Airfoil_Self_Noise Data Set

This dataset contains the hourly and daily count of rental bikes between years 2011 and 2012 in Capital bikeshare system with the corresponding weather and seasonal information.

Lets get started!

Loading the data

In [1]:
```
import pandas as pd;
import numpy as np;
import matplotlib.pyplot as plt;
import seaborn as sns;
%matplotlib inline
```

In [2]:
```
df = pd.read_csv('day.csv')
```

In [3]:
```
df.head(4)
```

Out[3]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000

Data Check

Check if any kind of data processing required

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 16 columns):
instant       731 non-null int64
dteday        731 non-null object
season        731 non-null int64
yr            731 non-null int64
mnth          731 non-null int64
holiday       731 non-null int64
weekday       731 non-null int64
workingday    731 non-null int64
weathersit    731 non-null int64
temp          731 non-null float64
atemp         731 non-null float64
hum           731 non-null float64
windspeed     731 non-null float64
casual        731 non-null int64
registered    731 non-null int64
cnt           731 non-null int64
dtypes: float64(4), int64(11), object(1)
memory usage: 91.5+ KB
```

In [4]:
```
df.describe()
```

Out[4]:

	instant	season	yr	mnth	holiday	weekday	working
count	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000
mean	366.000000	2.496580	0.500684	6.519836	0.028728	2.997264	0.68399
std	211.165812	1.110807	0.500342	3.451913	0.167155	2.004787	0.46523
min	1.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.00000
25%	183.500000	2.000000	0.000000	4.000000	0.000000	1.000000	0.00000
50%	366.000000	3.000000	1.000000	7.000000	0.000000	3.000000	1.00000
75%	548.500000	3.000000	1.000000	10.000000	0.000000	5.000000	1.00000
max	731.000000	4.000000	1.000000	12.000000	1.000000	6.000000	1.00000

In [8]:
```
df.isnull().T.any().T.any()
```

Out[8]: False

In [9]:
```
df.isnull().sum()
```

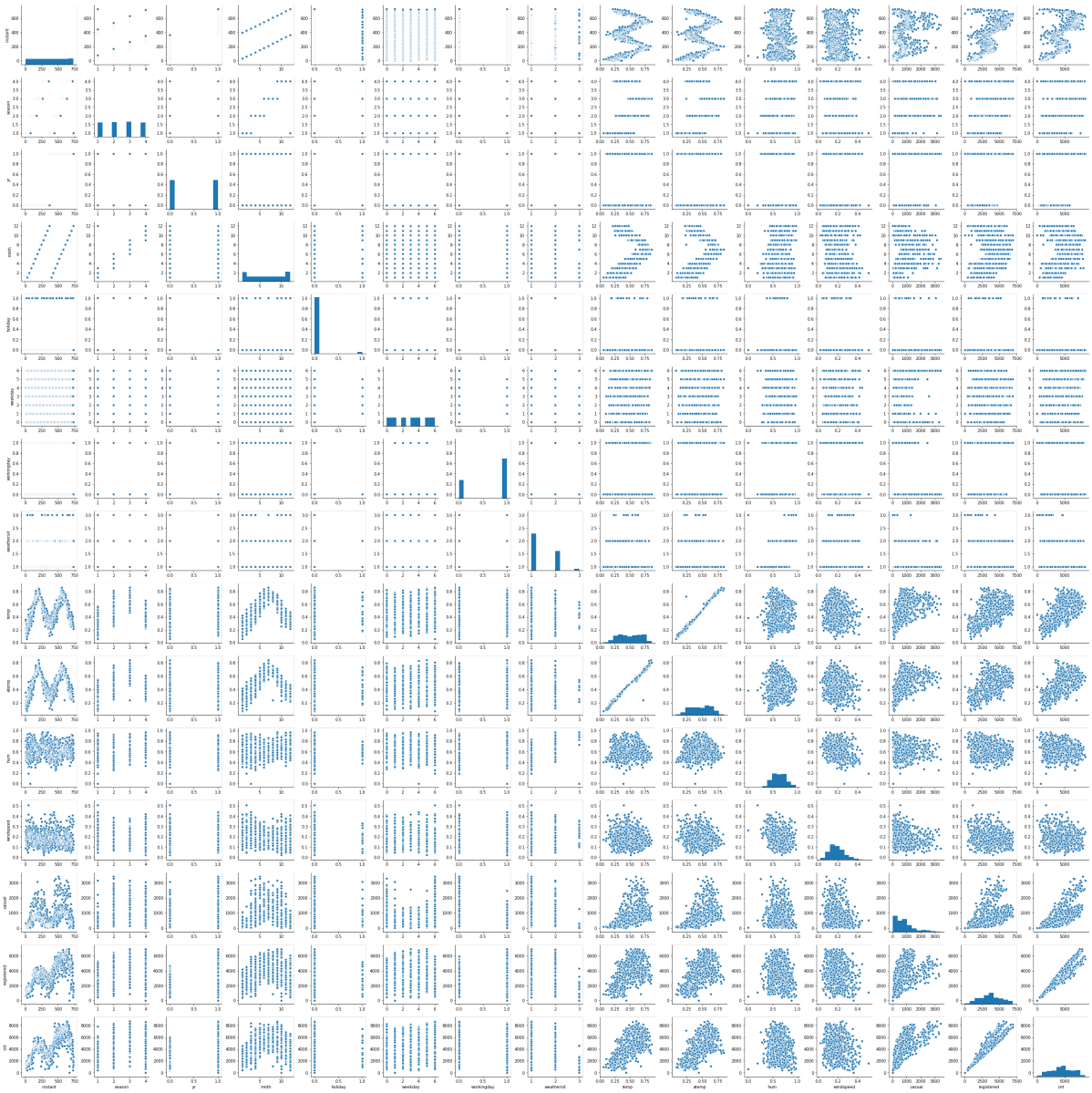
Out[9]:
```
instant       0
dteday        0
season        0
yr            0
mnth          0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
casual        0
registered    0
cnt           0
dtype: int64
```

Exploratory Data Analysis

In [8]:
```
plt.figure(figsize=(15,8))
sns.pairplot(df)
```

Out[8]: <seaborn.axisgrid.PairGrid at 0x185eaf60>

<matplotlib.figure.Figure at 0x185eaef0>

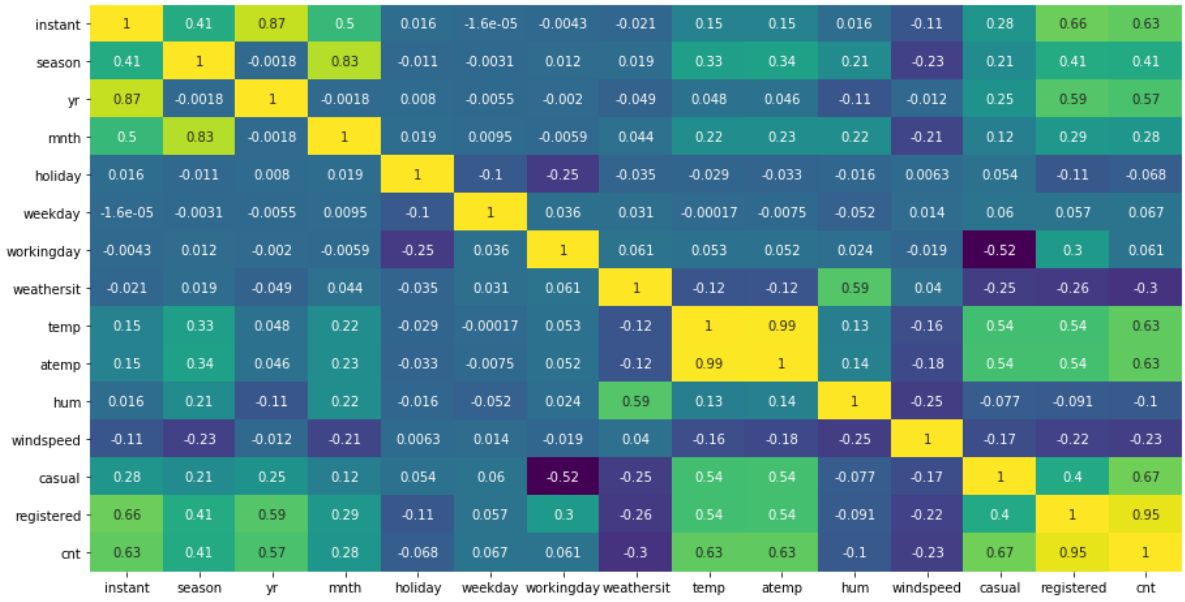


In [10]:
```
sns.heatmap(df.isnull(), cbar=False, cmap='viridis', yticklabels=False,xticklabels=False)
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x23340470>

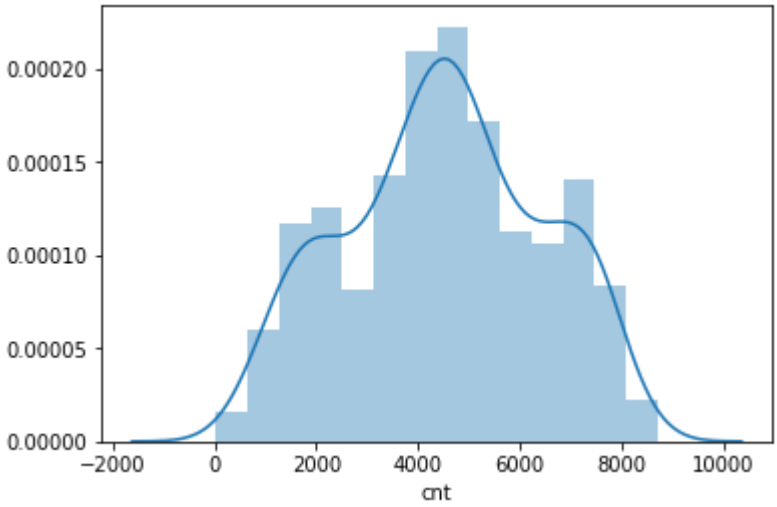
In [16]:
```
plt.figure(figsize=(15,8))
sns.heatmap(df.corr(), annot=True, cmap='viridis',cbar=False)
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x2486c160>



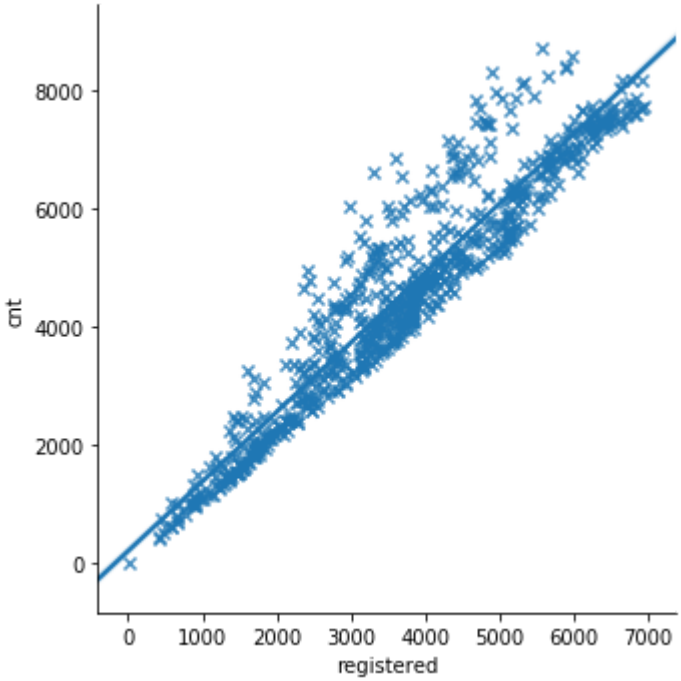
In [10]:
```
sns.distplot(df.cnt)
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0xbd417f0>



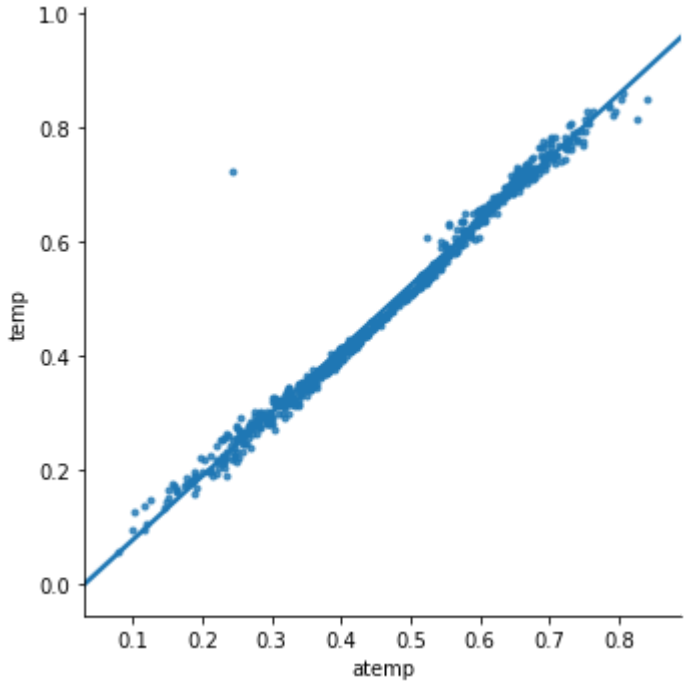
In [12]:
```
sns.lmplot(x='registered',y='cnt', data=df, markers='x')
```

Out[12]: <seaborn.axisgrid.FacetGrid at 0xbe26a20>



In [15]:
```
sns.lmplot(x='atemp',y='temp', data=df, markers='.', )
```

Out[15]: <seaborn.axisgrid.FacetGrid at 0xc705358>



Here Linear Relationship is found between few independent variables. E.g. Registered users(Dependent Variable) Vs Count. Atemp Vs temp. etc. Hence moving ahead with the Linear Regression

Training and Testing Data

Now that we've explored the data a bit, let's go ahead and split the data into training and testing sets.

In [16]:
```
df.head()
```

Out[16]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957

In [17]:
```
df.drop('dteday', axis=1, inplace=True)
```

In [18]:
```
X = df.drop('cnt', axis=1)
y = df['cnt']
```

Training the Model

Now its time to train our model on our training data!

Import LinearRegression from sklearn.linear_model

In [19]:
```
from sklearn.model_selection import train_test_split
```

In [20]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_
state=101)
```

In [21]:
```
from sklearn.linear_model import LinearRegression
```

In [22]:
```
lr = LinearRegression()
```

In [23]:
```
lr.fit(X_train, y_train)
```

Out[23]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

Predicting the Variables

In [24]:
```
pred = lr.predict(X_test)
```

Evaluating the Model

Now evaluating the model with r2 score and coefficients and errors

R2 Score

In [25]:
```
from sklearn import metrics
```

In [26]:
```
print(metrics.r2_score(y_test, pred))
```

```
1.0
```

Coefficients

In [27]:
```
coefficients = pd.DataFrame(data=lr.coef_, index=df.columns[:-1],columns=['Coe
fficients'])
```

In [29]:
```
coefficients
```

Out[29]:

	Coefficients
instant	-1.007784e-14
season	4.449774e-13
yr	3.440248e-12
mnth	2.684519e-13
holiday	1.795068e-12
weekday	-5.992678e-14
workingday	-5.828836e-13
weathersit	7.207646e-13
temp	-2.160196e-13
atemp	-2.122852e-12
hum	1.331753e-12
windspeed	1.820089e-12
casual	1.000000e+00
registered	1.000000e+00

Interpreting the coefficients:

- Holding all other features fixed, 1 unit increase in count('cnt') is associated with **increase of 1 registered user**.

Calculating Errors

In [30]:
```
print('MAE: ', metrics.mean_absolute_error(y_test, pred))
print('MSE: ',metrics.mean_squared_error(y_test, pred))
print('RMSE: ',np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

```
MAE:  2.33419748177e-12
MSE:  8.25400529275e-24
RMSE:  2.87297847064e-12
```

Plotting the Test Values Vs Predicted Values

In [32]:
```
plt.title('Y Test Values Vs Predictions')
plt.xlabel('Y Test Values')
plt.ylabel('Predicted Values')
plt.scatter(y_test, pred)
```

Out[32]: <matplotlib.collections.PathCollection at 0xd62f8d0>