21BAI1217 LAB 3 EVALUATION

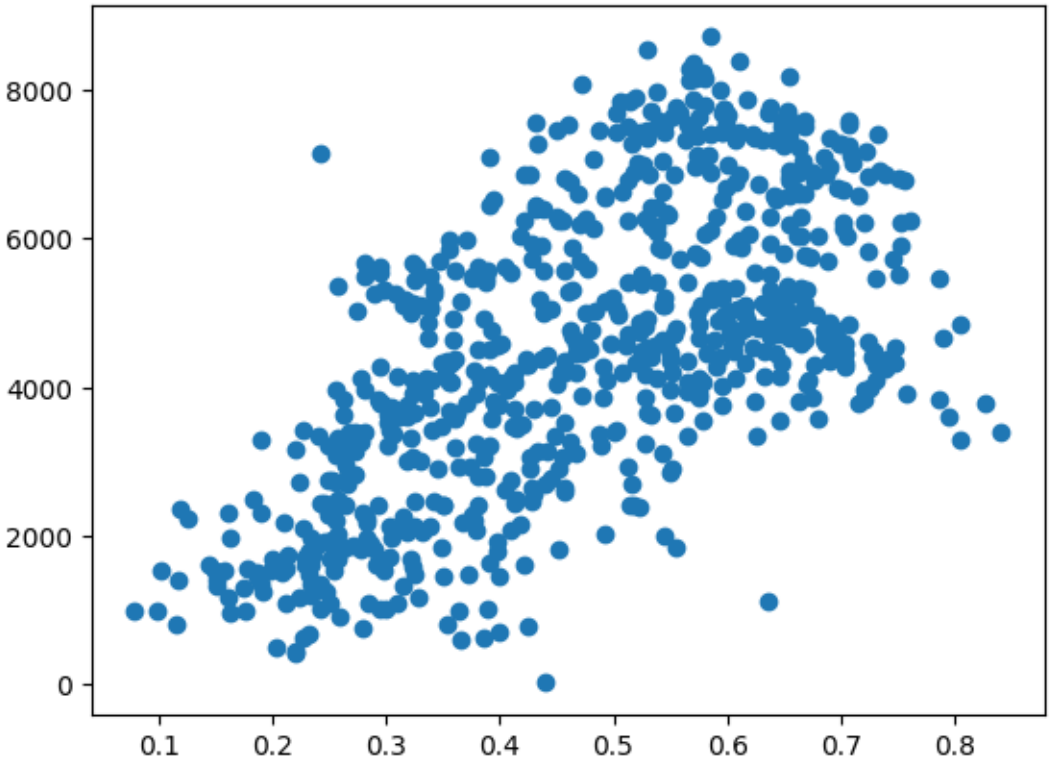
May 17, 2023

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

Part 1 - DEFINE —Step1.Define the problem——> load and plot the data by doing the regression with one variable, use only two fields the normalized high temperature in C, and the total number of bike rentals.

[2]:
```
df = pd.read_csv('/content/rental bike.txt')
```

[7]:
```
plt.scatter(df['atemp'], df['cnt'])
plt.show()
```



Part 2 - DISCOVER —-Step2.Load Dataset—->Check Head, info and describe , shape of dataset by query

[4]:
```
df.head()
```

[4]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	\
0	1	2011-01-01	1	0	1	0	6	0	
1	2	2011-01-02	1	0	1	0	0	0	
2	3	2011-01-03	1	0	1	0	1	1	
3	4	2011-01-04	1	0	1	0	2	1	
4	5	2011-01-05	1	0	1	0	3	1	

	weathersit	temp	atemp	hum	windspeed	casual	registered	\
0	2	0.344167	0.363625	0.805833	0.160446	331	654	
1	2	0.363478	0.353739	0.696087	0.248539	131	670	
2	1	0.196364	0.189405	0.437273	0.248309	120	1229	
3	1	0.200000	0.212122	0.590435	0.160296	108	1454	
4	1	0.226957	0.229270	0.436957	0.186900	82	1518	

	cnt
0	985
1	801
2	1349
3	1562
4	1600

[8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   instant     731 non-null    int64
 1   dteday      731 non-null    object
 2   season      731 non-null    int64
 3   yr          731 non-null    int64
 4   mnth        731 non-null    int64
 5   holiday     731 non-null    int64
 6   weekday     731 non-null    int64
 7   workingday  731 non-null    int64
 8   weathersit  731 non-null    int64
 9   temp        731 non-null    float64
 10  atemp       731 non-null    float64
 11  hum         731 non-null    float64
 12  windspeed   731 non-null    float64
 13  casual      731 non-null    int64
 14  registered  731 non-null    int64
```

```
 15  cnt         731 non-null    int64
dtypes: float64(4), int64(11), object(1)
memory usage: 91.5+ KB
```

[9]:
```
df.describe()
```

[9]:

	instant	season	yr	mnth	holiday	weekday
count	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000
mean	366.000000	2.496580	0.500684	6.519836	0.028728	2.997264
std	211.165812	1.110807	0.500342	3.451913	0.167155	2.004787
min	1.000000	1.000000	0.000000	1.000000	0.000000	0.000000
25%	183.500000	2.000000	0.000000	4.000000	0.000000	1.000000
50%	366.000000	3.000000	1.000000	7.000000	0.000000	3.000000
75%	548.500000	3.000000	1.000000	10.000000	0.000000	5.000000
max	731.000000	4.000000	1.000000	12.000000	1.000000	6.000000

	workingday	weathersit	temp	atemp	hum	windspeed
count	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000
mean	0.683995	1.395349	0.495385	0.474354	0.627894	0.190486
std	0.465233	0.544894	0.183051	0.162961	0.142429	0.077498
min	0.000000	1.000000	0.059130	0.079070	0.000000	0.022392
25%	0.000000	1.000000	0.337083	0.337842	0.520000	0.134950
50%	1.000000	1.000000	0.498333	0.486733	0.626667	0.180975
75%	1.000000	2.000000	0.655417	0.608602	0.730209	0.233214
max	1.000000	3.000000	0.861667	0.840896	0.972500	0.507463

	casual	registered	cnt
count	731.000000	731.000000	731.000000
mean	848.176471	3656.172367	4504.348837
std	686.622488	1560.256377	1937.211452
min	2.000000	20.000000	22.000000
25%	315.500000	2497.000000	3152.000000
50%	713.000000	3662.000000	4548.000000
75%	1096.000000	4776.500000	5956.000000
max	3410.000000	6946.000000	8714.000000

[5]:
```
df.shape
```

[5]:
```
(731, 16)
```

Part 3 – Clean the Dataset —-Step3.Clean Dataset— Check for null count column wise

[11]:
```
df.isnull().sum(axis=0)
```

[11]:
```
instant       0
dteday        0
season        0
yr            0
```

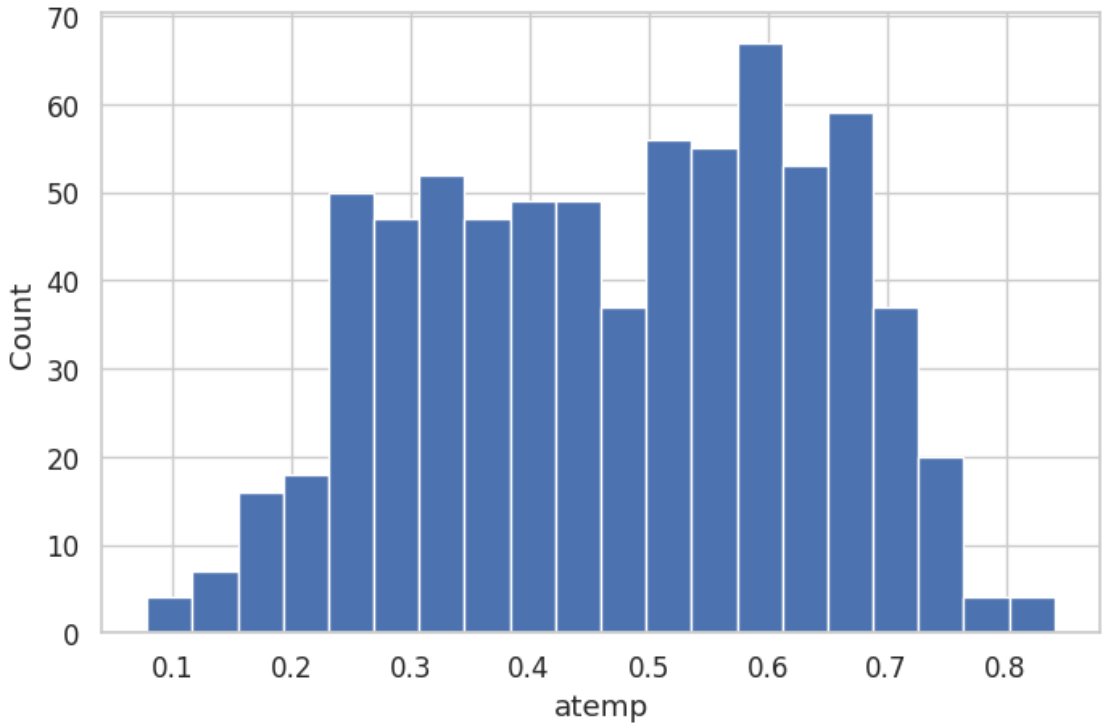
```
mnth          0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
casual        0
registered    0
cnt           0
dtype: int64
```

—Step4.Explore the Data (EDA)– a.Visualizing the Charges data Target Variable by using distplot

[27]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

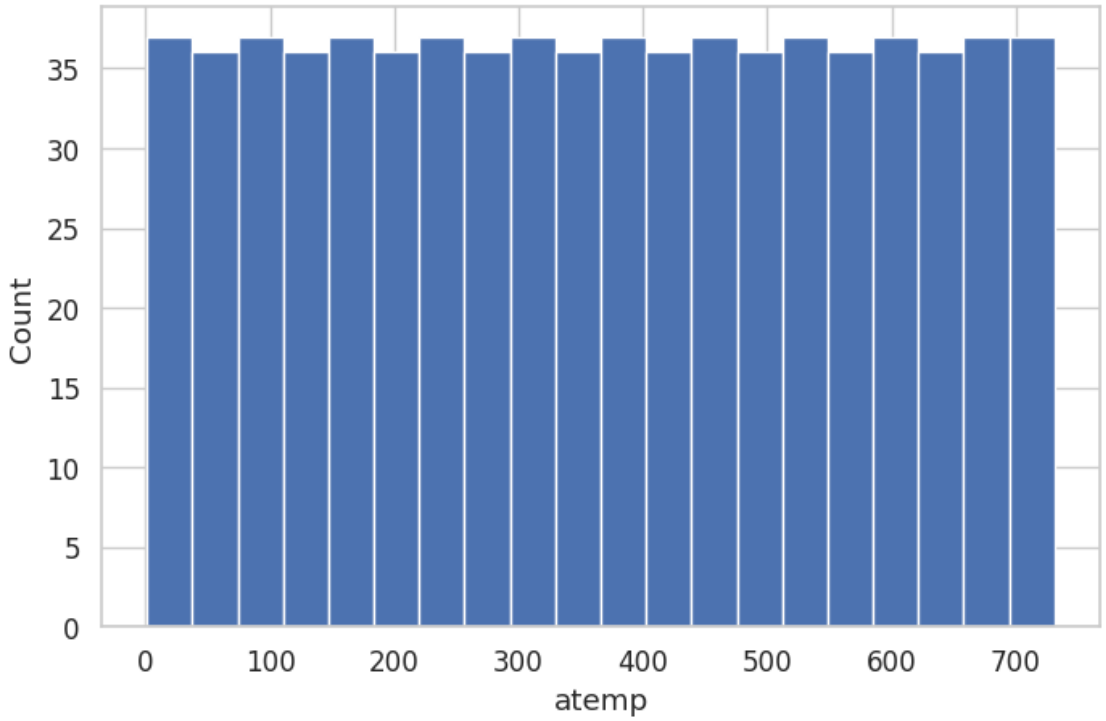
[31]:
```
sns.set(style='whitegrid', palette="deep", font_scale=1.1, rc={"figure.figsize":
 ↪ [8, 5]})
sns.distplot(
    df['atemp'], norm_hist=False, kde=False, bins=20, hist_kws={"alpha": 1}
).set(xlabel='atemp', ylabel='Count');
```

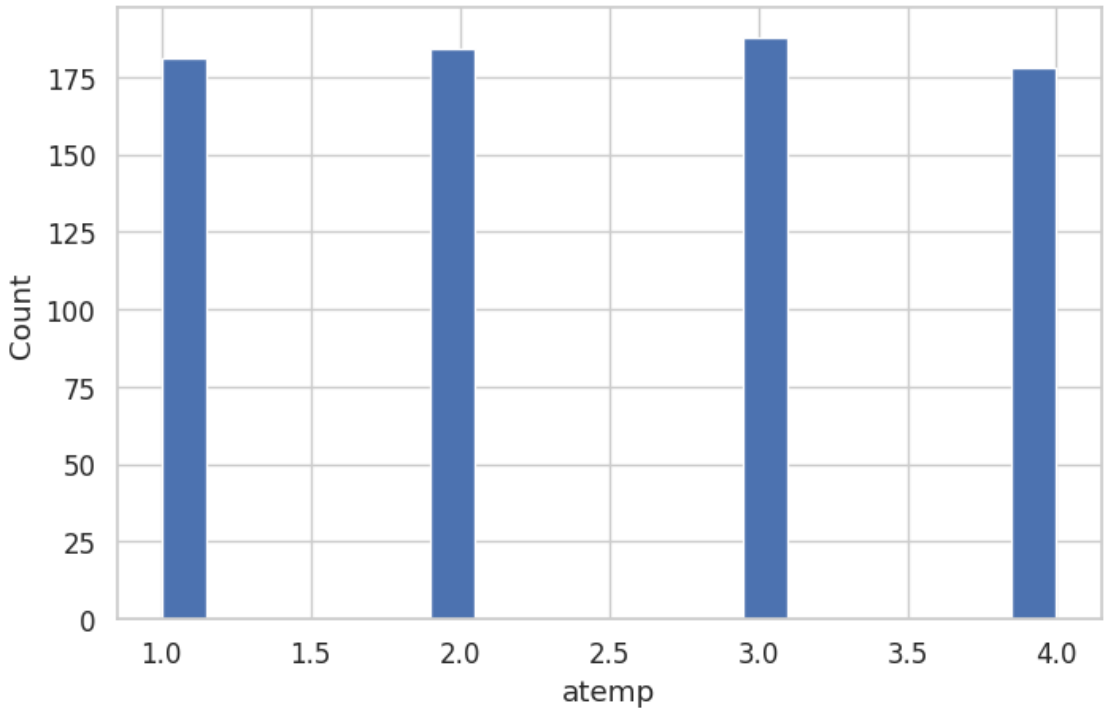
```
<ipython-input-31-5f7e9c00e574>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(
```

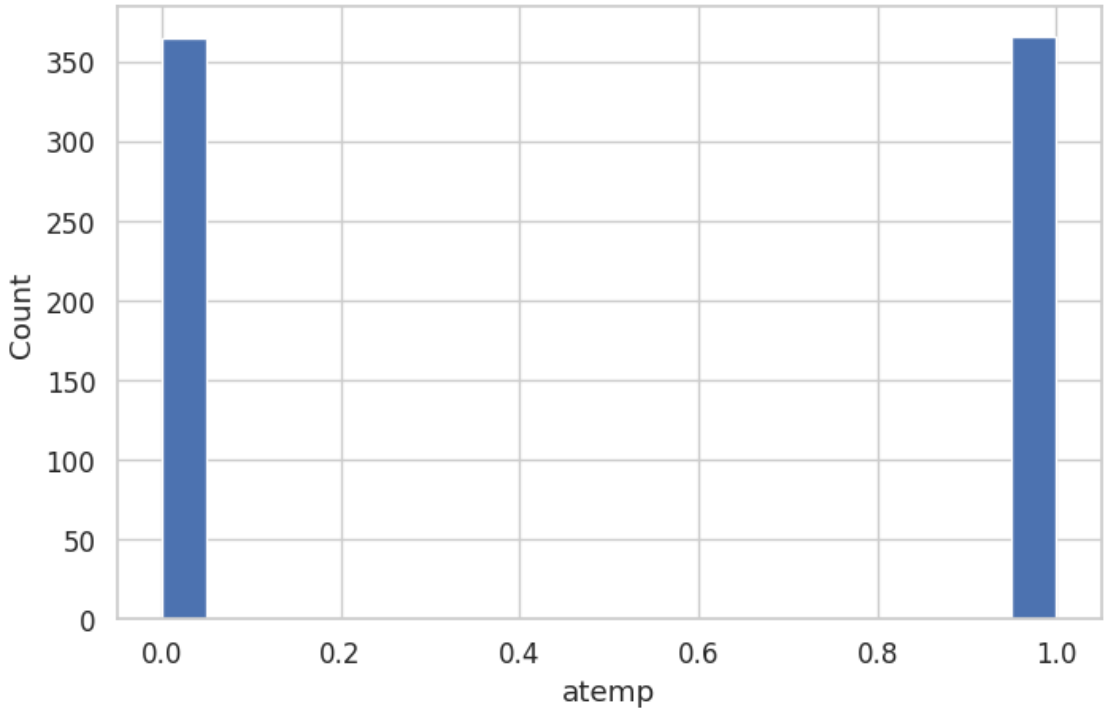


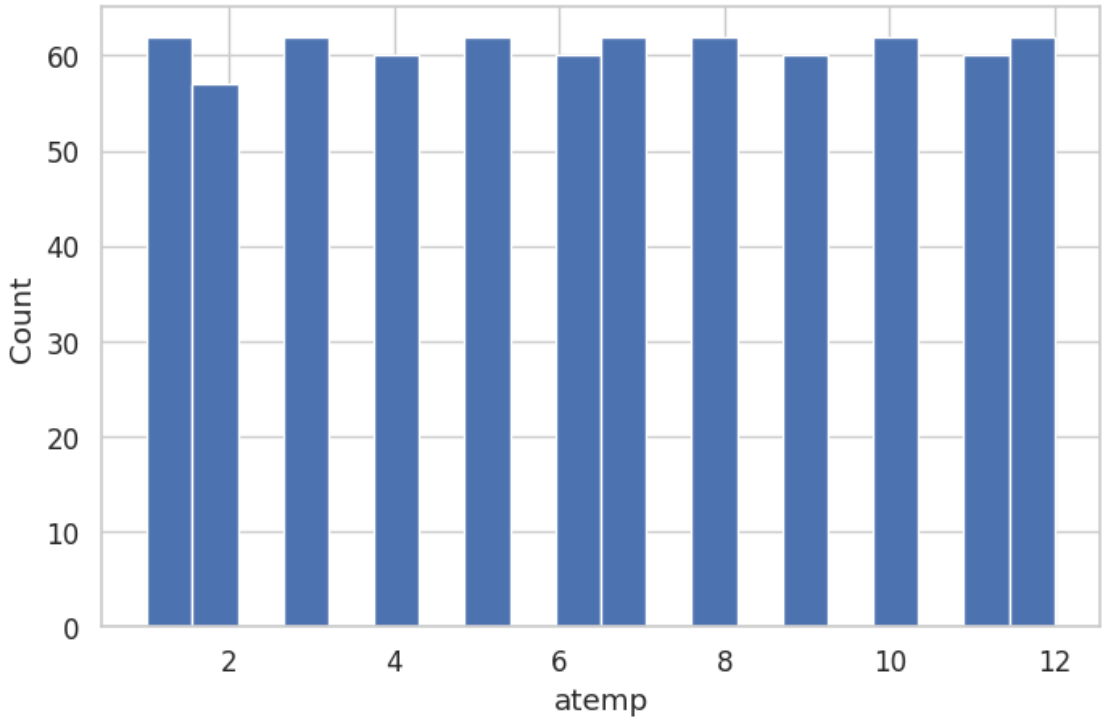
[29]:
```
for i in df.columns:
  if(df[i].dtypes=='int64'):
    sns.set(style='whitegrid', palette="deep", font_scale=1.1, rc={"figure.
↪figsize": [8, 5]})
    sns.distplot(
    df[i], norm_hist=False, kde=False, bins=20, hist_kws={"alpha": 1}
    ).set(xlabel='atemp', ylabel='Count');
    plt.show()
    print()
```

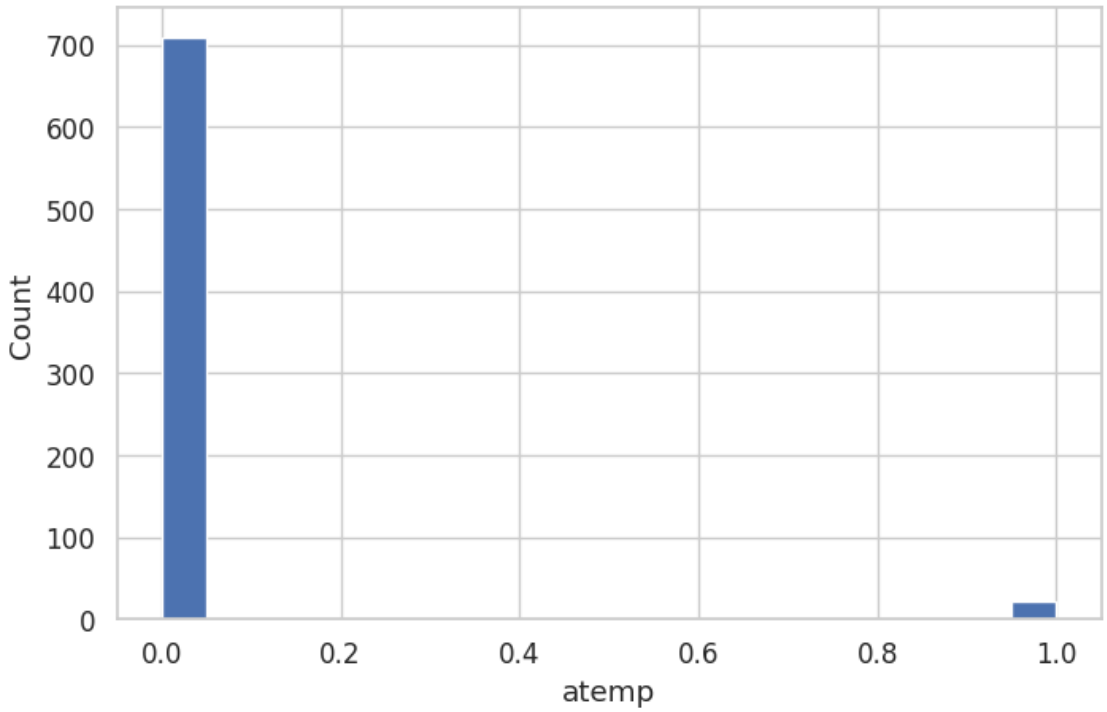
```
<ipython-input-29-591000bee4f2>:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(
```

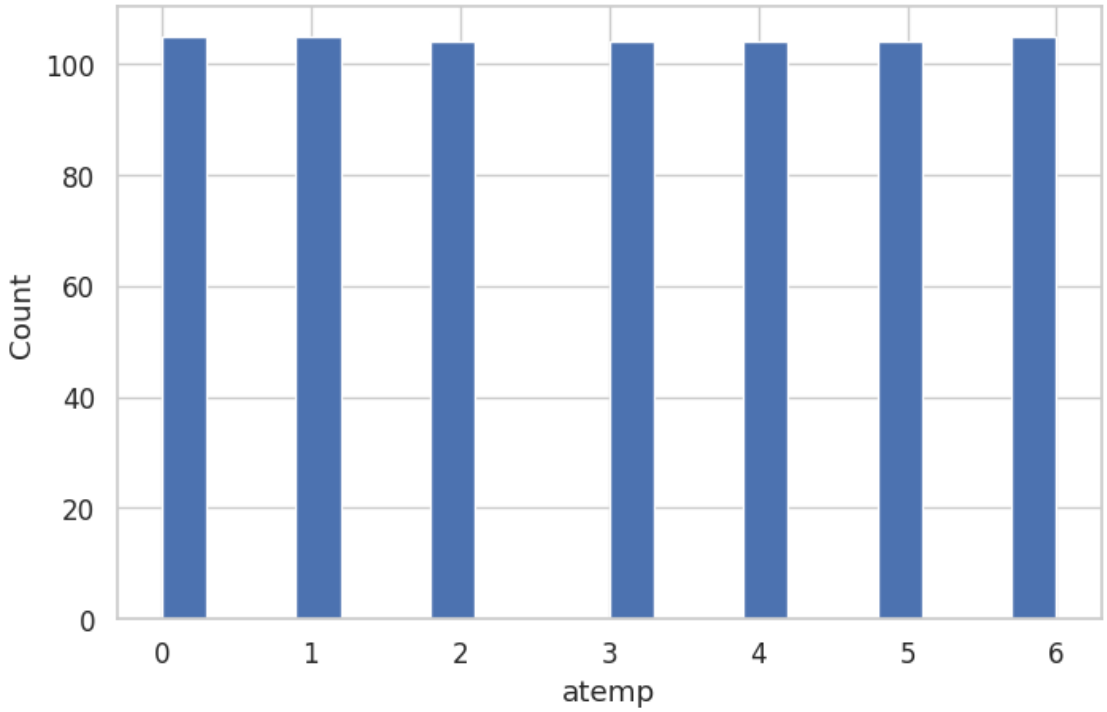

Count
35
30
25
20
15
10
5
0
0
100
200
300
400
500
600
700
atemp

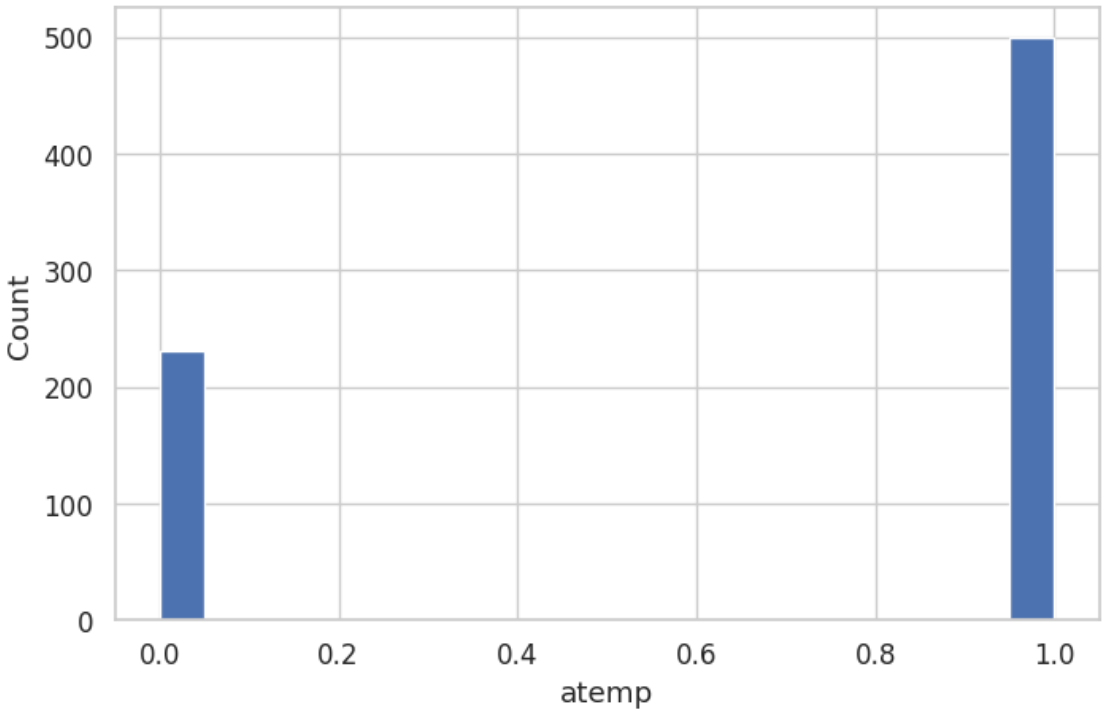

Count
175
150
125
100
75
50
25
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
atemp

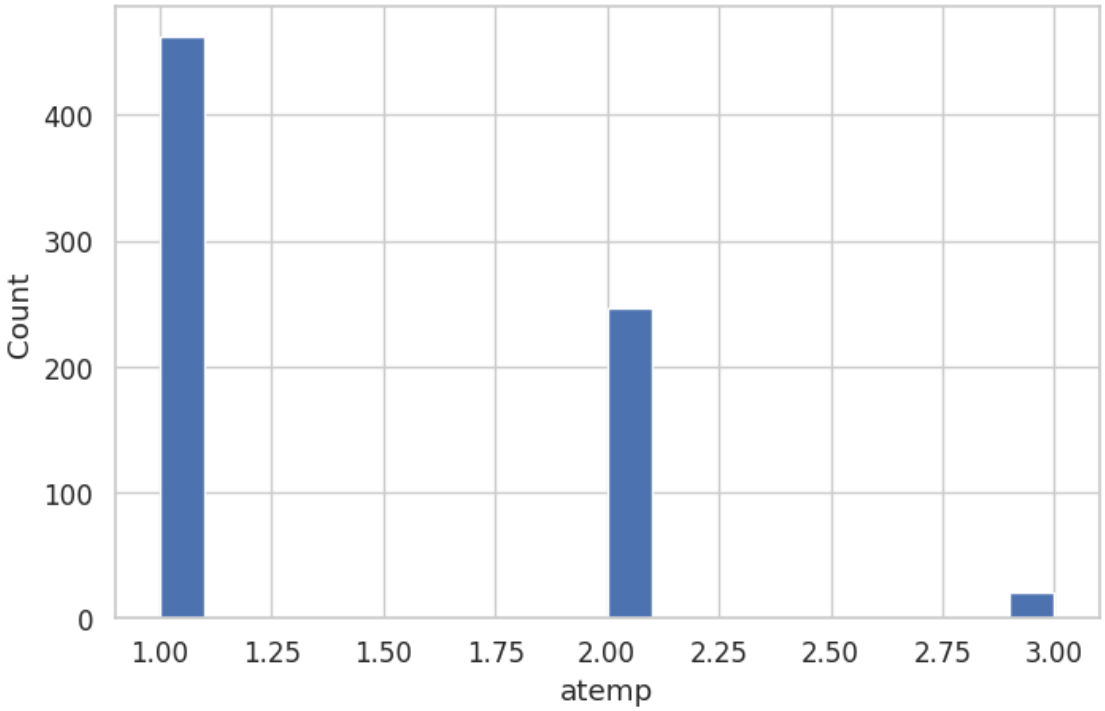

350
300
250
200
150
100
50
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
atemp

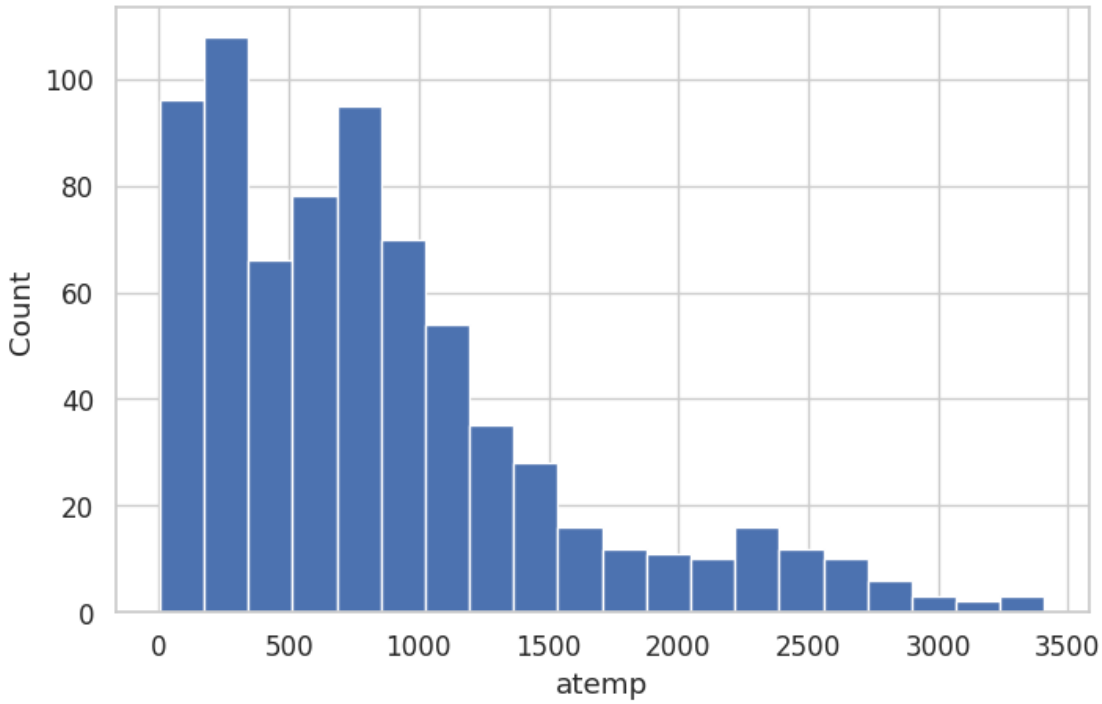

Count
60
50
40
30
20
10
0
2
4
6
8
10
12
atemp

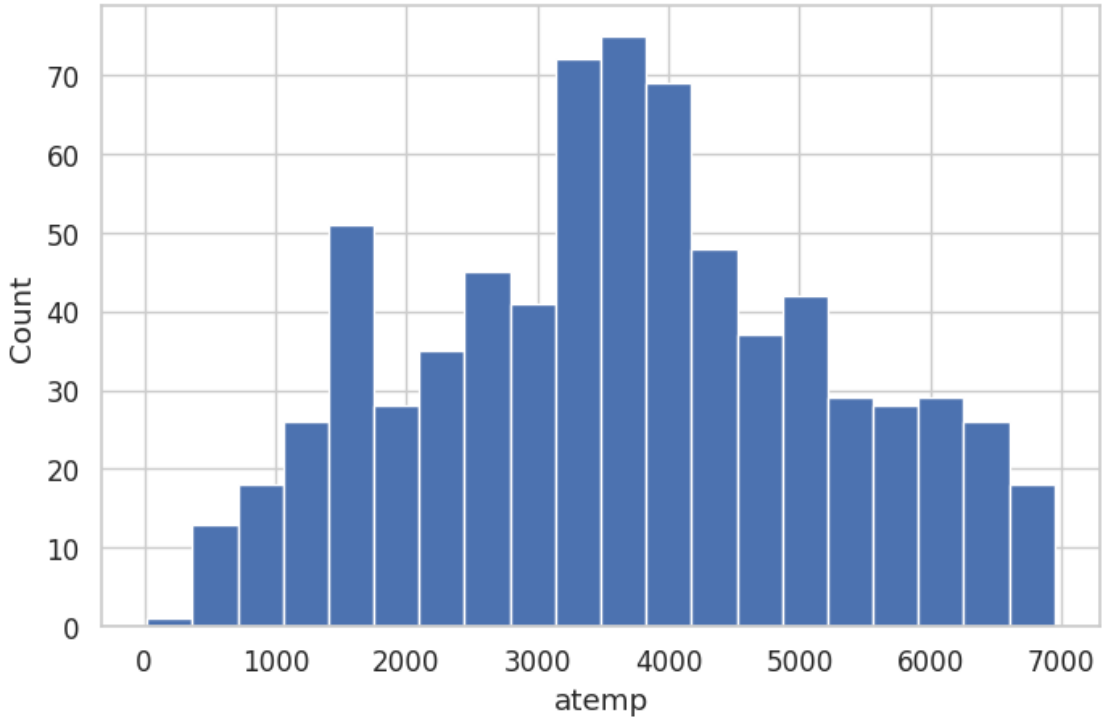

Count
700
600
500
400
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0
atemp

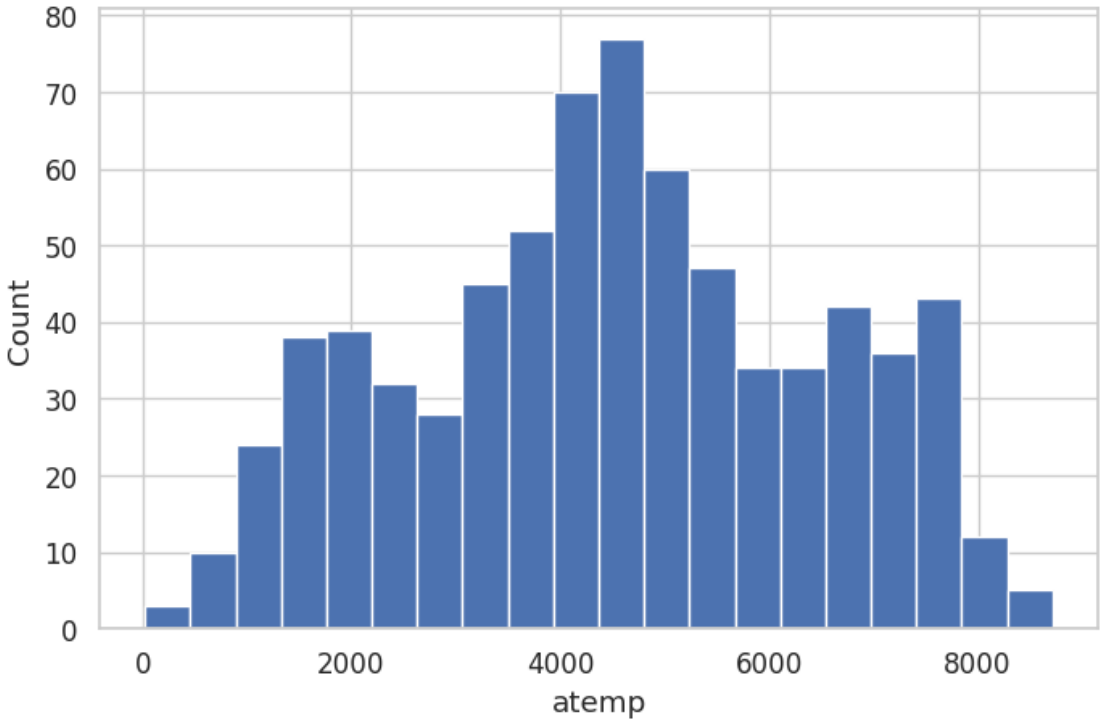

Count
100
80
60
40
20
0
0
1
2
3
4
5
6
atemp


500
400
300
200
100
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
atemp


400
300
200
100
0
Count
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
atemp

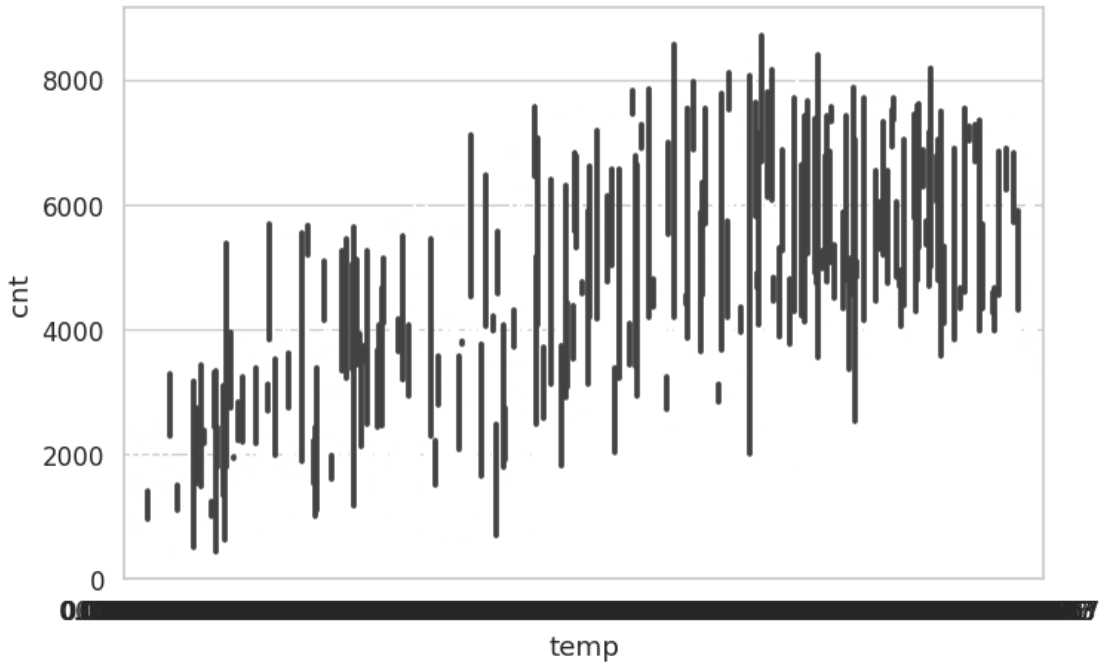

Count
100
80
60
40
20
0
0
500
1000
1500
2000
2500
3000
3500
atemp

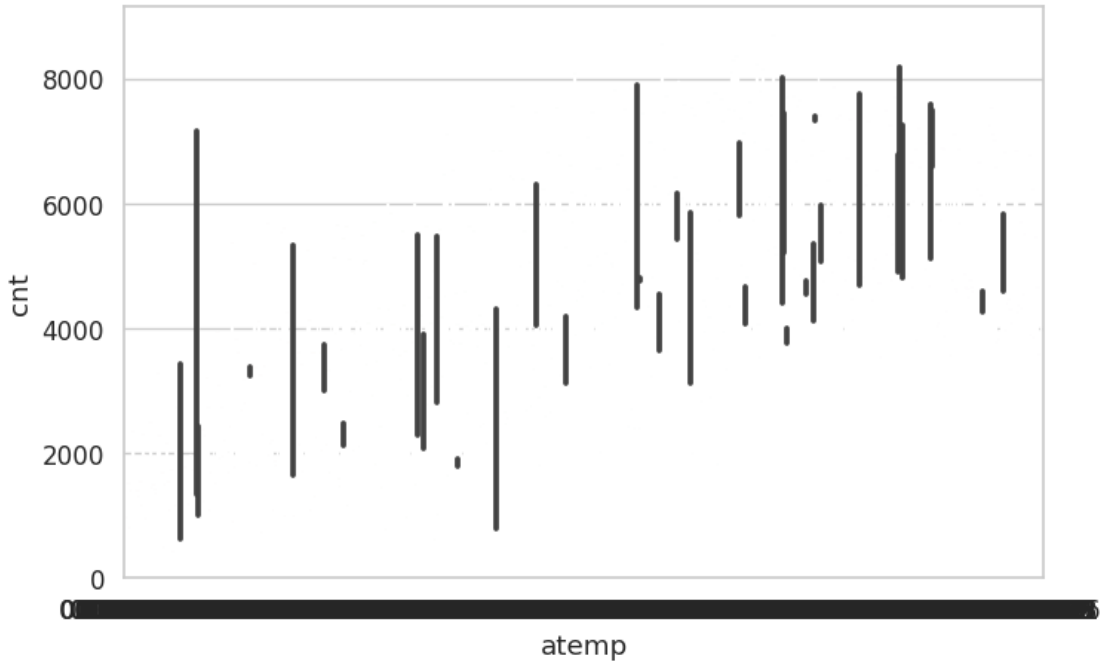

Count
70
60
50
40
30
20
10
0
0
1000
2000
3000
4000
5000
6000
7000
atemp

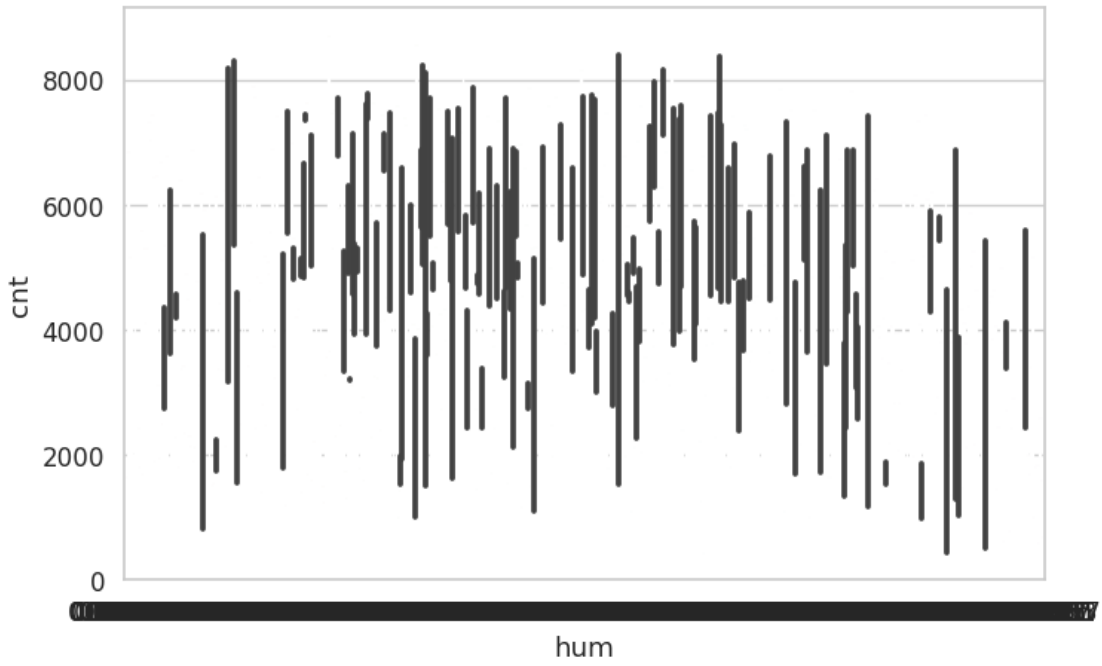


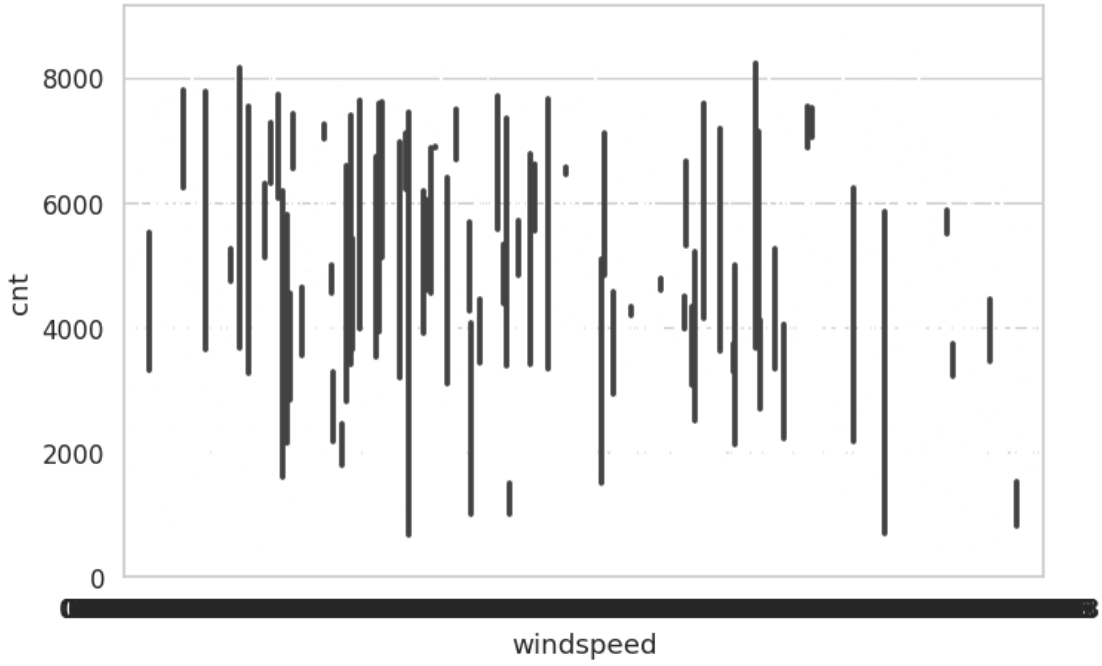
b.Visualizing the below data by using bar plot temp atemp humidity windspeed

[34]:
```
lis=['temp','atemp','hum','windspeed']
for i in lis:
  sns.barplot(x=i, y='cnt', data=df)
  plt.show()
```


8000
6000
4000
2000
0
cnt
temp

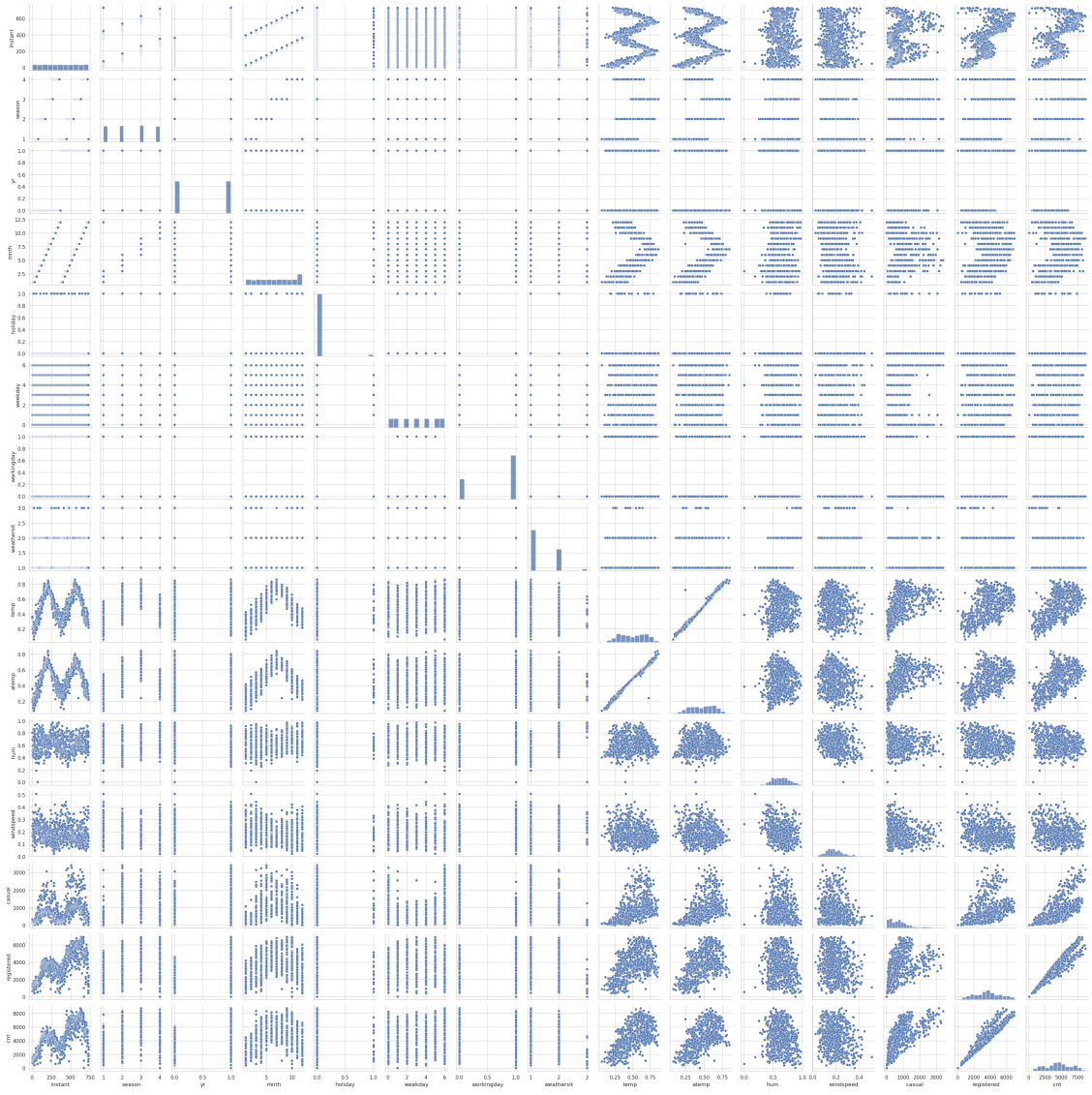

8000
6000
4000
2000
0
cnt
atemp



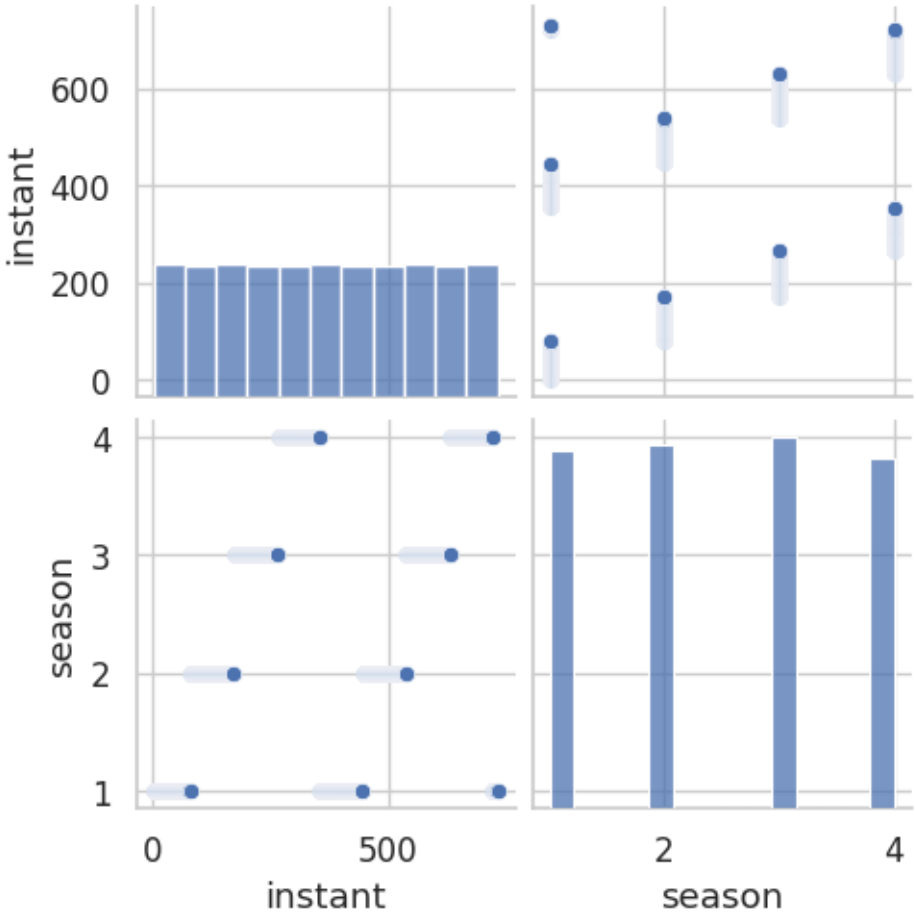


c.Visualizing any 2 Numerical data by using pairplot

[38]:
```
sns.pairplot(df)
plt.show()
```



[41]:
```
df3=df.iloc[:, :3]
sns.pairplot(df3)
plt.show()
```



Plot a heatmap and look at the correlation

[45]:
```
df.corr()
```

```
<ipython-input-45-2f6f6606aa2c>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  df.corr()
```

[45]:

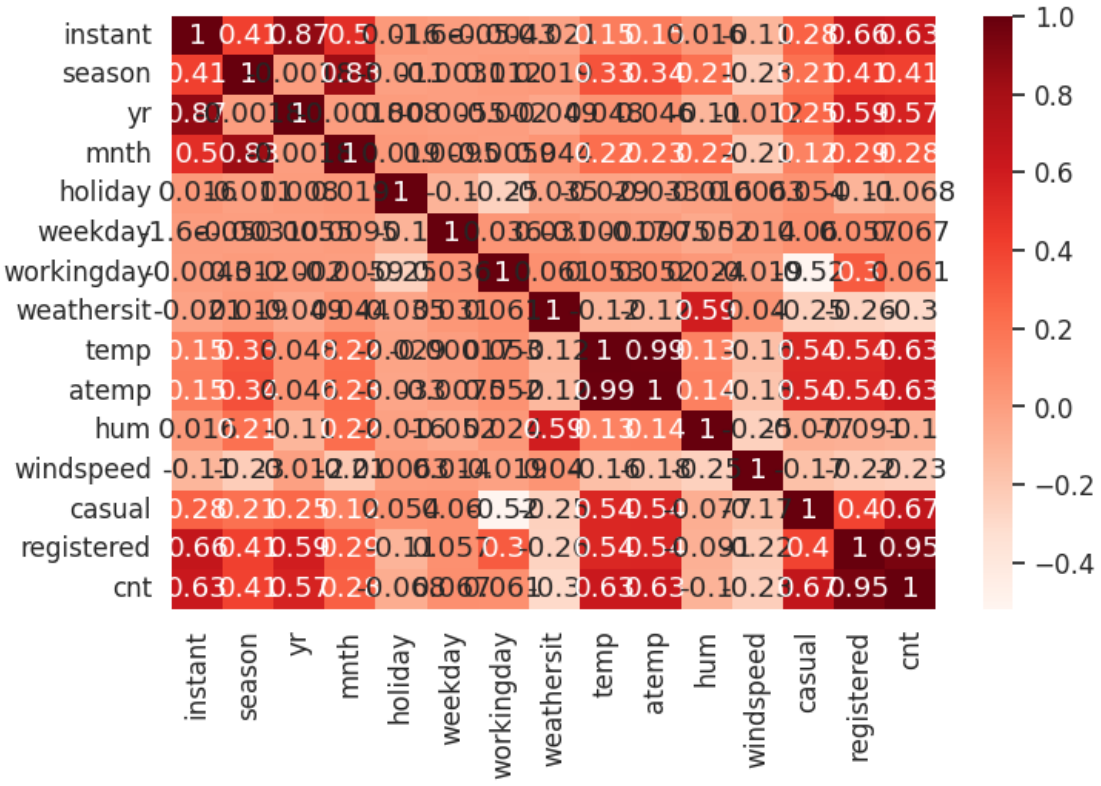
	instant	season	yr	mnth	holiday	weekday	\
instant	1.000000	0.412224	0.866025	0.496702	0.016145	-0.000016	
season	0.412224	1.000000	-0.001844	0.831440	-0.010537	-0.003080	
yr	0.866025	-0.001844	1.000000	-0.001792	0.007954	-0.005461	
mnth	0.496702	0.831440	-0.001792	1.000000	0.019191	0.009509	
holiday	0.016145	-0.010537	0.007954	0.019191	1.000000	-0.101960	
weekday	-0.000016	-0.003080	-0.005461	0.009509	-0.101960	1.000000	
workingday	-0.004337	0.012485	-0.002013	-0.005901	-0.253023	0.035790	

```
weathersit -0.021477  0.019211 -0.048727  0.043528 -0.034627  0.031087
temp        0.150580  0.334315  0.047604  0.220205 -0.028556 -0.000170
atemp       0.152638  0.342876  0.046106  0.227459 -0.032507 -0.007537
hum         0.016375  0.205445 -0.110651  0.222204 -0.015937 -0.052232
windspeed  -0.112620 -0.229046 -0.011817 -0.207502  0.006292  0.014282
casual      0.275255  0.210399  0.248546  0.123006  0.054274  0.059923
registered  0.659623  0.411623  0.594248  0.293488 -0.108745  0.057367
cnt         0.628830  0.406100  0.566710  0.279977 -0.068348  0.067443

            workingday  weathersit      temp     atemp       hum  windspeed  \
instant      -0.004337   -0.021477  0.150580  0.152638  0.016375  -0.112620
season        0.012485    0.019211  0.334315  0.342876  0.205445  -0.229046
yr           -0.002013   -0.048727  0.047604  0.046106 -0.110651  -0.011817
mnth         -0.005901    0.043528  0.220205  0.227459  0.222204  -0.207502
holiday      -0.253023   -0.034627 -0.028556 -0.032507 -0.015937   0.006292
weekday       0.035790    0.031087 -0.000170 -0.007537 -0.052232   0.014282
workingday    1.000000    0.061200  0.052660  0.052182  0.024327  -0.018796
weathersit    0.061200    1.000000 -0.120602 -0.121583  0.591045   0.039511
temp          0.052660   -0.120602  1.000000  0.991702  0.126963  -0.157944
atemp         0.052182   -0.121583  0.991702  1.000000  0.139988  -0.183643
hum           0.024327    0.591045  0.126963  0.139988  1.000000  -0.248489
windspeed    -0.018796    0.039511 -0.157944 -0.183643 -0.248489   1.000000
casual       -0.518044   -0.247353  0.543285  0.543864 -0.077008  -0.167613
registered    0.303907   -0.260388  0.540012  0.544192 -0.091089  -0.217449
cnt           0.061156   -0.297391  0.627494  0.631066 -0.100659  -0.234545

              casual  registered       cnt
instant     0.275255    0.659623  0.628830
season      0.210399    0.411623  0.406100
yr          0.248546    0.594248  0.566710
mnth        0.123006    0.293488  0.279977
holiday     0.054274   -0.108745 -0.068348
weekday     0.059923    0.057367  0.067443
workingday -0.518044    0.303907  0.061156
weathersit -0.247353   -0.260388 -0.297391
temp        0.543285    0.540012  0.627494
atemp       0.543864    0.544192  0.631066
hum        -0.077008   -0.091089 -0.100659
windspeed  -0.167613   -0.217449 -0.234545
casual      1.000000    0.395282  0.672804
registered  0.395282    1.000000  0.945517
cnt         0.672804    0.945517  1.000000
```

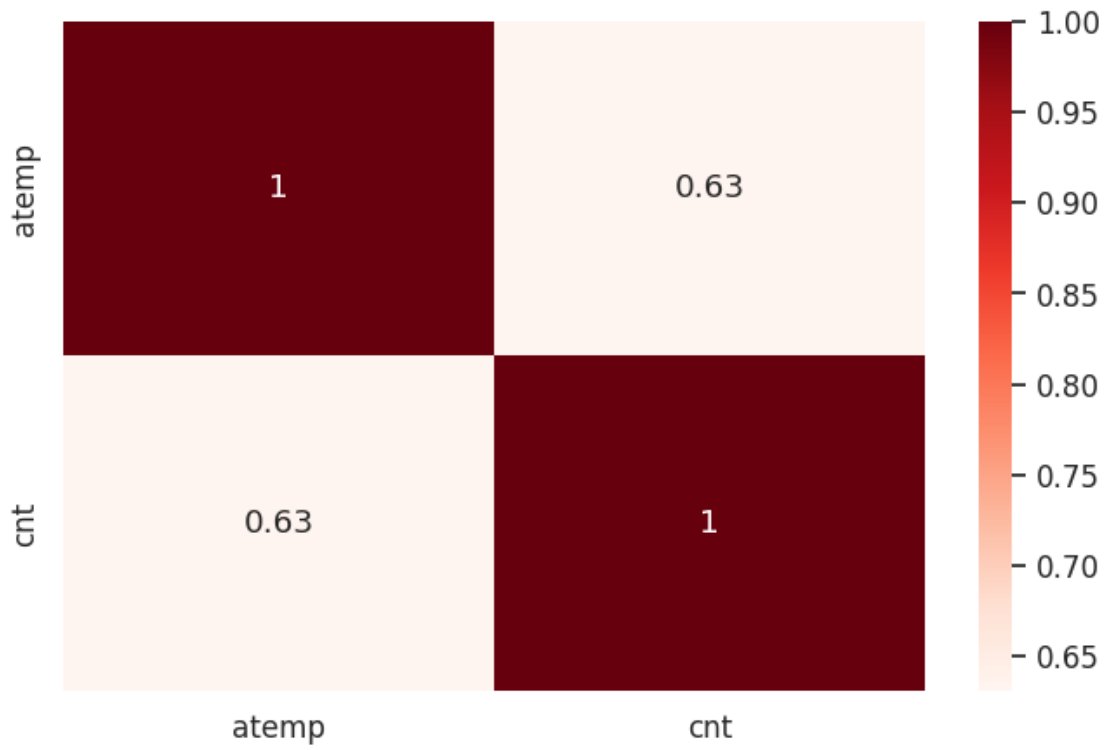
[42]:
```
sns.heatmap(df.corr(), annot=True, cmap = 'Reds')
plt.show()
```

```
<ipython-input-42-5183f45435d6>:1: FutureWarning: The default value of
```

```
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  sns.heatmap(df.corr(), annot=True, cmap = 'Reds')
```



[44]:
```
sns.heatmap(df[['atemp','cnt']].corr(), annot=True, cmap = 'Reds')
plt.show()
```



Split the data into train and test

[72]:
```
X = df[['atemp']]
y = df['cnt']
```

[73]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
↪random_state=42)
```

[74]:
```
print(X_train)
```

```
        atemp
682  0.323225
250  0.555361
336  0.310604
260  0.490537
543  0.640792
..        …
71   0.380091
106  0.445696
270  0.575158
435  0.359670
102  0.417283
```

```
[584 rows x 1 columns]
```

[75]:
```
print(y_train)
```

```
682    4094
250    1842
336    3614
260    4274
543    7335
       …
71     2417
106    3744
270    3907
435    4911
102    2162
Name: cnt, Length: 584, dtype: int64
```

Training a Linear Regression Model

[76]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

[76]:
```
LinearRegression()
```

[77]:
```
regressor.intercept_
```

[77]:
```
1011.2925446276845
```

[78]:
```
regressor.coef_
```

[78]:
```
array([7417.73429531])
```

[79]:
```
print(regressor.coef_[0])
```

```
7417.734295310026
```

Making Predictions

Visualize the Actual Vs Predicted

[80]:
```
def calc(slope, intercept, xyz):
    return slope*xyz+intercept
```

[81]:
```
score = calc(regressor.coef_, regressor.intercept_, 9.5)
print(score)
```

```
[71479.76835007]
```

[82]:
```
y_pred = regressor.predict(X_test)
y_pred
```

[82]:
```
array([4490.61048678, 2330.74428561, 3376.15525078, 4106.61663551,
       5047.86777572, 5834.83004257, 2081.54550196, 3811.58367165,
       5965.8420657 , 6003.3683835 , 2729.98158085, 4172.14490028,
       5399.22359608, 3160.67006951, 3254.41539553, 3427.61949132,
       3473.98033067, 5314.96555222, 5848.85697813, 2425.58001857,
       5726.99102139, 6846.17877186, 3886.43602843, 2780.13288242,
       3156.43454322, 5689.59822281, 3518.03425465, 4738.80045857,
       2954.64991719, 4050.39762729, 2889.12165242, 4940.2809575 ,
       4499.9790852 , 5235.26199722, 4247.15302947, 4621.74861139,
       5811.30840712, 5600.6150822 , 4204.90161493, 2431.52162374,
       5197.80243903, 2903.12633477, 5961.20598176, 5586.58072891,
       6129.65530987, 3980.21102539, 6232.82115845, 4420.42388488,
       6050.15945143, 4256.46228601, 5652.16833555, 2832.93231514,
       3230.99019062, 6415.38643493, 5801.93980871, 5497.59758831,
       4884.27706357, 4335.95072672, 5577.27889011, 6373.32788148,
       4963.61714959, 5422.64880099, 3418.19155104, 5595.97899826,
       5432.0173994 , 6359.29352819, 6443.57382525, 6424.72536241,
       3820.97452327, 5188.40416967, 4471.85103675, 3914.62341875,
       3933.32352691, 4701.34090038, 3633.64706138, 4668.63610987,
       3099.88173696, 4888.69803321, 1886.06595007, 6406.05492519,
       2692.46268079, 2921.90803801, 3413.65189765, 4270.64499399,
       3469.77447533, 3071.69434663, 5197.95079371, 3774.18345534,
       3366.73472823, 3095.17889341, 2392.84555713, 5951.87447202,
       3647.65174373, 4537.46831432, 2903.15600571, 3034.29413032,
       5477.73289586, 1896.50270223, 4724.82544715, 5881.65078145,
       5853.43372019, 6317.13854419, 5904.92763166, 2921.8857848 ,
       4050.59790612, 3549.35934658, 3441.587085  , 2861.06036358,
       6494.94905298, 2750.97376891, 6382.62972028, 4223.69815363,
       3919.38560417, 5254.02144725, 4818.50401357, 4560.76741775,
       3610.9191235 , 3076.47878525, 5745.75788916, 3109.18357576,
       4214.32955522, 4364.0194333 , 4920.50527786, 2416.24850883,
       4646.88731291, 3717.96444711, 6555.89315795, 3287.17952791,
       3385.49417826, 5633.46822739, 3394.82568801, 4022.30666751,
       3099.85206602, 3216.86682453, 5506.93651578, 3113.87900157,
       4401.47899149, 4631.1172098 , 3273.7015047 , 6228.12573264,
       4893.46021862, 3966.14700116, 3427.61949132, 6495.03806579,
       3835.00887656, 5371.09554763, 5094.84428701])
```

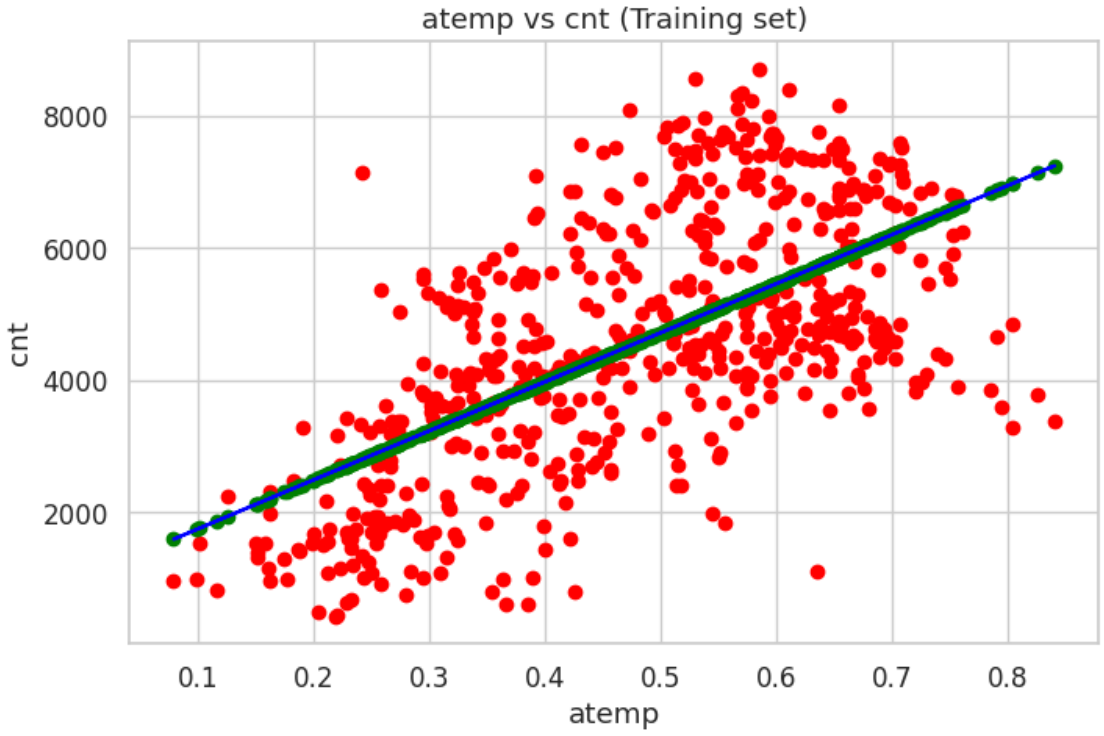
[83]:
```
df_preds = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
print(df_preds)
```

```
     Actual    Predicted
703    6606  4490.610487
33     1550  2330.744286
```

```
300     3747  3376.155251
456     6041  4106.616636
633     7538  5047.867776
..       …          …
70      2132  3427.619491
192     4258  6495.038066
328     2792  3835.008877
165     5180  5371.095548
135     3958  5094.844287

[147 rows x 2 columns]
```

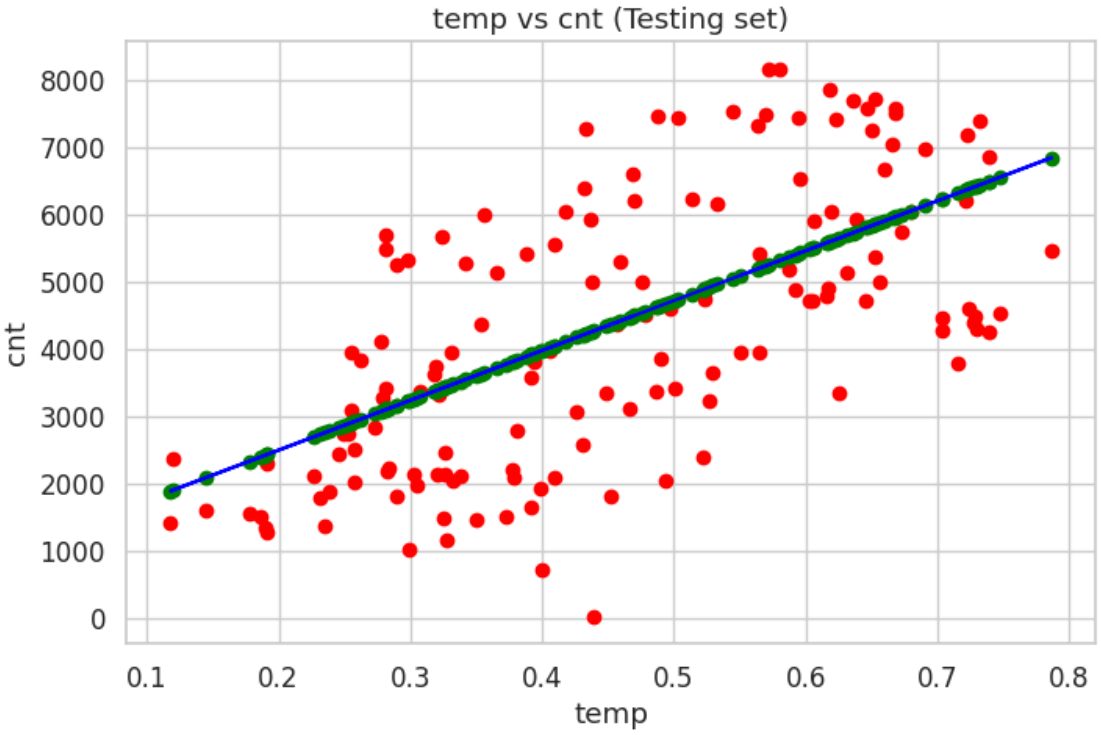
Visualize

[84]:
```
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.scatter(X_train, regressor.predict(X_train), color = 'green')
plt.title('atemp vs cnt (Training set)')
plt.xlabel('atemp')
plt.ylabel('cnt')
plt.show()
```



[85]:
```
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_test, regressor.predict(X_test), color = 'blue')
```

```
plt.scatter(X_test, regressor.predict(X_test), color = 'green')
plt.title('temp vs cnt (Testing set)')
plt.xlabel('temp')
plt.ylabel('cnt')
plt.show()
```



Using from sklearn.metrics import mean_absolute_error, mean_squared_error

[86]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
```

[87]:
```
mae
```

[87]: 1282.3164895854777

[88]:
```
mse
```

[88]: 2359187.3227619664

[89]:
```
rmse
```

[89]: 1535.9646228875085

[ ]: