Computers and Chemical Engineering 138 (2020) 106834

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

Combining machine learning and process engineering physics towards enhanced accuracy and explainability of data-driven models☆

Timur Bikmukhametov*, Johannes Jäschke

Dept. of Chemical Engineering, Norwegian University of Science and Technology (NTNU), Trondheim NO-7491, Norway

ARTICLE INFO

Article history:
Received 20 December 2019
Revised 21 March 2020
Accepted 25 March 2020
Available online 19 April 2020

Keywords:
Machine learning
Explainable machine learning
Hybrid modeling
First principles modeling
Process engineering
Virtual flow metering

ABSTRACT

Machine learning models are often considered as black-box solutions which is one of the main reasons why they are still not widely used in operation of process engineering systems. One approach to overcome this problem is to combine machine learning with first principles models of a process engineering system. In this work, we investigate different methods of combining machine learning with first principles and test them on a case study of multiphase flowrate estimation in a petroleum production system. However, the methods can be applied to any process engineering system. The results show that by adding physics-based models to machine learning, it is possible not only to improve the performance of the purely black-box machine learning models, but also to make them more transparent and interpretable. We also propose a step-by-step procedure for selecting a method for combining physics and machine learning depending on the process engineering system conditions.

© 2020 The Authors. Published by Elsevier Ltd.
This is an open access article under the CC BY license. (http://creativecommons.org/licenses/by/4.0/)

1. Introduction

Over the past several years, machine learning risen popularity due to advances in the development of deep neural networks and computational power of hardware which allows training of such networks with millions of parameters. Major advances of machine learning applications and research are mainly related to computer vision and natural language processing, and, in some cases, the performance of the developed algorithms reaches or even exceeds the human performance (Liu et al., 2019). The success of deep learning also attracted attention of the process industry towards various machine learning models (Qin and Chiang, 2019; Shang and You, 2019) where historically models are constructed based on the first principles of physics resulting in mass, momentum and energy balances written for a system under consideration. These models are usually "transparent" meaning that the change of the model input leads to the expected change of model output because the relations between the input and output are written explicitly. However, the construction cost of these models may be high, especially if the process system is large and has a complex non-linear behavior. Also, deep understanding of the system is required to create an accurate model with a correct physical behavior. In addition, these models are often required to be tuned to the process under consideration to make accurate predictions of the estimated variable (Matzopoulos, 2011).

In contrast to first principles models, machine learning models in process engineering systems estimate variables directly from data by exploiting the ability of finding complex patterns without providing an explicit form of it. This makes machine learning models easier to construct in comparison to first principles models. However, in addition to data requirements, a major drawback of these models is their black-box nature, and making machine learning algorithms transparent is currently an active field of research (Roscher et al., 2019). In the recent review of opportunities of machine learning for process data analytics, Qin and Chiang (2019) emphasized that in order to make machine learning algorithms widely applicable for process systems, we need to incorporate first principles knowledge into machine learning algorithms, consider uncertainties and produce interpretable solutions.

In this paper, we address the issue of explainability of machine learning models through combining them with first principles models. In addition, we show how combining machine learning models with first principles can improve their accuracy. To demonstrate the performance of the proposed methods, we use a

☆ The authors gratefully acknowledge the financial support from the center for research-based innovation SUBPRO, which is financed by the Research Council of Norway, major industry partners, and NTNU. Particularly, the authors acknowledge Equinor for providing field data for this study as well as its employees Audun Faanes, Tor Kjelby and Torstein Kristoffersen for assisting with getting the data quickly.

* Corresponding author.
E-mail addresses: timur.bikmukhametov@ntu.no (T. Bikmukhametov), johannes.jaschke@ntnu.no (J. Jäschke).

https://doi.org/10.1016/j.compchemeng.2020.106834
0098-1354/© 2020 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license. (http://creativecommons.org/licenses/by/4.0/)

multiphase flow estimation problem in oil and gas production systems. This case is well-suited for this purpose because multiphase flow is a complex phenomenon which is usually modeled as using first principles but can also be approached using machine learning models for some applications. In the literature, most attempts for combining first principles and machine learning models for process engineering systems in general, and oil and gas applications in particular, lead to estimation of first principles models parameters using machine learning algorithms. The history of such methods is relatively long. For instance, Psichogios and Ungar (1992) used a neural network to estimate unmeasured cell growth rate in a bioreactor process back in 90ʹs. For the oil and gas applications, a similar approach has been applied many times to estimate properties of an oil reservoir and multiphase flows in pipes such as permeability, porosity, rock type, reservoir fluid properties and liquid hold-up (Ahmadi and Chen, 2018; Anifowose et al., 2017; Klyuchnikov et al., 2019; Onwuchekwa et al., 2018; Kanin et al., 2019).

In this work, we do not use machine learning for estimation of first principles model parameters. Instead, we propose several other approaches to combine machine learning algorithms and first principles models. These approaches are much less investigated in the literature and there is a need to address this important issue. One example of such an approach, where, in addition to parameter estimation using a machine learning model, an approach of parallel combination of first principles models and machine learning model for a industrial hydrocracking unit was discussed by Bhutani et al. (2006). In this work, however, we enhance this approach proposing its several formulations and testing them at different system behavior and provide guidelines on which selecting the proper formulation depending on the system conditions.

As such, the main contributions of this paper are the following:

1. We propose new and enhance some of the previously reported methods of combining first principles and machine learning models and discuss advantages and disadvantages of each method (Section 2). In addition, we propose heuristic step-by-step guidelines on selecting a method for combining physics and machine learning for specific system conditions and desired accuracy (Section 6).
2. We investigate how physics-aware machine learning algorithms improve accuracy and explainability over pure data-driven estimation approaches via qualitative modeling of physical phenomena and model-agnostic feature analysis. In our case study, the analysis also shows how physics-aware machine learning models become able to reveal complex pattern behavior of the system, for instance, transient multiphase flow behavior in wells, gas condensation and release of gas from a hydrocarbon mixture (Section 5). However, we do not create a formal mathematical framework to get fully explainable machine learning models. Instead, via case study we make one step towards more transparent machine learning models by combining formal interpretability approach, machine learning and physics via a complex case study.
3. We show how consistent tuning using Bayesian optimization techniques contributes to consistent comparison of structurally different combinations of first principles and machine learning models (Sections 2.3.1 and 5).

In general, the problem of creating fully explainable machine learning is difficult, because it requires a rigorous definition of explainability, which is an ill-defined problem, because what is obvious to one person may be difficult to understand to another. Therefore, we adopt an engineering approach here in this paper, where we present different approaches for combining machine learning and simplified first principles models, and then analyze the results in terms of accuracy and feature influence. This feature importance is then analyzed and interpreted with physical systems understanding. In our case, the feature importance analysis give us new insights into the system, that are not obvious when setting up the model, as discussed in Section 5.1.2.

We believe that the results in this paper are a contribution towards the overall goal of better machine learning models. The overview of modelling approaches and the heuristic guidelines can be useful for practitioners. At the same time, it can be valuable for the research community to further test and develop methods and analyze the approaches.

The case study used in this work considers oil and gas production from a well which is a part of a petroleum production system. A typical petroleum production system consists of several main parts: a reservoir, production wells, flowlines, a processing facility, injection wells and transportation pipelines (Fig. 1). In the majority of cases, oil and gas is extracted from a reservoir in a form of a mixture and not as a single phase fluid. Often, in addition to oil and gas, formation water is present as another mixture phase. This mixture goes through production wells and flowlines to a processing facility where the phases are separated and processed. Then, water can be re-injected into the reservoir to enhance oil recovery or disposed to the environment if sufficiently cleaned. A part of the gas may also be re-injected and the remaining gas is transported together with oil for either further processing or usage as raw substances (Falcone et al., 2001). In case of an offshore platform, oil is typically transported by tankers.

Having good estimates of the produced volume of each phase (oil, gas and water) for each well allows to efficiently perform production optimization and reservoir management (Falcone et al., 2001). For instance, the reservoir model can be updated, so that water and gas re-injection strategies can be adjusted to increase overall oil production. In addition, insights about flow assurance issues can be obtained such as hydrate formation, erosion and (severe) slugging (Falcone et al., 2001; Patel et al., 2014).

A simple method to obtain the multiphase flowrates is to use test separators at the processing facility, shown in Fig. 1, by performing well testing. Here, single phase flowrates are estimated using a separator and flow meters at the separator outlets. By changing openings of a production choke of the well of interest and recording the changes of the total phase flowrate at the separator outlet, it becomes possible to estimate the rates of the well (Idso et al., 2014). This method typically produces quite accurate estimates of the flow. However, it has high operational expenses due to the production loss during the tests and requires high expertise from operating engineers. As such, such tests cannot be performed very often and usually the rates between the test are assumed to be constant which is generally not the case in practice. In addition, manipulating the production choke can often affect operation of surrounding wells which can lead to the drift of the operating points of these wells (Idso et al., 2014).

An alternative estimation method is the use of multiphase flow meters (MPFM) - hardware installations capable of estimating the flow in real time without separating the phases (Falcone et al., 2009). However, despite the ability to estimate the flow in real time, these devices are expensive and exposed to failure which introduces costly interventions, especially in offshore and subsea fields. In addition, MPFMs need to be re-calibrated by well testing from time to time due to changes of fluid properties (Falcone et al., 2001; Patel et al., 2014).

Another alternative is to create a mathematical model of the production system and estimate the flowrates by combining this model and readily available field measurements such as pressure, temperature and choke opening. This approach is called Virtual Flow Metering (VFM) and shown as a flow metering alternative in Fig. 1. A Virtual Flow Meter can work as a standalone metering solution or as a back-up system for a multiphase flow

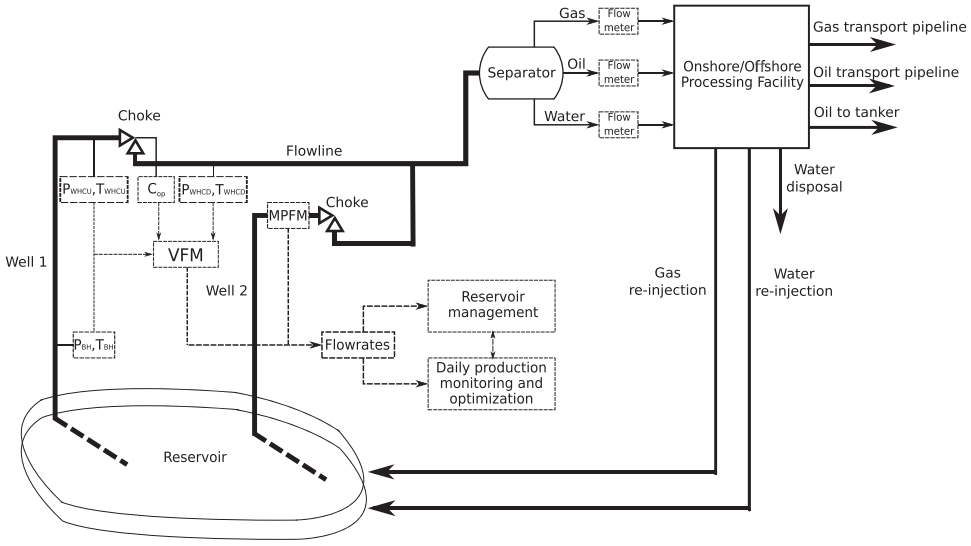


Fig. 1. Schematic representation of a typical production system with a multiphase flow meter (MPFM) and a Virtual Flow Meter (VFM) together with available measurements and typical production and processing stages. In the measurements, P denotes the pressure, T- the temperature, BH - bottomhole, WHCU - wellhead choke upstream, WHCD - wellhead choke downstream, C_{op} - the choke opening.

Table 1
Recent research work on machine learning applications for Virtual Flow Metering.

Applied algorithm	Short Summary	Reference
MLP neural network	Estimated liquid and gas rates using a neural networkensemble trained with scaled conjugate gradient and Bayesianregulation. Explored usage of meta-learners for the output layer.	AL-Qutami et al. (2018)
	Estimated gas rate with Gaussian radial basis function neuralnetwork to speed-up learning keeping good accuracy.	AL-Qutami et al. (2017)
	Estimated gas and liquid rates using multi-rate welltest data. Compared neural network with random forest.	Zangl et al. (2014)
LSTM	Estimated dynamic oil, gas and water rates for productionand severe slugging cases based on synthetic data. UsedLSTM to forecast the rates into the future.	Andrianov (2018)
	Compared LSTM with MLP neural network at steady state anddynamic conditions and showed advantages of using LSTM.	Shoeibi Omrani et al. (2018)
	Used LSTM as a model in the Kalman filter to correct biasprediction of a vanilla LSTM.	Loh et al. (2018)
	Used LSTM for multiphase flowrate estimation of transientunconventional wells data.	Sun et al. (2018)
Gradient boosting	Used gradient boosting to create a VFM as a back-up system fora MPFM. Showed how to combine the algorithm with welltest to replace MPFM. Performed sensitivity studies withrespect to the dataset size and validation methods.	Bikmukhametov and Jäschke (2019b)

meter (Lunde et al., 2013). There are two main alternatives for VFM modeling: first principles models and machine learning models (Bikmukhametov and Jäschke, 2019a). In case of first principles models, the mathematical model is constructed using laws of physics which describe the production system such as mass, momentum and energy conservation equations of the multiphase flow mixture though pipes and production choke valves. This approach requires deep domain knowledge, and the accurate models are often difficult to construct and solve numerically. In addition, due to the embedded optimization problem for parameter tuning, first principles VFM systems can perform very slowly for large production systems. However, such models are transparent and provide good overview of the processes in the production systems and the results can further be used to describe system behavior at different conditions.

In case of a machine learning model, as discussed above, deep understanding of the system behavior is not required and often the flow is estimated directly from data with no or little pre-processing. Different algorithms have been applied for machine learning Virtual Flow Metering including feed-forward neural network, gradient boosting and Long-Short-Term-Memory (LSTM) neural network. The most recent and relevant publications are summarized in Table 1. A comprehensive summary of the available literature and aspects of first principles and machine learning VFM systems is provided by Bikmukhametov and Jäschke (2019a).

Despite a reasonable accuracy of machine learning in VFM reported by the authors listed in Table 1, machine learning VFM systems, as other data-driven process engineering systems, typically provide a black-box solution which is hard to interpret. In fact, in all the mentioned works in Table 1, the authors introduce raw measurements directly into machine learning algorithms without implementing knowledge of multiphase flow physics. This is one of the main reasons why machine learning VFM systems are still rarely used in practice (Bikmukhametov and Jäschke, 2019a). To overcome this problem, we will investigate how different combinations of machine learning algorithms with first principles may improve accuracy and explainability of data-driven Virtual Flow Metering models.

This paper is organized as follows. In Section 2, we start with descriptions of the proposed methods, continue with machine learning algorithms description and the procedure for its tuning. In Section 3, we describe the system which is modeled by the methods investigated in this work as well as show how exactly these methods are adopted to Virtual Flow Metering. In Section 4, we describe case studies selected for investigation. In Section 5, we discuss the obtained results using the proposed methods. In Section 6, we summarize the most important points from Section 5 and propose a step-by-step procedure for selecting a method to combine first principles and machine learning depending on different conditions. Finally, in Section 7, we make conclusions from our work.

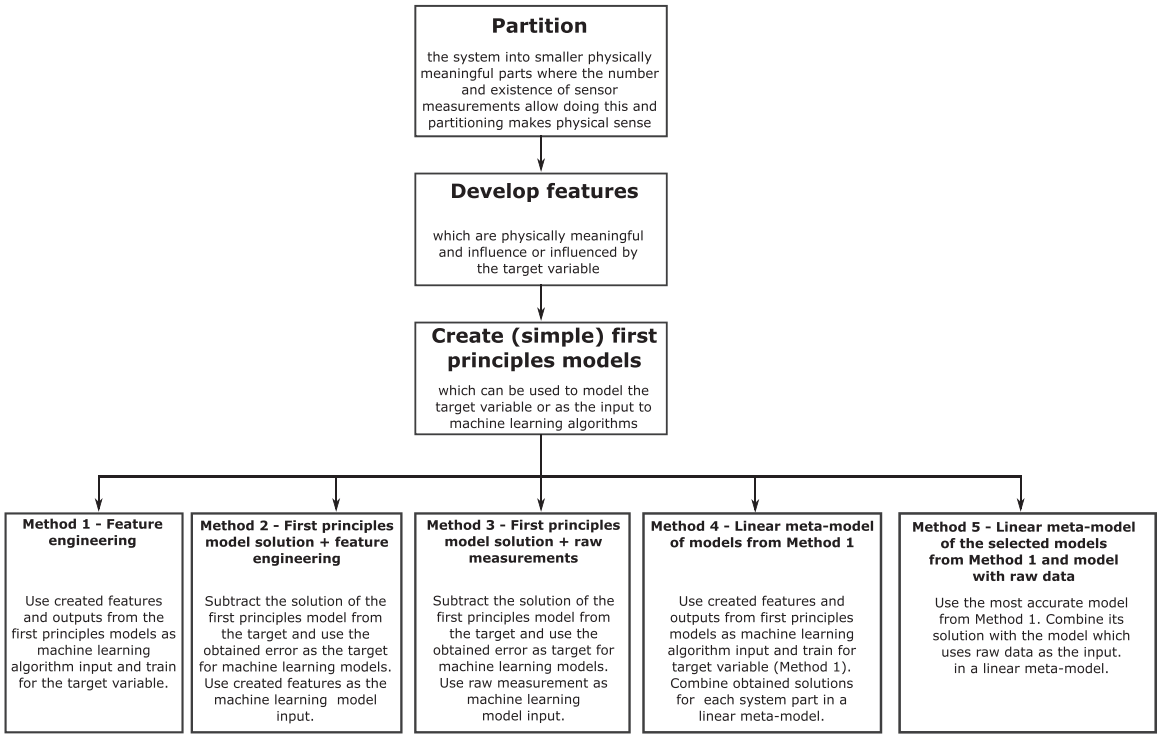


Fig. 2. Overview of the proposed combinations of first principles and machine learning models and main steps of their development for any process engineering system.

2. Methods

In this section, we describe the proposed methods for combining machine learning with first principles models and the machine learning algorithms which are used to implement the methods. We also discuss which parameters of the algorithms are tuned as well as the tuning procedure which is implemented using Bayesian Optimization approach.

2.1. Proposed methods for combining first principles and machine learning

2.1.1. Introduction to the proposed methods

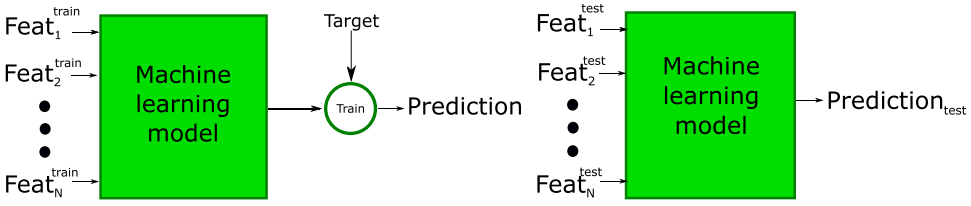
Up to now, in machine learning Virtual Flow Metering solutions reported in the literature, the measurements of pressures and temperatures available in the system have been used directly as input features without advanced transformations (Zangl et al., 2014; AL-Qutami et al., 2017; 2018). Apart from that, in some cases, complex structures have been used in order to create a model, for instance, simulated annealing combined with different types of learners within one algorithm (AL-Qutami et al., 2018). This makes the results from the obtained model even more unexplainable than a plain neural network.

To contribute to the improved explainability and accuracy of machine learning models for process engineering systems in general and petroleum production systems in particular, we propose and test several different methods. The short summary of the methods is shown in Fig. 2.

First, we propose to partition a big system under consideration into small physically meaningful parts. In this context, it means that the part may represent a part of the system, process or equipment whose parameters influence the target variable. Please note, that this action makes sense when it is viable physically and the number and existence of sensors allow doing this. Then, physically meaningful features should be developed which may include combinations of the original raw measurements alone or with external data which can be generated based on the process knowledge. In addition, simple first principles models may be constructed which are able to estimate the target variable at least qualitatively. In this work we do not assume that the first principles models can be build easily for the system under consideration. In many cases it is difficult or impossible to do due to the complexity. However, we do assume that the first principles models can be simplified to the level of abstraction which is enough for the machine learning model to understand the qualitative behavior of the system, when combined with the simplified first principles model. Then, the obtained features and models are used in several proposed methods. In Section 2.1.2, we describe each method in more detail, but in general, all of proposed methods give an opportunity to evaluate each model separately, which makes it easier to interpret the simulation result and the model behavior. In addition to the explainability advantages, it helps to create more accurate solutions than using raw data directly. This is because the main task for a machine learning model is to reveal how to combine the given features using the model parameters, for instance, weights of a neural network, to accurately estimate the target. However, if we directly introduce the way how original raw measurements should be combined based on known physics of a process, we reduce the feature space and possibly create features which contain more information about the target variable. Below, we provide the description of each method.

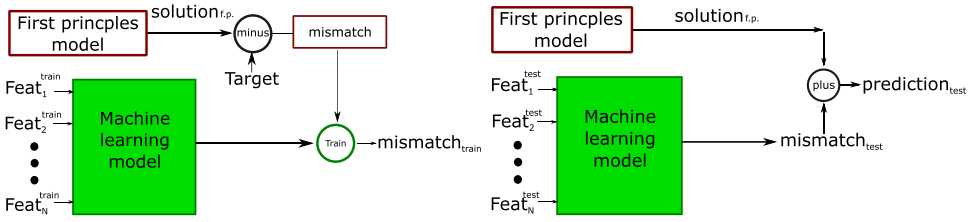
2.1.2. Detailed description of the proposed methods

Method 1 - Feature engineering. Feature engineering is a well-known method for constricting machine learning models, so, in this work, we propose the guidelines on how this method may be applied to get accurate and explainable results in physical process systems. The proposed approach includes creating physically meaningful features instead of using raw measurements directly. By physically meaningful features we mean combinations of the original raw measurements alone or with external data which can be generated based on the process knowledge. Preferably, the designed features should be well-interpretable, self-explanatory and related to a particular system part. At the same time, the fea-



(a) Training phase of Method 1 (b) Test phase of Method 1

Fig. 3. Training and test procedures for Method 1 - feature engineering.



(a) Training phase of Method 2 (b) Test phase of Method 2

Fig. 4. Training and test procedures for Method 2 - first principles model solutions and feature engineering.

tures should not necessarily be complex, but must contain some information about the target variable. The form and parameters included in the features will be case dependent and rely on expert domain knowledge. The general schematic representation of the training and test phases for feature engineering method is shown in Fig. 3.

Physically meaningful features may be further divided into simple features and complex first principles features. Simple features are the ones which are linear or non-linear combinations of the raw measurements. The complex first principles features may be, for instance, features represented as a solution of the equation which aims to model the target variable using the raw measurements and external data extracted using the process knowledge. In Section 3.3, we provide examples of features of the oil and gas production system.

Method 2 - First principles model solutions and feature engineering. In this method, we use solutions from the developed first principles models for each part of the process system. The obtained solutions are then subtracted from the true value of the target variable, such that we obtain the mismatch between the developed model and the actual value of the target. Then, this error is used as a target variable for a machine learning model. In this method, we again use created features as the input to the machine learning model to train the model to cover the mismatch.

The hypothesis here is that the obtained solution from the first principles model does not explain the system behavior, so that the created features are still well-correlated with the target variable (mismatch). As a result, by using them as the input machine learning features, we aim to learn the residual pattern of the system behavior. Fig. 4 summarizes the details of Method 2.

Method 3 - First principles model solution and raw measurements. This method is similar to Method 2. The major difference is that we use raw measurements from the system as the input to the machine learning models with the aim to explain the mismatch between the first principles model solution and the true target value.

The hypothesis in this method is that the obtained mismatch between the first principles model solution and the target does not contain the information which can be described by simple or complex first engineered features. For instance, it can be the case when some hidden patterns/disturbances exist in the system and is not covered by the proposed physical relationships implemented. If so, this method might work better than Method 2, in which we use the first principles features as the input to the algorithm aiming to cover the mismatch. Fig. 5 shows the details of Method 3.

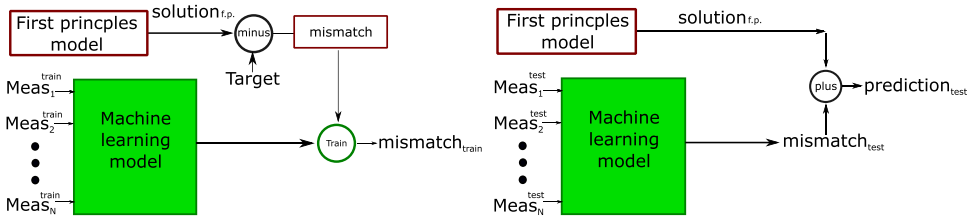
Method 4 - Linear meta-model of models with created features. In this method, we combine solutions from models created in Method 1 using a linear meta-model. The idea here is to give a weight to the prediction from each model of a particular system part, for instance, choke and tubing, and then sum the weighted predictions to get the final outcome. The weights for each sub-model are tuned using linear regression techniques, such that the weighted sum of the sub-models accurately describe the system.

The hypothesis here is that at certain conditions a particular model can produce better results than another model. If so, the output from this model multiplied by the associated weight will have a closer value to the target than another model. By combining the solutions from several models, we can take the advantage each model and obtain the solution which is more accurate then the solution from a single model. The schematic representation of the training and test phases for Method 4 is shown in Fig. 6.

Method 5 - Linear meta-model of the selected model with created features and model with raw data. In this method, we combine the solution of any of the models obtained using Method 1, i.e. the model with the created features, with the model which uses the raw data as features.

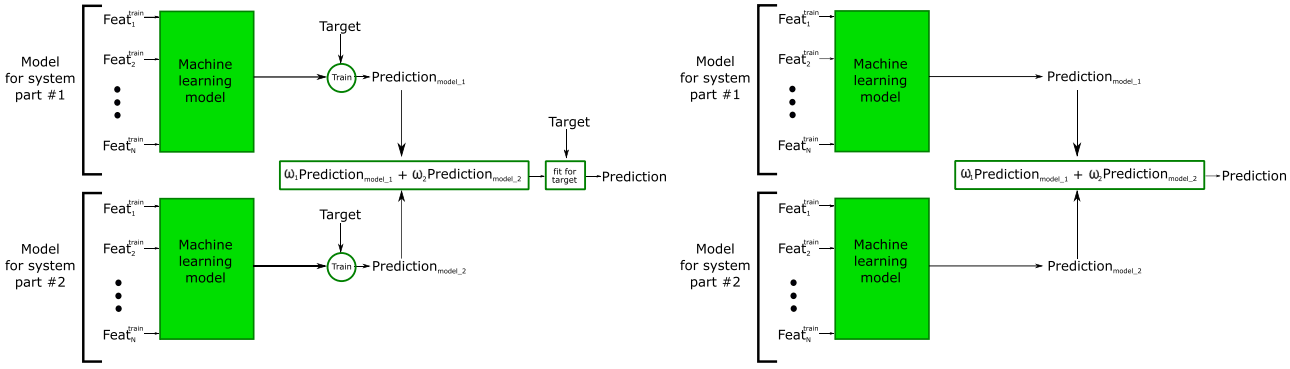
The hypothesis here is that even by obtaining the best model using the created features, there is still some unrevealed data structure which cannot be described by the model while the raw data model can do it, at least partially. As such, the combined solution will be more accurate than both the best feature engineered model and the model with raw input data. Fig. 7 shows the details of Method 5.

In some cases, one may argue that one specific model will be more accurate and another model will make the overall prediction worse. However, we do not give these weights just by assigning them, they are learned from the data which means, on average, the predictions from such a model will most likely be better on the new test set. This idea is similar to a normal linear regression case. For a particular point in space one value of weight can be better because then the line will go exactly thorough one particular point. However, this mostly likely not be beneficial for other points. As such, the proposed method is simply linear regression, but instead of features, we are giving model values, which on av-



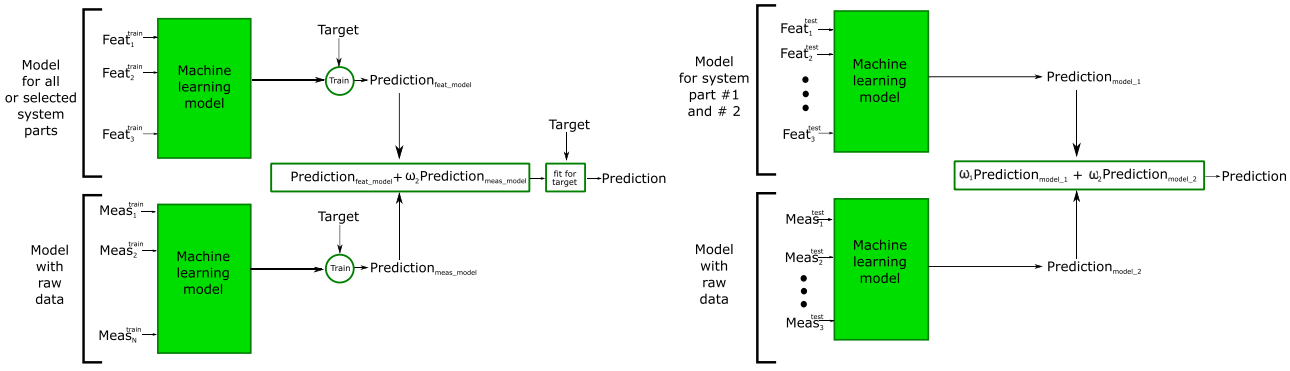
(a) Training phase of Method 3 (b) Test phase of Method 3

Fig. 5. Training and test procedures for Method 3 - first principles model solutions and raw measurements.



(a) Training phase of Method 4 (b) Test phase of Method 4

Fig. 6. Training and test procedures for Method 4 - linear meta-model of models with created features.



(a) Training phase of Method 5 (b) Test phase of Method 5

Fig. 7. Training and test procedures for Method 5 - linear meta-model of the selected model with created features and model with raw data.

erage will produce more accurate predictions on the test set. In an ideal case, one would want to have adaptive weights which know that for this particular point it needs to use this particular model is better and we just need to choose that model. However, this is another direction of research and will not be covered in our paper.

2.2. Applied machine learning algorithms

In this section, we provide a brief overview of the machine learning algorithms used in this paper, namely Gradient boosting, Multilayer Perceptron and Long-Short Term Memory Neural Networks. More specifically, we describe the selected hyperparameters and its tuning and the role in the model training and predictive capabilities.

2.2.1. Gradient boosting and tuned parameters

The first algorithm used for machine learning VFM in this work is gradient boosting with regression trees implemented in XGBoost package (Chen and Guestrin, 2016). Gradient boosting is based on sequential construction of shallow regression trees which form an ensemble and perform the final value estimate as a sum of the trees' predictions (Friedman, 2001). In XGBoost implementation, the algorithm minimizes the following objective function in an iterative manner adding a regression tree at each iteration:

$$J = \sum_{i=1}^{N} (y_i - [\hat{y}_i^{(t-1)} + f_t(x_i)])^2 + \Omega(f_t) \tag{1}$$

with

$$\Omega(f_t) = \gamma Z + \frac{1}{2}\lambda_{GB}\|w\|_2^2 \tag{2}$$

where y_i denotes the true value of the $i-th$ example, $i-$ the index of a training example, $t-$ the number of the currently constructed regression tree, $N-$ the number of examples in the training set, $\hat{y}_i^{(t-1)}-$ the sum of the $t-1$ trees, f_t- the prediction of the current tree on the $i-th$ example, $\Omega(f_t)-$ the regularization term, $\gamma-$ the penalty term for the model complexity expressed as the number of leaves Z, $\lambda_{GB}-$ the penalty term of the weight values w.

Table 2
Hyperparameter space and other parameters used for machine learning algorithms training.

Algorithm	Hyperparameter	Range (oil rate)	Range (gas rate)	Other parameters/concepts	Approach
Gradient boosting	maximum tree depth	[3:12]	[3:9]	Splitting algorithm	Greedy linear search
	regularization λ	[0.001:0.1]	[0.0001:0.1]	Maximum number of trees	200
	regularization γ	[0.001:0.1]	[0.0001:0.1]	Framework	XGBoost
	minimum child weight	[1:3]	[1:4]		
MLP neural network	learning rate	[0.0001:0.001]	[0.0001:0.001]	Activation function	ReLU
	regularization	[0.00005:0.005]	[0.00005:0.05]	Optimization algorithm	Adam
	number of layers	[1:5]	[1:5]	Maximum number of epochs	1000
	number of nodes	[5:25]	[10:30]	Framework	Tensorflow
LSTM	learning rate	[0.0001:0.01]	[0.0001:0.01]	Activation function	tanh
	window size	[1:15]	[1:15]	Optimization algorithm	Adam
	dropout rate	[0.05:0.2]	[0.05:0.2]	Maximum number of epochs	3
	number of LSTM units	[10:40]	[10:40]	Framework	Keras
	number of LSTM layers	[1:3]	[1:3]		

The algorithm minimizes the difference between the sum of all the previous predictions and the true value by adding a new tree. To manipulate the training of gradient boosting, we tune the maximum depth of regression trees which is responsible for the complexity of the function which algorithm is able to approximate. In addition, we tune regularizing coefficients γ and λ_{GB} to prevent overfitting as well as the "minimum child weight" parameter which identifies the minimum number of instances required in each node to become a split. The larger the "minimum child weight", the more conservative the algorithm becomes (Chen and Guestrin, 2016). By adjusting these parameters, we try to find a good trade-off between the bias and variance of the model and strengthen generalizing capabilities on unseen data. Table 2 summarizes the hyperparameters and its ranges used for tuning.

2.2.2. MLP Neural network and tuned parameters

Feed-forward (Multilayer Perceptron (MLP)) neural networks are one of the most popular types of artificial neural networks in machine learning (Goodfellow et al., 2016). They are constructed using neurons which are interconnected with weights and stacked in layers. The weights are used in order to fit the algorithm to the data which is typically done via backpropagation algorithm and gradient-based optimization method (Rumelhart et al., 1988) which usually minimizes the following objective function:

$$J = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{3}$$

where i denotes the index of a training example, $N-$ the number of training examples in the training set, $\hat{y}_i-$ the estimated value of the $i-th$ example, y_i- the true value of the $i-th$ example.

A feed-forward neural network is a universal approximator (Hornik et al., 1989) which means that it can approximate any function given sufficient structural complexity. To approximate nonlinear functions, neural networks use an activation function in each neuron. Among other alternatives, ReLU activation function is a popular choice for feed-forward neural network regression (Agarap, 2018) because it does not suffer from exploding and vanishing gradients, so we also use it in this work. Despite the advantage of strong approximating abilities, a neural network can easily overfit the training data which will create high variance on unseen data. As such, often the objective function in Eq. 3 is adjusted with a regularization term:

$$J = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 + \lambda_{NN}\|w\|_2^2 \tag{4}$$

where λ_{NN} denotes the regularization coefficient, $\|w\|_2^2-$ the l_2-norm of neural network weights.

In this work, we use the objective function as shown in Eq. (4) and consider the regularization coefficient λ_{NN} as a hyperparameter to be tuned. As in the case with gradient boosting, by tuning the regularization coefficient, we aim at finding a good trade-off between the bias and variance of the model and avoid overfitting to the data noise.

To minimize the cost function J in Eq. (4), we use the "Adam" gradient based optimization algorithm (Kingma and Ba, 2014). Despite the fact that the learning rate is proposed to be adaptive in Kingma and Ba (2014), in this work, we use it as a tuning hyperparameter to find a good initial guess of the learning rate for the algorithms. In addition, we also try to find the best architecture of the network, so we also tune the number of layers and neurons simultaneously with the learning rate and regularization coefficient. Table 2 summarizes hyperparameters of neural networks and its ranges used for tuning.

2.2.3. LSTM Neural network and tuned parameters

Long-Short-Term-Memory is a special type of recurrent artificial neural networks which is designed for sequential type of data such as speech and text (Hochreiter and Schmidhuber, 1997). In terms of regression for Virtual Flow Metering, the sequence of data is a series of measurements in time. So, by using an LSTM for modeling of the time dependent target, we use past measurements of pressure and temperature in order to predict the flow at the current time step. As such, one of the goals in using LSTM in this study is to evaluate how the sliding window approach which considers the past data can help in improving the current time step estimates. In previous works on using LSTM for VFM, this important parameter has not been investigated (Loh et al., 2018; Sun et al., 2018).

There are different possible configurations of LSTM such as many(inputs)-to-many(outputs), many-to-one or one-to-many. In this work, we use many-to-one configuration, such that several input units (measurements back in time) are used in order to estimate the oil and gas flowrates respectively, at the current time step. As such, one of the hyperparameters of the network is the time window that is the number of time steps in the past which are used for the current estimate. The number of time steps is then equal to the number of cells in a LSTM layer.

An LSTM neural network may consist of several layers, which is also the case for this work, and the layers have many-to-many connections. We tune the number of LSTM layers because it directly influences the possible complexity of the function which the network is able to approximate.

Greff et al. (2016) showed that, in addition to the number of layers and cells in the LSTM layer, it is also important to tune the learning rate in LSTM networks to achieve good performance, so we include it as a tuning hyperparameter. To regular-

ize LSTM and prevent overfitting, in the extensive LSTM tuning by Greff et al. (2016) it was proposed to use input noise instead of using ridge regularization as we do in the feed-forward (MLP) neural network case (Eq. 4). However, to experiment with more tuning parameters and regularization techniques, we use dropout (Cheng et al., 2017) on the weights which connect the last LSTM layer and one dense layer which computes the final estimates of the flowrate. The range of the LSTM hyperparameters and other parameters are shown in Table 2.

2.3. Hyperparameter tuning and comparison procedure of case studies

To be successfully applied to any data, machine learning algorithms have to be well tuned. By this, we mean that a good algorithm architecture has to be selected as well as the values of these hyperparameters have to be tuned accurately. This is typically one of the most difficult and critical parts of machine learning modeling. We need to find a good "bias-variance trade-off" (Hastie et al., 2005) meaning that the model that has small bias and variance on the training set must have comparable bias and variance on the test set, so it has good generalizing capabilities.

In general, it can be difficult to compare different algorithms in a fair manner, because it is possible to give one algorithm (or method) an advantage by making a more accurate tuning than for another one if desired. In our case, in addition to have 3 algorithms to compare, we also have 12 cases for each algorithm (Section 4). To produce accurate and trustworthy results, we developed a pipeline for training and evaluation the methods which is designed not to give an advantage to any case or algorithm. This pipeline uses Bayesian optimization for algorithms tuning as a core and searches the hyperparameters over the same space for all the cases for the same machine learning algorithm. In the next sections, we present the main concept behind Bayesian optimization and the developed tuning pipeline.

2.3.1. Bayesian optimization

Bayesian optimization is a suitable concept for optimizing a function which is computationally expensive to evaluate (Frazier, 2018). Typically, this function does not have an analytical expression and can be a black-box which gives output values for a given input. The idea behind Bayesian optimization is to create a surrogate model of the objective function with corresponding uncertainties using Gaussian Processes regression. Gaussian Processes is a Bayesian machine learning method that is why this optimization approach is called Bayesian. Using this approach, the algorithm decides where to evaluate the function next, given the observed objective function values and corresponding uncertainties (Frazier, 2018). This problem is known as the "exploration-exploitation" trade-off (Berger-Tal et al., 2014).

To describe Bayesian optimization in more detail, let us define the function to be optimized as $f(x)$. Here, $x \in \mathrm{IR}^{\mathrm{d}}$ is the function input where d is the dimension of the input space. In case of hyperparameter optimization for a machine learning algorithm, $f(x)$ can be an error function on a validation set and d is the number of hyperparameters, for instance, $d = 2$ if we tune the learning rate and regularization coefficient. Let us further assume that we have already evaluated the objective function at $x_{1:n}$, where n is the number of evaluated points. Our goal is to find such x^*, so that it gives the maximum (in case of hyperparameter optimization - minimum) of the function at x given $x_{1:n}$, i.e.:

$$x^* = arg\max_x(-f(x|x_{1:n})) \tag{5}$$

By solving Eq. (5), we find the set of hyperparameters which minimizes the error on the validation set. However, the only way to evaluate $f(x)$ is to give input x to the black-box function and obtain output, so that we cannot simply evaluate gradients for finding descent direction of the function.

To solve this problem, we use the concept of Gaussian Processes regression. First, we specify a normal prior distribution (typically with zero mean) over the entire parameter space $\mathrm{IR}^{\mathrm{N}\times\mathrm{d}}$, such that:

$$f(x_{1:N}) \sim \mathcal{N}(0, K(x_{1:N}, x_{1:N})) \tag{6}$$

where $K(x_{1:N}, x_{1:N})$ denotes the covariance between the points $x_{1:N}$ and $N-$ the number of parameter values.

Then, given the observed values $f(x_{1:n})$, we compute the mean and variance of the posterior distribution of the function $f(x)$ using the following expression for Gaussian Processes regression (Rasmussen and Williams, 2017):

$$\mu_n(x) = K(x, x_{1:n})K(x_{1:n}, x_{1:n})^{-1} f(x_{1:n}) \tag{7}$$

$$\sigma_n(x)^2 = K(x, x) - K(x, x_{1:n})K(x_{1:n}, x_{1:n})^{-1}K(x_{1:n}, x) \tag{8}$$

If at this point we were required to provide the best set of hyperparameters, we would say that this is x^* which gives $f_n^* = max(f(x_{1:n}))$. However, if we are allowed to take one more sample anywhere in the hyperparameter space, we can check if we can find a better set. After the new sample, the highest value of the function can be $f(x)$ in case $f(x) \geq f_n^*$ or it is going to be still f_n^*. As such, the possible improvement of the function is $f(x) - f_n^*$. Therefore, we want to make a new evaluation at x which produces the largest improvement. However, the challenge is that we do not know the value $f(x)$ until we take a sample of it. In this case, the solution is the fact that we can take the *expected value of this improvement* (Frazier, 2018):

$$EI_n(x) = E_n[f(x) - f_n^*] \tag{9}$$

where $EI_n(x)$ denotes the expectation of the posterior distribution given the values of $f(x_{1:n})$.

Finally, we take a new sample where $EI_n(x)$:

$$x_{n+1} = arg\max_x EI_n(x) \tag{10}$$

The sampling stops when the specified number of sampling steps is reached. This optimizing strategy (acquisition function) is called Expected Improvement and this is what we use in this work to find an optimal set of hyperparameters for the machine learning algorithms. Further details about how to compute the closed form of the acquisition function in Eq. (9) are provided by Frazier (2018).

2.3.2. Tuning pipeline

To tune the algorithms, first, we need to split the data into training, validation and test parts. Since we have time dependency in the data, we perform k-fold cross-validation with mixed folds. Since we perform an extensive hyperparameter optimization search for many cases, we cannot use nested k-fold cross-validation because it is computationally too costly. As such, we split the data into 60% for training, 25% for validation and 15% for test sets and use only one set to validate the algorithms.

Another challenge is to choose how to perform fine tuning of epochs and trees together with other hyperparameters. The procedure we propose is to fix a certain number of epochs/trees which corresponds to a relatively deep training. Then, for this fixed number, we select hyperparameters using Bayesian optimization based on the validation set error. Then, using the obtained set of hyperparameters for each algorithm, we perform fine tuning of the number of epochs/trees using early stopping by monitoring the error on the validation set. The same hyperparameter space is used for all the cases within one algorithm.

Having the algorithm ready, we re-train it on the combined training and validation sets by taking advantage of all the data

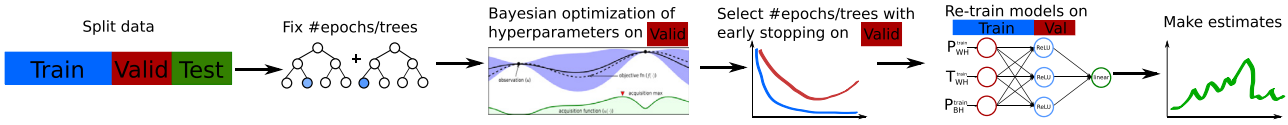


Fig. 8. Proposed pipeline for algorithms tuning and case comparison.

available for training and finally evaluate it on the test set. A summary of the proposed tuning pipeline is shown in Fig. 8. Table 2 shows hyperparameters and their range used in the tuning pipeline.

Note that the procedure above is designed such that we:

- Do not give any preference to any particular case and any particular algorithm;
- Select a good set of hyperparameters via extensive Bayesian optimization search;
- Avoid overfitting via early stopping and regularization;
- Allow another possibility for overfitting control by changing the hyperparameter space boundaries.

2.4. Feature analysis methods

Machine learning algorithms are often treated as black-box solutions which are hard to interpret. In this work, we investigate if features which are based on physics principles can contribute to a better understanding of machine learning models behavior.

There are several methods which can be used to interpret machine learning models. A good review of these methods is provided by Molnar et al. (2018). For non-linear models, the methods can be divided into model-specific and model-agnostic methods. For instance, tree-based algorithms such as gradient boosting and random forest have embedded model-specific methods which evaluate feature importance based on the selected criteria (Chen and Guestrin, 2016; Genuer et al., 2010). However, sometimes, different criteria produce different feature importance for the same algorithm, which can make results confusing.

As we compare different algorithms in this work and also aim to avoid misleading conclusions within each algorithm, we use model-agnostic approach for evaluation of feature importance. This means that such methods are applicable to any machine learning model and use the same principles independently of the algorithm under evaluation. This will allow us to compare different algorithm using the same principles, so the conclusions can be well generalized. We use the feature importance and partial dependence plots implemented in Skater Python library (Kramer et al., 2018). In this package, feature importance evaluation is based on the theoretic information criterion which estimates the entropy of the prediction change supplied by a perturbation of a feature. The partial dependence plots are adopted from Hastie et al. (2005), and describe the global influence of a particular feature given that other features are kept constant. As such, both methods are global interpretation methods, so that they analyze the influence of features over the entire dataset, rather than explaining the local behavior as, for instance, LIME method does (Ribeiro et al., 2016). We use global interpretation because, in this work, we are interested in the overall relationships between the features and the targets. However, local interpretations may also be useful for Virtual Flow Metering, for instance, when the flow pattern is changed and we are interested in the analysis of the conditions at which it happens. We keep these local investigations for future work.

We applied the methods discussed above to gradient boosting and feed-forward neural networks, but have not applied to the LSTM networks due to its dependency on a particular feature in time. Such analysis has not been implemented yet neither in the package nor by us, so we also keep this analysis for future work too.

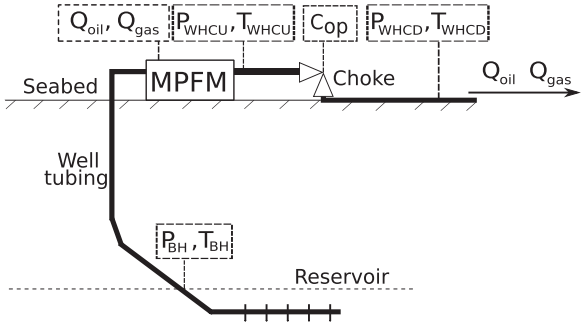


Fig. 9. Schematic representation of the investigated system with available measurements.

3. Description of the system and data

3.1. System and data

3.1.1. Overview

To test the proposed approaches, we use field data from one of the subsea fields on the Norwegian Continental Shelf. The system and the available measurements are shown in Fig. 9. For the input to the algorithms, the following data is available:

- Pressure and temperature at the bottomhole of the well (P_{BH}, T_{BH});
- Pressure and temperature upstream of the choke (P_{WHCU}, T_{WHCU});
- Pressure and temperature downstream of the choke (P_{WHCD}, T_{WHCD});
- Choke opening (C_{op});
- Well tubing length (L_{tubing});
- Well tubing diameter (D_{tubing});
- Fluid composition.

In this system, a multiphase flowmeter (MPFM) is installed at the wellhead which measures oil and gas flowrates, Q_{oil} and Q_{gas} respectively. The measurements of pressures, temperatures, choke opening and flowrates are available at every minute. The length and diameter of the tubing are fixed by the system design and do not change during operation. The data available for training is the historic production for 210 days and shown in Fig. 10. As we see in Fig. 10, the system has a very unstable behavior during the entire production time. The flowrates of oil and gas fluctuate a lot which makes difficult it for a machine learning algorithm to estimate it accurately.

We will construct machine learning algorithms based on the flowrate measurements from the meter, so that the machine learning model will work as a back-up system. If the multiphase flowmeter fails or starts producing unrealistic results, the VFM solution will infer the current flowrates. As such, the target variables for training are (Fig. 10):

- Oil flowrate (Q_{oil});
- Gas flowrate (Q_{gas}).

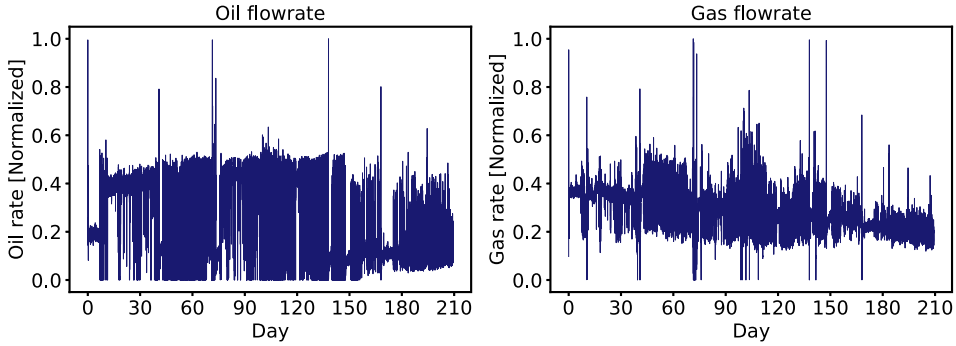


Fig. 10. Normalized oil and gas flowrates from the subsea well under consideration.

In oil and gas production operation, the fluid composition is not always available. In the dataset used in this work, the fluid composition is given, however, this information is assumed to be very unreliable, because it comes from measurements that were taken a long time ago, and the composition may have changed over time. As such, the fluid properties estimated using this composition can be misleading and give a noticeable estimation error. Therefore, we would like to avoid using estimates of fluid properties such as density or viscosity as much as we can in the first principles models, while maintaining the qualitative physical behavior of the created models.

In practice, there may be a possibility to re-estimate the fluid properties given a new Gas-Oil-Ratio (GOR) by performing a new well test, however, this information is not available in our setting. However, it is useful and interesting to investigate predictive capabilities of physics-aware machine learning algorithms under these realistic conditions.

3.2. Applied first principles models

In this section, we introduce the first principles models which are used in combinations with machine learning algorithms using the proposed methods discussed in Section 2.1. To create a combined Virtual Flow Metering systems with machine learning and physics of the multiphase flow in petroleum systems, we propose to use two first principles models: Bernoulli model for choke and No-Pressure-Wave drift flux model for tubing.

3.2.1. Choke model

The Bernoulli equation is often used to describe single phase flow in hydraulic systems and it is the basis for the simplest model used to describe fluid flow over a choke. The equation describes the fluid momentum between two points:

$$P_1 - P_2 = \frac{\rho}{2}(U_2^2 - U_1^2) \tag{11}$$

where P_1 denotes the pressure at point 1, P_2 - the pressure at point 2, ρ - the fluid density, U_1 - the fluid velocity at point 1, U_2 - the fluid velocity at point 2.

Typically, a choke is modeled as a sudden contraction which is called choke throat. In this case, Eq. (11) is applied between the point before the throat (point 1) and at the throat (point 2). Considering the fact that the volumetric flowrate $Q = A \cdot U$, we obtain:

$$Q = C_d A_2 \sqrt{\frac{2(P_1 - P_2)}{\rho\left(1 - \left(\frac{A_2}{A_1}\right)^2\right)}} \tag{12}$$

where C_d denotes the discharge coefficient, A_1– the pipe cross sectional area before the choke, A_2–the area of the choke throat.

The discharge coefficient C_d is typically a function of the choke opening C_{op}, and used to tune the model to the data at hand. Because measuring pressure at the choke throat is difficult, pressure measurement after (downstream) the choke is often considered as P_2. As such, according to the introduced notation in Figs. 1 and 9, $P_1 = P_{WHCU}$ and $P_2 = P_{WHCD}$.

Eq. (12) is valid for single phase flow. To apply this to a multiphase flow case, the single phase flowrate is usually multiplied by a two-phase multiplier, for instance, the Chisholm multiplier (Chisholm, 1983) which depends on the fluid properties at measured conditions. Because the fluid properties may change over time, and a two-phase multiplier as well as mixture density may be inaccurate in this case, we will not introduce two-phase multiplier into the choke model and will use the model outputs as computed below.

In addition, the mixture density ρ_{mix} must be introduced instead of single phase density ρ. The ratio A_2/A_1 can be considered as the choke opening (C_{op}) because A_1 is the constant area of the pipe and A_2 is changing depending on how open the choke is. As the discharge coefficient C_d is a function of the choke opening which form we do not necessarily know, we simplify it by assuming a linear relationship. Considering this, we obtain a simple expression for the mixture volumetric flowrate across the choke Q_{mix}^{choke} which is used in this work:

$$Q_{mix}^{choke} = C_{op} A_2 \sqrt{\frac{2(P_{WHCU} - P_{WHCD})}{\rho_{mix}\left(1 - C_{op}^2\right)}} \tag{13}$$

Finally, by multiplying Q_{mix}^{choke} with volumetric fraction of a fluid phase (for instance, oil), we can obtain the phase volumetric flowrate. The remarks on how the volumetric fractions are found are discussed in Section 3.2.3.

3.2.2. Tubing model

For the tubing model, we use a "No-Pressure-Wave" form of the drift flux multiphase flow model (Masella et al., 1998) which is described by the following equation:

$$\frac{dP}{dl} = \frac{\xi_{mix}\rho_{mix}U_{mix}^2}{2D_{tubing}} + \rho_{mix} g \sin(\beta) \tag{14}$$

where P denotes the fluid pressure, l– the pipe axial coordinate, ξ_{mix}– the friction factor coefficient, U_{mix}– the mixture velocity, D_{pipe}– the tubing diameter, ρ_{mix}– the mixture density, g– the

gravitational acceleration constant, $\beta-$ the inclination angle of the pipe.

This equation is a simplification of the transient drift flux model (Masella et al., 1998). Here, it is assumed that the flow is at steady state and that the effect of acoustic waves is negligible for the considered time scale. We can solve this equation given the pressures at the bottomhole and the wellhead and the fluid properties for the mixture. The same as for the choke model, the fluid properties can be the bottleneck for accurate predictions of the flow due to it is potential inaccuracy. Moreover, the friction coefficient is also dependent on them. As such, to avoid additional source of inaccuracy, we keep the friction coefficient constant.

Typically, Eq. (14) is solved numerically together with the mass balances for each phase in each discretization mesh point along the pipe axial coordinate. In this work, we test simplified first principles modeling approaches and want to use machine learning to take care of the model inaccuracies. As such, by averaging the fluid properties and the geometry over the pipe axial direction and integrating Eq. (14), the solution for the multiphase mixture in tubing becomes:

$$Q_{mix}^{tubing} = \sqrt{\frac{(P_{BH} - P_{WH} - \bar{\rho}_{mix} g H)\pi^2 D_{tubing}^5}{8\xi_{mix} L_{tubing} \bar{\rho}_{mix}}} \tag{15}$$

with

$$\bar{\rho}_{mix} = \frac{\rho_{mix@WH} + \rho_{mix@BH}}{2} \tag{16}$$

where P_{BH} and P_{WH} denote bottomhole and wellhead pressure respectively, $\rho_{mix@WH}$ and $\rho_{mix@BH}-$ the mixture density at the wellhead and bottomhole conditions respectively, $H-$ the height (elevation) of the tubing, $L_{tubing}-$ the tubing length, $\xi_{mix}-$ the friction factor coefficient, $\beta-$ the inclination angle of the tubing.

3.2.3. Remarks on fluid properties computation for first principles models

We see that even the simplest process models require accurate measurements and fluid properties data to compute the flowrate accurately. As discussed, in our case, the fluid properties can be relatively inaccurate because the composition is provided at the beginning of production when the system was installed, while the production data is at the late stage. This is exactly the place where machine learning can enter and solve the problem with less effort. As we still need some approximations of the fluid properties, we use the given (uncertain) fluid composition and Soave-Redlich-Kwong (SRK) Equation of State (EoS) (Soave, 1972) implemented in a commercial thermodynamic package. To obtain the phase volumetric fractions and densities, we use simple flashing. We iterate over the pressure and temperature condition range met in the problem, save the results in look-up tables and then interpolate the properties for any given pressures and temperatures at any time step. This approach is commonly used in first principles simulators of multiphase flow.

3.3. Adaptation of methods for combining machine learning and first principles for VFM

In this section, we discuss how the developed first principles models and features are used within the methods proposed in Section 2.1. The overall summary of the used features for the input in each method and case study can be found in Table 3 in Section 4.

Please note, that all the developed models in this work use a single output, such that we estimate only one target variable each time. As such, we constructed separate models for oil and gas rates. From our experience, making separate models for oil and gas (i.e. for different outputs) produces better results and allows much more flexible tuning. On the other hand, neural networks also allow multiple output regression without any single problem. As for gradient boosting, in this case, two separate algorithms are always needed.

3.3.1. Adaptation of method 1 - feature engineering

As discussed, we partition the system into the tubing and choke parts and create the following features:

- Pressure drop over the choke ($\Delta P_{choke} = P_{WHCU} - P_{WHCD}$);
- Pressure drop over the tubing ($\Delta P_{tubing} = P_{BH} - P_{WHCU}$);
- Temperature drop over the choke ($\Delta T_{choke} = T_{WHCU} - T_{WHCD}$);
- Temperature drop over the tubing ($\Delta T_{tubing} = T_{BH} - T_{WHCU}$).

Despite being simple, these features have direct relationships with the target variables - the oil and gas flowrates. This is because in any process system, pressure difference is the main driving force for the flow to go from one point to another. As such, pressure drop over the well tubing and choke can be a good indicator of the flow magnitude. At the same time, the flow magnitude and disturbances is what define the temperature drop over the tubing and choke. In terms of explainability, these features make more sense than raw pressure and temperature measurements which can be hard to interpret with respect to the change of the target variable (flowrate).

As we propose in Fig. 2, we can also use outputs from the created first principles models as input features to machine learning models. As such, we use the following additional features:

- Choke mixture volumetric flow (Q_{mix}^{choke}) (Eq. (13));
- Tubing mixture volumetric flow (Q_{mix}^{tubing}) (Eq. (15)).

The reason why we use mixture volumetric flow and not the separate oil and gas volumetric flows is the fact that we know that the fluid properties are inaccurate in our case, so we want to avoid using them as much as possible. At the same time, the mixture flow can be sufficient to give the machine learning algorithms additional insights about the qualitative behavior of the multiphase flow and improve prediction accuracy. Fig. 11 summarizes the adaptation of training and testing procedures shown in Fig. 3 for Virtual Flow Metering.

3.3.2. Adaptation of method 2 - first principles model solutions and feature engineering

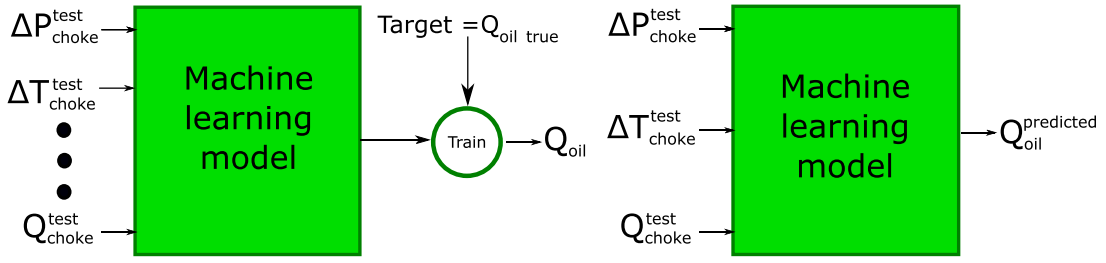
In this method, we use solutions of Eqs. (12) and Eq. (15) transformed to the phase volumetric flowrates via multiplying it by the phase volumetric fractions pre-computed in a thermodynamic package. The obtained solutions are then subtracted from the true phase rate, such that we obtain the mismatch between the developed model and the actual flow measurements from the field. Then, this error is used as a target variable for a machine learning model. The procedure of training and testing the algorithms for Method 2 shown in Fig. 4 is adapted for Virtual Flow Metering in Fig. 12.

3.3.3. Adaptation of method 3 - first principles model solution and raw measurements

As discussed before, Method 3 is similar to Method 2, while the main difference is that raw measurements are used as the input to the machine learning models with the aim to cover the mismatch between the first principles model solution and the target. When adapting this method to the Virtual Flow Metering example, pressure and temperature measurements along the systems are used as the input features to the machine learning models. Fig. 13 illustrates the adaptation of the method to Virtual Flow Metering.

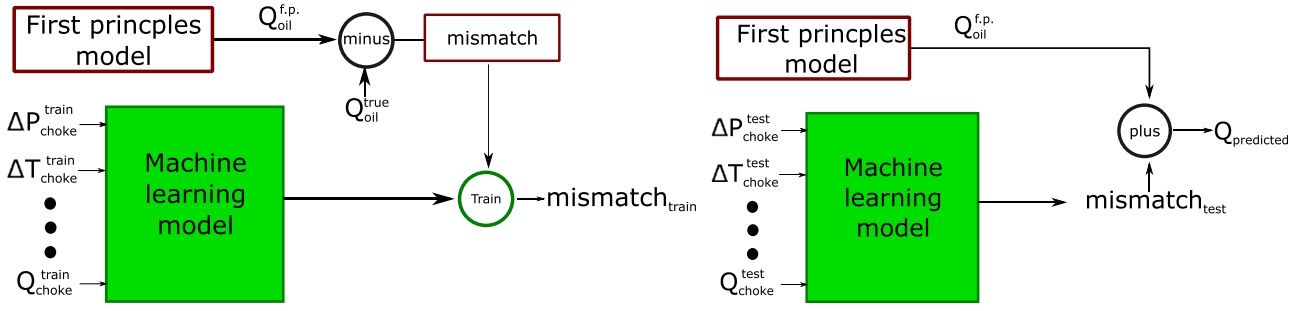
Table 3
Summary of the considered case studies.

Case study	ML algorithm input	First principles model solution	Training target
Case 1	P_{BH}, T_{BH}, P_{WHCU}, T_{WHCU}, P_{WHCD}, T_{WHCD}	not used	Q_{phase}^{true}
Case 2.1	ΔP_{choke}, ΔT_{choke}, Q_{mix}^{choke}	not used	Q_{phase}^{true}
Case 2.2	ΔP_{choke}, ΔT_{choke} Q_{mix}^{choke}	Choke model $Q_{phase}^{choke} = Q_{mix}^{choke}\alpha_{phase@WH}$	$mismatch = Q_{phase}^{true} - Q_{phase}^{choke}$
Case 2.3	P_{BH}, T_{BH}, P_{WHCU}, T_{WHCU}, P_{WHCD}, T_{WHCD}	Choke model $Q_{phase}^{choke} = Q_{mix}^{choke}\alpha_{phase@WH}$	$mismatch = Q_{phase}^{true} - Q_{phase}^{choke}$
Case 3.1	ΔP_{tubing}, ΔT_{tubing}, Q_{mix}^{tubing}	not used	Q_{phase}^{true}
Case 3.2	ΔP_{tubing}, ΔT_{tubing} Q_{mix}^{tubing}	Tubing model $Q_{phase}^{tubing} = Q_{mix}^{tubing}\alpha_{phase@WH}$	$mismatch = Q_{phase}^{true} - Q_{phase}^{tubing}$
Case 3.3	P_{BH}, T_{BH}, P_{WHCU}, T_{WHCU}, P_{WHCD}, T_{WHCD}	Tubing model $Q_{phase}^{tubing} = Q_{mix}^{tubing}\alpha_{phase@WH}$	$mismatch = Q_{phase}^{true} - Q_{phase}^{tubing}$
Case 4.1	ΔP_{choke}, ΔT_{choke}, Q_{mix}^{choke} ΔP_{tubing}, ΔT_{tubing}, Q_{mix}^{tubing}	not used not used	Q_{phase}^{true}
Case 4.2	ΔP_{choke}, ΔT_{choke}, Q_{mix}^{choke} ΔP_{tubing}, ΔT_{tubing}, Q_{mix}^{tubing}	Choke and tubing model $\tilde{Q}_{phase}^{choke/tubing} = (Q_{ophase}^{choke} + Q_{phase}^{tubing})/2$	$mismatch = Q_{phase}^{true} - \tilde{Q}_{phase}^{choke/tubing}$
Case 4.3	P_{BH}, T_{BH}, P_{WHCU}, T_{WHCU}, P_{WHCD}, T_{WHCD}	Choke and tubing model $\tilde{Q}_{phase}^{choke/tubing} = (Q_{phase}^{choke} + Q_{phase}^{tubing})/2$	$mismatch = Q_{phase}^{true} - \tilde{Q}_{phase}^{choke/tubing}$
Case 5	Q_{phase}^{choke}, Q_{phase}^{tubing}	not used	Q_{phase}^{true}
Case 6	Q_{phase}^{choke} or Q_{phase}^{tubing} or $Q_{phase}^{choke/tubing}$ and $Q_{phase}^{Case\,1}$	not used	Q_{phase}^{true}



(a) Training phase of Method 1 in VFM systems (b) Test phase of Method 1 in VFM systems

Fig. 11. Method 1 (feature engineering) adapted for Virtual Flow Metering.



(a) Training phase of Method 2 (b) Test phase of Method 2

Fig. 12. Method 2 (first principles model (choke and tubing) solutions and feature engineering) adapted for Virtual Flow Metering.

3.3.4. Adaptation of method 4 - linear meta-model of models with created features

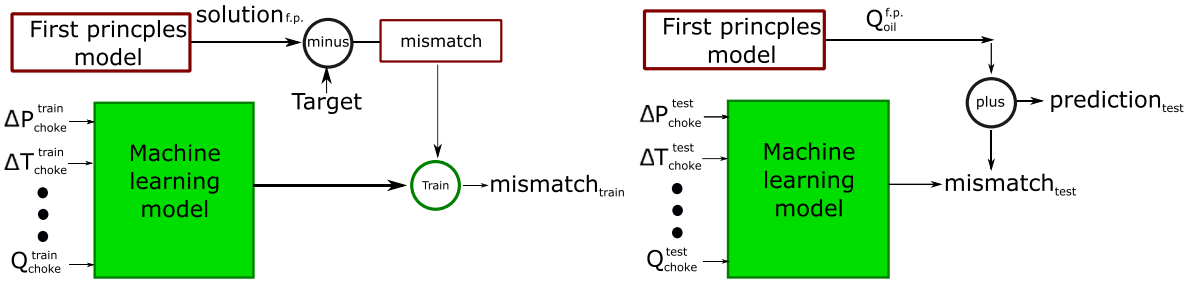
In this method, we combine solutions from models created in Method 1 for the choke and tubing system parts and then sum the weighted predictions to get the final outcome. Fig. 14 shows the adaptation of the method to Virtual Flow Metering.

3.3.5. Adaptation of method 5 - Linear meta-model of the selected model with created features and model with raw data

In this model, any of the models (choke or tubing or choke and tubing) with created features can be used together with the model which uses raw measurements as the input. In this work, we use the model with both choke and tubing features. Fig. 15 shows how the method is applied for Virtual Flow Metering systems.

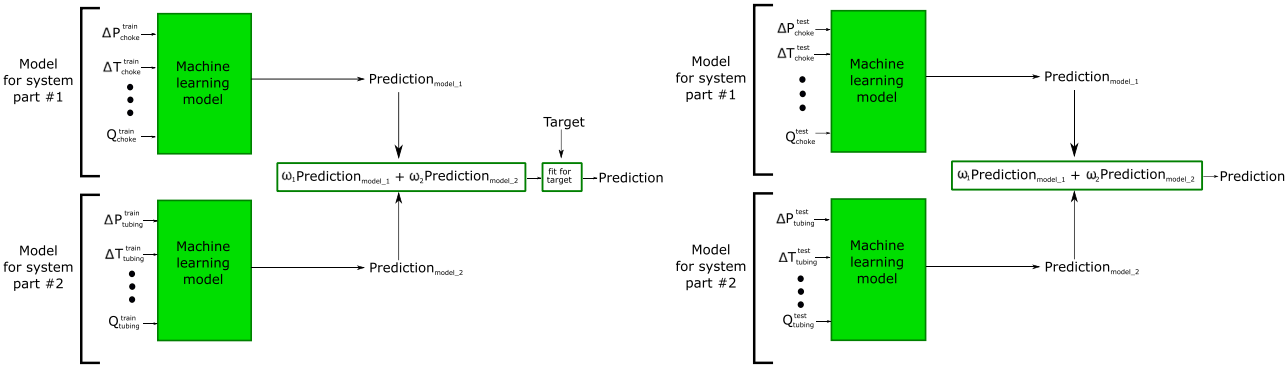
4. Case studies

This section provides an overview of the considered case studies. The summary of the selected case studies, input features, used first principles models and training targets is shown in Table 3.



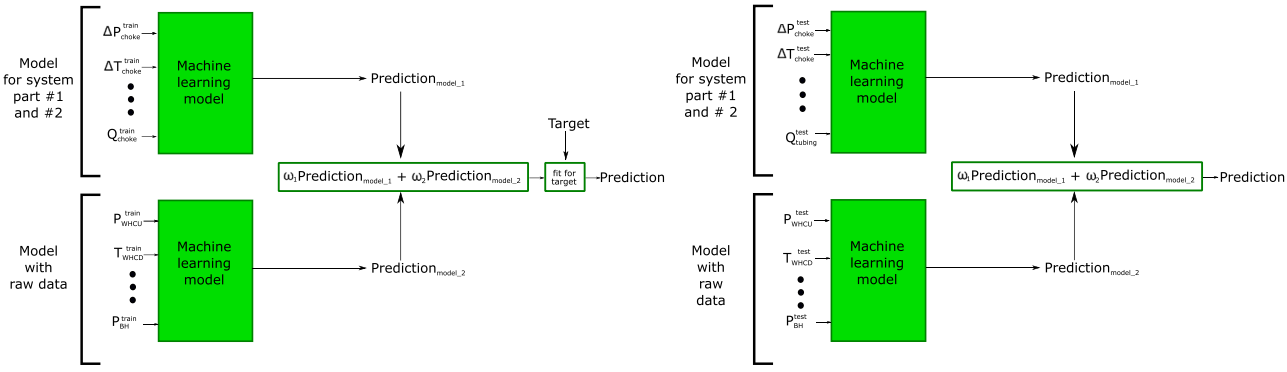
(a) Training phase of Method 3 (b) Test phase of Method 3

Fig. 13. Method 3 (first principles model (choke and tubing) solution and raw measurements) adapted for Virtual Flow Metering.



(a) Training phase of Method 4 (b) Test phase of Method 4

Fig. 14. Training and test procedures Method 4 - linear meta-model of models with created features.



(a) Training phase of Method 5 (b) Test phase of Method 5

Fig. 15. Training and test procedures Method 5 - Linear meta-model of the selected model with created features and model with raw data.

Case 1 is the base case which is used for comparison with all the other cases. In this case study, raw measurements are used as the input to the machine learning algorithms without any transformation. This is the approach which has been used in the data-driven Virtual Flow Meters in the literature so far and often used in estimation of parameters in process engineering systems using machine learning. This base case will be compared with the proposed approaches of combining machine learning with first principles models.

Case 2 considers methods 1, 2 and 3 applied to the production choke. More specifically, **Case 2.1** considers using Method 1 (feature engineering) applied for the choke, so that the inputs to the machine learning algorithms are ΔP_{choke}, ΔT_{choke} and Q_{mix}^{choke}. This case will show if the measurements related to the choke only can describe the flow accurately.

In **Case 2.2**, Method 2 (first principles model solution + feature engineering) is used such that the solution of the choke model for $Q_{oil/gas}$ is obtained and the mismatch between the solution and the true value is covered using an algorithm with choke features ΔP_{choke}, ΔT_{choke} and Q_{mix}^{choke} as inputs.

In **Case 2.3**, Method 3 (first principles model solutions + raw measurements) is used and raw measurements are used for the algorithm which is trained to cover the mismatch. Table 3 summarizes the used combinations for the cases.

Case 3 is similar to Case 2, but it considers methods 1, 2 and 3 applied to the production tubing. As such, **Cases 3.1, 3.2** and **3.3** are conceptually identical to Cases 2.1, 2.2 and 2.3, but instead of choke, tubing first principles model and features are used (Table 3).

Case 3 considers method 1, 2 and 3 applied to choke and tubing combined. More specifically, **Case 4.1** considers all the features for choke and tubing combined as the inputs to the machine learning model, so it again follows Method 1.

In **Case 4.2**, to use Method 2 we used averaged solution from the choke and tubing models for Q_{oil} and then cover the mismatch using both tubing and choke features.

In Case 4.3, we cover the mismatch by using the raw measurements as features. Again, the discussed case studies are summarized in Table 3.

Case 5 considers Method 4 - using a linear meta-model for the algorithms which are based on the engineered features. We consider this case only for the choke (Case 2.1) and tubing models (Case 3.1) combination. This is because this is the simplest yet effective way to combine all the available models and measurements.

Case 6 considers Method 5 - using a linear meta model which combines the selected model with engineered features (we selected the model from Case 4.1) with the model from Case 1 which uses the raw data as the input.

Remarks on other possible case studies. It is possible to create a meta-model over the other cases, and even a meta-model on top of other two meta-models which combine, for instance, first principles models solution and mismatches. However, it will introduce additional unexplanability of the models as well as it leads to a tricky implementation without necessarily providing a better accuracy.

5. Results and discussion

In this section, we describe the estimation results for gradient boosting, MLP neural networks and LSTM and feature analysis for gradient boosting and MLP neural networks. First, we thoroughly describe the estimation results from gradient boosting including the estimation accuracy and transparency of the data-driven models which is developed through combining first principles and machine learning models. After that, we discuss the results of neural networks and LSTM mostly focusing on the differences between the results of these algorithms and gradient boosting to avoid repetition of similar conclusions. We also discuss the differences between static algorithms (MLP neural network and gradient boosting) and dynamic algorithm (LSTM). Please note, that we do not provide a feature analysis of LSTM neural networks, because the current version of the feature analysis tool (Skater) does not provide such capabilities due to its dependency on a particular feature in time. Nevertheless, we decided to include the estimation results of the LSTM, because they provide insights about the importance of considering past time steps data in estimating the current time target value. As such, it provides the basis for conclusions on the conditions at which dynamic methods should be preferred over the static ones.

5.1. Analysis of gradient boosting results

5.1.1. Overview of flowrate estimation results

In this section, we analyze the results of oil and gas flowrate estimation using gradient boosting based models. While the original time resolution of the data is 1 minute, for better visualization, the estimation results are averaged over the period of 5 minutes. This allows to see if the algorithm is able to capture the general multiphase flow trends rather than occasional flowrate spikes which are not essential to capture. The results for oil and gas flow estimation using gradient boosting based models are shown in Fig. 16.

When to use feature engineering (Method 1) and when to use first principles-based models (Method 2 and 3)?

The first observation in Fig. 16 is that in some cases, feature engineering methods outperform other methods while in other they do not. As feature engineering method is the simplest one in terms of construction cost, it is useful to know under which conditions it can be effectively applied which will allow to avoid constructing more complex combinations of first principles and machine learning models.

First, consider the oil rate estimation case (Fig. 16a) and compare the feature engineering-based methods with other proposed methods. The accuracy of these models in comparison with the base case is discussed in the next sections.

We see that the cases with engineered features (Case 2.1, 3.1 and 4.1) generally outperform all the other models using the proposed methods (Case 2.2 and 2.3, Case 3.2 and 3.3, Case 4.2 and 4.3 respectively) except the meta-model approaches. However, in the gas flow estimation (Fig. 16b), when the first principles models are combined with the raw measurements (Case 2.3, 3.3 and 4.3), such models generally perform equally good or better than the feature engineering models.

The major difference between the oil and gas flowrates is the complexity of the system behavior. From the test set representation in Fig. 16 as well as from the entire dataset in Fig. 10, we see that the oil rate behavior is much more unstable and does not have the same trend in the train and test sets, while the gas rate has much smaller fluctuations and the trend is observable. As such, we see that when the system behavior is relatively complex (in this case - oil rate behavior) and the constructed first principles models are relatively simple as in this work, it is better to use feature engineering methods as a simple yet accurate solution. This is because for a complex system behavior, simple first principles models may not be accurate enough to accurately represent it. At the same time, well-engineered features may be well-correlated with the target variable (in this case - the oil flowrate), such that the model produces accurate results.

When the system behavior has a moderate complexity, (in this case - gas rate behavior), so that even simple first principles models are accurate enough, its combination with the raw data may be a good choice as it can be seen in Fig. 16 (Case 2.3, 3.3 and 4.3).

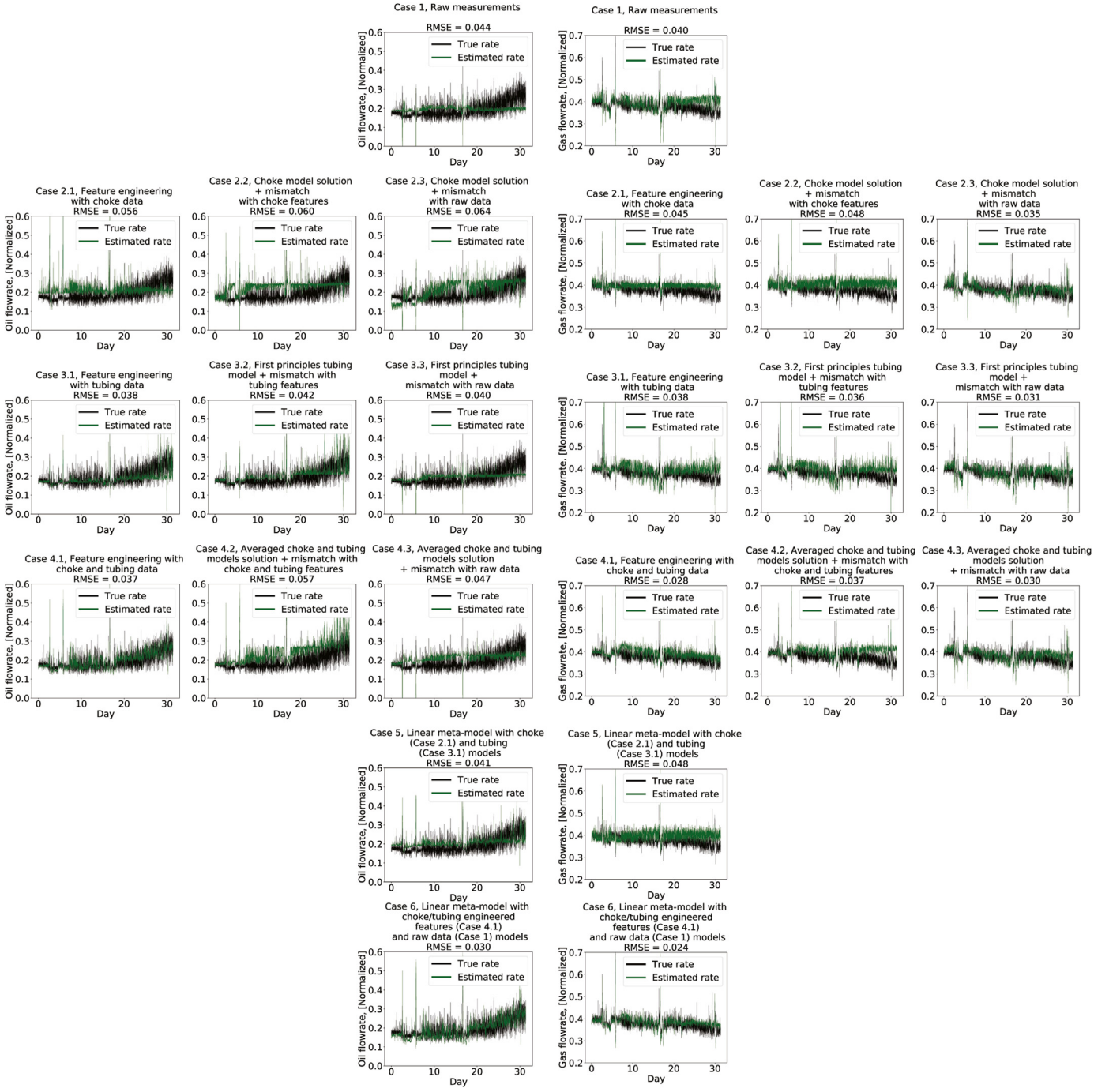
As for the Method 2, when the first principles models are used together with the simple and complex first principles features, it does not improve the performance as we see from Fig. 16 (Case 2.2, 3.2 and 4.2) for both oil and gas rates. The reason for this may be the fact that the features are better correlated with the original target variable and not the mismatch, while the first principles models solution may also not be very accurate, which in total deteriorates the performance, as we see in Fig. 16 (Case 2.2, 3.2 and 4.2), especially for the oil rate estimation.

Influence of first principles models accuracy on estimation accuracy and on capturing physical effects The next important discussion is related to the one in the section above with a particular focus on the influence of the accuracy of the first principles models on the estimation accuracy.

Comparing the choke-based (Case 2.1, 2.2 and 2.3) and tubing-based (3.1, 3.2 and 3.3) machine learning models, we see that the proposed first principle choke model is not a solid basis to accurately represent the oil rate behavior, such that it has a lower accuracy than the base case (Case 1) with just raw measurement input data. However, the choke-based machine learning models show a moderate accuracy for the gas estimation case and Case 2.3 even outperform the model in Case 1.

At the same time, the first principles tubing model and related features are relatively accurate to represent the flow well. Even though the model may not be able to account for high oil flowrate fluctuations, the general flow behavior is accurately represented and all the constructed tubing-based machine learning models outperform the model from Case 1.

As such, we conclude that in order to be applied alone for oil rate estimation within machine learning VFM, the proposed first principles choke model must be improved for this specific case. One possibility to do that is to change the model type which resolves the flow more accurately and accounts for more complex physics such as gas slip, as suggested by Schuller et al. (2006), who found that by introducing the slip relationship, a choke model typically produces more accurate estimates. Another possibility is to perform some pre-tuning of the model using simple linear regres-



(a) Predicted oil flowrates

(b) Predicted gas flowrates

Fig. 16. Oil and gas rates estimation by gradient boosting based models. The results are averaged over 5 minutes period, i.e. one point - 5 minute averaged rate.

sion techniques and then use the model as the basis for the combinations with machine learning algorithms as proposed in this work.

Despite the fact that the first principles tubing model shows a reasonable accuracy, it can also be further improved. For instance, the model can be solved numerically for a small number of mesh points along the tubing and only then combined with machine learning, so that the model will provide a more accurate solution than the one presented here. We keep these investigations for future work.

It is worth noting that despite the choke model alone is not accurate enough, it can still be used to further improve the accuracy of the tubing model results. We can see that Case 4.1 for both oil and gas rates produces one of the most accurate results, if not considering the meta-models from Case 6. This is likely because the choke and tubing models in this case can be better at estimating different flow conditions. In fact, we see that for the oil rate predictions, the choke model based algorithms (Case 2) follow a more transient behavior of the system, such that we see predictions of flowrate fluctuations with some occasional spikes which try to capture even higher fluctuations, but often overestimating them. This is different from the cases with the tubing model based machine learning models (Case 3), where we see a more smooth flow. This behavior is physically meaningful because the choke model is a lo-

cal (static) model which means that any flow disturbance across the choke will suddenly be reflected in the change of pressures and temperatures taken at the inlet and outlet of the system. On the other hand, if a flow disturbance occurs at the bottomhole of the well, this will be reflected at the bottomhole measurements first and only after some time reach the wellhead measurements. For the gas flow cases, the behavior of both models is relatively similar due to low flow fluctuations.

Improved model generalization One of the potential advantages in including physics information into machine learning models can potentially be improved model generalizability. This is the ability of the model perform well on the unseen data. In physical and process engineering systems, this can be exemplified if the machine learning model is able to describe system under conditions, which have been barely seen in the training set. In our case, the model generalization is seen to be improved significantly for several cases. indeed, in Case 4.1, we see that by combining the first principles models as features, we are able to capture the rising oil rate trend at the end of the estimation period as well as the decreasing gas rate trend, while when using the pure data-driven models (Case 1), it is not possible. As such, by combining the models, we can achieve improved machine learning model generalization and the overall improvement of the results. This strengthens the fact that the combined approach of physics-based machine learning modelling may be able to significantly improve machine learning model generalization, especially compared to a pure data-driven approach.

Meta-models performance

The next important discussion concerns the meta-models performance. While the meta-models have the highest construction cost, they may not necessarily have the highest accuracy. For instance, we see that the accuracy of Case 6 for both oil and gas rates is the highest among the cases, while Case 5 meta-models are not as accurate. The reason for this is the accuracy of the sub-models used in the meta-models. Since the accuracy of the model with choke-based features (Cases 2.1) is not high, it deteriorates the results of the meta-model. As such, we conclude that it is better to combine the models using engineered features within one algorithm (such as in Case 4.1), than in a meta-model (Case 5), if the performance of separate models is not accurate. However, if the models are accurate enough, the joint meta-model can further improve the performance, such as in Case 6.

Also, we conclude that when using the raw data models within a meta-model, we can improve the performance of the physics-based machine learning models and, at the same time, keep the overall model explainable because the weight of the raw data model in the meta-model shows its contribution to the overall solution. As such, we are able to see which part of the process is resolved by the physics-aware algorithms and which part is still unresolved and covered using the raw data algorithm. A detailed analysis is shown in Section 5.1.2.

Advantages of comparing different physics-aware machine learning models using the proposed approach

The observed results in the sections above emphasize one of the most important conclusions from this work: by analyzing and comparing the simulation results using the proposed approaches with physically meaningful features, separating the system into sub-parts and creating first principles models for each system part, it becomes possible to better understand the physical behavior of the system, make conclusions about the drawbacks of the applied models and propose solutions to create more accurate approaches. This is not the case when raw measurements are used directly. In that case, even if the solution is accurate, it is hard or impossible to comprehend if the solution is physically meaningful or not and propose future improvements.

In Table 4, we suggest possible cases which can be met by conducting *simultaneous* model analysis using Method 1 (feature engineering), Method 2 (first principles model solution + feature engineering) and Method 3 (first principles model solution + raw data). Here, we propose possible solutions to the problems which can arise during such analysis, so that the table can be used as an initial reference when the proposed methods applied to any process engineering system. For instance, if we see that feature engineering method produces low accuracy, while the combinations of the first principles models with raw data has high accuracy and adding features to such model reduces the accuracy (the first raw in the table), it is evident that in this case the features are poorly designed. As such, to further improve the accuracy, the features should be re-designed.

Of course, other situations may occur during the analysis which are not shown in Table 4, for instance, the accuracy of two methods are identically high or low, but using the logic in the table, it will be easy to comprehend the solution for any case.

5.1.2. Feature analysis

Apart from the potential of improving the accuracy of the models, first principles-based algorithms allow to check if the model follows the expected physical behavior. Also, it can help to reveal some additional patterns in the data. To explore these opportunities, we perform a feature analysis of the constructed models. Fig. 17 shows the feature importance while Fig. 18 shows the partial dependence plots for oil and gas rate (or oil and gas rate mismatch) prediction models based on the gradient boosting algorithm. As the values are standardized in model training and testing, they are removed from the partial dependencies plots because they do not add any value, while removing them allows a better visualization. The plots are used to identify the qualitative behavior of the models. It is also worth noting that the partial dependence plots are not produced for Case 5 and Case 6, because in these cases the meta-models are considered, so that the partial dependence plots will be identical to the plots of the meta-model sub-models and the importance of sub-models is shown in Fig. 17.

Poor features identification

The first interesting observation is that in the oil rate estimation case, the choke flow constructed feature has high importance when used in the separate choke model cases (Case 2.1 and 2.2), while when used with all the other features in Case 4.1, the feature has low importance. At the same time, the tubing mixture volume flow feature has high importance in both Case 3.1 and 4.1. From this observation, we conclude that the choke mixture flow feature is not representative enough to describe the flow behavior. This conclusion is supported by the rate estimation results shown in Fig. 16a where we see that the accuracy of Case 2.1 and 2.2 is low. We observe that by relying on the choke mixture flow feature, the algorithm makes poor flow estimates. At the same time, the tubing mixture flow feature is better designed and this is again confirmed by the results from Fig. 16a where the tubing based models are relatively accurate. When all the features are combined (Case 4.1 and 4.2), the gradient boosting algorithm is capable to distinguish good features (tubing model related), make the full use of relatively poor features (choke model related) and improve the estimates.

Transparency of meta-models

As for the importance of the meta-models features, Fig. 17 shows the absolute values of the meta-model weights. We see that the meta-models rely on the more accurate models (tubing and tubing/choke) which is what we would like to have. We see that in the gas estimation case, the meta-model gives higher weights to the raw data model than in the oil case. This coincides with what we saw in the rate estimation section, where adding the raw data model to the first principles model improved gas rate estimation accuracy (Case 2.3, 3.3 and 4.3) while in the

Table 4
Analysis of methods for combining first principles with machine learning and proposed problem solutions.

Accuracy				
Feature engineering	First principlesmodel solution+ mismatch withfeature engineering	First principlesmodel solution+ mismatch withraw data	Conclusion	Solution
low	average	high	Accurate first principles model,but features are poorly designed(too simple) to describe the system.	Re-design featuressuch that they describethe target more accurately
low	high	average	Accurate first principles model,features are somewhat correlated withmismatch but generally poorly designed	Re-design existing features makingthem more complex andaccurate than the created ones
average	low	high	Inaccurate first principles modeland features, raw data betterdescribe the system behavior	Entirely revise the modeland engineered features
average	high	low	Inaccurate first principles modelbut well designed features	Revise the model to furtherimprove accuracy of the solution
high	low	average	Inaccurate first principles modeland features are not correlated withthe mismatch	Revise the model, make itmore complex such that it betterdescribes the system behavior
high	average	low	Inaccurate first principles modelbut well-designed features	Revise the model, make it morecomplex such that it betterdescribes the system behavior

oil rate estimation, this does not improve the performance as significantly. As such, we can say that the constructed models which are based on the physics-based features (Case 4.1) are better in explaining the given dataset than the raw measurement models (Case 1), especially for estimation of the oil rate, because of its better performance and higher weight values in the meta-models. At the same time, there is still a potential in improving the created model accuracy which should increase importance of the physics-based machine learning model even further when combined in a linear meta-model with the raw measurement model.

From the estimation and feature importance observations regarding the meta-models, we conclude that, in addition to the improved estimation accuracy by using the linear meta-model structures, we can also evaluate the potential of improving the physics-based machine learning models itself. For instance, in Case 5-type meta-models, these improvements consider separate models for each system part, i.e. we can check if, for instance, by introducing the slip relation into the choke model, the importance of the choke model in the meta-model increases or decreases and compare with the obtained estimation results. More importantly, the same can be said for the physics-based machine learning model of the entire system used in Case 6. That is, by creating different (more complex or less complex) physics-aware models and comparing their contribution in the meta-model with the raw data models, we can check if the new proposed model reduces the influence of the raw data based model. The higher the influence of the model, the better the model is, because in such a case, the new constructed model will better explain the data and reveal the patterns which were previously unrevealed by a simpler model. Again, it is expected that the model with the higher importance will have higher estimation accuracy.

Insights about physical behavior of the system

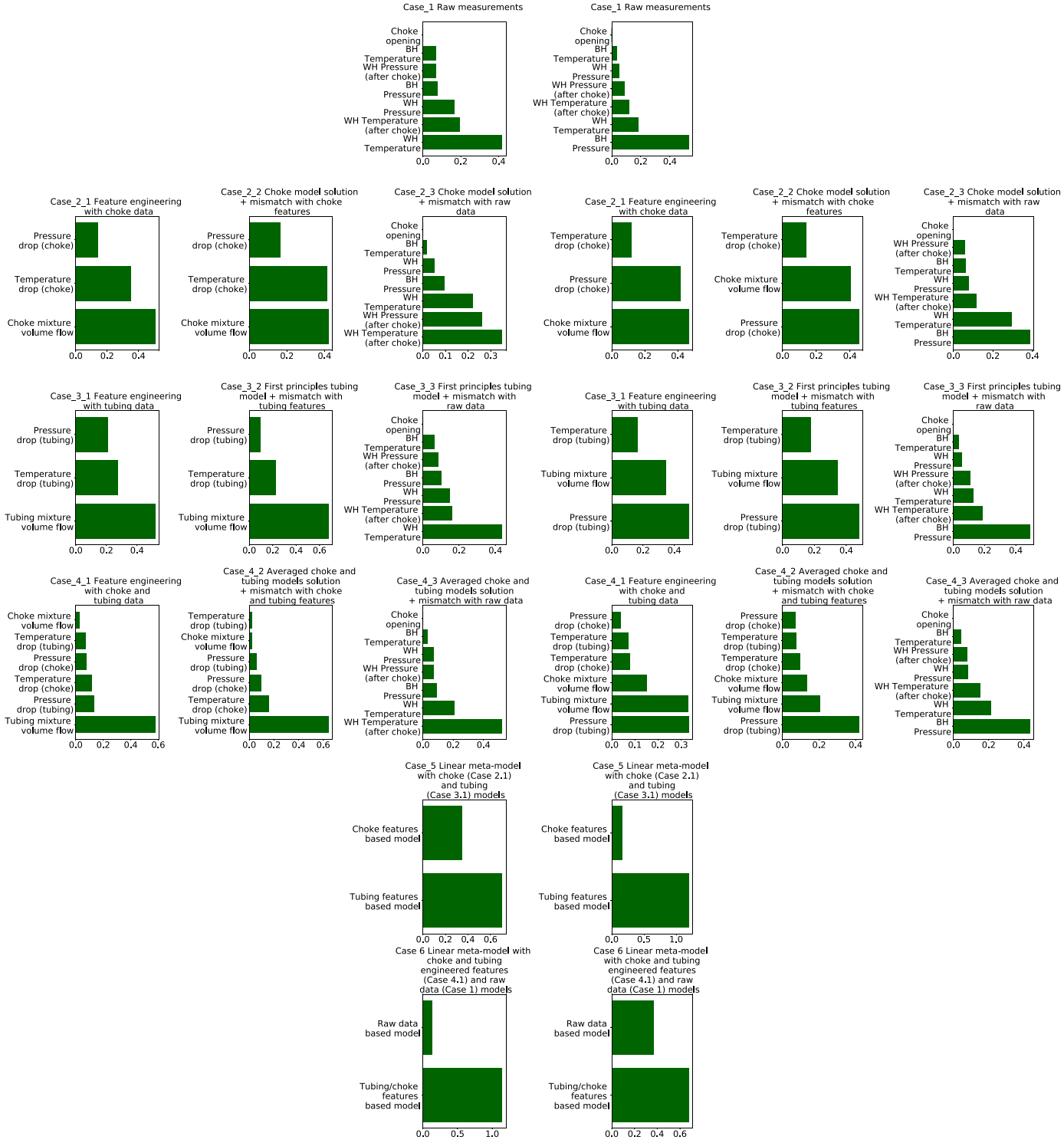
In addition to the better explainability of the machine learning algorithms, we can get extract insights about the fluid behavior in the system using the constructed feature importance and partial dependencies plots.

Temperature drop effect. Consider the partial dependence plots for Case 3.1 for both oil and gas predictions. We see that for the tubing case, the larger the temperature drop is, the higher oil flowrate is. This is the opposite to the gas case, where we see that with the rise of the temperature drop, the gas rate decreases. Such model behavior well corresponds to the thermodynamic behavior of a hydrocarbon mixture. That is, with the decrease of temperature of the mixture, more hydrocarbon mass starts being condensed from the gas phase and accumulated in the liquid phase. By considering that the reservoir temperature is relatively constant, the decrease of the temperature of the fluid will be mainly caused by the heat transfer along the well tubing. As such, more hydrocarbon will be observed in the liquid phase if the temperature drop along the tubing rises. Such behavior is more difficult to see for Case 4.1 because the algorithms rely less on the temperature drop feature, so that such partial dependence is less identifiable.

Explanation of complex multiphase flow behavior. Another observation for the tubing model is that with the increase of the mixture flow feature value, we see the increase of the gas rate and the decrease of the oil rate. As such, the model tells us that the increase of the mixture volumetric flowrate from the well will mainly correspond to the increase of the gas production and decrease of the oil production. This is exactly the behavior of most wells at the late production stage, when we see the increase of the gas and/or water production and decrease of the oil production. The model captures this relation from the training set.

At the same time, for the choke model in Case 2.1, we see that the increase of the constructed feature of the mixture flow and pressure drop corresponds to the increase of rates with some occasional non-linear fluctuations caused by the flow irregularities and non-smooth solutions produced by gradient boosting algorithm. For the oil rate, however, this dependence is more difficult to see because of the low importance for the algorithm when estimating the flow, as we observed in Fig. 17a. Such behavior gives us further insights about the system. In our case, the choke opening is almost always constant, so the increase of the pressure drop over the choke will correspond to the increase of the flowrate. As the mixture flowrate feature is also proportionally dependent on the pressure drop, it is positively correlated with the rates. The reason why the choke model behaves differently from the tubing model in terms of the increase/decrease of the phase rate when increasing the choke mixture rate is that the flowrate measurements are taking at the end of the tubing and before the choke (Fig. 9), as such the degassing/liquid accumulation effects are considered in the tubing model through the fluid density change, while in the choke model such effect is not considered.

Another observation is that the pressure related features such as pressure drop across the choke and tubing is generally more important for the gas estimation than for the oil estimation. This results is physically meaningful because the gas behavior is much



(a) Feature importance for oil estimation algorithms

(b) Feature importance for gas estimation algorithms

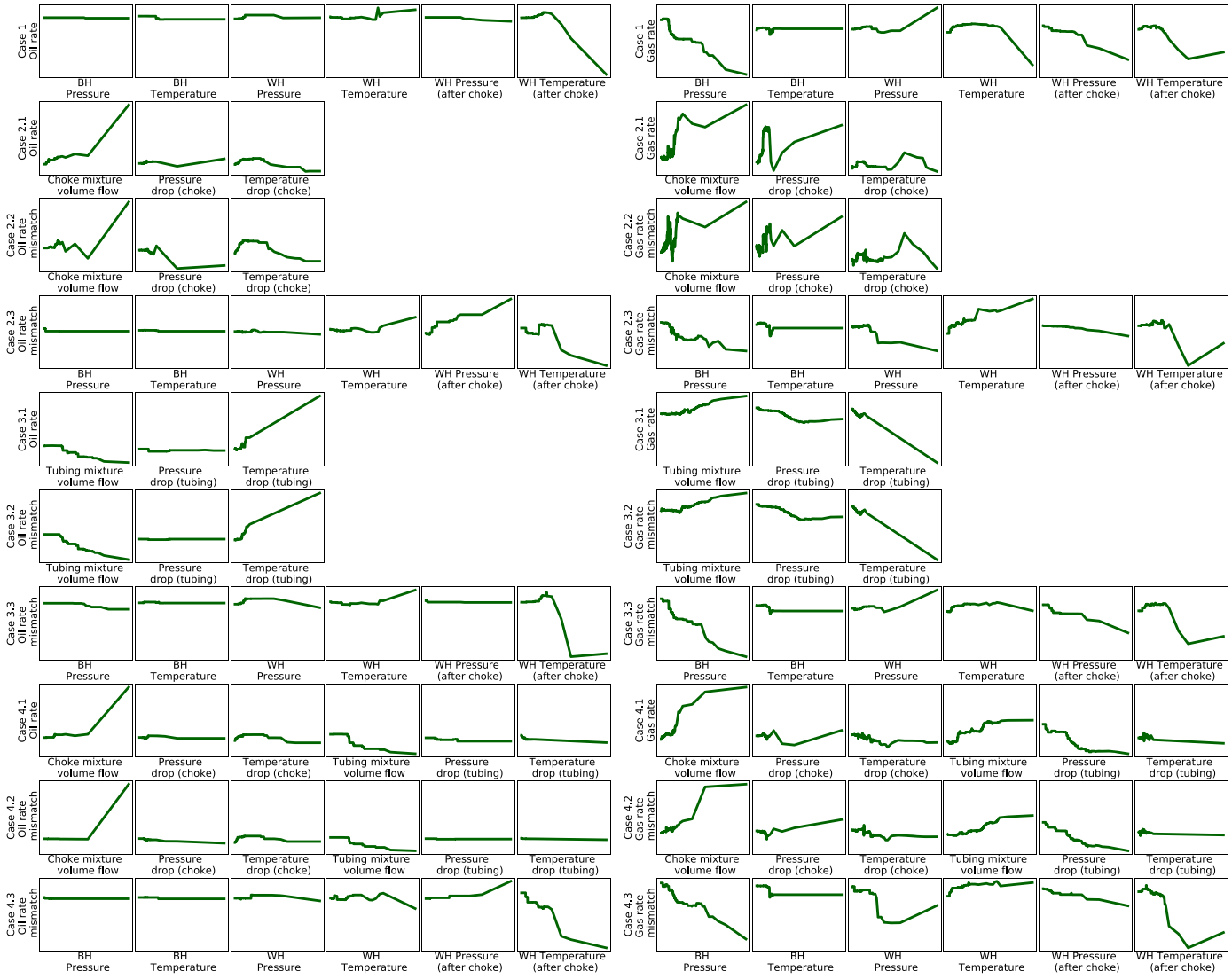
Fig. 17. Feature importance analysis for gradient boosting based estimation algorithms.

more affected by the pressure changes due to its high compressibility.

As such, through the provided analysis we see that it is possible to answer two questions: "Can we trust the model?" and "Can we get the new insights about the system?". More specifically, when we observe that the model is able to describe even a simple system behavior in a correct way, the trust in the model increases. In addition, after getting deeper insights about the model behavior thorough the analysis, we can dig into a more complex physics, for instance, which we did not think of before the analysis, e.g., the local degassing effects of the multiphase flow in case of our work.

Advantages of using the proposed feature analysis

Generally, we can say that, in addition to the evaluation of the algorithms transparency, the evaluation of the feature importance



(a) Partial dependence plots (oil predictions)

(b) Partial dependence plots (gas predictions)

Fig. 18. Partial dependence plots for gradient boosting based models. The plots show the qualitative relationship between the features and the target variable. Occasional spikes are associated with flow irregularities and non-smooth solutions provided by the gradient boosting algorithm.

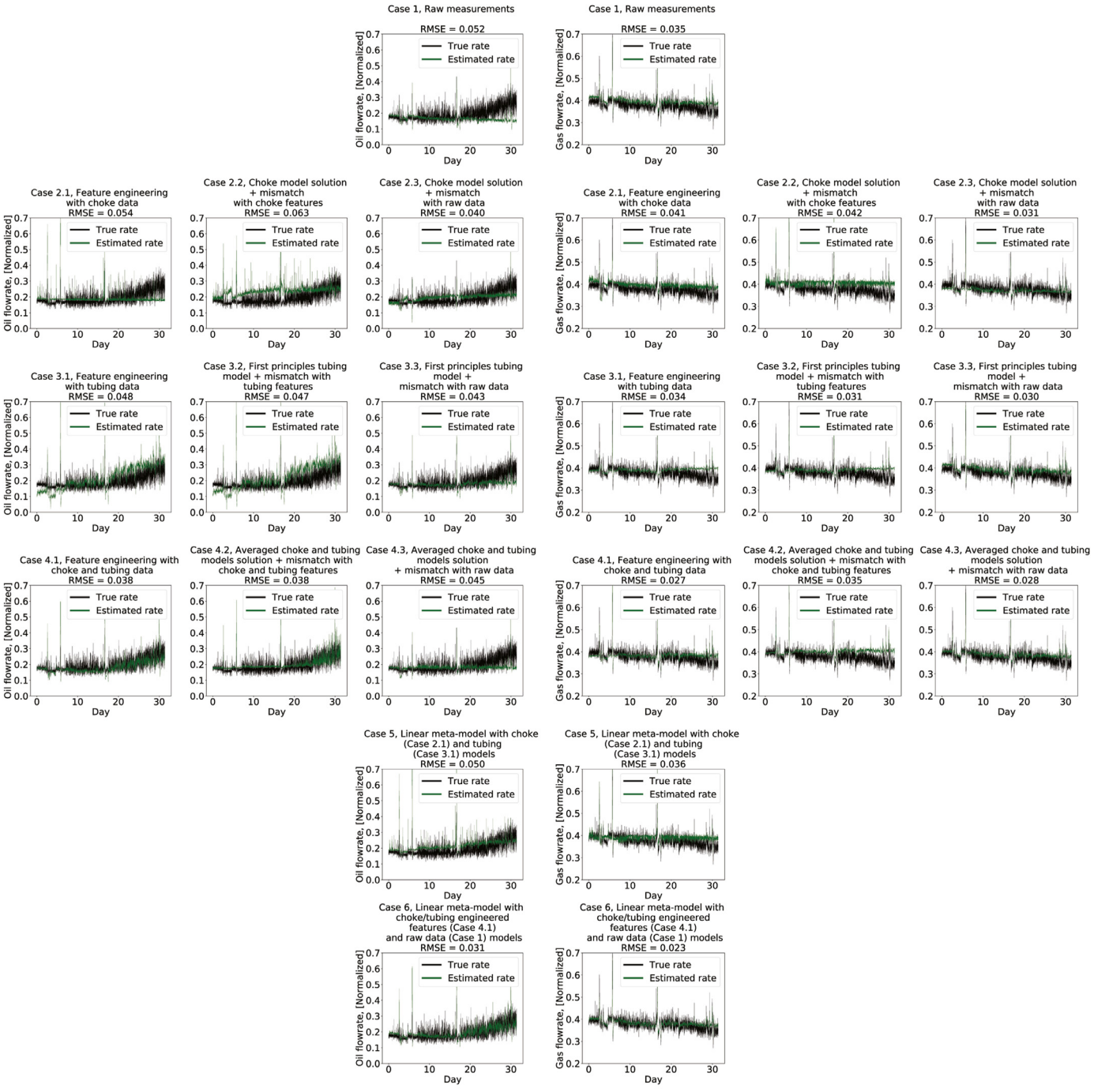
and partial dependencies shows possible directions for improving the estimation capabilities of each particular model. This means that by taking the discussed evaluation into account, we can distinguish which features and models behave non-physically, so that we can identify the bottlenecks and try to improve the models and features such that they better correspond to the expected physical behavior of the system. As a result, improvements of the estimation accuracy can be expected.

All the discussed observations would be hard or impossible to recover from the plots related to Case 1 only. An importance of a particular measurement can tell us much less than a physically interpretable feature as well as a comparative study of combinations of the first principles with machine learning algorithms. However, analysis of raw measurements can still be useful to conduct even for the model which uses raw measurements in order to see which measurements can be totally irrelevant and not included into the first principles features and models. For instance, in this case, this is the choke opening measurement in Case 1. This can be especially useful for modeling and analyzing of large scale systems.

5.2. Analysis of neural network results

5.2.1. Flowrate estimation results

Results overview and similarities between MLP neural network and gradient boosting results Fig. 19 shows the estimation results of the oil and gas flowrates using MLP neural networks. Generally, we see that the behavior of the models corresponds to the results obtained using the gradient boosting based models, such that most of the trends observed in the gradient boosting case, can also be observed here. For instance, we see that the combinations of choke and tubing features are able to improve the performance and to reconstruct the rising oil rate trend and decreasing gas rate trend at the end of the estimation period (Case 4.1). Apart from that, adding the raw data to the choke and tubing models in the gas estimation case boosts the performance (Case 2.3, 3.3 and 4.3). The same as with gradient boosting, the meta-model from Case 6 achieves the highest performance for both oil and gas rates. We will not go into detail about the neural networks behavior in cases where it is similar to the gradient boosting behavior, because these trends are well discussed in sections related to the gradient boosting results.



(a) Predicted oil flowrates (b) Predicted gas flowrates

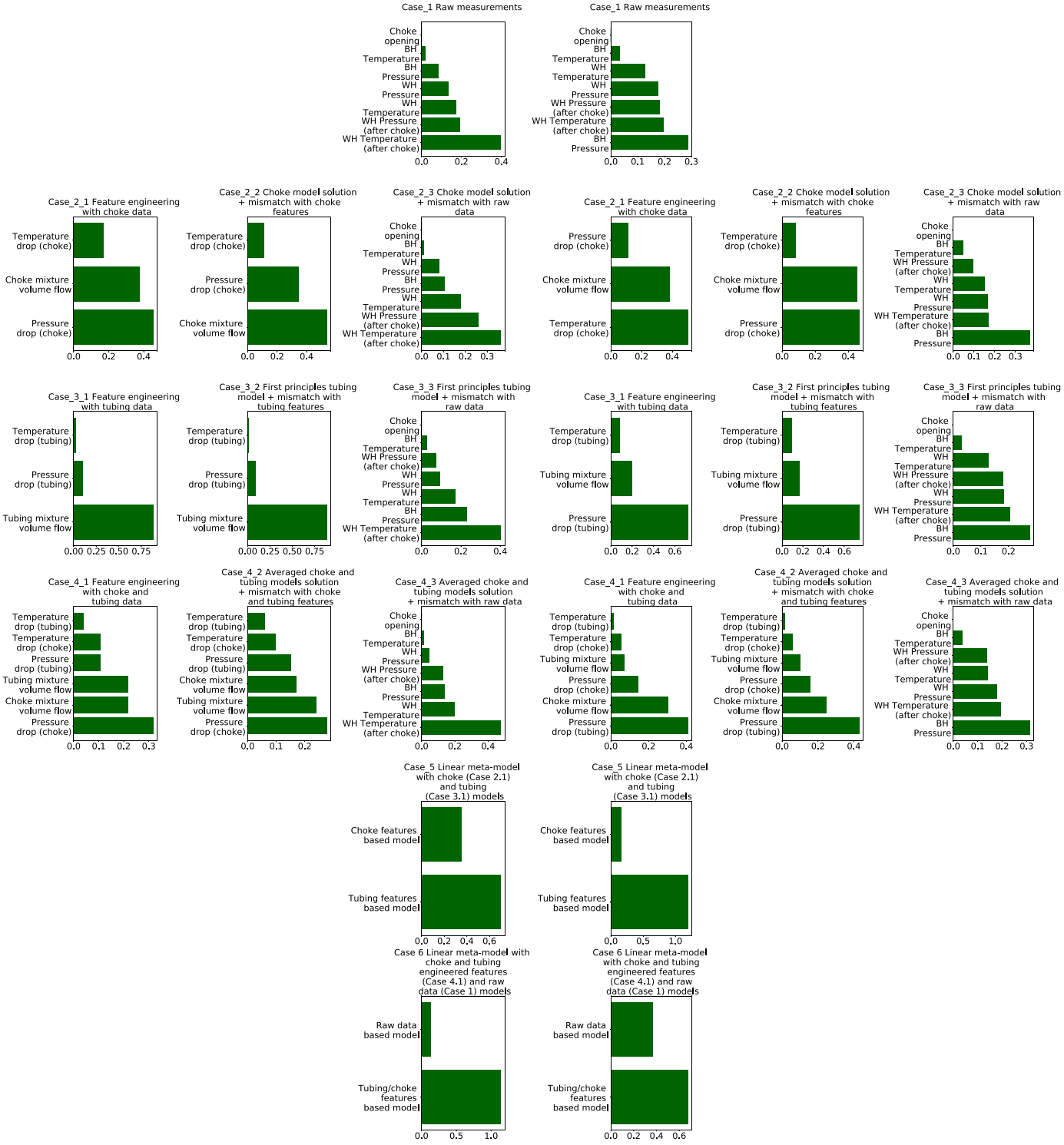
Fig. 19. Oil and gas rates estimation by MLP neural network based models. The results are averaged over 5 minutes period, i.e. one point - 5 minute averaged rate.

Instead, we will focus on the differences between gradient boosting and neural networks. To compare the results between two algorithms closer, Tables 5 and 6 show the comparative summary of the estimation results.

Differences between MLP neural network and gradient boosting results

The first difference between the results is that the neural networks are not able to use the tubing model and related features (Case 3.1, 3.2 and 3.3) as efficiently as gradient boosting providing less accurate results closer to the base case (Case 1) accuracy, while still more accurate. The reason for this may be the fact that, as will be shown in Section 5.2.2, the neural networks rely very much on the "Tubing mixture volume flow" feature and almost do not consider "Tubing pressure drop" and "Tubing temperature drop" features. However, the "Tubing mixture volume flow" alone may be too simple to accurately describe the flow. At the same time, gradient boosting based models do consider pressure and temperature drop features (Fig. 17a) which is likely why it helps the algorithm to better estimate the flowrates.

We also see that for the gas estimation cases, in each case the MLP neural networks outperform gradient boosting, which is not the same for the oil estimation. The exact reason for such behavior is unclear, but one explanation for such behavior may be that neural networks in general and MLP neural networks in particu-



(a) Feature importance for neural network (oil prediction) (b) Feature importance for neural network (gas prediction)

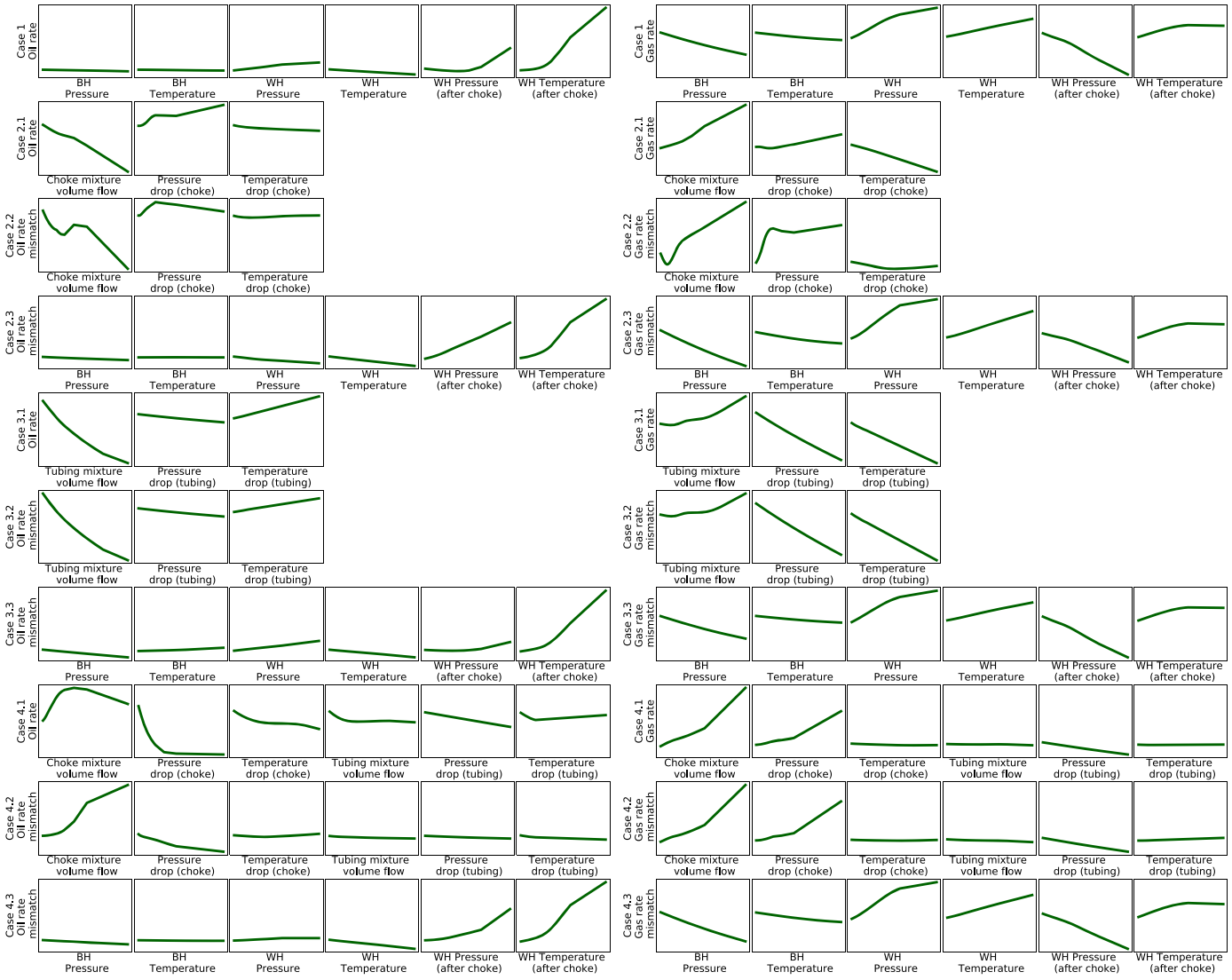
Fig. 20. Feature importance analysis for MLP neural network.

lar may be better at regression of more smooth values and system behavior (in this case - gas flow) because neural networks are typically better than gradient boosting in interpolation tasks. This is because gradient boosting with regression trees produces piecewise constant predictions while neural networks produce smooth interpolation approximations. This hypothesis will also be checked and discussed later for LSTM neural networks.

5.2.2. Feature analysis

Feature analysis overview and similarities between MLP neural network and gradient boosting results

Fig. 20 shows the feature analysis of the MLP neural networks based models. We see that for most of the cases with physics-based features, the feature importances are similar between the neural networks and gradient boosting. Some minor differences



(a) Partial dependence plots (oil predictions)

(b) Partial dependence plots (gas predictions)

Fig. 21. Partial dependence plots for MLP neural network based models. The plots show the qualitative relationship between the features and the target variable. The plots do not have non-convex spikes as in the gradient boosting case due to smooth solutions provided by neural networks.

exist, but they are caused by differences in the algorithms nature. What is more important is that the partial dependencies plots shown in Fig. 21 produce similar trends in most of the cases between neural networks and gradient boosting, except the fact that the plots produced by neural networks are more smooth. As such, we see that the algorithms interpret the physical behavior of the system in a similar way. This emphasizes that the consistent approach for machine learning modeling with first principles proposed in this work allows to produce consistent results and reveal the main structure of the data. It also emphasizes the fact that using a model-agnostic approach for feature evaluation can be more suitable and insightful than model-specific ones and gives the opportunity to better evaluate the validity of the produced estimates.

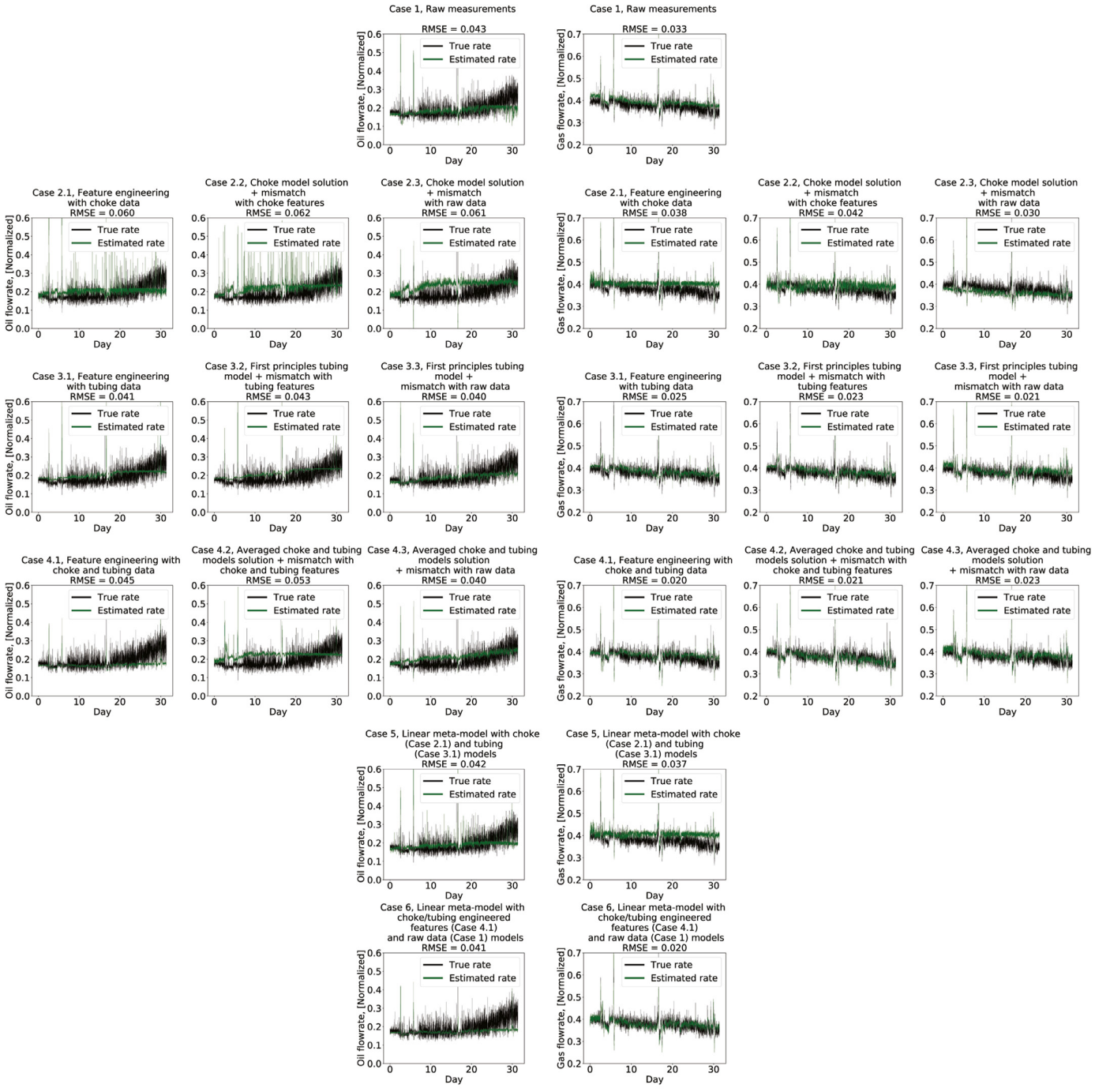
Differences between MLP neural network and gradient boosting feature analysis results

The major difference between MLP neural networks and gradient boosting partial dependencies plots is that the neural networks estimate the decrease of the oil flow when the choke mixture flow increases when used in Case 2.1 and 2.2. This is different from what gradient boosting suggests and what we would expect based on the physical understanding of the system. At the same time, when all the features are used together (Case 4.1 and 4.2), the neural networks give a more similar behavior to the gradient boosting which we found to be physically meaningful. The reason why the neural network gives this unreasonable evaluation of the choke mixture flow feature when used in the choke model alone is hard to explain and kept for future investigations. Despite some small differences, we can still see the big advantage of using physics-based features in terms of improved and consistent explainability of different models and algorithms, when compared to the raw data models.

5.3. Analysis of LSTM results

5.3.1. Flowrate estimation results

As discussed, for the LSTM neural network, only the estimation results are analyzed while the feature importance analysis is not conducted because the LSTM dependence on time step features which is not implemented in the Skater library used for the analysis. Despite the absence of feature importance analysis, we decided to include these results because it allows to analyze the depen-



(a) Predicted oil flowrates (b) Predicted gas flowrates

Fig. 22. Oil and gas rates estimation by LSTM neural network. The results are averaged over 5 minutes period, i.e. one point - 5 minute averaged rate.

dence of the results accuracy at the current time step on the data from the past time steps and compare it with the static approach used in MLP neural networks and gradient boosting.

Results overview and similarities between LSTM and gradient boosting/MLP neural network results

Fig. 22 shows the simulation results for the oil and gas flowrate estimation using LSTM neural networks. From the results we see some of the trends observed for gradient boosting and MLP neural networks can also be observed here. For instance, for the oil flowrate estimation, the pure choke-based machine learning models (Case 2.1, 2.2 and 2.3) do not perform well while try to capture dynamic behavior of the system which is represented by flowrate spike estimates. Also, similar to MLP neural networks and gradient boosting, the tubing-based machine learning models (Case 3.1, 3.2 and 3.3) capture a more steady state behavior of the system producing smooth oil flow estimation results. We also see that, similar to MLP neural networks, LSTM neural networks produce better results for gas rate estimation case, while in oil rate estimation cases gradient boosting is generally more accurate. This fact confirms the hypothesis, made for MLP neural networks previously, that the reason for such behavior is that neural network are generally better at predicting smooth regression trends due to their high interpola-

tion capabilities. However, trends with high fluctuations may be, in some cases, better described by piecewise constant approximations by gradient boosting.

Differences between LSTM and gradient boosting/MLP neural network results

The first difference between the algorithms to notice is that even when combining choke and tubing features in the oil case (Case 4.1), the rising trend at the end of the estimation period is not captured. So, we can see that, despite the LSTM network uses the previous measurements to predict the flow at the current time step, it does not necessarily help to produce an accurate estimate. In fact, we observed that the MLP neural network and gradient boosting, being totally different algorithms but both taking only the current time step measurements, were able to estimate the increasing flowrate trend accurately. The LSTM, however, in Case 4.1, takes 15 past time step measurements to make a prediction. As such, we conclude that it is not always a good idea to take the past measurements into account in order to predict the target variable value at the current time step. For instance, in case of the oil flowrate, the flow behavior is irregular which makes it difficult for the LSTM network to accurately reconstruct the time dependent flow pattern.

For the gas case, however, the LSTM generally performs much better than gradient boosting and MLP neural networks. We see that the gas flow behavior is much more stable, so that it is easier for the LSTM to reconstruct the time dependent pattern of the flow. The average time window chosen by the LSTM networks among the cases for the gas rate estimation via Bayesian optimization is 10, however, the exact value is case dependent.

When should we use LSTM in estimation of process system variables?

Summarizing all the results on LSTM and comparing it with static models of MLP neural networks and gradient boosting, we can conclude that using LSTM may be beneficial when the trend of the target variable is relatively smooth, but more importantly, assumed to have a time dependent pattern. In this work, the gas flowrate follows such a trend. This can be seen not only on the estimation result of the test data set show in Fig. 22b, but also in Fig. 10 where the entire gas flowrate trend is shown. In fact, in Fig. 10 we see that despite the fluctuating behavior, the gas flowrate has a systematic decreasing trend in time. The oil flowrate, however, has many irregular ups and downs, such that it is nearly impossible for a machine learning algorithm to reconstruct such a behavior. Therefore, when an LSTM is used to estimate it, the algorithm may try to learn the trend in time which simply does not exist which leads to high estimation error.

6. Results summary

In this section, we summarize the results of 72 cases which consider 12 different case studies of combining machine learning with first principles using 3 different machine learning algorithms. First, using Fig. 23, we describe how the reader can choose a suitable method for combining first principles models with machine learning depending on the process system under consideration by following a step-by-step approach. In Table 5, we summarize and average the RMSE for each algorithm and each flowrate. Such comparison is intended to shown which algorithm was the most accurate on average over the case studies conducted in this work for different system behaviors (oil and gas rates). Table 6 shows the average error over the algorithms (MLP neural network, gradient boosting, LSTM) for each first principles model as well as the averaged meta-model results to give more insights about the accuracy of the proposed methods. Then, we discuss the overall applicability of each method and its potential improvements.

How to choose the best method for your system?

In this work, we extensively tested several methods which vary by their accuracy, construction cost and applicability to different system conditions. In Fig. 23, we summarize the procedure of how all the proposed methods can be most effectively used when applied to modeling of a new process engineering system. By following these guidelines, the reader will hopefully be able to quickly choose the method which satisfies the desired accuracy and the amount of time available for model construction. Below in this section, we summarize the results which we observed in this work and which became the basis for creating Fig. 23 with the proposed selection guidelines.

Most accurate algorithms for oil and gas flowrates

From Table 5, we see that, among the algorithms, in the oil case, MLP neural network performs best on average (mean RMSE=0.0458), while gradient boosting shows slightly worse mean RMSE and LSTM performs relatively poorly. However, for the gas flowrate, the mean RMSE of LSTM is much lower than for MLP neural network and gradient boosting. As such, we confirm so-called "No Free Lunch" theorem (Wolpert et al., 1997) which states that there is no single algorithm which fits best for all cases. In this particular case, since the oil flowrate fluctuations are highly irregular, the time dependent pattern which LSTM is trying to find may not exist, so that the learning of the non-existing pattern deteriorates the results. As such, the algorithms which use only one time step measurements (MLP neural network and gradient boosting) outperform LSTM and perform almost equally well. At the same time, when the flow fluctuations have a more regular behavior and may have a time dependent pattern as in the gas rate case, LSTM is able to fit the data much better by taking advantage of the previous time step data.

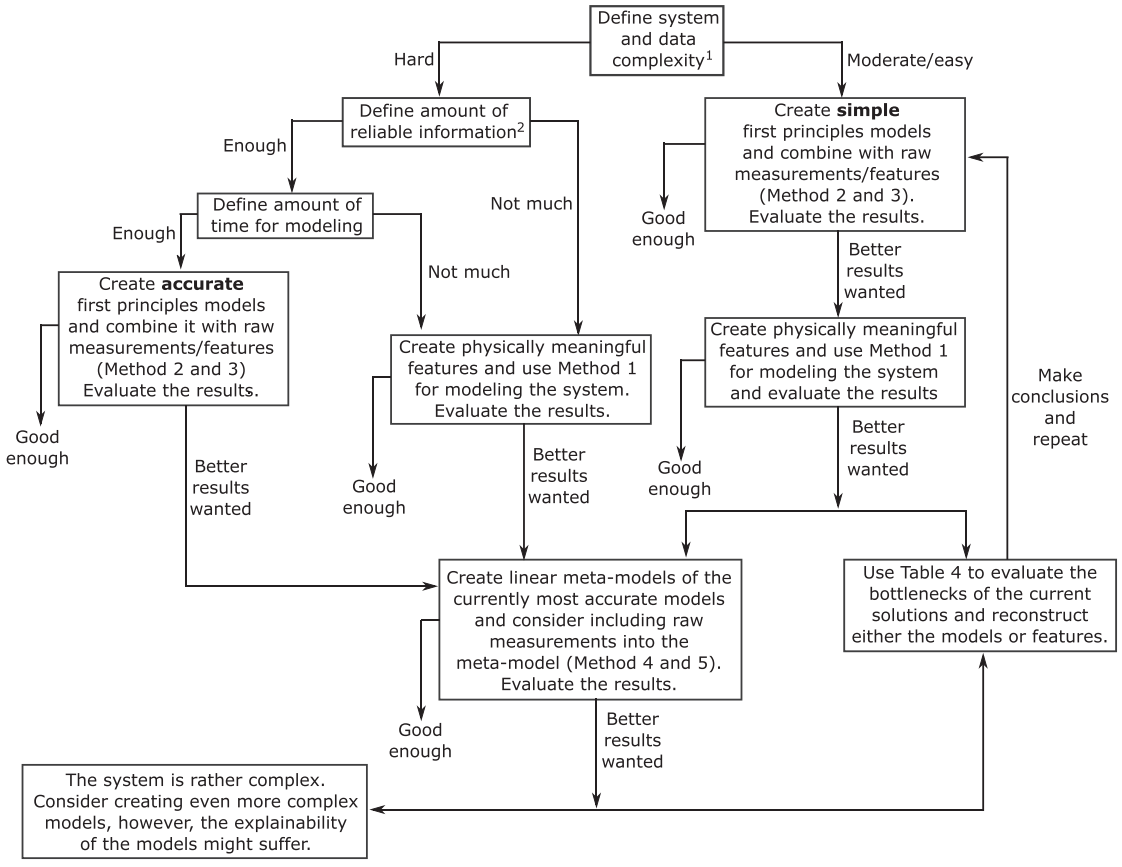
Superior accuracy of meta-models and applicability of feature engineering

From Table 6 we see that in the meta-model which combines the model with all the created features for choke and tubing with the raw data model (Method 5) outperforms all the other methods which results in the best averaged performance (RMSE=0.0340 and 0.0233). If choosing between the methods which do not consider meta-modeling, feature engineering (Method 1) shows the best performance (italic values). From this we conclude that if the goal of modeling is to achieve the highest performance, the linear meta-models which combines most of the available information is a good choice. Such models are slightly more difficult to construct, but they still maintain the interpretable behavior and generally improve the performance. Otherwise, the feature engineering method is another possibility which is slightly easier to construct that the meta-models, less accurate but still shows a good performance.

Applicability and future improvements of Methods 2 and 3 - combinations of first principles model solutions and machine learning models

As for the other methods of combining first principles and machine learning (Method 2 and 3), in this case, they performed less accurately than other discussed methods. However, this does not mean that we should not consider them to apply for other cases. For instance, as we observed for the gas estimation case, the combination of the first principles models with raw data (Case 2.3, 3.3 and 4.3) produced accurate performance. However, in the oil rate estimation case which has irregular behavior, the proposed models appear to be too simple to describe the system behavior accurately. As such, to be successfully applied, the models would need to be improved prior to modeling.

One possibility to do this is to pre-solve the models numerically for a small number of mesh points while maintaining the computational efficiency, so that the solution may not be very accurate, but still much better than one averaged over the entire system. An-



1 - The system complexity does not have a specific metric and should be defined by a domain knowledge expert. It may include the size of the system, complexity of the physical phenomenon, lack of physical understanding and first principles models for parts of the system, potential hidden patterns in the phenomenon, target variable behavior such as trend and distribution, etc.

2 - Amount of reliable information is case dependent and does not have a specific metric. It depends on the amount of noise in the data, outliers, number of measurements which are correlated with with the target variable, number of measurements which can be used to create first principles models, amount of data which can be obtained externally based on the process knowledge, etc.

Fig. 23. Summary of the method selection for process system modeling by combining first principles and machine learning models.

Table 5
RMSE summary for the conducted case studies. Underlined values - lowest error within the algorithm for each flowrate, bold values - lowest error for each flowrate within all algorithms, italic bold values - lowest mean error for each flowrate for all cases.

Case	Oil rate			Gas rate		
	XGBoost	MLP NN	LSTM	XGBoost	MLP NN	LSTM
Case 1	0.044	0.052	0.043	0.040	0.035	0.033
Case 2.1	0.056	0.054	0.060	0.045	0.041	0.038
Case 2.2	0.060	0.063	0.062	0.048	0.042	0.042
Case 2.3	0.064	0.040	0.061	0.035	0.031	0.030
Case 3.1	0.038	0.048	0.041	0.038	0.034	0.025
Case 3.2	0.042	0.047	0.043	0.036	0.031	0.023
Case 3.3	0.040	0.043	0.040	0.031	0.030	0.021
Case 4.1	0.037	0.038	0.045	0.028	0.027	0.020
Case 4.2	0.057	0.038	0.053	0.037	0.035	0.021
Case 4.3	0.047	0.045	0.040	0.030	0.028	0.023
Case 5	0.041	0.050	0.042	0.048	0.036	0.037
Case 6	**0.030**	0.031	0.041	0.024	0.023	**0.020**
Mean error	0.0463	***0.0458***	0.0476	0.0367	0.0328	***0.0278***

other possibility is to slightly pre-tune the model to the data, for instance, using a linear model, and then use in combination with machine learning. Finally, the complexity of the models can be increased, so that the model become able to better resolve the system behavior, for instance, in this case, this can be a slip model for the choke equation. However, still computational efficiency should be considered, so that the models should not become very complex. All these proposed improvements will lead to the fact that the solutions from the first principles model will be more accurate, as such the mismatch between the actual target value and the solution will be lower and easier to be covered by a machine learning algorithm.

Table 6
Summary of RMSE averaged over the algorithms for each method and each flowrate, e.g. choke oil RMSE of 0.0566 = (0.056 + 0.054 + 0.060)/3. Underlined values - the lowest error within each method for each flowrate, italic values - the lowest error among the methods *excluding* meta-models, bold values - lowest error for each flowrate among all methods.

Model	Method											
	Base method Raw data measurements as features		Method 1 Feature engineering		Method 2 First principles model solution + feature engineering		Method 3 First principles model solution + raw data model		Method 4 Linear meta-model of models with created features (Case 2 and 3)		Method 5 Linear meta-model of selected model with created features and raw data model (Case 4 and base case)	
	Oil	Gas	Oil	Gas	Oil	Gas	Oil	Gas	Oil	Gas	Oil	Gas
Choke (Case 2)	-	-	0.0566	0.0413	0.0616	0.0440	0.0550	0.0320	0.0443	0.0403	-	-
Tubing (Case 3)	-	-	0.0423	0.0323	0.0440	0.0300	0.0410	0.0273				
Choke and tubing (Case 4)	-	-	*0.0400*	*0.0250*	0.0467	0.0310	0.0440	0.0270	-	-	**0.0340**	**0.0223**
Raw data (base case)	0.0463	0.0360	-	-	-	-	-	-				

7. Conclusions

In this paper, we propose and analyze several methods for combining first principles models with machine learning applied to multiphase flowrate estimation problem in petroleum production systems. For the machine learning algorithms, MLP and LSTM neural networks and gradient boosting were chosen. The proposed methods for combining first principles with machine learning were applied to all the aforementioned algorithms. The algorithms were systematically tuned via a pipeline which is based on the Bayesian optimization approach which ensures fair and accurate model tuning and comparison.

We found that by introducing first principles models into machine learning algorithms, it becomes possible to improve the estimation performance if compared to approaches when raw measurement data is used directly, as it has been done for the considered problem in many other works reported in the literature. We found that for the irregular system behavior, it is better to use static models such as MLP neural networks or gradient boosting and not take past measurements into account. On the other hand, when the system behavior is less complex and has a time-dependent pattern, LSTM neural networks which consider the past measurements show the superior performance.

We discovered that linear meta-models which combine physics-aware machine learning algorithms with raw measurement models show the most accurate performance while maintaining good interpretability. Feature engineering method can also be a good choice to incorporate first principles into machine learning because it has lower development cost than linear meta-models while maintains a reasonable performance. The methods which combine first principles models solution with machine learning showed less accurate performance for complex system behavior than the meta-models and feature engineering approach, while in less complex system they were accurate enough. As such, to be applied for complex systems, the first principles models should be relatively accurate, such that they produce a reasonable solution and small mismatch between its solution and the true target value which can further be covered by machine learning algorithms.

Another important finding is that by introducing physics-based features into machine learning algorithms, it is possible to create much more interpretable models than models which use raw data directly. We showed that by using model-agnostic feature importance evaluation methods and revealing partial dependences between the features and the target, it is possible not only to ensure that the obtained machine learning model behaves physically feasible, but also reveal additional insights about the complex system behavior, hidden patterns, physical phenomena and identify possible directions for the model improvements.

Overall, we conclude that to successfully apply machine learning to complex process engineering systems in general and Virtual Flow Metering in particular, we need to incorporate first principles approaches into machine learning algorithms. This approach creates more accurate and, more importantly, more transparent data-driven solutions which will develop more trust to these systems among the operating professionals and will further contribute to a more efficient and reliable systems operation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.compchemeng.2020.106834

CRediT authorship contribution statement

Timur Bikmukhametov: Writing - original draft, Writing - review & editing, Conceptualization, Methodology, Software, Formal analysis, Investigation. **Johannes Jäschke:** Writing - original draft, Supervision, Conceptualization.

References

Agarap, A.F., 2018. Deep learning using rectified linear units (relu). arXiv:1803.08375.
Ahmadi, M.A., Chen, Z., 2018. Comparison of machine learning methods for estimating permeability and porosity of oil reservoirs via petro-physical logs. Petroleum.
AL-Qutami, T.A., Ibrahim, R., Ismail, I., Ishak, M.A., 2017. Radial basis function network to predict gas flow rate in multiphase flow. In: Proceedings of the 9th International Conference on Machine Learning and Computing. ACM, pp. 141–146.
AL-Qutami, T.A., Ibrahim, R., Ismail, I., Ishak, M.A., 2018. Virtual multiphase flow metering using diverse neural network ensemble and adaptive simulated annealing. Expert Syst. Appl. 93, 72–85.
Andrianov, N., 2018. A machine learning approach for virtual flow metering and forecasting. IFAC-PapersOnLine 51 (8), 191–196.
Anifowose, F.A., Labadin, J., Abdulraheem, A., 2017. Hybrid intelligent systems in petroleum reservoir characterization and modeling: the journey so far and the challenges ahead. J. Pet. Explor. Prod. Technol. 7 (1), 251–263.
Berger-Tal, O., Nathan, J., Meron, E., Saltz, D., 2014. The exploration-exploitation dilemma: a multidisciplinary framework. PLoS ONE 9 (4), e95693.
Bhutani, N., Rangaiah, G., Ray, A., 2006. First-principles, data-based, and hybrid modeling and optimization of an industrial hydrocracking unit. Ind. Eng. Chem. Res. 45 (23), 7807–7816.
Bikmukhametov, T., Jäschke, J., 2019. First principles and machine learning virtual flow metering: a literature review. J. Petrol. Sci. Eng. 106487.
Bikmukhametov, T., Jäschke, J., 2019. Oil production monitoring using gradient boosting machine learning algorithm. IFAC-PapersOnLine 52 (1), 514–519.

Chen, T., Guestrin, C., 2016. Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and data Mining. ACM, pp. 785–794.
Cheng, G., Peddinti, V., Povey, D., Manohar, V., Khudanpur, S., Yan, Y., 2017. An exploration of dropout with lstms.. In: Interspeech, pp. 1586–1590.
Chisholm, D., 1983. Two-phase flow in pipelines and heat exchangers. G. Godwin in association with Institution of Chemical Engineers.
Falcone, G., Hewitt, G., Alimonti, C., 2009. Multiphase flow metering: Principles and applications, 54. Elsevier.
Falcone, G., Hewitt, G., Alimonti, C., Harrison, B., et al., 2001. Multiphase flow metering: current trends and future developments. In: SPE Annual Technical Conference and Exhibition. Society of Petroleum Engineers.
Frazier, P.I., 2018. A tutorial on bayesian optimization arXiv:1807.02811.
Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. Ann. Stat. 1189–1232.
Genuer, R., Poggi, J.-M., Tuleau-Malot, C., 2010. Variable selection using random forests. Pattern Recognit. Lett. 31 (14), 2225–2236.
Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press. http://www.deeplearningbook.org.
Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J., 2016. Lstm: a search space odyssey. IEEE Trans. Neural Netw. Learn. Syst. 28 (10), 2222–2232.
Hastie, T., Tibshirani, R., Friedman, J., Franklin, J., 2005. The elements of statistical learning: data mining, inference and prediction. Math. Intell. 27 (2), 83–85.
Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.
Hornik, K., Stinchcombe, M., White, H., 1989. Multilayer feedforward networks are universal approximators. Neural Netw. 2 (5), 359–366.
Idso, E.S., Sperle, I.L., Aasheim, R., Wold, M.S., et al., 2014. Automatic subsea deduction well testing for increased accuracy and reduced test time. In: Abu Dhabi International Petroleum Exhibition and Conference. Society of Petroleum Engineers.
Kanin, E., Osiptsov, A., Vainshtein, A., Burnaev, E., 2019. A predictive model for steady-state multiphase pipe flow: machine learning on lab data. J. Petrol. Sci. Eng. 180, 727–746.
Kingma, D. P., Ba, J., 2014. Adam: A method for stochastic optimization arXiv:1412.6980.
Klyuchnikov, N., Zaytsev, A., Gruzdev, A., Ovchinnikov, G., Antipova, K., Ismailova, L., Muravleva, E., Burnaev, E., Semenikhin, A., Cherepanov, A., et al., 2019. Data–driven model for the identification of the rock type at a drilling bit. J. Petrol. Sci. Eng. 178, 506–516.
Kramer, A., Choudhary, P., et al., 2018. Skater: Python library for model interpretation/explanations. https://github.com/oracle/Skater.
Liu, X., Faes, L., Kale, A.U., Wagner, S.K., Fu, D.J., Bruynseels, A., Mahendiran, T., Moraes, G., Shamdas, M., Kern, C., et al., 2019. A comparison of deep learning performance against health-care professionals in detecting diseases from medical imaging: a systematic review and meta-analysis. Lancet Digit. Health 1 (6), e271–e297.
Loh, K., Omrani, P. S., van der Linden, R., 2018. Deep learning and data assimilation for real-time production prediction in natural gas wells arXiv:1802.05141.
Lunde, G.G., Rudrum, G., Angelo, P., Holmas, K., Setyadi, G.R., et al., 2013. Ormen lange flow assurance system (fas)-online flow assurance monitoring and advice. In: OTC Brasil. Offshore Technology Conference.
Masella, J., Tran, Q., Ferre, D., Pauchon, C., 1998. Transient simulation of two-phase flows in pipes. Int. J. Multiphase Flow 24 (5), 739–755.
Matzopoulos, M., 2011. Dynamic process modeling: combining models and experimental data to solve industrial problems. Process Syst. Eng. 7, 1–33.
Molnar, C., Casalicchio, G., Bischl, B., 2018. Iml: an r package for interpretable machine learning.. J. Open Sour. Softw. 3 (26), 786.
Onwuchekwa, C., et al., 2018. Application of machine learning ideas to reservoir fluid properties estimation. In: SPE Nigeria Annual International Conference and Exhibition. Society of Petroleum Engineers.
Patel, P., Odden, H., Djoric, B., Garner, R.D., Vea, H.K., et al., 2014. Model based multiphase metering and production allocation. In: Offshore Technology Conference-Asia. Offshore Technology Conference.
Psichogios, D.C., Ungar, L.H., 1992. A hybrid neural network-first principles approach to process modeling. AIChE J. 38 (10), 1499–1511.
Qin, S.J., Chiang, L.H., 2019. Advances and opportunities in machine learning for process data analytics. Comput. Chemi. Eng. 126, 465–473.
Rasmussen, C.E., Williams, C., 2017. Gaussian processes for machine learning. 2006. Cited on 95.
Ribeiro, M.T., Singh, S., Guestrin, C., 2016. Why should i trust you?: Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp. 1135–1144.
Roscher, R., Bohn, B., Duarte, M. F., Garcke, J., 2019. Explainable machine learning for scientific insights and discoveries arXiv:1905.08883.
Rumelhart, D.E., Hinton, G.E., Williams, R.J., et al., 1988. Learning representations by back-propagating errors. Cognit. Model. 5 (3), 1.
Schuller, R.B., Munaweera, S.J., Selmer-Olsen, S., Solbakken, T., et al., 2006. Critical and sub-critical oil/gas/water mass flow rate experiments and predictions for chokes. SPE Prod. Oper. 21 (03), 372–380.
Shang, C., You, F., 2019. Data analytics and machine learning for smart process manufacturing: recent advances and perspectives in the big data era. Engineering.
Shoeibi Omrani, P., Dobrovolschi, I., Belfroid, S., Kronberger, P., Munoz, E., et al., 2018. Improving the accuracy of virtual flow metering and back-allocation through machine learning. In: Abu Dhabi International Petroleum Exhibition & Conference. Society of Petroleum Engineers.
Soave, G., 1972. Equilibrium constants from a modified redlich-kwong equation of state. Chem. Eng. Sci. 27 (6), 1197–1203.
Sun, J., Ma, X., Kazi, M., et al., 2018. Comparison of decline curve analysis dca with recursive neural networks rnn for production forecast of multiple wells. SPE Western Regional Meeting. Society of Petroleum Engineers.
Wolpert, D.H., Macready, W.G., et al., 1997. No free lunch theorems for optimization. IEEE Trans. Evol. Comput. 1 (1), 67–82.
Zangl, G., Hermann, R., Schweiger, C., et al., 2014. Comparison of methods for stochastic multiphase flow rate estimation. In: SPE Annual Technical Conference and Exhibition. Society of Petroleum Engineers.