D2.1 Data Models – Initial Version

Lead Partner:	AAU
Version:	1
Dissemination Level:	Public
Work Package:	WP2
Due date:	31/05/2021
Submission Date:	31/05/2021

Abstract:
This deliverable presents how *models* can be used to compress large amounts of time series data efficiently. This is needed to be able to store and analyse the huge amounts of sensor data available from modern wind turbines and solar panels. The currently available MORE data from ENGIE is then investigated in details. It is found that many parts of the time series can be modelled by a quadratic function. Based on this a new model type is proposed to be added to the ModelarDB model-based Time Series Management System (TSMS). Further, it is found that many of the time series can be derived from other timeseries. Based on this it is proposed how to avoid physical storage of time series that can be derived from other time series. The ENGIE dataset is also compressed by use of models in ModelarDB and it is found that even with lossless compression, the required storage is reduced by 19% and with an error bound of 10%, the reduction is bigger than 76%, compared to the original data size.

Document Revision History

Date	Version	Author/Editor/Contributor	Summary of main changes / Status
05/03/2021	V1	Nguyen Thi Thao Ho	First version
19/05/2021	V2	Christian Thomsen Torben Bach Pedersen Søren Kejser Jensen Nguyen Thi Thao Ho	More content added
21/05/2021	V3	Christian Thomsen Torben Bach Pedersen Søren Kejser Jensen Nguyen Thi Thao Ho	AAU internal review
25/05/2021	V4	Seshu Tirupathi (IBM) Manolis Terrovitis (Athena)	MORE internal review
30/05/2021	V5	Christian Thomsen Søren Kejser Jensen Nguyen Thi Thao Ho	Addressed comments from MORE internal review
31/05/2021	V6	Christian Thomsen Søren Kejser Jensen Nguyen Thi Thao Ho	Final version

Disclaimer

Copyright

Contents

1 Introduction

WP2 will create models, techniques, data structures, and algorithms for efficient and effective ingestion, querying, analytics, and transfer of sensor data points as models. Large amounts of high-frequency sensor data will first be ingested to and become represented as models at the edge nodes (i.e., RES installations) and will then be transferred to the data center.

WP2 is led by AAU and further Athena RC, InAccess, IBM, Laborelec, and ModelarData participate.

2 Task 2.1: Data modelling and summarization

Using the models provided by ModelarDB [1-4] as a base, this task will develop the models and techniques to be used for modelling and summarization of streaming sensor data. Models will natively capture various inaccuracies in the data and offer parameterization for users to specify trade-offs between query accuracy and storage needs. This document describes the first version of the models used in MORE. It will later be followed by the final version based on the feedback provided by the development of the analytic modules. This deliverable will thus investigate the currently available ENGIE data in MORE, and propose model types that can be used to represent it.

3 Motivation

High quality sensors with wired power and connectivity are used to monitor modern wind turbines and solar panels. The high-quality time series produced by these sensors have a regular sampling interval, and the few incorrect and out-of-order data points that do occur are corrected by existing cleaning procedures. However, while the owners and manufacturers of wind turbines would like to sample the sensors at a high frequency to improve the monitoring and allow for more detailed analysis, it is infeasible with current technology due to the large the amount of storage required. Currently, it is thus typical to only store simple aggregates over a fixed window size, for example 1–10 minute averages. An example is shown in Figure 1, where twelve data points are reduced to just a single average. It is clear from this example that reducing each sub-sequence to just a single data point removes all outliers and fluctuations in the sub-sequence which could have indicated problems with the monitored entity.

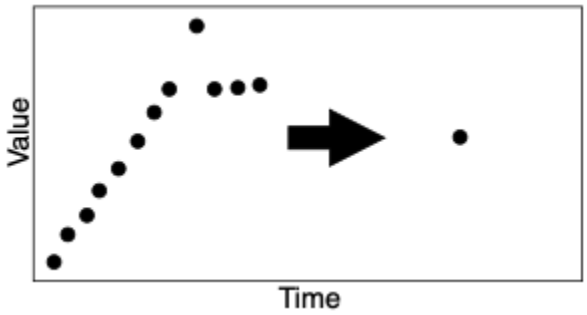


Figure 1 Compression of time series as simple aggregates

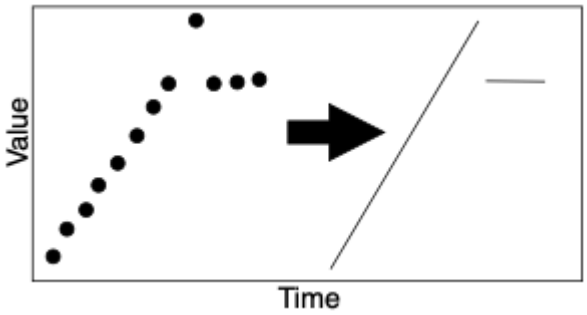


Figure 2 Model-based compression of time series

To reduce both the amount of storage required and the query processing time, a model- based approach can be used to store time series with the outliers and fluctuations intact. We use the term *model* for any representation from which a time series can be reconstructed within a user-defined error bound (possibly zero). As an example, a linear function given by the equation $v = a * t + b$ can efficiently represent an increasing, decreasing or constant sub-sequence of a time series using only the two floating-point values a and b. However, as the structures of time series often change over time, the most appropriate model type to use can be different for each sub-sequence, so multiple model types should be used even for a single time series. In addition, time series are often correlated. For example, two co-located temperature sensors could produce similar values and thus be positively linearly correlated. These correlated time series should be compressed together to further reduce the amount of storage required. The benefit of model-based storage for time series is illustrated in Figure 2. The values of the sub-sequence from Figure 1 are very efficiently represented using a linear function and a constant function which in total only requires three floating- point values while preserving the structure of the sequence.

As stated, the high-quality time series produced in the energy domain have a regular sampling interval, and the few incorrect or out-of-order data points that do occur are corrected using existing cleaning procedures. For this reason, the only anomaly that must be considered when ingesting these time series are missing data points. While interpolation can be utilized to approximate the missing values, their absence can indicate problems with the monitored entity. For this reason, interpolation of missing values should be an explicit decision performed by a data analyst, and not an implicit action performed by the Time Series Management System (TSMS). ModelarDB [1-4] is a model-based TSMS which will be used and further developed in the MORE project.

4 Preliminaries

The following definitions and examples are based on [1-4] as the new data models build upon the work presented in these publications.

Definition 1 (Time Series): A time series TS is a sequence of data points, in the form of timestamp and value pairs, ordered by time in increasing order TS = $\langle(t_1, v_1), (t_2, v_2), \ldots\rangle$. For each pair (t_i, v_i), $1 \leq i$, the timestamp t_i represents the time when the value $v_i \in R$ was recorded. A time series TS = $\langle(t_1, v_1), \ldots, (t_n, v_n)\rangle$ consisting of a fixed number of n data points is a bounded time series.

Definition 2 (Regular Time Series): A time series TS = $\langle(t_1, v_1), (t_2, v_2), \ldots\rangle$ is considered regular if the time elapsed between each data point is always the same, i.e., $t_{i+1} - t_i = t_{i+2} - t_{i+1}$ for $1 \leq i$ and irregular otherwise.

Definition 3 (Sampling Interval): The sampling interval SI of a regular time series TS = $\langle(t_1, v_1), (t_2, v_2), \ldots\rangle$ is the time elapsed between each pair of consecutive data points in the time series $SI = t_{i+1} - t_i$ for $1 \leq i$.

As an example, the time series TS_e = ⟨(100,0.156),(200,0.139),(300,0.122), (400, 0.106), (500, 0.92) . . .⟩ is a regular unbounded time series with a 100 millisecond sampling interval. A bounded time series can be constructed from TS_e by extracting the data points with the timestamps $100 \leq t \leq 500$.

Definition 4 (Gap): A gap between a regular bounded time series $TS_1 = \langle(t_1, v_1), \ldots, (t_s, v_s)\rangle$ and a regular time series $TS_2 = \langle(t_e, v_e), (t_{e+1}, v_{e+1}), \ldots\rangle$ with the same sampling interval SI and recorded from the same source, is a pair of timestamps $G = (t_s, t_e)$ with $t_e = t_s + m \times SI$, $m \in N \geq 2$, and where no data points exist between t_s and t_e.

Definition 5 (Regular Time Series with Gaps): A regular time series with gaps is a regular time series, TS = $\langle(t_1, v_1), (t_2, v_2), \ldots\rangle$ where $v_i \in R \cup \{\perp\}$ for $1 \leq i$. All sub-sequences in TS of the form $\langle(t_s, v_s), (t_{s+1}, \perp), \ldots, (t_{e-1}, \perp), (t_e, v_e)\rangle$ where $v_s, v_e \in R$, are denoted as gaps $G = (t_s, t_e)$.

No gaps are present in TS_e as a value exists for all timestamps matching the sampling interval. However, TS_g = ⟨(100, 0.156), (200, 0.139), (400, 0.106), (500, 0.92) . . .⟩ has no value for t = 300 and is thus two regular time series separated by a gap. For simplicity, we say that multiple time series from the same source separated by gaps is one time series containing gaps. As the sampling interval is undefined for a time series with gaps, we define regular time series with gaps as a time series where all data points in a gap have the special value ⊥ indicating that no values were collected. As an example, TS_{rg} = ⟨(100, 0.156), (200, 0.139), (300, ⊥), (400, 0.106), (500, 0.92)...⟩ has a sampling interval of 100 milliseconds.

Definition 6 (Model): A model of a time series TS = $\langle(t_1, v_1), (t_2, v_2), \ldots\rangle$ is a function m. For each t_i, $1 \leq i$, m is a real-valued mapping from t_i to an estimate of the value v_i for the corresponding data point in TS.

Definition 7 (Model Type): A model type is pair of functions $M_T = (m_t, e_t)$. $m_t(TS, \varepsilon)$ is a partial function, which when defined for a bounded time series TS and a non-negative real number ε, returns a model m of TS such that $e_t(TS, m) \leq \varepsilon$. e_t is a mapping from TS and m to a non-negative real number representing the error of the values estimated by m. We call ε the error bound.

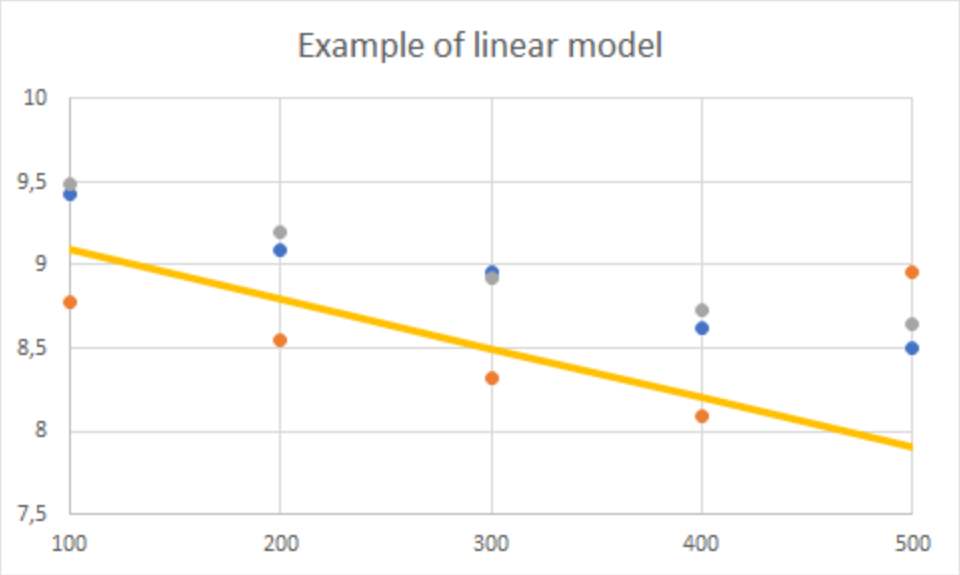
A model type (e.g., linear regression) determines the set of parameters required to create a specific model of that type for approximating the values of a time series. Models represent the values of a time series as the function m with the error of the representation within the error bound ε as computed by the model type's function m_t. We say that a model is fitted to a bounded regular time series, for example TS_b = ⟨(100, 0.156), (200, 0.139), (300, 0.122), (400, 0.106), (500, 0.92)⟩, when determining the parameters of a model using a model type. A single model may also be able to represent a group of time series within a given error bound if the time series have similar values.

Definition 8 (Time Series Group): A time series group TSG is a set of regular time series, possibly with gaps, TSG = {TS_1,...,TS_n}, where for TS_i, $TS_j \in$ TSG it holds that they have the same sampling interval SI and that t_{1i} mod SI = t_{1j} mod SI where t_{1i} and t_{1j} are the first timestamp of T_{Si} and T_{Sj}, respectively.

A time series group is restricted to only containing time series with the same sampling interval and aligned timestamps. This ensures that a data point is received from all time series in a group at each sampling interval unless gaps occur. If the correlated time series do not consist of approximately the same values, scaling can be applied to allow a single stream of models to represent the values of all time series in the group. By representing the data points from a time series group with only one model per dynamically sized sub-sequence, the compression ratio can be significantly increased compared to a model-based representation of each individual time series.

Definition 9 (Segment): A segment is a 5-tuple S = (t_s, t_e, SI, G_{ts} : TSG → $2^{\{t_s, t_s + \mathrm{SI}, \ldots, t_e\}}$, m) representing the data points for a bounded time interval of a time series group TSG. The 5-tuple consists of start time t_s, end time t_e , sampling interval SI, a function G_{ts} which for the TS ∈ TSG gives the set of timestamps for which v = ⊥ in TS, and where the values of all other timestamps are defined by the model m multiplied by a scaling constant $C_{TS} \in R$.

As an example of a segment we use the following three time series: TS_1 = ⟨(100, 9.43), (200, 9.09), (300, 8.96), (400, 8.62), (500, 8.50)⟩, TS_2 = ⟨(100, 8.78), (200, 8.55), (300,8.32), (400,8.09), (500,8.96)⟩, and TS_3 = ⟨(100, 9.49), (200, 9.20), (300, 8.92), (400, 8.73), (500, 8.65)⟩. These three time series are grouped together in a time series group and compressed together. The time series group can be represented using the linear function m = −0.003ti + 9.40. Using the uniform norm this model has an error of |8.96 − (−0.003 × 500 + 9.40)| = 1.06. If we assume that the error bound is one, a segment S = (100, 400, 100, G_{ts} = ∅, m = -0.003_{ti} + 9.40), 1 ≤ i ≤ 4, must be created in order for model-based representation to not exceed the error bound. The following plot shows the values of the three time series TS_1, TS_2, and TS_3 as well the linear model *m* (shown with a yellow line) that can represent them within the error bound in this segment S. Instead of storing the individual data points of the three time series, it is thus possible to store the parameters for S. Note that for later time stamps, other model types may be used in the segments representing them.



5 Existing model types in ModelarDB

The already existing open-source version of ModelarDB comes with support for three model types adapted to be able to compress time series groups [4]: PMC-Mean [5] for constant models, Swing [6] for linear models, and Gorilla [7] for lossless compression of floats. These relatively simple model types can already compress RES data very well [4]. ModelarDB is, however, extensible such that new model types can be added and then used by the system. In the following we will thus explore the data currently available in MORE to identify other model types that could be beneficial to use with such data and thus should be added to ModelarDB. When more datasets become available in MORE, more model types may be added.

6 Data analysis process

The goal of this data analysis process is to identify data models that can best summarize the considered time series data. We use datasets provided by ENGIE as a use case to demonstrate the analysis process which consists of multiple steps: (1) data exploration: provides an overview of the dataset, (2) meta data analysis: extracts useful information by analyzing metadata, (3) data visualization: represents data graphically to interpret trends, patterns, (4) analyzing repeated patterns such as motifs to improve data compression. We discuss in details each of the steps below.

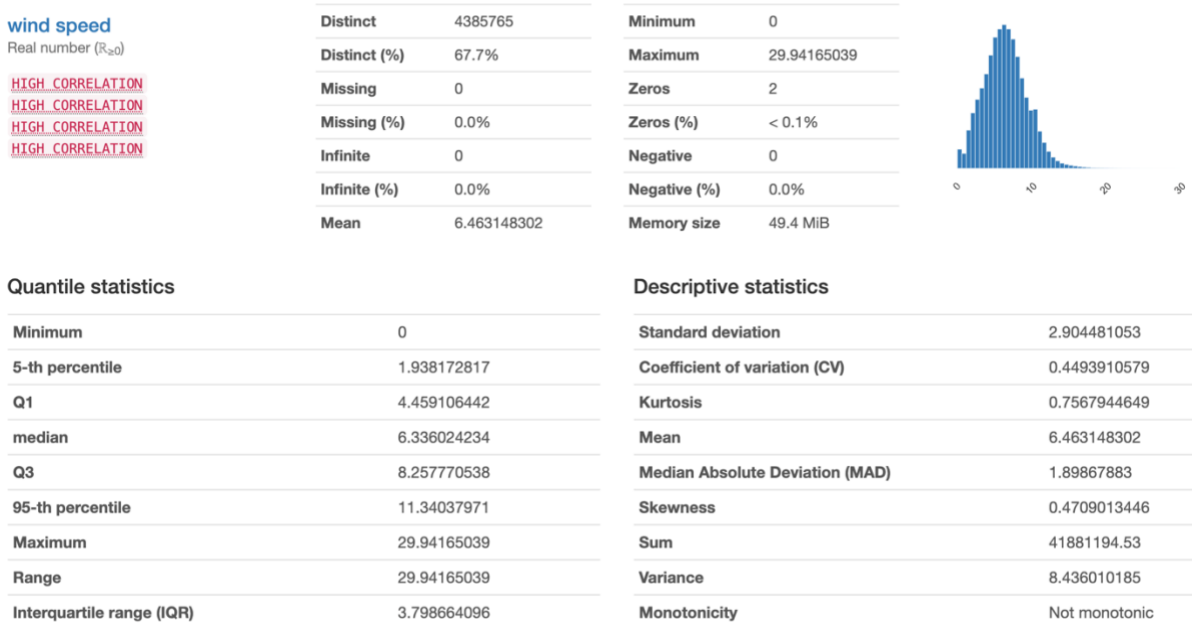
6.1 Data exploration

We first explore the provided ENGIE data to understand their general characteristics. ENGIE has provided historical data of 18 wind turbines from 4 wind parks. The data are provided in two different granularities: 2-seconds frequency and aggregated 10-mins frequency. We focus in this analysis the 2-seconds frequency granularity.

Two different formats of data are provided for the 2-seconds granularity: raw anonymized data and post treated data. While the post treated data have 10 variables with values have been corrected and preprocessed, the raw data have 138 different variables with many missing values. Furthermore, raw data

are sampled with irregular time interval. For this reason, we focus on post treated datasets and summarize the descriptive statistics of the main variables below. The discussion below is relied on the analysis of BEBEZE02_scada_high_frequency_part1.csv datafile.

Wind speed: the wind speed values are numeric measured in m/s. On the analyzed dataset, its values are within the range [0-30]. The wind speed distribution is slightly right-skewed and mesokurtic, with the skewness and kurtosis values are 0.47 and 0.75, respectively. The slightly heavy tail on the right represents strong wind speeds, which occur in less than 1% of the total records. The wind speed is also highly correlated with two other variables: rotor speed and active power.



wind speed

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	4385765
Distinct (%)	67.7%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	6.463148302

Minimum	0
Maximum	29.94165039
Zeros	2
Zeros (%)	< 0.1%
Negative	0
Negative (%)	0.0%
Memory size	49.4 MiB

Quantile statistics

Minimum	0
5-th percentile	1.938172817
Q1	4.459106442
median	6.336024234
Q3	8.257770538
95-th percentile	11.34037971
Maximum	29.94165039
Range	29.94165039
Interquartile range (IQR)	3.798664096

Descriptive statistics

Standard deviation	2.904481053
Coefficient of variation (CV)	0.4493910579
Kurtosis	0.7567944649
Mean	6.463148302
Median Absolute Deviation (MAD)	1.89867883
Skewness	0.4709013446
Sum	41881194.53
Variance	8.436010185
Monotonicity	Not monotonic

Figure 3 Summary statistics of wind speed

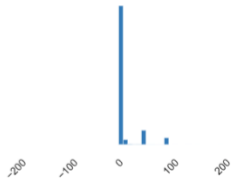
Pitch angle: the pitch angle variable measures the angle of the blades in a wind turbine. As measured in degree, the angle value range is [-360, 360], most of which are zero (more than 60% of the total records). The pitch angle is highly correlated with rotor speed.

pitch angle
Real number (ℝ)

HIGH CORRELATION
HIGH CORRELATION
ZEROS

Distinct	1312972
Distinct (%)	20.3%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	8.31743891

Minimum	-240.4859496
Maximum	200.7599945
Zeros	3987502
Zeros (%)	61.5%
Negative	75
Negative (%)	< 0.1%
Memory size	49.4 MiB



Quantile statistics

Minimum	-240.4859496
5-th percentile	0
Q1	0
median	0
Q3	1.284832855
95-th percentile	45
Maximum	200.7599945
Range	441.2459441
Interquartile range (IQR)	1.284832855

Descriptive statistics

Standard deviation	21.19879134
Coefficient of variation (CV)	2.548716206
Kurtosis	7.241088583
Mean	8.31743891
Median Absolute Deviation (MAD)	0
Skewness	2.7970827
Sum	53896995.82
Variance	449.3887545
Monotonicity	Not monotonic

Figure 4 Summary statistics of pitch angle

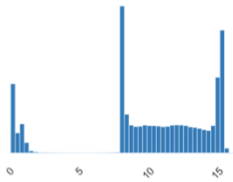
Rotor speed: the rotor speed (rpm) measures the speed rotation of the rotor. The speed value range is from 0 to >16, with more than 5% of values are zero. The majority of the data record the rotor speed from 7 to 15. Rotor speed is highly correlated with wind speed and pitch angle.

rotor speed
Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
ZEROS

Distinct	6045569
Distinct (%)	93.3%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	10.24244455

Minimum	0
Maximum	16.47107079
Zeros	375330
Zeros (%)	5.8%
Negative	0
Negative (%)	0.0%
Memory size	49.4 MiB



Quantile statistics

Minimum	0
5-th percentile	0
Q1	8.19563099
median	10.88593537
Q3	14.1068845
95-th percentile	15.24083
Maximum	16.47107079
Range	16.47107079
Interquartile range (IQR)	5.911253509

Descriptive statistics

Standard deviation	4.549574596
Coefficient of variation (CV)	0.444188355
Kurtosis	0.214913806
Mean	10.24244455
Median Absolute Deviation (MAD)	2.710336667
Skewness	-0.9967858987
Sum	66371030.45
Variance	20.69862901
Monotonicity	Not monotonic

Figure 5 Summary statistics of rotor speed

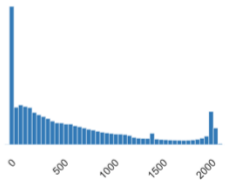
Active power: the active power (kW) measured the power produced by the wind turbine, which is highly correlated with the wind speed and the rotor speed. On the analyzed dataset, 5% of the records has zero active power, and 10% has negative values, the remaining data has the majority less than the mean value.

active power
Real number (ℝ)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
ZEROS

Distinct	4649275
Distinct (%)	71.7%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	623.3602873

Minimum	-77.807191
Maximum	2256.302246
Zeros	295524
Zeros (%)	4.6%
Negative	669451
Negative (%)	10.3%
Memory size	49.4 MiB



Quantile statistics

Minimum	-77.807191
5-th percentile	-0.2667500079
Q1	101.2951905
median	411.8117112
Q3	964.7881775
95-th percentile	2045.633271
Maximum	2256.302246
Range	2334.109437
Interquartile range (IQR)	863.492987

Descriptive statistics

Standard deviation	632.4719738
Coefficient of variation (CV)	1.014617047
Kurtosis	-0.1146629288
Mean	623.3602873
Median Absolute Deviation (MAD)	372.2986992
Skewness	1.00209992
Sum	4039374038
Variance	400020.7976
Monotonicity	Not monotonic

Figure 6 Summary statistics of active power

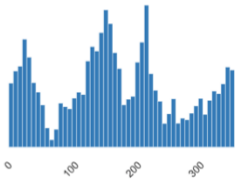
Cor. Nacelle direction: nacelle direction (degree) measures the angle made with the wind. As can be seen in the distribution, nacelle direction values are highly repeated, with only 4% distinct values over the entire dataset. Nacelle direction is highly correlated with sin_nacelle_dir (the sine of nacelle direction) and cos_nacelle_dir (the cosine of nacelle direction).

cor. nacelle direction
Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	258598
Distinct (%)	4.0%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	173.0060184

Minimum	0.09727041485
Maximum	359.9510498
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	49.4 MiB



Quantile statistics

Minimum	0.09727041485
5-th percentile	16.41390038
Q1	103.1210861
median	164.5839844
Q3	237.0010654
95-th percentile	345.6821594
Maximum	359.9510498
Range	359.8537794
Interquartile range (IQR)	133.8799793

Descriptive statistics

Standard deviation	100.3581648
Coefficient of variation (CV)	0.5800848188
Kurtosis	-0.8773287131
Mean	173.0060184
Median Absolute Deviation (MAD)	66.95069122
Skewness	0.1037279052
Sum	1121078826
Variance	10071.76125
Monotonicity	Not monotonic

Figure 7 Summary statistics of nacelle direction

Cor. Wind direction: corrected wind direction (degree) measures the angle made by the direction the wind flows, with the values range from 0 to 360. It can be used together with nacelle direction to derive the relative angle the nacelle made with the wind. As can be seen from the distribution, the wind direction values are divided relatively equally into 4 different ranges: 0-90, 90-180, 180-270, 270-360, with 99% distinct values over the dataset. Wind direction is highly correlated with sin_wind_dir (the sine of wind direction), and sin_nacelle_dir (the sine of nacelle direction).

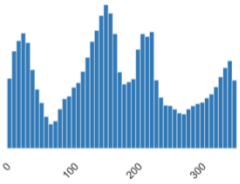
cor. wind direction

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	6435742
Distinct (%)	99.3%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	171.4053209

Minimum	0.0002565387365
Maximum	359.9985962
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	49.4 MiB



Quantile statistics

Minimum	0.0002565387365
5-th percentile	15.40282488
Q1	101.2206444
median	164.4093702
Q3	235.7895939
95-th percentile	343.4067665
Maximum	359.9985962
Range	359.9983397
Interquartile range (IQR)	134.5689495

Descriptive statistics

Standard deviation	100.212142
Coefficient of variation (CV)	0.5846501233
Kurtosis	-0.900302955
Mean	171.4053209
Median Absolute Deviation (MAD)	67.78885246
Skewness	0.1069206765
Sum	1110706308
Variance	10042.4734
Monotonicity	Not monotonic

Figure 8 Summary statistics of wind direction

Figure 9 summarizes the Spearman's rank correlation coefficient between the variables in the analyzed dataset, with -1 indicating total negative monotonic correlation, 0 indicating no monotonic correlation, and 1 indicating total positive monotonic correlation.

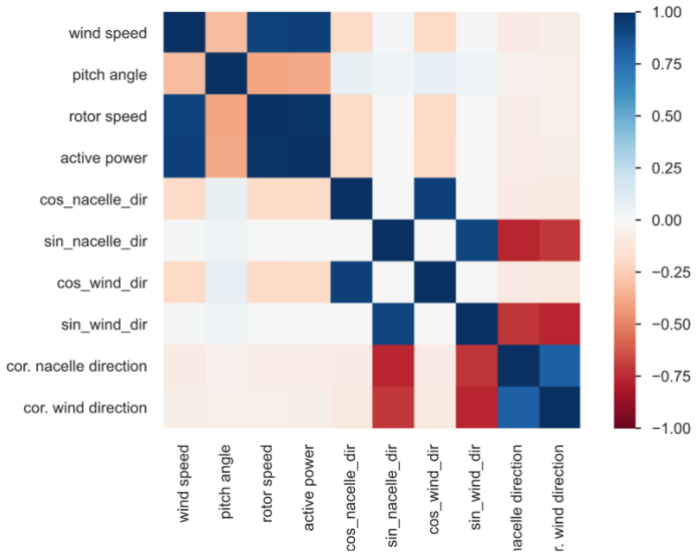


Figure 9 The Spearman's rank correlation coefficient between the variables

6.2 Meta data analysis

Based on the general understanding about the data obtained from the data exploration process, in this analysis step, we look at the metadata of provided datasets to extract information that might be used to improve the compression procedure of ModelarDB. Specifically, we look at the variable's names and analyze their corresponding data to find the potential relations between variables.

We found that some variables can be derived from other variables using mathematical functions. In particular, in provided datasets, following variables can be derived:

- cos_wind_dir can be derived from cor. wind direction by applying the cosine function: cos_wind_dir = cos(cor. wind direction*π/180)
- sin_wind_dir can be derived from cor. wind direction by applying the sine function: sin_wind_dir = sin(cor. wind direction*π/180)
- cos_nacelle_dir can be derived from cor. nacelle direction by applying the cosine function: cos_nacelle_dir = cos(cor. nacelle direction*π/180)
- sin_nacelle_dir can be derived from cor. nacelle direction by applying the sine function: sin_nacelle_dir = sin(cor. nacelle direction*π/180)

Using this derivation, we further improve ModelarDB by storing derived time series as a function of original time series. We detail this improvement in Section 7.

6.3 Data visualization

Next, we visualize each individual variables under different time intervals to inspect patterns and trends. The extracted insights will be incorporated to improve ModelarDB compression.

Wind speed: first, we plot and zoom in the wind speed values using different time intervals: month (Figure 10), hour (Figure 11), and 5-minute interval (Figure 12). We can see that wind speed values follow a more random pattern than repeated one. This random pattern is closely related to the random nature of the wind.

However, by zooming in the wind speed into minute intervals, we observe that some segments of the time series can be well represented by a quadratic function, for example, the two segments highlighted by red-curves in Figure 12. From this observation, we add a new data model representing the polynomial function with degree 2 to ModelarDB.

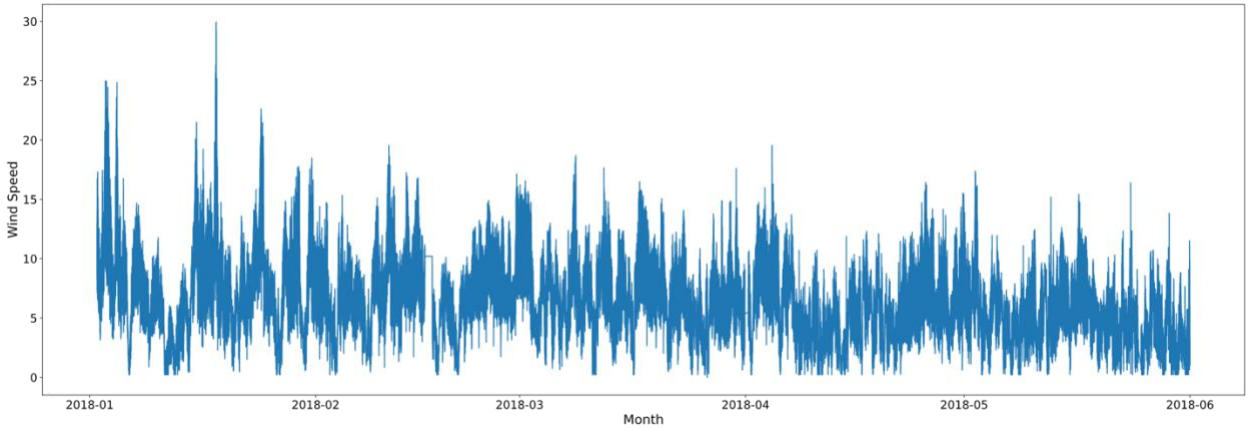


Figure 10 Wind speed in month interval from Jan-2018 to June-2018 (Turbine BEBEZE02)

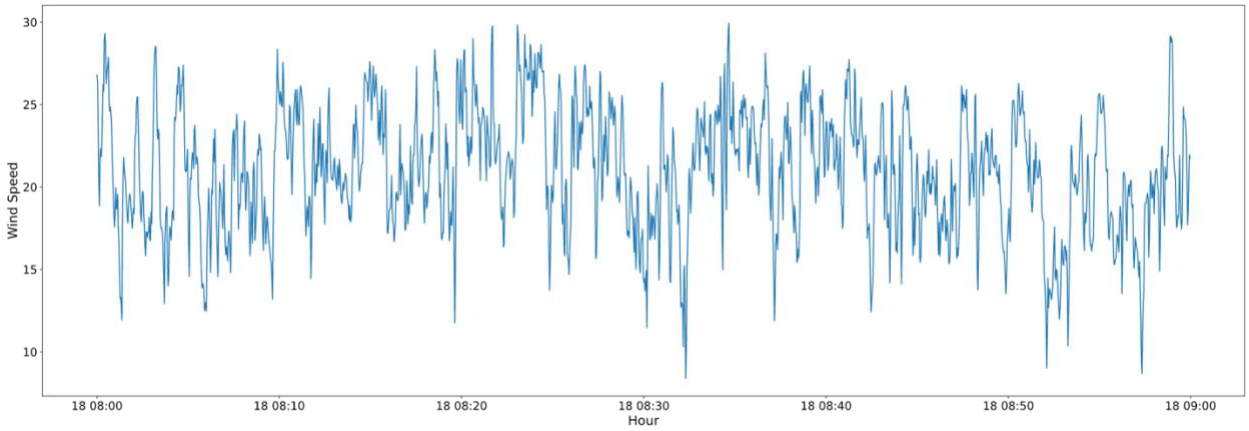


Figure 11 Wind speed in hour interval (Turbine BEBEZE02)

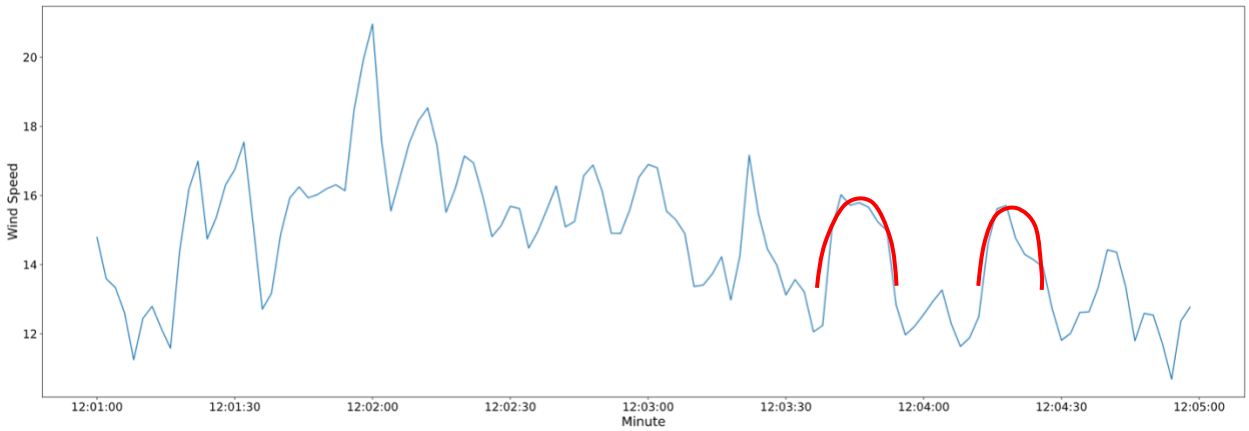


Figure 12 Wind speed in 5 minute interval (Turbine BEBEZE02)

Active power: next, we visualize the active power by plotting its data under different time intervals: day (Figure 13) and hour (Figure 14). It can be seen that no clear repeated patterns can be detected. However, similar to the wind speed, we observe that some segments of the time series can be well presented using a quadratic function, as zoomed in in Figure 14.

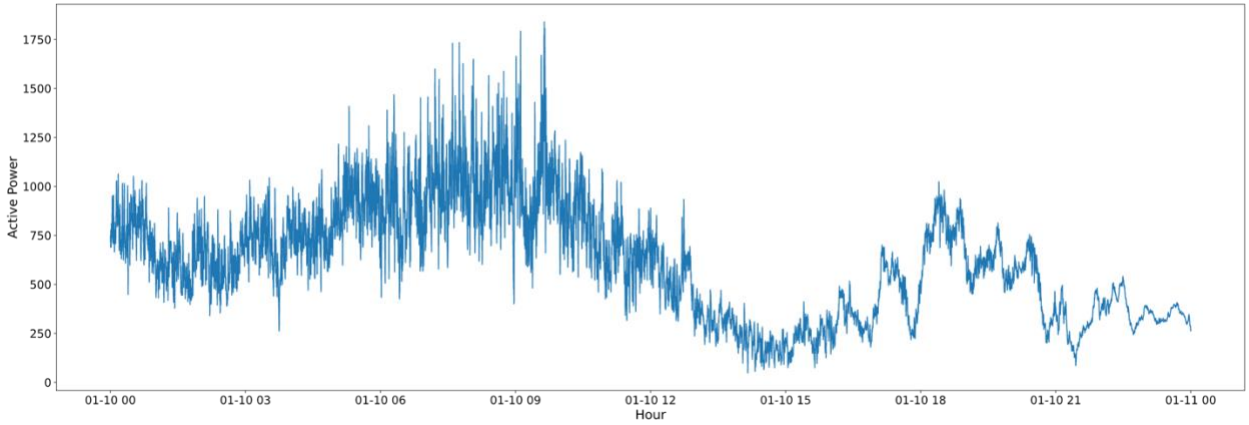


Figure 13 Active power in one day interval (Turbine BEBEZE02)

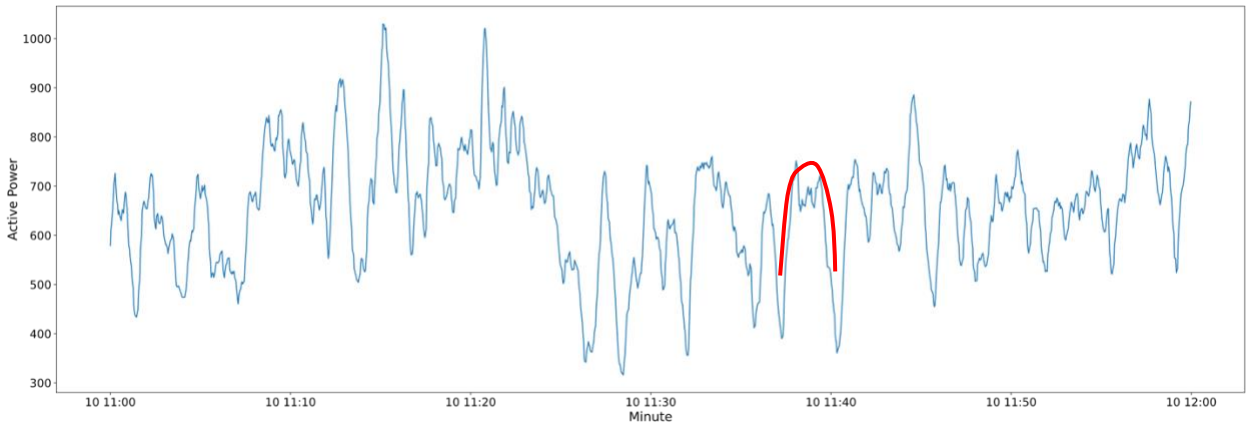


Figure 14 Active power in one hour interval (Turbine BEBEZE02)

Pitch angle: we visualize the pitch angle data under the intervals: week (Figure 15), and day (Figure 16). As can be seen, pitch angle time series have a more regular pattern compared to wind speed or active power. Pitch angle has many repeated values that can be well represented by constant functions (which is already supported in ModelarDB). For example, the time series in Figure 15 can be mainly represented by three constant functions: $f_1 = 0$, $f_2 = 47$, and $f_3 = 85$. We can also see that the constant functions used are also repeated, which enables the possibility to combine similar models in ModelarDB. We will explore this new feature for the next version of ModelarDB.

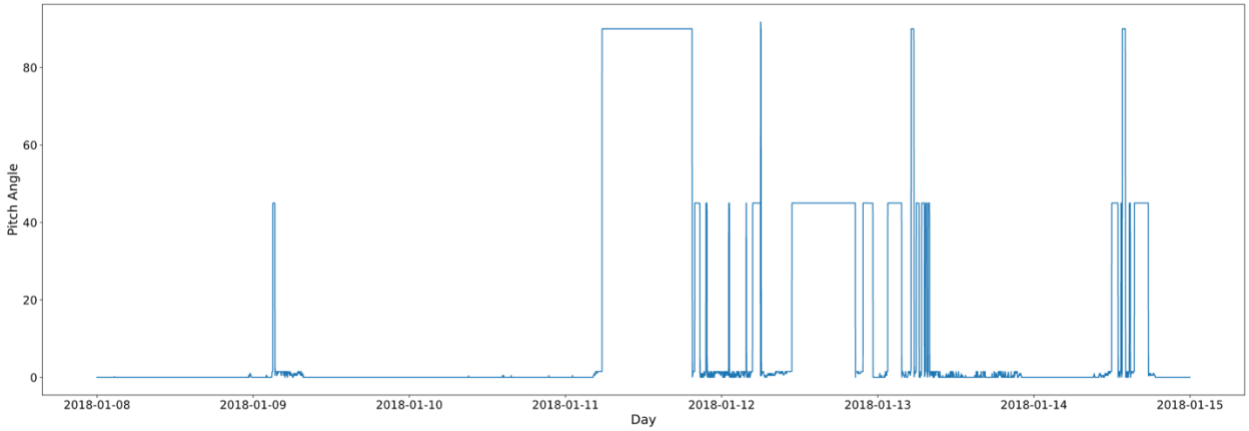


Figure 15 Pitch angle in week interval (Turbine BEBEZE02)

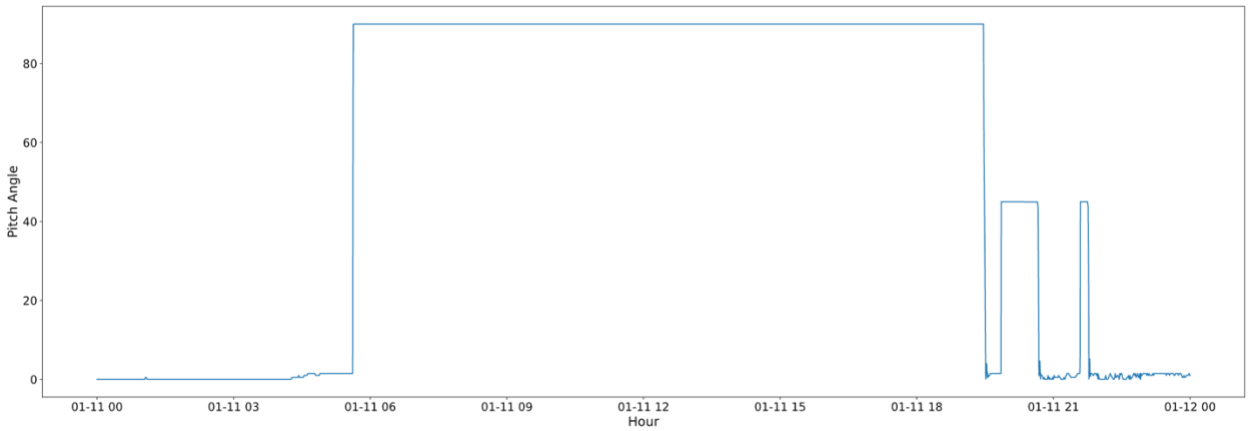


Figure 16 Pitch angle in day interval (Turbine BEBEZE02)

Rotor speed: we visualize rotor speed data under different time intervals: day (Figure 17), and hour (Figure 18). As can be seen, rotor speed data have random pattern as wind speed. Thus, no useful information is extracted from this variable.

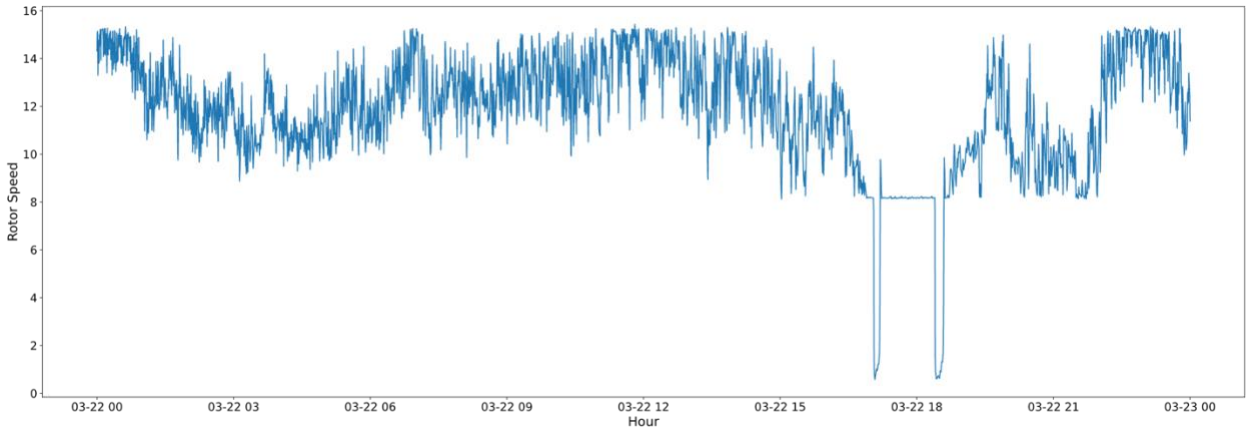


Figure 17 Rotor speed in day interval (Turbine BEBEZE02)

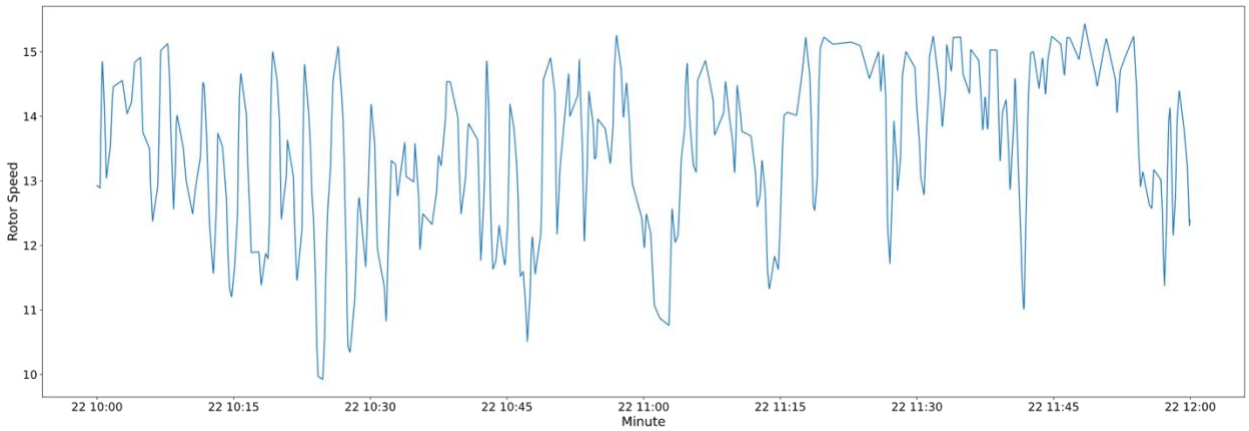


Figure 18 Rotor speed in hour interval (Turbine BEBEZE02)

Cor. nacelle direction: Next, we visualize corrected nacelle direction under the time intervals: week (Figure 19), day (Figure 20), and hour (Figure 21). Similar to pitch angle, nacelle direction data contain repeated constants, and thus, can be well represented by the constant function. For example, the time series in Figure 21 can be well represented by 10 different constant functions which correspond to 10 distinct constant values. The repeated constant functions used throughout the time series also enable the possibility to combine similar data models, as in the case of the pitch angle.

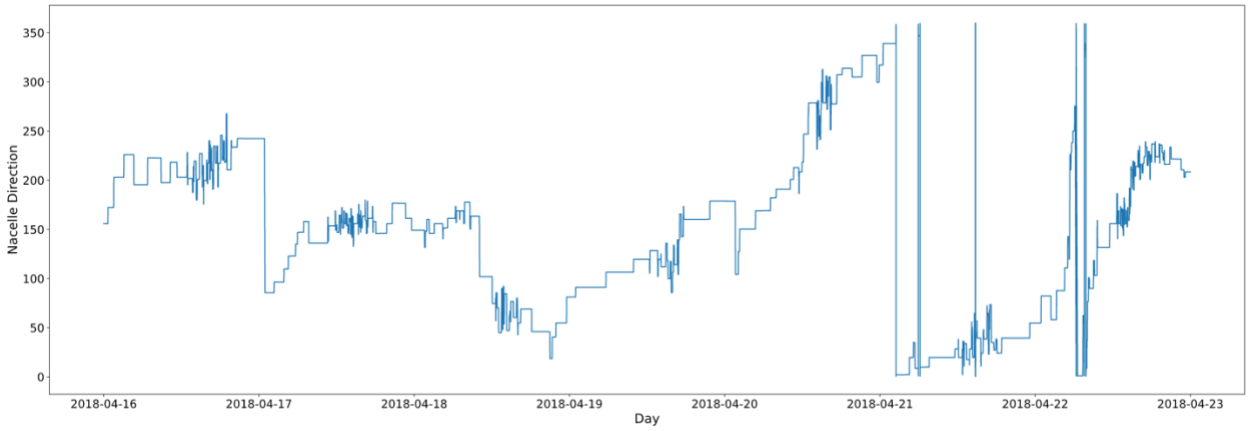


Figure 19 Nacelle direction in week interval (Turbine BEBEZE02)

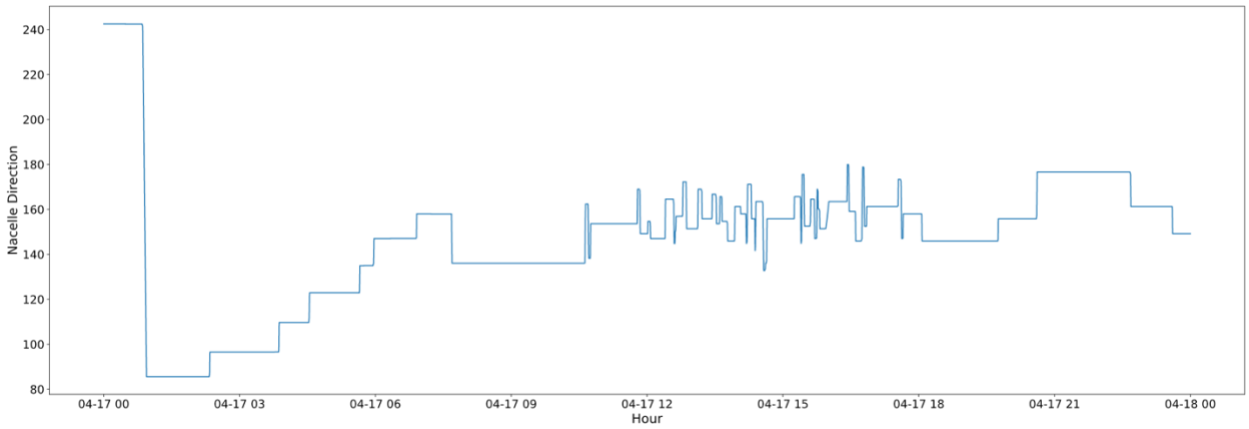


Figure 20 Nacelle direction in day interval (Turbine BEBEZE02)

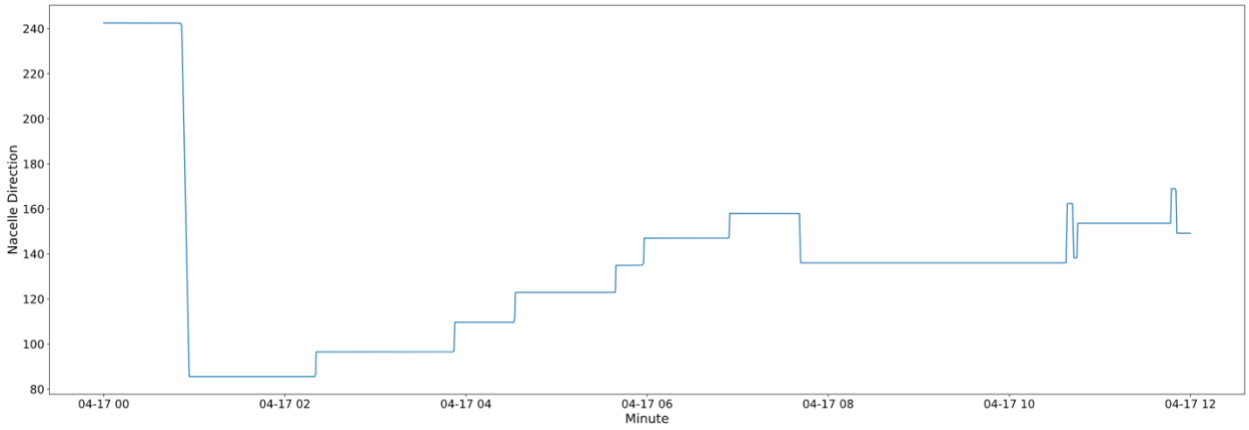


Figure 21 Nacelle direction in hour interval (Turbine BEBEZE02)

Cor. wind direction: next, we visualize the corrected wind direction under different time intervals: day (Figure 22), and hour (Figure 23). However, unlike nacelle direction, the wind direction data exhibit a random pattern, and thus, no repeated patterns can be extracted from this variable to further improve ModelarDB compression. Some segments in wind direction time series can be well represented by a quadratic function, as in Figure 23).

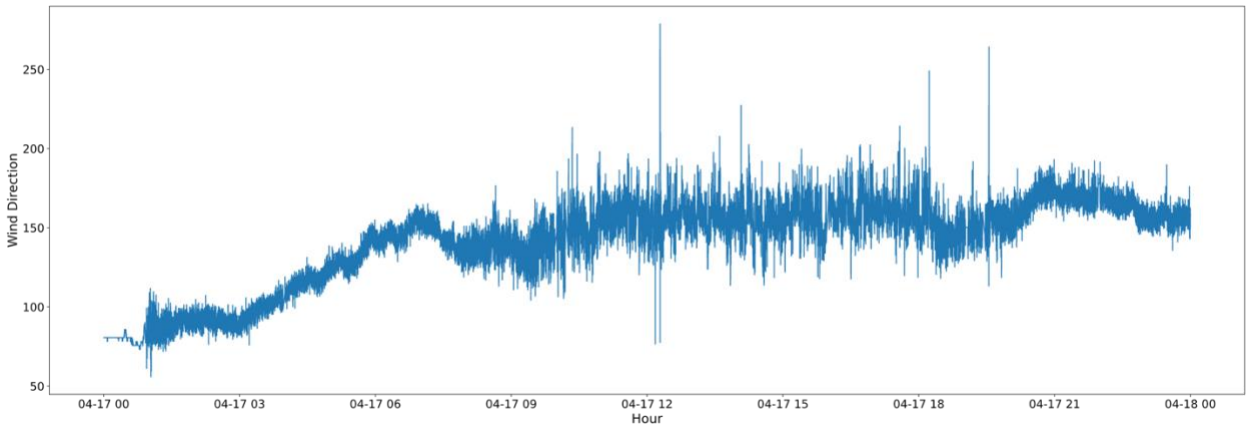


Figure 22 Wind direction in day interval (Turbine BEBEZE02)

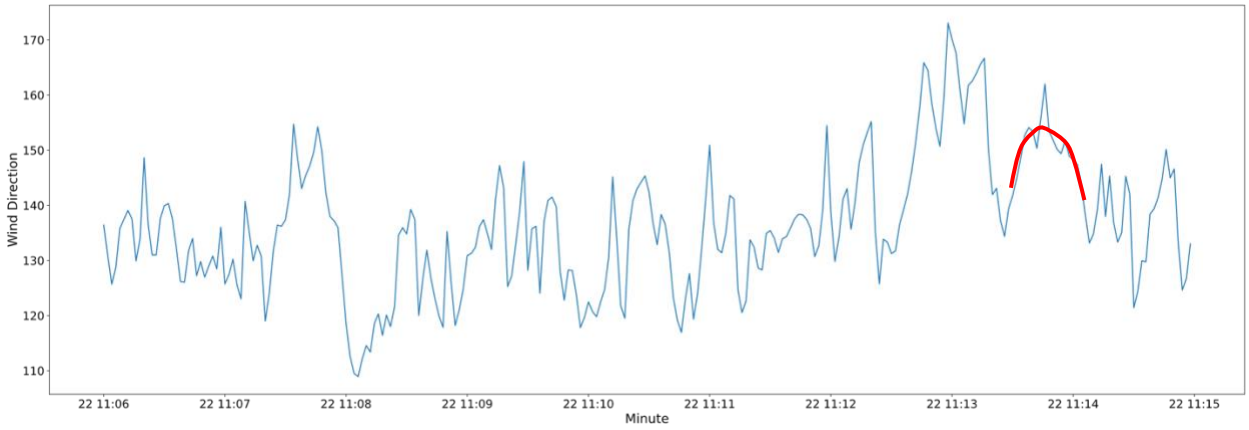


Figure 23 Wind direction in hour interval (Turbine BEBEZE02)

In conclusion, the data visualization step allows to analyze pattern and trend on the time series. Throughout this analysis, we find that the polynomial function of degree 2 can be used to represent some segments in ENGIE data. Furthermore, some variables can be well represented by repeated constant functions, such as the pitch angle or nacelle direction, which enable the combination of similar data models. This potential will be explored in the next version of ModelarDB.

6.4 New data model: polynomial function of degree 2

Upon identifying the new data model in section 6.3, we proceed to test its fitness on ENGIE data. Specifically, we fit the quadratic function on several segments of ENGIE time series, and compare its fitting error to the error obtained when using linear function. It is shown that in some time series segments, the quadratic function results in better representation, i.e., lower residual error, than the existing data models such as the linear function.

Figures 24 shows the fitting on a specific segment of the Wind Speed time series, using the quadratic function (left) and two linear functions (right). It can be seen that the quadratic function obtains a lower fitting error than the two linear functions, i.e., 16.9 vs. 20.11.

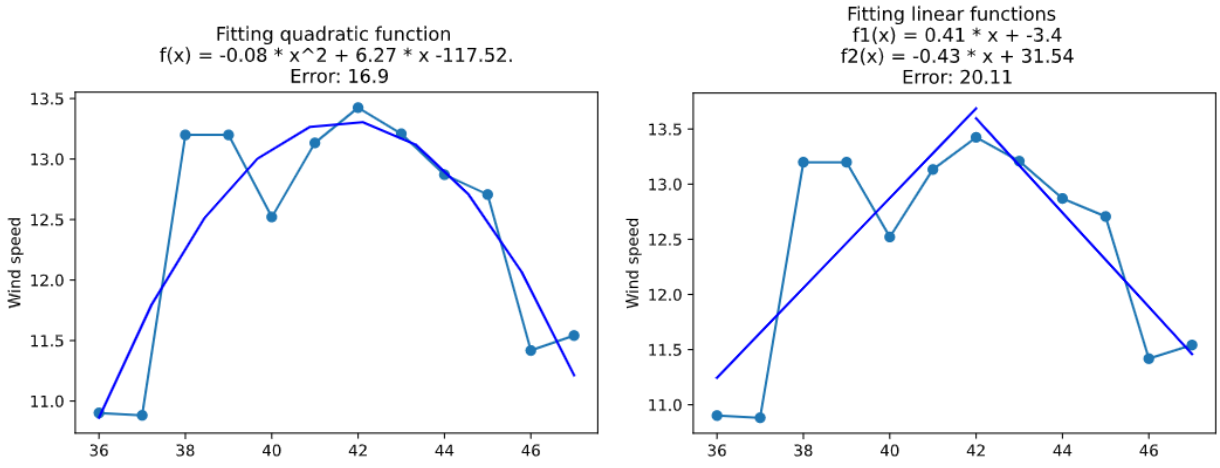


Figure 24 Fitting wind speed using quadratic function (left) and linear function (right)

Figures 25 shows the fitting on a specific segment of the Active Power time series, using the quadratic function (left) and two linear functions (right). It can be seen that the quadratic function obtains a lower fitting error than the two linear functions, i.e., 73.7 vs. 90.58.

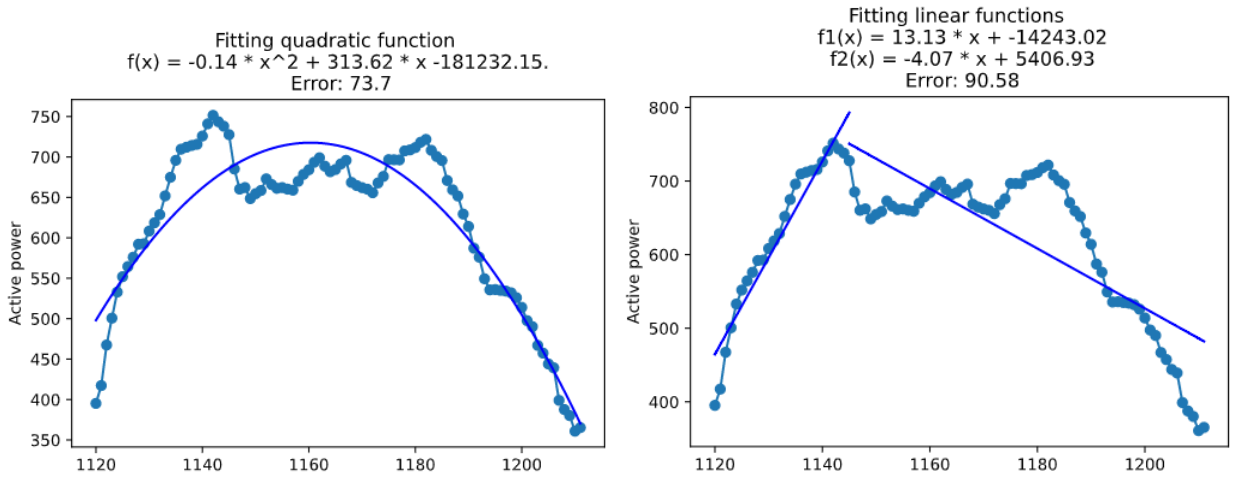


Figure 25 Fitting active power using quadratic function (left) and linear function (right)

In conclusion, we show that in ENGIE data, the polynomial function of degree 2 can represent better some time series segments than the existing data models such as the linear function. For this reason, we add the polynomial function of degree 2 as the new data model in ModelarDB. Implementation of this new data model on data streams will be added to the next version of ModelarDB.

6.5 Utilizing motifs to improve ModelarDB compression

In this analysis, we aim to search for motifs which are the repeated subsequences in a time series. As motifs represent the repeated subsequences in the time series, motifs can be grouped and represented by the same data model. Thus, if we can identify such motifs in the time series, it can potentially help improve the compression of ModelarDB.

We utilize the MatrixProfile (MP) [8] library which supports the motif search. Generally, to search for motifs in a time series, MP first computes the pairwise z-normalized Euclidean distances between a window and a time series. The window has a predefined length. Based on the computed Euclidean distances, MP selects only the nearest neighbors (based on the predefined max_matches parameter) which represent the k-top similar subsequences in the time series.

In this deliverable, we analyze whether motifs are present in ENGIE data, and how to search for them. This will serve as the basis to extend ModelarDB in the next version. In the following, we present the findings we found in ENGIE data with respect to motifs.

Wind speed: Figures 26, 27, 28 represent the computed matrix profile and the motifs found for wind speed, using window length = 60 (2 mins). As can be seen in Figure 28, motifs are present for the wind speed. However, the found motifs are not the best aligned subsequences. Furthermore, the found motifs are very complex and cannot be represented by a single data model. Thus, the possibility to group these motifs using the same data model while needing to satisfy the predefined error bound is low in ModelarDB.

For this reason, better aligned motifs that can be represented by a single data model are needed in order to be utilized in ModelarDB. However, finding motifs depends on the predefined window length. A method to choose a window length which can result in good motifs is still open for investigating in the next version of Task 2.1.

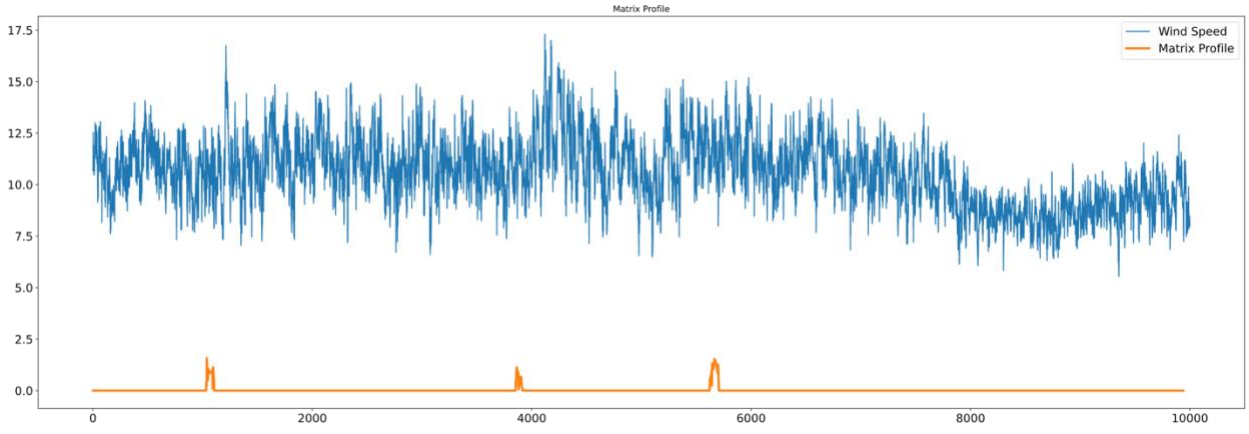


Figure 26 Matrix Profile of Wind Speed (Turbine BEBEZE02)

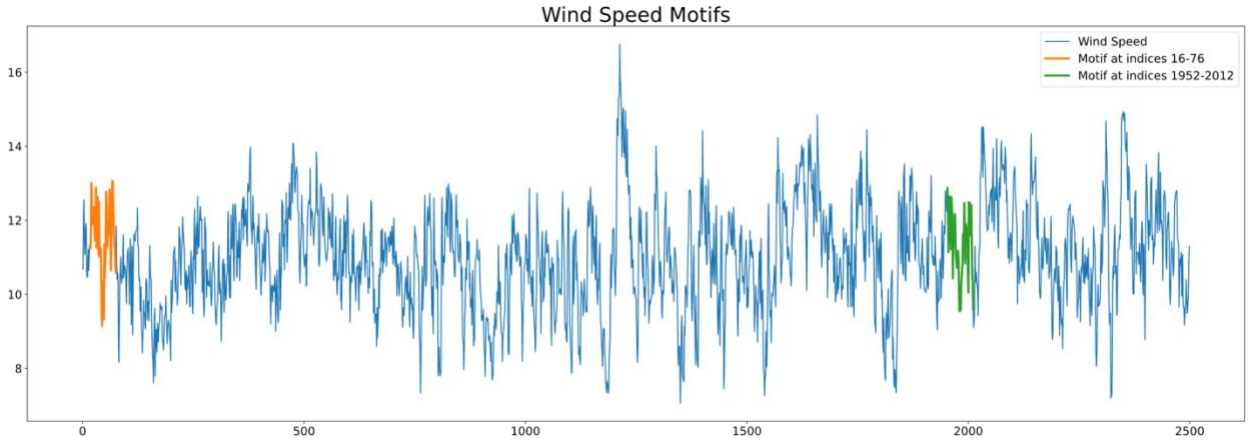


Figure 27 Example of motifs in Wind Speed time series using window length = 60 (Turbine BEBEZE02)

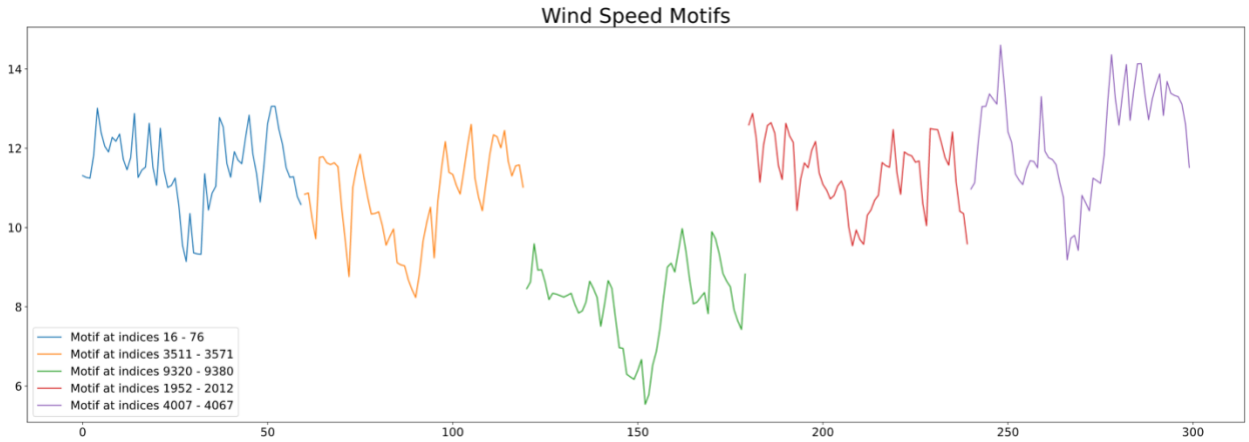


Figure 28 Example of motifs in Wind Speed time series using window length = 60 (Turbine BEBEZE02)

Active power: Figures 29, 30, 31 illustrate the computed matrix profile and the motifs found for active power, using window length = 300 (10 mins). Similar to the wind speed, Figure 31 also shows some motifs present for the active power but they cannot be represented by a single data model. The better motifs are needed in order to be utilized by ModelarDB, which depends on finding the right window length.

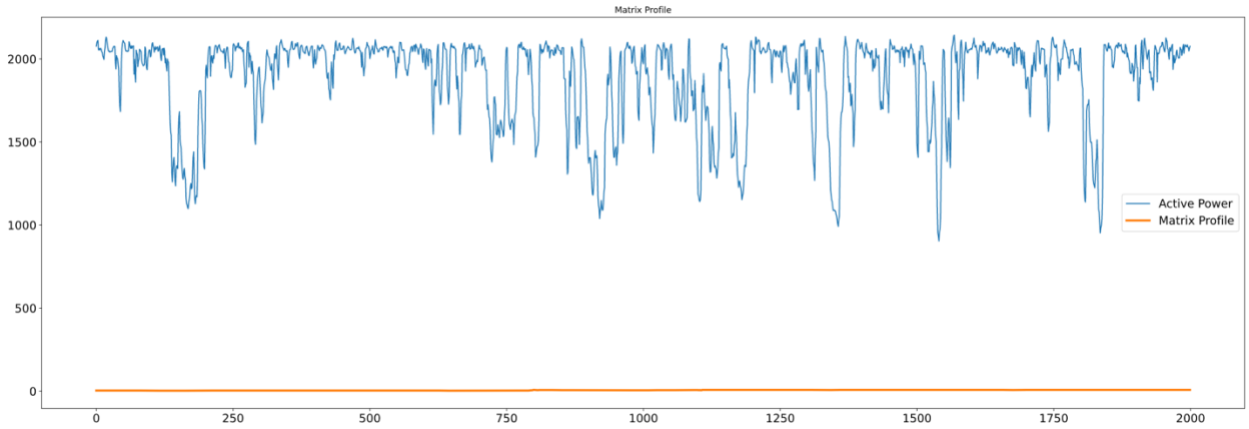


Figure 29 MatrixProfile of Active Power (Turbine BEBEZE02)

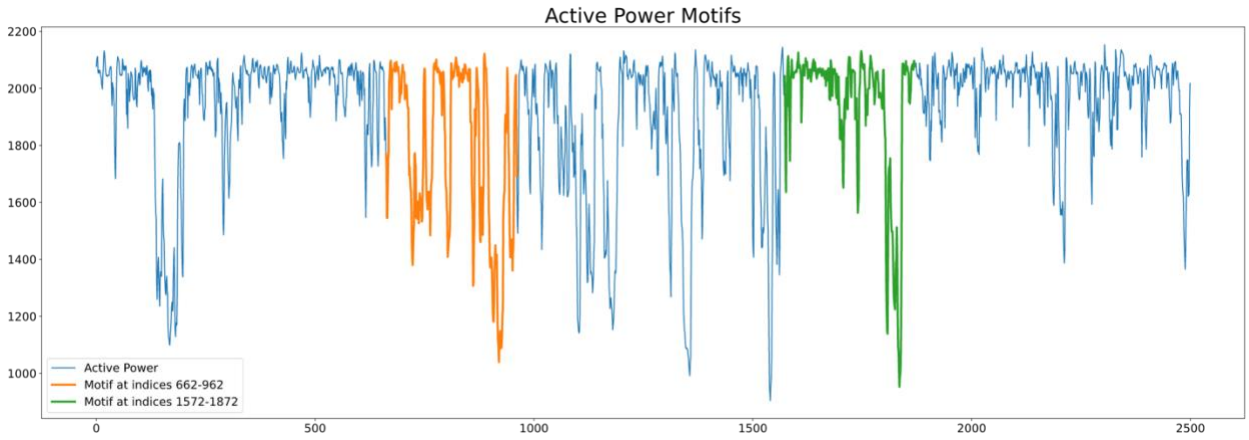


Figure 30 Example of motifs in Active Power time series using window length = 300 (Turbine BEBEZE02)

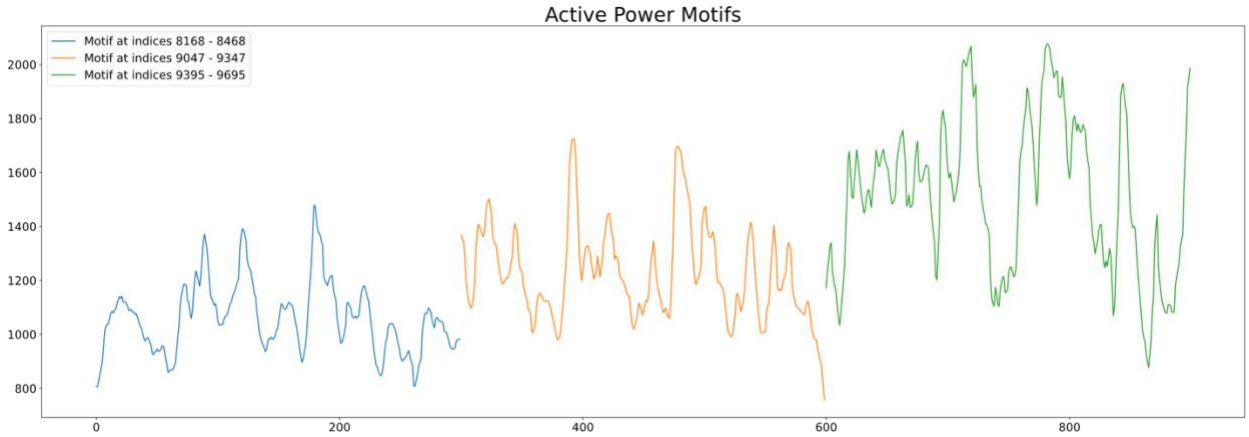


Figure 31 Example of motifs in Active Power time series using window length = 300 (Turbine BEBEZE02)

Pitch angle: Figure 32, 33, 34 illustrate the computed matrix profile and the motifs found for pitch angle, using window length = 250. Here, Figure 33, 34 also show some motifs present for the pitch angle. However, similar to the other two time series presented above, the found motifs cannot be represented by a single data model. The better motifs are needed in order to be utilized by ModelarDB, which depends on finding the right window length.

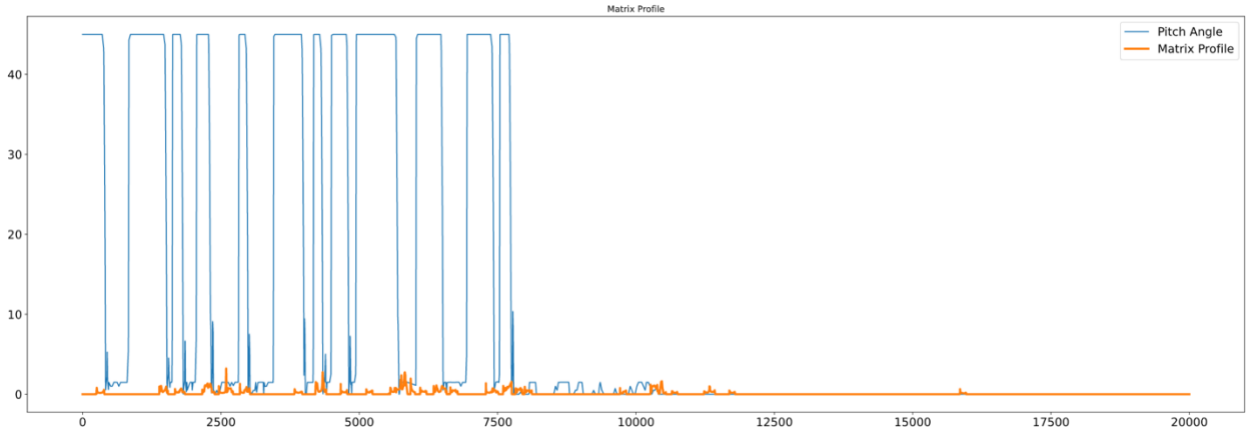


Figure 32 Matrix Profile of Pitch Angle (Turbine BEBEZE02)

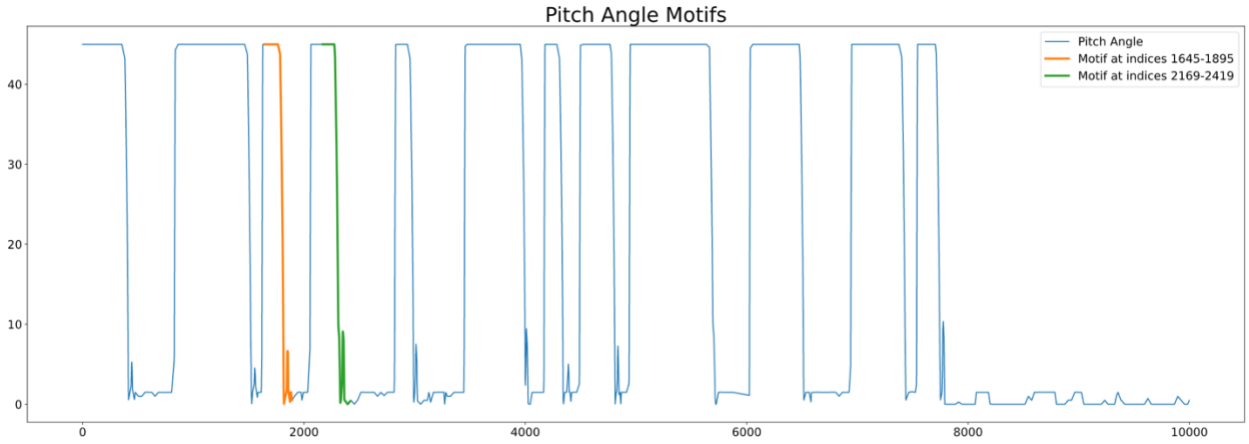


Figure 33 Example of motifs in Pitch Angle time series using window length = 250 (Turbine BEBEZE02)

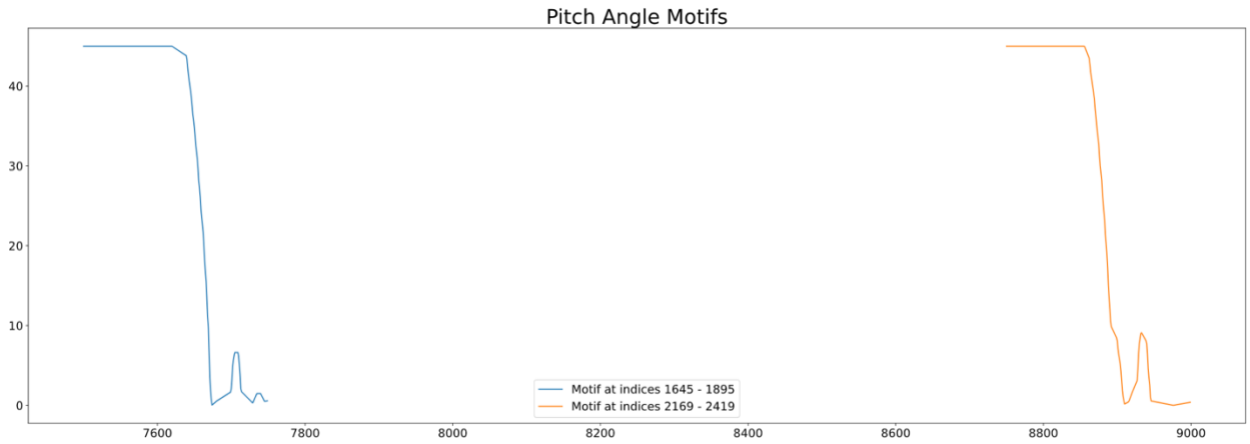


Figure 34 Example of motifs in Pitch Angle time series using window length = 250 (Turbine BEBEZE02)

In conclusion, as motifs depend solely on the length of the window, this parameter plays an important role in deciding how good the motifs can be detected. Furthermore, in order for motifs to be utilized by ModelarDB, they must be represented by a single data model. Thus, utilizing motifs to further improve the compression in ModelarDB is still open for investigation in the next version of Task 2.1. The next step for this analysis would be researching an automatic method to find a good window length for a given time series. A possible direction would be to rely on the current segment length of ModelarDB.

7 Derived time series compression

The time series in a data set are often correlated, e.g., co-located temperature sensors often produce similar values and exhibit positive linear correlation. ModelarDB can group such time series and compress them as one stream of models. This significantly reduces the amount of storage required compared to compressing each time series separately. To exploit this, it is, however, necessary that a single model can represent the values from all the time series in the group for every timepoint. In some cases when the values are very similar, e.g., a linear model can do this. In other cases, the values can be very different and a model type like Gorilla can then be used since a single Gorilla model can store values from each of the time series in the group. As a result, the reduction in storage is highly dependent on the time series actual values, and in typical cases, it rarely matches the number of time series that are compressed together (which is the optimal case). For example, compressing seven time series together does not necessarily provide a seven-times reduction in the amount of storage required. However, it can also be the case that the relationship between values is different time series is *static* in the sense that values of one time series can be computed exactly from values of another time series by a given function. In that case, only a single time series need to be physically stored as the values of the other time series can be *derived* from it (i.e., calculated using a function) during query execution.

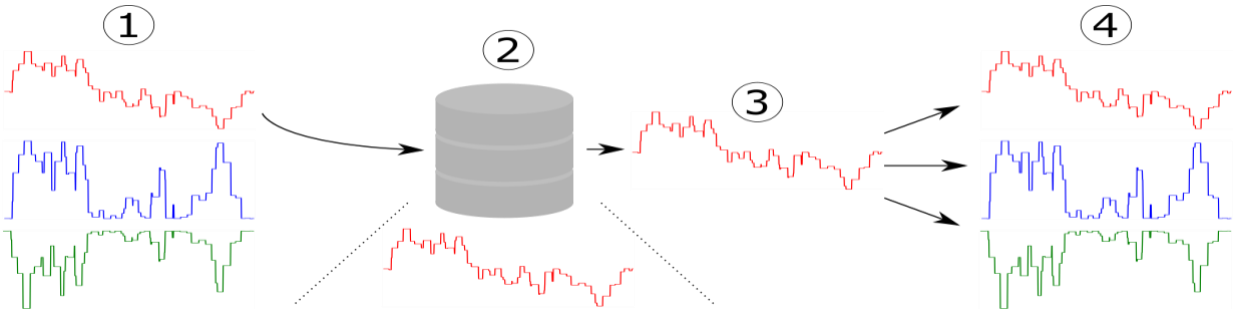


Figure 35 Example of efficiently compressing time series by not storing them and instead deriving them from other time series

ModelarDB has been extended so users can specify which time series can be derived from another time series, thus allowing the system to not store any data points for the derived time series. This is exemplified in Figure 35. At (1) a sensor measures the direction of an object which is turned over time. This produces the time series shown in red. Two other time series with the cosine and sine, respectively, of the angels in the first time series are also needed for later analysis. These time series are shown in blue and green, respectively. However, as these two can be computed directly from the original measurements, ModelarDB only stores the original time series (red) at (2) and not the two derived time series (blue and green). When ModelarDB receives a query, it reads the original time series (3) and applies the user-defined functions `cos(value*π/180)` and `sin(value*π/180)` where `value` is measured in degree, to create the cosine (blue) and sine (green) time series (4).

ModelarDB cannot detect that a time series can be derived from another as the time series are being ingested as streams. So, users must explicitly specify all derived time series in ModelarDB's configuration file as follows:

```
1. modelardb.source.derived     tid     derived_source     function(value,
   scaling_factor) -> 32-Bit Float
2. modelardb.source.derived    source    derived_source    function(value,
   scaling_factor) -> 32-Bit Float
```

The only difference between the two methods is that in (1) the derived time series is specified in terms of the `tid` of an existing time series, while in (2) the derived time series is specified in terms of the `source` (filename or socket) of an existing time series. The second parameter `derived_source`, is the user-defined source used to associate dimension members to the derived time series. This allows users to have separate dimension members associated with the original and the derived time series. The last parameter, `function`, is the user-defined Scala code to be executed for each value in the original time series. The function receives the value of each data point and the user-defined scaling factor (by default 1.0) as input and must return the value for the derived time series. One scaling factor can be set per time series and is used to scale all values in the time series during ingestion and query processing. The two arguments are provided as 32-bit floating-point numbers, and the new value must also be returned as a 32-bit floating-point number. The specification of derived time series is currently only stored in the configuration file, and not in the database, to make it easy for users to add, modify, and remove derived time series. Also, a derived time series is a direct one-to-one mapping of the values in the original time series to the values in the derived time series. Thus, the deriving function cannot be used to aggregate values (e.g., to change

the sampling interval) or generate additional values (e.g., for forecasting). However, these restrictions are only limitations of the current implementation and not the method.

8 ModelarDB testing on ENGIE data

We run ModelarDB on provided ENGIE data to test the compression rate and analyze the models used. We report the results below.

8.1 Analyzing the compression rate on ENGIE data

First, we report the compression results on ENGIE data under different error bounds in Table 1. In this table, the original size on disk of ENGIE data in ORC format with 32-bit floating point is 6.3 GB. After being compressed by ModelarDB with 0%-error bound, the size is reduced to 5.1 GB which corresponds to 19.05% of reduction. Similarly, the size is reduced by 31.75% for 1%-error bound, 46.03% for 5%-error bound, and 57.14% for 10%-error bound.

	Size on disk (GB)	**Percentage of reduction (%)**
ENGIE data (ALL)	6.3	-
Error bound = 0%	5.1	19.05
Error bound = 1%	4.3	31.75
Error bound = 5%	3.4	46.03
Error bound = 10%	2.7	57.14

Table 1 Compression results on entire ENGIE data

Next, we test the compression rate using the derived time series feature. In the provided ENGIE datasets, 72 out of 180 time series are derivable. The compression results are shown in Table 2. Here, with 0%-error bound, the size is reduced to 3.3 GB which corresponds to 47.62% of reduction. Similarly, the size and the percentage of reduction for 1%-, 5%- and 10%-error bound respectively are: 2.7 GB and 57.14%, 2 GB and 68.25%, 1.5 GB and 76.19%.

	Size on disk (GB)	**Percentage of reduction (%)**
ENGIE data (ALL)	6.3	-
Error bound = 0%	3.3	47.62
Error bound = 1%	2.7	57.14

Error bound = 5%	2	68.25
Error bound = 10%	1.5	76.19

Table 2 Compression results on ENGIE data without the derived time series

8.2 Analyzing the models used on ENGIE data

Next, we analyze the models used on ENGIE data, shown in Table 3 under different error bounds. It is intuitive to see that with 0%-error bound, the Gorilla data model [7] for lossless compression is used the most (i.e., 34065405 times which corresponds to 94.85% of the total models used), followed by PMC-Mean [5] for constant models, and then Swing [6] for linear models. As error bound increases, the use of Gorilla data model decreases, while the use of PMC_Mean and Swing increases.

ENGIE ALL	**Models used**	**Count**	**Percentage (%)**
Error bound = 0%	PMC_Mean	1163060	3.24
	SwingFilter	686982	1.91
	FacebookGorilla	34065405	94.85
Error bound = 1%	PMC_Mean	2997751	9.06
	SwingFilter	4411786	13.33
	FacebookGorilla	25678809	77.61
Error bound = 5%	PMC_Mean	5528634	17.45
	SwingFilter	7380953	23.30
	FacebookGorilla	18771980	59.25
Error bound = 10%	PMC_Mean	6403500	22.40
	SwingFilter	8674973	30.35
	FacebookGorilla	13506738	47.25

Table 3 Models used on ENGIE data

Table 4 reports the models used on ENGIE data without the derived time series under different error bounds. We can also observe the same trend of models used with respect to the error bound. With lossless compression, the Gorilla data model is the most used. With lossy compression, PMC_Mean and Swing models dominate.

ENGIE using derived time series function	**Models used**	**Count**	**Percentage (%)**
Error bound = 0%	PMC_Mean	643293	2.72
	SwingFilter	438921	1.86

	FacebookGorilla	22559344	95.42
Error bound = 1%	PMC_Mean	2081413	9.44
	SwingFilter	3867723	17.55
	FacebookGorilla	16091676	73.01
Error bound = 5%	PMC_Mean	3874409	18.59
	SwingFilter	5708522	27.39
	FacebookGorilla	11259573	54.02
Error bound = 10%	PMC_Mean	4568643	24.43
	SwingFilter	6510217	34.81
	FacebookGorilla	7621662	40.76

Table 4 Models used on ENGIE data without the derived time series

8.3 Analyzing the models used on each time series

Next, we report the models used for each time series in ENGIE data. Here, we discuss the results on the wind turbine BEBEZE01 data only.

Table 5 reports the models used in each time series on the data of the turbine BEBEZE01. With lossless compression (0%-error bound), we can see that the most used data model is Gorilla, while the constant model PMC_Mean and the linear model Swing are scarcely used. However, as the error bound increases, the PMC_Mean and Swing models are increasingly used and the Gorilla model is decreasingly used.

Furthermore, each individual time series also exhibits the difference in terms of models used. In the time series whose data are more random, such as active power or wind speed, Gorilla data model is used many more times than the PMC_Mean and Swing models. In the time series where data follow a more regular pattern such as pitch angle or nacelle direction, PMC_Mean and Swing models are utilized more.

	Time Series	**Model Type**	**Count**	**Percentage (%)**
Error bound = 0%	active_power	PMC_Mean	14	> 0.00
	active_power	SwingFilter	5022	1.27
	active_power	FacebookGorilla	390687	98.73
	cos_nacelle_dir	PMC_Mean	13755	29.86
	cos_nacelle_dir	SwingFilter	3085	6.70
	cos_nacelle_dir	FacebookGorilla	29227	63.44
	cos_wind_dir	PMC_Mean	35	0.01
	cos_wind_dir	SwingFilter	3122	0.80
	cos_wind_dir	FacebookGorilla	388866	99.19

	nacelle_direction	PMC_Mean	13699	29.75
	nacelle_direction	SwingFilter	1162	2.52
	nacelle_direction	FacebookGorilla	31182	67.72
	pitch_angle	PMC_Mean	31002	20.08
	pitch_angle	SwingFilter	4969	3.22
	pitch_angle	FacebookGorilla	118388	76.70
	rotor_speed	PMC_Mean	2503	0.65
	rotor_speed	SwingFilter	16610	4.34
	rotor_speed	FacebookGorilla	363207	95.00
	sin_nacelle_dir	PMC_Mean	13786	28.23
	sin_nacelle_dir	SwingFilter	5459	11.18
	sin_nacelle_dir	FacebookGorilla	29585	60.59
	sin_wind_dir	PMC_Mean	38	0.01
	sin_wind_dir	SwingFilter	1158	0.30
	sin_wind_dir	FacebookGorilla	389882	99.69
	wind_direction	PMC_Mean	35	0.01
	wind_direction	SwingFilter	754	0.19
	wind_direction	FacebookGorilla	389855	99.80
	wind_speed	PMC_Mean	12	> 0.00
	wind_speed	SwingFilter	5167	1.31
	wind_speed	FacebookGorilla	390160	98.69
Error bound = 1%	active_power	PMC_Mean	19855	4.81
	active_power	SwingFilter	37663	9.12
	active_power	FacebookGorilla	355347	86.07
	cos_nacelle_dir	PMC_Mean	14179	43.38
	cos_nacelle_dir	SwingFilter	5115	15.65
	cos_nacelle_dir	FacebookGorilla	13395	40.98
	cos_wind_dir	PMC_Mean	14551	3.78
	cos_wind_dir	SwingFilter	9187	2.39
	cos_wind_dir	FacebookGorilla	360810	93.83
	nacelle_direction	PMC_Mean	14261	46.50
	nacelle_direction	SwingFilter	6187	20.17
	nacelle_direction	FacebookGorilla	10222	33.33
	pitch_angle	PMC_Mean	38440	22.58
	pitch_angle	SwingFilter	79248	46.55
	pitch_angle	FacebookGorilla	52571	30.88
	rotor_speed	PMC_Mean	73223	21.60
	rotor_speed	SwingFilter	202138	59.63

	rotor_speed	FacebookGorilla	63614	18.77
	sin_nacelle_dir	PMC_Mean	14525	42.81
	sin_nacelle_dir	SwingFilter	5269	15.53
	sin_nacelle_dir	FacebookGorilla	14132	41.66
	sin_wind_dir	PMC_Mean	6529	1.73
	sin_wind_dir	SwingFilter	5084	1.35
	sin_wind_dir	FacebookGorilla	366092	96.93
	wind_direction	PMC_Mean	5591	1.48
	wind_direction	SwingFilter	4878	1.29
	wind_direction	FacebookGorilla	366932	97.23
	wind_speed	PMC_Mean	1348	0.35
	wind_speed	SwingFilter	3278	0.86
	wind_speed	FacebookGorilla	375110	98.78
Error bound = 5%	active_power	PMC_Mean	92239	17.99
	active_power	SwingFilter	191009	37.26
	active_power	FacebookGorilla	229351	44.74
	cos_nacelle_dir	PMC_Mean	10442	43.82
	cos_nacelle_dir	SwingFilter	7169	30.09
	cos_nacelle_dir	FacebookGorilla	6217	26.09
	cos_wind_dir	PMC_Mean	44763	11.86
	cos_wind_dir	SwingFilter	44413	11.77
	cos_wind_dir	FacebookGorilla	288115	76.36
	nacelle_direction	PMC_Mean	6337	40.51
	nacelle_direction	SwingFilter	6733	43.04
	nacelle_direction	FacebookGorilla	2572	16.44
	pitch_angle	PMC_Mean	40384	23.91
	pitch_angle	SwingFilter	79764	47.22
	pitch_angle	FacebookGorilla	48763	28.87
	rotor_speed	PMC_Mean	44347	28.91
	rotor_speed	SwingFilter	105877	69.01
	rotor_speed	FacebookGorilla	3192	2.08
	sin_nacelle_dir	PMC_Mean	12055	44.35
	sin_nacelle_dir	SwingFilter	7326	26.95
	sin_nacelle_dir	FacebookGorilla	7798	28.69
	sin_wind_dir	PMC_Mean	29566	7.60
	sin_wind_dir	SwingFilter	26873	6.91
	sin_wind_dir	FacebookGorilla	332600	85.49
	wind_direction	PMC_Mean	126172	26.27

	wind_direction	SwingFilter	120598	25.11
	wind_direction	FacebookGorilla	233516	48.62
	wind_speed	PMC_Mean	38905	9.10
	wind_speed	SwingFilter	33985	7.95
	wind_speed	FacebookGorilla	354537	82.95
Error bound = 10%	active_power	PMC_Mean	118378	23.73
	active_power	SwingFilter	242319	48.57
	active_power	FacebookGorilla	138258	27.71
	cos_nacelle_dir	PMC_Mean	7761	42.47
	cos_nacelle_dir	SwingFilter	6887	37.69
	cos_nacelle_dir	FacebookGorilla	3624	19.83
	cos_wind_dir	PMC_Mean	53219	15.58
	cos_wind_dir	SwingFilter	61159	17.90
	cos_wind_dir	FacebookGorilla	227293	66.52
	nacelle_direction	PMC_Mean	3924	41.68
	nacelle_direction	SwingFilter	3771	40.06
	nacelle_direction	FacebookGorilla	1719	18.26
	pitch_angle	PMC_Mean	41854	24.30
	pitch_angle	SwingFilter	86098	49.99
	pitch_angle	FacebookGorilla	44267	25.70
	rotor_speed	PMC_Mean	21875	30.13
	rotor_speed	SwingFilter	48592	66.93
	rotor_speed	FacebookGorilla	2136	2.94
	sin_nacelle_dir	PMC_Mean	9508	43.36
	sin_nacelle_dir	SwingFilter	7194	32.80
	sin_nacelle_dir	FacebookGorilla	5228	23.84
	sin_wind_dir	PMC_Mean	48548	12.40
	sin_wind_dir	SwingFilter	53686	13.71
	sin_wind_dir	FacebookGorilla	289341	73.89
	wind_direction	PMC_Mean	74954	28.40
	wind_direction	SwingFilter	66577	25.23
	wind_direction	FacebookGorilla	122353	46.37
	wind_speed	PMC_Mean	161029	27.82
	wind_speed	SwingFilter	190294	32.88

Table 5 Models used in each time series on wind turbine BEBEZE01 data

9 Conclusion

This deliverable has presented how models can be used to compress large amounts of time series data efficiently. The available ENGIE data was then investigated in details. Based on this, a new quadratic model type was proposed to be added to ModelarDB. Further, it was found that many of the time series can be derived from other timeseries. Based on this, support for derived time series has been added to ModelarDB. The ENGIE dataset was also compressed by use of models in ModelarDB and it was found that even with lossless compression, the required storage was reduced by 19% and with an error bound of 10%, the reduction was bigger than 76%, compared to the original data size.

References

[1] S. K. Jensen, T. B. Pedersen, and C. Thomsen: *ModelarDB: Modular Model-Based Time Series Management with Spark and Cassandra*. PVLDB 11(11), 2018

[2] S. K. Jensen, T. B. Pedersen, and C. Thomsen: *Demonstration of ModelarDB: Model-Based Management of Dimensional Time Series*. Proc. of SIGMOD, 2019

[3] S. K. Jensen: *Model-Based Time Series Management at Scale.* PhD Thesis, The Technical Faculty of IT and Design, Aalborg University, 2019

[4] S. K. Jensen, T. B. Pedersen, C. Thomsen: *Scalable Model-Based Management of Correlated Dimensional Time Series in ModelarDB+*. Proc. of ICDE, 2021.

[5] I. Lazaridis and S. Mehrotra: *Capturing Sensor-Generated Time Series with Quality Guarantees*. Proc. of ICDE, 2003

[6] H. Elmeleegy, A. K. Elmagarmid, E. Cecchet, W. G. Aref, and W. Zwaenepoel: *Online Piece-wise linear Approximation of Numerical Streams with Precision Guarantees*. PVLDB, 2(1), 2009.

[7] T. Pelkonen et al.: *Gorilla: A Fast, Scalable, In-Memory Time Series Database*. PVLDB, 8(12), 2015.

[8] H. A. Dau, E. J. Keogh: *Matrix Profile V: A Generic Technique to Incorporate Domain Knowledge into Motif Discovery*. KDD 2017: 125-134.