In [1]:
```
#importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
#Reading the dataset
df=pd.read_csv('day.csv')
df.head()
```

Out[2]:

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

1. Inspecting the dataframe

In [3]:
```
# Check the number of rows and columns in the dataframe
df.shape
```

Out[3]: (730, 16)

In [4]:
```
# Check the column-wise info of the dataframe
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
```

In [5]:
```
# Check the summary for the numeric columns
df.describe()
```

Out[5]:

	instant	season	yr	mnth	holiday	weekday	workingday
count	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000
mean	365.500000	2.498630	0.500000	6.526027	0.028767	2.995890	0.690411
std	210.877136	1.110184	0.500343	3.450215	0.167266	2.000339	0.462641
min	1.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000
25%	183.250000	2.000000	0.000000	4.000000	0.000000	1.000000	0.000000
50%	365.500000	3.000000	0.500000	7.000000	0.000000	3.000000	1.000000
75%	547.750000	3.000000	1.000000	10.000000	0.000000	5.000000	1.000000
max	730.000000	4.000000	1.000000	12.000000	1.000000	6.000000	1.000000

In [6]:
```
#visualize missing values if any
import klib
klib.missingval_plot(df)
```

```
No missing values found in the dataset.
```

In [7]:
```
# Converting date to Pandas datetime format
df['dteday'] = pd.to_datetime(df['dteday'])
```

In [8]:
```
df.head()
```

Out[8]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	tem
0	1	2018-01-01	1	0	1	0	1	1	2	14.1108
1	2	2018-02-01	1	0	1	0	2	1	2	14.9025
2	3	2018-03-01	1	0	1	0	3	1	1	8.0509
3	4	2018-04-01	1	0	1	0	4	1	1	8.2000
4	5	2018-05-01	1	0	1	0	5	1	1	9.3052

2. Data Cleaning and Analysis

In [9]:
```
#Changing the season, weathersit, mnth, weekday columns from numeri
cal values to categorical strings
df.season=df.season.map({1:'spring', 2:'summer', 3:'fall', 4:'winte
r'})
df.weathersit=df.weathersit.map({1:'Best', 2:'Neutral', 3:'Bad',
4:'Worse'})
df.mnth=df.mnth.map({1:'Jan',2:'Feb',3:'Mar',4:'Apr',5:'May',6:'Jun
e',7:'Jul',8:'Aug',9:'Sep',10:'Oct',11:'Nov',12:'Dec'})
df.weekday=df.weekday.map({1:'Mon',2:'Tue',3:'Wed',4:'Thu',5:'Fri',
6:'Sat',0:'Sun'})
df.head()
```

Out[9]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	tem
0	1	2018-01-01	spring	0	Jan	0	Mon	1	Neutral	14.1108
1	2	2018-02-01	spring	0	Jan	0	Tue	1	Neutral	14.9025
2	3	2018-03-01	spring	0	Jan	0	Wed	1	Best	8.0509
3	4	2018-04-01	spring	0	Jan	0	Thu	1	Best	8.2000
4	5	2018-05-01	spring	0	Jan	0	Fri	1	Best	9.3052

Note: For the column weathersit, the alias is

Best :

Clear, Few clouds, Partly cloudy, Partly cloudy

Neutral:

Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist

Bad :

Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds

Worse :

Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

In [10]:
```
#The column 'instant' is very insignificant. Hence dropping that co
lumn.
df=df.drop('instant',axis=1)
df.shape
```

Out[10]: (730, 15)

In [11]:
```
#Inserting a new variable day in the dataframe.
df.insert(4,'day','')
df['day']=pd.DatetimeIndex(df['dteday']).day
df.head()
```

Out[11]:

	dteday	season	yr	mnth	day	holiday	weekday	workingday	weathersit	temp
0	2018-01-01	spring	0	Jan	1	0	Mon	1	Neutral	14.110847
1	2018-02-01	spring	0	Jan	1	0	Tue	1	Neutral	14.902598
2	2018-03-01	spring	0	Jan	1	0	Wed	1	Best	8.050924
3	2018-04-01	spring	0	Jan	1	0	Thu	1	Best	8.200000
4	2018-05-01	spring	0	Jan	1	0	Fri	1	Best	9.305237

In [12]:
```
print('------------------day values------------------')
print(df.day.value_counts())
print('--------------workingday values---------------')
print(df.workingday.value_counts())
print('----------------weekday values----------------')
print(df.weekday.value_counts())
```

```
------------------day values------------------
16    24
15    24
2     24
3     24
4     24
5     24
6     24
7     24
8     24
9     24
10    24
11    24
12    24
13    24
14    24
1     24
17    24
18    24
19    24
20    24
21    24
22    24
23    24
24    24
25    24
26    24
27    24
28    24
30    22
29    22
31    14
Name: day, dtype: int64
--------------workingday values---------------
1    504
0    226
Name: workingday, dtype: int64
----------------weekday values----------------
Mon    105
Tue    105
Wed    104
Fri    104
Sun    104
Thu    104
Sat    104
Name: weekday, dtype: int64
```

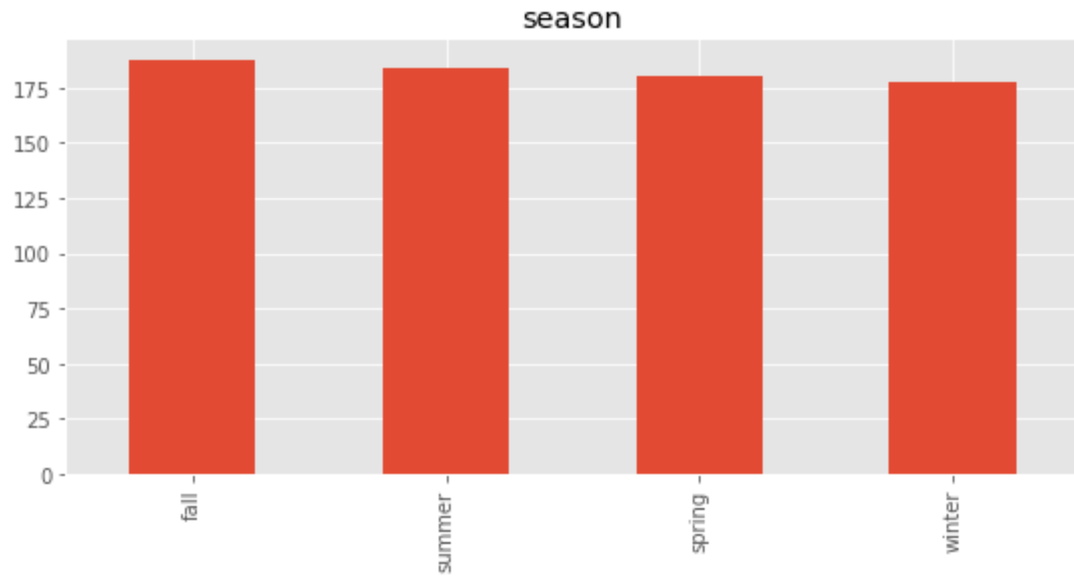
In [13]:
```
#dropping dteday
df=df.drop('dteday', axis=1)
df.shape
```

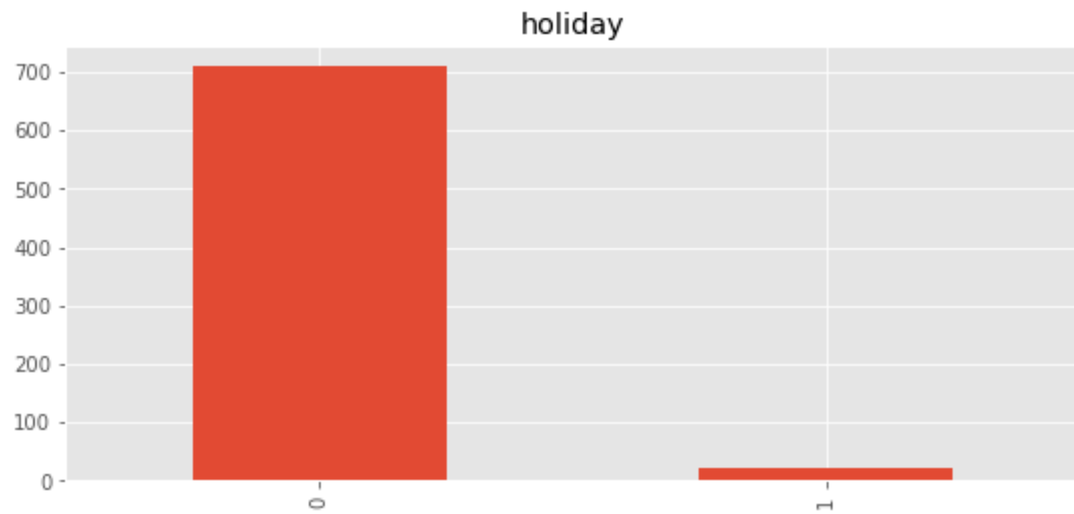
Out[13]: (730, 15)

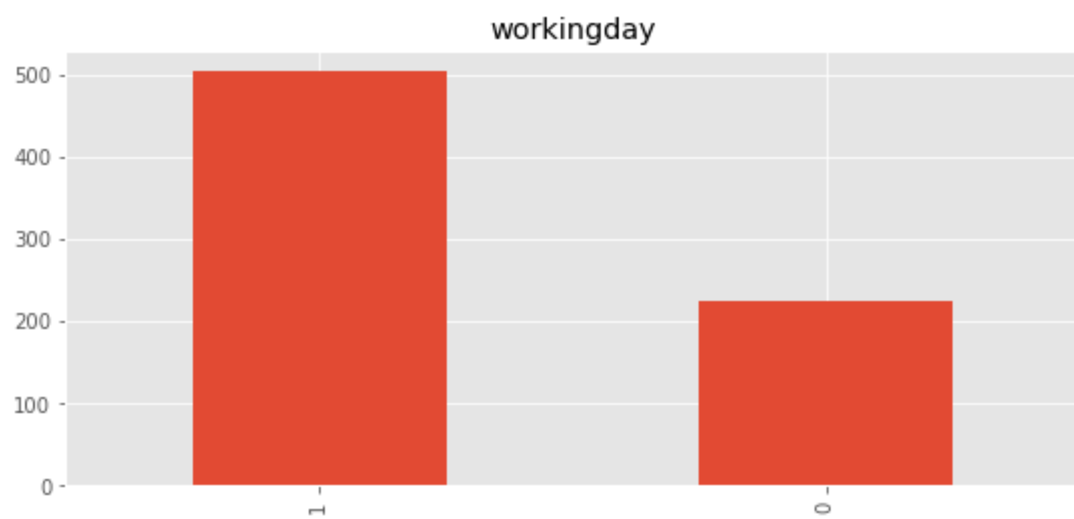
Visualization

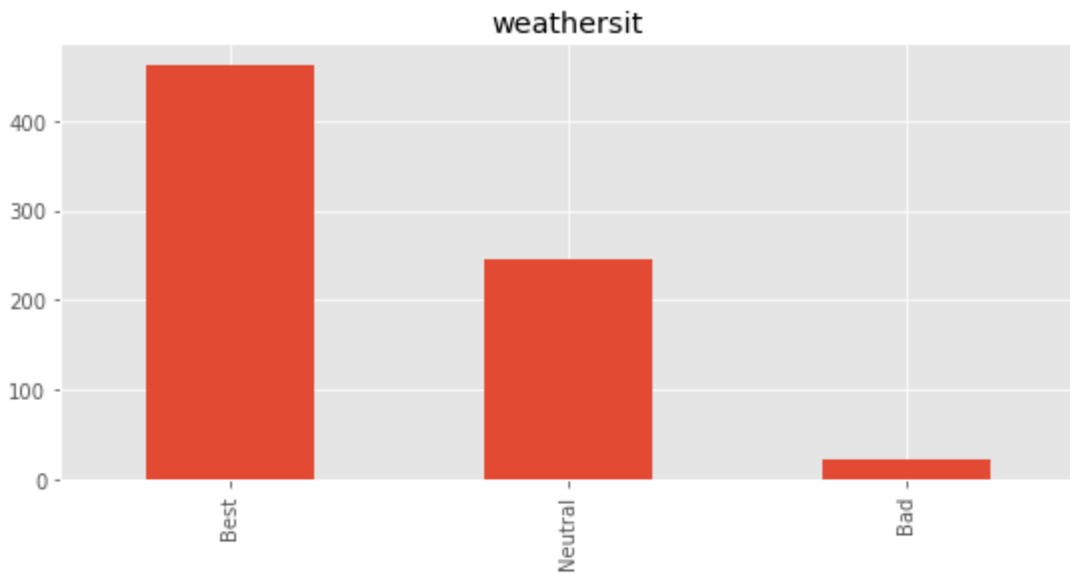
2.1 Univariate Analysis

In [14]:
```
# Univariate analysis of few seemingly significant categorical vari
ables:
univariate_categorical_cols=['season','holiday','workingday','weath
ersit']
plt.style.use('ggplot')
for column in univariate_categorical_cols:
    plt.figure(figsize=(10,4))
    plt.subplot(121)
    df[column].value_counts().plot(kind='bar')
    plt.title(column)
```


season
175
150
125
100
75
50
25
0
fall
summer
spring
winter


holiday
700
600
500
400
300
200
100
0
0
1


workingday
500
400
300
200
100
0
1
0



In [15]:
```
print('Number of holidays in 2018: ',len(df[(df['holiday']==1) & (d
f['yr']==0)]))
print('Number of holidays in 2019: ',len(df[(df['holiday']==1) & (d
f['yr']==1)]))
```

```
Number of holidays in 2018:  10
Number of holidays in 2019:  11
```

Inferences:

1. Even though the margin is minimum, the number of days in fall is maximum and winter is minimum. Number of days as per season in decreasing order: Fall, Summer, Spring, Winter.
2. The number of public holidays is 21 in 2 years. Number of holidays in 2018 and 2019 are 10 and 11 respectively
3. The number of non-working days(Public holidays+weekends) is slightly less than half the number of working days which can be favourable for bike renting for exploring different places during non working days but can be non-favourable as well since the daily commute to office during the working days can be hampered.
4. Weather situation is mostly best case scenario and neutral compared to bad and worse which is favourable for renting bikes.

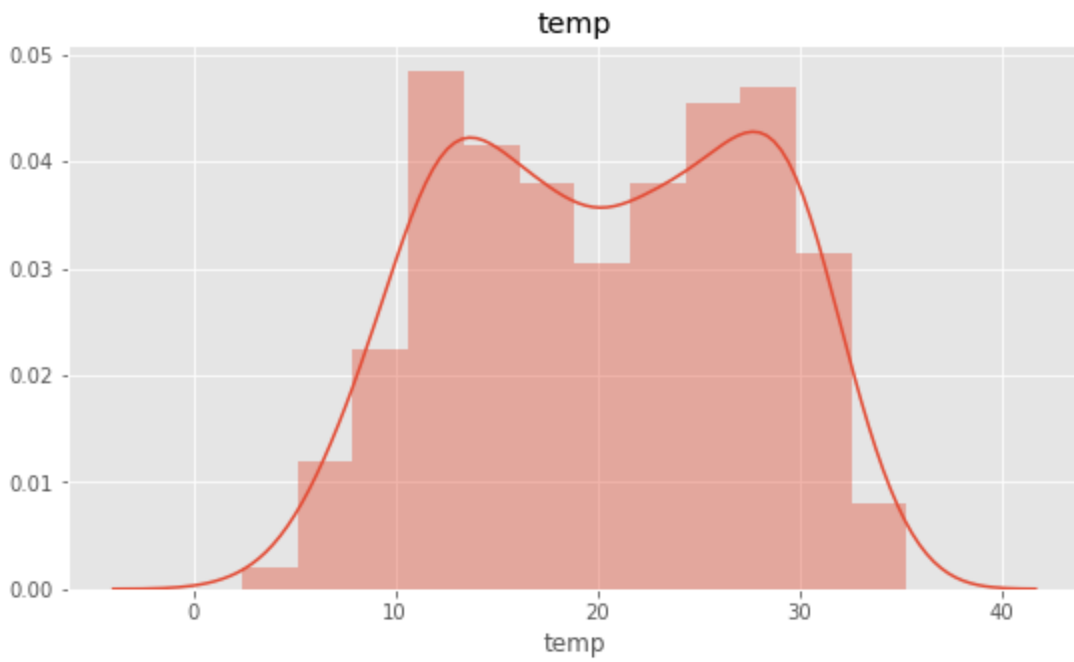
In [16]:

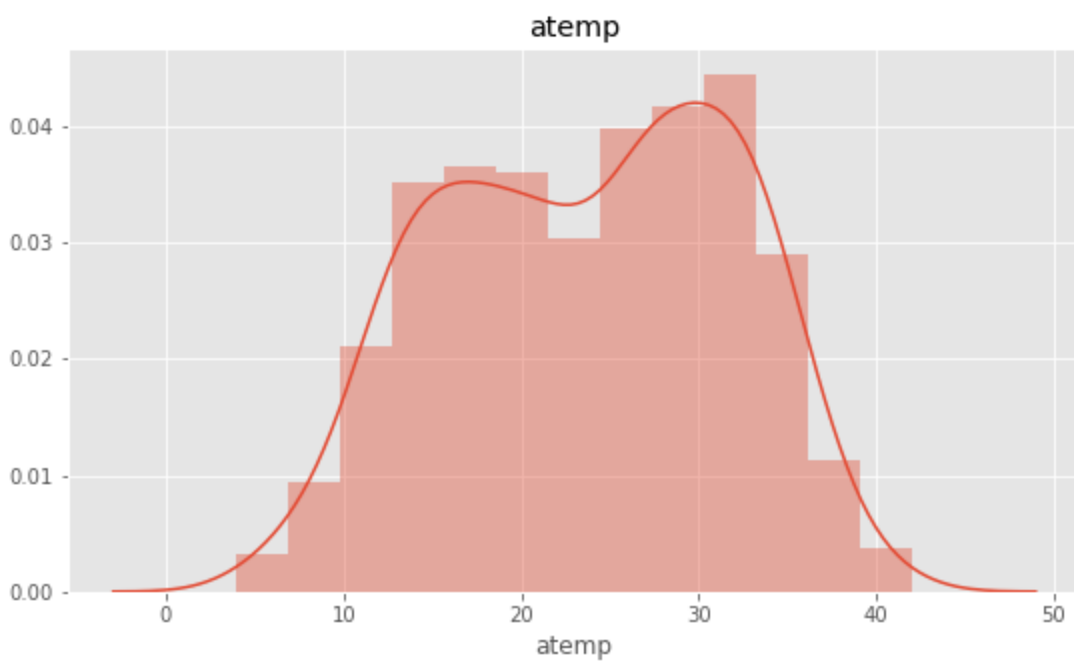
```
# Univariate analysis of few seemingly significant continuous varia
bles:
univariate_numerical_cols=df.select_dtypes(include=np.number)
univariate_numerical_cols=list(univariate_numerical_cols)
univariate_numerical_cols
univariate_continuous_var=[i for i in univariate_numerical_cols if
i not in ['yr',
 'mnth',
 'day',
 'holiday',
 'weekday',
 'workingday']]
univariate_continuous_var
```

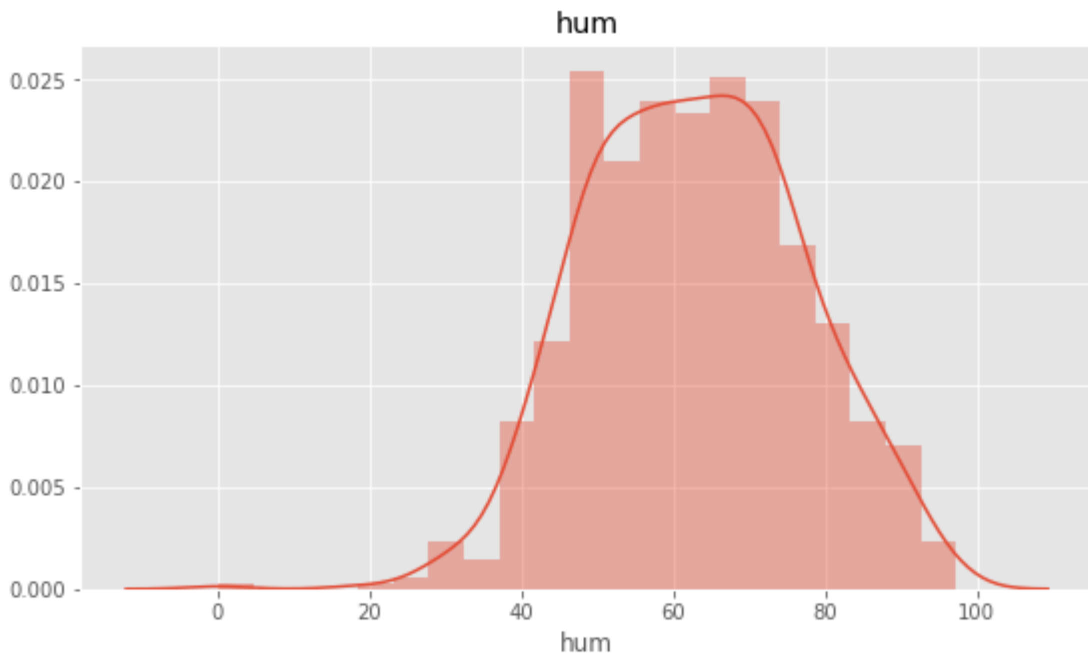
Out[16]:

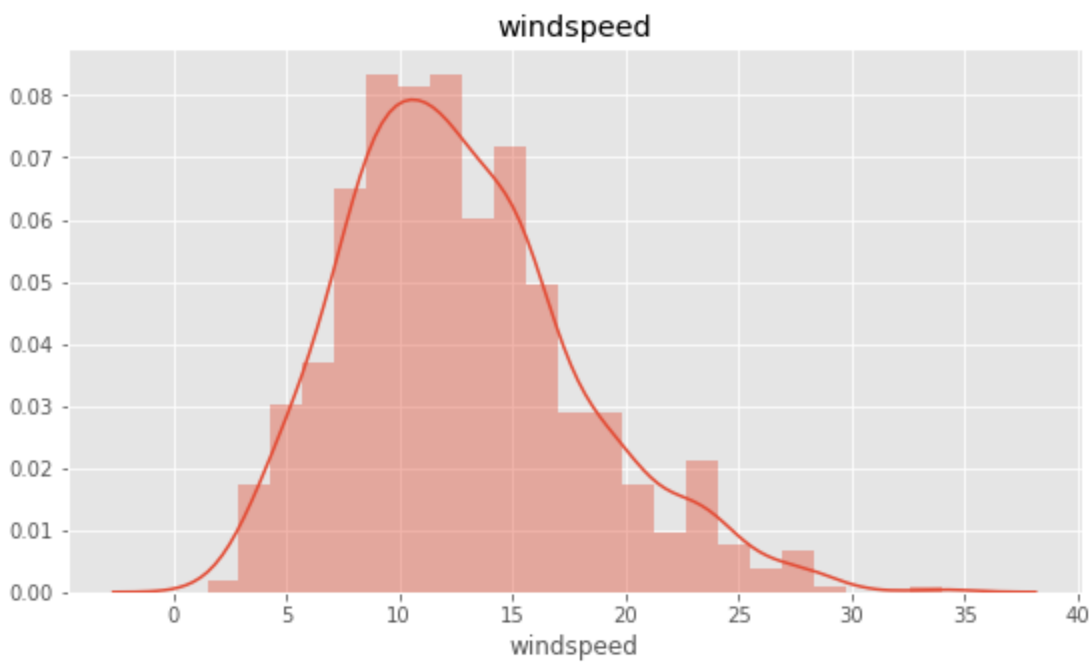
```
['temp', 'atemp', 'hum', 'windspeed', 'casual', 'registered', 'cnt']
```

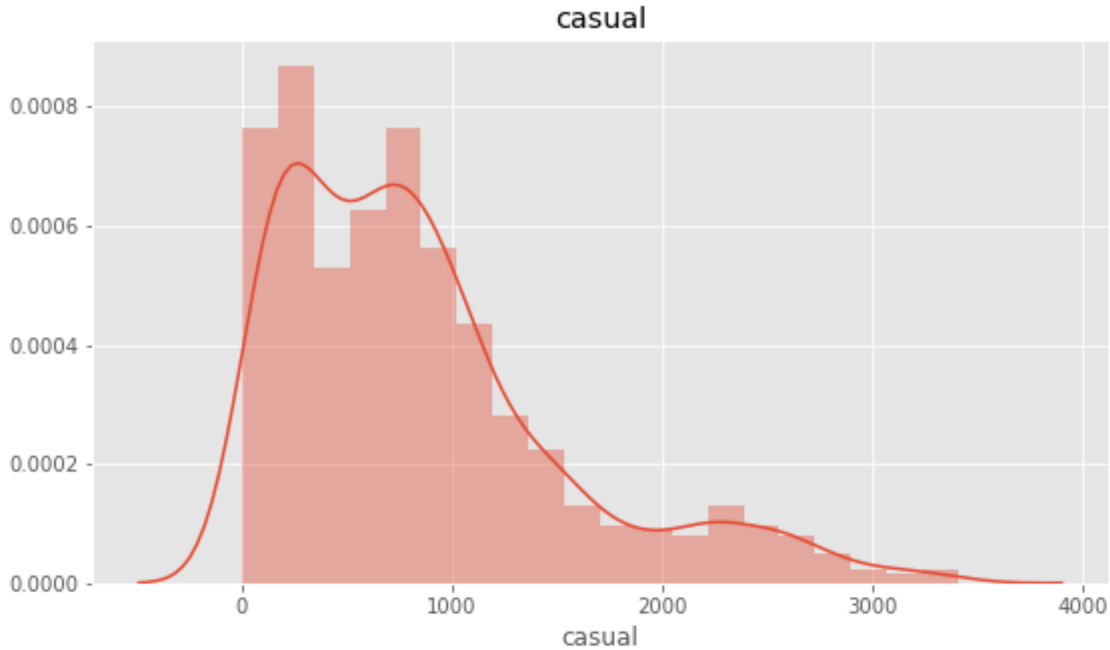
In [17]:
```
for column in univariate_continuous_var:
    plt.figure(figsize=(20,5))
    plt.subplot(121)
    sns.distplot(df[column])
    plt.title(column)
```

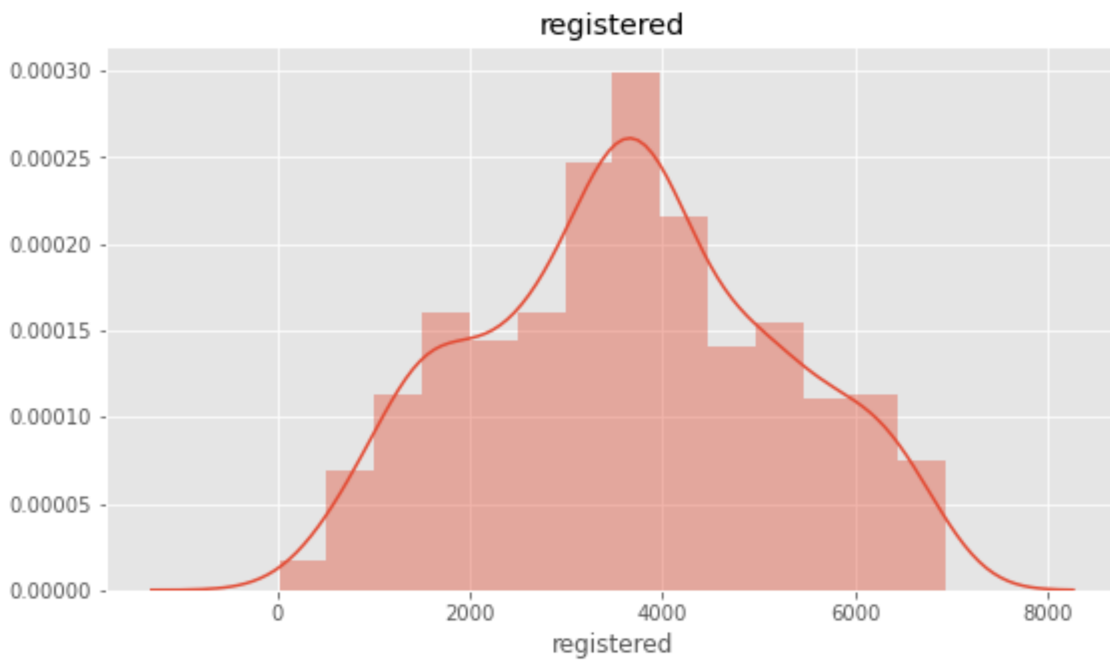

temp
0.05
0.04
0.03
0.02
0.01
0.00
0
10
20
30
40
temp

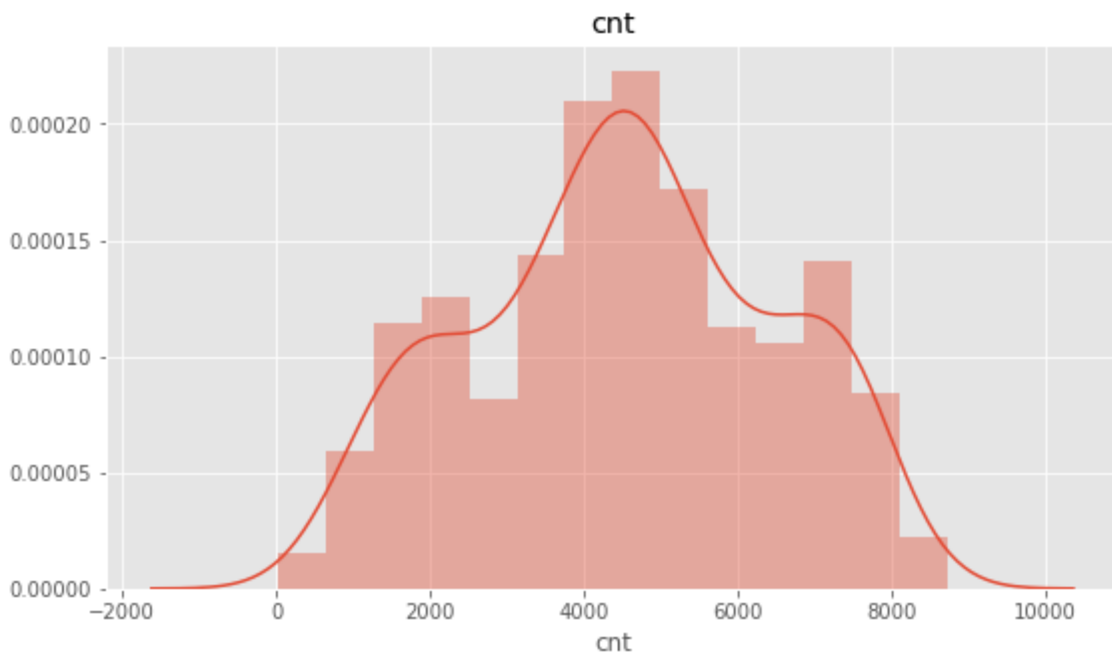

atemp
0.04
0.03
0.02
0.01
0.00
0
10
20
30
40
50
atemp


hum
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100
hum


windspeed
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
5
10
15
20
25
30
35
40
windspeed


casual
0.0008
0.0006
0.0004
0.0002
0.0000
0
1000
2000
3000
4000
casual


registered
0.00030
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
0
2000
4000
6000
8000
registered



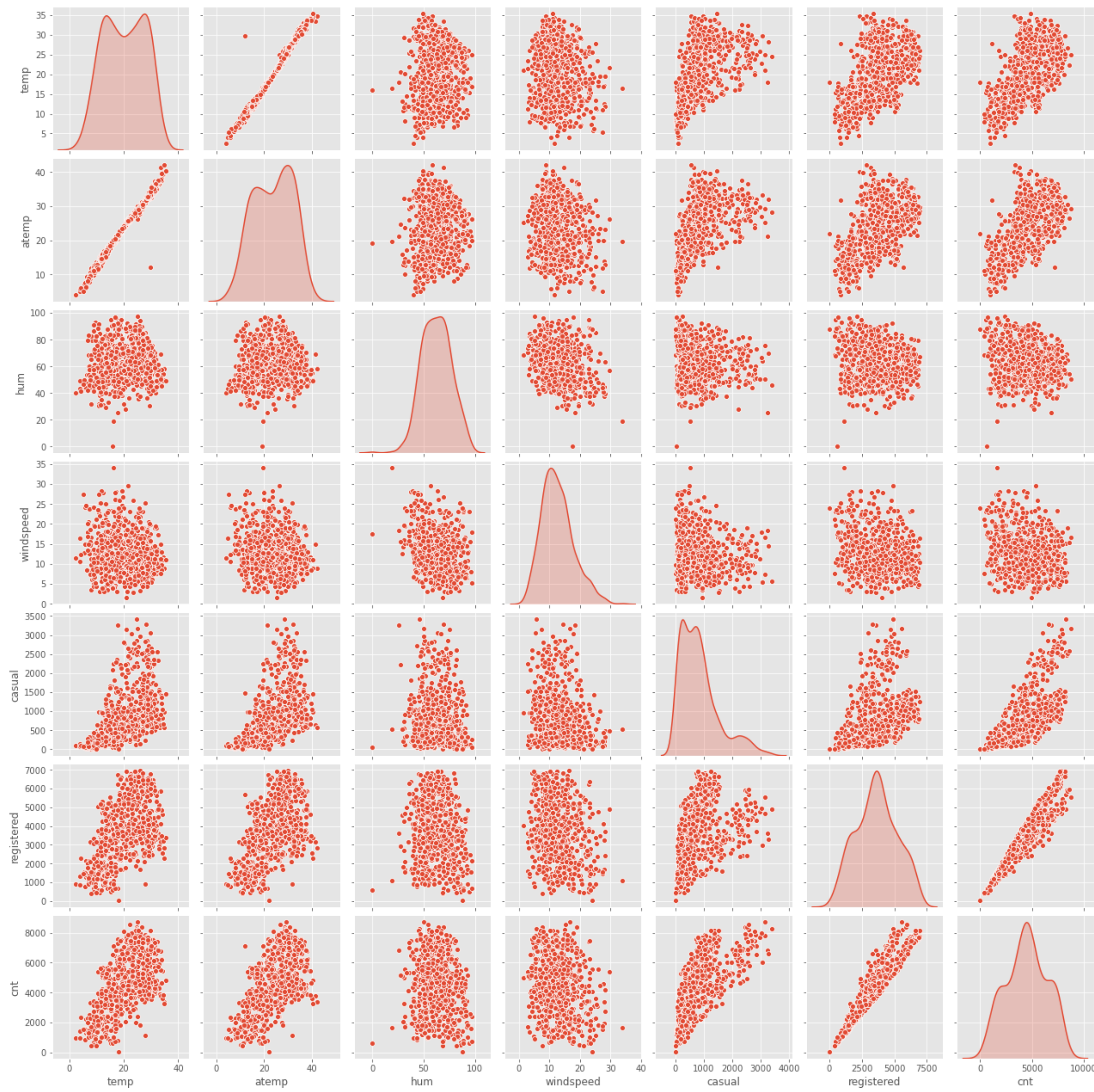
Inferences:

1. Values of temperature and feeling temperature are differently distributed.
2. Humidity is almost randomly distributed with a mean of around 61-63.
3. The KDE of windspeed is almost a normal distribution with a right skew because of a few days with windspeed over 30.
4. The spread of casual users is not normally distributed where as that of registered users is normally distributed ultimately leading to cnt to be spread normally distributed.

2.2 Bivariate Analysis

In [18]:
```
df_continuous=df[univariate_continuous_var]
```

In [19]:
```
#Bivariate analysis of continuos variables with cnt
sns.pairplot(df_continuous,diag_kind='kde')
plt.show()
```



Inference

1. Huge corelation between temp and atemp. Hence only one of the 2 variables will be in the model.
2. temp/atemp shows some linear relationship with cnt.
3. hum and windspeed doesn't show much of a linear relationship with cnt.
4. Casual and registered shows linear relationship with cnt out of which the linear relationship shown by registered users is very significant.
5. Rest there are not any significant linear relationships.

In [20]:
```
#Bivariate analysis of categorical variables with cnt
plt.figure(figsize=(30,48))
plt.subplot(8,2,1)
sns.boxplot(x='yr', y='cnt', data=df)

plt.subplot(8,2,2)
sns.barplot(x='season', y='cnt', data=df)

plt.subplot(8,2,3)
sns.boxplot(x='holiday', y='cnt', data=df)

plt.subplot(8,2,4)
sns.boxplot(x='weathersit', y='cnt', data=df)

plt.subplot(8,2,5)
sns.barplot(x='weathersit', y='windspeed', data=df)

plt.subplot(8,2,6)
sns.boxplot(x='workingday', y='cnt', data=df)

plt.subplot(8,2,7)
sns.barplot(x='mnth', y='windspeed', data=df)

plt.subplot(8,2,8)
sns.barplot(x='season', y='windspeed', data=df)

plt.subplot(8,2,9)
sns.lineplot(x='day', y='cnt', data=df)

plt.subplot(8,2,10)
sns.boxplot(x = 'mnth', y = 'cnt', data = df)

plt.subplot(8,2,11)
sns.barplot(x='mnth', y='cnt', data=df)

plt.subplot(8,2,12)
sns.barplot(x='weekday', y='cnt', data=df)

plt.subplot(8,2,13)
sns.barplot(x='mnth', y='casual', data=df)

plt.subplot(8,2,14)
sns.barplot(x='weekday', y='casual', data=df)

plt.subplot(8,2,15)
sns.barplot(x='mnth', y='registered', data=df)

plt.subplot(8,2,16)
sns.barplot(x='weekday', y='casual', data=df)

plt.show()
```


cnt
yr
cnt
spring
summer
fall
winter
season
cnt
holiday
cnt
Neutral
Best
Bad
weathersit
windspeed
Neutral
Best
Bad
weathersit
cnt
workingday
windspeed
Jan
Feb
Mar
Apr
May
June
Jul
Aug
Sep
Oct
Nov
Dec
mnth
windspeed
spring
summer
fall
winter
season
cnt
day
cnt
mnth
cnt
mnth
cnt
Mon
Tue
Wed
Thu
Fri
Sat
Sun
weekday
casual
mnth
casual
weekday
registered
mnth
casual
weekday

In [21]:

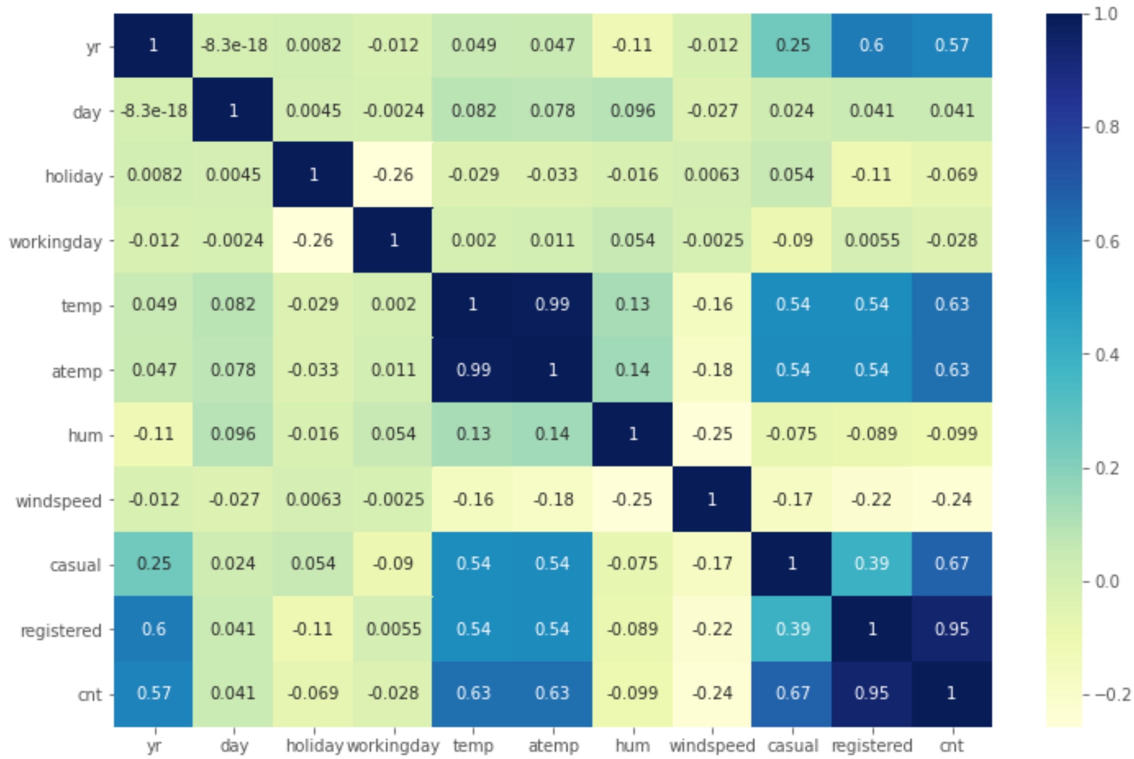
```
print('------------Winter Months-------------------------')
print('Months')
print(df[df['season']=='winter'].mnth.value_counts())
print('------------Spring Months-------------------------')
print('Months')
print(df[df['season']=='spring'].mnth.value_counts())
print('------------Summer Months-------------------------')
print('Months')
print(df[df['season']=='summer'].mnth.value_counts())
print('--------------Fall Months-------------------------')
print('Months')
print(df[df['season']=='fall'].mnth.value_counts())
```

```
------------Winter Months-------------------------
Months
Oct    62
Nov    60
Dec    40
Sep    16
Name: mnth, dtype: int64
------------Spring Months-------------------------
Months
Jan    62
Feb    56
Mar    40
Dec    22
Name: mnth, dtype: int64
------------Summer Months-------------------------
Months
May     62
Apr     60
June    40
Mar     22
Name: mnth, dtype: int64
--------------Fall Months-------------------------
Months
Aug     62
Jul     62
Sep     44
June    20
Name: mnth, dtype: int64
```

Inferences

1. The cnt in the year 2019 was way more than that in 2018. The 75th percentile of the cnt in 2018 is almost equivalent to 25 percentile in 2019.
2. Number of bikes booked according to seasons in a decreasing order: Fall, Summer, Winter and Spring.
3. The trend of increasing use of bike starts from january(lowest) till June then stays almost the same till september and then starts dropping. There's a scope to increase the bike usage in the months from january till May and from October to december. The drop of bike usage from october till December might be explained by the winter season and less bike usage from January to April might be explained by higher windspeed.
4. Days of the week doesn't matter much. Almost similar number of bikes are rented same number of times everyday in a week but Monday and tuesday have relatively less bookings.
5. The average count of bikes rented is more during non-public holidays.
6. The average count of bikes rented when the weather is situation is 'Clear, Few clouds, Partly cloudy, Partly cloudy' or 'Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist' termed as 'Best' and 'Neutral' is much more compared to other situations termed as 'Bad' and 'Worse'- Wind speed during the bad weather situations is more than 'Best' and 'neautral' weather situations and hence more number of bikes are rented in such situations.
7. The line-graph trend shows that the count of bikes rented is least from 1st-4th day, peaks from 6th-10th day in a month and again dips till 13th day and kind of stays almost constant throughout the month.
8. There was a drop of bike rents by casual users in the winters and in the first 2 months of spring and less drop in the registered users. These people may be regular office going people or fitness enthusiasts.

In [22]:
```
#Checking the colinearlity amongst the variables
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(),annot=True,cmap="YlGnBu")
plt.show()
```



Since we have casual+registered=cnt and inferences are built from casual and registered records, let's drop them since these columns seem irrelevant for the model. Also it is a given that increasing casual or registered users both will be profitable factor for the business.

In [23]:
```
df=df.drop(['casual', 'registered'],axis=1)
df.head()
```

Out[23]:

	season	yr	mnth	day	holiday	weekday	workingday	weathersit	temp	atemp
0	spring	0	Jan	1	0	Mon	1	Neutral	14.110847	18.18125
1	spring	0	Jan	1	0	Tue	1	Neutral	14.902598	17.68695
2	spring	0	Jan	1	0	Wed	1	Best	8.050924	9.47025
3	spring	0	Jan	1	0	Thu	1	Best	8.200000	10.60610
4	spring	0	Jan	1	0	Fri	1	Best	9.305237	11.46350

Also temp and atemp are very highly corelated and their respective colinearities with cnt are also same. Hence dropping atemp since feeling temperature can be relatively less accurate compared to temperature.

In [24]:
```
df=df.drop('atemp',axis=1)
```

In [25]:
```
df.head()
```

Out[25]:

	season	yr	mnth	day	holiday	weekday	workingday	weathersit	temp	hum	wi
0	spring	0	Jan	1	0	Mon	1	Neutral	14.110847	80.5833	1(
1	spring	0	Jan	1	0	Tue	1	Neutral	14.902598	69.6087	16
2	spring	0	Jan	1	0	Wed	1	Best	8.050924	43.7273	16
3	spring	0	Jan	1	0	Thu	1	Best	8.200000	59.0435	1(
4	spring	0	Jan	1	0	Fri	1	Best	9.305237	43.6957	12

3. Preparing data for modelling

In [26]:
```
#Creating Dummy variables

def dummies(x,dataframe):
    temp = pd.get_dummies(dataframe[x], drop_first = True)
    dataframe = pd.concat([dataframe, temp], axis = 1)
    dataframe.drop([x], axis = 1, inplace = True)
    return dataframe
# Applying the function to the bikeSharing

df = dummies('season',df)
df = dummies('mnth',df)
df = dummies('weekday',df)
df = dummies('weathersit',df)
df.head()
```

Out[26]:

	yr	day	holiday	workingday	temp	hum	windspeed	cnt	spring	summer	...	(
0	0	1	0	1	14.110847	80.5833	10.749882	985	1	0	...	
1	0	1	0	1	14.902598	69.6087	16.652113	801	1	0	...	
2	0	1	0	1	8.050924	43.7273	16.636703	1349	1	0	...	
3	0	1	0	1	8.200000	59.0435	10.739832	1562	1	0	...	
4	0	1	0	1	9.305237	43.6957	12.522300	1600	1	0	...	

5 rows × 30 columns

In [27]:
```
df.shape
```

Out[27]: (730, 30)

In [28]:
```
df.describe()
```

Out[28]:

	yr	day	holiday	workingday	temp	hum	windspeed
count	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000
mean	0.500000	15.720548	0.028767	0.690411	20.319259	62.765175	12.763620
std	0.500343	8.802278	0.167266	0.462641	7.506729	14.237589	5.195841
min	0.000000	1.000000	0.000000	0.000000	2.424346	0.000000	1.500244
25%	0.000000	8.000000	0.000000	0.000000	13.811885	52.000000	9.041650
50%	0.500000	16.000000	0.000000	1.000000	20.465826	62.625000	12.125325
75%	1.000000	23.000000	0.000000	1.000000	26.880615	72.989575	15.625589
max	1.000000	31.000000	1.000000	1.000000	35.328347	97.250000	34.000021

8 rows × 30 columns

3.1 Spliting the data into test and train

In [29]:
```
import sklearn
from sklearn.model_selection import train_test_split

df_train, df_test= train_test_split(df,train_size=0.7, random_state
=100)
print(df_train.shape)
print(df_test.shape)
```

```
(510, 30)
(220, 30)
```

3.2 Rescalling the features:

In [30]:
```
#Rescaling even the target variables since a target variable with a large spread of values, in turn, may result
#in large error gradient values causing weight values to change dramatically, making the learning process unstable.

from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()

need_rescale=['temp','hum','windspeed','cnt']
df_train[need_rescale]=scaler.fit_transform(df_train[need_rescale])

df_train.describe()
```

Out[30]:

	yr	day	holiday	workingday	temp	hum	windspeed
count	510.000000	510.000000	510.000000	510.000000	510.000000	510.000000	510.000000
mean	0.507843	15.631373	0.025490	0.711765	0.537440	0.650480	0.320883
std	0.500429	8.852724	0.157763	0.453386	0.225858	0.145846	0.169803
min	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	8.000000	0.000000	0.000000	0.339853	0.538643	0.199179
50%	1.000000	16.000000	0.000000	1.000000	0.542596	0.653714	0.296763
75%	1.000000	23.000000	0.000000	1.000000	0.735215	0.754830	0.414447
max	1.000000	31.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 30 columns

3.3 Splitting train dataset into X and y

In [31]:
```
y_train=df_train.pop('cnt')
X_train=df_train
```

4. Model Building

In [32]:
```
#Since the total number of variables are 30, using RFE to calculate
the best 15 variables to be used for model building

from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression

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

rfe=RFE(lm, 15)
rfe=rfe.fit(X_train,y_train)

list(zip(X_train.columns,rfe.support_,rfe.ranking_))
```

Out[32]:
```
[('yr', True, 1),
 ('day', False, 15),
 ('holiday', True, 1),
 ('workingday', False, 3),
 ('temp', True, 1),
 ('hum', True, 1),
 ('windspeed', True, 1),
 ('spring', True, 1),
 ('summer', True, 1),
 ('winter', True, 1),
 ('Aug', False, 9),
 ('Dec', True, 1),
 ('Feb', False, 2),
 ('Jan', True, 1),
 ('Jul', 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),
 ('Best', True, 1),
 ('Neutral', True, 1)]
```

In [33]:
```
col = X_train.columns[rfe.support_]
col
```

Out[33]:
```
Index(['yr', 'holiday', 'temp', 'hum', 'windspeed', 'spring', 'summe
r',
       'winter', 'Dec', 'Jan', 'Jul', 'Nov', 'Sep', 'Best', 'Neutral
'],
      dtype='object')
```

In [34]:
```
X_train.columns[~rfe.support_]
```

Out[34]:
```
Index(['day', 'workingday', 'Aug', 'Feb', 'June', 'Mar', 'May', 'Oct
', 'Mon',
       'Sat', 'Sun', 'Thu', 'Tue', 'Wed'],
      dtype='object')
```

In [35]:
```
X_train_rfe=X_train[X_train.columns[rfe.support_]]
X_train_rfe.head()
```

Out[35]:

	yr	holiday	temp	hum	windspeed	spring	summer	winter	Dec	Jan	Jul	No
576	1	0	0.815169	0.725633	0.264686	0	0	0	0	0	1	(
426	1	0	0.442393	0.640189	0.255342	1	0	0	0	0	0	(
728	1	0	0.245101	0.498067	0.663106	1	0	0	1	0	0	(
482	1	0	0.395666	0.504508	0.188475	0	1	0	0	0	0	(
111	0	0	0.345824	0.751824	0.380981	0	1	0	0	0	0	(

After minimizing the number of variables using RFE, using statsmodel to build an optimized model.

In [36]:
```
#Defining 2 functions model and VIF to train model and calculate VI
F repeatatively.
import statsmodels.api as sm

def model(X,y):
    X=sm.add_constant(X)
    lm_model=sm.OLS(y,X).fit()
    print(lm_model.summary())
    return X

from statsmodels.stats.outliers_influence import variance_inflation
_factor

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

First Model:

In [37]:
```
#Training the first model
X_train1=model(X_train_rfe,y_train)
```

```
                             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:                Thu, 28 Jan 2021   Prob (F-statistic):
8.15e-189
Time:                        22:03:09   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          0.0732      0.048      1.540      0.124      -0.020
0.167
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
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
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
Jul           -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
Best           0.2465      0.026      9.331      0.000       0.195
0.298
Neutral        0.1922      0.025      7.687      0.000       0.143
```

```
0.241
==============================================================================
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.
26.0
==============================================================================

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

In [38]:

```python
#Calculating Variance Inflation Factor
VIF(X_train1)
```

Out[38]:

	Features	VIF
0	const	143.15
14	Best	10.36
15	Neutral	8.95
6	spring	5.27
3	temp	4.42
8	winter	3.83
7	summer	2.77
4	hum	1.92
12	Nov	1.77
10	Jan	1.68
9	Dec	1.50
11	Jul	1.49
13	Sep	1.34
5	windspeed	1.21
1	yr	1.04
2	holiday	1.03

In [39]:
```
#VIF of Best > 10. But according to experience it seems people are more likely to use bikes in the best weather situations
# andhence seems significant.
#Let's drop Dec to see the difference in the significance of other variables and R squared
X_train1=X_train1.drop('Dec',axis=1)
```

Second Model:

In [40]:
```
X_train1=model(X_train1,y_train)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.844
Model:                            OLS   Adj. R-squared:                  0.839
Method:                 Least Squares   F-statistic:                     190.8
Date:                Thu, 28 Jan 2021   Prob (F-statistic):          4.41e-189
Time:                        22:03:09   Log-Likelihood:                 512.08
No. Observations:                 510   AIC:                            -994.2
Df Residuals:                     495   BIC:                            -930.6
Df Model:                          14
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0629      0.047      1.326      0.185      -0.030       0.156
yr             0.2302      0.008     28.371      0.000       0.214       0.246
holiday       -0.0920      0.026     -3.582      0.000      -0.142      -0.042
temp           0.5055      0.035     14.369      0.000       0.436       0.575
hum           -0.1697      0.038     -4.497      0.000      -0.244      -0.096
windspeed     -0.1858      0.026     -7.190      0.000      -0.237      -0.135
spring        -0.0562      0.021     -2.652      0.008      -0.098      -0.015
summer         0.0479      0.015      3.168      0.002       0.018       0.078
winter         0.0972      0.018      5.421      0.000       0.062       0.132
Jan           -0.0341      0.018     -1.936      0.053      -0.069       0.001
Jul           -0.0559      0.018     -3.057      0.002      -0.092      -0.020
Nov           -0.0236      0.017     -1.362      0.174      -0.058       0.010
Sep            0.0802      0.017      4.775      0.000       0.047       0.113
Best           0.2404      0.026      9.131      0.000       0.189       0.292
Neutral        0.1876      0.025      7.511      0.000       0.139       0.237
==============================================================================
```

```
==========
Omnibus:                       60.634   Durbin-Watson:
2.047
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
138.746
Skew:                          -0.640   Prob(JB):
7.44e-31
Kurtosis:                       5.211   Cond. No.
25.9
====================================================================
==========

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

In [41]:
```
VIF(X_train1)
```

Out[41]:

	Features	VIF
0	const	141.50
13	Best	10.23
14	Neutral	8.88
6	spring	5.20
3	temp	3.97
8	winter	3.76
7	summer	2.68
4	hum	1.90
9	Jan	1.57
10	Jul	1.49
11	Nov	1.49
12	Sep	1.31
5	windspeed	1.21
1	yr	1.04
2	holiday	1.03

In [42]:
```
#R squared remained almost the same. Variable Nov seems insignifica
nt
```

In [43]:
```
X_train1=X_train1.drop('Nov',axis=1)
```

Third model

In [44]:
```
X_train1=model(X_train1,y_train)
```

```
                            OLS Regression Results
==============================================================================
==========
Dep. Variable:                    cnt   R-squared:
0.843
Model:                            OLS   Adj. R-squared:
0.839
Method:                 Least Squares   F-statistic:
205.0
Date:                Thu, 28 Jan 2021   Prob (F-statistic):
7.59e-190
Time:                        22:03:09   Log-Likelihood:
511.13
No. Observations:                 510   AIC:
-994.3
Df Residuals:                     496   BIC:
-935.0
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
==========
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
----------
const          0.0572      0.047      1.210      0.227      -0.036
0.150
yr             0.2301      0.008     28.339      0.000       0.214
0.246
holiday       -0.0963      0.026     -3.773      0.000      -0.146
-0.046
temp           0.5124      0.035     14.706      0.000       0.444
0.581
hum           -0.1681      0.038     -4.452      0.000      -0.242
-0.094
windspeed     -0.1874      0.026     -7.253      0.000      -0.238
-0.137
spring        -0.0519      0.021     -2.476      0.014      -0.093
-0.011
summer         0.0502      0.015      3.336      0.001       0.021
0.080
winter         0.0919      0.018      5.247      0.000       0.057
0.126
Jan           -0.0333      0.018     -1.892      0.059      -0.068
0.001
Jul           -0.0556      0.018     -3.039      0.003      -0.092
-0.020
Sep            0.0827      0.017      4.951      0.000       0.050
0.116
Best           0.2392      0.026      9.084      0.000       0.187
0.291
Neutral        0.1866      0.025      7.469      0.000       0.138
0.236
==============================================================================
==========
Omnibus:                       58.633   Durbin-Watson:
```

```
2.057
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
131.919
Skew:                          -0.626   Prob(JB):
2.26e-29
Kurtosis:                       5.154   Cond. No.
25.8
====================================================================
==========

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

In [45]:
```
VIF(X_train1)
```

Out[45]:

	Features	VIF
0	const	140.41
12	Best	10.21
13	Neutral	8.87
6	spring	5.08
3	temp	3.89
8	winter	3.59
7	summer	2.65
4	hum	1.90
9	Jan	1.57
10	Jul	1.49
11	Sep	1.30
5	windspeed	1.21
1	yr	1.04
2	holiday	1.02

In [46]:
```
#R squared remained almost the same. Variable Jan seems insignifica
nt
```

In [47]:
```
X_train1=X_train1.drop('Jan',axis=1)
```

Fourth Model

In [48]:
```
X_train1=model(X_train1,y_train)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.842
Model:                            OLS   Adj. R-squared:                  0.838
Method:                 Least Squares   F-statistic:                     220.6
Date:                Thu, 28 Jan 2021   Prob (F-statistic):          2.95e-190
Time:                        22:03:09   Log-Likelihood:                 509.29
No. Observations:                 510   AIC:                            -992.6
Df Residuals:                     497   BIC:                            -937.5
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0478      0.047      1.015      0.311      -0.045       0.140
yr             0.2294      0.008     28.208      0.000       0.213       0.245
holiday       -0.0969      0.026     -3.787      0.000      -0.147      -0.047
temp           0.5299      0.034     15.728      0.000       0.464       0.596
hum           -0.1726      0.038     -4.569      0.000      -0.247      -0.098
windspeed     -0.1822      0.026     -7.074      0.000      -0.233      -0.132
spring        -0.0564      0.021     -2.700      0.007      -0.097      -0.015
summer         0.0531      0.015      3.536      0.000       0.024       0.083
winter         0.0976      0.017      5.643      0.000       0.064       0.132
Jul           -0.0572      0.018     -3.123      0.002      -0.093      -0.021
Sep            0.0833      0.017      4.973      0.000       0.050       0.116
Best           0.2369      0.026      8.983      0.000       0.185       0.289
Neutral        0.1843      0.025      7.364      0.000       0.135       0.233
==============================================================================
Omnibus:                       57.486   Durbin-Watson:                   2.051
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
```

```
130.221
Skew:                          -0.612   Prob(JB):
5.28e-29
Kurtosis:                       5.151   Cond. No.
25.7
====================================================================
==========

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

In [49]:
```
VIF(X_train1)
```

Out[49]:

	Features	VIF
0	const	138.87
11	Best	10.19
12	Neutral	8.85
6	spring	5.02
3	temp	3.61
8	winter	3.48
7	summer	2.62
4	hum	1.89
9	Jul	1.48
10	Sep	1.30
5	windspeed	1.19
1	yr	1.03
2	holiday	1.02

In [50]:
```
#All the variables seems significant now after evaluating P>|t| and VIF

#R squared from model summary is 0.842
r2=0.842

#Calculating adjusted R squared:
n = X_train1.shape[0]

# Number of features (predictors, p) is the shape along axis 1
p = X_train1.shape[1]

# We find the Adjusted R-squared using the formula

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

Out[50]:
```
0.8378588709677419
```

The variables Best has a VIF slightly greater than 10. But best case weather scenario must be kept while building the model. Hence considering the above model to be the ideal one. But let's drop a few more variables to see the changes in R squared, F-statistic and Prob (F-statistic) and if we could come up with a better model

In [51]:
```
X_train1=X_train1.drop('Best',axis=1)
```

Fifth Model

In [52]:
```
X_train1=model(X_train1,y_train)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.816
Model:                            OLS   Adj. R-squared:                  0.812
Method:                 Least Squares   F-statistic:                     201.1
Date:                Thu, 28 Jan 2021   Prob (F-statistic):          3.01e-175
Time:                        22:03:10   Log-Likelihood:                 470.93
No. Observations:                 510   AIC:                            -917.9
Df Residuals:                     498   BIC:                            -867.0
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.3419      0.037      9.366      0.000       0.270       0.414
yr             0.2299      0.009     26.257      0.000       0.213       0.247
holiday       -0.0869      0.028     -3.158      0.002      -0.141      -0.033
temp           0.5685      0.036     15.795      0.000       0.498       0.639
hum           -0.3057      0.037     -8.169      0.000      -0.379      -0.232
windspeed     -0.2292      0.027     -8.440      0.000      -0.283      -0.176
spring        -0.0430      0.022     -1.915      0.056      -0.087       0.001
summer         0.0602      0.016      3.731      0.000       0.029       0.092
winter         0.1028      0.019      5.523      0.000       0.066       0.139
Jul           -0.0631      0.020     -3.196      0.001      -0.102      -0.024
Sep            0.0815      0.018      4.519      0.000       0.046       0.117
Neutral       -0.0220      0.011     -2.062      0.040      -0.043      -0.001
==============================================================================
Omnibus:                       95.895   Durbin-Watson:                   2.033
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              249.907
Skew:                          -0.933   Prob(JB):
```

```
5.41e-55
Kurtosis:                        5.877   Cond. No.
19.2
====================================================================
==========

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

In [53]:
```
#The value of R squared decreased and value of F-statistic dropped
significantly which shows that the fourth model was more
#fit then the fifth. Still trying to drop spring to see if better m
odel can be achieved.
```

In [54]:
```
VIF(X_train1)
```

Out[54]:

	Features	VIF
0	const	71.84
6	spring	4.99
3	temp	3.56
8	winter	3.48
7	summer	2.61
4	hum	1.60
9	Jul	1.48
11	Neutral	1.39
10	Sep	1.30
5	windspeed	1.14
1	yr	1.03
2	holiday	1.01

In [55]:
```
X_train1=X_train1.drop('spring',axis=1)
```

Sixth Model

In [56]:
```
X_train1=model(X_train1,y_train)
```

```
                           OLS Regression Results
====================================================================
==========
Dep. Variable:                    cnt   R-squared:
0.815
Model:                            OLS   Adj. R-squared:
0.811
Method:                 Least Squares   F-statistic:
219.7
Date:                Thu, 28 Jan 2021   Prob (F-statistic):
1.22e-175
Time:                        22:03:10   Log-Likelihood:
469.06
No. Observations:                 510   AIC:
-916.1
Df Residuals:                     499   BIC:
-869.5
Df Model:                          10
Covariance Type:            nonrobust
====================================================================
==========
                 coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------
----------
const          0.2961      0.028     10.704      0.000       0.242
0.350
yr             0.2289      0.009     26.118      0.000       0.212
0.246
holiday       -0.0886      0.028     -3.213      0.001      -0.143
-0.034
temp           0.6198      0.024     25.761      0.000       0.573
0.667
hum           -0.3124      0.037     -8.361      0.000      -0.386
-0.239
windspeed     -0.2340      0.027     -8.631      0.000      -0.287
-0.181
summer         0.0819      0.012      7.096      0.000       0.059
0.105
winter         0.1312      0.011     11.562      0.000       0.109
0.153
Jul           -0.0558      0.019     -2.872      0.004      -0.094
-0.018
Sep            0.0914      0.017      5.279      0.000       0.057
0.125
Neutral       -0.0207      0.011     -1.933      0.054      -0.042
0.000
====================================================================
==========
Omnibus:                       93.118   Durbin-Watson:
2.045
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
227.239
Skew:                          -0.931   Prob(JB):
4.53e-50
Kurtosis:                       5.689   Cond. No.
```

```
16.4
==============================================================================

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

In [57]:
```
#The value of R squared remained the same but the value of F-statistic has increased and almost similar to our ideal fourth model.
#This can be our another ideal model.
#Even though Neutral is one more vital variable for our ideal fourth model, its p-value is higher.
#Let's try to drop that variable.
```

In [58]:
```
VIF(X_train1)
```

Out[58]:

	Features	VIF
0	const	41.04
4	hum	1.59
3	temp	1.58
8	Jul	1.43
10	Neutral	1.39
6	summer	1.33
7	winter	1.28
9	Sep	1.19
5	windspeed	1.13
1	yr	1.03
2	holiday	1.01

In [59]:
```
X_train1=X_train1.drop('Neutral',axis=1)
```

Seventh Model

In [60]:
```
X_train1=model(X_train1,y_train)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.814
Model:                            OLS   Adj. R-squared:                  0.810
Method:                 Least Squares   F-statistic:                     242.4
Date:                Thu, 28 Jan 2021   Prob (F-statistic):          4.86e-176
Time:                        22:03:10   Log-Likelihood:                 467.16
No. Observations:                 510   AIC:                            -914.3
Df Residuals:                     500   BIC:                            -872.0
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.3098      0.027     11.552      0.000       0.257       0.362
yr             0.2278      0.009     25.978      0.000       0.211       0.245
holiday       -0.0868      0.028     -3.139      0.002      -0.141      -0.032
temp           0.6283      0.024     26.480      0.000       0.582       0.675
hum           -0.3492      0.032    -10.838      0.000      -0.412      -0.286
windspeed     -0.2380      0.027     -8.778      0.000      -0.291      -0.185
summer         0.0812      0.012      7.016      0.000       0.058       0.104
winter         0.1334      0.011     11.788      0.000       0.111       0.156
Jul           -0.0553      0.019     -2.841      0.005      -0.094      -0.017
Sep            0.0910      0.017      5.240      0.000       0.057       0.125
==============================================================================
Omnibus:                       87.662   Durbin-Watson:                   2.031
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              196.855
Skew:                          -0.909   Prob(JB):                     1.79e-43
Kurtosis:                       5.441   Cond. No.                         14.5
==============================================================================
```

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

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

In [61]:
```
VIF(X_train1)
```

Out[61]:

	Features	VIF
0	const	38.36
3	temp	1.53
8	Jul	1.43
6	summer	1.33
7	winter	1.27
9	Sep	1.19
4	hum	1.18
5	windspeed	1.13
1	yr	1.03
2	holiday	1.01

This model again has lesser R squared than the fourth model but the F-statistic is much more than that.

There are 2 models that can be considered as the best fits:

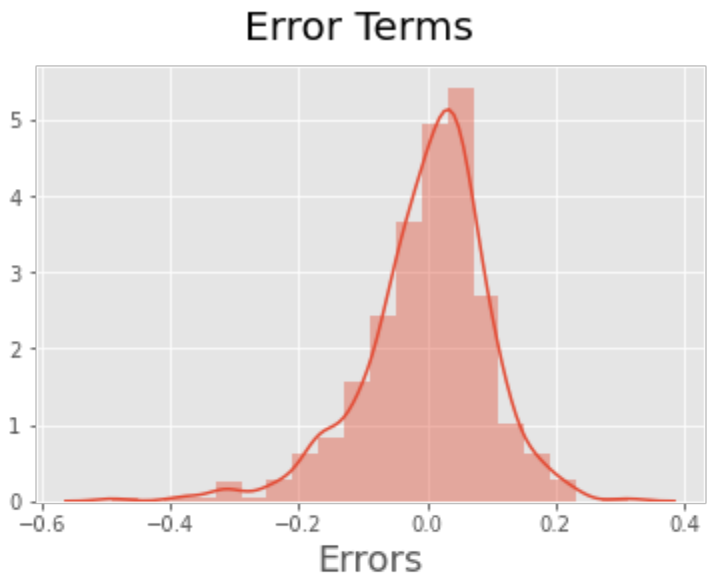
Fourth model and the Seventh model

5. Residual Analysis of the trained data

In [62]:
```
#Rebuilding the seventh model
lm_model7=sm.OLS(y_train,X_train1).fit()
y_train_pred7=lm_model7.predict(X_train1)
```

In [63]:
```
%matplotlib inline
fig = plt.figure()
sns.distplot((y_train - y_train_pred7), bins = 20)
fig.suptitle('Error Terms', fontsize = 20)                  # Plot
heading
plt.xlabel('Errors', fontsize = 18)
```

Out[63]: Text(0.5, 0, 'Errors')



In [64]:
```
#Rebuilding the fourth model
X_train_rfe=sm.add_constant(X_train_rfe)
X_train_rfe.head()
```

Out[64]:

	const	yr	holiday	temp	hum	windspeed	spring	summer	winter	Dec	Jan
576	1.0	1	0	0.815169	0.725633	0.264686	0	0	0	0	0
426	1.0	1	0	0.442393	0.640189	0.255342	1	0	0	0	0
728	1.0	1	0	0.245101	0.498067	0.663106	1	0	0	1	0
482	1.0	1	0	0.395666	0.504508	0.188475	0	1	0	0	0
111	1.0	0	0	0.345824	0.751824	0.380981	0	1	0	0	0

In [65]:
```
X_train_rfe.drop(['Dec','Nov','Jan'], axis=1, inplace=True)
X_train_rfe.head()
```

Out[65]:

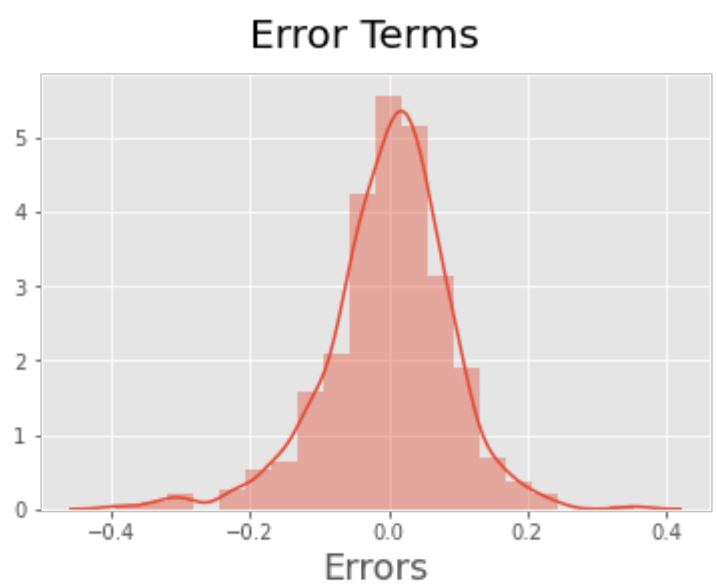
	const	yr	holiday	temp	hum	windspeed	spring	summer	winter	Jul	Sep	B
576	1.0	1	0	0.815169	0.725633	0.264686	0	0	0	1	0	
426	1.0	1	0	0.442393	0.640189	0.255342	1	0	0	0	0	
728	1.0	1	0	0.245101	0.498067	0.663106	1	0	0	0	0	
482	1.0	1	0	0.395666	0.504508	0.188475	0	1	0	0	0	
111	1.0	0	0	0.345824	0.751824	0.380981	0	1	0	0	0	

Columns of X_train_rfe are similar to the fourth model

In [66]:
```
lm_model4=sm.OLS(y_train,X_train_rfe).fit()
y_train_pred4=lm_model4.predict(X_train_rfe)
```

In [67]:
```
fig = plt.figure()
sns.distplot((y_train - y_train_pred4), bins = 20)
fig.suptitle('Error Terms', fontsize = 20)                  # Plot heading
plt.xlabel('Errors', fontsize = 18)
```

Out[67]: Text(0.5, 0, 'Errors')



Residual Analysis shows that error terms for both the models gives almost a normal distribution but the R squared value is better for the fourth model compared to the seventh model. Also normality of error distribution is slightly better for fourth model compared to seventh model.

Hence selecting the fourth model for prediction.

6. Making Predictions

6.1 Preparing data for prediction.

In [68]:
```
df_test.head()
```

Out[68]:

	yr	day	holiday	workingday	temp	hum	windspeed	cnt	spring	summer	...
184	0	7	1	0	29.793347	63.7917	5.459106	6043	0	0	...
535	1	20	0	1	32.082500	59.2083	7.625404	6211	0	1	...
299	0	27	0	0	19.270000	81.2917	13.250121	2659	0	0	...
221	0	8	0	1	31.433347	42.4167	13.417286	4780	0	0	...
152	0	6	0	0	29.315000	30.5000	19.583229	4968	0	1	...

5 rows × 30 columns

In [69]:
```
#rescaling columns from the list need_rescale=['temp','hum','windsp
eed','cnt']

df_test[need_rescale]=scaler.transform(df_test[need_rescale])

df_train.head()
```

Out[69]:

	yr	day	holiday	workingday	temp	hum	windspeed	spring	summer	winter	.
576	1	31	0	1	0.815169	0.725633	0.264686	0	0	0	.
426	1	3	0	0	0.442393	0.640189	0.255342	1	0	0	.
728	1	30	0	1	0.245101	0.498067	0.663106	1	0	0	.
482	1	28	0	0	0.395666	0.504508	0.188475	0	1	0	.
111	0	22	0	0	0.345824	0.751824	0.380981	0	1	0	.

5 rows × 29 columns

In [70]:
```
df_test.describe()
```

Out[70]:

	yr	day	holiday	workingday	temp	hum	windspeed
count	220.000000	220.000000	220.000000	220.000000	220.000000	220.000000	220.000000
mean	0.481818	15.927273	0.036364	0.640909	0.558718	0.638221	0.313293
std	0.500809	8.700715	0.187620	0.480828	0.233187	0.148694	0.159584
min	0.000000	1.000000	0.000000	0.000000	0.046591	0.261915	-0.042808
25%	0.000000	9.000000	0.000000	0.000000	0.355429	0.529197	0.198843
50%	0.000000	15.500000	0.000000	1.000000	0.558172	0.625590	0.300126
75%	1.000000	24.000000	0.000000	1.000000	0.755981	0.743798	0.402718
max	1.000000	31.000000	1.000000	1.000000	0.984424	1.002146	0.807474

8 rows × 30 columns

6.2 Prediction with model 4

In [71]:
```
y_test=df_test.pop('cnt')
X_train_rfe=X_train_rfe.drop('const',axis=1)
X_test_model4=df_test[X_train_rfe.columns]
X_test_model4.head()
```

Out[71]:

	yr	holiday	temp	hum	windspeed	spring	summer	winter	Jul	Sep	Best	Ne
184	0	1	0.831783	0.657364	0.084219	0	0	0	1	0	0	
535	1	0	0.901354	0.610133	0.153728	0	1	0	0	0	1	
299	0	0	0.511964	0.837699	0.334206	0	0	1	0	0	0	
221	0	0	0.881625	0.437098	0.339570	0	0	0	0	0	1	
152	0	0	0.817246	0.314298	0.537414	0	1	0	0	0	1	

In [72]:
```
X_test_model4.shape
```

Out[72]: (220, 12)

In [73]:
```
#Adding constant to dataframe
X_test_model4=sm.add_constant(X_test_model4)
```

In [74]:
```
#Prediction
y_test_pred_model4=lm_model4.predict(X_test_model4)
```

In [75]:
```
#Calculating Test data R-squared:
from sklearn.metrics import r2_score
r2=r2_score(y_test, y_test_pred_model4)
print(r2)
```

```
0.8151738700604121
```

In [76]:
```
#Calculating adjusted R squared:
n = X_test_model4.shape[0]

# Number of features (predictors, p) is the shape along axis 1
p = X_test_model4.shape[1]

# Calculating Adjusted R-squared using the formula

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

Out[76]:
```
0.8035100851613118
```

Train R squared: 0.842

Train Adjusted R squared: 0.8378588709677419

Test R squared: 0.8378588709677419

Test Adjusted R squared: 0.8035100851613118

In [77]:
```
#Checking Homoscedasticity for train and test data
plt.figure(figsize=(12,8))
plt.subplot(1,2,1)
sns.scatterplot(y=y_train - y_train_pred4, x=y_train_pred4)
plt.subplot(1,2,2)
sns.scatterplot(y=y_test - y_test_pred_model4, x=y_test_pred_model
4)
plt.show()
```

There is no clustering or pattern below or above 0.0 on the Y-axis. This model is giving best results compared to other 6 models formed earlier while training.

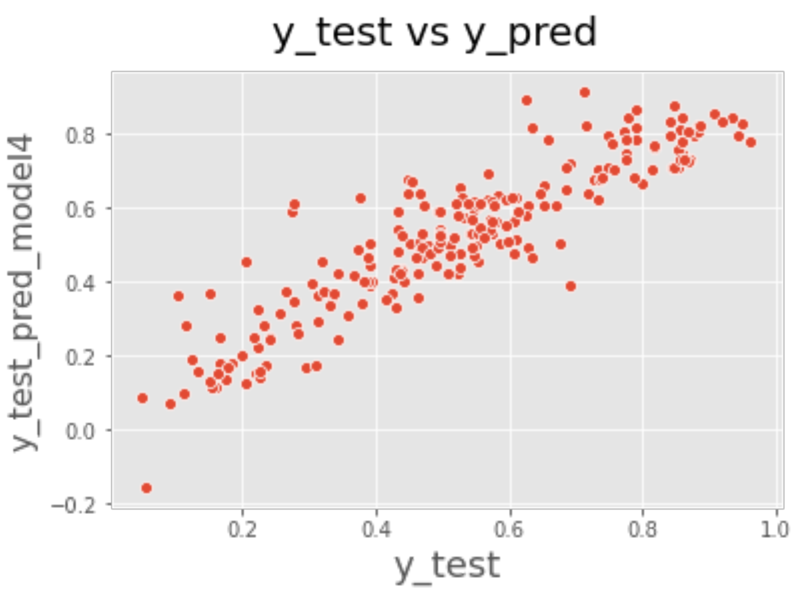
In [78]:
```
# Evaluating the Algorithm
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y
_test_pred_model4))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_t
est_pred_model4))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_erro
r(y_test, y_test_pred_model4)))
```

```
Mean Absolute Error: 0.0695347736271711
Mean Squared Error: 0.008837328237214432
Root Mean Squared Error: 0.094007064826078
```

Lower values of MAE, MSE and RMSE shows vouches for the good performance of the model.

In [79]:
```
# understanding the spread.
fig = plt.figure()
sns.scatterplot(y_test,y_test_pred_model4)
fig.suptitle('y_test vs y_pred', fontsize=20)              # Plot h
eading
plt.xlabel('y_test', fontsize=18)                          # X-labe
l
plt.ylabel('y_test_pred_model4', fontsize=16)
```

Out[79]:
```
Text(0, 0.5, 'y_test_pred_model4')
```



Based on the very close value of R squared and Adjusted R squared values of the train and test data sets and based on y_test and y_pred graph, it can be infereed that the our linear regression model has the below equation for it's best fitted line:

cnt= 0.0478 + 0.2294 yr -0.0969 holiday + 0.5299 temp -0.1726 hum -0.1822 windspeed -0.0564 spring + 0.0531 summer+ 0.0976 winter -0.0572 Jul + 0.0833 sept + 0.2369 (Clear, Few clouds, Partly cloudy, Partly cloudy) + 0.1843 (Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist)

In [80]:
```
print(lm_model4.summary())
```

```
                            OLS Regression Results
======================================================================
==========
Dep. Variable:                    cnt   R-squared:
0.842
Model:                            OLS   Adj. R-squared:
0.838
Method:                 Least Squares   F-statistic:
220.6
Date:                Thu, 28 Jan 2021   Prob (F-statistic):
2.95e-190
Time:                        22:03:12   Log-Likelihood:
509.29
No. Observations:                 510   AIC:
-992.6
Df Residuals:                     497   BIC:
-937.5
Df Model:                          12
Covariance Type:            nonrobust
======================================================================
==========
                 coef    std err          t      P>|t|      [0.025
0.975]
----------------------------------------------------------------------
----------
const          0.0478      0.047      1.015      0.311      -0.045
0.140
yr             0.2294      0.008     28.208      0.000       0.213
0.245
holiday       -0.0969      0.026     -3.787      0.000      -0.147
-0.047
temp           0.5299      0.034     15.728      0.000       0.464
0.596
hum           -0.1726      0.038     -4.569      0.000      -0.247
-0.098
windspeed     -0.1822      0.026     -7.074      0.000      -0.233
-0.132
spring        -0.0564      0.021     -2.700      0.007      -0.097
-0.015
summer         0.0531      0.015      3.536      0.000       0.024
0.083
winter         0.0976      0.017      5.643      0.000       0.064
0.132
Jul           -0.0572      0.018     -3.123      0.002      -0.093
-0.021
Sep            0.0833      0.017      4.973      0.000       0.050
0.116
Best           0.2369      0.026      8.983      0.000       0.185
0.289
Neutral        0.1843      0.025      7.364      0.000       0.135
0.233
======================================================================
==========
Omnibus:                       57.486   Durbin-Watson:
2.051
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
```

```
130.221
Skew:                          -0.612   Prob(JB):
5.28e-29
Kurtosis:                       5.151   Cond. No.
25.7
====================================================================
==========

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

Hypothesis Testing

Null hypothesis states that there is no relationship between the X variables and the Y variables meaning the coefficients of the independent variables is zero. From the final model summary, it is evident that all our coefficients are not equal to zero which means We REJECT the NULL HYPOTHESIS

The company should focus on the following factors:

1. People are less likely to use their service at low or extreme temperatures. So either the company can function to half the capacity or minimum capacity to reduce operational costs for better profits and provide service for regular registered customers mostly. Similarly in days with increase in humidity and windspeed. Discounts or offers won't help as well since it's inconvenient to commute using bikes in such situations.
2. There will be increase in the number of users with increase in year since people will start adapting to renting bikes more often. There might be chances that because of covid just been around the corner, the trend might not follow immediately but giving a year more will definitely see rise in number of users.
3. People are more likely to use their service in the best or the neutral weather environments i.e;Clear, Few clouds, Partly cloudy, Partly cloudy OR Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist.