# Importing Libraries  
  
import os  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.cross\_validation import train\_test\_split  
from sklearn.model\_selection import GridSearchCV  
from sklearn.metrics import mean\_squared\_error  
from math import sqrt  
from scipy.stats import chi2\_contingency  
  
  
from sklearn.linear\_model import LinearRegression  
from sklearn.tree import DecisionTreeRegressor  
from sklearn.ensemble import RandomForestRegressor  
  
%matplotlib inline

# Setting Working Directory  
  
os.chdir("C:\\Users\\DELL\\Desktop\\project 2")

# Loading Data  
  
data = pd.read\_csv('project\_2.csv')

data.shape

(731, 32)

type(data)

pandas.core.frame.DataFrame

data.columns

Index(['yr', 'holiday', 'workingday', 'temp', 'atemp', 'hum', 'windspeed',  
 'casual', 'registered', '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')

data.describe

<bound method NDFrame.describe of yr holiday workingday temp atemp hum windspeed casual \  
0 0 0 0 0.344167 0.363625 0.805833 0.160446 331   
1 0 0 0 0.363478 0.353739 0.696087 0.248539 131   
2 0 0 1 0.196364 0.189405 0.437273 0.248309 120   
3 0 0 1 0.200000 0.212122 0.590435 0.160296 108   
4 0 0 1 0.226957 0.229270 0.436957 0.186900 82   
5 0 0 1 0.204348 0.233209 0.518261 0.089565 88   
6 0 0 1 0.196522 0.208839 0.498696 0.168726 148   
7 0 0 0 0.165000 0.162254 0.535833 0.266804 68   
8 0 0 0 0.138333 0.116175 0.434167 0.361950 54   
9 0 0 1 0.150833 0.150888 0.482917 0.223267 41   
10 0 0 1 0.169091 0.191464 0.686364 0.122132 43   
11 0 0 1 0.172727 0.160473 0.599545 0.304627 25   
12 0 0 1 0.165000 0.150883 0.470417 0.301000 38   
13 0 0 1 0.160870 0.188413 0.537826 0.126548 54   
14 0 0 0 0.233333 0.248112 0.498750 0.157963 222   
15 0 0 0 0.231667 0.234217 0.483750 0.188433 251   
16 0 1 0 0.175833 0.176771 0.537500 0.194017 117   
17 0 0 1 0.216667 0.232333 0.861667 0.146775 9   
18 0 0 1 0.292174 0.298422 0.741739 0.208317 78   
19 0 0 1 0.261667 0.255050 0.538333 0.195904 83   
20 0 0 1 0.177500 0.157833 0.457083 0.353242 75   
21 0 0 0 0.059130 0.079070 0.400000 0.171970 93   
22 0 0 0 0.096522 0.098839 0.436522 0.246600 150   
23 0 0 1 0.097391 0.117930 0.491739 0.158330 86   
24 0 0 1 0.223478 0.234526 0.616957 0.129796 186   
25 0 0 1 0.217500 0.203600 0.862500 0.293850 34   
26 0 0 1 0.195000 0.219700 0.687500 0.113837 15   
27 0 0 1 0.203478 0.223317 0.793043 0.123300 38   
28 0 0 0 0.196522 0.212126 0.651739 0.145365 123   
29 0 0 0 0.216522 0.250322 0.722174 0.073983 140   
.. .. ... ... ... ... ... ... ...   
701 1 0 0 0.347500 0.359208 0.823333 0.124379 892   
702 1 0 1 0.452500 0.455796 0.767500 0.082721 555   
703 1 0 1 0.475833 0.469054 0.733750 0.174129 551   
704 1 0 1 0.438333 0.428012 0.485000 0.324021 331   
705 1 0 1 0.255833 0.258204 0.508750 0.174754 340   
706 1 0 1 0.320833 0.321958 0.764167 0.130600 349   
707 1 0 0 0.381667 0.389508 0.911250 0.101379 1153   
708 1 0 0 0.384167 0.390146 0.905417 0.157975 441   
709 1 0 1 0.435833 0.435575 0.925000 0.190308 329   
710 1 0 1 0.353333 0.338363 0.596667 0.296037 282   
711 1 0 1 0.297500 0.297338 0.538333 0.162937 310   
712 1 0 1 0.295833 0.294188 0.485833 0.174129 425   
713 1 0 1 0.281667 0.294192 0.642917 0.131229 429   
714 1 0 0 0.324167 0.338383 0.650417 0.106350 767   
715 1 0 0 0.362500 0.369938 0.838750 0.100742 538   
716 1 0 1 0.393333 0.401500 0.907083 0.098258 212   
717 1 0 1 0.410833 0.409708 0.666250 0.221404 433   
718 1 0 1 0.332500 0.342162 0.625417 0.184092 333   
719 1 0 1 0.330000 0.335217 0.667917 0.132463 314   
720 1 0 1 0.326667 0.301767 0.556667 0.374383 221   
721 1 0 0 0.265833 0.236113 0.441250 0.407346 205   
722 1 0 0 0.245833 0.259471 0.515417 0.133083 408   
723 1 0 1 0.231304 0.258900 0.791304 0.077230 174   
724 1 1 0 0.291304 0.294465 0.734783 0.168726 440   
725 1 0 1 0.243333 0.220333 0.823333 0.316546 9   
726 1 0 1 0.254167 0.226642 0.652917 0.350133 247   
727 1 0 1 0.253333 0.255046 0.590000 0.155471 644   
728 1 0 0 0.253333 0.242400 0.752917 0.124383 159   
729 1 0 0 0.255833 0.231700 0.483333 0.350754 364   
730 1 0 1 0.215833 0.223487 0.577500 0.154846 439   
  
 registered cnt ... mnth\_11 mnth\_12 weekday\_1 weekday\_2 \  
0 654 985 ... 0 0 0 0   
1 670 801 ... 0 0 0 0   
2 1229 1349 ... 0 0 1 0   
3 1454 1562 ... 0 0 0 1   
4 1518 1600 ... 0 0 0 0   
5 1518 1606 ... 0 0 0 0   
6 1362 1510 ... 0 0 0 0   
7 891 959 ... 0 0 0 0   
8 768 822 ... 0 0 0 0   
9 1280 1321 ... 0 0 1 0   
10 1220 1263 ... 0 0 0 1   
11 1137 1162 ... 0 0 0 0   
12 1368 1406 ... 0 0 0 0   
13 1367 1421 ... 0 0 0 0   
14 1026 1248 ... 0 0 0 0   
15 953 1204 ... 0 0 0 0   
16 883 1000 ... 0 0 1 0   
17 674 683 ... 0 0 0 1   
18 1572 1650 ... 0 0 0 0   
19 1844 1927 ... 0 0 0 0   
20 1468 1543 ... 0 0 0 0   
21 888 981 ... 0 0 0 0   
22 836 986 ... 0 0 0 0   
23 1330 1416 ... 0 0 1 0   
24 1799 1985 ... 0 0 0 1   
25 472 506 ... 0 0 0 0   
26 416 431 ... 0 0 0 0   
27 1129 1167 ... 0 0 0 0   
28 975 1098 ... 0 0 0 0   
29 956 1096 ... 0 0 0 0   
.. ... ... ... ... ... ... ...   
701 3757 4649 ... 0 1 0 0   
702 5679 6234 ... 0 1 1 0   
703 6055 6606 ... 0 1 0 1   
704 5398 5729 ... 0 1 0 0   
705 5035 5375 ... 0 1 0 0   
706 4659 5008 ... 0 1 0 0   
707 4429 5582 ... 0 1 0 0   
708 2787 3228 ... 0 1 0 0   
709 4841 5170 ... 0 1 1 0   
710 5219 5501 ... 0 1 0 1   
711 5009 5319 ... 0 1 0 0   
712 5107 5532 ... 0 1 0 0   
713 5182 5611 ... 0 1 0 0   
714 4280 5047 ... 0 1 0 0   
715 3248 3786 ... 0 1 0 0   
716 4373 4585 ... 0 1 1 0   
717 5124 5557 ... 0 1 0 1   
718 4934 5267 ... 0 1 0 0   
719 3814 4128 ... 0 1 0 0   
720 3402 3623 ... 0 1 0 0   
721 1544 1749 ... 0 1 0 0   
722 1379 1787 ... 0 1 0 0   
723 746 920 ... 0 1 1 0   
724 573 1013 ... 0 1 0 1   
725 432 441 ... 0 1 0 0   
726 1867 2114 ... 0 1 0 0   
727 2451 3095 ... 0 1 0 0   
728 1182 1341 ... 0 1 0 0   
729 1432 1796 ... 0 1 0 0   
730 2290 2729 ... 0 1 1 0   
  
 weekday\_3 weekday\_4 weekday\_5 weekday\_6 weathersit\_2 weathersit\_3   
0 0 0 0 1 1 0   
1 0 0 0 0 1 0   
2 0 0 0 0 0 0   
3 0 0 0 0 0 0   
4 1 0 0 0 0 0   
5 0 1 0 0 0 0   
6 0 0 1 0 1 0   
7 0 0 0 1 1 0   
8 0 0 0 0 0 0   
9 0 0 0 0 0 0   
10 0 0 0 0 1 0   
11 1 0 0 0 0 0   
12 0 1 0 0 0 0   
13 0 0 1 0 0 0   
14 0 0 0 1 1 0   
15 0 0 0 0 0 0   
16 0 0 0 0 1 0   
17 0 0 0 0 1 0   
18 1 0 0 0 1 0   
19 0 1 0 0 1 0   
20 0 0 1 0 0 0   
21 0 0 0 1 0 0   
22 0 0 0 0 0 0   
23 0 0 0 0 0 0   
24 0 0 0 0 1 0   
25 1 0 0 0 0 1   
26 0 1 0 0 0 0   
27 0 0 1 0 1 0   
28 0 0 0 1 0 0   
29 0 0 0 0 0 0   
.. ... ... ... ... ... ...   
701 0 0 0 0 1 0   
702 0 0 0 0 0 0   
703 0 0 0 0 0 0   
704 1 0 0 0 0 0   
705 0 1 0 0 0 0   
706 0 0 1 0 1 0   
707 0 0 0 1 1 0   
708 0 0 0 0 1 0   
709 0 0 0 0 1 0   
710 0 0 0 0 1 0   
711 1 0 0 0 1 0   
712 0 1 0 0 0 0   
713 0 0 1 0 0 0   
714 0 0 0 1 0 0   
715 0 0 0 0 1 0   
716 0 0 0 0 1 0   
717 0 0 0 0 0 0   
718 1 0 0 0 0 0   
719 0 1 0 0 1 0   
720 0 0 1 0 1 0   
721 0 0 0 1 0 0   
722 0 0 0 0 0 0   
723 0 0 0 0 1 0   
724 0 0 0 0 1 0   
725 1 0 0 0 0 1   
726 0 1 0 0 1 0   
727 0 0 1 0 1 0   
728 0 0 0 1 1 0   
729 0 0 0 0 0 0   
730 0 0 0 0 1 0   
  
[731 rows x 32 columns]>

# Exploratory data Analysis¶

data.head()

<tr style="text-align: right;">  
 <th></th>  
 <th>instant</th>  
 <th>dteday</th>  
 <th>season</th>  
 <th>yr</th>  
 <th>mnth</th>  
 <th>holiday</th>  
 <th>weekday</th>  
 <th>workingday</th>  
 <th>weathersit</th>  
 <th>temp</th>  
 <th>atemp</th>  
 <th>hum</th>  
 <th>windspeed</th>  
 <th>casual</th>  
 <th>registered</th>  
 <th>cnt</th>  
</tr>

<tr>  
 <th>0</th>  
 <td>1</td>  
 <td>01/01/2011</td>  
 <td>1</td>  
 <td>0</td>  
 <td>1</td>  
 <td>0</td>  
 <td>6</td>  
 <td>0</td>  
 <td>2</td>  
 <td>0.344167</td>  
 <td>0.363625</td>  
 <td>0.805833</td>  
 <td>0.160446</td>  
 <td>331</td>  
 <td>654</td>  
 <td>985</td>  
</tr>  
<tr>  
 <th>1</th>  
 <td>2</td>  
 <td>02/01/2011</td>  
 <td>1</td>  
 <td>0</td>  
 <td>1</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>2</td>  
 <td>0.363478</td>  
 <td>0.353739</td>  
 <td>0.696087</td>  
 <td>0.248539</td>  
 <td>131</td>  
 <td>670</td>  
 <td>801</td>  
</tr>  
<tr>  
 <th>2</th>  
 <td>3</td>  
 <td>03/01/2011</td>  
 <td>1</td>  
 <td>0</td>  
 <td>1</td>  
 <td>0</td>  
 <td>1</td>  
 <td>1</td>  
 <td>1</td>  
 <td>0.196364</td>  
 <td>0.189405</td>  
 <td>0.437273</td>  
 <td>0.248309</td>  
 <td>120</td>  
 <td>1229</td>  
 <td>1349</td>  
</tr>  
<tr>  
 <th>3</th>  
 <td>4</td>  
 <td>04/01/2011</td>  
 <td>1</td>  
 <td>0</td>  
 <td>1</td>  
 <td>0</td>  
 <td>2</td>  
 <td>1</td>  
 <td>1</td>  
 <td>0.200000</td>  
 <td>0.212122</td>  
 <td>0.590435</td>  
 <td>0.160296</td>  
 <td>108</td>  
 <td>1454</td>  
 <td>1562</td>  
</tr>  
<tr>  
 <th>4</th>  
 <td>5</td>  
 <td>05/01/2011</td>  
 <td>1</td>  
 <td>0</td>  
 <td>1</td>  
 <td>0</td>  
 <td>3</td>  
 <td>1</td>  
 <td>1</td>  
 <td>0.226957</td>  
 <td>0.229270</td>  
 <td>0.436957</td>  
 <td>0.186900</td>  
 <td>82</td>  
 <td>1518</td>  
 <td>1600</td>  
</tr>

features = pd.DataFrame(data.columns)  
#features.to\_csv('features.csv')

# Continous variables  
  
cnames = ['temp','atemp','hum','windspeed']  
cat\_names = ['season', 'mnth', 'holiday', 'weekday', 'workingday', 'weathersit']

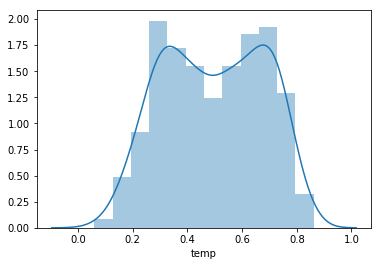
# Corelation between continous variables  
  
corr = data[cnames].corr()  
corr  
#corr.to\_csv('Correlations.csv')

<tr style="text-align: right;">  
 <th></th>  
 <th>temp</th>  
 <th>atemp</th>  
 <th>hum</th>  
 <th>windspeed</th>  
</tr>

<tr>  
 <th>temp</th>  
 <td>1.000000</td>  
 <td>0.991702</td>  
 <td>0.126963</td>  
 <td>-0.157944</td>  
</tr>  
<tr>  
 <th>atemp</th>  
 <td>0.991702</td>  
 <td>1.000000</td>  
 <td>0.139988</td>  
 <td>-0.183643</td>  
</tr>  
<tr>  
 <th>hum</th>  
 <td>0.126963</td>  
 <td>0.139988</td>  
 <td>1.000000</td>  
 <td>-0.248489</td>  
</tr>  
<tr>  
 <th>windspeed</th>  
 <td>-0.157944</td>  
 <td>-0.183643</td>  
 <td>-0.248489</td>  
 <td>1.000000</td>  
</tr>

sns.distplot(data['temp'])  
#plt.savefig('temp.png')

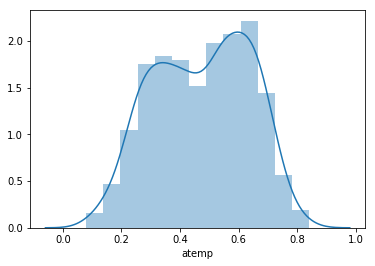
C:\Users\DELL\Anaconda4\lib\site-packages\matplotlib\axes\\_axes.py:6462: UserWarning: The 'normed' kwarg is deprecated, and has been replaced by the 'density' kwarg.  
 warnings.warn("The 'normed' kwarg is deprecated, and has been "  
  
  
  
  
  
<matplotlib.axes.\_subplots.AxesSubplot at 0x234a6a30198>



png

sns.distplot(data['atemp'])  
#plt.savefig('atemp.png')

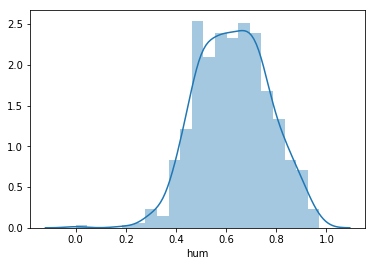
C:\Users\DELL\Anaconda4\lib\site-packages\matplotlib\axes\\_axes.py:6462: UserWarning: The 'normed' kwarg is deprecated, and has been replaced by the 'density' kwarg.  
 warnings.warn("The 'normed' kwarg is deprecated, and has been "  
  
  
  
  
  
<matplotlib.axes.\_subplots.AxesSubplot at 0x234a6e2ab00>



png

sns.distplot(data['hum'])  
#plt.savefig('hum.png')

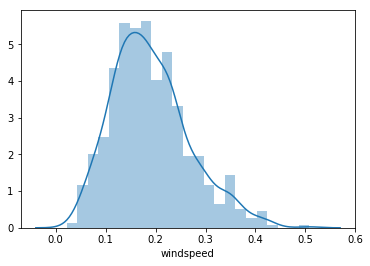
C:\Users\DELL\Anaconda4\lib\site-packages\matplotlib\axes\\_axes.py:6462: UserWarning: The 'normed' kwarg is deprecated, and has been replaced by the 'density' kwarg.  
 warnings.warn("The 'normed' kwarg is deprecated, and has been "  
  
  
  
  
  
<matplotlib.axes.\_subplots.AxesSubplot at 0x234a6e983c8>



png

sns.distplot(data['windspeed'])  
#plt.savefig('windspeed.png')

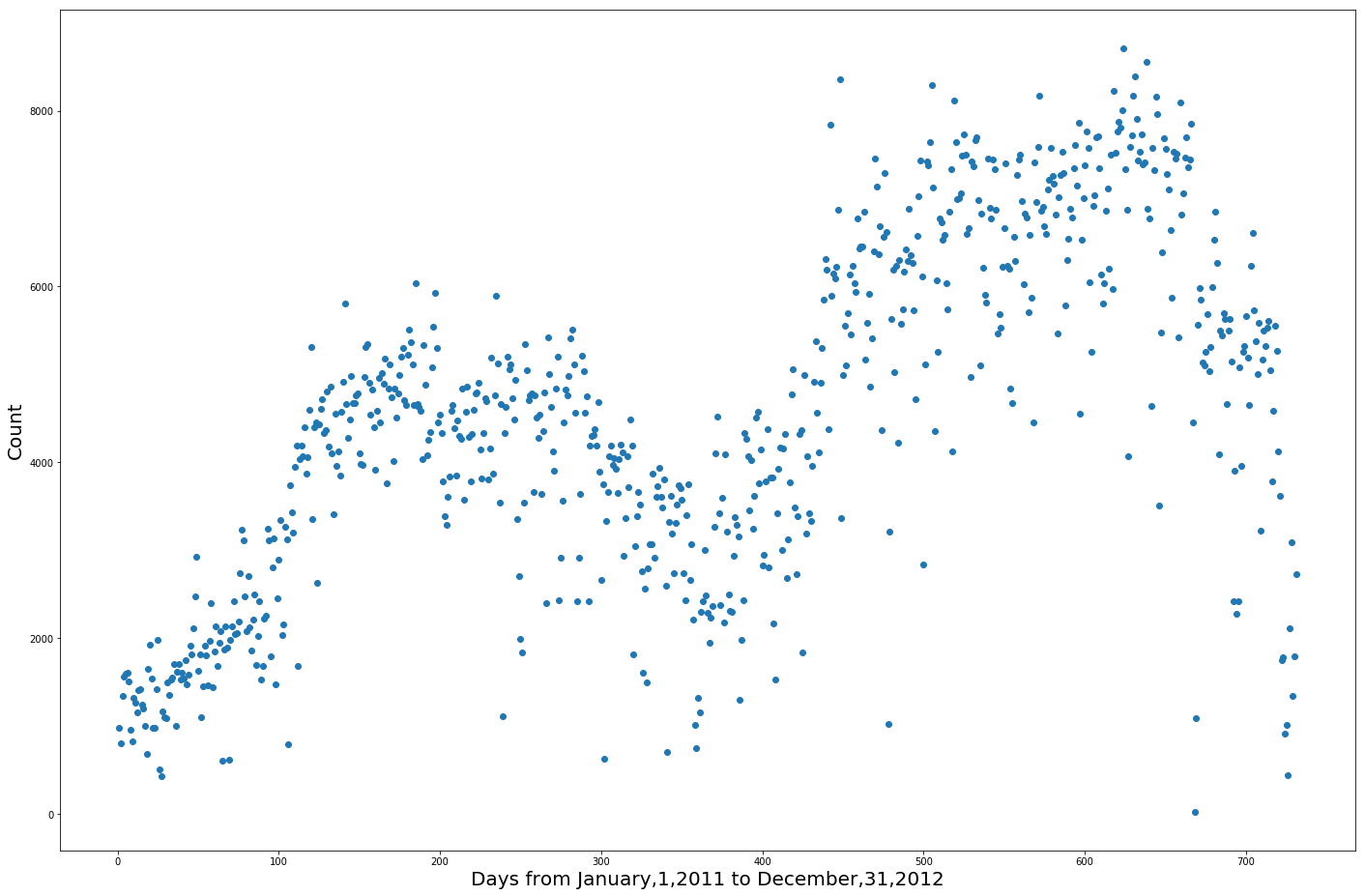
C:\Users\DELL\Anaconda4\lib\site-packages\matplotlib\axes\\_axes.py:6462: UserWarning: The 'normed' kwarg is deprecated, and has been replaced by the 'density' kwarg.  
 warnings.warn("The 'normed' kwarg is deprecated, and has been "  
  
  
  
  
  
<matplotlib.axes.\_subplots.AxesSubplot at 0x234a6f07b00>



png

plt.figure(figsize=(24,16))  
plt.scatter(data['instant'], data['cnt'])  
plt.xlabel('Days from January,1,2011 to December,31,2012', fontsize = 20)  
plt.ylabel('Count', fontsize =20)  
#plt.savefig('RentCount.png')

Text(0,0.5,'Count')



png

# Creating Dummy Variables for non-binary categorical variables  
  
for i in ['season','mnth','weekday','weathersit']:  
 temp = pd.get\_dummies(data[i], prefix = i)  
 data = data.join(temp)  
 data.drop(i, axis =1,inplace = True)

data.drop(['instant','dteday','season\_1','mnth\_1','weekday\_0','weathersit\_1'],axis=1,inplace = True)

data.head()

<tr style="text-align: right;">  
 <th></th>  
 <th>yr</th>  
 <th>holiday</th>  
 <th>workingday</th>  
 <th>temp</th>  
 <th>atemp</th>  
 <th>hum</th>  
 <th>windspeed</th>  
 <th>casual</th>  
 <th>registered</th>  
 <th>cnt</th>  
 <th>...</th>  
 <th>mnth\_11</th>  
 <th>mnth\_12</th>  
 <th>weekday\_1</th>  
 <th>weekday\_2</th>  
 <th>weekday\_3</th>  
 <th>weekday\_4</th>  
 <th>weekday\_5</th>  
 <th>weekday\_6</th>  
 <th>weathersit\_2</th>  
 <th>weathersit\_3</th>  
</tr>

<tr>  
 <th>0</th>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0.344167</td>  
 <td>0.363625</td>  
 <td>0.805833</td>  
 <td>0.160446</td>  
 <td>331</td>  
 <td>654</td>  
 <td>985</td>  
 <td>...</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>1</td>  
 <td>1</td>  
 <td>0</td>  
</tr>  
<tr>  
 <th>1</th>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0.363478</td>  
 <td>0.353739</td>  
 <td>0.696087</td>  
 <td>0.248539</td>  
 <td>131</td>  
 <td>670</td>  
 <td>801</td>  
 <td>...</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>1</td>  
 <td>0</td>  
</tr>  
<tr>  
 <th>2</th>  
 <td>0</td>  
 <td>0</td>  
 <td>1</td>  
 <td>0.196364</td>  
 <td>0.189405</td>  
 <td>0.437273</td>  
 <td>0.248309</td>  
 <td>120</td>  
 <td>1229</td>  
 <td>1349</td>  
 <td>...</td>  
 <td>0</td>  
 <td>0</td>  
 <td>1</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
</tr>  
<tr>  
 <th>3</th>  
 <td>0</td>  
 <td>0</td>  
 <td>1</td>  
 <td>0.200000</td>  
 <td>0.212122</td>  
 <td>0.590435</td>  
 <td>0.160296</td>  
 <td>108</td>  
 <td>1454</td>  
 <td>1562</td>  
 <td>...</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>1</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
</tr>  
<tr>  
 <th>4</th>  
 <td>0</td>  
 <td>0</td>  
 <td>1</td>  
 <td>0.226957</td>  
 <td>0.229270</td>  
 <td>0.436957</td>  
 <td>0.186900</td>  
 <td>82</td>  
 <td>1518</td>  
 <td>1600</td>  
 <td>...</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>1</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
 <td>0</td>  
</tr>

5 rows × 32 columns

# Splitting the data into train and test sets  
  
train,test = train\_test\_split(data,test\_size =0.2, random\_state =0)

# Preparing Data for modelling  
  
X\_train = train.drop(['casual','registered','cnt','temp'],axis=1)  
X\_test = test.drop(['casual','registered','cnt','temp'],axis=1)  
y\_casual = train['casual']  
y\_registered = train['registered']  
y\_cnt = train['cnt']

# Evaluation Functions  
  
def MAPE(y\_true, y\_pred):   
 mape = np.mean(np.abs((y\_true - y\_pred) / y\_true))\*100  
 return mape  
#Calculate MAPE  
  
def RMSE(y\_true, y\_pred):  
 rms = sqrt(mean\_squared\_error(y\_true, y\_pred))  
 return rms  
#Calculate RMSE

# Regression Models

# Multiple linear Regression

from sklearn.linear\_model import LinearRegression

# Grid Search for best Parameters  
  
reg\_lm = LinearRegression()  
params\_lm = [{'copy\_X':[True, False],  
 'fit\_intercept':[True,False],  
 'normalize':[True, False]}]  
grid\_search\_lm = GridSearchCV(reg\_lm, param\_grid = params\_lm, cv =10, n\_jobs =-1)  
grid\_search\_lm = grid\_search\_lm.fit(X\_train,y\_cnt)

grid\_search\_lm.best\_score\_

0.8103662854156968

grid\_search\_lm.best\_params\_

{'copy\_X': True, 'fit\_intercept': True, 'normalize': True}

# Training with best paramaeters  
  
reg\_lm\_best = LinearRegression(copy\_X=True, fit\_intercept=True, normalize=True)  
reg\_lm\_best.fit(X\_train,y\_cnt)

LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=1, normalize=True)

# Evaluating on training set  
  
y\_pred\_lm = reg\_lm\_best.predict(X\_train)  
mape1\_lm = MAPE(y\_cnt, y\_pred\_lm)  
rmse1\_lm = RMSE(y\_cnt, y\_pred\_lm)  
print('MAPE : {:.2f}'.format(mape1\_lm))  
print('RMSE : {:.2f}'.format(rmse1\_lm))

MAPE : 46.16  
RMSE : 765.90

# Evaluating on Test Set  
  
y\_pred\_lm = reg\_lm\_best.predict(X\_test)  
mape2\_lm = MAPE(test['cnt'], y\_pred\_lm)  
rmse2\_lm = RMSE(test['cnt'], y\_pred\_lm)  
print('MAPE : {:.2f}'.format(mape2\_lm))  
print('RMSE : {:.2f}'.format(rmse2\_lm))

MAPE : 19.08  
RMSE : 794.45

# Decision Tree Regressor

from sklearn.tree import DecisionTreeRegressor

# Grid Search for best Parameters  
  
reg\_dt = DecisionTreeRegressor(random\_state = 0)  
params = [{'max\_depth':[2,4,6,8,10,12,15],  
 'max\_features':['auto','sqrt'],  
 'min\_samples\_leaf':[2,4,6,8,10]}]  
grid\_search\_dt = GridSearchCV(reg\_dt, param\_grid = params, cv =10, n\_jobs =-1)  
grid\_search\_dt = grid\_search\_dt.fit(X\_train,y\_cnt)

grid\_search\_dt.best\_score\_

0.7923245462173745

grid\_search\_dt.best\_params\_

{'max\_depth': 12, 'max\_features': 'auto', 'min\_samples\_leaf': 10}

# Training with best parameters  
  
reg\_dt\_best = DecisionTreeRegressor(random\_state = 0, max\_depth = 12,  
 min\_samples\_leaf = 10, max\_features = 'auto')  
reg\_dt\_best.fit(X\_train,y\_cnt)

DecisionTreeRegressor(criterion='mse', max\_depth=12, max\_features='auto',  
 max\_leaf\_nodes=None, min\_impurity\_decrease=0.0,  
 min\_impurity\_split=None, min\_samples\_leaf=10,  
 min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,  
 presort=False, random\_state=0, splitter='best')

# Evaluating on training set  
  
b = reg\_dt\_best.predict(X\_train)  
mape1\_dt = MAPE(y\_cnt,b)  
rmse1\_dt = RMSE(y\_cnt,b)  
print('MAPE : {:.2f}'.format(mape1\_dt))  
print('RMSE : {:.2f}'.format(rmse1\_dt))

MAPE : 51.81  
RMSE : 706.58

# Evaluating on test set  
  
y\_pred\_dt = reg\_dt\_best.predict(X\_test)  
mape2\_dt = MAPE(test['cnt'],y\_pred\_dt)  
rmse2\_dt = RMSE(test['cnt'],y\_pred\_dt)  
print('MAPE : {:.2f}'.format(mape2\_dt))  
print('RMSE : {:.2f}'.format(rmse2\_dt))

MAPE : 25.13  
RMSE : 892.36

# Random Forest Regressor

from sklearn.ensemble import RandomForestRegressor

# Grid Search for best Parameters  
  
reg\_rf = RandomForestRegressor(random\_state = 0)  
params\_rf = [{'max\_depth':[8,10,12,15],  
 'max\_features':['auto','sqrt'],  
 'min\_samples\_leaf':[2,4,6,8,10],  
 'n\_estimators': [200, 500, 600],  
 'oob\_score':[True, False]}]  
grid\_search\_rf = GridSearchCV(reg\_rf, param\_grid = params\_rf, cv =10, n\_jobs =-1)  
grid\_search\_rf = grid\_search\_rf.fit(X\_train,y\_cnt)

grid\_search\_rf.best\_score\_

0.8467041706322699

grid\_search\_rf.best\_params\_

{'max\_depth': 12,  
 'max\_features': 'auto',  
 'min\_samples\_leaf': 2,  
 'n\_estimators': 500,  
 'oob\_score': True}

reg\_rf\_best = RandomForestRegressor(random\_state = 0, max\_depth = 15,  
 max\_features = 'auto', min\_samples\_leaf = 2,  
 n\_estimators = 600, oob\_score = True)  
reg\_rf\_best.fit(X\_train,y\_cnt)

RandomForestRegressor(bootstrap=True, criterion='mse', max\_depth=15,  
 max\_features='auto', max\_leaf\_nodes=None,  
 min\_impurity\_decrease=0.0, min\_impurity\_split=None,  
 min\_samples\_leaf=2, min\_samples\_split=2,  
 min\_weight\_fraction\_leaf=0.0, n\_estimators=600, n\_jobs=1,  
 oob\_score=True, random\_state=0, verbose=0, warm\_start=False)

# Evaluating on training set  
  
c = reg\_rf\_best.predict(X\_train)  
mape1\_rf = MAPE(y\_cnt,c)  
rmse1\_rf = RMSE(y\_cnt,c)  
print('MAPE : {:.2f}'.format(mape1\_rf))  
print('RMSE : {:.2f}'.format(rmse1\_rf))

MAPE : 27.10  
RMSE : 368.15

# Evaluating on test set  
  
y\_pred\_rf = reg\_rf\_best.predict(X\_test)  
mape2\_rf = MAPE(test['cnt'],y\_pred\_rf)  
rmse2\_rf = RMSE(test['cnt'],y\_pred\_rf)  
print('MAPE : {:.2f}'.format(mape2\_rf))  
print('RMSE : {:.2f}'.format(rmse2\_rf))

MAPE : 20.02  
RMSE : 708.95

# Result

result = pd.DataFrame()  
result['Model'] = ['Multiple Linear Regressor',  
 'Decision Tree Regressor', 'Random Forest Regressor']  
result['Training MAPE'] = [mape1\_lm, mape1\_dt, mape1\_rf]  
result['Training RMSE'] = [rmse1\_lm, rmse1\_dt, rmse1\_rf]  
result['Test MAPE'] = [mape2\_lm, mape2\_dt, mape2\_rf]  
result['Test RMSE'] = [rmse2\_lm, rmse2\_dt, rmse2\_rf]  
#result.to\_csv('result.csv')

# Evaluating on Test Set  
  
y\_pred = reg\_svr\_best.predict(X\_test)  
mape2\_svr = MAPE(test['cnt'],y\_pred)  
rmse2\_svr = RMSE(test['cnt'],y\_pred)  
print('MAPE : {:.2f}'.format(mape2\_svr))  
print('RMSE : {:.2f}'.format(rmse2\_svr))

MAPE : 18.01  
RMSE : 765.09

result

<tr style="text-align: right;">  
 <th></th>  
 <th>Model</th>  
 <th>Training MAPE</th>  
 <th>Training RMSE</th>  
 <th>Test MAPE</th>  
 <th>Test RMSE</th>  
</tr>

<tr>  
 <th>0</th>  
 <td>Multiple Linear Regressor</td>  
 <td>46.161484</td>  
 <td>765.895325</td>  
 <td>19.081074</td>  
 <td>794.448747</td>  
</tr>  
<tr>  
 <th>1</th>  
 <td>Decision Tree Regressor</td>  
 <td>51.809725</td>  
 <td>706.577452</td>  
 <td>25.133274</td>  
 <td>892.361957</td>  
</tr>  
<tr>  
 <th>2</th>  
 <td>Random Forest Regressor</td>  
 <td>27.097224</td>  
 <td>368.149119</td>  
 <td>20.023176</td>  
 <td>708.951493</td>  
</tr>

# Evaluating on Test Set  
  
y\_pred = reg\_svr\_best.predict(X\_test)  
mape2\_svr = MAPE(test['cnt'],y\_pred)  
rmse2\_svr = RMSE(test['cnt'],y\_pred)  
print('MAPE : {:.2f}'.format(mape2\_svr))  
print('RMSE : {:.2f}'.format(rmse2\_svr))

MAPE : 18.01  
RMSE : 765.09

# Output using Selected Model i.e. Random Forest Regressor

#pd.DataFrame(y\_pred\_rf).to\_csv('Output.csv')