Dataset Description

The dataset for the remainder of this quiz is the Appliances Energy Prediction data. 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). The attribute information can be seen below.

Attribute Information:

Date, time year-month-day hour:minute:second

Appliances, energy use in Wh

lights, energy use of light fixtures in the house in Wh

T1, Temperature in kitchen area, in Celsius

RH_1, Humidity in kitchen area, in %

T2, Temperature in living room area, in Celsius

RH_2, Humidity in living room area, in %

T3, Temperature in laundry room area

RH_3, Humidity in laundry room area, in %

T4, Temperature in office room, in Celsius

RH_4, Humidity in office room, in %

T5, Temperature in bathroom, in Celsius

RH_5, Humidity in bathroom, in %

T6, Temperature outside the building (north side), in Celsius

RH_6, Humidity outside the building (north side), in %

T7, Temperature in ironing room , in Celsius

RH_7, Humidity in ironing room, in %

T8, Temperature in teenager room 2, in Celsius

RH_8, Humidity in teenager room 2, in %

T9, Temperature in parents room, in Celsius

RH_9, Humidity in parents room, in %

To, Temperature outside (from Chievres weather station), in Celsius

Pressure (from Chievres weather station), in mm Hg

RH_out, Humidity outside (from Chievres weather station), in %

Wind speed (from Chievres weather station), in m/s

Visibility (from Chievres weather station), in km

Tdewpoint (from Chievres weather station), Â °C

rv1, Random variable 1, nondimensional

rv2, Random variable 2, nondimensional

To answer some questions, you will need to normalize the dataset using the MinMaxScaler after removing the following columns: [“date”, “lights”]. The target variable is “Appliances”. Use a 70-30 train-test set split with a random state of 42 (for reproducibility). Run a multiple linear regression using the training set and evaluate your model on the test set.

In [46]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

%matplotlib inline
```

In [47]:
```
df=pd.read_csv('downloads\energydata_complete.csv')
```

In [48]:
```
df.head()
```

Out[48]:

	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 [49]:
```
df.drop(columns=['date', 'lights'], inplace=True)
```

In [50]:
```
df.head()
```

Out[50]:

	Appliances	T1	RH_1	T2	RH_2	T3	RH_3	T4	RH_4	
0	60	19.89	47.596667	19.2	44.790000	19.79	44.730000	19.000000	45.566667	17.166
1	60	19.89	46.693333	19.2	44.722500	19.79	44.790000	19.000000	45.992500	17.166
2	50	19.89	46.300000	19.2	44.626667	19.79	44.933333	18.926667	45.890000	17.166
3	50	19.89	46.066667	19.2	44.590000	19.79	45.000000	18.890000	45.723333	17.166
4	60	19.89	46.333333	19.2	44.530000	19.79	45.000000	18.890000	45.530000	17.200

5 rows × 27 columns

In [51]:
```
df.isnull().sum()
```

Out[51]:
```
Appliances     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 [52]:
```
#Firstly, we normalise our dataset to a common scale using the min max scaler
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
normalised_df = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
features_df = normalised_df.drop(columns=['Appliances'])
heating_target = normalised_df['Appliances']
```

In [53]:
```
predictor = df["T2"].values.reshape(-1,1)
response = df["T6"].values.reshape(-1,1)
```

In [54]:
```
#Now, we split our dataset into the training and testing dataset. Recall that
 we had earlier segmented the features and target variables.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(predictor, response, test_
size=0.3, random_state=42)
```

In [55]:
```
from sklearn.linear_model import LinearRegression

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)
```

In [56]:
```
from sklearn.metrics import r2_score
r2_score = r2_score(y_test, predicted_values)
round(r2_score, 2)
```

Out[56]: 0.64

MLR

In [62]:
```
#Now, we split our dataset into the training and testing dataset. Recall that
 we had earlier segmented the features and target variables.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(features_df, heating_targe
t, test_size=0.3, random_state=42)
```

In [63]:
```
from sklearn.linear_model import LinearRegression

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)
```

In [64]:
```
print(linear_model.intercept_)
```

```
0.15290295882253052
```

Measuring Regression performance

In [65]:
```
from sklearn.metrics import r2_score
r2_score = r2_score(y_test, predicted_values)
round(r2_score, 3)
```

Out[65]: 0.149

In [67]:
```
#MAE
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(y_test, predicted_values)
round(mae, 2)
```

Out[67]: 0.05

In [68]:
```
#RSS
rss = np.sum(np.square(y_test - predicted_values))
round(rss, 2)
```

Out[68]: 45.35

In [69]:
```
#RMS
from sklearn.metrics import  mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, predicted_values))
round(rmse, 2)
```

Out[69]: 0.09

In [70]:
```
# coefficient of determination = r-squared

from sklearn.metrics import r2_score
r2_score = r2_score(y_test, predicted_values)
round(r2_score, 2)
```

Out[70]: 0.15

Ridge Regression

In [79]:
```
from sklearn.linear_model import Ridge
ridge_reg = Ridge(alpha=0.4)
ridge_reg.fit(x_train, y_train)
```

Out[79]:
```
Ridge(alpha=0.4, copy_X=True, fit_intercept=True, max_iter=None,
      normalize=False, random_state=None, solver='auto', tol=0.001)
```

Lasso Regression

In [86]:
```
from sklearn.linear_model import Lasso
lasso_reg = Lasso(alpha=0.001)
lasso_reg.fit(x_train, y_train)
```

Out[86]:
```
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 [81]:
```
#comparing the effects of regularisation
def get_weights_df(model, feat, df):
    #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', df]
    weights_df[df].round(3)
    return weights_df
```

In [82]:
```
linear_model_weights = get_weights_df(linear_model, x_train, 'Linear_Model_Wei
ght')
ridge_weights_df = get_weights_df(ridge_reg, x_train, 'Ridge_Weight')
lasso_weights_df = get_weights_df(lasso_reg, x_train, 'Lasso_weight')
```

In [83]:
```
final_weights = pd.merge(linear_model_weights, ridge_weights_df, on='Features'
)
final_weights = pd.merge(final_weights, lasso_weights_df, on='Features')
```

In [84]:
```
final_weights
```

Out[84]:

	Features	Linear_Model_Weight	Ridge_Weight	Lasso_weight
0	RH_2	-0.456698	-0.411071	-0.000000
1	T_out	-0.321860	-0.262172	0.000000
2	T2	-0.236178	-0.201397	0.000000
3	T9	-0.189941	-0.188916	-0.000000
4	RH_8	-0.157595	-0.156830	-0.000110
5	RH_out	-0.077671	-0.054724	-0.049557
6	RH_7	-0.044614	-0.045977	-0.000000
7	RH_9	-0.039800	-0.041367	-0.000000
8	T5	-0.015657	-0.019853	-0.000000
9	T1	-0.003281	-0.018406	0.000000
10	rv2	0.000770	0.000748	-0.000000
11	rv1	0.000770	0.000748	-0.000000
12	Press_mm_hg	0.006839	0.006584	-0.000000
13	T7	0.010319	0.010098	-0.000000
14	Visibility	0.012307	0.012076	0.000000
15	RH_5	0.016006	0.016152	0.000000
16	RH_4	0.026386	0.024579	0.000000
17	T4	0.028981	0.027384	-0.000000
18	Windspeed	0.029183	0.030268	0.002912
19	RH_6	0.038049	0.035519	-0.000000
20	RH_3	0.096048	0.095135	0.000000
21	T8	0.101995	0.101028	0.000000
22	Tdewpoint	0.117758	0.083128	0.000000
23	T6	0.236425	0.217292	0.000000
24	T3	0.290627	0.288087	0.000000
25	RH_1	0.553547	0.519525	0.017880

In [87]:
```
from sklearn.linear_model import Ridge
ridge_reg = Ridge(alpha=0.4)
ridge_reg.fit(x_train, y_train)
```

Out[87]:
```
Ridge(alpha=0.4, copy_X=True, fit_intercept=True, max_iter=None,
      normalize=False, random_state=None, solver='auto', tol=0.001)
```

In [88]:
```
#comparing the effects of regularisation
def get_weights_df(model, feat, df):
    #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', df]
    weights_df[df].round(3)
    return weights_df
```

In [90]:
```
#RMS
from sklearn.metrics import  mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, predicted_values))
round(rmse, 3)
```

Out[90]:
```
0.088
```

In [ ]:
```

```