## Preprocessing

A subset of the full data was loaded for exploration and to determine the best steps for preprocessing.

As a first step, the column holding the datetime information was converted from the codified format to standard datetime format following the convention descibed in the dataset documentation.

The data also comes in a random list of meter Ids with their corresponding timestamp code and power consumption for each 30-minute block of time. To make the data organised by meterId and power consumption by timestamp, the data was grouped by meter ID and power consumption, using the timestamp column as the new index.

The data was then filtered to remove meter Ids tagged as control meters based on the dataset documentation to ensure that the final model is built using only actual customer data.

After filtering the data, a plot of the distrubution of missing values for each meter ID revealed that a number of meters had excessive missing data points. Thus, to solve for this, the data was filtered once again to remove meter Ids with missing value counts above 20%. This threshold was chosen because it had minimal effect on the statistical measures of central tendency of the data.

Based on the chosen approach for the project, all meter ID data for each timestamp would be summed up to give one representative power consumption figure for the entire sample space for each timestamp. To this end, it was observed that filling up the remaining missing values in each separate column by using a forward filling strategy before summing up only changed the prefilled sum by values which could be considered as margin of error values. Thus, the data was simply naively summed after filtering for columns with excessive missing values to reduce the computational cost of the preprocessing pipeline.

Afterwards, the data was then scaled with a min-max scaler to put all values between 0 and 1, which is easier for deep learning models to work with.

Finally, the data was reshaped in a sliding window approach which allows the model to consider the previous 1440 timesteps in order to predict the next timestep.

## Model Building, Training and Evaluation

An AutoEncoder LSTM model was used with then encoder and decoder layers being LSTM layers. A number of variations with single direction and bi-directional encoder input and decoder output layer, as well as variation in number of encoder and decoder layers and layer sizes, including the use of normalisation and attention layers were experimented with.

It was observed that the model with bi-directional layers and larger layer sizes was able to capture the pattern of the data and yield the best results. The models were evaluated on RMSE scores and then NRMSE for the final model to enable comparison with works in literature.

[Side Quest]

As a side experiment, it was observed that the way the training data is structured has an impact on how well the model is able to predict power consumption hypothetically outside of the dataset. When the data is structured to predict a few timesteps into the future, it is able to accurately predict a few steps outside the dataset but predicton quality quickly falters as it begins to make predictions on its own predictions, which would in theory lead to error propagation with multiple levels of abstraction from real data.

On the other hand, when the model is trained to predict a larger number of timesteps into the future, the model would perform well for the first 2 or 3 \* number\_of\_timestamps trained for, and then subsequent predictions would begin to phase out of the expected values, in a slipping of it’s timesamp representation ahead of the expected outcome.