Synthetic Time Series

Given:

A set of smart meter readings, composed of mRID, timestamp and value (Wh) in the <keyspace>.measured_value table in Cassandra, we can decompose the timestamp into the following components:

- tick - the integer number of the (15 minute) interval (0 <= tick <= readings per day)
- day - the integer day of the week (1 = Sunday, 2 = Monday ... 7 = Saturday)
- week - the integer week number in a year (1 <= week <= 52)
- average - the average smart meter reading value over all readings for an mRID (average = sum (readings) / count (readings)) stored in the <keyspace>.measured_value_stats table in Cassandra

providing a data set with rows like:

    (mRID, tick, day, week, average, value)

Assuming:

- one year is like any other (since the meter readings we have don't quite cover a full year, this is not a problem)
- the average is a unbiased estimator of the yearly energy consumption (kWh) of a customer, for example 3000kWh for a typical apartment, so we can compute the average from the yearly energy consumption, or vice-versa, using:
  - average = (yearly energy consumption) * 1000.0 / 365.25 / (readings per day) Wh
  - yearly energy consumption = average * (readings per day) * 365.25 / 1000.0 kWh
- the smart meter readings have equal intervals between them (15 minutes)

Compute:

A decision tree model for regression using machine learning by:

- partitioning the data set into training (70%) and testing (30%) records
- using a DecisionTreeRegressor model with
  - label = value
  - feature vector = (tick, day, week, average)

Then one can use the resulting model to generate new smart meter readings. The signature for this functionality currently looks like this:

```
      def generateTimeSeries (mrid: String, start: Calendar, end: Calendar,
period: Int, yearly_kWh: Double): Unit
```

where:

- mrid is the rdf:ID of the EnergyConsumer for which the generated time series is desired
- start and end are the first and last timestamps of the generated time series
- period is the number of milliseconds between readings (15 * 60 * 1000 = 900000)
- yearly_kWh is the EnergyConsumer energy consumption on a yearly basis (kWh)

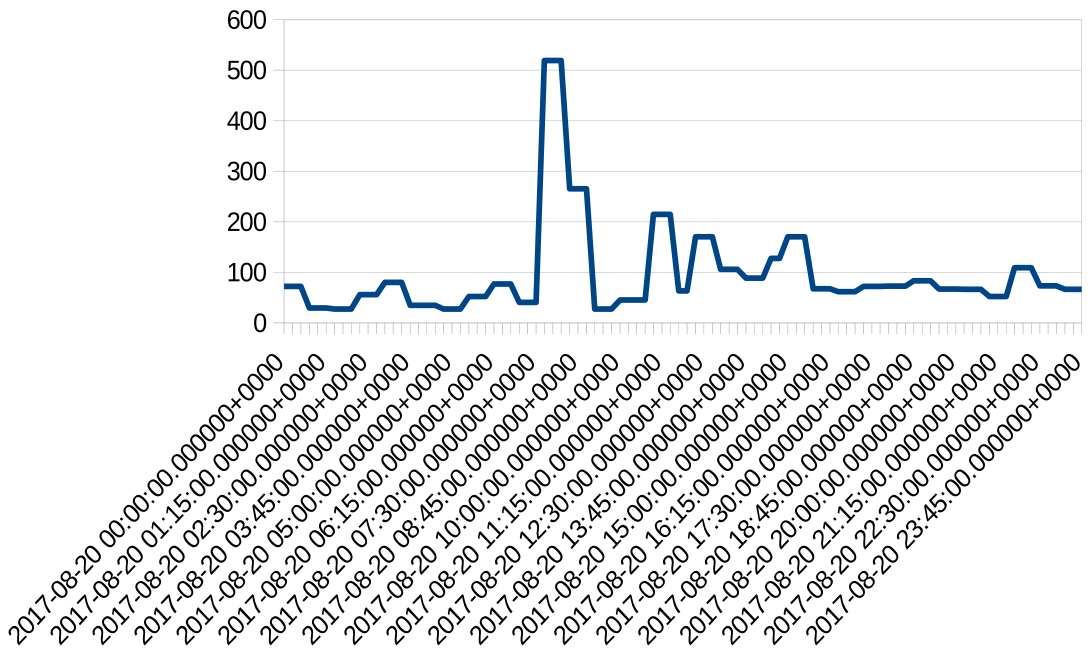
The average value is computed from the yearly kWh.
Then for each timestamp between start and end by period, the tick, day, week, and average values are assembled and submitted to the model.
It produces a prediction - the estimate of the smart meter reading - for each timestamp.
The results are written to the <keyspace>.synthesized_value table in Cassandra.

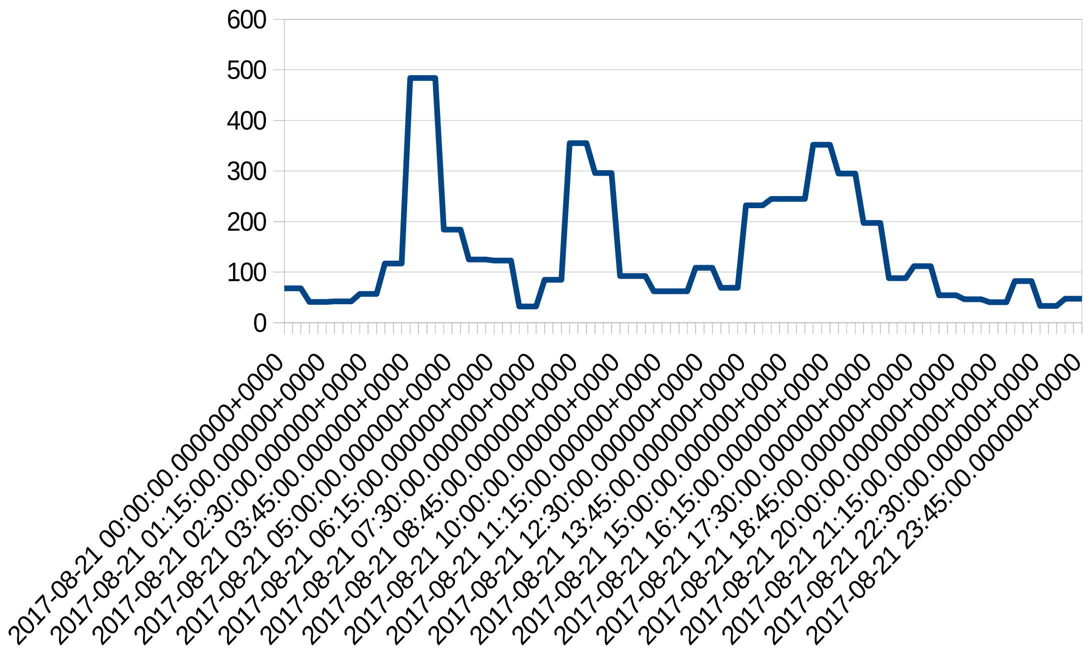
Results:
A sample over three days is shown below and attached as a spreadsheet for model parameters:

- maxDepth = 20
- maxBins = 32
- minInfoGain = 0.0

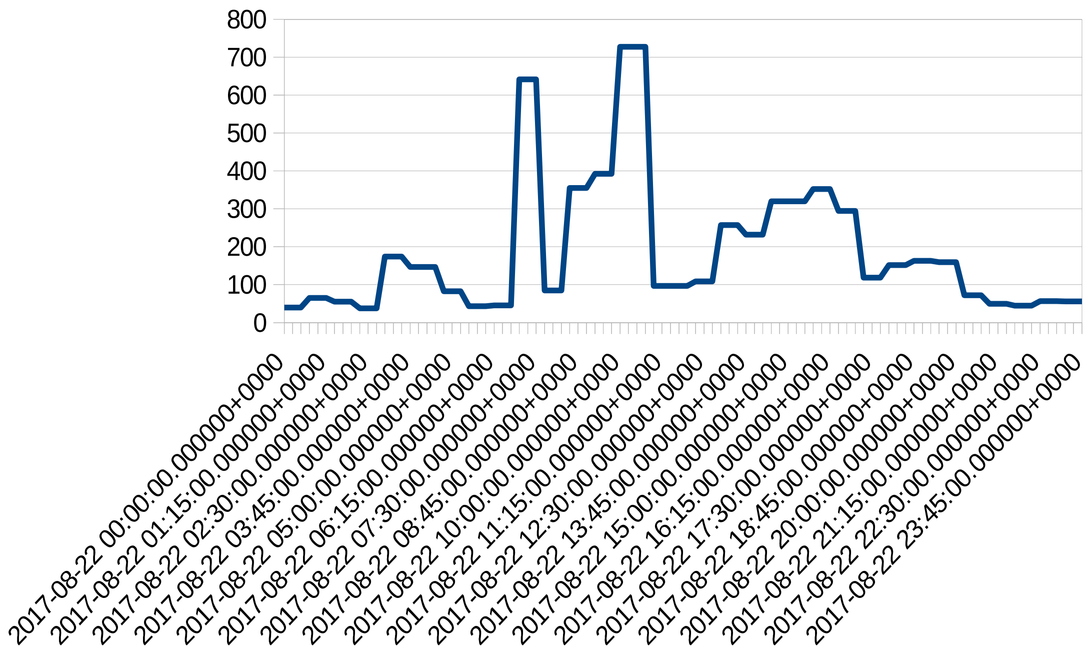
applied over a subset of the available smart meter readings from eastern Switzerland.
The root mean square (RMS) error for the model over the testing values (30% held back from training) is 585Wh. This is a rather large error, and has the intuitive meaning that the predicted smart meter reading value is on average 500Wh different than the actual reading. For a 15 minute interval, this is the equivalent of two microwave ovens continually on or not.
This may not be a very good metric for evaluating the result. Perhaps a better metric is how the average over all values compares to the desired yearly consumption, which for one test case is 184.3 vs. a desired average of 205.5 or 10%. A visual inspection of the shapes of the curves also seems to agree with the typical profile.

Optimization:
The model can also be used in a hyperparameter tuning run to pick the "best" values.
This is currently in process.





600
500
400
300
200
100
0
2017-08-21 00:00:00.000000+0000
2017-08-21 01:15:00.000000+0000
2017-08-21 02:30:00.000000+0000
2017-08-21 03:45:00.000000+0000
2017-08-21 05:00:00.000000+0000
2017-08-21 06:15:00.000000+0000
2017-08-21 07:30:00.000000+0000
2017-08-21 08:45:00.000000+0000
2017-08-21 10:00:00.000000+0000
2017-08-21 11:15:00.000000+0000
2017-08-21 12:30:00.000000+0000
2017-08-21 13:45:00.000000+0000
2017-08-21 15:00:00.000000+0000
2017-08-21 16:15:00.000000+0000
2017-08-21 17:30:00.000000+0000
2017-08-21 18:45:00.000000+0000
2017-08-21 20:00:00.000000+0000
2017-08-21 21:15:00.000000+0000
2017-08-21 22:30:00.000000+0000
2017-08-21 23:45:00.000000+0000


800
700
600
500
400
300
200
100
0
2017-08-22 00:00:00.000000+0000
2017-08-22 01:15:00.000000+0000
2017-08-22 02:30:00.000000+0000
2017-08-22 03:45:00.000000+0000
2017-08-22 05:00:00.000000+0000
2017-08-22 06:15:00.000000+0000
2017-08-22 07:30:00.000000+0000
2017-08-22 08:45:00.000000+0000
2017-08-22 10:00:00.000000+0000
2017-08-22 11:15:00.000000+0000
2017-08-22 12:30:00.000000+0000
2017-08-22 13:45:00.000000+0000
2017-08-22 15:00:00.000000+0000
2017-08-22 16:15:00.000000+0000
2017-08-22 17:30:00.000000+0000
2017-08-22 18:45:00.000000+0000
2017-08-22 20:00:00.000000+0000
2017-08-22 21:15:00.000000+0000
2017-08-22 22:30:00.000000+0000
2017-08-22 23:45:00.000000+0000