000      -0.093
-0.058
weathersit_3    -0.3112      0.027    -11.486      0.000      -0.364
-0.258
==============================================================================
=======
Omnibus:                       62.037   Durbin-Watson:
2.027
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
114.440
Skew:                          -0.729   Prob(JB):
1.41e-25
Kurtosis:                       4.806   Cond. No.
12.4
==============================================================================
=======
```

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

# Model 6

- Removing the variable 'mnth_3' based on its High 'p-value'.

In [76]:
```
X_train_new = X_train_new.drop(["mnth_3"], axis = 1)
```

## VIF Check

In [77]:
```
# Check for the VIF values of the feature variables.
from statsmodels.stats.outliers_influence import variance_inflation_

# Create a dataframe that will contain the names of all the feature
vif = pd.DataFrame()
vif['Features'] = X_train_new.columns
vif['VIF'] = [variance_inflation_factor(X_train_new.values, i) for i
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[77]:

| | Features | VIF |
|---|---|---|
| 2 | temp | 4.72 |
| 3 | windspeed | 4.02 |
| 1 | workingday | 4.01 |
| 0 | yr | 2.00 |
| 7 | weekday_6 | 1.65 |
| 4 | season_2 | 1.56 |
| 8 | weathersit_2 | 1.52 |
| 5 | season_4 | 1.38 |
| 6 | mnth_9 | 1.20 |
| 9 | weathersit_3 | 1.07 |

In [78]:
```
# Add a constant
X_train_lm6 = sm.add_constant(X_train_new)

# Create a first fitted model
lr6 = sm.OLS(y_train, X_train_lm6).fit()
```

In [79]:
```
# Check the parameters obtained

lr6.params
```

Out[79]:
```
const           0.084143
yr              0.230846
workingday      0.043203
temp            0.563615
windspeed      -0.155191
season_2        0.082706
season_4        0.128744
mnth_9          0.094743
weekday_6       0.056909
weathersit_2   -0.074807
weathersit_3   -0.306992
dtype: float64
```

In [80]:

```
# Print a summary of the linear regression model obtained
print(lr6.summary())
```

```
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:                    cnt   R-squared:
0.824
Model:                            OLS   Adj. R-squared:
0.821
Method:                 Least Squares   F-statistic:
233.8
Date:                Sun, 07 Jan 2024   Prob (F-statistic):          3.
77e-181
Time:                        23:13:30   Log-Likelihood:
482.39
No. Observations:                 510   AIC:
-942.8
Df Residuals:                     499   BIC:
-896.2
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
=========
                   coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
---------
const            0.0841      0.020      4.168      0.000       0.044
0.124
yr               0.2308      0.008     27.226      0.000       0.214
0.248
workingday       0.0432      0.012      3.745      0.000       0.021
0.066
temp             0.5636      0.020     28.119      0.000       0.524
0.603
windspeed       -0.1552      0.027     -5.648      0.000      -0.209
-0.101
season_2         0.0827      0.011      7.805      0.000       0.062
0.104
season_4         0.1287      0.011     11.910      0.000       0.108
0.150
mnth_9           0.0947      0.016      5.910      0.000       0.063
0.126
weekday_6        0.0569      0.015      3.796      0.000       0.027
0.086
weathersit_2    -0.0748      0.009     -8.268      0.000      -0.093
-0.057
weathersit_3    -0.3070      0.027    -11.274      0.000      -0.360
-0.253
==============================================================================
=======
Omnibus:                       62.965   Durbin-Watson:
2.028
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
123.899
Skew:                          -0.715   Prob(JB):
1.25e-27
Kurtosis:                       4.946   Cond. No.
12.3
==============================================================================
=======

Notes:
```

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

### Insights

- This model looks good, as there seems to be VERY LOW Multicollinearity between the predictors and the p-values for all the predictors seems to be significant. For now, we will consider this as our final model (unless the Test data metrics are not significantly close to this number).

# Final Model Interpretation

## Hypothesis Testing:

Hypothesis testing states that:

- H0:B1=B2=...=Bn=0
- H1: at least one Bi!=0

### lr6 model coefficient values

- const 0.084143
- yr 0.230846
- workingday 0.043203
- temp 0.563615
- windspeed -0.155191
- season_2 0.082706
- season_4 0.128744
- mnth_9 0.094743
- weekday_6 0.056909
- weathersit_2 -0.074807
- weathersit_3 -0.306992

### Insights

- From the lr6 model summary, it is evident that all our coefficients are not equal to zerowhich means **We REJECT the NULL HYPOTHESIS**

### F Statistics

F-Statistics is used for testing the overall significance of the Model: Higher the F-Statistics, more significant the Model is.

- F-statistic: 233.8
- Prob (F-statistic): 3.77e-181

The F-Statistics value of 233 (which is greater than 1) and the p-value of '~0.0000' states that the overall model is significant

# ASSUMPTIONS

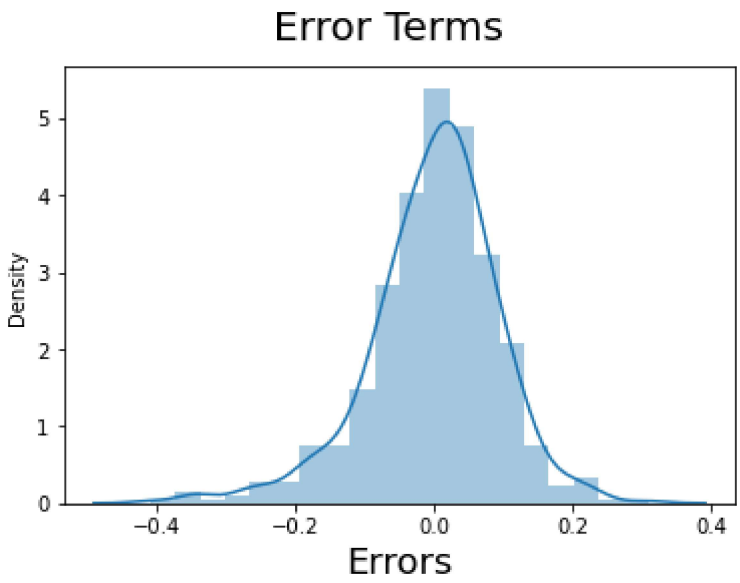
## Error terms are normally distributed with mean zero (not X, Y)

- Residual Analysis Of Training Data

In [81]:
```
y_train_pred = lr6.predict(X_train_lm6)
```

In [82]:
```
res = y_train-y_train_pred
# Plot the histogram of the error terms
fig = plt.figure()
sns.distplot((res), bins = 20)
fig.suptitle('Error Terms', fontsize = 20)                  # Plot h
plt.xlabel('Errors', fontsize = 18)                         # X-labe
```

Out[82]: Text(0.5, 0, 'Errors')



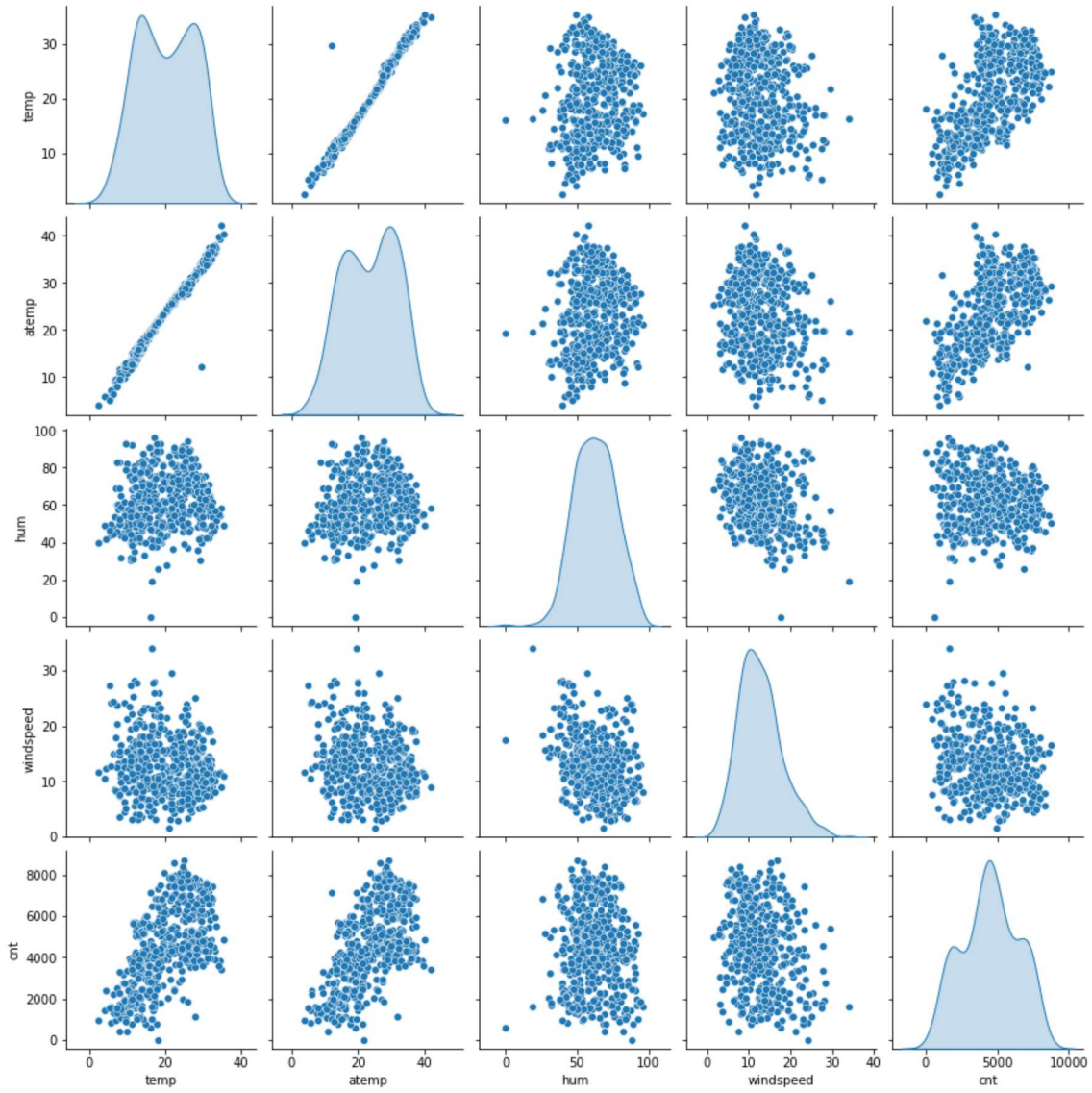

### Insights

- From the above histogram, we could see that the Residuals are normally distributed. Hence our assumption for Linear Regression is valid.

# There is a linear relationship between X and Y

In [83]:
```
bike_new=bike_new[[ 'temp', 'atemp', 'hum', 'windspeed','cnt']]

sns.pairplot(bike_num, diag_kind='kde')
plt.show()
```




## Insight

- Using the pair plot, we could see there is a linear relation between temp and atemp variable with the predictor ‘cnt’.

# There is No Multicollinearity between the predictor variables

In [84]:
```
# Check for the VIF values of the feature variables.
from statsmodels.stats.outliers_influence import variance_inflation_

# Create a dataframe that will contain the names of all the feature
vif = pd.DataFrame()
vif['Features'] = X_train_new.columns
vif['VIF'] = [variance_inflation_factor(X_train_new.values, i) for i
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[84]:

| | Features | VIF |
|---|---|---|
| 2 | temp | 4.72 |
| 3 | windspeed | 4.02 |
| 1 | workingday | 4.01 |
| 0 | yr | 2.00 |
| 7 | weekday_6 | 1.65 |
| 4 | season_2 | 1.56 |
| 8 | weathersit_2 | 1.52 |
| 5 | season_4 | 1.38 |
| 6 | mnth_9 | 1.20 |
| 9 | weathersit_3 | 1.07 |

# Insight

- From the VIF calculation we could find that there is no multicollinearity existing between the predictor variables, as all the values are within permissible range of below 5

# MAKING PREDICTION USING FINAL MODEL

Now that we have fitted the model and checked the assumptions, it's time to go ahead and make predictions using the final model (lr6)

## Applying the scaling on the test sets

In [85]:
```
# Apply scaler() to all numeric variables in test dataset. Note: we
# as we want to use the metrics that the model learned from the trai
# In other words, we want to prevent the information leak from train

num_vars = ['temp', 'atemp', 'hum', 'windspeed','cnt']

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

In [86]:
```
df_test.head()
```

Out[86]:

| | yr | holiday | workingday | temp | atemp | hum | windspeed | cnt | season_2 |
|---|---|---|---|---|---|---|---|---|---|
| **22** | 0 | 0 | 0 | 0.046591 | 0.025950 | 0.453529 | 0.462217 | 0.110907 | 0 |
| **468** | 1 | 0 | 0 | 0.543115 | 0.536771 | 0.522511 | 0.347424 | 0.855729 | 1 |
| **553** | 1 | 0 | 0 | 0.951196 | 0.933712 | 0.596104 | 0.212829 | 0.534975 | 0 |
| **504** | 1 | 0 | 0 | 0.699909 | 0.662746 | 0.551083 | 0.478229 | 0.817648 | 1 |
| **353** | 0 | 0 | 1 | 0.407087 | 0.416610 | 0.618615 | 0.080770 | 0.428900 | 0 |

5 rows × 30 columns

In [87]:
```
df_test.describe()
```

Out[87]:

| | yr | holiday | workingday | temp | atemp | hum | windspee |
|---|---|---|---|---|---|---|---|
| **count** | 219.000000 | 219.000000 | 219.000000 | 219.000000 | 219.000000 | 219.000000 | 219.00000 |
| **mean** | 0.493151 | 0.041096 | 0.684932 | 0.551225 | 0.527528 | 0.662567 | 0.34670 |
| **std** | 0.501098 | 0.198967 | 0.465607 | 0.229463 | 0.215434 | 0.143562 | 0.15955 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.046591 | 0.025950 | 0.301299 | 0.07309 |
| **25%** | 0.000000 | 0.000000 | 0.000000 | 0.356479 | 0.348019 | 0.553031 | 0.23268 |
| **50%** | 0.000000 | 0.000000 | 1.000000 | 0.557653 | 0.549198 | 0.662338 | 0.32820 |
| **75%** | 1.000000 | 0.000000 | 1.000000 | 0.751309 | 0.709163 | 0.762338 | 0.43570 |
| **max** | 1.000000 | 1.000000 | 1.000000 | 0.984424 | 0.980934 | 1.010390 | 0.82438 |

8 rows × 30 columns

## Dividing into X_test and y_test

In [88]:
```
y_test = df_test.pop('cnt')
X_test = df_test
X_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 219 entries, 22 to 313
Data columns (total 29 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   yr            219 non-null    int64  
 1   holiday       219 non-null    int64  
 2   workingday    219 non-null    int64  
 3   temp          219 non-null    float64
 4   atemp         219 non-null    float64
 5   hum           219 non-null    float64
 6   windspeed     219 non-null    float64
 7   season_2      219 non-null    uint8  
 8   season_3      219 non-null    uint8  
 9   season_4      219 non-null    uint8  
 10  mnth_2        219 non-null    uint8  
 11  mnth_3        219 non-null    uint8  
 12  mnth_4        219 non-null    uint8  
 13  mnth_5        219 non-null    uint8  
 14  mnth_6        219 non-null    uint8  
 15  mnth_7        219 non-null    uint8  
 16  mnth_8        219 non-null    uint8  
 17  mnth_9        219 non-null    uint8  
 18  mnth_10       219 non-null    uint8  
 19  mnth_11       219 non-null    uint8  
 20  mnth_12       219 non-null    uint8  
 21  weekday_1     219 non-null    uint8  
 22  weekday_2     219 non-null    uint8  
 23  weekday_3     219 non-null    uint8  
 24  weekday_4     219 non-null    uint8  
 25  weekday_5     219 non-null    uint8  
 26  weekday_6     219 non-null    uint8  
 27  weathersit_2  219 non-null    uint8  
 28  weathersit_3  219 non-null    uint8  
dtypes: float64(4), int64(3), uint8(22)
memory usage: 18.4 KB
```

In [89]:
```
#Selecting the variables that were part of final model.
col1=X_train_new.columns
X_test=X_test[col1]
# Adding constant variable to test dataframe
X_test_lm6 = sm.add_constant(X_test)
X_test_lm6.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 219 entries, 22 to 313
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   const         219 non-null    float64
 1   yr            219 non-null    int64  
 2   workingday    219 non-null    int64  
 3   temp          219 non-null    float64
 4   windspeed     219 non-null    float64
 5   season_2      219 non-null    uint8  
 6   season_4      219 non-null    uint8  
 7   mnth_9        219 non-null    uint8  
 8   weekday_6     219 non-null    uint8  
 9   weathersit_2  219 non-null    uint8  
 10  weathersit_3  219 non-null    uint8  
dtypes: float64(3), int64(2), uint8(6)
memory usage: 11.5 KB
```

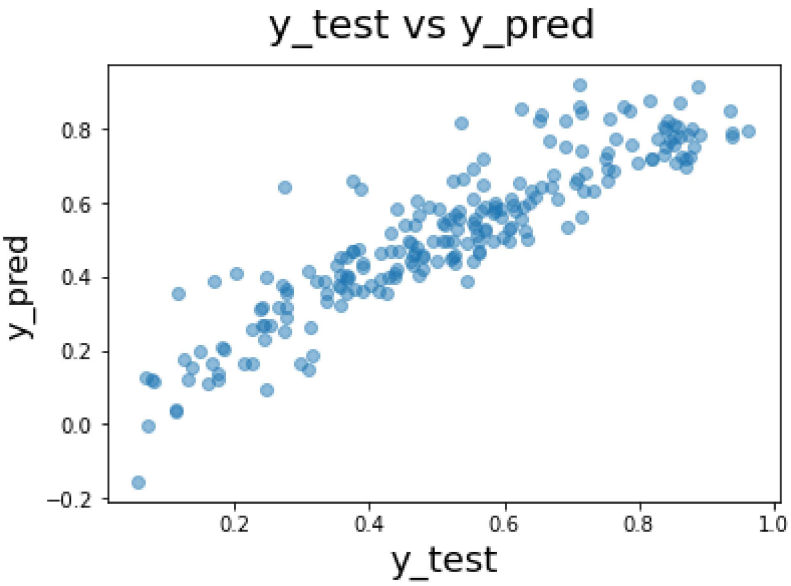
In [90]:
```
# Making predictions using the final model (lr6)

y_pred = lr6.predict(X_test_lm6)
```

# MODEL EVALUATION

In [91]:
```
# Plotting y_test and y_pred to understand the spread

fig = plt.figure()
plt.scatter(y_test, y_pred, alpha=.5)
fig.suptitle('y_test vs y_pred', fontsize = 20)              # Plot
plt.xlabel('y_test', fontsize = 18)                          # X-lab
plt.ylabel('y_pred', fontsize = 16)
plt.show()
```




# R^2 Value for TEST

In [92]:
```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

Out[92]: 0.8203092200749708

# Adjusted R^2 Value for TEST

In [93]:
```
# We already have the value of R^2 (calculated in above step)

r2=0.8203092200749708
```

In [94]:
```
# Get the shape of X_test
X_test.shape

```

Out[94]: (219, 10)

In [95]:
```
# n is number of rows in X

n = X_test.shape[0]


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

# We find the Adjusted R-squared using the formula

adjusted_r2 = 1-(1-r2)*(n-1)/(n-p-1)
adjusted_r2
```

Out[95]: 0.8116702402708829

# Final Result Comparison

- Train R^2 :0.824
- Train Adjusted R^2 :0.821
- Test R^2 :0.820
- Test Adjusted R^2 :0.812
- This seems to be a really good model that can very well 'Generalize' various datasets.

In [ ]:
```

```