COMPARING MACHINE LEARNING METHODS ON CONCEPT DRIFT DETECTION FOR PREDICTIVE MAINTENANCE

Jan Zenisek[(a)], Josef Wolfartsberger[(b)], Christoph Sievi[(c)], Michael Affenzeller[(d)]

[(a),(b),(c),(d)] University of Applied Sciences Upper Austria, Hagenberg
[(a),(d)] Institute for Formal Models and Verification, Johannes Kepler University Linz, Austria

[(a)]jan.zenisek@fh-hagenberg.at, [(b)]josef.wolfartsberger@fh-steyr.at,
[(c)]christoph.sievi@fh-wels.at, [(d)]michael.affenzeller@fh-hagenberg.at

ABSTRACT

In this work we present a comparison of various machine learning algorithms with the objective of detecting concept drifts in data streams characteristical for condition monitoring of industrial production plants. Although there is a fair number of contributions employing machine learning algorithms in related fields such as traditional time series forecasting or concept drift learning, data sets with sensor streams from a production plant are rarely covered. This work aims at shedding some light on the matter of how efficient the depicted algorithms perform on concept drift detection to pave the way for Predictive Maintenance (PdM) and which intermediate data processing steps therefore might be beneficial.

Keywords: machine learning, predictive maintenance, concept drift detection, time series regression

1. INTRODUCTION

The motivation for this study originates from the trending concept of Predictive Maintenance (PdM), one of the key topics in the ongoing Industry 4.0 movement (Lee et al. 2014). In contrast to corrective maintenance and the established fixed-interval based preventive maintenance, PdM takes it one step forward and aims at preventing breakdowns of production plants by taking a plant's current and previous condition into account. The implementation of Predictive Maintenance requires collecting operational data and hence, to attach sensor equipment to the plant in order to keep track of its condition over the time. Downstream analysis and reasoning systems continuously evaluate the sensed time series in order to detect factual need for maintenance and subsequently determine a proper time slot to trigger correcting actions. Besides reducing the chance for total outages, such a real-time acting strategy also prevents unnecessary overhauls, which leads to improved predictability and higher productivity.

For the data analysis part frequently machine learning models such as Random Forests, Linear Regression, Bayesian Networks (Mattes et al. 2012), Markov-Models (Cartella et al. 2015), or Support Vector Machines (Widodo and Yang 2007) are applied. Regarding the underlying problem of time series forecasting Ahmed et al. 2010 provides a comparison of eight standard models on several macroscopic, economic data series. In recent work of Graff et al. 2017 also the forecasting capabilities of Genetic Programming are compared to Auto Regressive Moving Average (ARMA) models. Krawczyk et al. 2017 surveys the characteristics of non-stationary data streams before discussing several ensemble learning approaches for classification and regression tasks. Also Winkler et al. 2015 investigates data streams with changing behaviour and adapts a Genetic Programming based learning technique to detect such drifts during the model training phase.

This study differs from the stated literature in a way that it aims to combine several of the mentioned aspects by asking the specific question how efficient different machine learning techniques are at detecting changing behaviour in sensor time series. Since one can assume that the detection of such unexpected drifts might give a good indicator for necessary checks on the real plant, i.e. the time series streaming source, this methodological comparison aims to be a valuable contribution to the currently intensively investigated topic of Predictive Maintenance.

In Section 2 of this work we present synthetically generated data sets, which have been created following a reverse modelling approach in order to obtain realistic sensor time series. This includes a method to introduce concept drifts representing characteristic deterioration of plant conditions. Further on, Section 3 outlines a general strategy for data preprocessing, training of various machine learning regression models and their online evaluation on the generated time series in order to compare their competitiveness regarding drift detection. The potential of each algorithm for the specified task is analysed based on experiments in Section 4. Herein, results of the conducted tests are illustrated and the actual comparison is performed. Conclusively, Section 5 provides a brief summary of the presented content and some ideas for successive work, which might enhance this study regarding both the presented methods and application scenarios.

2. EXPERIMENT SETUP

Generating reasonable time series for industrial condition monitoring is a non-trivial task because there is generally only little high quality real-world data available to orient to and even less which was authorized for publication. However, such synthetic data sets are necessary for algorithm development, since they guarantee reproducibility and controllability, which was our motivation for proposing following synthesizing method.

2.1. Reverse Data Modelling

In most of the real-world sensor data sets, which we received from our partners in production industry, we observed some sort of cyclic behaviour (e.g. production of one unit), stationarity (e.g. regulated climatic conditions), seasonal effects (e.g. light conditions), trend (e.g. wear and tear) and noise (i.e. measurement deviation), all of which and many more aspects to consider when generating synthetic streams. Based on these observations, we developed a methodology to reproduce realistic uni- and multivariate time series and implemented it in form of a publicly available software application (https://github.com/smartfactorylab/dsg). Further details concerning the functionality of the application can be found in Zenisek et al. 2018. The goal of this approach is to enable the reuse of mathematical models – e.g. results from applying the Box-Jenkins method (Box and Jenkins 1970) or computing Fourier series by harmonic analysis (Stein 1993) on real-world time series – in order to produce new synthetic time series with the characteristics of the original data. In the following, the modelling approach is briefly summarized on the basis of an example.

Within the scope of this work we consider a sinusoid formula (1) as representation of a production plant's periodic behaviour, similar to what we obtained from a real-world data set.

$$cy_t = 10 + 2 * \left(\sin\left(\frac{t}{5}\right) + \frac{\sin\left(\frac{t}{2.5}\right)}{2} + \frac{\sin(t)}{7} \right) \quad (1)$$

In (1) t refers to an integer counter variable which starts at *1* and increases by *1* for each generated value. Starting from this basic signal, arbitrary time series may be derived in order to receive more or less correlated features. However, in the context of this work we initially consider a univariate problem and hence, derive only one series. The subsequent definition (2) of an exemplary autoregressive moving-average (ARMA, cf. Box and Jenkins 1970) model represents a sensor attached to the previously conceptualized plant and therefore, depends on the defined cycle to some extent.

$$s_t = c + \varepsilon + P + Q + C \quad (2)$$

$$with \quad c = 5.0, \quad \varepsilon \sim N(0, 0.2), \quad P = [0.35, 0.2, 0.1],$$
$$Q = [0.2, 0.15], \quad C = [0.75, 0.35, 0.1]$$

Equation (2) illustrates the definition of an ARMA model with a constant value c, a normal distributed error term ε and the weighting vectors P for the autoregressive and Q for the moving-average part, which determine the impact of past values and error terms. In line with this, we add the cyclic series' (1) last *3* values with the respecting weights in vector C to the model. Moreover, we introduce two additional time series, both same configured as the series in Equation (2), but with an increased error term, simulating higher noise levels: $\varepsilon \sim N(0, 0.35)$ and $\varepsilon \sim N(0, 0.5)$. Short sample segments for all three versions of the cycle derived series (cf. *s0.2*, *s0.35*, *s0.5*) are depicted in Figure 1, together with the cyclic series *cy* itself.

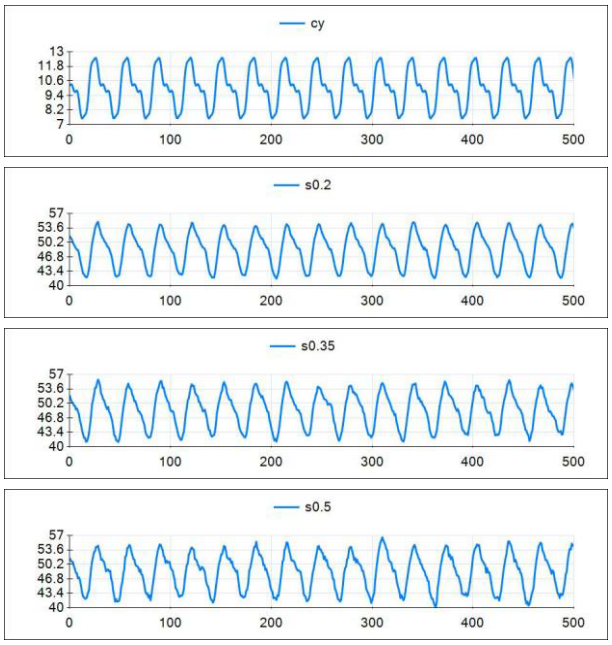

Figure 1: Basic plant cycle *cy* and the derived ARMA series *s0.2*, *s0.35* and *s0.5*

The described procedure is related to the approach of modelling ARMA models with exogenous parts (cf. ARMAX, Bauer 2009). However, instead of fitting a model to existing time series, we reverse the process and use synthetic or previously built models to produce new, reasonable series. The shown models are inspired by those that we modelled based on a real-world sensor signal. Due to the dependencies of a time series' current value to its own predecessors and values from exogenous series, it might take several sample generations to reach a stationary level. Therefore, we perform a burn-in phase and pre-generate values, which are not exported to the subsequently considered data stream.

2.2. Introducing Concept Drifts

In the context of PdM, a concept drift means an unwanted change regarding the behaviour of a monitored plant, caused by expectable gradual deterioration and hard-to-predict abrupt fractures. Despite this, usual definitions also refer to change of seasonality, which e.g. might affect climate related time series, as concept change

(Tsymbal 2004). In this work, wear and tear are conceived as long-term trend in recorded data, while damage caused by rather abrupt fatigue failure might be indicated by a short-term trend. Concerning the degradation course, most commonly models with exponential or logistic behaviour have been found to suit well for describing the so-called *run-to-failure*. While Ryll and Freund (2010) illustrate a macroscopic model with a sigmoid form for describing increasing wear, Saxena et al. surveys several mathematical models with exponential characteristics for damage propagation. However, also more linear progression might fit for some cases, as for instance considering friction linings, which degrade in a constant fashion when stressed.
Based on these considerations, we developed two functions representing one gradual (3) and one more abrupt (4) increase of wear.

$$d1_t = \frac{\tanh\left(\frac{t-150}{50} - 3.5\right)}{2} + 0.5 \quad (3)$$

$$d2_t = \frac{\tanh\left(\frac{t-150}{20} - 3.5\right)}{2} + 0.5 \quad (4)$$

Both functions are dependent to the same counter variable *t* as the functions *cy* and *s*, and generate values from 0 to 1 with logistic behaviour (cf. Figure 2). However, the first 150 values of series *d1* and *d2* are set to 0.0 in order to indicate normal behaviour in the beginning and the functions are only used to produce values after *t=150*.

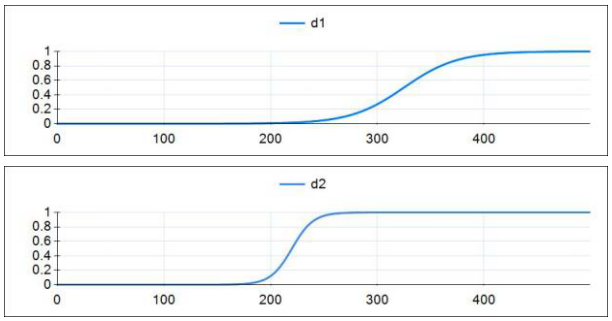

Figure 2: Deterioration signals *d1* and *d2*

In order to simulate the growing effect of deterioration in the data stream, we modified the presented basic plant cycle signal (1).
In Equation (5) we adapted the second and third sinusoid term of the plant cycle by including the current value of the deterioration series *d1*. By this means the impact of the second term is slowly reduced, while the impact of the third term increases.

$$cy_t = 10 + 2 * \left(\sin\left(\frac{t}{5}\right) + \frac{\sin\left(\frac{t}{2,5}\right)}{2+d1_t*0.5} + \frac{\sin(t)}{7-d1_t*5}\right) \quad (5)$$

As illustrated in Figure 3, the resulting signals' progression appeal noisier than their pristine ancestors (cf. Figure 1). While the drift is visually still clearly recognizable for signal *s0.2 (d1)*, it is almost swallowed by the larger noise in *s0.5(d1)*.

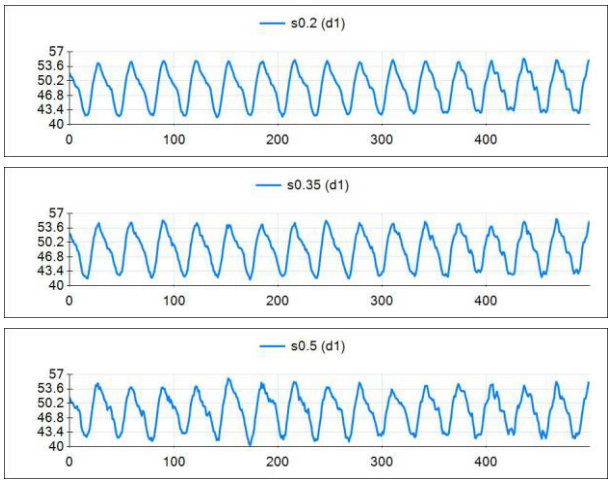

Figure 3: Gradually drifting cycle-derived series

The more abruptly increasing deterioration signal *d2* modifies the cycle by introducing an additional sinusoid term as depicted in Equation (6).

$$cy_t = 10 + 2 * \left(\sin\left(\frac{t}{5}\right) + \frac{\sin\left(\frac{t}{2,5}\right)}{2} + \frac{\sin(t)}{7} - d2_t * \frac{\sin\left(\frac{i}{3,5}\right)}{5}\right) \quad (6)$$

The introduction of this new harmonic is motivated by the consideration of real-world failures which might bring a system "out of beat" rapidly, e.g. due to a deposition.

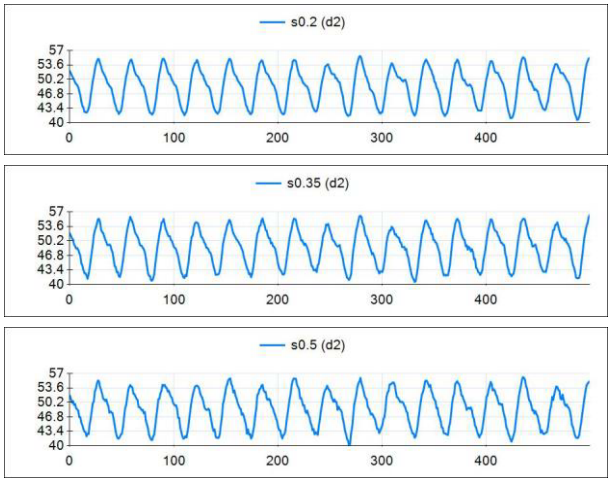

Figure 4: Abruptly drifting cycle-derived series

A similar approach to change a synthetic system's behaviour over time has been used by Winkler et al. 2015 for linear concept drifts. However, the various characteristics of the drifting series presented in this work (gradual/abrupt, noise level, type of change) provide a more realistic testbed for the subsequently described methods.

3. METHOD

The goal of the subsequently proposed strategy, regardless of the employed machine learning algorithm, is to model the state of a plant in which it is allegedly working "normal" even under varying ambient terms. Eventually, the strategy enables to alert if this concept (cf. "normal") starts to drift towards an "erroneous" or "unknown" state. There are basically two approaches for detecting drifting behaviour in time series.

- Classifying time series either as "normal" or "erroneous" based on models such as decision trees or instance-based comparison algorithms.
- Forecasting values of a series in an *n-step-ahead* fashion with regression algorithms and comparing predicted values with actually sensed ones; Herein, an increasing prediction error indicates a drifting concept.

Training these models, or respectively, building proper instance pools, requires data sets with correctly classified sample series. In this work we focus on the second approach, since we see more potential in reasoning on the base of the real-valued prediction error. The predictions are initially performed in a *1-step-ahead* fashion, meaning that the prediction error is calculated as difference between the algorithm forecast and the next value in the original series, which has not yet been available to the algorithm. However, we also consider *n-step-ahead* forecasting with varying *n* in a smaller experiment sequence in Section 4.3. The employed algorithms are trained on data streams without drifting concept and tested against streams with one. In Figure 5 the overall methodological approach of this comparison is illustrated. The subsequent Sections elaborate on the depicted system components.

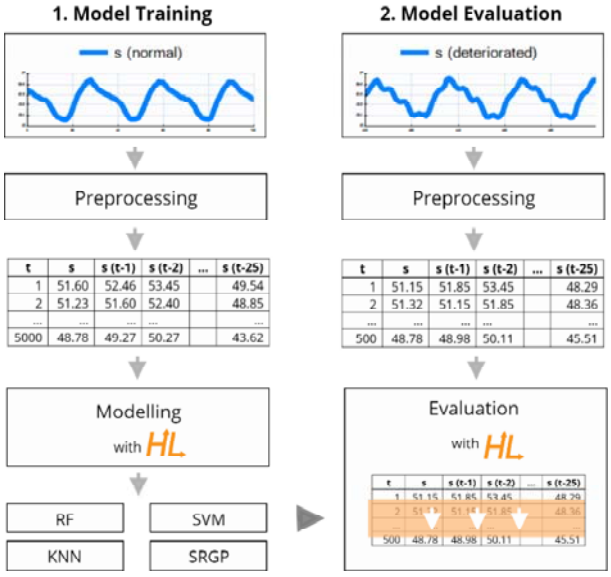

Figure 5: From model training to evaluation

3.1. Preprocessing

The success of machine learning algorithms heavily depends on the quality and form of the available data. Since solely synthetic data is used for this comparison, usually necessary tasks like handling missing values, outliers or time-related irregularities are no issue. Nevertheless, especially when dealing with time series, most algorithms such as those we employed for this study require transforming the data in some way beforehand, or otherwise they will not be able to build prediction models at all.

Among the most frequently applied preprocessing techniques on time series is their decomposition into parts for season, cycle, trend and noise, e.g. by using local regression based *STL* (Cleveland et al. 1990). Such transformations are well studied on economic data, however, require exact parametrization e.g. regarding seasonality – the periodic plant cycle in the context of this study – and therefore, further intermediate steps and tests. Instead, this work aims for comparing the algorithms' capabilities of modelling exactly such characteristics without preliminary investigation. In case they are not capable to do so automatically, it might be interesting if these algorithms can compensate by using other data transformations. Furthermore, any decomposition causes additional time-consumption, since more series emerge. This might hamper the already time-sensitive online condition monitoring situation, where new data arrives at least every second. Nevertheless, we plan to investigate opportunities with decomposition techniques in future work.

Similar to the strategy Ahmed et al. (2010) proposed for the related purpose of plain time series forecasting, we focus on the applicability of using time lagged values and performing series differencing as preprocessing, but for concept drift detection. Herein, time lagged values are extracted by simply transposing the historically seen last *n* values in a series to a new feature vector, which is subsequently provided to the machine learning algorithm. In analogy, a similar lagged-value vector may be built by computing the differences between the last *n+1* values. Accordingly, machine learning models built with a differentiated input vector forecast the next gradient instead of the upcoming real value. For this preprocessing step we performed the *5-point-stencil* (cf. Equation (7)), which is a more sophisticated and robust variant for numerical differentiation. The stencil is used to approximate the first derivative for the function of the real-valued series *s* at the time *t* by using four neighbouring points of *t* in *s*.

$$s'_t = \frac{-s_{t-2} + 8*s_{t-1} - 8*s_{t+1} + s_{t+2}}{12} \quad (7)$$

For all the experiments we performed, the time lag maximum of unprocessed and differentiated series was set to *n=25*. Considering a series with 5000 events, this results with a matrix consisting of 5000 rows and 26 columns (i.e. s, s_{t-1}, s_{t-2}, … s_{t-25}).

ISBN 978-88-85741-03-4; Affenzeller, Bruzzone, Jiménez, Longo, Merkuryev and Piera Eds.

3.2. Machine Learning Algorithms

For this comparison a small set of prominent machine learning algorithms, which either train models (e.g. trees), or use training instances (e.g. by averaging samples) to perform time series regression has been compiled. For more details regarding the algorithms, the reader is referred to the stated literature.

- K-Nearest-Neighbor Regression (KNN), instance-based, Altman (1992)
- Support Vector Machine Regression (SVM), meta-instance-based, Cortes and Vapnik (1995)
- Random Forest Regression (RF), model-based, Breiman (2001)
- Symbolic Regression by Genetic Programming (SRGP), model-based, Koza (1992), Affenzeller et al. (2009)

All experiments have been performed with the open source framework HeuristicLab (Wagner et al. 2014, https://heuristiclab.com). Although the utilized algorithm implementations correspond to the state of the art, there exist many variants, strongly adapted for specific problem domains. However, this study employs basic forms in order to allow more general statements regarding the capabilities of each algorithm type and to ease reproducibility. For similar reasons, this work also does not concentrate on tuning algorithm parameters extensively.

3.3. Evaluation

The algorithms we used for this comparison are among the most popular ones for time series forecasting (cf. Ahmed et al. 2010, Affenzeller et al. 2009) and learning under the influence of concept drifts (cf. Krawczyk et al. 2017, Winkler et al. 2015). This study aims to shed light on how efficiently they can be applied on the task of the detection of concept drifts. To a large extent, this depends on how these algorithms are evaluated on a constantly updated data stream.

As mentioned at the beginning of this section, we monitor the relation between the model forecasts and the factual values. If the prediction error increases for a certain period, models do not fit the current data anymore, which might indicate a concept drift. For a fair comparison, this obviously requires that all models were capable of performing forecasts with high accuracy in the first place, i.e. on training data which is qualified as normal behaviour. In order to mitigate being prone to outliers or short-term anomalies and to make the general trend more visible, we use a sliding window evaluation schema. This way, the last m forecasts are compared to the actually sensed values and the Pearson Correlation Coefficient PR^2 is computed and averaged. For all experiments we performed, the window size was set to $m=25$ and the window movement step size to 1.

Depending on the accuracy results achieved on the training data, a properly set lower-bound threshold might be reasonable to determine the "point of failure" in case of the threshold's transgression. The algorithm comparison could be performed by measuring the timestamp for this transgression regarding each model and the factual start of deterioration in series d. Another way to accomplish the algorithm detection capability comparison we aim for is to check against the entire actual drift function, which is available in such a synthetic scenario. Therefore, we compute the correlation of the prediction quality progress and the deterioration series d. By this means, high negative correlation values indicate good algorithm performance in the presence of a drifting time series.

4. EXPERIMENTS AND RESULTS

In the following section the conducted experiments are documented and the collected results are discussed.

4.1. Model Training

For the determination of reasonable algorithm settings we performed a parameter grid search by creating experiments using HeuristicLab. The scope of this search orients on comparisons in the work of Ahmed et al. 2010 and Winkler et al. 2015 and is listed in Table 1.

Table 1: Settings of the parameter grid search

KNN	K: 2 … 100
SVM	γ: 0.5, 0.75, 1, 1.25, 1.5; ϵ: 0.05, 0.1, 0.25, 0.5, 1, 2, 4; C: 1, Degree: 3; Nu: 0.5; RBF Kernel
RF	Trees: 100, 250, 500, 1000; R: 0.2, 0.3 … 0.8, M: 0.4, 0.5 … 0.8
SRGP	Tree Length/Depth: 25, 35, 50/50; Symbols: +, -, *, /, sin, cos, exp, log; PopSize: 100; Generations: 1000; MutRate: 15%; CrossRate: 100%; Proportional Selector; Offspring Selection; SelPressure: 100

Each individual configuration has been tested by performing 10 repetitions, with exception of the KNN algorithm, as it does not base on any kind of stochasticity. The search has been performed on each of the produced non-deteriorated data sets (cf. *s0.2*, *s0.35* and *s0.5*) independently and the most accurate models have been picked for the subsequent evaluation. Genetic Programming tends to be more prone to stochastic effects due to its evolutionary process for model creation. Hence, we also provide the average model accuracy of GP's top 10 models (cf. øSRGP) which is a common practice and known as ensemble modelling.

The forecasting accuracies of the best models on data sets processed by building time lagged value vectors are listed in Table 2. Although, there is a slight decrease of accuracy aligned to the level of noise in the used data set, all of the algorithms accomplished creating very accurate models.

Table 2: Algorithm forecasting accuracy as PR^2 on data sets with time lagged input vector

PR^2	KNN	SVM	RF	SRGP	øSRGP
s0.2	0.9961	**0.9972**	0.9970	0.9963	0.9954
s0.35	0.9891	**0.9918**	0.9917	0.9907	0.9889
s0.5	0.9740	0.9809	0.9809	**0.9810**	0.9793

In Table 3 the accuracies for models trained on the differentiated series are presented. The decrease of accuracy linked to noise is herein significantly higher, however, still good models could have been developed.

Table 3: Algorithm forecasting accuracy as PR^2 on data sets with differentiated time lagged input vector

PR^2	KNN	SVM	RF	SRGP	øSRGP
s0.2	0.9711	**0.9746**	0.9711	0.9649	0.9546
s0.35	0.9186	**0.9287**	0.9200	0.9054	0.8837
s0.5	0.8455	**0.8622**	0.8481	0.8239	0.8050

Overall, the SVM algorithm created the most accurate models, although, the performance of the others is on the same level, since the quality difference is only marginal in terms of real numbers. All of the listed values originate from the evaluation of solely the test partitions (34 %) of the compiled 5000-event long data series with no concept drift present.

4.2. Simulation-Based Evaluation

In order to perform the described evaluation process, we implemented a simulation environment on the base of HeuristicLab, which is now publicly available. Within a run of this simulation, a prepared data set is replayed value by value such that the preliminary trained models are evaluated on a continuously progressing stream, as they would be in a real online scenario. In this course, the previously mentioned evaluation criterion as well as several plots and statistics for further investigation are constantly updated (cf. Figure 6, 7, 8). The reader is referred to Figure 5, where the overall process from model training to stream-wise evaluation is depicted.

We tested the algorithms on a drifting stream (cf. Equations (5) and (6)) with 500 events. Further on, we computed the correlation of prediction quality and deterioration (cf. *PR*) as proposed in Section 3.3. Again, all of the tested algorithms obtained good results, which means that they are all capable of detecting the introduced drift – their prediction quality negatively correlates with the course of the drift. Figure 6 visually supports this inference, since the decay of prediction quality can be clearly observed.

Table 4 lists the correlation values of the decreasing model accuracy and the increasing drift on data sets with different noise level (cf. *s0.2*, *s0.35* and *s0.5*), preprocessed by building lagged value vectors and most importantly, including two kinds of concept drifts (cf. *d1* and *d2*). Table 5 solely differs in the preprocessing method applied on the data set. The results in both tables show that overall, each of the employed algorithms is capable of achieving sufficiently high prediction accuracy in most of the data sets.

Table 4: Algorithm concept drift detection performance on data sets with time lagged input vector

PR	KNN	SVM	RF	SRGP	øSRGP
$s0.2_{d1}$	-0.8519	**-0.8587**	-0.8066	-0.774	-0.7564
$s0.35_{d1}$	**-0.8476**	-0.8362	-0.8401	-0.7695	-0.7596
$s0.5_{d1}$	-0.7139	-0.6717	**-0.7226**	-0.2009	-0.5452
$s0.2_{d2}$	**-0.6342**	-0.554	-0.6199	-0.4835	-0.3426
$s0.35_{d2}$	**-0.5548**	-0.4964	-0.4643	-0.4402	-0.3769
$s0.5_{d2}$	**-0.4813**	-0.2998	-0.2516	-0.0485	-0.0378

In line with decreasing model training accuracy on noisy data sets, also the concept drift detection performance decays, but much faster. Furthermore, the abrupt concept drift *d2* seems to be much harder to detect than the gradual *d1*. The best detector is KNN, whereas SRGP is the worst, especially on the noisy, *d2*-influenced data sets, where a detection is almost not possible anymore. The values of Table 4 are also visually presented in Figure 6.

Table 5: Algorithm concept drift detection performance on data sets with differentiated time lagged input vector

PR	KNN	SVM	RF	SRGP	øSRGP
$s0.2_{d1}$	-0.8422	-0.8235	**-0.8458**	-0.0346	-0.6083
$s0.35_{d1}$	-0.8338	-0.8439	**-0.8478**	-0.5850	-0.6573
$s0.5_{d1}$	**-0.7077**	-0.5991	-0.6267	-0.1321	-0.2326
$s0.2_{d2}$	-0.3948	**-0.4044**	-0.4021	-0.1802	-0.0844
$s0.35_{d2}$	-0.2444	**-0.2563**	-0.2454	-0.1286	-0.1599
$s0.5_{d2}$	-0.2068	-0.2238	**-0.2395**	-0.0927	-0.1149

In Table 5 one can observe similar detection behaviour for the differentiated series. However, this time RF, SVM and KNN perform on the same level, while SRGP trails behind.

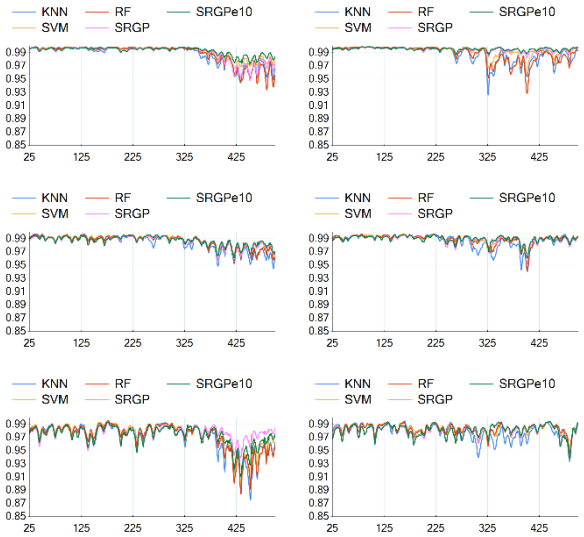

Figure 6: Progress of PR^2 received from model evaluation in a sliding window (size *m=25*) fashion; The charts visualize the results from Table 4 with *d1* left- and *d2* right-hand-side.

4.3. N-Step-Lookahead

In a smaller experiment setup we tested how the algorithms perform with a larger forecasting horizon. Therefore, we prepared the *s0.5* data set with a larger offset between the target value and the lagged input vector and trained new prediction models. The accuracy of these new models slowly decays with increasing forecasting horizon, however, still very accurate models

with all tested machine learning algorithms could be developed (cf. Tables 6 and 7). The reason for this might be the strong periodic behaviour of the trained signal and only minor influence of noise.

Table 6: Algorithm forecasting accuracy as PR^2 on data set *s0.5 (d1)* with time lagged input variables and increasing lookahead horizon (1 to 25 – see first column)

PR^2	KNN	SVM	RF	SRGP	øSRGP
1	0.9740	0.9809	0.9809	**0.9810**	0.9793
5	0.9664	**0.9676**	0.9664	0.9557	0.9454
10	**0.9641**	0.9631	0.9632	0.9557	0.9520
15	**0.9652**	0.9650	0.9639	0.9564	0.9514
20	**0.9627**	0.9623	**0.9627**	0.9541	0.9512
25	**0.9618**	0.9612	0.9621	0.9535	0.9509

Table 7: Algorithm forecasting accuracy as Average Relative Error (ARE) in percentage on data set *s0.5 (d1)* with time lagged input variables and increasing lookahead horizon (1 to 25)

ARE (%)	KNN	SVM	RF	SRGP	øSRGP
1	1.085	**0.924**	0.925	0.927	0.972
5	1.249	**1.215**	1.236	1.429	1.571
10	**1.284**	1.298	1.299	1.399	1.472
15	1.280	**1.273**	1.288	1.447	1.516
20	1.298	1.308	**1.297**	1.455	1.486
25	1.357	1.386	**1.352**	1.479	1.531

Interestingly, the detection quality decays quite fast with increasing lookahead horizon (cf. Table 8), while the trained model accuracy remained rather stable (cf. Tables 6, 7). Finding reasons for this requires successive investigation, which will be part of future work.

Table 8: Algorithm concept drift detection performance on data sets with time lagged input variables

PR	KNN	SVM	RF	SRGP	øSRGP
1	-0.711	-0.6706	**-0.7239**	-0.215	-0.5451
5	-0.702	-0.7137	**-0.7201**	-0.4311	-0.6377
10	-0.5824	**-0.596**	-0.576	-0.3171	-0.1047
15	**-0.6214**	-0.5674	-0.4734	-0.2589	-0.0742
20	-0.5998	**-0.6402**	-0.594	-0.3087	-0.1939
25	-0.5778	**-0.6201**	-0.4905	-0.1041	-0.0471

The main advantage of an increased forecasting horizon is that the point of finding concept drifts may be earlier (cf. Figure 7), which is possible according to our tests. However, this is not in the focus of this study. Future work might also deal with the trade-off between forecasting horizon and detection quality.

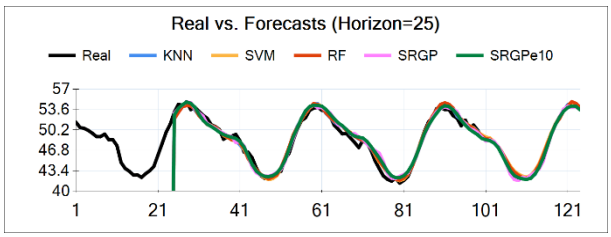

Figure 7: Real signal vs. 25-step-lookahead forecasts

4.4. Algorithm Comparison

After further investigating the simulation statistics and plots, we conclude that although all employed algorithms proved to be good detectors, their microscopic prediction behaviour differs to some extent. While KNN, SVM and RF naturally tend to produce rather "smooth" progressions, the symbolic regression model is more sensitive to changing points in the signal. In Figure 8, one can observe that the models do not change this behaviour when the concept drift is introduced. GP utilizes a symbolic vocabulary (variables, constants, trigonometric functions etc.) to build mathematical functions. With a similar vocabulary we generated the synthetic time series. Although GP supposedly has the power to capture the series' characteristics best due to this similarity, it is neither the first ranked predictor nor detector. It seems that the model adapts its prediction behaviour to the introduced drift and anticipates the drift in a lagged fashion. Since this behaviour is in fact not intended, we are looking forward to investigate how GP in particular and the other algorithms react to other training settings and differently preprocessed series in future work.

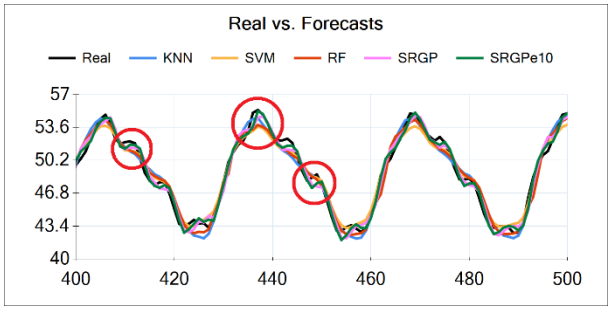

Figure 8: Real vs. forecasted values with introduced drift

In terms of runtime performance, SVM and RF are faster in training than SRGP by far. For KNN there is no training at all, since it performs distance comparisons during evaluation time. This becomes an important detail when considering the necessity for iteratively training models in online situations. Despite this, the evaluation of symbolic regression models is quite fast, since it requires solely computing a single mathematical function. In contrast, the runtime for KNN increases dramatically with additional dimensions, larger instance pool and a higher number *k* of samples to consider.

5. CONCLUSION AND OUTLOOK

In the first part of this study we presented an approach to synthesize time series with and without drifting concepts. On the basis of these data sets machine learning models have been trained and a methodology to test and compare the trained models capabilities for concept drift detection has been depicted. Subsequently, the results of the conducted experiments have been discussed. The main contribution of this work is the developed and implemented evaluation methodology in form of a simulation, which might be of value as a testbed for upcoming *what-if* scenario experiments. Furthermore,

we have shown, that all employed algorithms are able to produce models, which may serve as quite good concept drift detectors. However, their performance decays rapidly with increasing noise, the speed of the drift and the forecasting horizon.

Based on these promising results and initial algorithm assessments, we plan to perform further larger scaled and differently configured experiments in a subsequent study. In this scope we aim for additional insights regarding the strengths and weaknesses of one or another algorithm and preprocessing method on more complex problem instances. Future work might also tackle the following issues:

- Dealing with multivariate time series
- Testing against real-world data
- Enlarging the comparison regarding the utilized machine learning algorithms
- Applying decomposition techniques as additional preprocessing step
- Employing ensemble models for more robust forecasting and new evaluation possibilities
- Testing *rolling* horizon evaluation

ACKNOWLEDGMENTS

The work described in this paper was done within the project "Smart Factory Lab" which is funded by the European Fund for Regional Development (EFRE) and the state of Upper Austria as part of the program "Investing in Growth and Jobs 2014-2020".

REFERENCES

Affenzeller M., Wagner S., Winkler S. and Beham A., 2009. Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications. Chapman & Hall / CRC Press.

Ahmed N. K., Atiya A. F., Gayar N. E. and El-Shishiny H., 2010. An empirical comparison of machine learning models for time series forecasting. Econometric Reviews, 29 (5-6), 594-621.

Altman N. S., 1992. An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician, 46 (3), 175-185.

Bauer D., 2009. Estimating ARMAX systems for multivariate time series using the state approach to subspace algorithms. Journal of Multivariate Analysis, 100 (3), 397-421.

Box G. E. and Jenkins G. M., 1970. Time series analysis: forecasting and control. Holden-Day, San Francisco, CA.

Breiman L., 2001. Random forests. Machine Learning, 45 (1), 5-32.

Cartella F., Lemeire J., Dimiccoli L. and Sahli, H., 2015. Hidden semi-Markov models for predictive maintenance. Mathematical Problems in Engineering, 2015.

Cleveland R. B., Cleveland W. S. and Terpenning I., 1990. STL: A seasonal-trend decomposition procedure based on loess. Journal of Official Statistics, 6 (1), 3.

Cortes C. and Vapnik V., 1995. Support-vector networks. Machine Learning, 20 (3), 273-297.

Graff M., Escalante H. J., Ornelas-Tellez F. and Tellez E. S., 2017. Time series forecasting with genetic programming. Natural Computing, 16 (1), 165-174.

Koza J., 1992. Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, 1992.

Krawczyk B., Minku L. L., Gama J., Stefanowski J. and Woźniak, M., 2017. Ensemble learning for data stream analysis: A survey. Information Fusion, 37, 132-156.

Lee J., Kao H. A., and Yang S., 2014. Service Innovation and Smart Analytics for Industry 4.0 and Big Data Environment. Procedia Cirp, 16, 3-8.

Mattes A., Schöpka U., Schellenberger M., Scheibelhofer P. and Leditzky G., 2012. Virtual equipment for benchmarking predictive maintenance algorithms. In Simulation Conference (WSC), Proceedings of the 2012 Winter, pp. 1-12. IEEE.

Ryll F., and Freund C., 2010. Grundlagen der Instandhaltung. In Instandhaltung technischer Systeme, 23-101. Springer, Berlin, Heidelberg.

Saxena A., Goebel K., Simon D. and Eklund N., 2008. Damage Propagation Modeling for Aircraft Engine Run-To-Failure Simulation. Prognostics and Health Management, 2008. Proceedings of 2008 IEEE International Conference on PHM, 1-9.

Stein E. M., 1993. Harmonic Analysis: Real-Variable Methods, Orthogonality, and Oscillatory Integrals. Princeton University Press.

Tsymbal A., 2004. The problem of concept drift: definitions and related work. Computer Science Department, Trinity College Dublin, 106 (2).

Wagner S., Kronberger G., Beham A., Kommenda M., Scheibenpflug A., Pitzer E., ... and Affenzeller M., 2014. Architecture and Design of the HeuristicLab Optimization Environment. In Advanced Methods and Applications in Computational Intelligence, pp. 197-261. Springer, Heidelberg.

Winkler S.M., Affenzeller M., Kronberger G., Kommenda M., Burlacu B. and Wagner S., 2015. Sliding window symbolic regression for detecting changes of system dynamics. Genetic Programming Theory and Practice XII. Springer 91-107.

Widodo A., and Yang B. S., 2007. Support vector machine in machine condition monitoring and fault diagnosis. Mechanical systems and signal processing, 21 (6), 2560-2574.

Zenisek J., Wolfartsberger J., Sievi C. and Affenzeller M., 2018. Streaming Synthetic Time Series for Simulated Condition Monitoring. IFAC-PapersOnLine, 51, pp. 6.