Tangent Works

September 2019, Issue 1

Benefits of Truly Automated AI: Why is TIM for Time-Series Unique?

Ján Dolinský

Robert Tóth

Table of Contents

1 Introduction

This paper describes peculiarities of time-series modelling and anomaly detection that are difficult to address using AutoML. The focus of AutoML is the selection of an appropriate modelling technique and its hyper-parameters suitable for the task at hand. In this effort, AutoML solutions scan through many different ML libraries, create models, and tune their corresponding hyper-parameters. This has traditionally been a laborious task.

In time-series modelling, however, identification of significant features and an overall modelling framework (how to address changing dynamics in time-series, dynamic data availability, multi-point and multi-situational forecasts, etc.) are far more important than a choice of a modelling technique and its associated hyper-parameters.

Tangent Works (TW) identified this challenge and developed a model-building framework that addresses time-series modelling problems. TIM (Tangent Information Modeler) is an automatic model building engine for time-series and anomaly detection where a single pass through the data generates one single high-quality model. The term *InstantML* is introduced here to refer to modelling strategies that identify features rather than a modelling technique and its hyper-parameters and that are significantly faster (requiring only seconds, up to few minutes on a standard hardware) than a typical AutoML session. TIM as an example of an InstantML strategy is fast and can identify new features and create a model on-demand, even each time a forecast is required. This enables *instant forecasting* where the user is not even exposed to a model, which can be thrown away right after the forecast is calculated.

TIM has its companion TIM Studio that is designed to serve as a machine learning development and operations (ML DevOps) platform for time-series modelling. TIM and TIM Studio have an API that enables integration with other tools and business processes. This allows building an end-to-end solution that covers data preparation, business process orchestration including continuous data integration, automatic model building (development), and forecast calculation (or anomaly detection). The API offers user administration and other related functionalities, too.

TIM is configurable for advanced users who might want TIM to test their own features instead of specialised features built in for business users in a particular industry. Open dictionaries* of built-in features can be augmented or replaced with user-defined features via an API or TIM Studio. TIM and its open dictionaries offer data scientists the adventure of conquering an entire industry rather than a single use case or a dataset at hand.

In business user mode, TIM generates high-quality models with zero degrees of freedom, meaning that no tuning of engine parameters is required from a user. TIM has been tested numerous times. It is worth mentioning that TIM in business user mode won the General Energy Forecasting Competition (GEFCom) 2017 where TIM built over 150 models. TIM also won the Hokkaido Electric Power Company (HEPCO) 2019 competition, and when TIM is applied

to the 3003 time-series data of the past M3 competition it ranks neck and neck with the previous winner.

* Planned as a feature in 2020.

2 Time-Series Problems are Complex and Dynamic

2.1 Changing Dynamics in Time-Series Problems

Phenomena represented by time-series are dynamic. A model that worked yesterday may not be up to date anymore. There are numerous examples from different industries where this is the case: in finance, forecasting a changing portfolio of assets, in trading where the influencing factors change so often that model re-building from scratch is mandatory, in energy where portfolios of production assets (e.g. solar farms) change occasionally, to name some examples.

Figures 1 and 2 show examples of such time-series. Figure 1 plots an electricity consumption time-series and Figure 2 an aggregated gas consumption. The two signals have one thing in common – structural changes. The structural change can be of various types - changes in mean, variance, seasonality, etc. Sometimes the change in structure is so radical, that it appears to be a completely new and different time-series to the naked eye.

When trying to teach ML algorithms how to play “Go” or how to distinguish between images of lungs to detect lung cancer, more data samples typically make you happy, because they enhance performance of your model. This, in general, is not the case when modelling time-series. New observations in your time-series data might give you a headache because your model (possibly identified by AutoML) may become useless from one hour to the next. Repeating the AutoML search is an option that is often not an optimal one. These new situations require identifying new significant features rather than asking whether a neural net should be used again and whether it should be comprised of two hidden layers instead of one.

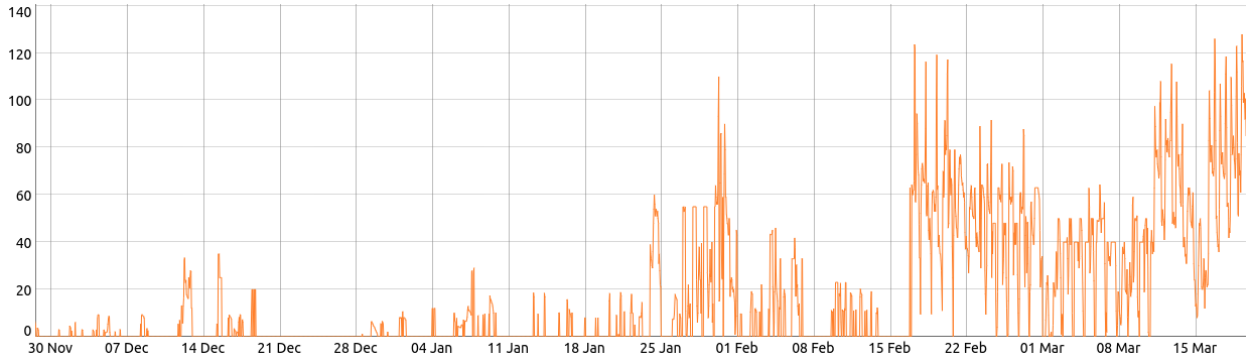

Figure 1: Electricity consumption time-series with a changing dynamics.

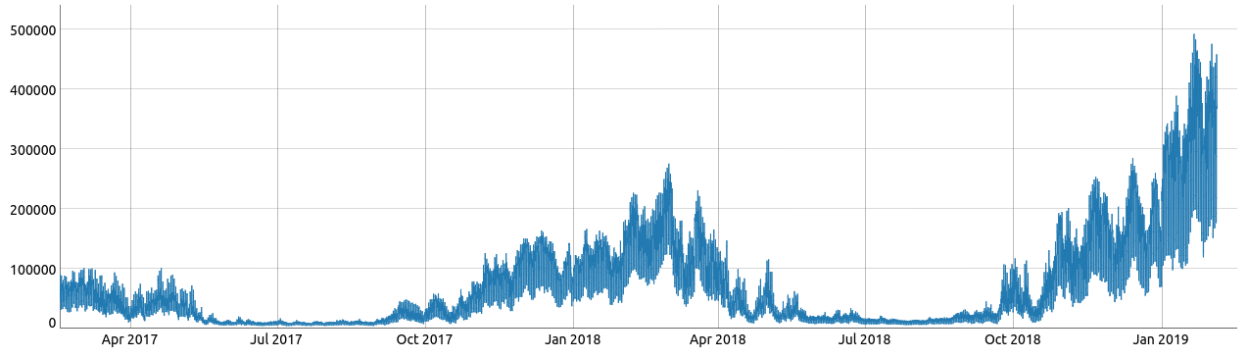

Figure 2: Aggregated gas consumption time-series where a portfolio has grown significantly between heating season 2017/2018 and 2018/2019.

TIM offers different ways to adapt to new situations. Models may be rebuilt or recalibrated. Model rebuilding identifies new features from scratch and builds a new model, whereas model recalibration adjusts model parameters only and leaves model structure (features) intact.

It is also crucial to understand when you should rebuild your models along with what data you should use when you do rebuild. TIM's anomaly detection functionality can help you in triggering the rebuilding or recalibration process and assessing what data should be used. Alternatively, rebuilding (or recalibration) can be scheduled on a regular basis (e.g. once per day, once per week, etc.) over a fixed chunk of data (e.g. a sliding window of one year).

In time-series, setting up a continuous model rebuilding pipeline is often a must to address structural changes and to maintain accuracy at the desired level. TIM automates the repetitive and "boring" tasks of forecasting and lets the engineer focus on the bigger picture.

2.2 Data Availability Problem

Another complexity that must be considered is the fact that how your data are available to your model may also vary from time to time. Figure 3 and 4 illustrate such a situation.

Your model discovers that ElectricityConsumption(t) = 10000*AverageTemperature(t) – 5000*Protests(t)

Date	Electricity consumption	Average temperature	Climate protests around the globe
1.1. 2017	154 879	20.6	11
2.1. 2017	187 566	20.9	0
3.1. 2017	175 798	22.7	2
4.1. 2017	155 555	16.5	5
5.1. 2017	165 378	14.2	3
6.1. 2017	144 444	19.6	12
7.1. 2017	179 631	21.5	1
8.1. 2017	198 755	23.9	1
9.1. 2017	133 568	23.2	23

Figure 3

It is 10th of January. You wish to forecast electricity consumption for 11th of January. Your model is useless now.

Date	Electricity consumption	Average temperature	Climate protests around the globe
1.1. 2017	154 879	20.6	11
2.1. 2017	187 566	20.9	0
3.1. 2017	175 798	22.7	2
4.1. 2017	155 555	16.5	5
5.1. 2017	165 378	14.2	3
6.1. 2017	144 444	19.6	12
7.1. 2017	179 631	21.5	1
8.1. 2017	198 755	23.9	1
9.1. 2017	133 568	23.2	23
10.1. 2017	?	?	5
11.1. 2017	????	?	?

Figure 4

Imagine that using available historical data, the following relationship is identified: "ElectricityConsumption(t) = 1000*AverageTemperature(t) – 5000*ClimateProtests(t)". It is now the 11th of January (see Figure 4), the values of both explanatory variables are unavailable, and therefore it is not possible to use the identified model and calculate a forecast. Repeating an AutoML search with the new constrains for calculating ElectricityConsumption(t), namely, including AverageTemperature available up to t-2 days ago and ClimateProtests up to t-1, is a time-consuming option with significant computational cost. Doing this each time the data availability changes is nearly impossible with an AutoML approach.

With TIM, users can define plausible ranges of lags for each variable in the model building process to ensure that the generated model complies with the production data availability. In some situations, however, observations in each column arrives in a database independently and asynchronously with no fixed data availability scheme in place. In such situations, it is affordable and advantageous to let TIM build a new model each time a forecast is required. This is referred to as *real-time InstanML* (RTInstantML). TIM examines the data to check current data availability and builds a model that complies just-in-time for a forecast calculation. In this way, TIM gets the most out of the data no matter what your current data availability is.

This behaviour is especially advantageous with IoT data streams that do not always reliably deliver the most up-to-date data for a forecast calculation, and in any situation where data availability cannot be guaranteed up front. It is also vital for building continuous time-series ML DevOps pipelines with these data uncertainties.

2.3 Multi-point Forecasts

Situations where a single-point forecast is required are rare in industrial practice. Multi-point forecasts are required in many industrial verticals. For example, at 8:00 you might want to

forecast the price of a commodity for each hour that day between 9:00 and 24:00, resulting in 16 points in total.

Multi-point forecasts have traditionally been addressed by multi-output models or recurrent strategies. Intuitively, building and optimizing a multi-output model is harder than a single output model because model parameters are optimized for each of the outputs, and this optimization happens simultaneously. Optimizing for several outputs simultaneously may sometimes result in a contradictory optimization problem. Multi-output models are often more complex (they might have more hidden layers in a neural net or a larger decision tree, etc.) and elucidating knowledge from such a model is difficult.

Recurrent strategies, on the other hand, optimize a single output model that is then propagated in time in a recurrent way; for example the forecast for y(t+1) is used again for calculating y(t+2). Recurrent strategies are, however, prone to diverge quickly after only a few update steps. This renders them impractical for broader industrial adoption.

TIM addresses multi-point forecasts by creating a set of multiple single-point models. Each point of a forecast has its own individual model that considers corresponding individual data availability constrains. This set of models is referred to as a ModelZOO. When forecasting, TIM takes the ModelZOO and dispatches the correct model for calculation of each point of a multi-point forecast. This approach has several advantages. First, by addressing each point individually, TIM can often achieve greater accuracy because the model is optimized for a single point and takes into account its individual data availability constrains. For instance, the price forecast for 9:00 may use an actual price from 8:00 (lagged feature of y(t-1)) which often helps, whereas the forecast for 10:00 cannot use y(t-1) because it is not available yet but can use y(t-2): This information is automatically recognized by TIM. Each model's features can also be examined and compared, if for example there are rather different features driving the price forecast at 9:00 compared to 12:00.

2.4 Multi-situational Multi-point Forecast

In many time-series applications, it is often important to issue a multi-point forecast several times per day (e.g. every hour). Each such a situation typically has a unique data availability scheme: For example, at 8:00 I "see" my target until t-6 hours ago, but at 9:00 the data in my database system gets updated with actuals up to t-2 hours ago. Models for 8:00 can take advantage of lagged features from t-6 hours and further back in time but models for 9:00 can take advantage of more recent data from the morning update, i.e. from t-2 and further in time.

Forecasting 10:00 at 8:00 and at 9:00 are therefore two significantly different situations.

A set of forecasting situations with their corresponding data availability schemas is referred to as a *forecasting routine*. TIM lets users describe their forecasting routine either graphically via the TIM Studio front-end or using an API. When training, TIM's multi-situational layer will assemble a ModelZOO that accounts for all the multi-point forecasts in all the situations of the

user's forecasting routine. When forecasting, TIM will dispatch the most appropriate model from the ModelZOO suitable for a situation at hand (e.g. it is now 9:00 and the forecast for 10:00 is calculated using the most appropriate model from ModelZOO) via its multiple model dispatch mechanism. This allows a user to get the most out of the data as each point in all the multi-point forecasts in all situations of a forecasting routine can be handled individually, considering corresponding data availability constrains.

2.5 Transparent Models: Let Your Data Speak

Models assembled in a ModelZOO by TIM are not black-box models. Features identified by TIM for each individual model in the ModelZOO are available to a user for inspection. TIM also extracts canonical features from a ModelZOO to provide a helicopter view on driving features of a dataset at hand.

TIM by default assembles general additive models (GAMs). The identified features may, however, be used as an input to other modelling techniques including AutoML pipelines. It is worth mentioning that doing this rarely results in an increase in performance (e.g. see modelling techniques used in GEFCom 2017) whilst it decreases automation significantly.

3 Instant Forecasting is Affordable

With TIM's InstantML, producing forecasts instantly from data is affordable. How does it work? TIM looks in your latest data and detects a corresponding data availability and forecasting routine out of it. There is no need to setup your data availability scheme for the situation you are currently in – it is defined by the way your data are organized and TIM will recognize this automatically. TIM will then generate a model and produce a forecast in one single API call. The model is then discarded, as the next time a forecast is required the entire process is simply repeated. This is inherently different to AutoML. All TIM needs to know from a user is how many steps ahead the forecast should be calculated.

This way of producing forecasts is referred to as *real-time InstantML* (RTInstantML). It is especially advantageous if your data availability scheme changes often (see also Section 2.2) or if forecasts are required ad-hoc, i.e. at any time of a day. In addition, RTInstantML does not require any model management nor model storage, which simplifies implementation complexity and shortens time to value.

4 Operational IT Benefits

4.1 Building Efficient Large-scale Forecasting Systems

Math within TIM makes TIM often much faster than a typical AutoML session. TIM does not require high-end hardware and runs well on a laptop or even on a mobile device. Nevertheless, the cloud architecture of TIM is designed to scale whilst using computational resources in a cost-efficient way. Figure 5 depicts TIM's architecture.

TIM provides a *single* API that supports a comprehensive set of ML operations (InstanML training, forecasting with a model, InstanML forecasting, anomaly detection, authorization, etc.). The application logic that provides TIM API is designed to scale to the number of incoming requests by means of a variable number of workers running. This is cost-efficient because you pay for the performance only when you need it. A job to train a large number of models (e.g., 1000) can be scheduled for a Sunday night and at that very moment TIM will spawn large number of workers to get the job done quickly. As soon as the forecast is completed, TIM will discard redundant workers and will run a conservative number of workers to save the infrastructure costs.

Having a scalable single API is also advantageous for model deployment at scale which is discussed next.

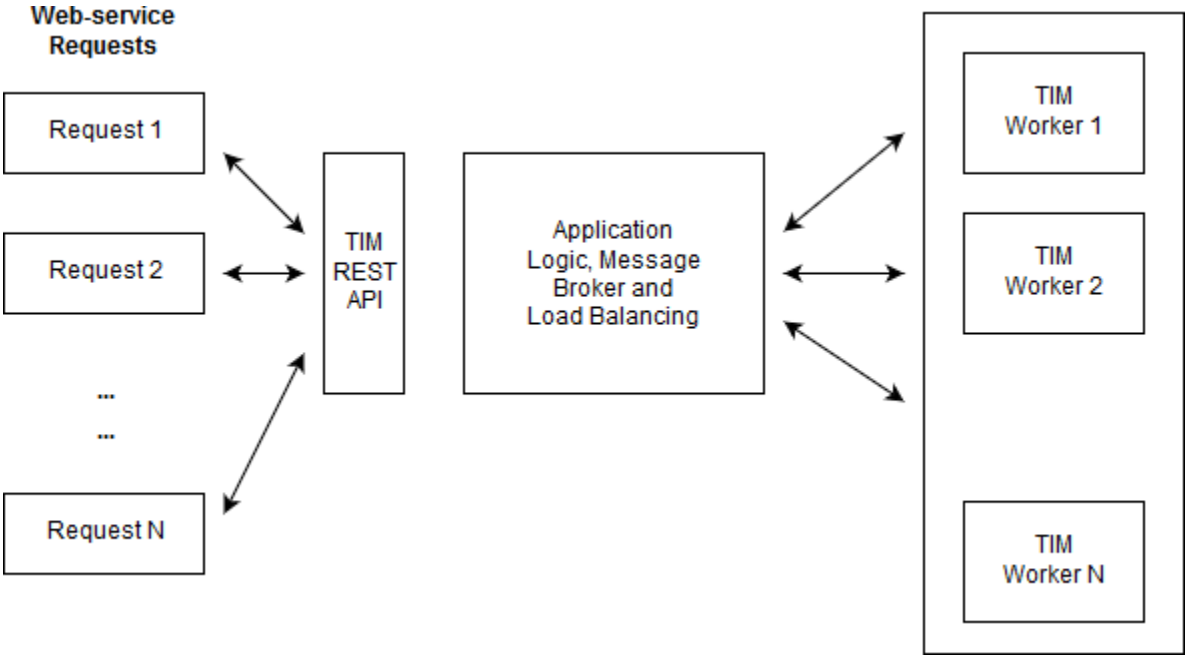

Figure 5: TIM's single API cloud architecture.

4.2 Model Deployment via a Single API: Large-scale Forecasting Perspective vs. Deploying Each Model Individually

There is a significant gap between developing a model and its deployment in a production use. Once a model is developed (e.g. in R or Python) there is no straightforward way of how to consume it and calculate a forecast or detect an anomaly by a business process that has access to most recent data. Most of the current platforms address this challenge by encapsulating in a container all the software components required for a model to run, including the model itself. Such a container typically has a REST API that then exposes the model to the outside world. Relevant data are then pushed to the model for a forecast calculation via the provided REST API.

This typical approach helps to narrow down the gap between developing a model (e.g. in R or Python) and deploying it in production. At the same time, however, it implies some limitations that are worth pointing out, especially when it comes to building of large-scale forecasting systems in a cloud infrastructure. The container storing a model of interest must be up and running each time a model is required. Imagine a large-scale scenario with 1000 models where you don't know when models will be required. In a cloud infrastructure, this results in 1000 containers up and running all the time, exposing the individual models via their individual APIs. Obviously, there is a considerable cost of having this number of containers running. It is roughly equal to having 1000 virtual machines running in a cloud. This is not efficient because in production the models are often used either on an ad-hoc basis or on a regular schedule. Forecast calculation time is typically in the milliseconds to seconds range. The rest of the time the container is running but idle.

TIM's architecture is different. There is a single API (i.e. a single end-point) that facilitates all possible types of requests (training, forecasting, etc.). This API is always up and running and in our 1000-model example the same API would be used to calculate forecasts of all the models. In the background, TIM runs just few workers that can facilitate all possible types of TIM requests by interpreting any model previously built with TIM. The number of workers automatically scales with the number of incoming requests. Even with the ad-hoc timing of forecasting requests for the 1000-model scenario, TIM would likely use just a few workers (i.e. 3 to 5 on average) to serve all the asynchronous forecasting requests. The gap between the cost and efficiency of TIM's *single API* solution versus "*one model one API*" architecture is tremendous. An easy calculation of monthly cost of having 3-5 containers up and running in a cloud service versus 1000 containers running reveals how big the gap can be.

Furthermore, implementing a single API is likely to be easier and less error-prone than managing 1000 end points, which could even represent different modelling libraries.

4.3 Anomaly Detection Included

TIM API includes anomaly detection for which the user taps into the same architectural benefits as in the forecasting scenarios. TIM's architecture is lean on cloud resources but scales on demand.

TIM offers univariate anomaly detection and multivariate anomaly detection with a defined KPI. Anomalies are rare events. TIM uses its model-building capabilities to build a normal behaviour learner out of the historical data. Any departures from this normality are then detected. Multi-variate anomaly detection without a defined KPI is planned in our roadmap for 2020.

The usage flow is therefore identical to forecasting. When training, historical data are sent to the API and the model is returned. When detecting/forecasting, models and data are sent to the API and the detection/forecast is sent back. In the case of RTInstantML detection/forecasting, historical data up to the most recent timestamp are sent to the API and the detection/forecast is produced directly.

5 Business Benefits

5.1 Unification vs. Using Several Solutions

The core of TIM is designed for generic time-series forecasting. In the energy industry, TIM covers most sub-domains, i.e. electricity load, gas consumption, district heating and cooling, solar production, wind production, price forecasting, and others using the default pre-built dictionaries. Traditionally, specialized solutions and specialized models were developed to address these diverse forecasting challenges. It is often the case that within a company there are several solutions from different vendors in use to cover e.g. electricity load forecasting, solar and wind, and price forecasting. TIM unifies this traditionally siloed view. Figure 6 provides coverage examples of forecasting and anomaly detection problems in selected industries.

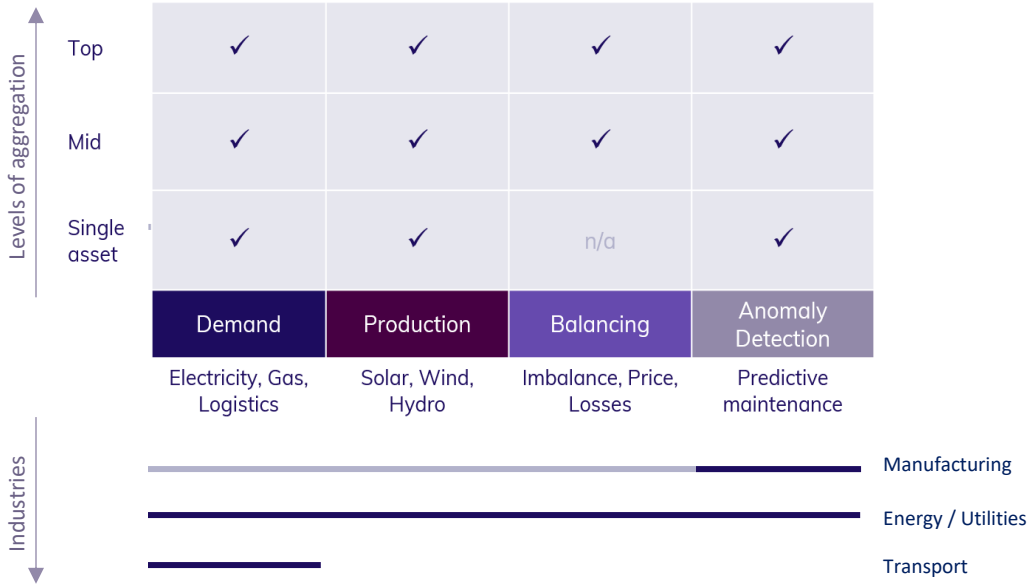

Figure 6: Examples of coverage using TIM.

5.2 Unification vs. Using Several Models for Different Prediction Horizons

Even for a single quantity in the same sub-domain, such as the electricity load of a major city, there is often a need to develop several models for different prediction horizons and different situations. Let's take an example of electricity load forecasting of a major city where every day at 10:00 and at 15:00 an intra-day forecast, a day-ahead forecast, D+2, D+3 to D+7, and month-ahead forecasts are required. Traditionally, an individual model for each of these prediction horizons is created for the two situations at 10:00 and at 15:00. Even if we consider intra-day, day-ahead, and D+2 horizons only (not considering the individual points within these horizons) and the two situations throughout the day, we end up with 6 models already (3 horizons * 2 situations = 6 models). TIM's multi-situational multi-point layer addresses this by creating a single ModelZOO for a dataset and then dispatches an appropriate model for each point of each situation in production when forecasting. This automates otherwise tedious multi-point multi-situational forecasting to the full extent using a single API call (in an InstantML or RTInstantML flavour).

5.3 Unification: Forecasting and Anomaly Detection in one API

Forecasting and anomaly detection have traditionally been viewed as two different application areas. Many software solutions therefore address either forecasting or anomaly detection. Combination of the two is rare. From a mathematical point of view, however, forecasting and anomaly detection have many aspects in common. TIM leverages this fact and provides both functionalities in a single solution via a single API.

5.4 TIM as a Data Science Tool

TIM is designed to identify features in data quickly. This also makes it an efficient data science exploratory tool. Where AutoML searches for a modelling technique, TIM's focus is the identification of features. An engineer, even without a mathematical background, may carry out experiments on different datasets quickly and get actionable insights about which explanatory variables and features matter. This is different to AutoML where a result "only" indicates which modelling technique and hyper-parameters produce the lowest error out of all the tested modelling techniques. It is often the case that adding an appropriate feature results in an enormous increase in performance.

Advanced users may configure open dictionaries of pre-defined features or implement their own and let TIM search through them. This robust approach allows a data scientist to address

numerous problems with a single configuration. For instance, the entire energy industry (see Figure 6) is addressed with a default set of feature dictionaries.

TIM, augmented with its companion TIM Studio, provides an end-to-end ML DevOps platform designed for continuous time-series model development and operations with InstantML capabilities.

6 FAQ

Q: Is TIM a model? If, so is it a neural network, support vector machine, or random forest?

A: No, TIM is a model building strategy that identifies significant features and generates a unique GAM for a dataset of interest. TIM optimizes model structure and model parameters at the same time. Therefore, models built by TIM for two different datasets may be substantially different.

Q: How does TIM build a model?

A: TIM expands original explanatory variables into many different features and then reduces them into a parsimonious GAM model.

Q: How is this different to AutoML functionalities in other solutions?

A: TIM is a modelling strategy that identifies features (InstantML) rather than a particular modelling technique and its hyper-parameters (AutoML). Features identified by TIM are by default assembled into a GAM but can be used as an input for other modelling techniques, even an AutoML pipleline.

Q: Is TIM a machine learning library?

A: No, TIM is an end-to-end automated model building and forecasting solution for time-series consisting of state-of-the-art automatic model building for time-series, including automatic feature engineering, and multiple model dispatch for different prediction horizons and data availability situations. With its companion TIM Studio, one gets an end-to-end time-series ML DevOps platform.

7 About Authors

Ján Dolinský, received the M.S. degree from the Technical University of Košice, Košice, Slovakia,in 2004 with Prof. Peter Sinčák being his supervisor. He was granted honorary scholarship of the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan, in 2005 and received the degree of Doctor of Engineering from the Kyushu University, Fukuoka, Japan, in 2009 where he studied under supervision of Prof. Hideyuki Takagi. In the same year he was awarded Japan Society for the Promotion of Science (JSPS) research fellowship and he worked with Prof. Sadanori Konishi on topics of statistical modelling, time-series, and information criteria.

From 2004 to 2005, he was a System Engineer at the Oracle Corporation, Prague, Czech Republic. From 2011 to 2013 he worked as data analyst at ESET Corp, Bratislava, Slovakia.

In 2013 he co-founded Tangent Works with a mission to provide automatic model building engine for time-series. He is the author of award-winning Tangent Information Modeller (TIM).

His research interest covers Clifford algebra, information criteria and information geometry, automatic model building for time-series, probabilistic forecasting, and nonlinear estimation. In software engineering his interest includes Julia language, SIMD, BLAS, syntactic loop fusion, distributed map reduce, databases for time-series, and cloud computing (Azure Container Instances, Kubernetes, scalable computing architectures).

Robert Tóth, is one of the early members of the Tangent Works team and a significant contributor to TIM research and development.

In his younger years, he participated in many mathematical competitions and helped raise young Slovak mathematical talents as part of his extracurricular activities. He received his master's degree from Comenius University, Bratislava, Slovakia, in 2015 from financial mathematics. During his studies, he started to incline more towards machine learning and data science and shortly after he joined Tangent Works he continued to work on various ML problems. As the company’s direction moved to time-series problems, he did so as well while competing in several forecasting competitions like GEFCom, HEPCO, and M3. His knowledge gained over the years helped shape TIM to its current form.

His research interest therefore lies mostly in time-series related topics, but he also enjoys programming in Julia and problem solving which stayed from his younger days.