Appliances_Energy Data Set

The data set is at 10 min for about 4.5 months. The house temperature and humidity conditions were monitored with a ZigBee wireless sensor network. Each wireless node transmitted the temperature and humidity conditions around 3.3 min. Then, the wireless data was averaged for 10 minutes periods. The energy data was logged every 10 minutes with m-bus energy meters. Weather from the nearest airport weather station (Chievres Airport, Belgium) was downloaded from a public data set from Reliable Prognosis (rp5.ru), and merged together with the experimental data sets using the date and time column. Two random variables have been included in the data set for testing the regression models and to filter out non predictive attributes (parameters).

Lets get started!

Loading the data

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

In [2]:
```
%matplotlib inline
```

In [3]:
```
df = pd.read_csv('energydata_complete.csv')
```

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

Out[4]:

	date	Appliances	lights	T1	RH_1	T2	RH_2	T3	RH_3	
0	2016-01-11 17:00:00	60	30	19.89	47.596667	19.2	44.790000	19.79	44.730000	19.00
1	2016-01-11 17:10:00	60	30	19.89	46.693333	19.2	44.722500	19.79	44.790000	19.00
2	2016-01-11 17:20:00	50	30	19.89	46.300000	19.2	44.626667	19.79	44.933333	18.92
3	2016-01-11 17:30:00	50	40	19.89	46.066667	19.2	44.590000	19.79	45.000000	18.89
4	2016-01-11 17:40:00	60	40	19.89	46.333333	19.2	44.530000	19.79	45.000000	18.89

5 rows × 29 columns

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19735 entries, 0 to 19734
Data columns (total 29 columns):
date           19735 non-null object
Appliances     19735 non-null int64
lights         19735 non-null int64
T1             19735 non-null float64
RH_1           19735 non-null float64
T2             19735 non-null float64
RH_2           19735 non-null float64
T3             19735 non-null float64
RH_3           19735 non-null float64
T4             19735 non-null float64
RH_4           19735 non-null float64
T5             19735 non-null float64
RH_5           19735 non-null float64
T6             19735 non-null float64
RH_6           19735 non-null float64
T7             19735 non-null float64
RH_7           19735 non-null float64
T8             19735 non-null float64
RH_8           19735 non-null float64
T9             19735 non-null float64
RH_9           19735 non-null float64
T_out          19735 non-null float64
Press_mm_hg    19735 non-null float64
RH_out         19735 non-null float64
Windspeed      19735 non-null float64
Visibility     19735 non-null float64
Tdewpoint      19735 non-null float64
rv1            19735 non-null float64
rv2            19735 non-null float64
dtypes: float64(26), int64(2), object(1)
memory usage: 4.4+ MB
```

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

Out[6]:

	date	Appliances	lights	T1	RH_1	T2	RH_2	T3	RH_3	
0	2016-01-11 17:00:00	60	30	19.89	47.596667	19.2	44.790000	19.79	44.730000	19.0
1	2016-01-11 17:10:00	60	30	19.89	46.693333	19.2	44.722500	19.79	44.790000	19.0
2	2016-01-11 17:20:00	50	30	19.89	46.300000	19.2	44.626667	19.79	44.933333	18.9
3	2016-01-11 17:30:00	50	40	19.89	46.066667	19.2	44.590000	19.79	45.000000	18.8
4	2016-01-11 17:40:00	60	40	19.89	46.333333	19.2	44.530000	19.79	45.000000	18.8

5 rows × 29 columns

Data Cleaning as required

In [9]:
```
df.drop('date',axis=1, inplace=True)
```

In [10]:
```
df.head()
```

Out[10]:

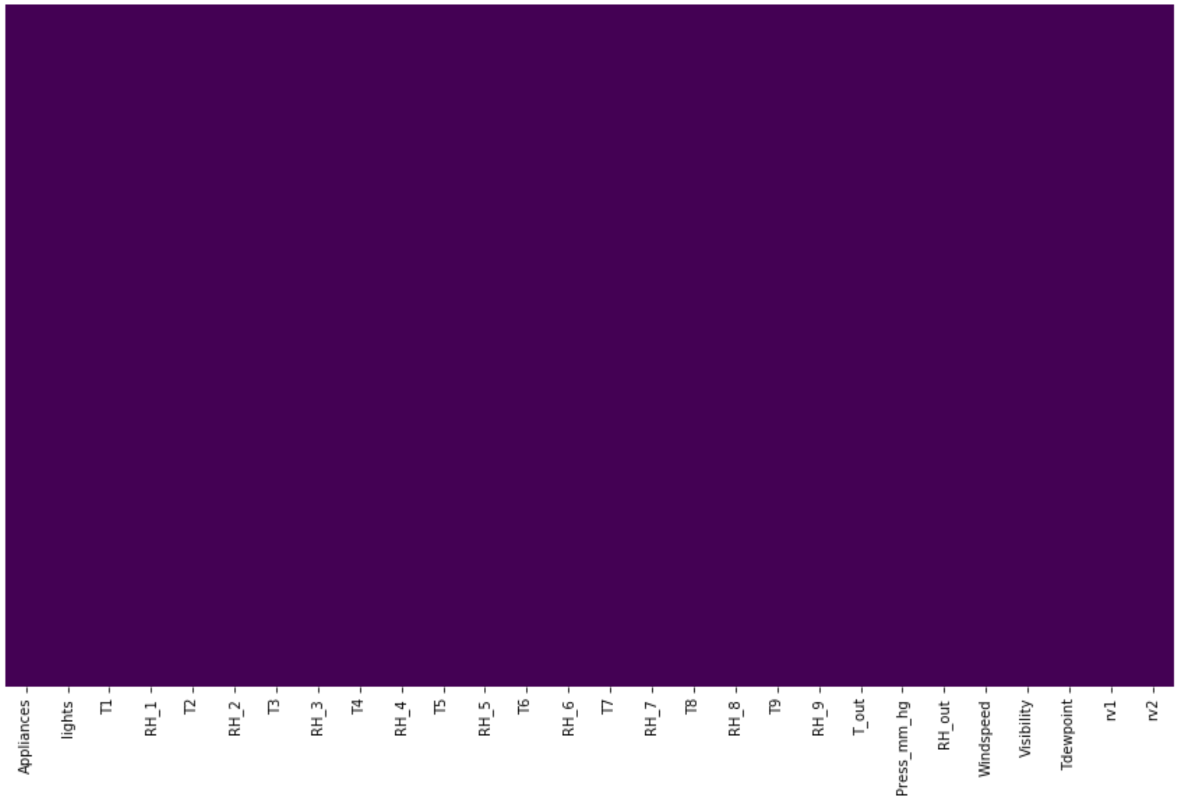
	Appliances	lights	T1	RH_1	T2	RH_2	T3	RH_3	T4	
0	60	30	19.89	47.596667	19.2	44.790000	19.79	44.730000	19.000000	45.
1	60	30	19.89	46.693333	19.2	44.722500	19.79	44.790000	19.000000	45.
2	50	30	19.89	46.300000	19.2	44.626667	19.79	44.933333	18.926667	45.
3	50	40	19.89	46.066667	19.2	44.590000	19.79	45.000000	18.890000	45.
4	60	40	19.89	46.333333	19.2	44.530000	19.79	45.000000	18.890000	45.

5 rows × 28 columns

Exploratory Data Analysis

In [32]:
```
plt.figure(figsize=(15,9))
sns.heatmap(df.isnull(), cbar=False, yticklabels=False, cmap='viridis')
```

Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x11e6f240>

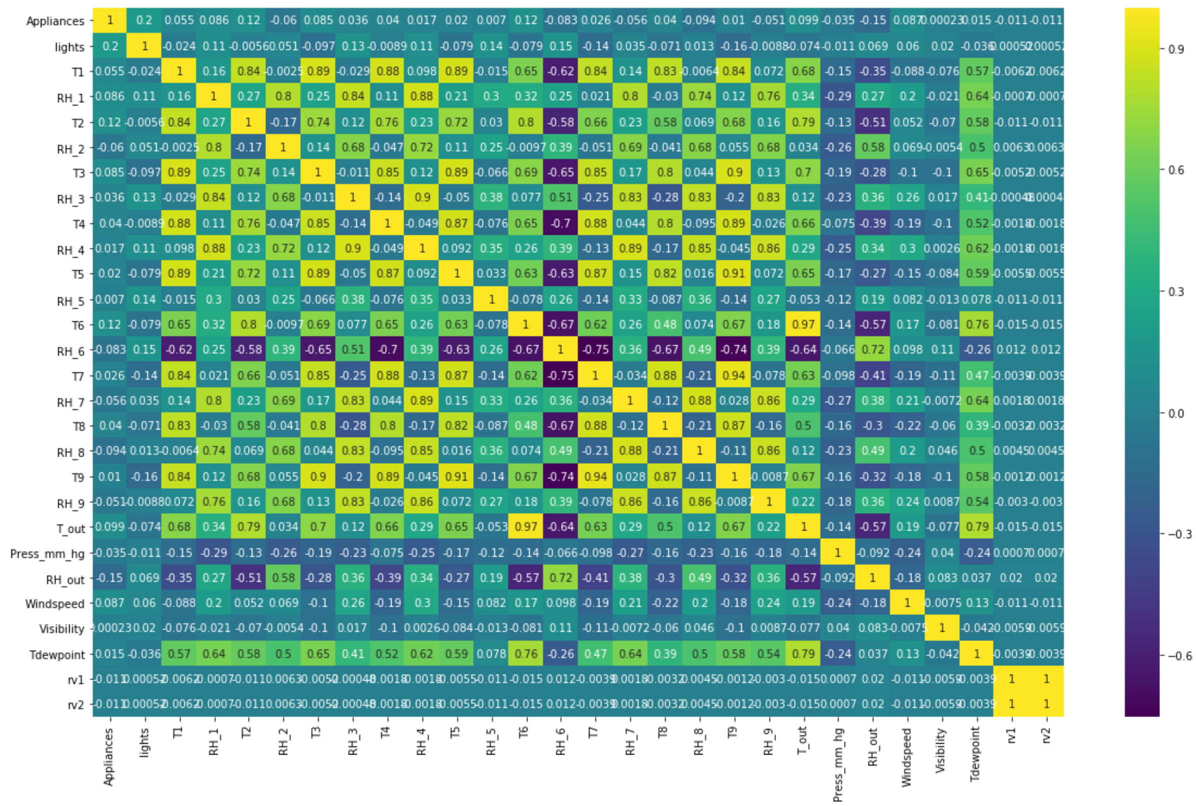


The above heatmap shows no spots, that means we do not have any null data in our dataset

Checking the correlation

In [30]:
```
plt.figure(figsize=(20,12))
sns.heatmap(df.corr(), annot=True, cmap='viridis')
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0xf90ac18>



Preparing Training & Test Data

In [11]:
```
from sklearn.model_selection import train_test_split
```

In [12]:
```
X_train, X_test, y_train, y_test = train_test_split(df.drop('Appliances', axis
=1), df['Appliances'], test_size=0.3,
                                                    random_state=101)
```

Initializing Model & Training the same

In [13]:
```
from sklearn.linear_model import LinearRegression
```

In [14]:
```
lm = LinearRegression()
```

In [15]:
```
lm.fit(X_train, y_train)
```

Out[15]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

Checking the coefficients fo the trained model

In [16]:
```
print(lm.coef_)
```

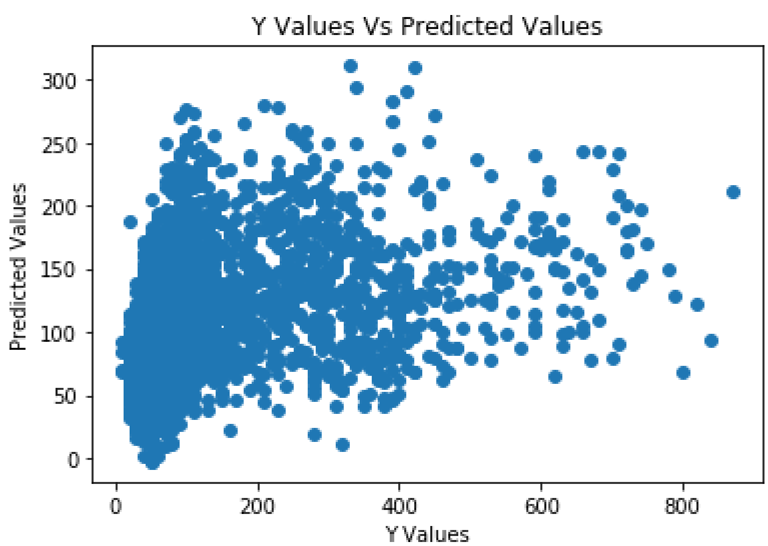
```
[  2.21381857  -0.46375014  14.30189183 -16.66308548 -12.75324926
  24.29752924   4.869307    -5.11914647  -0.18411024  -1.00348894
   0.2288889    6.97473352   0.2955164    1.86226864  -1.53987426
   8.07118621  -4.57404837 -13.32788725  -0.91907821  -9.84535376
   0.18863666  -1.06362364   1.66923607   0.2074002    4.86279984
  -0.03644794  -0.03644794]
```

Predicting & Representing the Values

In [17]:
```
prediction = lm.predict(X_test)
```

In [18]:
```
plt.scatter(y_test, prediction)
plt.xlabel('Y Values')
plt.ylabel('Predicted Values')
plt.title('Y Values Vs Predicted Values')
```

Out[18]: `Text(0.5,1,'Y Values Vs Predicted Values')`



Evaluating the Model

In [22]:
```
from sklearn import metrics
```

In [23]:
```
print("MAE: ", metrics.mean_absolute_error(y_test, prediction))
print("MSE: ", metrics.mean_squared_error(y_test, prediction))
print("RMSE: ", np.sqrt(metrics.mean_squared_error(y_test, prediction)))
```

```
MAE:  53.6273829358
MSE:  8845.48199288
RMSE:  94.0504226087
```

In [24]:
```
df.columns
```

Out[24]:
```
Index(['Appliances', 'lights', 'T1', 'RH_1', 'T2', 'RH_2', 'T3', 'RH_3', 'T
4',
       'RH_4', 'T5', 'RH_5', 'T6', 'RH_6', 'T7', 'RH_7', 'T8', 'RH_8', 'T9',
       'RH_9', 'T_out', 'Press_mm_hg', 'RH_out', 'Windspeed', 'Visibility',
       'Tdewpoint', 'rv1', 'rv2'],
      dtype='object')
```

In [25]:
```
coefficients = pd.DataFrame(lm.coef_, index=['lights', 'T1', 'RH_1', 'T2', 'RH
_2', 'T3', 'RH_3', 'T4',
       'RH_4', 'T5', 'RH_5', 'T6', 'RH_6', 'T7', 'RH_7', 'T8', 'RH_8', 'T9',
       'RH_9', 'T_out', 'Press_mm_hg', 'RH_out', 'Windspeed', 'Visibility',
       'Tdewpoint', 'rv1', 'rv2'], columns=['Coefficients'])
```

In [26]:
```
coefficients
```

Out[26]:

	Coefficients
lights	2.213819
T1	-0.463750
RH_1	14.301892
T2	-16.663085
RH_2	-12.753249
T3	24.297529
RH_3	4.869307
T4	-5.119146
RH_4	-0.184110
T5	-1.003489
RH_5	0.228889
T6	6.974734
RH_6	0.295516
T7	1.862269
RH_7	-1.539874
T8	8.071186
RH_8	-4.574048
T9	-13.327887
RH_9	-0.919078
T_out	-9.845354
Press_mm_hg	0.188637
RH_out	-1.063624
Windspeed	1.669236
Visibility	0.207400
Tdewpoint	4.862800
rv1	-0.036448
rv2	-0.036448

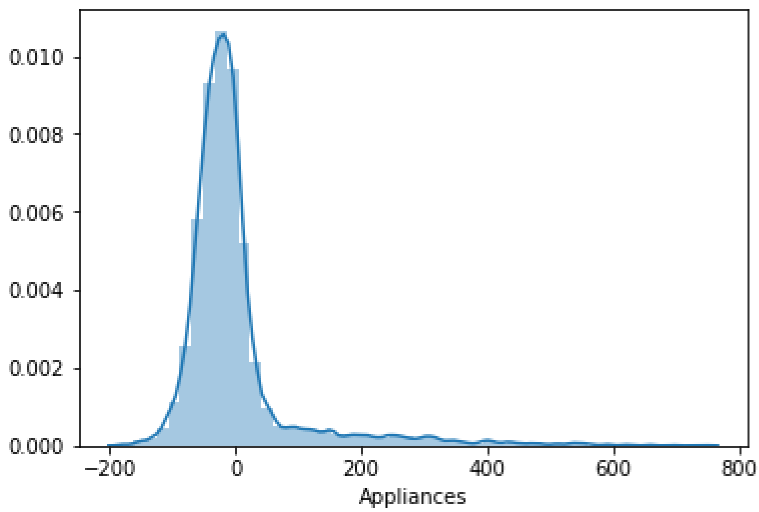
Residuals

You should have gotten a very good model with a good fit. Let's quickly explore the residuals to make sure everything was okay with our data.

Plot a histogram of the residuals and make sure it looks normally distributed. Use either seaborn distplot, or just plt.hist().

In [27]:
```
sns.distplot((y_test - prediction))
```

Out[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xc4d54a8>
```



Thus completes our project!