In [1]:
```
import pandas as pd
import numpy as np

df=pd.read_csv("energydata_complete.csv")
df.head()
```

Out[1]:

	date	Appliances	lights	T1	RH_1	T2	RH_2	T3	RH_3	T4	...	T9	RH_9	T_out	Press_mm_hg	RH_out	Windspeed	Visibility	Tdewpoint	rv1	rv2
0	2016-01-11 17:00:00	60	30	19.89	47.596667	19.2	44.790000	19.79	44.730000	19.000000	...	17.033333	45.53	6.600000	733.5	92.0	7.000000	63.000000	5.3	13.275433	13.275433
1	2016-01-11 17:10:00	60	30	19.89	46.693333	19.2	44.722500	19.79	44.790000	19.000000	...	17.066667	45.56	6.483333	733.6	92.0	6.666667	59.166667	5.2	18.606195	18.606195
2	2016-01-11 17:20:00	50	30	19.89	46.300000	19.2	44.626667	19.79	44.933333	18.926667	...	17.000000	45.50	6.366667	733.7	92.0	6.333333	55.333333	5.1	28.642668	28.642668
3	2016-01-11 17:30:00	50	40	19.89	46.066667	19.2	44.590000	19.79	45.000000	18.890000	...	17.000000	45.40	6.250000	733.8	92.0	6.000000	51.500000	5.0	45.410389	45.410389
4	2016-01-11 17:40:00	60	40	19.89	46.333333	19.2	44.530000	19.79	45.000000	18.890000	...	17.000000	45.40	6.133333	733.9	92.0	5.666667	47.666667	4.9	10.084097	10.084097

5 rows × 29 columns

In [6]:
```
# Drop columns "date and lights"
df_columns = df.drop(["date","lights"], axis=1)
df_columns
```

Out[6]:

	Appliances	T1	RH_1	T2	RH_2	T3	RH_3	T4	RH_4	T5	...	T9	RH_9	T_out	Press_mm_hg	RH_out	Windspeed	Visibility	Tdewpoint	rv1	rv2
0	60	19.890000	47.596667	19.200000	44.790000	19.790000	44.730000	19.000000	45.566667	17.166667	...	17.033333	45.5300	6.600000	733.5	92.000000	7.000000	63.000000	5.300000	13.275433	13.275433
1	60	19.890000	46.693333	19.200000	44.722500	19.790000	44.790000	19.000000	45.992500	17.166667	...	17.066667	45.5600	6.483333	733.6	92.000000	6.666667	59.166667	5.200000	18.606195	18.606195
2	50	19.890000	46.300000	19.200000	44.626667	19.790000	44.933333	18.926667	45.890000	17.166667	...	17.000000	45.5000	6.366667	733.7	92.000000	6.333333	55.333333	5.100000	28.642668	28.642668
3	50	19.890000	46.066667	19.200000	44.590000	19.790000	45.000000	18.890000	45.723333	17.166667	...	17.000000	45.4000	6.250000	733.8	92.000000	6.000000	51.500000	5.000000	45.410389	45.410389
4	60	19.890000	46.333333	19.200000	44.530000	19.790000	45.000000	18.890000	45.530000	17.200000	...	17.000000	45.4000	6.133333	733.9	92.000000	5.666667	47.666667	4.900000	10.084097	10.084097
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19730	100	25.566667	46.560000	25.890000	42.025714	27.200000	41.163333	24.700000	45.590000	23.200000	...	23.200000	46.7900	22.733333	755.2	55.666667	3.333333	23.666667	13.333333	43.096812	43.096812
19731	90	25.500000	46.500000	25.754000	42.080000	27.133333	41.223333	24.700000	45.590000	23.230000	...	23.200000	46.7900	22.600000	755.2	56.000000	3.500000	24.500000	13.300000	49.282940	49.282940
19732	270	25.500000	46.596667	25.628571	42.768571	27.050000	41.690000	24.700000	45.730000	23.230000	...	23.200000	46.7900	22.466667	755.2	56.333333	3.666667	25.333333	13.266667	29.199117	29.199117
19733	420	25.500000	46.990000	25.414000	43.036000	26.890000	41.290000	24.700000	45.790000	23.200000	...	23.200000	46.8175	22.333333	755.2	56.666667	3.833333	26.166667	13.233333	6.322784	6.322784
19734	430	25.500000	46.600000	25.264286	42.971429	26.823333	41.156667	24.700000	45.963333	23.200000	...	23.200000	46.8450	22.200000	755.2	57.000000	4.000000	27.000000	13.200000	34.118851	34.118851

19735 rows × 27 columns

In [51]:
```
np.random.seed(42)

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
normalised_df=pd.DataFrame(scaler.fit_transform(df_columns))

# Create data

x= df_columns.drop("Appliances",axis=1)
y= df_columns["Appliances"]

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.3)

from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train,y_train)
predicted_value= model.predict(x_test)

#from sklearn.metrics import mean_absolute_error
#mae= mean_absolute_error(y_test, predicted_value)

from sklearn.metrics import mean_absolute_error
mae= mean_absolute_error(y_test,predicted_value)
(mae,2)
```

Out[51]:
```
(53.6429776558496, 2)
```

In [38]:
```
round(mae,2)
```

Out[38]:
```
53.64
```

In [ ]:
```
# RSS

print("Residual sum of squares: %.2f"
  % ((y - predicted_value) ** 2).sum())
```

In [54]:
```
# RMSE
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, predicted_value))
round(rmse, 3)
```

Out[54]:
```
93.64
```

In [46]:
```
# Coefficient of determination
from sklearn.metrics import r2_score
r2 = r2_score(y_test,predicted_value)
round(r2,2)
```

Out[46]:
```
0.15
```

In [63]:
```
# RIDGE MODEL
np.random.seed(42)
import numpy as np
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
normalised_df=pd.DataFrame(scaler.fit_transform(df_columns))

# Create data
rnp = np.random.RandomState(42)
x= df_columns.drop("Appliances",axis=1)
y= df_columns["Appliances"]

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.3)

from sklearn.linear_model import Ridge
clf = Ridge(alpha=0.4)
clf.fit(x_train,y_train)
value= model.predict(x_test)

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

Out[63]:
```
93.64
```

In [69]:
```
# LASSO REGRESSION MODEL
#Create data

x= df_columns.drop("Appliances",axis=1)
y= df_columns["Appliances"]

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.3)

from sklearn import linear_model
reg = linear_model.Lasso(alpha=0.001)
reg.fit(x_train,y_train)
value= model.predict(x_test)

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

```
C:\Users\Aboya\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:647: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or c
onsider increasing regularisation. Duality gap: 2.869e+06, tolerance: 1.467e+04
  model = cd_fast.enet_coordinate_descent(
```

Out[69]:
```
93.971
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: