In [11]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib as plt
import warnings
warnings.filterwarnings('ignore')

# For randomized data splitting
from sklearn.model_selection import train_test_split

# To build linear regression_model
import statsmodels.api as sm

# To check model performance
from sklearn.metrics import mean_absolute_error, mean_squared_error

from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from statsmodels.compat import lzip
from sklearn import model_selection
from math import sqrt

```

In [12]:
```
energy = pd.read_csv("C://Users//E7270//Desktop//Hamoye/energydata_complete.
```

In [13]:
```
energy.head()
```

Out[13]:

| | date | Appliances | lights | T1 | RH_1 | T2 | RH_2 | T3 | RH_3 | T4 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2016-01-11 17:00:00 | 60 | 30 | 19.89 | 47.596667 | 19.2 | 44.790000 | 19.79 | 44.730000 | 19.000000 | ... |
| 1 | 2016-01-11 17:10:00 | 60 | 30 | 19.89 | 46.693333 | 19.2 | 44.722500 | 19.79 | 44.790000 | 19.000000 | ... |
| 2 | 2016-01-11 17:20:00 | 50 | 30 | 19.89 | 46.300000 | 19.2 | 44.626667 | 19.79 | 44.933333 | 18.926667 | ... |
| 3 | 2016-01-11 17:30:00 | 50 | 40 | 19.89 | 46.066667 | 19.2 | 44.590000 | 19.79 | 45.000000 | 18.890000 | ... |
| 4 | 2016-01-11 17:40:00 | 60 | 40 | 19.89 | 46.333333 | 19.2 | 44.530000 | 19.79 | 45.000000 | 18.890000 | ... |

5 rows × 29 columns

In [14]:
```
1 energy.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
```

In [16]:
```
1 energy.drop(['date'], axis = 1)
```

Out[16]:

| | Appliances | lights | T1 | RH_1 | T2 | RH_2 | T3 | RH_3 | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 60 | 30 | 19.890000 | 47.596667 | 19.200000 | 44.790000 | 19.790000 | 44.730000 | 19.000 |
| 1 | 60 | 30 | 19.890000 | 46.693333 | 19.200000 | 44.722500 | 19.790000 | 44.790000 | 19.000 |
| 2 | 50 | 30 | 19.890000 | 46.300000 | 19.200000 | 44.626667 | 19.790000 | 44.933333 | 18.926 |
| 3 | 50 | 40 | 19.890000 | 46.066667 | 19.200000 | 44.590000 | 19.790000 | 45.000000 | 18.890 |
| 4 | 60 | 40 | 19.890000 | 46.333333 | 19.200000 | 44.530000 | 19.790000 | 45.000000 | 18.890 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 19730 | 100 | 0 | 25.566667 | 46.560000 | 25.890000 | 42.025714 | 27.200000 | 41.163333 | 24.700 |
| 19731 | 90 | 0 | 25.500000 | 46.500000 | 25.754000 | 42.080000 | 27.133333 | 41.223333 | 24.700 |
| 19732 | 270 | 10 | 25.500000 | 46.596667 | 25.628571 | 42.768571 | 27.050000 | 41.690000 | 24.700 |
| 19733 | 420 | 10 | 25.500000 | 46.990000 | 25.414000 | 43.036000 | 26.890000 | 41.290000 | 24.700 |
| 19734 | 430 | 10 | 25.500000 | 46.600000 | 25.264286 | 42.971429 | 26.823333 | 41.156667 | 24.700 |

19735 rows × 28 columns

In [17]:
```
1 simple_linear_reg_df = energy[['T2', 'T6']].sample(15, random_state=2)
```

In [18]:
```
1 energy.isnull().sum()
```

Out[18]:
```
date           0
Appliances     0
lights         0
T1             0
RH_1           0
T2             0
RH_2           0
T3             0
RH_3           0
T4             0
RH_4           0
T5             0
RH_5           0
T6             0
RH_6           0
T7             0
RH_7           0
T8             0
RH_8           0
T9             0
RH_9           0
T_out          0
Press_mm_hg    0
RH_out         0
Windspeed      0
Visibility     0
Tdewpoint      0
rv1            0
rv2            0
dtype: int64
```

In [27]:
```
1 T2 = x
2 T6 = y
3 rss = np.sum(np.square(y - x))
4 round(rss, 3)
```

Out[27]:
```
3453843.036
```

In [26]:
```
1 energy.drop(['lights','date'], axis = 1)
```

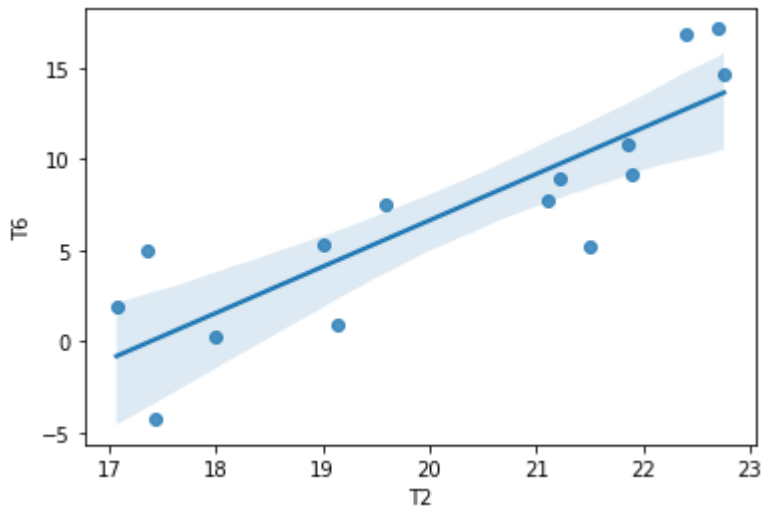
Out[26]:

| | Appliances | T1 | RH_1 | T2 | RH_2 | T3 | RH_3 | T4 | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 60 | 19.890000 | 47.596667 | 19.200000 | 44.790000 | 19.790000 | 44.730000 | 19.000000 | 4 |
| 1 | 60 | 19.890000 | 46.693333 | 19.200000 | 44.722500 | 19.790000 | 44.790000 | 19.000000 | 4 |
| 2 | 50 | 19.890000 | 46.300000 | 19.200000 | 44.626667 | 19.790000 | 44.933333 | 18.926667 | 4 |
| 3 | 50 | 19.890000 | 46.066667 | 19.200000 | 44.590000 | 19.790000 | 45.000000 | 18.890000 | 4 |
| 4 | 60 | 19.890000 | 46.333333 | 19.200000 | 44.530000 | 19.790000 | 45.000000 | 18.890000 | 4 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 19730 | 100 | 25.566667 | 46.560000 | 25.890000 | 42.025714 | 27.200000 | 41.163333 | 24.700000 | 4 |
| 19731 | 90 | 25.500000 | 46.500000 | 25.754000 | 42.080000 | 27.133333 | 41.223333 | 24.700000 | 4 |
| 19732 | 270 | 25.500000 | 46.596667 | 25.628571 | 42.768571 | 27.050000 | 41.690000 | 24.700000 | 4 |
| 19733 | 420 | 25.500000 | 46.990000 | 25.414000 | 43.036000 | 26.890000 | 41.290000 | 24.700000 | 4 |
| 19734 | 430 | 25.500000 | 46.600000 | 25.264286 | 42.971429 | 26.823333 | 41.156667 | 24.700000 | 4 |

19735 rows × 27 columns

In [28]:
```
1 import seaborn as sns
2 import matplotlib as plt
3 sns.regplot(x="T2", y="T6",
4             data=simple_linear_reg_df)
```

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x23444818608>




In [29]:
```
1 # independent variables
2 X = energy.drop(["Appliances"], axis=1)
3 # dependent variable
4 y = energy[["Appliances"]]
```

In [30]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.30, random_state=42
)
```

In [32]:
```
X_train.drop(['lights','date'], axis = 1)
```

Out[32]:

| | T1 | RH_1 | T2 | RH_2 | T3 | RH_3 | T4 | RH_4 | T5 |
|---|---|---|---|---|---|---|---|---|---|
| 9129 | 21.500000 | 35.626667 | 17.790000 | 40.590000 | 21.700000 | 35.26 | 20.39 | 33.863333 | 19.600000 |
| 2453 | 19.500000 | 44.560000 | 18.700000 | 44.290000 | 19.823333 | 44.50 | 18.10 | 43.860000 | 17.200000 |
| 9152 | 20.790000 | 35.400000 | 16.890000 | 42.030000 | 21.700000 | 36.00 | 19.70 | 33.200000 | 19.290000 |
| 12694 | 22.100000 | 43.260000 | 19.963333 | 45.500000 | 23.390000 | 39.79 | 21.10 | 39.060000 | 20.660000 |
| 16952 | 24.700000 | 42.360000 | 29.856667 | 31.790000 | 26.171429 | 38.59 | 25.10 | 39.760000 | 23.166667 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 11284 | 21.760000 | 40.900000 | 19.390000 | 43.090000 | 23.000000 | 39.00 | 21.50 | 39.790000 | 20.100000 |
| 11964 | 22.390000 | 43.700000 | 22.000000 | 42.066667 | 22.700000 | 41.23 | 21.29 | 44.120000 | 19.700000 |
| 5390 | 20.290000 | 35.700000 | 18.200000 | 37.590000 | 20.100000 | 37.59 | 18.20 | 35.290000 | 18.500000 |
| 860 | 21.790000 | 35.560000 | 20.434000 | 35.116000 | 21.200000 | 36.90 | 21.10 | 35.663333 | 18.050000 |
| 15795 | 21.323333 | 37.730000 | 19.890000 | 38.566667 | 22.700000 | 36.59 | 19.70 | 39.433333 | 19.200000 |

13814 rows × 26 columns

In [33]:
```
energy = energy.drop(columns="lights")
```

In [45]:
```
#X_train = X_train.drop(columns="date")
X_test = X_test.drop(columns="lights")
X_test
```

Out[45]:

| | T1 | RH_1 | T2 | RH_2 | T3 | RH_3 | T4 | RH_4 | |
|---|---|---|---|---|---|---|---|---|---|
| 8980 | 20.890000 | 35.400000 | 17.760000 | 39.163333 | 20.290000 | 36.900000 | 19.760000 | 34.200000 | 18. |
| 2754 | 21.890000 | 53.100000 | 21.290000 | 45.360000 | 21.633333 | 49.226667 | 20.533333 | 40.966667 | 17. |
| 9132 | 21.390000 | 35.500000 | 17.633333 | 40.530000 | 21.666667 | 35.200000 | 20.290000 | 33.760000 | 19. |
| 14359 | 21.390000 | 41.033333 | 23.890000 | 34.840000 | 22.033333 | 36.933333 | 22.390000 | 35.236000 | 19. |
| 8875 | 19.963333 | 35.126667 | 16.463333 | 40.126667 | 20.000000 | 36.400000 | 19.260000 | 34.966667 | 17. |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 831 | 22.100000 | 37.933333 | 21.196667 | 35.133333 | 21.856667 | 41.100000 | 21.856667 | 39.100000 | 18. |
| 10993 | 21.700000 | 38.290000 | 19.200000 | 41.560000 | 23.000000 | 38.326667 | 20.260000 | 39.260000 | 20. |
| 11761 | 20.926667 | 39.400000 | 18.700000 | 43.000000 | 21.790000 | 37.400000 | 19.790000 | 38.290000 | 19. |
| 12364 | 22.000000 | 41.230000 | 20.700000 | 41.900000 | 23.000000 | 38.790000 | 21.500000 | 40.656667 | 20. |
| 11863 | 20.890000 | 42.290000 | 19.033333 | 44.963333 | 21.600000 | 39.090000 | 19.700000 | 41.466667 | 19. |

5921 rows × 26 columns

In [39]:
```
print(X_train.head())
```

```
          T1       RH_1         T2   RH_2         T3   RH_3     T4       RH_4
\
9129   21.50  35.626667  17.790000  40.59  21.700000  35.26  20.39  33.863333
2453   19.50  44.560000  18.700000  44.29  19.823333  44.50  18.10  43.860000
9152   20.79  35.400000  16.890000  42.03  21.700000  36.00  19.70  33.200000
12694  22.10  43.260000  19.963333  45.50  23.390000  39.79  21.10  39.060000
16952  24.70  42.360000  29.856667  31.79  26.171429  38.59  25.10  39.760000

              T5    RH_5  ...         T9       RH_9      T_out  Press_mm_hg  \
9129   19.600000  40.425  ...  19.463333  38.260000   0.250000   766.400000
2453   17.200000  52.000  ...  17.200000  46.163333   3.166667   765.266667
9152   19.290000  39.900  ...  19.390000  39.067500  -1.566667   766.000000
12694  20.660000  58.054  ...  20.290000  37.400000   8.833333   753.366667
16952  23.166667  60.130  ...  23.100000  44.466667  21.433333   752.100000

          RH_out  Windspeed  Visibility  Tdewpoint        rv1        rv2
9129   83.000000   2.000000   65.000000  -2.350000  36.226675  36.226675
2453   85.333333   2.000000   40.000000   0.966667  43.199767  43.199767
9152   89.333333   1.333333   60.666667  -3.100000  24.976055  24.976055
12694  81.000000   1.666667   26.000000   5.733333  16.161125  16.161125
16952  51.000000   2.000000   40.000000  10.800000  17.055346  17.055346

[5 rows x 26 columns]
```

In [46]:
```
linear_model = LinearRegression()
#fit the model to the training dataset
linear_model.fit(X_train, y_train)
#obtain predictions
predicted_values = linear_model.predict(X_test)
#MAE
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(y_test, predicted_values)
round(mae, 3)
```

Out[46]:
```
53.643
```

In [47]:
```
rss = np.sum(np.square(y_test - predicted_values))
round(rss, 3)
```

Out[47]:
```
Appliances    5.191850e+07
dtype: float64
```

In [48]:
```
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, predicted_values))
round(rmse, 3)
```

Out[48]:
```
93.64
```

In [49]:
```
from sklearn.metrics import r2_score
r2_score = r2_score(y_test, predicted_values)
round(r2_score, 3)
```

Out[49]:
```
0.149
```

In [10]:
```
from sklearn.linear_model import Ridge
ridge_reg = Ridge(alpha=0.4)
ridge_reg.fit(X_train, y_train)

```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-10-2e0066c0935b> in <module>
      1 from sklearn.linear_model import Ridge
      2 ridge_reg = Ridge(alpha=0.4)
----> 3 ridge_reg.fit(X_train, y_train)

NameError: name 'X_train' is not defined
```

In [52]:
```
from sklearn.linear_model import Lasso
lasso_reg = Lasso(alpha=0.001)

lasso_reg.fit(X_train, y_train)
```

Out[52]:
```
Lasso(alpha=0.001, copy_X=True, fit_intercept=True, max_iter=1000,
      normalize=False, positive=False, precompute=False, random_state=None,
      selection='cyclic', tol=0.0001, warm_start=False)
```

In [ ]:
```
olsmodel_final = sm.OLS(y_train, X_train).fit()
print(olsmodel_final.summary())
```

In [ ]:
```
from sklearn.linear_model import Ridge

def get_weights_df(model, feat, col_name):
    #this function returns the weight of every feature
    weights = pd.Series(model.coef_, feat.columns).sort_values()
    weights_df = pd.DataFrame(weights).reset_index()
    weights_df.columns = ['Features', col_name]
    weights_df[col_name].round(3)
    return weights_df

```

In [54]:
```
from sklearn.linear_model import Ridge
#ridge_weights_df = get_weights_df(ridge_reg, X_train, 'Ridge_Weight')
lasso_weights_df = get_weights_df(lasso_reg, X_train, 'Lasso_weight')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-54-5df571a1e2d8> in <module>
      1 from sklearn.linear_model import Ridge
      2 #ridge_weights_df = get_weights_df(ridge_reg, X_train, 'Ridge_Weight')
----> 3 lasso_weights_df = get_weights_df(lasso_reg, X_train, 'Lasso_weight')

NameError: name 'get_weights_df' is not defined
```

In [55]:
```
#linear_model_weights = get_weights_df(model, x_train, 'Linear_Model_Weight'
#ridge_weights_df = get_weights_df(ridge_reg, x_train, 'Ridge_Weight')
#lasso_weights_df = get_weights_df(lasso_reg, x_train, 'Lasso_weight')

final_weights = pd.merge(linear_model_weights, ridge_weights_df, on='Feature
final_weights = pd.merge(final_weights, lasso_weights_df, on='Features')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-55-c6c0fa20f1ee> in <module>
      3 #lasso_weights_df = get_weights_df(lasso_reg, x_train, 'Lasso_weight')
      4
----> 5 final_weights = pd.merge(linear_model_weights, ridge_weights_df, on='Fe
atures')
      6 final_weights = pd.merge(final_weights, lasso_weights_df, on='Features'
)

NameError: name 'linear_model_weights' is not defined
```

In [ ]:
```
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, predicted_values))
round(rmse, 3)
```

In [ ]:
```
rss = np.sum(np.square(y_test - predicted_values))
round(rss, 3)
```

In [50]:
```
rr = Ridge(alpha=0.01)
rr.fit(X_train, y_train)
pred_train_rr= rr.predict(X_train)
print(np.sqrt(mean_squared_error(y_train,pred_train_rr)))
print(r2_score(y_train, pred_train_rr))

pred_test_rr= rr.predict(X_test)
print(np.sqrt(mean_squared_error(y_test,pred_test_rr)))
print(r2_score(y_test, pred_test_rr))
```

```
95.21565985217909

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-50-4518f6ab8d5e> in <module>
      3 pred_train_rr= rr.predict(X_train)
      4 print(np.sqrt(mean_squared_error(y_train,pred_train_rr)))
----> 5 print(r2_score(y_train, pred_train_rr))
      6
      7 pred_test_rr= rr.predict(X_test)

TypeError: 'numpy.float64' object is not callable
```

In [53]:
```
model_lasso = Lasso(alpha=0.01)
model_lasso.fit(X_train, y_train)
pred_train_lasso= model_lasso.predict(X_train)
print(np.sqrt(mean_squared_error(y_train,pred_train_lasso)))
print(r2_score(y_train, pred_train_lasso))

pred_test_lasso= model_lasso.predict(X_test)
print(np.sqrt(mean_squared_error(y_test,pred_test_lasso)))
print(r2_score(y_test, pred_test_lasso))
```

```
95.21569972708544

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-53-eb1ad2134ad7> in <module>
      3 pred_train_lasso= model_lasso.predict(X_train)
      4 print(np.sqrt(mean_squared_error(y_train,pred_train_lasso)))
----> 5 print(r2_score(y_train, pred_train_lasso))
      6
      7 pred_test_lasso= model_lasso.predict(X_test)

TypeError: 'numpy.float64' object is not callable
```

In [ ]:
```

```