

1. Exploring a database (15%)

Address of the database:

<https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction>

Content of the database

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).

Size of the database

There are 19736 rows and 29 columns of data.

Attribute Information format and units.

Column1: 'date' includes year/month/day/hour/minute/second with an interval of ten minutes. Date format.

Column2: 'Appliances' includes energy use in Wh. Integer format.

Column3: 'lights' includes energy use of light fixtures in the house in Wh. Integer format.

Columns4,6,8,10,12,14,16,18,20: 'T' includes the temperature in different areas in the house, in Celsius. Float format.

Columns5,7,9,11,13,15,17,19,21: 'RH_' includes the humidity in different areas in the house, in %. Float format.

Column22: 'T_out' includes the temperature outside, in Celsius. Float format.

Column23: 'Press_mm_hg' includes atmospheric pressure, in mm-Hg. Float format.

Column24: 'RH_out' includes Humidity outside, in %. Float format.

Column25: 'Windspeed' in m/s. Float format.

Column26: 'Visibility' in km. Float format.

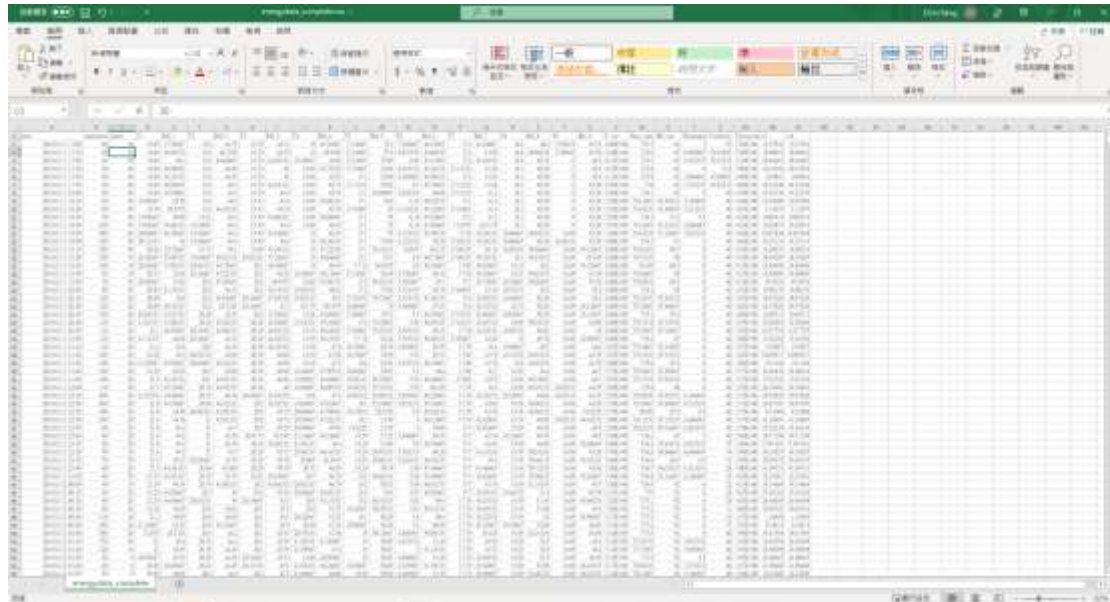
Column27: 'Tdewpoint', $^{\circ}\text{C}$. Float format.

Column28: 'rv1' Random variable 1, nondimensional. Float format.

Column29: 'rv2' Random variable 2, nondimensional. Float format.

Application

Aggregating energy use from households allow us to predict energy demand and optimize the distribution and energy storage use in a power producer point of view. The data also allow consumers to visualize their use of energy and plan their electricity usage to lower cost of utility bills and energy consumption in general.

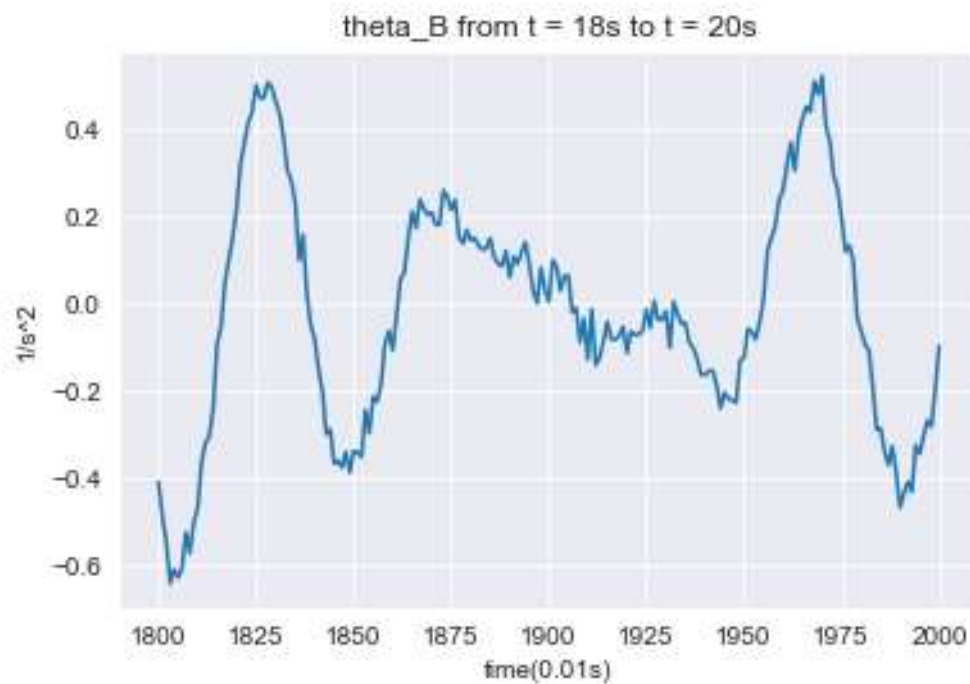
A screenshot of a Microsoft Excel spreadsheet displaying a large table of data. The spreadsheet has a green title bar and a ribbon at the top with various tabs like 'File', 'Home', 'Insert', etc. The main area is filled with a grid of cells containing numerical data. The data appears to be organized in columns, with some columns having headers that are partially visible. The spreadsheet is showing a large number of rows, with the data extending down to row 1048576. The status bar at the bottom indicates the active cell is A1 and the sheet is named 'Sheet1'.

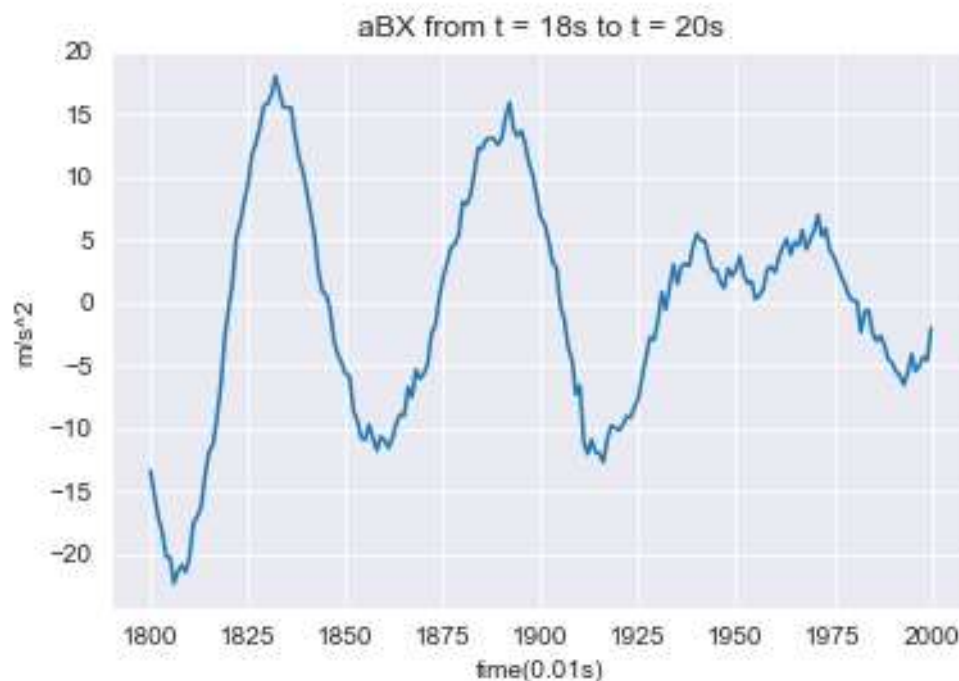
Source: Candanedo, Luis M., et al. "Data Driven Prediction Models of Energy Use of Appliances in a Low-Energy House." Energy and Buildings, Elsevier, 31 Jan. 2017, <https://www.sciencedirect.com/science/article/pii/S0378778816308970>.

2. Dealing with Time-Series Data (60%)

(a)

```
In [36]: # theta_B = (a5-a6)/30
# aBX = a7*theta_B**7.5
# time = 18 to 20s = 1800 to 2000 column data
data["theta_B"] = (data["A5"]-data["A6"])/30 # add column theta_B in dataframe
data["aBX"] = data["A7"]-data["theta_B"]**7.5 #add column aBX in dataframe
data_18_to_20s = data.loc[1800:2000] # select data from 18 to 20 seconds
theta_B_plot = sns.lineplot(data = data_18_to_20s["theta_B"])
sns.set_style("darkgrid")
theta_B_plot.set_xlabel("time(0.01s)")
theta_B_plot.set_ylabel("m/s^2")
theta_B_plot.set_title("theta_B from t = 18s to t = 20s")
theta_B_plot.figure.savefig("theta_B from t = 18s to t = 20s.png")
```





(b)

```
In [14]: # maximum absolute value of aBX
print('The maximum absolute value of aBX is', round(data['aBX'].abs().max(),2), 'm/s^2')
# the second when max aBX happens
second = data['aBX'].abs() == data['aBX'].abs().max()
second = second.index[second==True].tolist()
second = second[0]*0.01
print('It occurs at',second,'seconds')
```

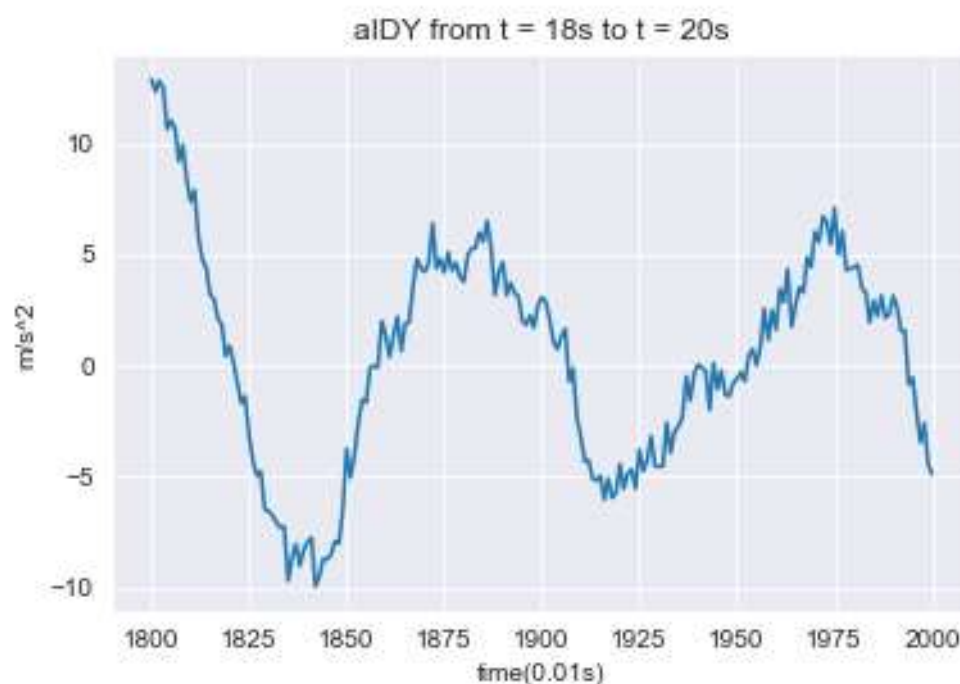
The maximum absolute value of aBX is 22.36 m/s².

It occurs at 18.06 seconds.

(c)

```
In [37]: #Inter-story acceleration aIDV
#Define as aIDV = aBV - aAV = ((A5 + A6) - (A1 + A2))/2
data['aIDV'] = ((data['A5']+data['A6'])-(data['A1']+data['A2']))/2
data_18_to_20s = data.loc[1800:2000]
aIDV_plot = sns.lineplot(data = data_18_to_20s['aIDV'])
sns.set_style("darkgrid")
aIDV_plot.set_xlabel("time(0.01s)")
aIDV_plot.set_ylabel("m/s^2")
aIDV_plot.set_title("aIDV from t = 18s to t = 20s")
aIDV_plot.figure.savefig("aIDV from t = 18s to t = 20s.png")
```

```
In [16]: print('The maximum absolute value of aIDV is', round(data['aIDV'].abs().max(),2), 'm/s^2')
```



The maximum absolute value of aIDY is 13.81 m/s².

(d)

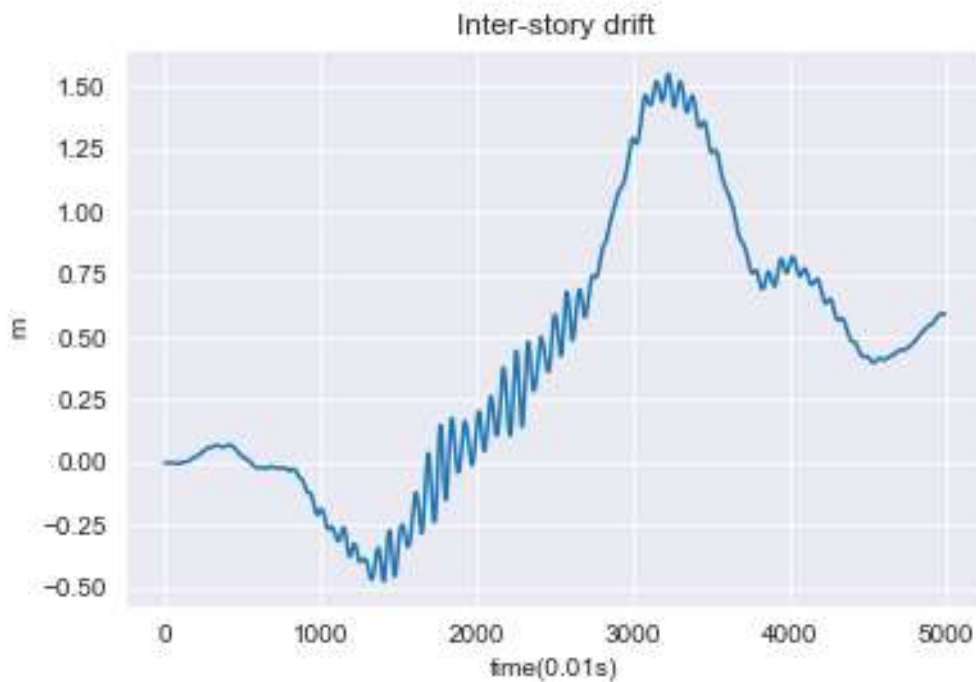
```
In [38]: #inter-story velocity = integral of the inter-story acceleration
data['int_sto_v'] = data['aIDY']*0.01
for i in range(1,5000):
    data['int_sto_v'][i] = data['int_sto_v'][i-1] + data['aIDY'][i]*0.01

int_sto_v_plot = sns.lineplot(data = data['int_sto_v'])
sns.set_style("darkgrid")
int_sto_v_plot.set_xlabel("time(0.01s)")
int_sto_v_plot.set_ylabel("m/s")
int_sto_v_plot.set_title("inter-story velocity")
```

```
In [41]: #inter-story drift = integral of the inter-story velocity
data['int_sto_drift'] = data['int_sto_v']*0.01
for i in range(1,5000):
    data['int_sto_drift'][i] = data['int_sto_drift'][i-1] + data['int_sto_v'][i]*0.01

int_sto_drift_plot = sns.lineplot(data = data['int_sto_drift'])
sns.set_style("darkgrid")
int_sto_drift_plot.set_xlabel("time(0.01s)")
int_sto_drift_plot.set_ylabel("m")
int_sto_drift_plot.set_title("inter-story drift")
int_sto_drift_plot.figure.savefig("Inter-story drift.png")
```

```
In [32]: print('The maximum absolute value of inter-story drift without noise is', round(data['int_sto_drift'].abs().max(),2), 'm')
```



The maximum absolute value of inter-story drift is 1.55 m.

(e)

```
In [24]: data_from_0_to_5s = data.loc[0:5000]
data_a1_from_0_to_5s = data_from_0_to_5s['A1'] # extract first five seconds data from A1
print('The standard deviation of the sensors noise is', round(data_a1_from_0_to_5s.std(),2)) #find and print the std of A1 from 0 to 5s
```

The standard deviation of a1 sensor's noise is 0.61.

(f)

If we consider all sensor has the same noise as a1,

The formula to derive for σ_{IDY} is $\sigma_{IDY}^2 = \sum_{k=1}^m a_k^2 \sigma_k^2 = \left(\frac{1}{2}\right)^2 * \sigma_{a5}^2 + \left(\frac{1}{2}\right)^2 * \sigma_{a6}^2 +$

$$\left(-\frac{1}{2}\right)^2 * \sigma_{a1}^2 + \left(-\frac{1}{2}\right)^2 * \sigma_{a2}^2 = \frac{1}{4} * 0.61^2 * 4 = 0.3721$$

$$\sigma_{IDY} = 0.610$$

If we calculate the noise of all sensors individually,

the standard deviation of a2 noise is 0.59

the standard deviation of a5 noise is 0.63

the standard deviation of a6 noise is 0.64

The formula to derive for σ_{IDY} is $\sigma_{IDY}^2 = \sum_{k=1}^m a_k^2 \sigma_k^2 = \left(\frac{1}{2}\right)^2 * \sigma_{a5}^2 + \left(\frac{1}{2}\right)^2 * \sigma_{a6}^2 +$

$$\left(-\frac{1}{2}\right)^2 * \sigma_{a1}^2 + \left(-\frac{1}{2}\right)^2 * \sigma_{a2}^2 = 0.3817$$

$$\sigma_{IDY} = 0.617$$

(g)

```

In [25]: my_cols = ["A1", "A2", "A3", "A4", "A5", "A6", "A7", "A8"]
         data_no_noise = pd.read_csv('SeismicResponse_noNoise.txt', names=my_cols, delimiter='\t')

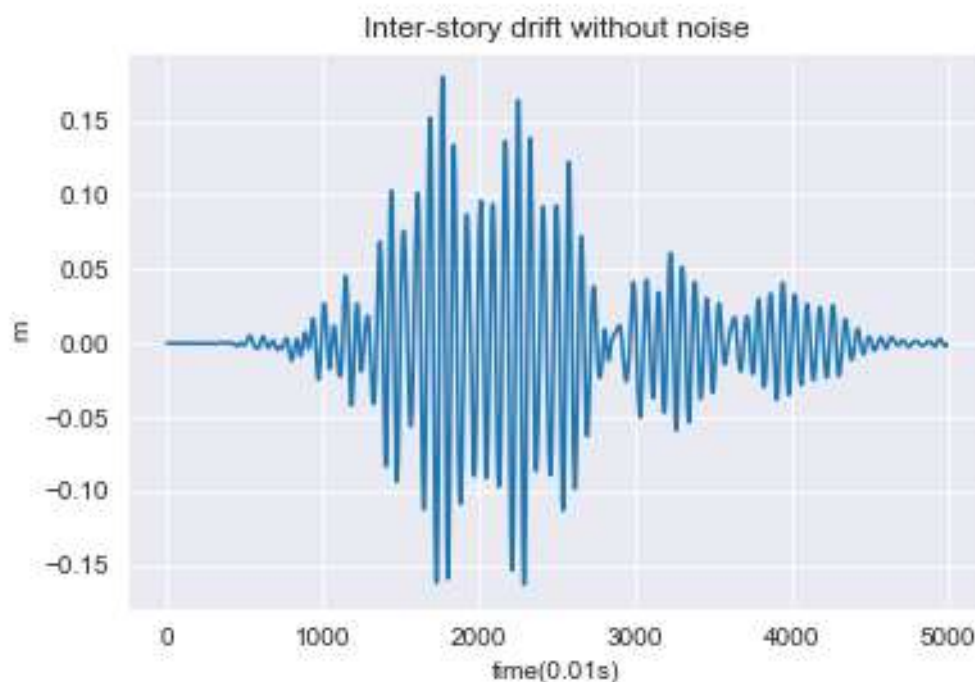
In [27]: # find the value of inter-story drift without noise
         data_no_noise["aIDV"] = ((data_no_noise["A5"]+data_no_noise["A6"])-(data_no_noise["A1"]-data_no_noise["A2"]))/2
         data_no_noise.head()
         data_no_noise["int_sto_v"] = data_no_noise["aIDV"]*0.01
         data_no_noise.head()
         for i in range(1,5000):
             data_no_noise["int_sto_v"][i] = data_no_noise["int_sto_v"][i]+data_no_noise["int_sto_v"][i-1]

         data_no_noise["int_sto_drift"] = data_no_noise["int_sto_v"]*0.01
         for i in range(1,5000):
             data_no_noise["int_sto_drift"][i] = data_no_noise["int_sto_drift"][i]-data_no_noise["int_sto_drift"][i-1]

In [28]: int_sto_drift_no_noise_plot = sns.lineplot(data = data_no_noise["int_sto_drift"])
         sns.set_style("darkgrid")
         int_sto_drift_no_noise_plot.set_xlabel("time(0.01s)")
         int_sto_drift_no_noise_plot.set_ylabel("m")
         int_sto_drift_no_noise_plot.set_title("Inter-story drift without noise")
         int_sto_drift_no_noise_plot.figure.savefig("inter-story drift without noise.png")

In [33]: print("The maximum absolute value of inter-story drift without noise is", round(data_no_noise["int_sto_drift"].abs().max(),2), 'm')

```



The maximum absolute value of inter-story drift without noise is 0.18 m.

The signal from this plot is without noise, therefore, the true value of the of inter-story drift can be obtained.

In (d), calculating the inter-story drift requires four sensor data, when all four sensors include noise, the real value of the inter-story drift is distorted.

(h)

In (c) we are first calculating inter-story acceleration for every time instance, then we find the time instance with the highest absolute value. The second method in the

prompt grabs the maximum of aBY and aAY from different time instances and subtract to find the difference. We will only get the same result from the two methods when the two maximum happens in the same time instance. The more consistent method would be method provided by (c).

(i)

Yes, we can obtain building movement from sensor a₄ and a₈. For example, to calculate θ_A and θ_B we can use this formula instead.

$$\theta_B(t) = \frac{a_8(t) - a_7(t)}{L_y}$$

$$\theta_A(t) = \frac{a_4(t) - a_3(t)}{L_y}$$

No, I don't think we can obtain a more accurate result, because when involving more sensors that record noise into our calculations, we will bring about more uncertainty into our results.

3. Uncertainty Propagation (25%)

$$\mu = 0.2 * 150 - 3 * 10 + 30 = 30MPa$$

$$\sigma = \sqrt{0.2^2 * 40^2 + 3^2 * 2^2} = 10$$