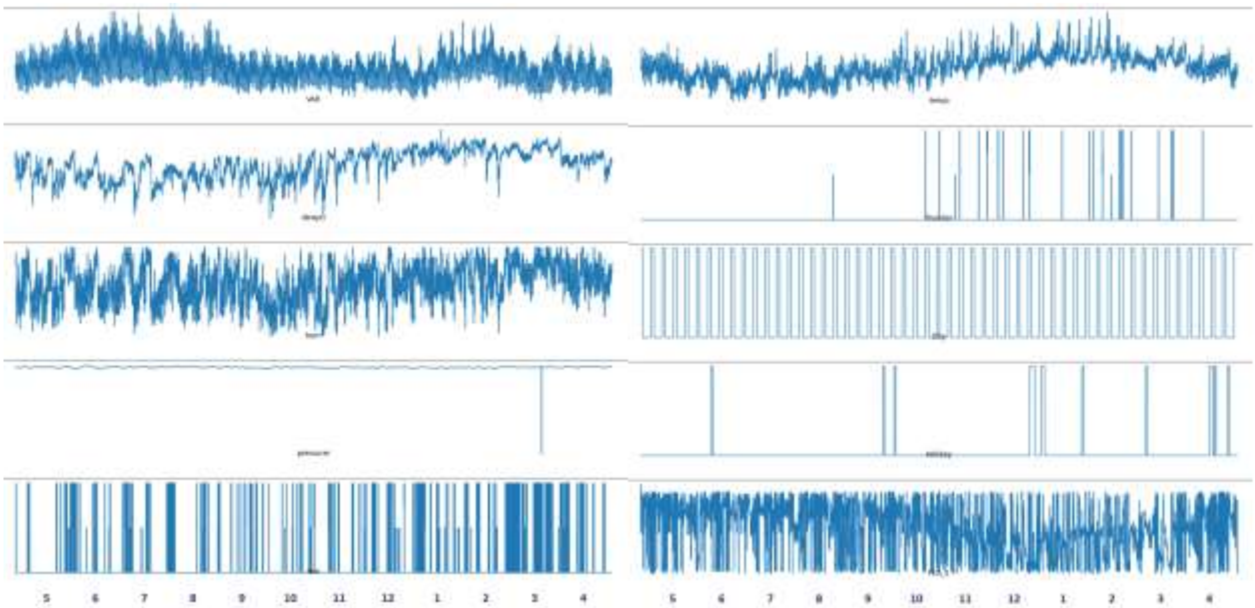
Exploratory data analysis

Let's have a look at the data to see

VAR
The dataset we are working with contains the variable VAR, as well as weather details .

Let's a have a look at each variable individually and see if we can spot any patterns.



tempi
Let's have a look at temperature (tempi), first. Apparetnly, temperature tends to be the lowest in June, July and August, and the highest in January and February. This means the data was probably collected from a country in the south hemisphere, possibly Australia.

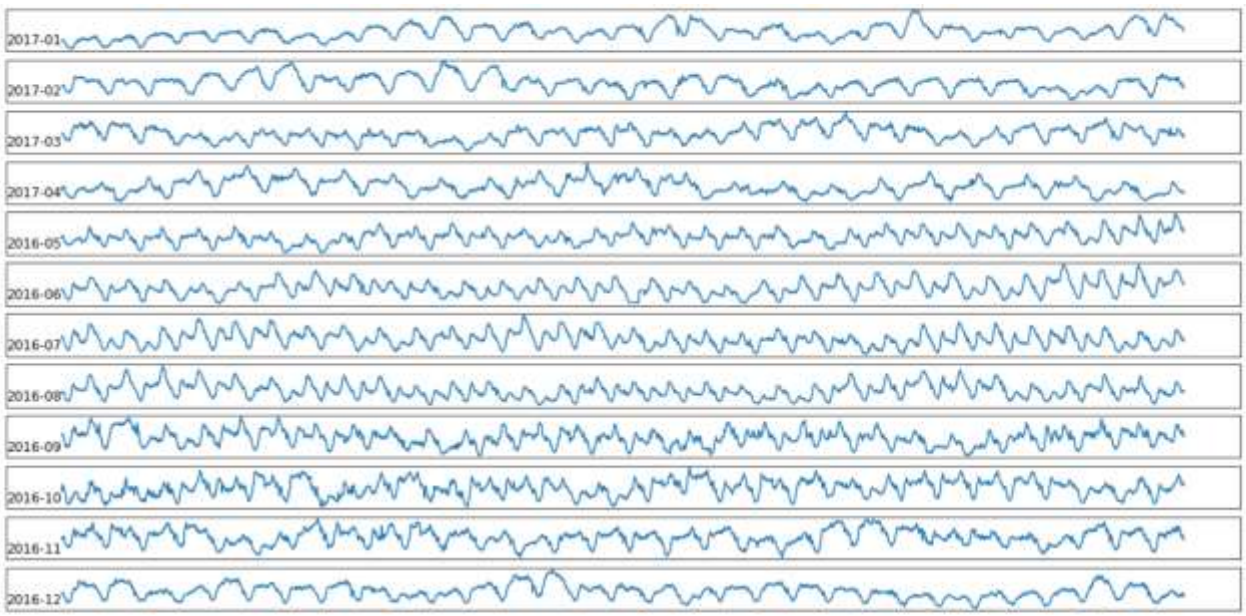
VAR
This makes sense when we look at power consumption (VAR), which is the highest between June and August – cold months during which heating is required – as well as January and February, likely due to high air conditioning usage. On the other hand, in spring and fall, when the temperatures are mild, power consumption is moderate.

hum, dewpti, thunder
High humidity (hum) and dew point (dewpti) values as well as thunder frequency are consistent with high temperatures in warmer months, which may suggest humid subtropical climate.

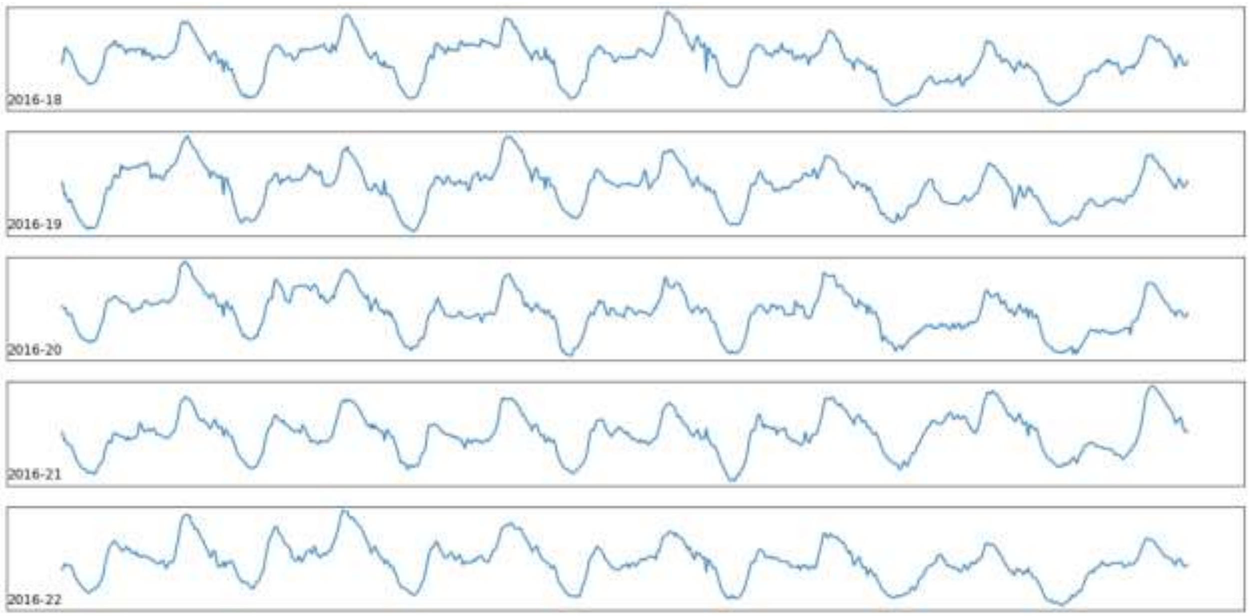
pressure
There appears to be a data issue with pressure in March. A good way to deal with missing or inaccurate values is to replace them with estimates. This will require inspecting pressure data in more detail but one easy way to deal with the issue would be to use the average pressure value at the same time of the day, from a few days before and after the corrupted value occurred.

For now, let's have a closer looks at power consumption (VAR), since it's the value we are modeling.

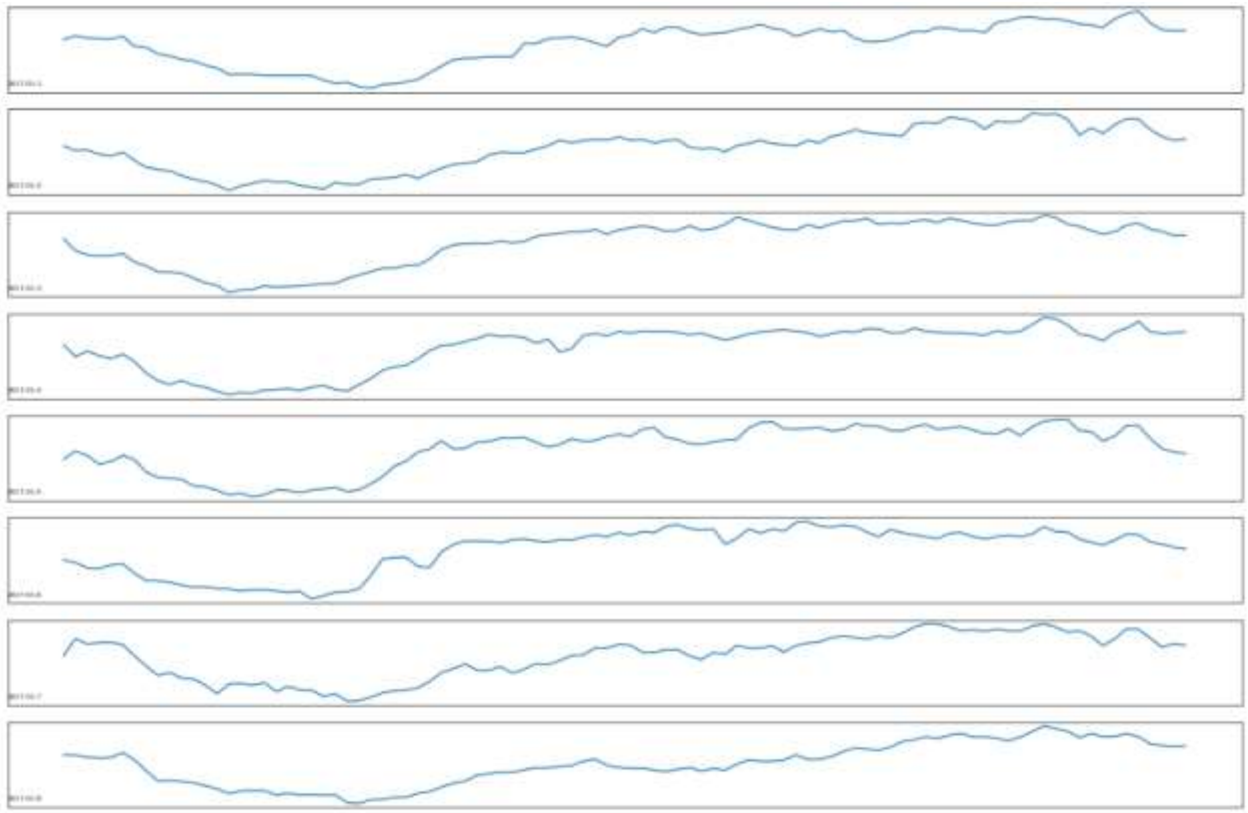
Zooming into individual months reveals a daily pattern: most likely high usage during daytime and low usage at night. Interestingly, colder months are characterized by a double daily hump. The reason is not clear at this point; one explanation could be that that in colder months people consume more power in the morning, as the get ready for work (first daily peak) and in the evening, as they turn the heat up upon returning home (second daily peak). On the other hand, warmer months miss the second peak because the temperature gets lower in the evening anyway, limiting the need for air conditioning.



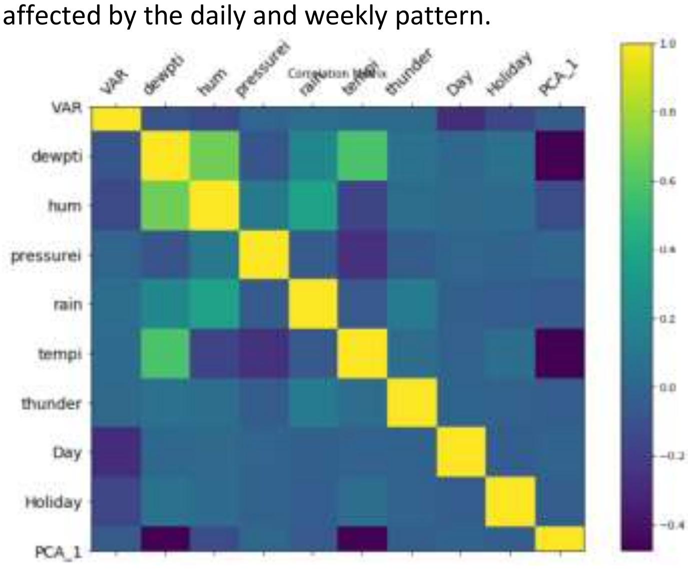
Looking even closer, at weekly data, here's a good question to ask: Is there a weekly pattern? For example, are weekends different than week days? There is no clear answer yet. There might be a subtle pattern but it's not easy to recognize with the naked eye.



Finally, the daily view reveals how power consumption fluctuates depending on time of day. We notice that the beginning of the day (left side of the chart) starts off quite high. This is leftover usage from the day before, since many adults remain active at least until midnight while entertaining, doing chores, or simply hanging out with lights switched on. Between 2-6 am the power usage dips for the night, then picks up again in the morning and reaches it's maximum around noon (middle of the chart). From there, it stays at a relatively high level until midnight (right side), as household members go about their day. This hourly pattern is very strong and will play a significant role in determining VAR values throughout the day.



A quick look at correlation matrix sheds more light on our knowledge/findings so far. First of all, there is a strong negative correlation between power consumption and working/non working days (Day and Holiday variables). Specifically, power consumption is lower during weekends and bank holidays. On the other hand there is no direct correlation between power consumption and temperature, despite our earlier observations. This is because the relationship between the two variables is indirect; it's strongly affected by the daily and weekly pattern.



Problems like this, with intricate variable relationships, where if-then rules are hard to grasp intuitively, are good candidates for deep learning models.

Convolutional Neural Networks

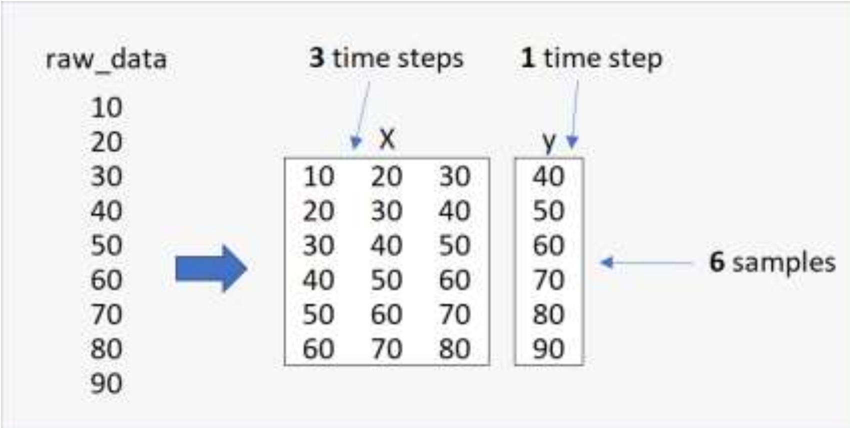
Traditionally, LSTM (Long short-term memory) networks are used for modeling time series. In this exercise, however, we will explore CNN, or Convolutional Neural Networks.

Data transformation

Before we get started, we need to transform out dataset in a way that a CNN can understand it. Specifically, CNN calls for a three-dimensional dataset, which consists of so called samples, time steps, and features. To develop an intuition on how this thee-dimensional structure might look, let's first translate the time series problem into a supervised machine learning problem.

Imagine a simple string of numbers, 10, 20, 30 and so on, until 90. How can we represent it as a classical supervised machine learning problem with an **X** matrix containing features, and a **y** vecor containing the target variable? One way would be to take each number, or observation, and notice the values that came before it.

For example, we know that before 90, there is 60, 70 and 80. We could choose to model this relationship as 60, 70 and 80 being our input, and 90 – output data. To use CNN terminology, we have just constructed a single **sample**, with 3 input **time steps** and 1 output **time step**. But here's the tricky part. The three time steps in the input, are a single **feature**. Remember we decomposed a single time series into multiple steps, but these steps are still part of a single feature.



Here's Python code that performs the transformation:

```
# split a univariate sequence into samples
def split_sequence(sequence, n_steps):
    X, y = list(), list()
    for i in range(len(sequence)):
        # find the end of this pattern
        end_ix = i + n_steps
        # check if we are beyond the sequence
        if end_ix > len(sequence)-1:
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return array(X), array(y)
```

```
# define input sequence
raw_seq = [10, 20, 30, 40, 50, 60, 70, 80, 90]
# choose a number of time steps
n_steps = 3
# split into samples
X, y = split_sequence(raw_seq, n_steps)
# summarize the data
print("X shape: " + str(X.shape))
print("y shape: " + str(y.shape))
for i in range(len(X)):
    print(X[i], y[i])
```

```
X shape: (6, 3)
y shape: (6,)
[10 20 30] 40
[20 30 40] 50
[30 40 50] 60
[40 50 60] 70
[50 60 70] 80
[60 70 80] 90
```

For a more elaborate example, let’s imagine three time series as follows.

X		y
10	15	25
20	25	45
30	35	65
40	45	85
50	55	105
60	65	125
70	75	145
80	85	165
90	95	185

Our goal is to predict the value of y, given X. For instance, given that X = 95 and 105, what will be the value of y?

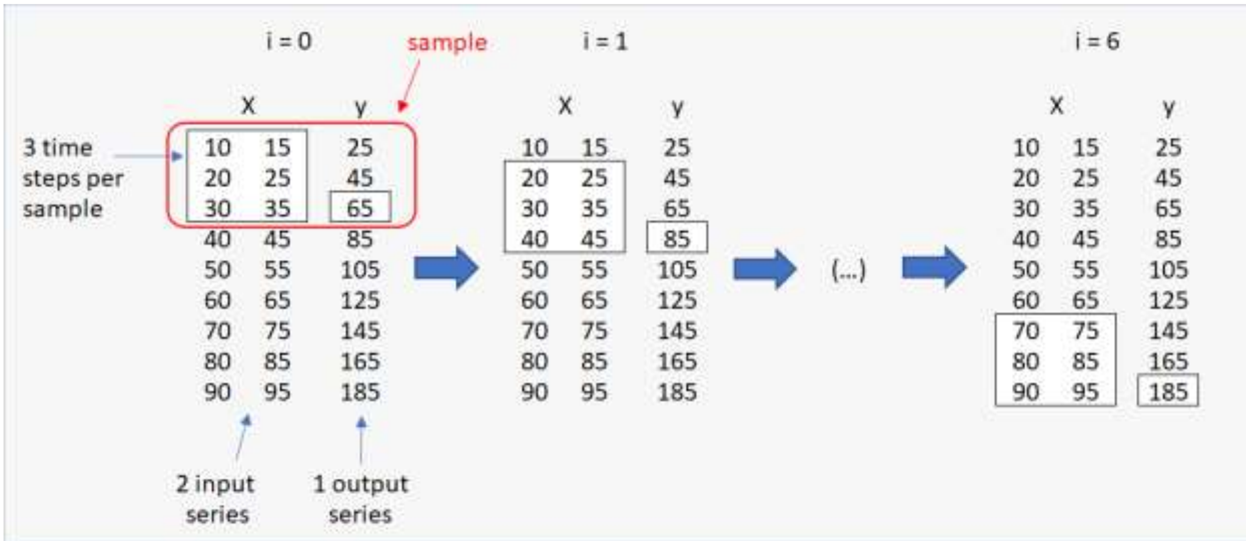
A keen eye will observe that y is the sum of X. 10 + 15 = 25. 20 + 25 = 45 and so on, so 90 + 95 = 195. Of course, in real life, variable relationships can be much harder to spot, and we want to construct a reliable mechanism that will predict that 95 and 105 are associated with the number 195, without knowing the nature of the relationship.

We could use each row as a single observation and try to predict the outcome, but this approach doesn’t take advantage of the temporal structure of the data. So let’s expand the concept of an observation from a single row to, say, three.

We will build our first **sample** that associates the number 65, with not only 30 and 35, but also the two preceding pairs, 20 and 25, as well as 10 and 15. The input of our sample will therefore consist of two **time series** and three **time steps.**

To construct more samples, we simply slide the window down, until we reach the end of the dataset. We will then have created 7 samples.

The dimensionality of our CNN-ready input dataset is now [7, 3, 2]. That is 7 samples, 3 time steps, 2 features.



Having performed an exploratory analysis of the available data, we move on to constructing