practice project

July 12, 2022

[1]:
```
#load the file using pandas
#import all required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
data=pd.read_csv('hour.csv')
data
```

[2]:
```
       instant      dteday  season  yr  mnth  hr  holiday  weekday  \
0            1  2011-01-01       1   0     1   0        0        6
1            2  2011-01-01       1   0     1   1        0        6
2            3  2011-01-01       1   0     1   2        0        6
3            4  2011-01-01       1   0     1   3        0        6
4            5  2011-01-01       1   0     1   4        0        6
…        …         …     …  ..   …  ..      …      …
17374    17375  2012-12-31       1   1    12  19        0        1
17375    17376  2012-12-31       1   1    12  20        0        1
17376    17377  2012-12-31       1   1    12  21        0        1
17377    17378  2012-12-31       1   1    12  22        0        1
17378    17379  2012-12-31       1   1    12  23        0        1

       workingday  weathersit  temp   atemp   hum  windspeed  casual  \
0               0           1  0.24  0.2879  0.81     0.0000       3
1               0           1  0.22  0.2727  0.80     0.0000       8
2               0           1  0.22  0.2727  0.80     0.0000       5
3               0           1  0.24  0.2879  0.75     0.0000       3
4               0           1  0.24  0.2879  0.75     0.0000       0
…           …         …   …     …   …        …     …
17374           1           2  0.26  0.2576  0.60     0.1642      11
17375           1           2  0.26  0.2576  0.60     0.1642       8
17376           1           1  0.26  0.2576  0.60     0.1642       7
17377           1           1  0.26  0.2727  0.56     0.1343      13
17378           1           1  0.26  0.2727  0.65     0.1343      12

       registered  cnt
```

```
0              13   16
1              32   40
2              27   32
3              10   13
4               1    1
…           …  …
17374         108  119
17375          81   89
17376          83   90
17377          48   61
17378          37   49

[17379 rows x 17 columns]
```

[3]:
```
data.head()
```

[3]:
```
   instant      dteday  season  yr  mnth  hr  holiday  weekday  workingday  \
0        1  2011-01-01       1   0     1   0        0        6           0
1        2  2011-01-01       1   0     1   1        0        6           0
2        3  2011-01-01       1   0     1   2        0        6           0
3        4  2011-01-01       1   0     1   3        0        6           0
4        5  2011-01-01       1   0     1   4        0        6           0

   weathersit  temp   atemp   hum  windspeed  casual  registered  cnt
0           1  0.24  0.2879  0.81        0.0       3          13   16
1           1  0.22  0.2727  0.80        0.0       8          32   40
2           1  0.22  0.2727  0.80        0.0       5          27   32
3           1  0.24  0.2879  0.75        0.0       3          10   13
4           1  0.24  0.2879  0.75        0.0       0           1    1
```

[5]:
```
#shape
data.shape
```

[5]:
```
(17379, 17)
```

[6]:
```
#null values
data.isnull()
```

[6]:
```
       instant  dteday  season     yr   mnth     hr  holiday  weekday  \
0        False   False   False  False  False  False    False    False
1        False   False   False  False  False  False    False    False
2        False   False   False  False  False  False    False    False
3        False   False   False  False  False  False    False    False
4        False   False   False  False  False  False    False    False
…        …     …     …    …    …    …      …      …
17374    False   False   False  False  False  False    False    False
17375    False   False   False  False  False  False    False    False
```

```
17376    False  False  False  False  False  False    False    False
17377    False  False  False  False  False  False    False    False
17378    False  False  False  False  False  False    False    False

       workingday  weathersit   temp  atemp    hum  windspeed  casual  \
0           False       False  False  False  False      False   False
1           False       False  False  False  False      False   False
2           False       False  False  False  False      False   False
3           False       False  False  False  False      False   False
4           False       False  False  False  False      False   False
…           …         …    …    …    …        …     …
17374       False       False  False  False  False      False   False
17375       False       False  False  False  False      False   False
17376       False       False  False  False  False      False   False
17377       False       False  False  False  False      False   False
17378       False       False  False  False  False      False   False

       registered    cnt
0           False  False
1           False  False
2           False  False
3           False  False
4           False  False
…           …    …
17374       False  False
17375       False  False
17376       False  False
17377       False  False
17378       False  False

[17379 rows x 17 columns]
```

[7]:

```
data.isnull().sum()
```

[7]:

```
instant       0
dteday        0
season        0
yr            0
mnth          0
hr            0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
```

```
windspeed     0
casual        0
registered    0
cnt           0
dtype: int64
```

[8]:

```
#summary
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17379 entries, 0 to 17378
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   instant     17379 non-null  int64
 1   dteday      17379 non-null  object
 2   season      17379 non-null  int64
 3   yr          17379 non-null  int64
 4   mnth        17379 non-null  int64
 5   hr          17379 non-null  int64
 6   holiday     17379 non-null  int64
 7   weekday     17379 non-null  int64
 8   workingday  17379 non-null  int64
 9   weathersit  17379 non-null  int64
 10  temp        17379 non-null  float64
 11  atemp       17379 non-null  float64
 12  hum         17379 non-null  float64
 13  windspeed   17379 non-null  float64
 14  casual      17379 non-null  int64
 15  registered  17379 non-null  int64
 16  cnt         17379 non-null  int64
dtypes: float64(4), int64(12), object(1)
memory usage: 2.3+ MB
```

[9]:

```
data.describe()
```

[9]:

```
          instant        season            yr          mnth            hr  \
count  17379.0000  17379.000000  17379.000000  17379.000000  17379.000000
mean    8690.0000      2.501640      0.502561      6.537775     11.546752
std     5017.0295      1.106918      0.500008      3.438776      6.914405
min        1.0000      1.000000      0.000000      1.000000      0.000000
25%     4345.5000      2.000000      0.000000      4.000000      6.000000
50%     8690.0000      3.000000      1.000000      7.000000     12.000000
75%    13034.5000      3.000000      1.000000     10.000000     18.000000
max    17379.0000      4.000000      1.000000     12.000000     23.000000

            holiday       weekday    workingday    weathersit          temp  \
count  17379.000000  17379.000000  17379.000000  17379.000000  17379.000000
mean       0.028770      3.003683      0.682721      1.425283      0.496987
std        0.167165      2.005771      0.465431      0.639357      0.192556
min        0.000000      0.000000      0.000000      1.000000      0.020000
25%        0.000000      1.000000      0.000000      1.000000      0.340000
50%        0.000000      3.000000      1.000000      1.000000      0.500000
75%        0.000000      5.000000      1.000000      2.000000      0.660000
max        1.000000      6.000000      1.000000      4.000000      1.000000

              atemp           hum     windspeed        casual    registered  \
count  17379.000000  17379.000000  17379.000000  17379.000000  17379.000000
mean       0.475775      0.627229      0.190098     35.676218    153.786869
std        0.171850      0.192930      0.122340     49.305030    151.357286
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        0.333300      0.480000      0.104500      4.000000     34.000000
50%        0.484800      0.630000      0.194000     17.000000    115.000000
75%        0.621200      0.780000      0.253700     48.000000    220.000000
max        1.000000      1.000000      0.850700    367.000000    886.000000

                cnt
count  17379.000000
mean     189.463088
std      181.387599
min        1.000000
25%       40.000000
50%      142.000000
75%      281.000000
max      977.000000
```

[10]:
```
#check duplicate data
duplicate=data[data.duplicated()]
duplicate
```

[10]:
```
Empty DataFrame
Columns: [instant, dteday, season, yr, mnth, hr, holiday, weekday, workingday,
weathersit, temp, atemp, hum, windspeed, casual, registered, cnt]
Index: []
```

[11]:
```
#Sanity checks
#Check if registered + casual = cnt for all the records. The two must add to␣
 ↪cnt,
#if not the row is junk and should be dropped.
data['registered']+data['casual']!=data['cnt']
```

[11]:
```
0        False
1        False
2        False
```

```
3        False
4        False
         …
17374    False
17375    False
17376    False
17377    False
17378    False
Length: 17379, dtype: bool
```

[3]:
```
#Sum
np.sum(data['registered']+data['casual']!=data['cnt'])
```

[3]:
```
0
```

[4]:
```
#suppose reg+casual!=cnt, then the code will be
data.drop(data[data['registered']+data['casual']!=data['cnt']].
 ↪index,inplace=True)
```

[16]:
```
#Month values should be 1-12 only
data['mnth'].unique()
```

[16]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

[17]:
```
#Hour should be 0-23
data['hr'].unique()
```

[17]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23])
```

[3]:
```
#Variables 'casual', 'registered' are redundant and need to be dropped.
#'Instant' is the index, and needs to be dropped too.
#The date column dteday will not be used in the model building, and hence needs␣
 ↪to be dropped.
#Create new dataframe named 'inp1'.
list=['casual','registered','dteday','instant']
inp1=data.drop(list,axis=1).copy()
```

[4]:
```
inp1
```

[4]:
```
       season  yr  mnth  hr  holiday  weekday  workingday  weathersit  temp  \
0           1   0     1   0        0        6           0           1  0.24
1           1   0     1   1        0        6           0           1  0.22
2           1   0     1   2        0        6           0           1  0.22
3           1   0     1   3        0        6           0           1  0.24
4           1   0     1   4        0        6           0           1  0.24
…          … ..     …  ..        …        …           …           …     …
```

```
17374        1    1    12   19         0         1            1            2   0.26
17375        1    1    12   20         0         1            1            2   0.26
17376        1    1    12   21         0         1            1            1   0.26
17377        1    1    12   22         0         1            1            1   0.26
17378        1    1    12   23         0         1            1            1   0.26

        atemp   hum  windspeed  cnt
0      0.2879  0.81     0.0000   16
1      0.2727  0.80     0.0000   40
2      0.2727  0.80     0.0000   32
3      0.2879  0.75     0.0000   13
4      0.2879  0.75     0.0000    1
…         …    …           …   …
17374  0.2576  0.60     0.1642  119
17375  0.2576  0.60     0.1642   89
17376  0.2576  0.60     0.1642   90
17377  0.2727  0.56     0.1343   61
17378  0.2727  0.65     0.1343   49

[17379 rows x 13 columns]
```

[21]:
```
inp1.shape
```

[21]:
```
(17379, 13)
```

[22]:
```
#Univariate analysis -
#Describe the numerical fields in the dataset using pandas describe method
inp1.describe()
```

[22]:
```
             season            yr          mnth            hr       holiday  \
count  17379.000000  17379.000000  17379.000000  17379.000000  17379.000000
mean       2.501640      0.502561      6.537775     11.546752      0.028770
std        1.106918      0.500008      3.438776      6.914405      0.167165
min        1.000000      0.000000      1.000000      0.000000      0.000000
25%        2.000000      0.000000      4.000000      6.000000      0.000000
50%        3.000000      1.000000      7.000000     12.000000      0.000000
75%        3.000000      1.000000     10.000000     18.000000      0.000000
max        4.000000      1.000000     12.000000     23.000000      1.000000

            weekday    workingday    weathersit          temp         atemp  \
count  17379.000000  17379.000000  17379.000000  17379.000000  17379.000000
mean       3.003683      0.682721      1.425283      0.496987      0.475775
std        2.005771      0.465431      0.639357      0.192556      0.171850
min        0.000000      0.000000      1.000000      0.020000      0.000000
25%        1.000000      0.000000      1.000000      0.340000      0.333300
50%        3.000000      1.000000      1.000000      0.500000      0.484800
75%        5.000000      1.000000      2.000000      0.660000      0.621200
```

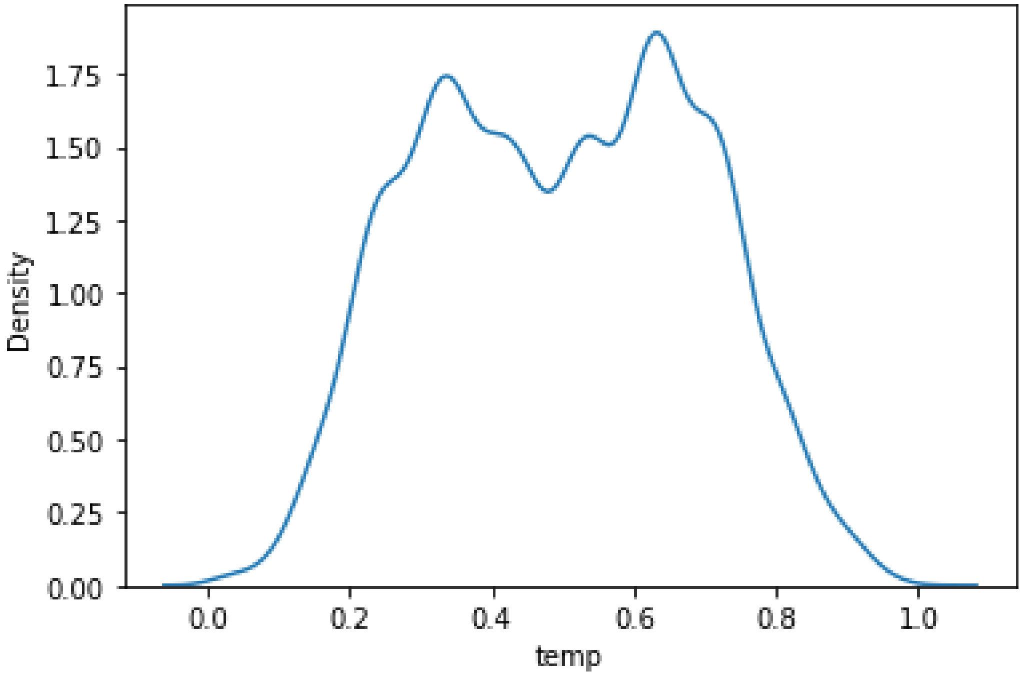
```
max        6.000000      1.000000      4.000000      1.000000      1.000000

                 hum     windspeed           cnt
count  17379.000000  17379.000000  17379.000000
mean       0.627229      0.190098    189.463088
std        0.192930      0.122340    181.387599
min        0.000000      0.000000      1.000000
25%        0.480000      0.104500     40.000000
50%        0.630000      0.194000    142.000000
75%        0.780000      0.253700    281.000000
max        1.000000      0.850700    977.000000
```

[23]:

```
#Make density plot for temp.
#This would give a sense of the centrality and the spread of the distribution.
sns.kdeplot(data['temp'])
```

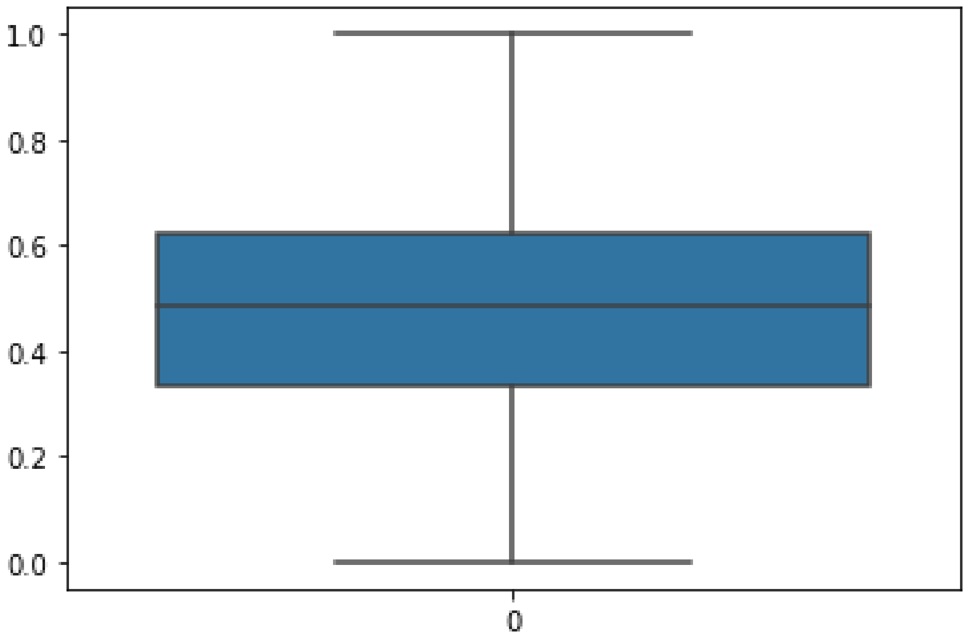
[23]:

```
<AxesSubplot:xlabel='temp', ylabel='Density'>
```



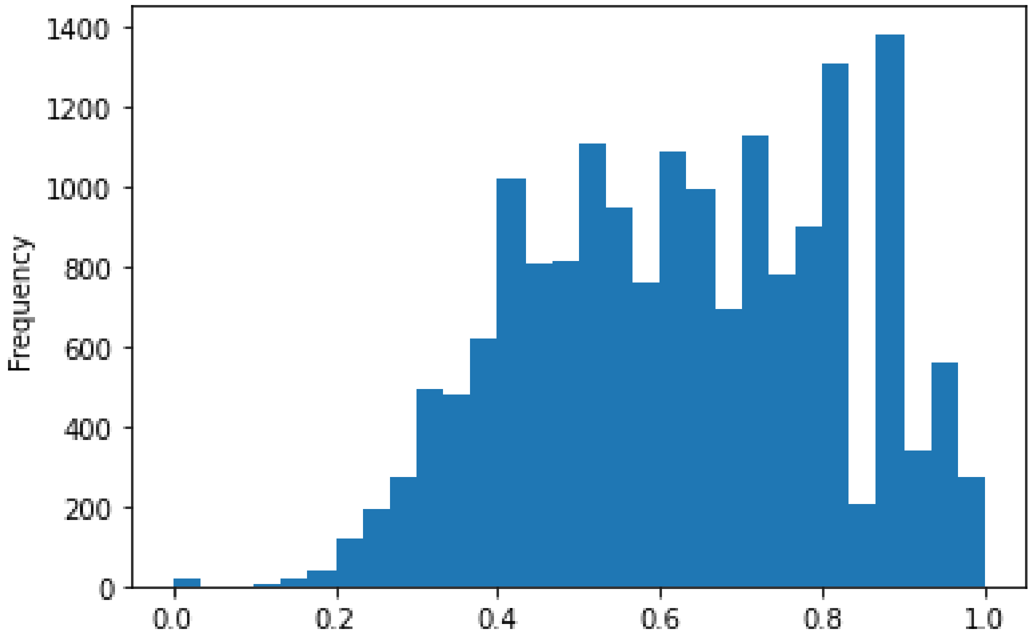
[29]:

```
#Boxplot for atemp.
#Are there any outliers?
sns.boxplot(data=inp1.atemp)
plt.show()
#there are no outliers
```



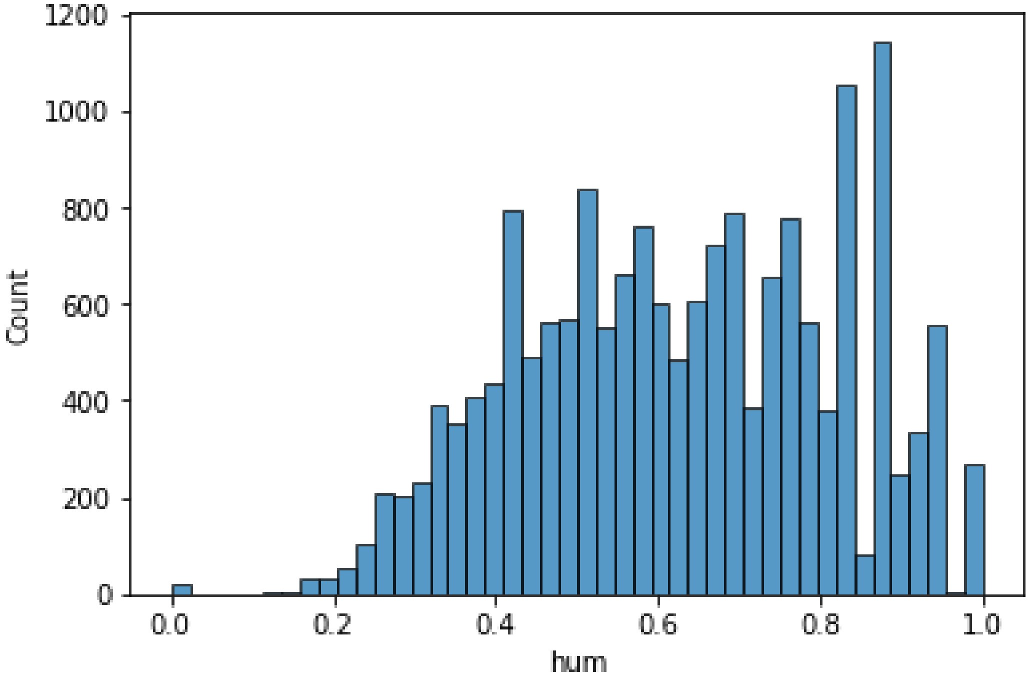
[14]:

```
#Histogram for hum
#Do you detect any abnormally high values?
inp1.hum.plot.hist(bins=30)#Gives clear value if no.of bins increased
plt.show()
```



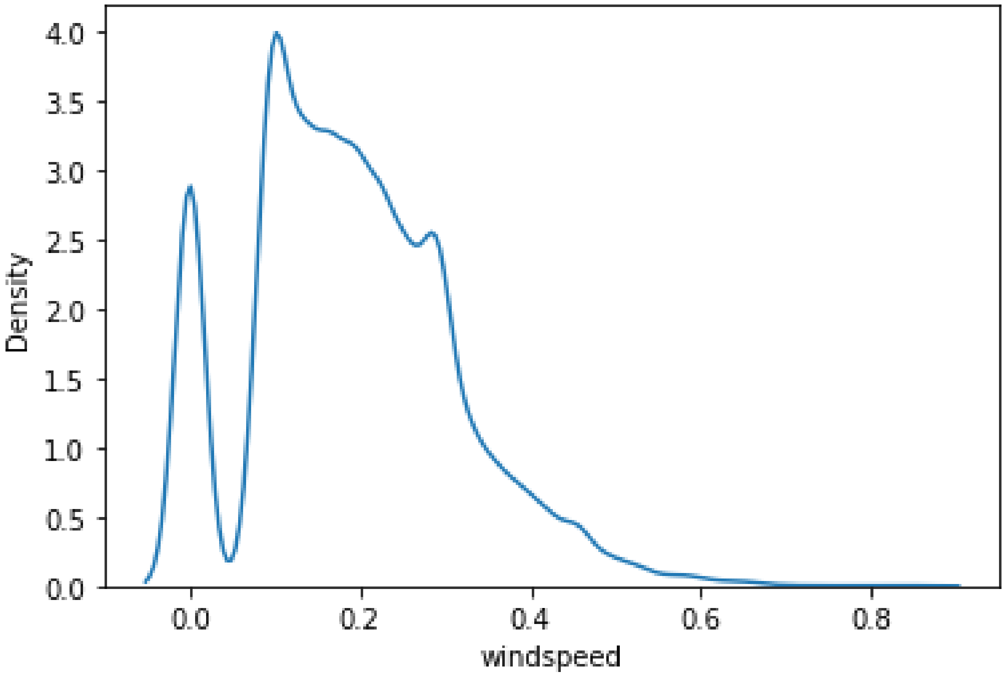
```
[13]: sns.histplot(data=inp1.hum) #better plot is produced using seaborn
```

```
[13]: <AxesSubplot:xlabel='hum', ylabel='Count'>
```



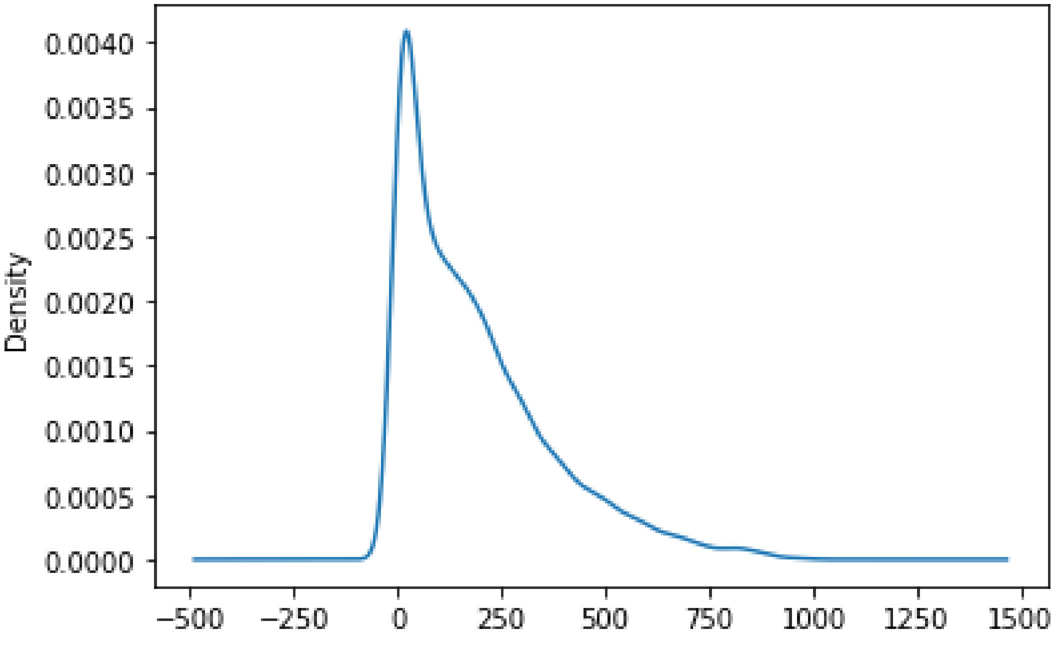
```
[15]: #Density plot for windspeed
      sns.kdeplot(data['windspeed'])
```

```
[15]: <AxesSubplot:xlabel='windspeed', ylabel='Density'>
```



[20]:
```
#Box and density plot for cnt - this is the variable of interest.
#Do you see any outliers in the boxplot?
#Does the density plot provide a similar insight?
inp1.cnt.plot.density()
```

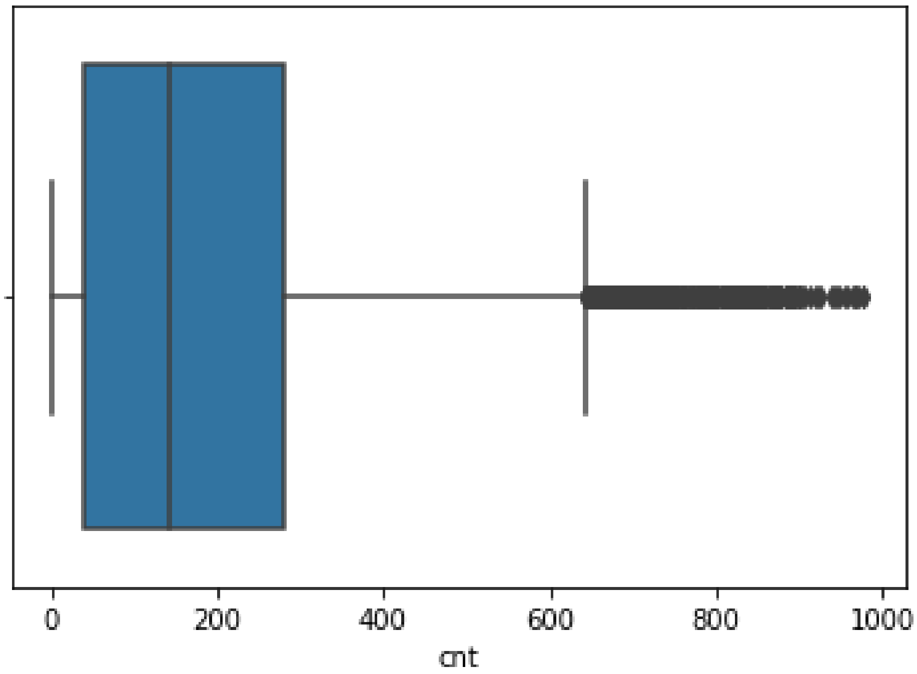
[20]:
```
<AxesSubplot:ylabel='Density'>
```



[21]:

```
sns.boxplot(inp1.cnt)
plt.show()
#we have a lot of outliers present at the higher side
```

```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



[22]:

```
#Outlier treatment -
#Cnt - looks like some hours have rather high values of cnt. We’ll need to␣
↪treat these outliers so that they don’t skew our analysis and our model.
#Find out the following percentiles - 10, 25, 50, 75, 90, 95, 99
#Decide the cutoff percentile and drop records with values higher that the␣
↪cutoff.
#Name the new dataframe ‘inp2’.
inp1.cnt.quantile([0.1,0.25,0.5,0.75,0.90,0.95,0.99])
```

[22]:

```
0.10      9.00
0.25     40.00
0.50    142.00
```

```
0.75    281.00
0.90    451.20
0.95    563.10
0.99    782.22
Name: cnt, dtype: float64
```

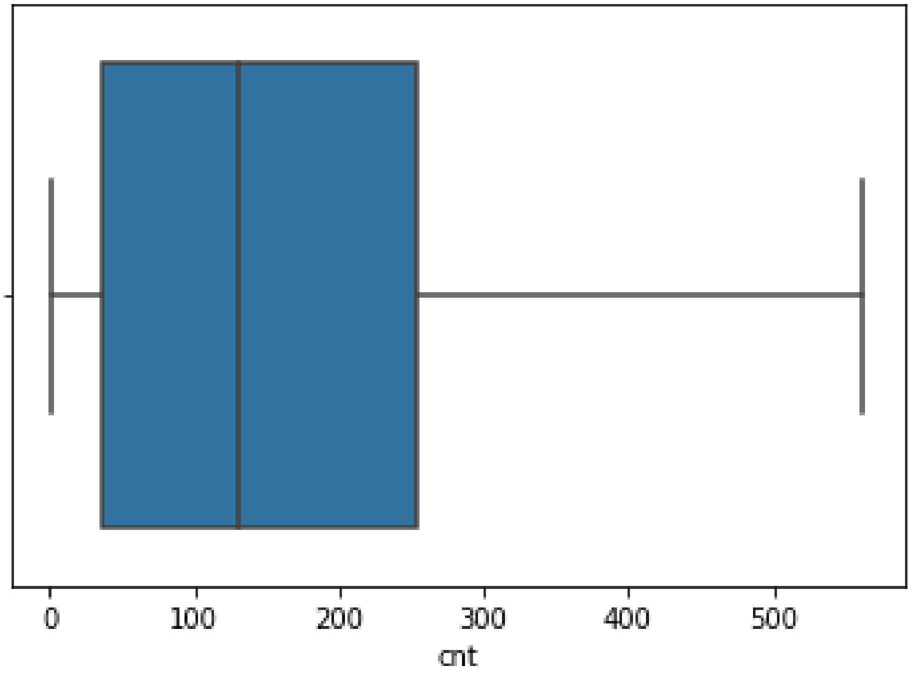
[5]:

```
#Decide the cutoff percentile and drop records with values higher that the␣
 ↪cutoff.
#Name the new dataframe ‘inp2’.
#The cut off is decided at 95 percentile
inp2=inp1[inp1.cnt<563].copy()
```

[26]:

```
sns.boxplot(inp2.cnt)
plt.show()
```

```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



[31]:

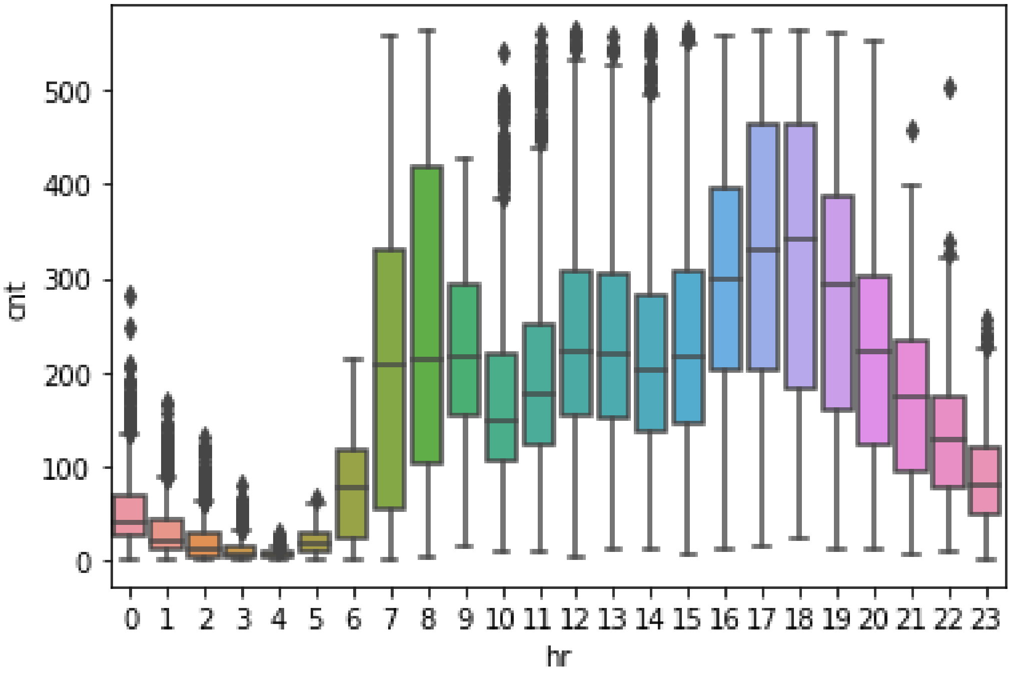
```
#Bi-variate analysis
#Make box plot for cnt vs hr
#What kind of pattern do you see?
sns.boxplot('hr','cnt',data=inp2)
```

```
plt.figure(figsize=[12,6])
```

```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

[31]:

```
<Figure size 864x432 with 0 Axes>
```



```
<Figure size 864x432 with 0 Axes>
```

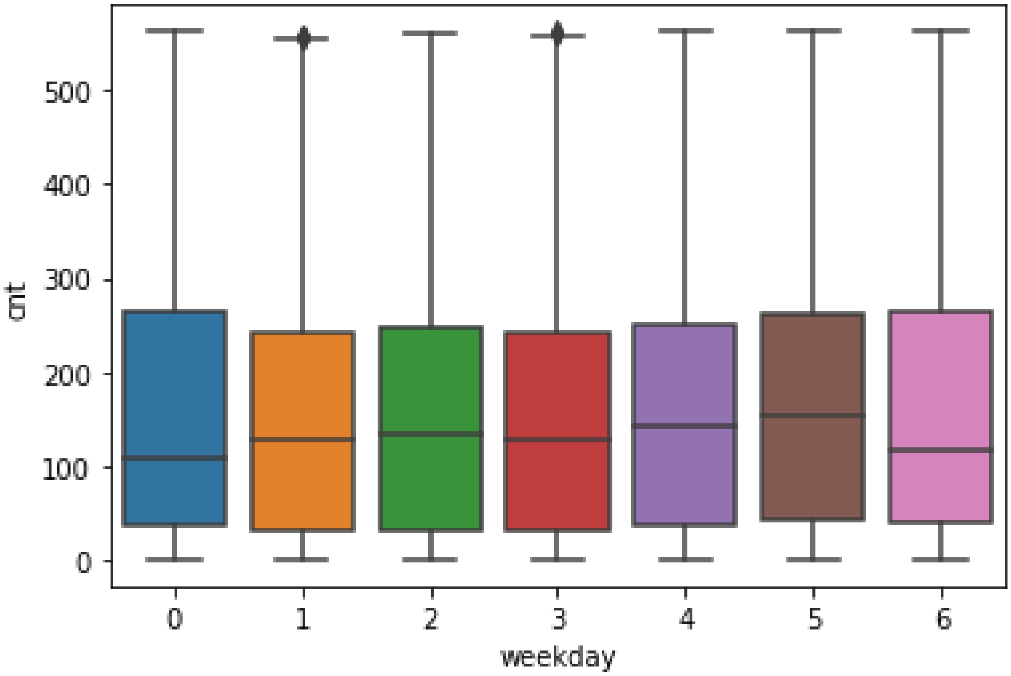
[32]:

```
#Make boxplot for cnt vs weekday
sns.boxplot('weekday','cnt',data=inp2)
plt.figure(figsize=[12,6])
```

```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

[32]:

```
<Figure size 864x432 with 0 Axes>
```

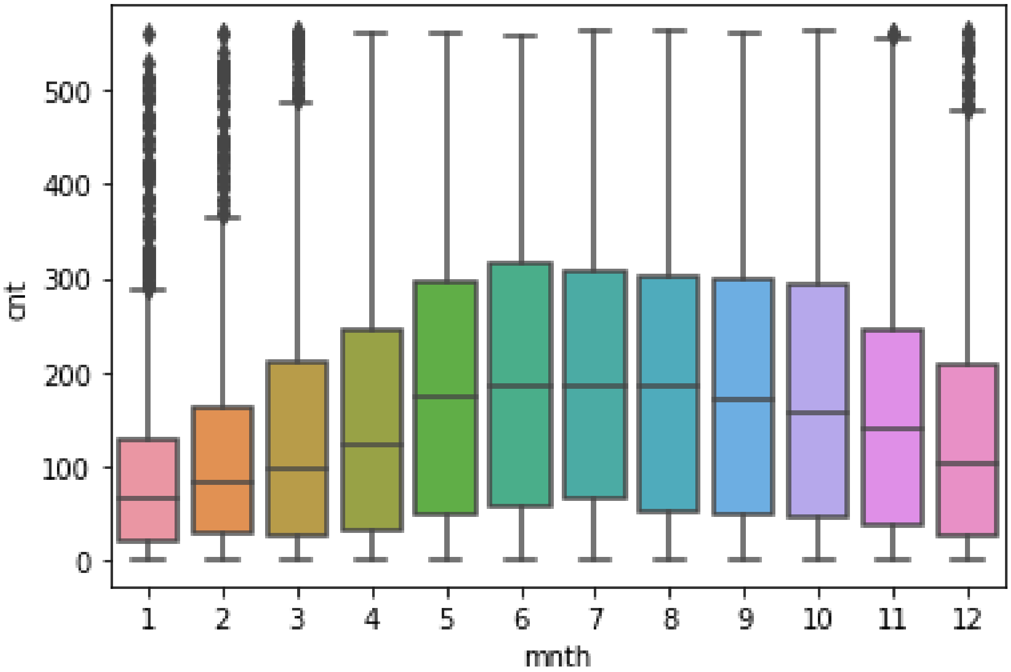


```
<Figure size 864x432 with 0 Axes>
```

[33]:
```
#Make boxplot for cnt vs month
sns.boxplot('mnth','cnt',data=inp2)
plt.figure(figsize=[12,6])
```

```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

[33]:
```
<Figure size 864x432 with 0 Axes>
```

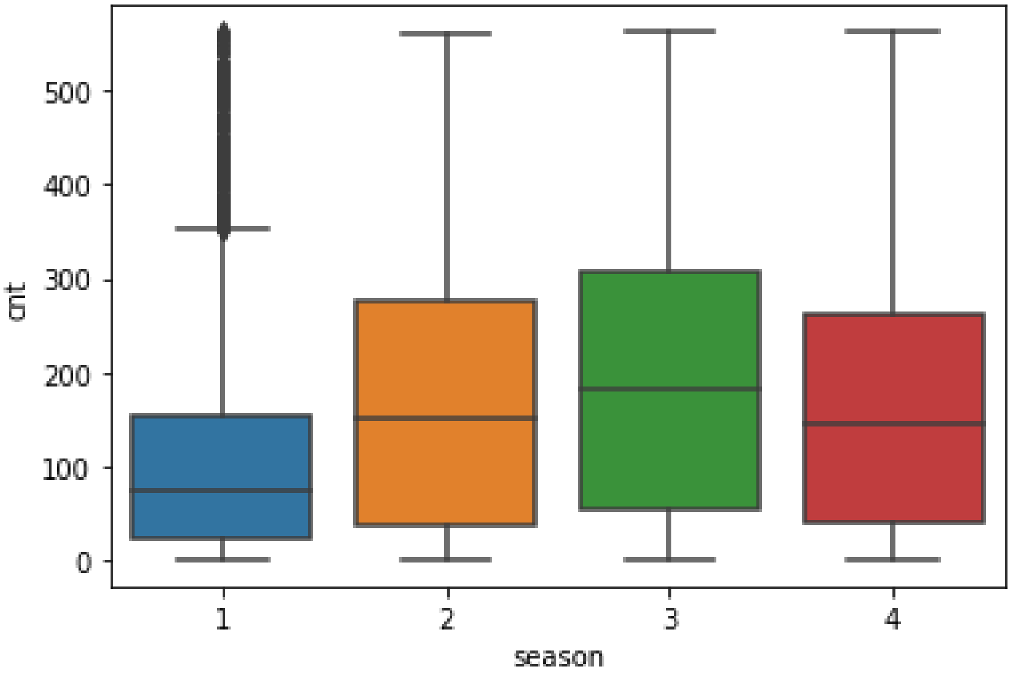


```
<Figure size 864x432 with 0 Axes>
```

[34]:
```
#Make boxplot for cnt vs season
sns.boxplot('season','cnt',data=inp2)
plt.figure(figsize=[12,6])
```

```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

[34]:
```
<Figure size 864x432 with 0 Axes>
```

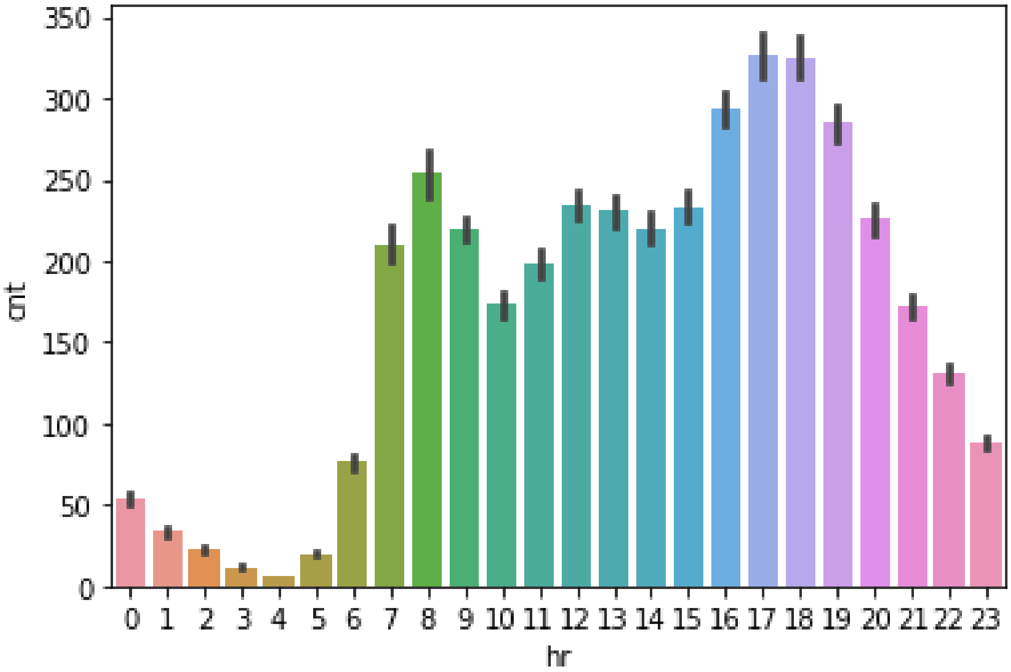


```
<Figure size 864x432 with 0 Axes>
```

[35]:
```
#Make a bar plot with the median value of cnt for each hr
#Does this paint a different picture than the box plot?
sns.barplot('hr','cnt',data=inp2)
```

```
/usr/local/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

[35]:
```
<AxesSubplot:xlabel='hr', ylabel='cnt'>
```



[6]:

```
#Make a correlation matrix for variables - atemp, temp, hum, windspeed
#Which variables have the highest correlation?
list1=['atemp','temp','hum','windspeed']
corrs=inp2[list1].corr()
corrs
```

[6]:

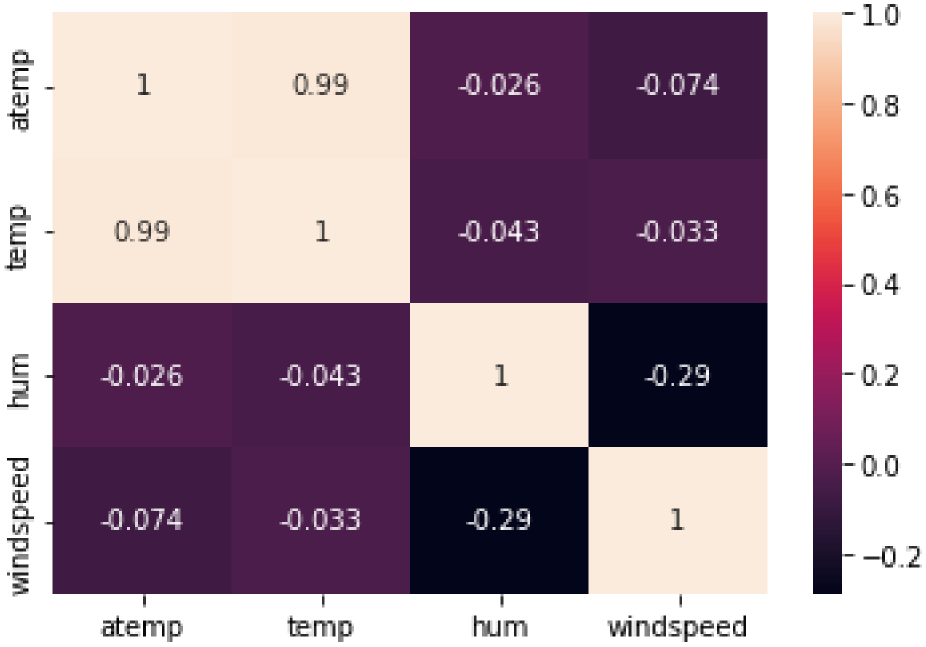
```
              atemp      temp       hum  windspeed
atemp      1.000000  0.988218 -0.025747  -0.073985
temp       0.988218  1.000000 -0.042603  -0.033209
hum       -0.025747 -0.042603  1.000000  -0.288648
windspeed -0.073985 -0.033209 -0.288648   1.000000
```

[37]:

```
#create the heat map
sns.heatmap(corrs,annot=True)
```

[37]:

```
<AxesSubplot:>
```


atemp
temp
hum
windspeed
1
0.99
-0.026
-0.074
0.99
1
-0.043
-0.033
-0.026
-0.043
1
-0.29
-0.074
-0.033
-0.29
1
atemp
temp
hum
windspeed
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

1 Data pre-processing

2 A few key considerations for the pre-processing –

3 We seem to have plenty of categorical features.

4 Since these categorical features can't be used in the predictive model, we need to convert to a suitable numerical representation.

5 Instead of creating dozens of new dummy variables, we will try to club levels of categorical features wherever possible.

6 For a feature with high number of categorical levels, we can club the values that are very similar in value for the target variable

7 First, create a copy of the dataframe into inp3

8 Treating 'mnth' column

9 For values 5,6,7,8,9,10 – replace with a single value 5.

10 This is because these have very similar values for cnt.

11 Get dummies for the updated 6 'mnth' values

```
[39]: inp2.head()
```

```
[39]:    season  yr  mnth  hr  holiday  weekday  workingday  weathersit  temp  \
      0       1   0     1   0        0        6           0           1  0.24
      1       1   0     1   1        0        6           0           1  0.22
      2       1   0     1   2        0        6           0           1  0.22
      3       1   0     1   3        0        6           0           1  0.24
      4       1   0     1   4        0        6           0           1  0.24

          atemp   hum  windspeed  cnt
      0  0.2879  0.81        0.0   16
      1  0.2727  0.80        0.0   40
      2  0.2727  0.80        0.0   32
```

```
3  0.2879  0.75        0.0   13
4  0.2879  0.75        0.0    1
```

```
[7]: inp3=inp2.copy()
```

```
[8]: #Treating 'mnth' column
     #For values 5,6,7,8,9,10 - replace with a single value 5.
     #This is because these have very similar values for cnt.
     #using isin function
     inp3.mnth[inp3.mnth.isin([5,6,7,8,9,10])]=5
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
```

```
[9]: inp3['mnth'].value_counts()
```

```
[9]: 5     8126
     12    1455
     1     1429
     3     1412
     11    1392
     4     1349
     2     1339
     Name: mnth, dtype: int64
```

```
[10]: #Treating 'hr' column
      #Create new mapping: 0-5: 0, 11-15: 11, other values are untouched.
      #Again,the bucketing is done in a way that hr values with similar levels of cnt␣
       ↪are treated the same.
      inp3.hr[inp3.hr.isin([0,1,2,3,4,5])]=0
      inp3.hr[inp3.hr.isin([11,12,13,14,15])]=11
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
```

```
[11]: inp3['hr'].value_counts()
```

```
[11]: 0     4276
      11    3482
      23     728
      22     728
      10     727
      9      727
      21     727
      20     727
      6      725
      7      724
      16     689
      19     671
      8      547
      18     546
      17     478
      Name: hr, dtype: int64
```

```
[12]: #Get dummy columns for season, weathersit, weekday, mnth, hr.
      #We needn’t club these further, because as seen from the box plots,
      #the levels seem to have different values for the median cnt.
      list2=['season','weathersit','weekday','mnth','hr']
      inp3=pd.get_dummies(inp3,columns=list2)
```

```
[13]: inp3.head()
```

```
[13]:    yr  holiday  workingday  temp   atemp   hum  windspeed  cnt  season_1  \
      0   0        0           0  0.24  0.2879  0.81        0.0   16         1
      1   0        0           0  0.22  0.2727  0.80        0.0   40         1
      2   0        0           0  0.22  0.2727  0.80        0.0   32         1
      3   0        0           0  0.24  0.2879  0.75        0.0   13         1
      4   0        0           0  0.24  0.2879  0.75        0.0    1         1

         season_2  …  hr_10  hr_11  hr_16  hr_17  hr_18  hr_19  hr_20  hr_21  \
      0         0  …      0      0      0      0      0      0      0      0
      1         0  …      0      0      0      0      0      0      0      0
      2         0  …      0      0      0      0      0      0      0      0
      3         0  …      0      0      0      0      0      0      0      0
      4         0  …      0      0      0      0      0      0      0      0

         hr_22  hr_23
```

```
0       0       0
1       0       0
2       0       0
3       0       0
4       0       0

[5 rows x 45 columns]
```

[46]:
```
#Train test split - apply 70-30 split
#call the new dataframes df_train, df_test
from sklearn.model_selection import train_test_split
df_train,df_test=train_test_split(inp3,test_size=0.3,random_state=100)
```

[47]:
```
df_train.shape
```

[47]:
```
(11551, 45)
```

[48]:
```
df_test.shape
```

[48]:
```
(4951, 45)
```

[49]:
```
#Separate X and Y for df_train and df_test.
#example - you should have X_train, y_train from df_train.
#y_train should be the cnt column from inp3, X_train should be all other␣
 ↪columns.
y_train=df_train.pop('cnt')
X_train=df_train
```

[50]:
```
y_test=df_test.pop('cnt')
X_test=df_test
```

[51]:
```
X_train
```

[51]:
```
       yr  holiday  workingday  temp   atemp   hum  windspeed  season_1  \
9491    1        0           0  0.24  0.2273  0.70     0.2239         1
8763    1        0           1  0.24  0.2576  0.70     0.1045         1
6559    0        0           1  0.50  0.4848  0.82     0.1045         0
2655    0        0           1  0.70  0.6515  0.65     0.2239         0
5646    0        0           0  0.76  0.6818  0.40     0.2985         0
…     ..       …           …    …      …     …    …          …
17181   1        0           0  0.34  0.3333  0.34     0.1940         1
79      0        0           1  0.22  0.2121  0.51     0.2985         1
12455   1        0           1  0.52  0.5000  0.63     0.0000         0
14761   1        0           1  0.50  0.4848  0.72     0.0896         0
5686    0        0           1  0.66  0.6212  0.50     0.2239         0

       season_2  season_3  …  hr_10  hr_11  hr_16  hr_17  hr_18  hr_19  \
9491          0         0  …      0      0      0      0      0      0
8763          0         0  …      0      0      0      0      0      0
6559          0         0  …      0      0      0      0      0      0
2655          1         0  …      0      1      0      0      0      0
5646          0         1  …      0      0      0      1      0      0
…         …       …  …    …    …    …    …    …    …
17181         0         0  …      0      0      1      0      0      0
79            0         0  …      0      1      0      0      0      0
12455         1         0  …      0      0      0      0      0      0
14761         0         1  …      0      0      0      0      0      0
5686          0         1  …      0      0      0      0      0      0

       hr_20  hr_21  hr_22  hr_23
9491       0      0      0      0
8763       0      0      0      1
6559       0      1      0      0
2655       0      0      0      0
5646       0      0      0      0
…      …    …    …    …
17181      0      0      0      0
79         0      0      0      0
12455      0      0      0      0
14761      0      0      0      0
5686       0      0      0      0

[11551 rows x 44 columns]
```

[52]:

```
y_train
```

[52]:

```
9491      92
8763      60
6559     201
2655     203
5646     398
        …
17181    190
79        57
12455     10
14761    205
5686     218
Name: cnt, Length: 11551, dtype: int64
```

[53]:

```
X_test
```

[53]:

```
       yr  holiday  workingday  temp   atemp   hum  windspeed  season_1  \
14790   1        0           1  0.66  0.6212  0.50     0.1642         0
10590   1        0           1  0.50  0.4848  0.94     0.0896         0
```

```
10635   1        0           0  0.52  0.5000  0.83     0.1642              0
13498   1        0           0  0.60  0.5455  0.88     0.2537              0
9899    1        0           1  0.40  0.4091  0.62     0.2836              1
…    ..       …         …   …     …   …       …       …
13832   1        0           0  0.88  0.8182  0.42     0.2985              0
1807    0        0           1  0.34  0.3030  0.66     0.3881              0
3598    0        0           1  0.64  0.6212  0.29     0.1642              0
14206   1        0           1  0.60  0.5455  0.88     0.1045              0
15498   1        0           1  0.36  0.3485  0.57     0.1940              0

       season_2  season_3  …  hr_10  hr_11  hr_16  hr_17  hr_18  hr_19  \
14790         0         1  …      0      1      0      0      0      0
10590         1         0  …      0      0      0      0      0      0
10635         1         0  …      0      0      0      0      0      0
13498         0         1  …      0      1      0      0      0      0
9899          0         0  …      0      1      0      0      0      0
…         …       …  …    …    …    …    …    …    …
13832         0         1  …      0      1      0      0      0      0
1807          1         0  …      0      0      0      0      0      0
3598          1         0  …      0      0      0      0      0      0
14206         0         1  …      0      0      0      0      0      0
15498         0         0  …      0      0      0      0      0      0

       hr_20  hr_21  hr_22  hr_23
14790      0      0      0      0
10590      0      0      0      0
10635      0      0      0      0
13498      0      0      0      0
9899       0      0      0      0
…      …    …    …    …
13832      0      0      0      0
1807       0      0      0      0
3598       1      0      0      0
14206      0      0      0      0
15498      0      0      0      1

[4951 rows x 44 columns]
```

[54]:
```
y_test
```

[54]:
```
14790    254
10590    318
10635     28
13498    380
9899     207
        …
13832    527
```

```
1807      13
3598     293
14206      3
15498    100
Name: cnt, Length: 4951, dtype: int64
```

[75]:

```
#Model building
#Use Linear regression as the technique
#Report the R2 on the train set
from sklearn.linear_model import LinearRegression
lin_reg=LinearRegression()
```

[76]:

```
#using fit() method for training
lin_reg.fit(X_train,y_train)
```

[76]:

```
LinearRegression()
```

[77]:

```
y_pred=lin_reg.predict(X_test)
```

[78]:

```
y_pred
```

[78]:

```
array([283.1875, 253.5   , 135.625 , …, 272.125 ,  68.875 , 151.6875])
```

[79]:

```
#Reporting r2 for the model
from sklearn.metrics import r2_score
print(r2_score(y_pred,y_test))
```

```
0.49236824233484466
```

[83]:

```
from sklearn.metrics import r2_score
print(r2_score(lin_reg.predict(X_train),y_train))
```

```
0.5074631433908097
```

[ ]:

```

```