```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.r
```

```
#import libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
file = '/content/drive/MyDrive/hamoye/energydata_complete.csv'
energy_file = pd.read_csv(file)
```

```
energy_file.head()
```

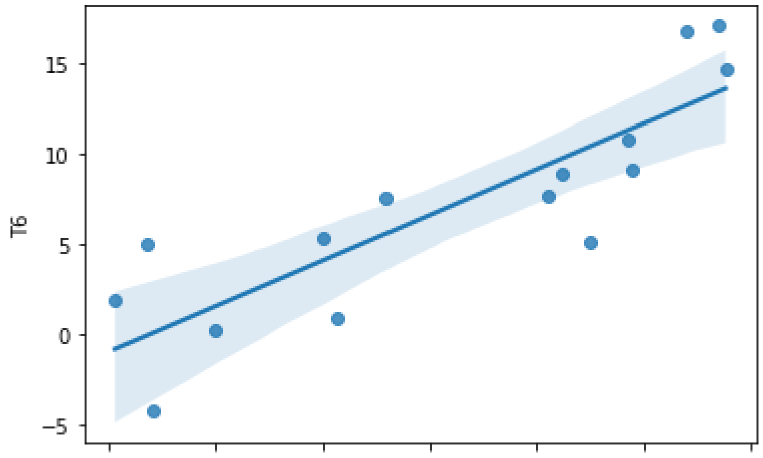
	date	Appliances	lights	T1	RH_1	T2	RH_2	T3	RH_3	
0	2016-01-11 17:00:00	60	30	19.89	47.596667	19.2	44.790000	19.79	44.730000	19
1	2016-01-11 17:10:00	60	30	19.89	46.693333	19.2	44.722500	19.79	44.790000	19
2	2016-01-11 17:20:00	50	30	19.89	46.300000	19.2	44.626667	19.79	44.933333	18
3	2016-01-11 17:30:00	50	40	19.89	46.066667	19.2	44.590000	19.79	45.000000	18
4	2016-01-11 17:40:00	60	40	19.89	46.333333	19.2	44.530000	19.79	45.000000	18

5 rows × 29 columns

```
linear_reg = energy_file[['T2', 'T6']].sample(15, random_state = 2)
```

```
sns.regplot(x='T2', y='T6', data = linear_reg)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7f45cd7190>
```



```
energy_df = energy_file.drop(['date', 'lights'], axis=1)
```

```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler = MinMaxScaler()
normalised_df = pd.DataFrame(scaler.fit_transform(energy_df), columns=energy_df.columns)
appliances_target = normalised_df['Appliances']
```

```
#split into train and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(energy_df, appliances_target, test_size=0.3
```

```
from sklearn.linear_model import LinearRegression #import linear regressiom
linear_model = LinearRegression()
linear_model.fit(x_train, y_train) #fit the model to the training dataset
pred_values = linear_model.predict(x_test) #predictions from the model
```

```
#MAE
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(y_test, pred_values)
round(mae, 2)
```

```
0.0
```

```
#RSS
rss = np.sum(np.square(y_test - pred_values))
round(rss, 2)
```

```
0.0
```

```
#RMSE
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, pred_values))
round(rmse, 3)
```

```
0.0
```

```
#RSquared
from sklearn.metrics import r2_score
r2_score = r2_score(y_test, pred_values)
round(r2_score, 2)
```

```
1.0
```

```
#Ridge regression
from sklearn.linear_model import Ridge
ridge_reg = Ridge(alpha=0.4)
ridge_reg.fit(x_train, y_train)
```

```
Ridge(alpha=0.4)
```

```
#Lasso regression
from sklearn.linear_model import Lasso
lasso_reg = Lasso(alpha=0.001)
lasso_reg.fit(x_train, y_train)
```

```
Lasso(alpha=0.001)
```

```
#comparing the effects of regularisation
def get_weights_df(model, feat, col_name):
  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

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

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-164-0a0f2901c296> in <module>
      7   return weights_df
      8
----> 9 linear_model_weights = get_weights_df(model, x_train, 'Linear_Model_Weight')
     10 ridge_weights_df = get_weights_df(ridge_reg, x_train, 'Ridge_Weight')
     11 lasso_weights_df = get_weights_df(lasso_reg, x_train, 'Lasso_weight')

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

SEARCH STACK OVERFLOW