UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

MARCOS VINICIUS LUDWIG PIVETTA

A Machine Learning Approach for Petroleum Production Forecasting in a Digital Twin

Thesis presented in partial fulfillment of the requirements for the degree of Master of Computer Science

Advisor: Prof. Dr. Joel Luis Carbonera

Porto Alegre
April 2024

CIP — CATALOGING-IN-PUBLICATION

Pivetta, Marcos Vinicius Ludwig

A Machine Learning Approach for Petroleum Production Forecasting in a Digital Twin / Marcos Vinicius Ludwig Pivetta. – Porto Alegre: PPGC da UFRGS, 2024.

86 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2024. Advisor: Joel Luis Carbonera.

1. Time Series Forecasting. 2. Petroleum Production Forecast. 3. Machine Learning. 4. Digital Twin. I. Carbonera, Joel Luis. II. Título.

ACKNOWLEDGEMENTS

This work was performed under the Petwin research project (PeTWIN.org). The research was supported by Higher Education Personnel Improvement Coordination (CAPES) code 0001, Brazilian National Council for Scientific and Technological Development (CNPq), Brazilian Federal Agency for Innovation FINEP financing, and Libra Consortium.

I want to thank my advisor, Prof. Luis Carbonera, who guided me thoroughly in the path of completing this master's, even with the multiple obstacles presented in its course. I would also like to thank Prof. Mara Abel and Prof. João Cesar Netto for being exceptional supervisors for all involved in the Petwin research project. Thanks to all my research group colleagues, who helped me immensely with the practical side of the scientific task. A special thanks to Artur Henrique Simon for providing feedback on my work and developing our architectural solution's front end.

Finally, thanks to my family and girlfriend Julia, who all provided additional support through the years and certainly will keep doing so for the upcoming ones.

ABSTRACT

Digital Twins (DTs), a forefront technology in Industry 4.0, find extensive applications in O&G, particularly in asset performance management. By virtually replicating a physical asset, DTs provide a digital platform for exploring various scenarios and problem-solving strategies without disrupting operational processes. Integrating DT principles and Machine Learning (ML) techniques holds significant promise for enhancing production forecasting capabilities in the O&G industry, enabling more informed decision-making and efficient production management. Traditional production forecasting methods in the industry face limitations such as dependence on expert experience, data availability, and computational costs. Leveraging ML algorithms offers a promising alternative to complex methods. In this work, we proposed a Proof of Concept (POC) digital twin component able to train ML petroleum forecasting models, execute them within what-if scenarios and present their results to end-users. We also gather results and raise insights from a model selection strategy executed by this DT component. This strategy investigated the impact on model performance of: (1) different lookback and horizon sizes for the training samples, (2) the use of future operational variables, and (3) the choice of different data sampling frequencies. The proposed architecture fulfilled the stakeholder's requirements and serves as an adequate digital twin component for the upstream sector. Results of the model comparison framework showed that the ensemble tree-based model XGBoost performed better than deep learning techniques for this particular dataset. We found that the optimal training window size depends on the sampling frequency of the data. We also determined that incorporating future operational features overall improves model performance. We conclude that better standards for data-driven petroleum forecasts are needed, especially in pre-processing and feature selection practices, which need further evaluation.

Keywords: Time Series Forecasting. Petroleum Production Forecast. Machine Learning. Digital Twin.

Uma Abordagem de Aprendizado de Máquina para Previsão de Produção de Petróleo para Gêmeos Digitais

RESUMO

Os Gêmeos Digitais (GDs), uma tecnologia de ponta na Indústria 4.0, encontram amplas aplicações no setor de Óleo e Gás (O&G), especialmente na gestão de desempenho de ativos. Ao replicar virtualmente um ativo físico, os GDs fornecem uma plataforma digital para explorar vários cenários e estratégias de resolução de problemas sem interromper os processos operacionais. A integração dos princípios dos GDs e técnicas de Aprendizado de Máquina (AM) oferece uma promessa importante para aprimorar as capacidades de previsão de produção na indústria de O&G, possibilitando uma tomada de decisão mais informada e uma gestão de produção mais eficiente. Métodos tradicionais de previsão de produção na indústria têm limitações, como dependência da experiência especializada, disponibilidade de dados e custos computacionais. A utilização de algoritmos de AM oferece uma alternativa promissora aos métodos complexos. Neste trabalho, propomos um componente Prova de Conceito (POC) para GDs capaz de treinar modelos de AM para previsão de petróleo, executá-los em cenários hipotéticos e apresentar os resultados ao usuário final. Também reunimos resultados e *insights* de uma estratégia de seleção de modelo executada por esse componente de GD. Esta estratégia investigou o impacto no desempenho do modelo de: (1) diferentes tamanhos de janelas amostrais de treinamento, (2) uso de variáveis operacionais futuras e (3) escolha de diferentes frequências de amostragem de dados. A arquitetura proposta atendeu aos requisitos das partes interessadas e serve como um componente de gêmeo digital adequado para o setor de *upstream*. Os resultados da estratégia de comparação de modelos mostraram que o modelo de árvore *ensemble* XGBoost apresentou melhor desempenho do que as técnicas de aprendizado profundo para este conjunto de dados específico. Verificamos que o tamanho ideal da janela amostral depende da frequência de amostragem dos dados. Também determinamos que a incorporação de características operacionais futuras melhora o desempenho do modelo. Concluímos que são necessários melhores padrões para previsões de petróleo baseadas em dados, especialmente quanto as práticas de pré-processamento e seleção de *features*, que necessitam de uma avaliação mais aprofundada.

Palavras-chave: Previsão de Séries Temporais. Previsão de Produção de Petróleo. Aprendizado de Máquina. Gêmeos Digitais.

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
ARIMA	Autoregressive Moving Average
CLI	Command-Line Interface
CNN	Convolutional Neural Network
CV	Cross-Validation
DCA	Decline Curve Analysis
DT	Digital Twin
EDA	Exploratory Data Analysis
EnKF	Ensemble Kalman Filter
EOR	Enhanced Oil Recovery
ES	Exponential Smoothing
EWT	Extended Well Test
FDM	Finite Difference Method
FNN	Feed-Forward Neural Network
FPSO	Floating Production Storage and Offloading
GOR	Gas-Oil Ratio
GPR	Gaussian Process Regression
GRN	Gated Residual Network
HONN	Higher Order Neural Network
ICV	Inflow Control Valve
IoT	Internet of Things
IQR	Interquartile Range
LR	Linear Regression
LSTM	Long Short-Term Memory

MBE	Material Balance Equation
MIF	Mixed Input Forecaster
ML	Machine Learning
MLMVN	Multilayer Neural Network with Multi-Valued Neurons
MLR	Multiple Linear Regression
MPFM	Multiphase Flow Meter
MSE	Mean Squared Error
NN	Neural Network
NEA	Nonlinear Extension for Linear Arps
O&G	Oil and Gas
PIMS	Plant Information Management System
P/T	Pressure/Temperature
POC	Proof of Concept
PSC	Production Sharing Contract
PVC	Persistent Volume Claim
RNN	Recurrent Neural Network
RMSE	Root Mean Squared Error
SARIMA	Seasonal Autoregressive Moving Average
TFT	Temporal Fusion Transformer
TSA	Time Series Analysis
VAR	Vector Autoregression
VFM	Virtual Flow Metering
XGBoost	eXtreme Gradient Boosting

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

CONTENTS

1 INTRODUCTION

Digital Twins (DTs), a cutting-edge technology of industry 4.0, have many applications in the O&G domain, including asset performance management (WANASINGHE et al., 2020). DTs offer a concise way to deal with the problem of sensor data communication between assets, history keeping, and what-if scenario execution in massive oil production facilities. It gives companies a digital environment where diverse scenarios and problem-solving strategies can be performed without disrupting reservoir, well, or platform operations (SIRCAR et al., 2023; GEP, 2020). By creating a virtual replica of a real asset using multiple data acquisition hardware and computational models, DTs can gather and apply data that helps drive decisions on the physical asset. Integrating DT principles and Machine Learning (ML) techniques is a promising way to add value to processes within upstream production operations (Honeywell, 2020), such as petroleum production forecasting. DTs and ML, in conjunction, when applied to this task, enable more informed decision-making and efficient production management.

Estimating oil wells' current and future production performance is a crucial task within oil field management (TARIQ et al., 2021). Many complex computational models based on multiphase substance flow through reservoir and facility equipment have been devised to address this problem. Nonetheless, highly uncertain factors relating to the fluid and subsurface dynamics pose a problem for this specific task (HUBBERT, 1940; HUBBERT, 1956; MUSKAT, 1946; BEAR, 1988). Furthermore, factors such as well-to-well interference, well completion, changes in the petrophysics of the reservoir over time, political decisions, and facility operational plans can also significantly impact the volume of petroleum produced by wells. To ensure the economic well-being of oil companies, it is essential to reduce the risks and uncertainties associated with petroleum extraction by estimating desirable and undesirable production behavior and determining the optimal parameters for production.

The more complex methods for production forecasts have disadvantages, such as reliance on professional experience, data availability, data acquisition, and computational costs. Furthermore, they are based on assumptions about fluid physics or oversimplifications (BALAJI et al., 2018). These downsides hinder their applicability in specific contexts. For example, the production engineer might want to quickly run a what-if scenario of production volume based on specific opening settings of flow-restricting valves on the well and the platform. It is unlikely that they will find the answer in a short period

using more complex numerical models. The same can be said for situations where multiple what-if scenarios are queued for execution. Moreover, data acquisition constraints could be such that no reliable numerical model exists for that field.

The adoption of the Industry 4.0 mindset in the Oil and Gas (O&G) industry led to an increased instrumentation of wells and production facilities (NGUYEN; GOSINE; WARRIAN, 2020), resulting in greater availability of real-time measurements of reservoir and well conditions, such as pressure and temperature in different well segments, valve operations, and fluid rates. Soft computing methods can leverage these data to train mathematical models that estimate current and predict future petroleum production. Future facility operational plans can also be used to feed training, allowing for hypothetical or what-if scenarios, depending on the operational plans used as input. Instead of the model-driven approach, this data-driven approach removes or reduces the need for professional experience, costly geological data acquisition, computational costs, and well shut-ins required to calibrate Multiphase Flow Meters (MPFMs).

As backed previously, short and long-term oil production estimation and forecasting are critical tasks in the O&G industry. Even with today's trend of collecting more granular real-time data from assets, the improvement of ML models that handle time series data, and the greater availability of hardware for training and inference of such models, literature still lacks works that compare ML models' accuracy at forecasting production time series and that simultaneously:

- test the impact and usefulness of future covariate models that take into account the plans of flow-restricting valve actuation.
- systematically compare the impact of different window sample sizes when training different data-driven models.
- use as input real-world hourly and sub-hourly observation from wells.

We hypothesized that new insights about using data-driven models in this task could be revealed if submitted to the different parameters and configurations in the list above and evaluated on real-world production data. A prior study we conducted has assessed partial results of such analysis (PIVETTA et al., 2023), but on a more limited scale. We constrain this pure data-driven analysis to a scenario where a reservoir model is non-existent or incomplete, reservoir behavior is similar to the data being analyzed, computational power is limited, and decision-making must be done promptly. At the same

time, knowing the prevalence of DTs for asset performance management in the current industry, we planned and developed a Proof of Concept (POC) digital-twin component for the prediction task. The POC was also used to execute the previously described model evaluation and generate results.

In summary, the objectives of this dissertation are:

1. to review and discuss studies that propose methods of production forecast that rely on temporal characteristics of the data.

2. to bring new insights about using data-driven production forecasting models by analyzing, in conjunction, the impact of future covariates, different window sizes, and sampling frequencies and evaluating them on the same real-world oil data with off-the-shelf ML techniques.

3. to propose and make available a deployable POC architecture that acts as a digital-twin component for forecasting oil production and for generating this work results.

This work is organized as follows. Section 2 gives the theoretical basis for understanding the subsequent work. We describe the concept of digital twins, petroleum production systems, and the main techniques to estimate petroleum production. Section 3 reviews the current literature for this task, highlighting the most recent state of research, the most prominent techniques, and observed methodological limitations. Section 4.1 presents our experimental setup, including the problem definition, description of the private dataset used, data exploration, preprocessing, feature selection rule, and Cross-Validation technique. The results of the experiments are discussed in Section 4.1.2. Section 4.2 describes the proposed DT component as a deployable model training and inference full-stack architecture. Finally, Section 5 concludes the work by summarizing our findings, discussing their implications, and suggesting future research directions.

2 THEORETICAL FOUNDATION

In the following subsections, we provide the theoretical foundation for understanding the problem of petroleum production forecasting and its connection to DTs. Section 2.1 explains the concept of DTs and how they can be used to create a virtual replica of an oil production system and the associated oil forecasting component for asset performance management. Section 2.2 describes, in general, how oil production systems work and the specific description of the oilfield where the data for experiments was taken from Libra. Section 2.3 details the challenges of effectively estimating petroleum production. Section 2.4 introduces the concept of time series forecasting and the leading techniques and concepts involved in this task. We dive deeper into ML methods for forecasting, such as Linear Regression (LR), eXtreme Gradient Boosting (XGBoost), Long Short-Term Memory (LSTM), and Temporal Fusion Transformer (TFT).

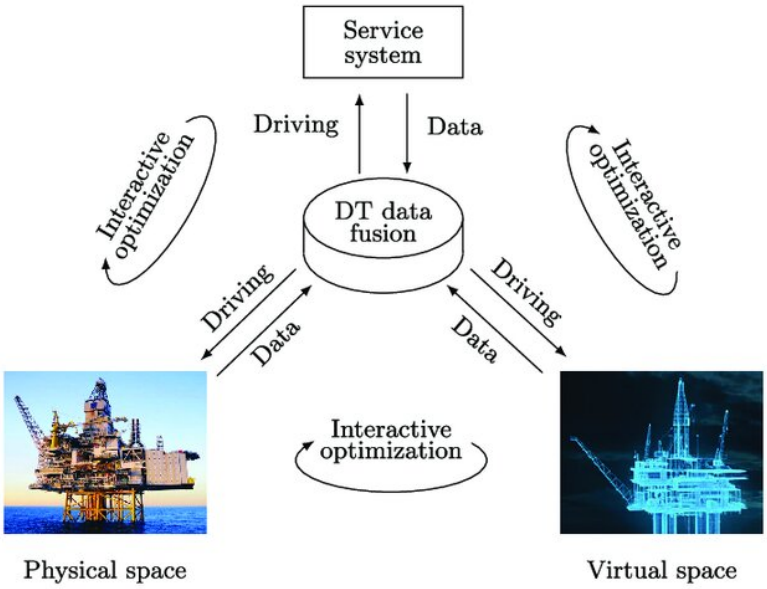
2.1 Digital Twins

Digital Twins (DTs) are an approach that integrates sensor technologies and computational models to replicate processes and physical assets in a virtual space. Industry 4.0 mindset has led companies to acquire Internet of Things (IoT) devices and technologies to deal with big data and analytics. Coupled with computing hardware and ML technology advances, these feats have enabled the recent steep growth of DTs in industries. Raw business-specific data gathered from IoT devices can be fed to computational models to provide real-time monitoring, prediction, optimization, controlling, and aid decision-making (RASHEED; SAN; KVAMSDAL, 2020) in the context of complex systems.

The concept of DTs, first conceived by Grieves (2015), has been iterated multiple times and sparked a series of debates about its true definition. Literature reviews (RASHEED; SAN; KVAMSDAL, 2020; SEMERARO et al., 2021; BARRICELLI; FOGLI, 2022; FULLER et al., 2020) have tried elucidating the concept and providing common reference architectures. Industries across different sectors aim to incorporate DTs because of their potential to enhance operational efficiency. Wanasinghe et al. (2020) have done an extensive literature review of DT tools in the O&G industry and highlighted that asset monitoring and lifecycle management is one of its most prominent fields of application. These findings drive our choice of proposing and building an architecture for the oil production forecast.

We fit our work in the five-component architecture proposed by Tao and Zhang (2017). Figure 2.1 shows the five components in action: (1) the physical asset, (2) the virtual asset, (3) the data management layer, (4) service systems, and (5) connections. The data management layer is the centerpiece of the architecture, receiving data from the other components and, in response, driving their execution. The service system component provides models, simulators, and visualizations as encapsulated sub-services. Our proposed architecture, described in Section 4.2, fits as a deployable service in this layer, ingesting sensor data from the data management layer, internally building models, and feeding back predictions.

Figure 2.1 – Digital Twin architecture proposed by Tao and Zhang (2017)



Source: (WANASINGHE et al., 2020)

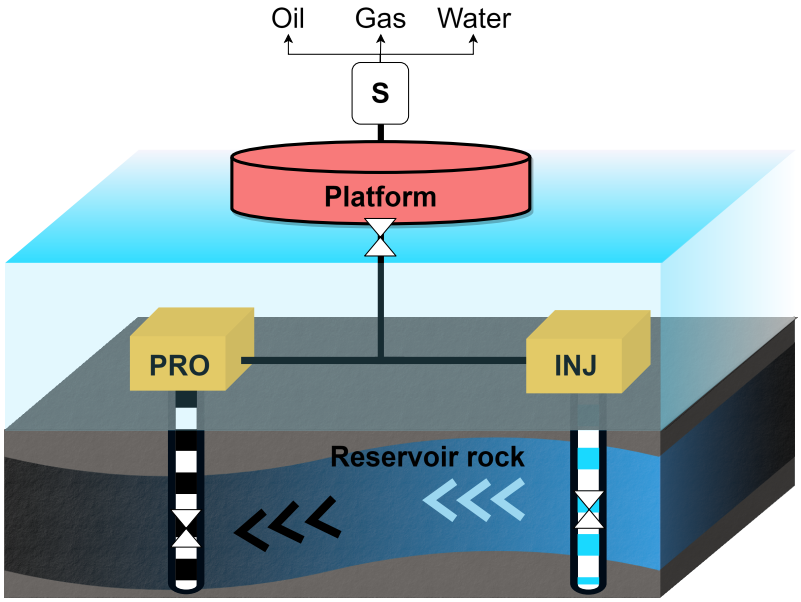
2.2 Oil Production Systems and Libra block

Under a Production Sharing Contract (PSC) in Brazil since 2013, Libra is a block comprising a large and thick oil carbonate reservoir located 180 kilometers off the coast of Rio de Janeiro in Santos Basin Pre-Salt ultra-deep waters with depths ranging from 1,700 m to 2,400 m. Libra has one recognized oilfield. Oilfields are subsurface or surface areas located in or on a unique geological structural feature and stratigraphic condition that contain single or multiple reservoirs (American Petroleum Institute, 1988).

To produce petroleum from these reservoirs, drilling boreholes into the subsurface where hydrocarbons are believed to be present was required. Specialized equipment was

then introduced into these holes for transportation and flow control at the many depths the wells produce from. Flowlines connected these wells to Floating Production Storage and Offloading (FPSO) vessels or other surface equipment. Figure 2.2 shows a simplified offshore oil production system analog to those in Libra. This production system in the figure comprises a producer well and injector well connected by a riser to a platform equipped with a device that separates the multiphase fluid into its parts: oil, gas, water, and solids. At the platform, a flow-restricting valve (also called a choke) can be seen, as well as valves within the wells to control the flow between different production depths, called Inflow Control Valves (ICVs).

Figure 2.2 – Schematic of a generic subsea production system. PRO: Production wellhead; INJ: Injector wellhead; S: Separator; ⧖ Valve



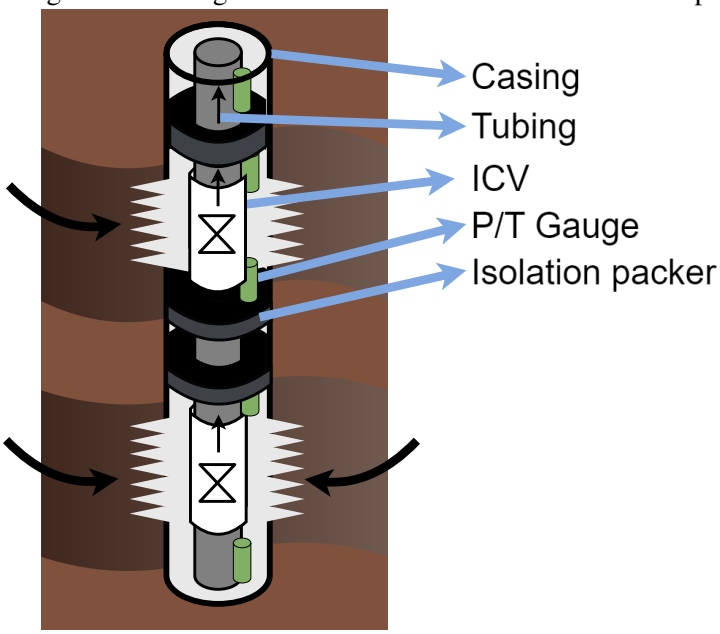
Source: Author

Libra was first explored under an Extended Well Test (EWT) project, mainly in the northwest section. EWT in Libra was part of a de-risking plan for reducing uncertainties in developing unconventional reservoirs. More specifically, it was done to evaluate the productivity and characteristics of the reservoir and fluid within, guiding future development plans. Libra's prospect area is around 1,500 km², the reservoir fluid has a GOR (Gas-Oil Rate) of 440 Sm^3/Sm^3, gravity of 29 °API, H_2S of around 15 ppm, and CO^2 contamination ranging from 40 to 45% in volume. Beneath a big layer of salt, the block can be divided into three production zones at most, with oil columns exceeding 400 m. It comprises structures formed by thick microbial carbonate reservoirs with relatively high heterogeneity from the Barra Velha and Itapema formations. Faults, intrusive and extrusive igneous rocks are also present in the block, which add to its complex structure

(COSTA et al., 2019; MOCZYDLOWER; FIGUEIREDO; PIZARRO, 2019; ANJOS et al., 2020; SILVA et al., 2020; ROVINA et al., 2019; COCCOLI et al., 2019).

Another essential characteristic of Libra is its use of smart wells. Innovations in sensor hardware, data transmission, and remote control systems have led to the adoption of this advanced type of well by many O&G companies (GAO; RAJESWARAN; NAKAGAWA, 2007). Smart wells provide constant access to downhole measurements, such as pressures, temperatures, and rates in different parts of the wells. They also allow remote actuation of downhole valves, such as ICVs (YETEN et al., 2004). All wells in Libra received intelligent completion and were equipped with ICVs and Pressure/Temperature (P/T) gauges. Figure 2.3 shows the essential elements of an intelligent oil well. Note the P/T gauges on different segments of the well and the ICVs that control flow from different production intervals.

Figure 2.3 – Diagram from a smart well and its basic components

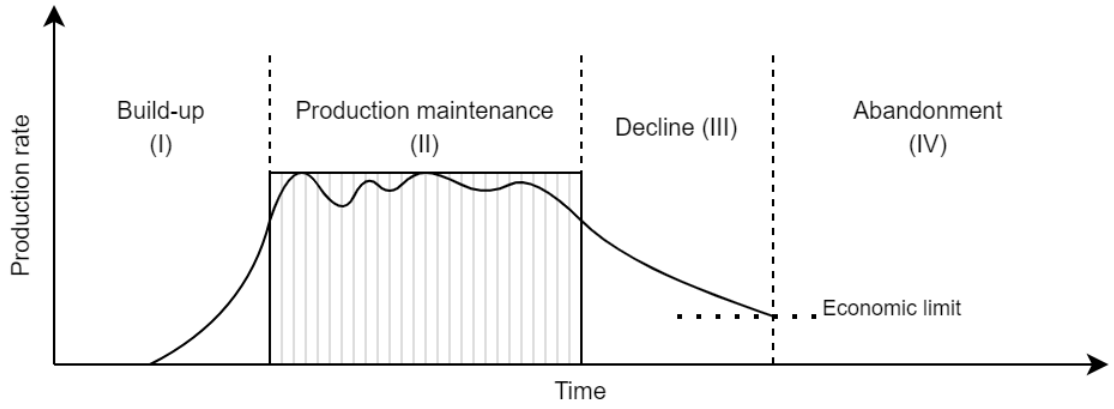


Source: Author

Since we focus on forecasting production volumes, it's essential first to understand how the whole-field production evolves in the long term (months to years) and short term (seconds to days). Figure 2.4 denotes an oilfield's usual production life cycle, with phases I-IV representing the different regimes the oilfield and, consequently, the oil wells go through.

As can be seen, the aggregate production rate of all wells in an oilfield goes through build-up (I), production maintenance (II), decline (III), and abandonment (IV) phases. The initial increase in production potential (I) is due to the digging and operating

Figure 2.4 – Life cycle of an oilfield



Source: Author

of new wells that exploit different reservoir regions. Phases (I) and (II) happen as natural drive mechanisms of the reservoir push the petroleum up to wells and the production facility. Existing pressure from multiple layers above allows for various production mechanisms: dissolved gas drive, gas cap drive, water drive, combination drive, and gravity drainage (CLARK, 1969). Production reaches a plateau in II, dictated by the production capacity of the surface installation. This stage also fluctuates because the reservoir loses its initial pressure, and the natural drives weaken. Additional recovery methods are applied to the reservoir to increase its production and compensate for this. Water and gas can be injected into the rock to increase its pressure. Enhanced Oil Recovery (EOR) can be made by introducing substances into the reservoir that change the physical attributes of the fluid (LAKE, 1989). Stimulating the flow of hydrocarbons to the well can also be done by acidization and fracturing (NNANNA; AJIENKA, 2005), especially in tight and unconventional shale reservoirs. Once additional recovery methods give diminishing returns, production rates decline steadily (III), as little or no intervention is done to improve production. Once the economic limit is reached, that is, the sale of produced hydrocarbons is no longer enough to maintain the facility's operational costs, and the oilfield is abandoned (IV).

All the small oscillations of oil production in the previous graph can be explained by short-term oil production behavior, which is more affected by the dynamics of the oil flow in the pipelines, gas-lift pump actuation, separators, and other surface facilities than by the long-term effect of reservoir physics (AL-JASMI et al., 2013). This is important to know if we want to identify a system that simultaneously considers whole-reservoir (long-term) effects and in or near-well (short-term) effects.

2.3 Estimating petroleum production

Petroleum production is mainly based on mechanistic fluid modeling through the reservoir rock and well pipelines. Darcy (1856) was the first to describe multiphase substance flow through porous media. While well-established, his principles do not accurately represent the flow of hydrocarbons through a reservoir or an oil well (GOVINDARAJAN, 2019). The multiphase and compressible nature of petroleum, as well as the highly heterogeneous and discontinuous nature of subsurface reservoirs, introduce complexities and uncertainties in the modeling of subsurface hydrocarbon flow (HUBBERT, 1940; HUBBERT, 1956; MUSKAT, 1946; BEAR, 1988).

Traditionally, different model-driven methods, and reservoir simulators are commonly employed for simulating full-reservoir long-term dynamics and production estimation at wells. These include *numerical*, *analytical*, and *production decline curve* methods, with numerical models being the most common (LI et al., 2022a). Numerical reservoir models use computationally demanding Finite Difference Methods (FDM) to solve differential equations representing the fluid's physics flowing through the medium. They also commonly go through a tuning process, where their parameters are fitted to reflect the actual behavior of production, a process called history matching, which can be done manually with the current knowledge about the reservoir (MATTAX; DALTON, 1990), or through more automated, but still laborious, optimization methods (RWECHUNGURA; DADASHPOUR; KLEPPE, 2011). Analytical or semi-analytical methods incorporate properties related to flow transport and geophysics in a less computationally expensive way, though they struggle in simulating reservoir heterogeneity (WANG et al., 2019). Analytical methods are also only suitable for simple flow geometries, such as linear or radial flow problems, but often overlook more complex effects like capillary pressure and compressibility (PAL, 2021). The simplest method is fitting a declining curve to extrapolate production rates since wells in conventional reservoirs follow the empirical equations of Arps (1945) because of the loss of reservoir pressure over time.

In addition to whole-reservoir simulation, we can estimate the oil flow in a well by considering the more immediate physical principles that dictate the inflow of multiphase fluid from the reservoir into wells through the pipelines and chokes. The so-called Virtual Flow Meters (VFMs) consider temperatures, pressures, valve opening measurements, and completion data near or at the well and infer production rates using mechanistic models of the fluid inflow from the reservoir, the thermal-hydraulic dynamics of the fluid within

the flowlines of the facility, the fluid properties, and the flow-restricting valve operations (BIKMUKHAMETOV; JÄSCHKE, 2020).

Many algorithms have been proposed to simplify the estimation of petroleum production without relying on complex methods. They go from simple analogy assumptions to what is known as a *proxy model*. Proxy models are mathematical simplifications of physics dictating fluid flow through a porous medium. They are mainly used for uncertainty analysis, risk analysis, and production optimization (SILVA; AVANSI; SCHIOZER, 2020). Out of these proxy models, we can highlight Machine Learning (ML) algorithms, that rely solely on the large amount of field data available to make assumptions. ML algorithms can use this data to infer complex system behavior related to petroleum production through an unrestrained optimization problem (BIKMUKHAMETOV; JÄSCHKE, 2020). This approach uses the same variables to build mechanistic models but assumes no domain knowledge.

Our work focuses on using time series data from the production system to predict oil rates with ML methods. We consider using the system's past data and temporal characteristics to model future transient oil flow phenomena. Estimating future oil rates using past system behavior is expected to align with the previously cited physical principles governing oil flow. It is important to note that our approach of using past data to forecast oil production has been previously documented in the scientific literature and commonly falls into the realm of time series forecasting. The concept of using ML methods for time series forecasting is explained in Section 2.4.

2.4 Time Series Forecasting

Time series forecasting is a type of Time Series Analysis (TSA) method in which past values of a time-labeled event are used to predict (or forecast) future values of this event. Take the finite sequence S, made up of y values observed sequentially at every same frequency, from time $t - l$ to t.

$$S = \{y_{t-l}, \ldots, y_{t-1}, y_t\} \tag{2.1}$$

We would like to know how these values are extrapolated into the future. We can take horizon k of some size, our sampled values in S, and come up with a function that extrapolates $\hat{y}_{t+1}, \hat{y}_{t+2}, \ldots, \hat{y}_{t+k}$ with minimal deviation from the real set. We call that our

forecast function $\hat{y}(k)$. The best $\hat{y}(k)$ for a problem depends on how it minimizes this deviation, measured by a loss function (LEE, 2008). Important concepts involving time series analysis are described in the following sections.

2.4.1 Loss and error functions

A loss function measures the difference between predicted values of a model $\hat{Y}$ and actual values Y. Time series forecasting aims to optimize model parameters and minimize the value of a properly chosen loss function to achieve the best forecasting generalization possible. The choice of the loss function is crucial because it affects the accuracy of the forecast. The loss function should be chosen based on the data distribution and business choices. It should reduce data bias, long-term error accumulation, and the effect of multicollinearity and outlier data (JADON; PATIL; JADON, 2022). A common loss function is the Mean Squared Error (MSE) (Equation 2.2)

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.2}$$

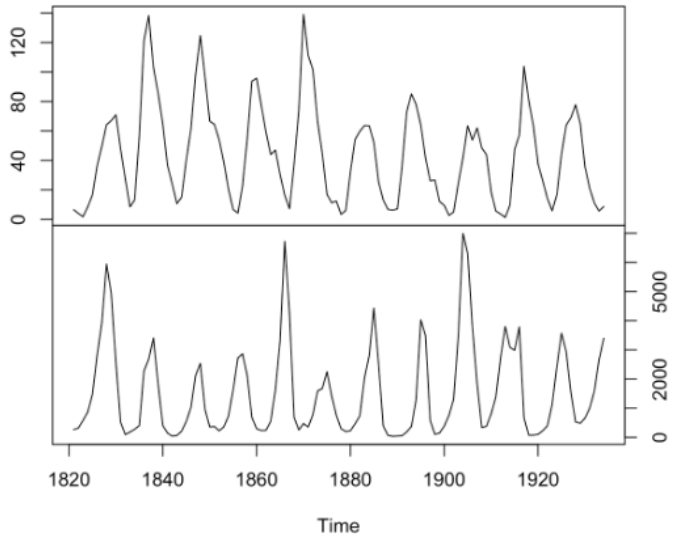
There are also other more comprehensible loss functions such as Root Mean Squared Error (RMSE) (Equation 2.3), which is commonly used to provide error metrics of a forecasting model on an evaluation dataset that was purposefully not used during training.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{2.3}$$

2.4.2 Multivariate time series

In more complex scenarios, where more system variables are available, we can build a multivariate vector $Z_t = (z_{1t}, \ldots, z_{kt})$ of size $k \times 1$ where z_i is a time series of interest (BOX et al., 2015). Multivariate analysis can help us comprehend the interconnections among the series over time and enhance the precision of forecasts by incorporating information from the related series into the forecasting process (BOX et al., 2015). The concept of multi-variability is important in scenarios where multiple system variables, or covariates, exist, like in oil production forecasting.

Figure 2.5 – An example of a predictor time series X and response time series Y



Source: (Holmes E. E.; Scheuerell M. D.; Ward E. J., 2021)

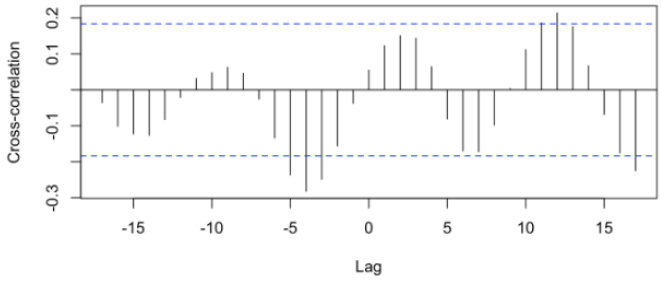
2.4.3 Cross-correlation

Verifying the degree of similarity between series is essential. The cross-correlation formula (Equation 2.4) allows us to determine the effect of l-lagged Y series on X. In simpler terms, this allows us to investigate whether one series precedes or follows another (or itself). It also allows us to measure the effect of every lag l of Y on the response X.

$$r_k(X,Y) = \frac{\sum((Y_t - \bar{Y})(X_{t-k} - \bar{X}))}{[\sum(Y_t - \bar{Y})^2]^{1/2}[\sum(X_t - \bar{X})^2]^{1/2}} \tag{2.4}$$

Cross-correlation is a valuable tool for feature selection, as it can give insight into possible predictors and their effect on the response variable. Figure 2.5 shows two time series. Consider the top one as X and the bottom as Y. From the cross-correlation graph in Figure 2.6, it's possible to see a significant negative correlation, for example, at lags of -3 to -5. The value of X in lags -3 to -5 negatively correlates to the value of Y in lag 0.

Figure 2.6 – Cross-Validation values between X and Y for each lag k of X



Source: (Holmes E. E.; Scheuerell M. D.; Ward E. J., 2021)

2.4.4 Stationarity

One of the most essential properties of a time series is stationary. When the underlying distribution function that generated a series does not vary with time, we call the series stationary. By consequence, a stationary series has a (I) mean $\mu_X(t)$ independent of t and (II) auto-covariance $\frac{\sum((X_t-\bar{X})(Y_{t-l}-\bar{Y}))}{n-1}$ independent of t for every l (BROCKWELL; DAVIS, 2002). In simpler terms, a stationary series behaves the same statistically across different time periods. Many classical time series analyses assume the input time series has constant mean and variance over time. Oil production and its related covariates are known to be non-stationarity (NING; KAZEMI; TAHMASEBI, 2022; SAGHEER; KOTB, 2019), posing a significant challenge for classical TSA, which assumes the stationarity of the time series. We can apply differencing to a time series to make it stationary, but this process is not always straightforward, especially in complex systems. Differencing can also negatively affect the model's capability of modeling specific temporal dependencies because the joint distribution changes which happen over time are lost in the process (LIU et al., 2023).

2.4.5 Granger Causality

Granger's causality (GRANGER, 1969) is a hypothesis test determining whether one time series can predict another. In a given context considering two time series, X and Y, if X passes the Granger causality test for causing Y, or in other words, rejects the null-hypothesis of non-Granger causality, it means the observations contained in X decrease prediction errors beyond the case where only the past values of Y are used to predict itself. If we consider the Vector Autoregressive Model (VAR) of a time series (Equation 2.5), where l_{max} is the maximum lookback length we want to consider. X causes Y if any coefficient b_l is not 0, rejecting the null hipothesis. Granger causality is a useful feature selection tool, as we can use it to disconsider variables that are proven not to cause the response variable.

$$\mathbf{Y}_t = \sum_{l=1}^{l_{\max}} a_l \mathbf{Y}_{t-l} + \sum_{l=1}^{l_{\max}} b_l \mathbf{X}_{t-l} + \eta_t \tag{2.5}$$

2.4.6 Prediction methods

There are roughly two categories for how forecasting models can output a multi-step prediction of size h (STEFANI, 2022):

1. Single output models work by iterating or concatenating the results of regressors with an output size of one and can be further divided into:

   (a) Iterative strategy learns a one-step model $f(X) = Y_{t+1}$ to predict the next value and uses it recursively h times to output the entire horizon.

   (b) Direct strategy uses a separate model to predict each element in the horizon directly. That is, it trains h models and concatenates their output, which are, respectively, $f_1(X) = Y_{t+1}, f_2(X) = Y_{t+2}, \ldots, f_h(X) = Y_{t+h}$.

2. Multiple output models use a single model that returns a vector of future values at once; this can be represented by $f(X) = (Y_{t+1}, Y_{t+2}, \ldots, Y_{t+h})$.

Iterative methods are the most straightforward to implement and train, as they only require one model for any forecasting horizon. However, they are prone to error accumulation and propagation, as any prediction error in the previous step will affect the subsequent predictions (BONTEMPI, 2008). Direct methods are generally more robust to model misspecification and error accumulation compared to iterative methods (MARCELLINO; STOCK; WATSON, 2006). Nonetheless, they do not consider dependency between future values since none of the h models are trained with all of them (BONTEMPI, 2008). The multi-output strategy, which has been applied to several real-world applications (BONTEMPI, 2008), fixes the direct approach by considering the dependencies between future values, although its significant variance and low bias present a challenge.

2.4.7 Machine Learning Methods

The traditional or *classical* methods in the field of TSA are based on statistical inference to achieve this goal. The most prominent ones are Autoregressive Integrated Moving Average (ARIMA), Seasonal Autoregressive Integrated Moving-Average (SARIMA), Vector Autoregression (VAR), and Exponential Smoothing (ES). These approaches have

been studied and used in econometrics for many years and usually provide a good baseline for more complex methods. They have disadvantages, including the aforementioned stationary assumption described in Section 2.4.4.

Aside from classical TSA techniques, a new contender for time series forecasting has appeared in recent years: Machine Learning. ML falls under the Artificial Intelligence (AI) umbrella, enabling systems to improve their performance through experience, essentially learning from data (GOODFELLOW; BENGIO; COURVILLE, 2016). When labeled data is used for learning, that is, all features of the data points along with their ground truth label are known, we call it supervised learning. Supervised learning in ML aims to discover a function that maps input data to corresponding output labels. The training process involves iteratively adjusting the model's parameters to minimize a predefined cost function, ultimately optimizing the model's performance. In recent years, ML models have increased in popularity. Specifically, Neural Networks (NN) have been demonstrated to be the go-to method for ML-based forecasting in the O&G field, as suggested by our literature review (Section 3).

In a ML sense, time series forecasting is often treated as a regression problem. That is, to predict a continuous numerical output based on a set of input features. ML algorithms such as Gradient-boosted trees, NNs, and Transformer-based architectures rely on a training phase where samples are fed into the model, and its parameters, or weights, are tuned using some optimization procedure like Gradient Descent. In summary, ML model training adheres to the steps described in Figure 2.7 and is explained in the subsequent text.

1. Choose dataset

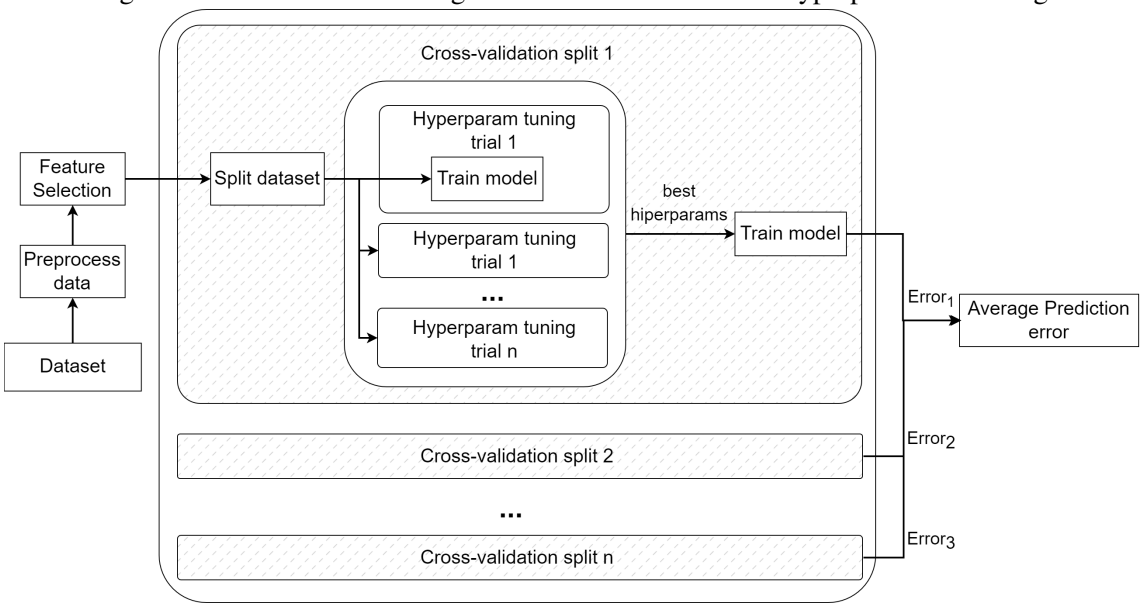
   A proper dataset is chosen and downloaded from a data source, such as a database or static file.

2. Preprocess data

   Domain knowledge and insights from the Exploratory Data Analysis (EDA) phase are used to propose data preprocessing techniques. Those usually act upon filtering out outliers and correctly filling gaps in the data. Common preprocessing techniques involve removing outliers by a defined threshold or distance from the mean. Filling gaps in the data is usually done by filling with the last known value, linear interpolation, or training an ML model that can estimate missing features.

3. Feature selection

Figure 2.7 – ML model training with Cross-Validation and hyperparameter tuning



Source: Author

Feature selection involves choosing a subset of pertinent features from a broader dataset. It aims to decrease data collinearity and dimensionality, speed up model training, and enhance the ML model's performance (BACCIU, 2016). There are roughly three ways of selecting features: wrapper, filter, and embedded (JIMÉNEZ et al., 2020). The wrapper method trains multiple models with a subset of features until an optimal subset is found. Filter methods involve using some metric to filter out irrelevant features before model training. These metrics can involve statistical tests like Pearson's correlation, cross-correlation, Granger Causality, Chi-Square, etc. Since filter methods do not require training multiple models, they are typically the fastest to use. Examples of filtering in time series context (SUN et al., 2015; CUKUR et al., 2015; WANG; LI; QIN, 2013; SAIKHU; ARIFIN; FATICHAH, 2019) proved the usefulness of a Granger causality and correlation-based feature selectors. Finally, the embedded strategy performs a combination of both wrapper and filter methods.

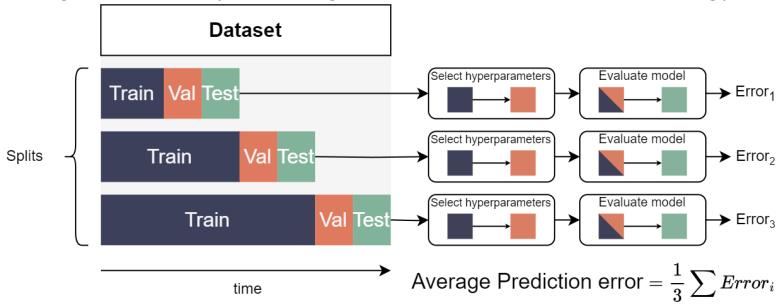
4. Cross-validation

Cross-validation (CV) is an evaluation technique where models are trained on different subsets of the dataset. It's often used to evaluate if a technique (LR, XGBoost, NN, etc.) will generalize well to an independent dataset. A common CV technique is called k-fold, where k equally sized sets (folds) are extracted from the dataset. In each split, $k - 1$ of these folds are used to train a model, and the remain-

ing fold is used for validation. CV output is an average prediction error calculated by taking the mean of the error of all splits. This average error allows us to assess the robustness of a specific ML technique on varying sets of data instead of the whole dataset.

K-fold CV works well for most regression and classification tasks but does not preserve the sequence of samples since the training folds can come after the test fold. Time series are a unique kind of data because they are often not independent over time and may exhibit autocorrelation. Additionally, the distribution of a time series may change over time, making it non-identically distributed (non-i.i.d). This nonadhesion to independent and identically distributed (i.i.d) asks for a unique CV technique, which preserves the sequential nature of the data. Day-chaining Nested CV, based on the expanding origin strategy (TASHMAN, 2000), is a type of CV that can adequately handle time series data. Figure 2.8 exemplifies a 3-split Day-chaining Nested CV, where the dataset is divided multiple times, with each division incorporating a progressively larger portion of the data. As previously noted, the training fold precedes the validation and test fold. The prediction error of each division is evaluated on the test set, while hyperparameters are selected based on a previous step utilizing the validation set. Finally, the average prediction error is calculated to assess the technique's prediction error.

Figure 2.8 – Day-chaining Nested Cross-Validation Strategy



Source: Author

5. Split dataset

   The dataset is split into training, validation, and test (or holdout) sets. Samples from the training dataset are used in model training. The validation set evaluates training performance on unseen data, and the test set provides the final metric of model performance.

6. Train model

(a) Tune parameters with training set

Samples from the training dataset, that is, a set $\{(X_i, Y_i)\}_{i=1}^{N}$ with N amount of members, where X are features and Y are response values, are effectively used for tuning the model parameters. Samples are fed in batches to the model, and a training loss is generated after each batch of samples is processed, and Gradient Descent tunes the parameters.

(b) Validate model on the evaluation set

For every model epoch, when the model has seen all training data, the validation set loss is also calculated. Validation samples are not used during training, so its loss measures the models' performance on an unknown data set. Validation loss is used in tandem with training loss for checking model overfitting and underfitting, as well as a good metric to determine when a model should stop training (early stopping parameter).

(c) Evaluate model on the test set

Finally, when training is finalized, the model is evaluated on a holdout or test set, giving us the final performance metric.

7. Tune hyperparameters

Hyperparameters are configurations set before model training, as opposed to model parameters (or weights) tuned and decided after model training. Hyperparameters describe learning rate, number of NN layers, number of units, depth of decision trees, etc. They can be (I) arbitrarily chosen, (II) found through extensive grid-search search, or (III) found through some heuristic procedure that best guides them to an optimal configuration. When a search algorithm is used, model training is run n times, with each n constituting a "trial" until an ideal set of hyperparameters is found. The heuristic procedure is usually favored because it can reasonably converge to the set of optimal hyperparameter candidates and abandon training when a hyperparameter configuration does not show promising results. Common heuristic methods for sampling favorable hyperparameter values utilize Bayesian optimization. Tree-Structured Parzen Estimator (TPE) (WATANABE, 2023), for example, at each trial, decides the values of the following parameters by maximizing the ratio between two Gaussian Mixture Models (GMMs), $l(x)$ and $g(x)$. $l(x)$ corresponds to the parameter values linked with the best objective results, while the second GMM, $g(x)$, is associated with the remaining parameter values. In addition to a sampling

algorithm, heuristic procedures can “prune” unpromising trials. For example, we can trim a trial if its best intermediate result falls below the median from prior trials at the same step.

The following sections describe the ML methods used in this work. We decided to include candidates from multiple groups: regression-based (LR), tree-based (XGBoost), RNN-based (LSTM), and Transformer-based (TFT).

2.4.7.1 Linear Regression

Linear regression (LR), usually generalized to Multiple Linear Regression (MLR) (NETER; WASSERMAN; KUTNER, 1983), is a statistical procedure for calculating the value of a dependent variable from one or more independent variables. The coefficients of a linear predictor function that describes the linear relationship between two or more variables are estimated from data to minimize the error ε. An LR model with an arbitrary number of k regressors has the notation:

$$Y_i = \beta_0 + \sum_{j=1}^{k} \beta_j X_{ij} + \varepsilon_i, \tag{2.6}$$

where Y_i is the i^{th} observation of the dependent variable, β_j is the regression coefficient for the j^{th} regressor, X_{ij} is the i^{th} observation of the j^{th} regressor and ε_i is the i^{th} error term. Fitting the coefficients $\hat{\beta}$ of the LR model can be done through the Ordinary Least Squares, Maximum-Likelihood estimation, Moore-Penrose pseudoinverse, and Gradient Descent:

$$\hat{\beta} = \left(X'X\right)^{-1} X'Y \tag{2.7}$$

To use LR for time series forecasting, every past and future known predictor from $t-l$ to $t+k$, where l is the lookback window and k is the forecast size becomes an element in the regressor matrix X.

2.4.7.2 XGBoost

XGBoost (CHEN; GUESTRIN, 2016) is a type of ensemble ML technique that involves boosting decision trees based on parameter tuning and error reduction through the Gradient Descent algorithm. The intuition behind ensemble learning is that combining multiple models can help reduce the individual models’ variance and bias, leading

to better overall performance. Ensemble learning can be used for both classification and regression tasks. XGBoost works on an ensemble of decision trees, a tree-like structure in which each internal node represents a decision based on a particular feature, and each leaf node corresponds to the predicted outcome or classification. Decision trees are constructed by decision tree learning algorithms that identify the tree structure that can make suitable decisions for a dataset using different criteria (for example, information gain). It is one of the most widely used, practical, and fast methods for supervised learning.

XGBoost minimizes a regularized objective function (Equation 2.8) that combines a convex loss function L measuring the dissimilarity between predicted and target outputs, with a penalty term Ω addressing the complexity of the model, specifically the regression tree functions.

$$L = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{2.8}$$

The training process of XGBoost unfolds iteratively, introducing new trees that predict the residuals or errors of previous trees, and optimize the objective function further. Take x as the input features, α as the learning rate, g and h as the gradient and hessians respectively of loss function L. For every m weak learner in M, we update the model $\hat{f}$ in the following way:

$$\begin{aligned} \hat{\phi}_m &= \arg\min_{\phi \in \Phi} \sum_{i=1}^{N} \frac{1}{2} \hat{h}_m(x_i) \left[\phi(x_i) - \frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)} \right]^2 \\ \hat{f}_m(x) &= \alpha \hat{\phi}_m(x) \\ \hat{f}_{(m)}(x) &= \hat{f}_{(m-1)}(x) + \hat{f}_m(x) \end{aligned} \tag{2.9}$$

Since XGBoost uses weak learners sequentially trained to correct prior models' predictions, it reduces prediction bias.

2.4.7.3 Long Short-Term Network

Long Short-Term Memory (LSTM) (HOCHREITER; SCHMIDHUBER, 1997) networks are a type of Recurrent Neural Network (RNN) architecture specifically tailored to excel at learning long-term dependencies in sequential data. Unlike standard RNNs, LSTM's architecture is projected to solve the vanishing gradient problem of deep NNs. This problem arises because gradients tend to become very small as they backpropagate through the network during training, making it challenging to learn long-term dependen-

cies. LSTMs have successfully been used in machine translation, speech recognition, and time series forecasting.

The main element within an LSTM cell is the cell state C, which propagates forwards for every cell. It is depicted as a horizontal line on the top of the diagram in Figure 2.9. The cell state determines the prediction (h_t) of that cell after linear operations with the outputs of the forget, input, control, and output gates. The predicted value is determined after the following operations in the LSTM cell:

1. The initial stage in an LSTM cell involves determining which information to discard from the cell state. The "forget gate" f_t (Equation 2.10) applies a sigmoid function to the previous prediction h_{t-1}, the current input x_t, weights W_f and bias terms b_f, outputs values between 0 or 1.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{2.10}$$

2. The "input gate" i_t (Equation 2.11), also a sigmoid layer, identifies and outputs the values to be updated.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{2.11}$$

3. The "control gate" g_t (Equation 2.12) a hyperbolic tangent layer, formulates a vector of candidate values to modify the cell state.

$$g_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{2.12}$$

4. The following step sets the new cell state C_t to a pointwise multiplication of f_t and the old cell state C_{t-1} plus the pointwise multiplication of i_t and g_t (Equation 2.13).

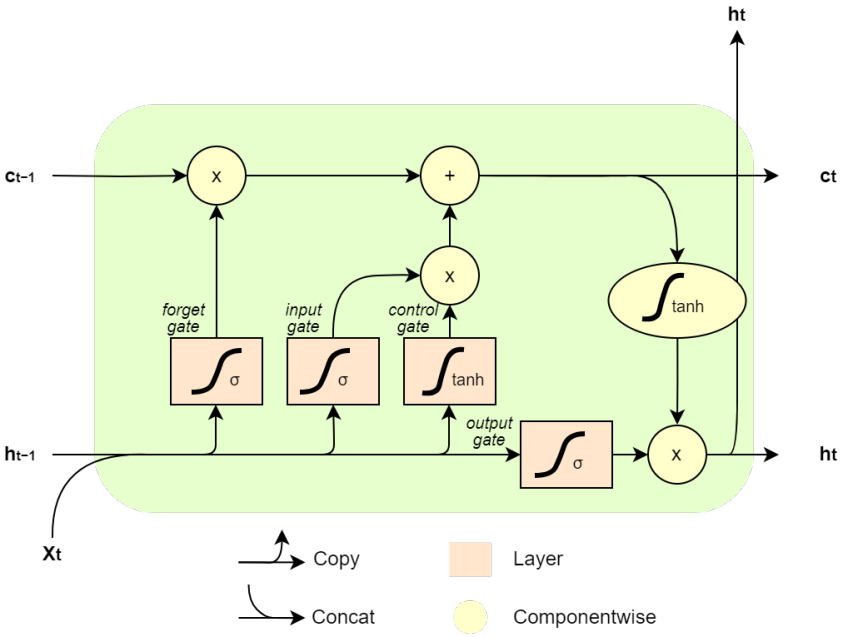
$$C_t = f_t \times C_{t-1} + i_t \times g_t \tag{2.13}$$

5. Finally, we set the predicted value h_t (Equation 2.15) to a pointwise multiplication of o_t, the "output gate" (Equation 2.14), and C_t.

$$o_t = \sigma\left(W_o [h_{t-1}, x_t] + b_o\right) \tag{2.14}$$

$$h_t = o_t \times \tanh\left(C_t\right) \tag{2.15}$$

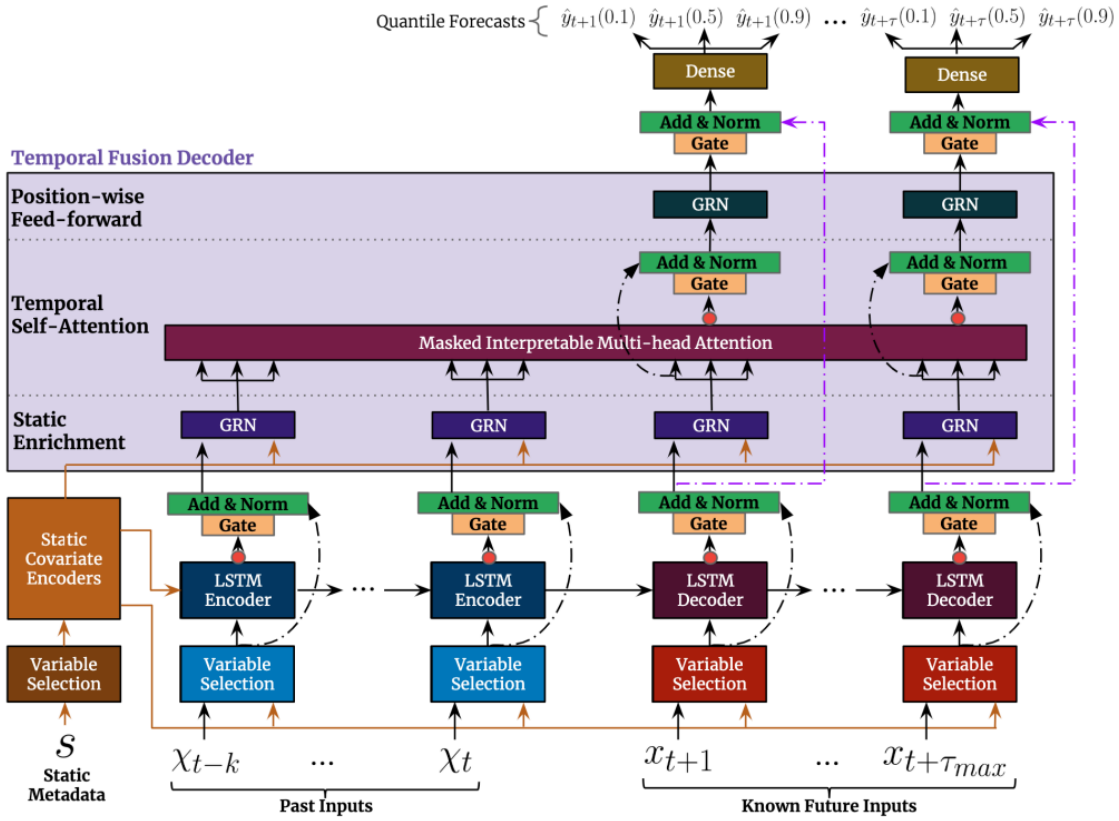
Figure 2.9 – Diagram for an LSTM cell



Source: Author

In this work, we use a fully recurrent LSTM architecture, where the target value h_t is set to the prediction of the previous cell. A prediction with a forecasting horizon of n is formulated through n iterations of RNNModel predictions, necessitating the availability of n future covariates.

Figure 2.10 – Temporal Fusion Transformer Architecture



Source: (LIM et al., 2019)

2.4.7.4 Temporal Fusion Transformers

Temporal Fusion Transformer (TFT) (LIM et al., 2019) is a recently proposed attention-based deep NN for multi-horizon forecasting that claims to achieve high performance and interpretability. TFT implements an encoder-encoder architecture with several components designed to handle the heterogeneity and complexity of multi-horizon forecasting inputs, such as static covariates and known future inputs. TFTs use gating mechanisms to skip over parts of the network when deemed fit. It also uses variable selection networks to select relevant input variables at each time step and suppress noisy features. A sequence-to-sequence layer is employed to enhance the locality of temporal features, and a self-attention layer captures long-term dependencies across different time steps. The whole TFT architecture can be seen in Figure 2.10. Its most important components are:

1. Gated Residual Network (GRN)

   An essential component of TFT is its gating mechanism, called the Gated Residual Network, represented in Equation 2.16. The GRN's main task is to suppress unnecessary components of the architecture. Take a as the primary input, c as the optional

context vector, ω as the index to denote weight sharing, W and b as the weights and biases, σ as the sigmoid function, and ELU as the exponential linear unit. The GRN takes the input and optional context vector and applies a nonlinear transformation to them, followed by a Gated Linear Unit (GLU) based gating mechanism that controls how much of the transformed input is added to the original input. The GLU controls how much the nonlinear contributions are added to a, meaning it could skip them altogether. A standard layer normalization operation then normalizes the output of the GRN.

$$\begin{aligned} \text{GRN}\omega(a, c) &= \text{LayerNorm}(a + \text{GLU}\omega(\eta_1)) \\ \eta_1 &= W_{1,\omega}\eta_2 + b_{1,\omega}1 \\ \eta_2 &= \text{ELU}(W_{2,\omega}a + W_{3,\omega}c + b_{2,\omega}) \end{aligned} \tag{2.16}$$

2. Variable Selection Network

   This component consists of the variable selection strategy for static, past, and future inputs, using GRNs and softmax layers to generate variable selection weights. These weights are then used to scale the input features before feeding them into the temporal processing component. Equation 2.17 shows how the variable selection network works. $\boldsymbol{v}_{\chi_t}$, a vector of variable selection weights, $\tilde{\boldsymbol{\xi}}_t^{(j)}$, the processed feature vector for variable j are combined into $\tilde{\boldsymbol{\xi}}_t$, the weighted vector of features.

$$\begin{aligned} \boldsymbol{v}_{\chi_t} &= \text{Softmax}\left(\text{GRN}_{v_\chi}\left(\boldsymbol{\Xi}_t, \boldsymbol{c}_s\right)\right) \\ \tilde{\boldsymbol{\xi}}_t^{(j)} &= \text{GRN}_{\tilde{\xi}(j)}\left(\boldsymbol{\xi}_t^{(j)}\right) \\ \tilde{\boldsymbol{\xi}}_t &= \sum_{j=1}^{m_\chi} v_{\chi_t}^{(j)} \tilde{\boldsymbol{\xi}}_t^{(j)} \end{aligned} \tag{2.17}$$

3. Temporal Fusion Decoder

   The Temporal Fusion Decoder layer uses a series of components to learn the temporal relationship of the data (Equation 2.18). First, an LSTM encoder-decoder structure is used to extract local patterns. This step generates a set of temporal futures used in the next step of this layer. If static covariates are used, an additional enrichment step is executed based on GRN. Next, the enriched features are fed into an Interpretable Multi-Head Attention mechanism, a modified version of the standard multi-head attention used in Transformer models for learning long-term relationships. Multi-head attention employs different heads for different represen-

tations of subspaces. The creators of TFT made it so the same value vectors were shared across all attention heads instead of using different values for each head. They also made it so that the attention outputs from other heads were aggregated by taking their average. These two changes make it easier to interpret the importance of each input feature based on the attention weights.

$$
\begin{aligned}
&\text{InterpretableMultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \tilde{\boldsymbol{H}} \boldsymbol{W}_H \\
\tilde{\boldsymbol{H}} &= \tilde{A}(\boldsymbol{Q}, \boldsymbol{K}) \boldsymbol{V} \boldsymbol{W}_V \\
&= \left\{ 1/H \sum_{h=1}^{m_H} A\left(\boldsymbol{Q}\boldsymbol{W}_Q^{(h)}, \boldsymbol{K}\boldsymbol{W}_K^{(h)}\right) \right\} \boldsymbol{V}\boldsymbol{W}_V \\
&= 1/H \sum_{h=1}^{m_H} \text{Attention}\left(\boldsymbol{Q}\boldsymbol{W}_Q^{(h)}, \boldsymbol{K}\boldsymbol{W}_K^{(h)}, \boldsymbol{V}\boldsymbol{W}_V\right)
\end{aligned}
\tag{2.18}
$$

4. Output

TFTs generate prediction intervals using quantile regression, which outputs multiple percentiles at each horizon. It also enables three interpretability use cases: identifying globally important variables, visualizing persistent temporal patterns, and detecting significant events or regimes.

3 RELATED WORKS

In recent years, there has been a surge in the utilization of data-driven methods for oil production forecasting, considering oil production system variables. This section reviews papers that use past temporal data and state-of-the-art machine learning methods to predict oil flow and better model transient phenomena. We include works that measure the effectiveness of tree-based regressors, classic Feed-Forward Neural Networks (FFNN), RNNs, and Transformer-based methods. While classical time series forecasting and curve-fitting methods have been used for extrapolating oil production, such as Holt-winters, ARIMA, Decline Curve Analysis (DCA), and VAR, we only consider ML methods for this work.

Yan et al. (2019) described a workflow and application for training an XGBoost model that can predict the oil production of wells based on historical data and optimize production values by recommending future steam allocation plans. The authors train one global model to forecast the production of multiple wells. The paper demonstrates that the XGboost model outperforms the naive baseline. The paper also demonstrates the effectiveness of using steam allocation plan as a future covariate (3% improvement).

Aizenberg et al. (2016) present a method for long-term forecasting of oil production using a multilayer Neural Network with Multi-Valued Neurons (MLMVN). MLMVN is a complex-valued NN that can approximate nonlinear functions without using derivatives. The authors apply MLMVN to a real-world data set of monthly oil production from 14 wells in the Gulf of Mexico and compare it with other methods such as ES, ARIMA, Elman, and RNNs. The production data was used to predict itself without additional covariates or future known values. The authors show that MLMVN can achieve better accuracy and stability for univariate and multivariate, one-step and multi-step ahead prediction. They also highlight the usefulness of predicting local variations instead of only a smooth curve projection.

Werneck et al. (2022) compare off-the-shelf data-driven approaches to perform 30-day production and pressure forecasting in wells in a carbonate reservoir. Experiments were done using real and synthetic data from a reservoir model, with varying numbers of pre-processing techniques and injection data. The authors could not determine the usefulness of using injection data as input, which may happen due to geological reasons like lack of connectivity between injector and producer well, operational reasons such as sub-optimal injection strategy, or modeling flaws in the data or the model itself. Compared

to Convolutional Neural Networks (CNNs), classic Transformers, N-beats, and Prophet, the best results were achieved using stacked RNN layers to consider long time-frames as input, such as LSTM and GRU.

LSTM-based NNs are widely used in time series forecast tasks due to the capability of capturing dependencies in sequential data and minimizing the vanishing gradient problem of traditional RNNs. By leveraging this capability, LSTMs can model intricate temporal patterns, such as oil production's inherently sequential nature.

Li et al. (2022b) proposed using LSTM and sliding window prediction to predict well flow rate considering manual operations of choke valve and operational time. The trained LSTM model showed better prediction accuracy when considering the manual operation variables. Horizon and lookback size of 3 days for the model showed better prediction accuracy. Using LSTM with a limited lookback length might be a limitation of this approach compared to a model with an attention layer that can absorb long-term behavior better.

Andrianov (2018) uses LSTM and sliding windows to estimate and forecast the multiphase flow rates of oil, gas, and water in O&G production systems. The paper shows that the LSTM model outperforms the FFNN, encoder-decoder-based LSTM, and numerical model in terms of accuracy and forecasting capability in a synthetic case of flow slugging and a variable rate well test. A sampling frequency of seconds and minutes was adopted by the authors, with an input and output window size of ≈ 100-200 seconds, without the use of future known variables. The authors highlight the importance of choosing the correct number of pressure and temperature variables that serve as input. They also point out that their LSTM architecture model is limited to output size = input size.

Razak et al. (2021) proposed an LSTM-based technique for long-term production forecasting of monthly rates in unconventional reservoirs using transfer learning. Completion parameters, geological, fluid, control variables, and past production data were used as input for models exposed to various flow regimes and shale plays. Although the results using transfer learning were promising, this technique requires data from other sources with similar operational conditions and geological and physical conditions of the reservoirs.

Wang et al. (2021) present a method for point-predicting daily flow rate in oilfields using an LSTM with downhole temperature and pressure data. The developed model has been validated with two production wells in the Volve Oilfield, North Sea. Results demonstrate that the technique is applicable for flow rate prediction in oilfields.

LSTM performs better for flow rate prediction when compared to other six ML methods, including Support Vector Machine (SVM), LR, tree-based methods, Gaussian Process Regression (GPR), and Bi-LSTM. The performance of the LSTM network is improved by incorporating a dropout layer. Achieving the best prediction results involves adjusting the number of LSTM layers and hidden units, yet increasing these components also prolongs training and prediction times. It's worth noting that a higher number of LSTM layers and hidden units may render the LSTM model unstable and hinder convergence. When comparing prediction accuracies, utilizing both downhole pressure and temperature data simultaneously as input parameters yields superior results compared to using only one at a time.

Loh, Omrani and Linden (2018) propose a method to predict 10-min frequency gas production from mature gas wells using a modified deep LSTM model and update flow rate predictions using Ensemble Kalman Filter (EnKF). The authors tested their approach on real data from two wells in the North Sea and compared it with a baseline approach without EnKF. The results show that the EnKF updated model can improve the accuracy and robustness of the predictions, especially for wells with complex end-of-life behavior and salt precipitation issues. The paper also discusses the challenges and benefits of applying deep learning and data assimilation methods to the O&G industry.

Omrani et al. (2019) compared first-principle physics-based approaches, DCA, deep learning, and hybrid models for 15-minute and daily frequency oil production. The results indicate that each production forecasting model's effectiveness depends on the complexity of the production behavior, the forecasting horizon, and the availability and accuracy of the data used. The performance of both hybrid and physical models was found to be dependent on the quality of the calibration (history matching) of the models employed. The study also found that deep learning models were more accurate in capturing the dynamic effects observed during production, especially for mature fields with frequent shut-ins and interventions. Moreover, hybrid models could better predict long-term production due to considering long-term reservoir depletion behavior given by the first-principle physics model.

Sagheer and Kotb (2019) propose an LSTM network to model the monthly time series data of petroleum production from two different oil fields. The method is compared with other models such as Nonlinear Extension for Linear Arps (NEA), ARIMA, Higher Order Neural Networks (HONN), and different RNN architectures. The results show that the proposed model outperforms the others, and stacking more LSTM layers ensures

better accuracy when long interval time series are used. A limitation of this work is that no other covariate was used except for the oil production itself.

Abdrakhmanov et al. (2021) proposed a novel approach to data-driven modeling of daily transient production of oil wells in the Volve field using the standard transformer encoder-decoder architecture. They demonstrate that transformers can capture transient dependencies and outperform RNNs in forecasting the bottom hole pressure and flow rate values. The authors found that training a "global model" (a model trained with data from more than one well) significantly improves the quality of predictions. Transfer learning was used to transfer the pre-trained weight of a well model to another. However, only bottom hole pressure was analyzed in this case.

Al-Ali and Horne (2023b) used a TFT model on daily sensor data from a Norwegian oil field called Volve. The model considers past data from various sensors and valves as input features and predicts the probabilistic oil production rate. The model is evaluated against a recurrent-based model named BlockRNN and proves superior in accuracy and uncertainty estimation.

More broad comparisons between deep-learning techniques have been made in the literature. Thavarajah et al. (2022) compared three deep-learning approaches: Mixed Input Forecaster (MIF), DeepAR, and TFT. The objective was to forecast ten months of multiphase rates of 166 wells in an unconventional reservoir. Lookback size between 1 and 15 months yielded better results than a naive baseline. The authors conclude that these deep-learning frameworks can capture dependencies between different flow phases, estimate confidence intervals, and extrapolate results for a new well with unique geology spacing and completion characteristics. MIF model performs best according to MAE and $R\hat{2}$, especially on short histories; TFT has the best mean MASE and is better for longer histories when compared to MIF; DeepAR has the most conservative confidence intervals. While future-known operational variables are possible inputs to these models, the authors only used static variables representing geology, spacing, and completion characteristics.

Alali and Horne (2023) compared N-BEATS to TFT, Block-RNN, and ARIMA to predict about 500 days of production. While TFT can model nonstationary behavior efficiently and output probabilistic forecasts, N-BEATS has beaten all the models when predicting the daily production of 2 oil wells. A similar methodology was applied in (AL-ALI; HORNE, 2023a), comparing LSTM and a BlockRNN pre-trained on the M4 time series competition multi-domain dataset. The N-BEATS model, which required no extensive hyperparameter tuning, outperformed BlockRNN.

Table 3.1 summarizes all the studies discussed in this Section, including the methodology, model training parameters, and results. As can be seen, most papers that include LSTMs, except for (AL-ALI; HORNE, 2023a), state that they beat methods such as CNNs, RNNs, sequence-to-sequence models, DeepAR, N-BEATS, classic Transformer architecture, Prophet, SVM, LR, GPR, Bi-LSTM, ARIMA and HONN. Al-Ali and Horne (2023a) controversially showed that a pre-trained N-BEATS architecture could outperform an LSTM network in the task, although in a reasonably limited scenario (daily frequency and model output size of 1). Thavarajah et al. (2022) show TFTs' competitiveness in forecasting monthly production compared to MIF and DeepAR, mainly when longer lookbacks are used. Al-Ali and Horne (2023b) state the superiority of TFTs over LSTMs in a daily scenario.

Notably, most works don't consider a wide range of lookback sizes, nor do they adequately describe the choices of window sizes or procedures used to choose them, except for Thavarajah et al. (2022), Sagheer and Kotb (2019) and Wang et al. (2021), who laid out cos between different lookback sizes. For horizon sizes, most papers fixed an arbitrary value. Only Andrianov (2018) and Omrani et al. (2019) used sampling frequencies lower than one day, 1s and 1min, and 15min, respectively. Moreover, only about half the papers consider future covariates when training the forecasting models. Of those, only Li et al. (2022b), Loh, Omrani and Linden (2018), Thavarajah et al. (2022) and Yan et al. (2019) use plannable features such as choke opening or amount of steam injection. While the papers mentioned earlier use 1-day, 10-min, and 1-min sampling frequencies, none make any comparisons with models trained on all these sampling frequencies on the same dataset.

In summary, there appears to be no research examining the impact of various ML approaches, different window sizes, operational variables, and sampling frequencies for the same set of data, which are the gaps in the literature we investigate with our model selection methodology in Section 4.1.

Table 3.1 – Analysis of selected papers

Authors	Techniques	Freq	Lookback	Output	Strategy	Horizon	Past vars	Future Vars
Abdrakhmanov et al. (2021)	Transformer, RNN, LSTM, GRU	1d	14	1		1d	daily measurements of oil, gas, and water flow rates, bottom hole temperature, and choke size in percentage	
Aizenberg et al. (2016)	MLMVN, ES, ARIMA, Elman, NARX	1m	60 & 120	1	S	60m	GOR	
Alali and Horne (2023)	ARIMA, BlockRNN, TFT, N-BEATS	1d		1-9	M/S	1-500	oil rate, bottom hole pressure, wellhead pressure, wellhead temperature, water rate, choke size	wellhead pressure
Al-Ali and Horne (2023a)	LSTM, Pre-trained N-BEATS, BlockRNN	1d		1	M	1	Bottom hole pressure data used in LSTM; N-BEATS did not require specific past variables	
Al-Ali and Horne (2023b)	TFT, BlockRNN	1d					Oil rate, bottom hole pressure, wellhead pressure, wellhead temperature, water rate, choke size	wellhead pressure
Andrianov (2018)	LSTM	1s/1min	187/122	187/122	M		Temperature and pressure readings	
Li et al. (2022b)	LSTM	1d	1-7		S		Choke size, daily opening time, and production data of previous days	Choke size, production time
Loh, Omrani and Linden (2018)	LSTM-EnKF	10min	36	1	S	1	Flow rate, tubing head pressures, temperature, choke opening	Choke opening
Omrani et al. (2019)	ANN, Hybrid ANN	15min/1d	1y-4y	1		6w - 10y	Cumulative gas production, wellhead pressure, reservoir pressure, choke opening	
Razak et al. (2021)	LSTM	1m	1/3	1/6	S/M	10m/5y	completion parameters, formation and fluid properties, operating controls, and early production response data	control trajectories
Sagheer and Kotb (2019)	LSTM, DRNN, DGRU, ARIMA, NEA, HONN	1m	1-6	1	S	75m	production data	
Thavarajah et al. (2022)	MIF, DeepAR, TFT	1m	1-30	10	M		Static features (geology, spacing, completion characteristics), historical production	
Wang et al. (2021)	LSTM, ensemble method, SVM, LR, tree-based and Gaussian Process	1d	80-300	1	S	300-700d	Past downhole temperature and pressure data	
Werneck et al. (2022)	GRU, CNN, RNN seq2seq, DeepAR, N-BEATS, Transformer, Prophet	1d	85	30	S	400d	production data, injection data, and well pressure	
Yan et al. (2019)	XGBoost	1d		30	M		temperature, pressure, steam volume, well status, oil volume	Steam allocation plans

S = Single-output, M = Multiple-output

4 AN APPROACH FOR BUILDING A PETROLEUM FORECASTING MODULE FOR A DIGITAL TWIN

This section describes our two main contributions. The first (Section 4.1) refers to an analysis of the impact on data-driven forecasting model performance of the different parameters and configurations previously described as gaps in the literature and evaluated with experiments done with data from stakeholder Petrobras. From this evaluation, we acquired important insights about model behavior using future covariates, different window sizes, and multiple sampling frequencies. This evaluation helped our second contribution, a POC architecture that acts as a DT component for production forecasting. Through conversations with Petrobras, we decided to describe and implement a microservice-based full-stack architecture (Section 4.2), which was also used to generate the results of our first contribution.

4.1 Evaluating Machine Learning Approaches for Petroleum Production Forecasting

Using the architecture described in Section 4.2, we apply the approach outlined in Section 4.1.1 to evaluate and compare the prediction error of four different time series ML techniques in various combinations of lookback sizes and horizons, as well as sampling frequencies. We also measured the impact of using a future operational feature, in this case, the *choke valve* actuation. Our experiments considered the following ML techniques for oil production forecasting: a Naive Baseline method, LR, XGBoost, LSTM, and TFT. Our naive baseline method involves repeating the last known production value k-times. For training models, we use python library *Darts* base classes, which extend on *scikit-learn* and *pytorch's* regression models. We evaluated the techniques with models trained on the dataset from one of the early exploration missions in the Libra block called SPA-1.

4.1.1 Methodology

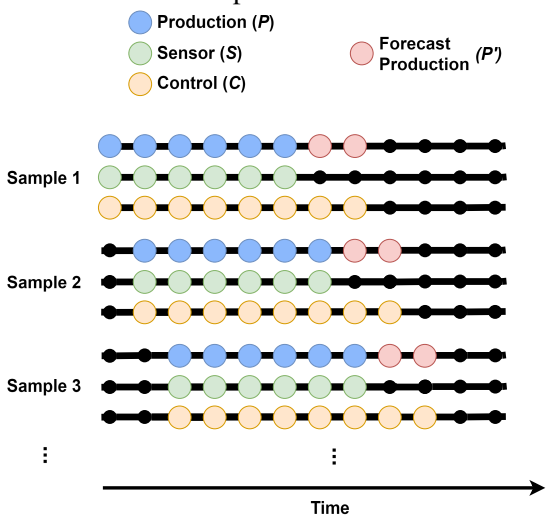
In this work, we assume $\mathbf{P} \in \mathbb{R}^{l}$, $\mathbf{S} \in \mathbb{R}^{s \times l}$, and $\mathbf{C} \in \mathbb{R}^{c \times (l+k)}$ as the sets of time series observations for our problem, representing respectively **production**, **sensor**, and **control variables** of a well or platform. Production variables relate to the flow rate of

produced fluids of a well. Sensor variables relate to physical attributes of certain parts of the well or platform, such as the temperature or pressure at the wellhead annular or choke valves. Control variables are the ones whose future values are known and can be directly controlled. An important example of a control variable is the choke valve opening and the amount of fluids injected into the reservoir for enhanced recovery. In this context, we can define a forecast function f that takes in l past values of **C**, **S**, **P** and k future known values of **C** and map them to a forecast $\hat{P}$ of size k (Equation 4.1). The variables s and c denote the amount of sensor and control variables.

$$
\begin{aligned}
f(P,S,C) &= \begin{bmatrix} \hat{P}(t+1) & \cdots & \hat{P}(t+2) & \hat{P}(t+k) \end{bmatrix}, where \\
P &= \begin{bmatrix} P(t-l) & \cdots & P(t-1) & P(t) \end{bmatrix} \\
S &= \begin{bmatrix} S_1(t-l) & \cdots & S_1(t-1) & S_1(t) \\ S_2(t-l) & \cdots & S_2(t-1) & S_2(t) \\ \vdots & \cdots & \vdots & \vdots \\ S_s(t-l) & \cdots & S_s(t-1) & S_s(t) \end{bmatrix} \\
C &= \begin{bmatrix} C_1(t-l) & \cdots & C_1(t) & \cdots & C_1(t+k) \\ C_2(t-l) & \cdots & C_2(t) & \cdots & C_2(t+k) \\ \vdots & \cdots & \vdots & \vdots & \cdots \\ C_c(t-l) & \cdots & C_c(t) & \cdots & C_c(t+k) \end{bmatrix}
\end{aligned}
\tag{4.1}
$$

For training, evaluating, and testing the ML methods, we rely on P, S, and C to construct samples that are fed into the models using the sliding window strategy. This strategy involves creating input-output pairs by sliding a window of size $l+k$ over the time series with a specified step size to generate overlapping segments. Figure 4.1 illustrates this procedure, with $l = 6$ and $k = 2$.

Figure 4.1 – Sample construction from sliding window strategy with lookback size of 6 and output size of 2



Pseudocode 1 describes our strategy for evaluating the errors of each ML method plus a baseline over a combination of parameters. We applied a day-chaining nested CV function ($cross\_val$), described in Section 4.1.1.4, for each combination of *lookback* (l), horizon (k), sampling frequency ($freq$), and for each ML method. The function returns the average Root-Mean Squared Error (RMSE) (*avg_pred_error*) of all splits, which is used in the comparisons presented in Section 4.1.2. The sampling frequencies we use are 10 minutes (10-min), hourly (1h), and daily (1-day). Ideal sizes for l and k are challenging to estimate; therefore, for this work, we tested a combination of 15 lookbacks and horizon sizes, where the horizon size is never greater than the lookback size. In summary, l and k are chosen from $window\_sizes$, which is equal to $[1, 6, 12, 24, 168]$ for 10-min and 1h frequency, and $[1, 7, 30, 60, 90]$ for 1-day frequency.

Algorithm 1 Model comparison methodology

Input: Multivariate time series dataset D with n timesteps
Require: M, a forecasting model

for $method$ in [baseline, lr, xgb, lstm, tft] **do**
  for $freq$ in [1day, 1h, 10min] **do**
    for l, k in $window\_sizes$ **do**
      if $l \geq k$ **then**
        Split D into K_{outer} splits
        for i = 1 to K_{outer} **do**
          Split D_i into $D_i^{outer\_train}$, $D_i^{outer\_validation}$ and $D_i^{outer\_test}$ following day-chaining CV strategy
          for j = 1 to K_{inner} **do**
            Train M_{method} on $D_i^{outer\_train}$ with suggested hyperparameter set p
            Evaluate M_{method} on $D_i^{outer\_test}$ and get error E_j^{test}
          Train M_{method} on $D_i^{outer\_train}$ with hyperparameter set p from T with the smallest E_j^{test}
          Evaluate M_{method} on $D_i^{outer\_test}$ and get E_i^{test}
        $avg\_pred\_error \leftarrow \sum_{i=1}^{k} E_i^{test} \times 1/K_{\text{outer}}$

4.1.1.1 Dataset

We used data from the SPA-1, which started around January 2018 and ended in September 2019 in sector 2 of the Mero field, Libra block. SPA-1 consisted of one producing well and one active injector well at a given moment. The producer was produced in two intervals, while the active injector was in three different intervals.

The dataset includes 114 time series features downloaded from the Plant Information Management System (PIMS) relating to rates, pressures, temperatures, and valve

openings of multiple sections of the wells, platform, and coupled components. The time series do not have a regular frequency, but on average, every variable is observed every 30 seconds by the sensors. Still, variations occur due to signal errors and how data is compressed before being inserted into the PIMS. In summary, the features include:

1. Well features:
   (a) Wellhead pressures and temperatures.
   (b) Tubing pressures and temperatures.
   (c) Annular pressures and temperatures.
   (d) ICV openings and pressures.
2. Platform:
   (a) Choke openings, pressures, and temperatures.
   (b) Gas, oil, and water rates.

4.1.1.2 Feature Selection

We performed Exploratory Data Analysis on the 114 features to determine their completeness and start/end time of observations. We consider the features measured between January 2018 and August 2019 in the feature selection step. Our feature selection strategy involved analyzing the temporal cross-correlation between time series and their ability to pass the Granger causality hypothesis test. We selected features that, at the same time:

1. have correlation value ≥ 0.65 or ≤ -0.65 with the production rate as the response variable.
2. pass the Granger causality hypothesis test with p-value = 0.05 in the direction of causing the production rate time series.

The correlation threshold of 0.65 was chosen from empirical experimentation. Values different from this would select variables that worsened model performance. We apply this analysis to every sample frequency (1-day, 1h, and 10-min). The cross-correlation plots of the selected features can be seen in Figures 4.2, 4.3 and 4.4, where the X-axis shows the lag value of the descriptive time series concerning the target time series, and the Y-axis shows the degree of linear correlation that both have. Table 4.1 describes the

list of selected features, while Appendix B gives statistical measurements about these features sampled to minute frequency. The choke position is the control feature whose future values are known and fed into the model.

Table 4.1 – Variables selected for training, for each sampling frequency

Description	Unit	1-day	1h	10-min
Measurement Date	Date	x	x	x
Well Oil Rate	m^3/h	x	x	x
Well Gas Rate	m^3/h	x	x	x
Choke Position	%	x	x	x
Downstream Temperature of Choke	°C	x		
Top Annular Pressure	kgf/cm²	x		
Top Tubing Pressure	kgf/cm²	x		
Upstream Production Temperature	°C	x	x	x

Figure 4.2 – Correlation between lagged features and the target feature (oil production) at 1-day frequency

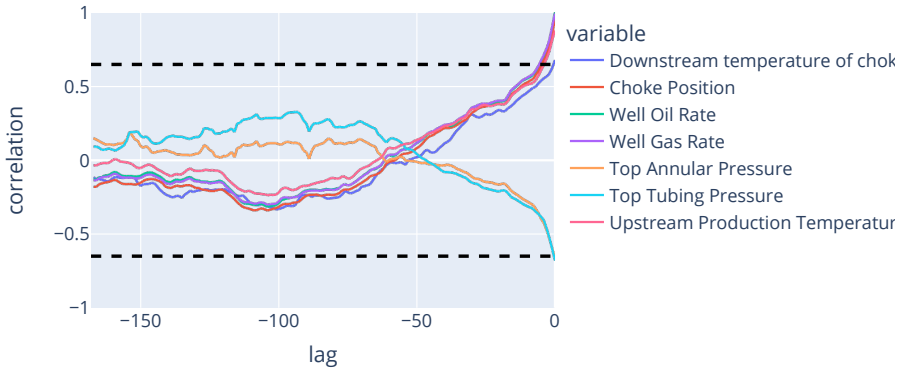


Figure 4.3 – Correlation between lagged features and the target feature (oil production) at 1h frequency

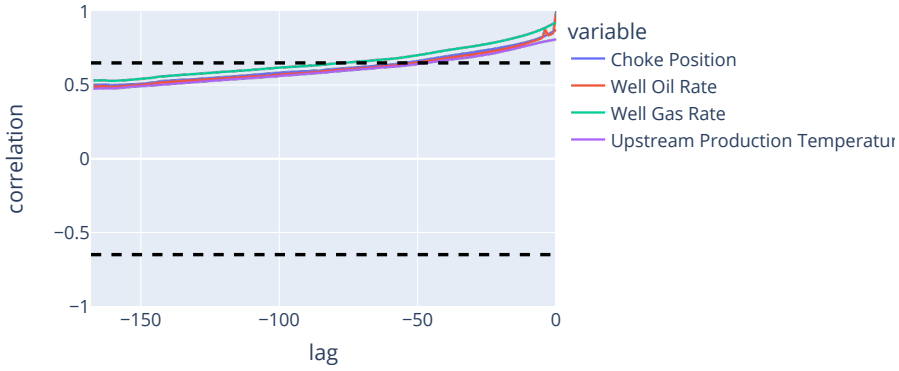


Figure 4.4 – Correlation between lagged features and the target feature (oil production) at 10-min frequency

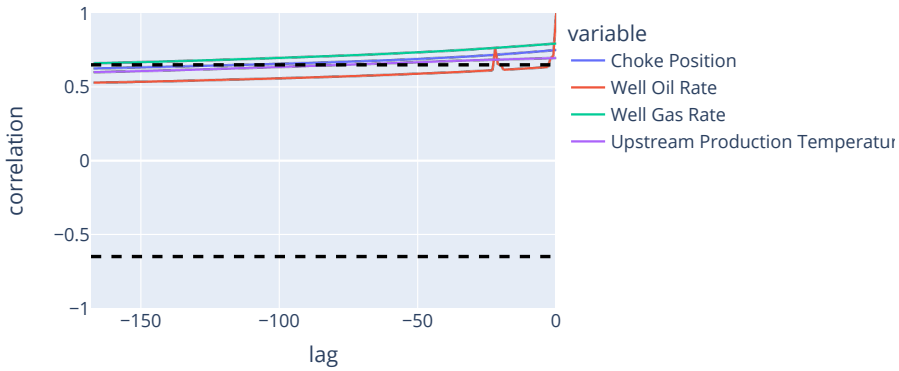


Table 4.7 shows the number of missing values present for each feature, while Figures 4.6 and 4.5 show the distribution of all input features. The rectangular box in Figure 4.6 represents the interquartile range (IQR), and the two extending lines define the range that encompasses most of the data, except outliers.

Figure 4.5 – Distribution of all input features with MinMax normalization

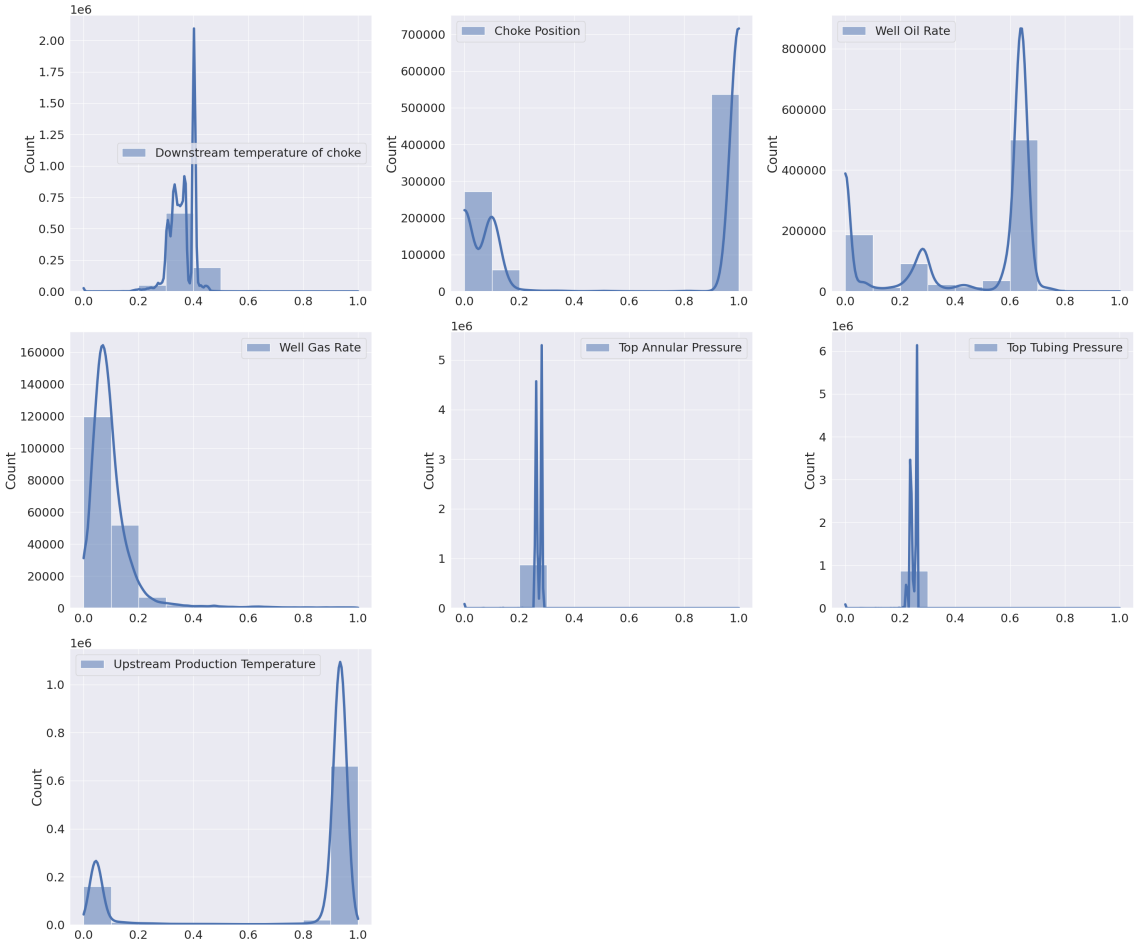


Figure 4.6 – Box Plot diagram of features with MinMax normalization

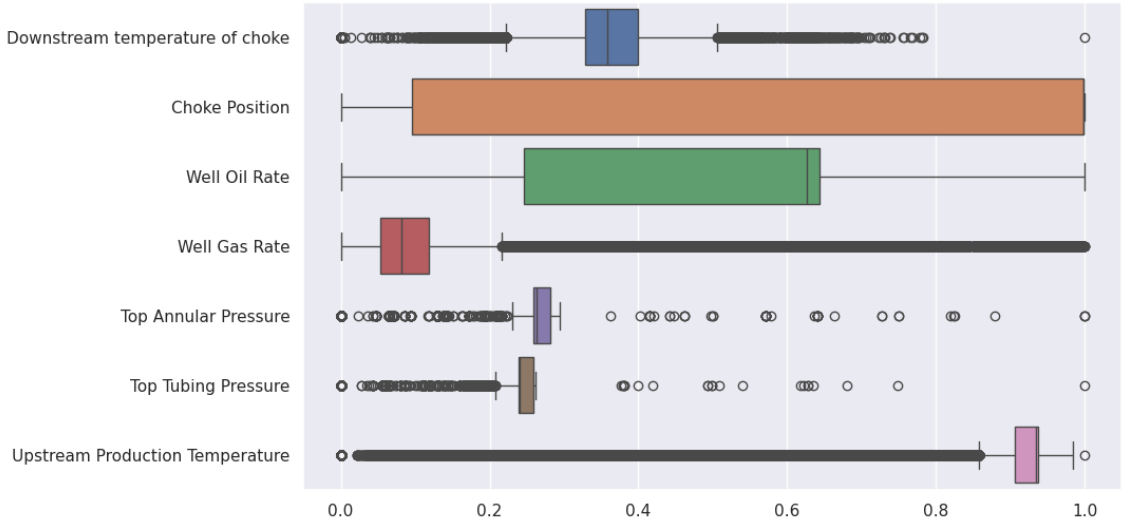
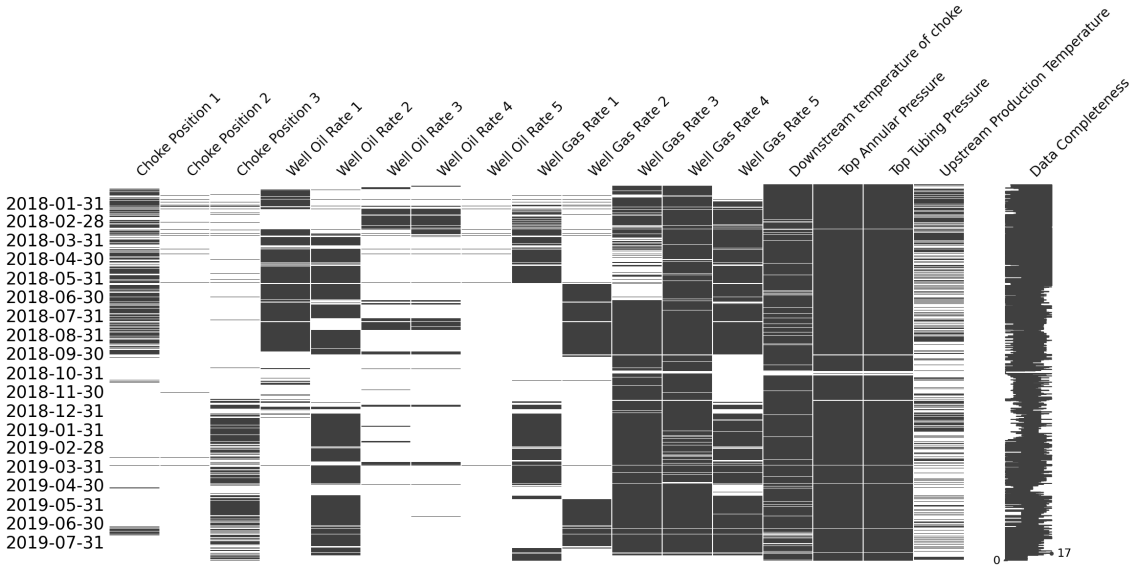


Figure 4.7 – Missing values for all input features



4.1.1.3 Pre-processing

Outlier removal was performed by removing data points outside the mean of the last 12 hours' standard deviation for each feature. Data points are then resampled to frequencies of 10 minutes, 1 hour, or 1 day using the mean. In accordance with the PI's system's inherent data compression strategy of removing redundant data points (AVEVA Group Limited and its subsidiaries, 2023), empty data points are filled using linear interpolation. The empty values for each feature can be seen in Figure 4.7. Features are then normalized using Min-Max ($X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}}$).

4.1.1.4 Model Training and Evaluation Strategy

The choice of the best forecasting technique (LR, XGBoost, LSTM, TFT, or baseline) required a robustness analysis involving the use of a specific CV technique for time series. We adopted a strategy, also used in a previous work (PIVETTA et al., 2023), to evaluate time series models, called day-chaining Nested CV. The technique's average prediction error is estimated from each CV split. For these experiments, the number of splits is equal to 3. We train the best candidate model for each split using the best hyperparameters from the hyperparameter tuning procedure. Early stopping stops the training of models based on the evolution of validation loss. We chose the early stopping parameters: minimum delta of 0.01 and patience of 3. We emphasize that all models used to adopt a multi-output prediction strategy (described in Section 2.4.6), except for LSTM, which performs iterative predictions until the horizon size is reached.

4.1.1.5 Hyperparameter Tuning

Heuristic methods[1] were used to determine the best hyperparameters for each method based on multiple refining trials. We used a Tree-Structured Parzen Estimator (TPE) to sample ideal hyperparameter values in each trial. For pruning unpromising trials, we used the Median strategy, where a trial is pruned if its current value is worse than the intermediate results at the same step of previous trials. Table 4.2 shows each technique's search ranges for each hyperparameter. The Linear Regression (RL) technique has no hyperparameters and is, therefore, not in the following table.

[1] We used the *Optuna* library to perform the search for the best hyperparameters for each model. TPESampler class is used for sampling ideal hyperparameter values, and MedianPruner class for pruning unpromising trials.

Table 4.2 – Value ranges for each hyperparameter of the techniques

Method	Hyperparameter	Space
XGB	learning_rate	(0.01, 0.1)
	max_depth	(1,32)
	min_child_weight	(1,10)
	gamma	(0,5)
	colsample_bytree	(0.5, 1.0)
	n_estimators	300
	trials_hypertuning	60
LSTM	units	(1, 16)
	n_layers	(1, 4)
	learning_rate	(0.1, 1)
	trials_hypertuning	60
TFT	hidden_size	(1, 32)
	hidden_cont_size	(1, 32)
	attn_head_size	(1, 4)
	batch_size	[16,32]
	learning_rate	(0.001, 0.01)
	dropout	[0.1, 0.5, 0.9]
	optimizer	adam
	lstm_layers	{1, 2}
	trials_hypertuning	60

4.1.2 Results

Tables 4.3, 4.4, and 4.5 display the average RMSE over all CV splits, for the different models (LR, XGBoost, LSTM, and TFT) and across various combinations of lookback, horizon, and sampling frequency. Values highlighted in blue show the overall best candidate for the given horizon. The last two rows show the average RMSE of each technique and lastly the average RMSE of the best candidates of each technique (average of the values in italics for each column). In the last column, we also present the average of all techniques for a particular lookback and horizon combination (excluding the baseline).

Table 4.3 – RMSE errors for 1-day frequency

horizon	looback	BASELINE	RL	XGBOOST	RNN	TFT	avg
1	**1**	20.11	14.11	10.94	22.51	42.07	22.41
	7	20.11	19.96	12.82	41.35	45.58	29.93
	30	20.11	64.33	15.21	47.10	61.74	47.10
	60	20.11	56.18	10.47	40.07	42.25	37.24
	90	20.11	55.50	7.63	47.45	28.54	34.78
7	**7**	45.60	25.65	19.27	41.97	44.56	32.86
	30	45.60	76.24	17.40	75.27	83.62	63.13
	60	45.60	70.19	16.69	59.85	53.15	49.97
	90	45.60	52.67	13.49	29.71	25.08	30.24
30	**30**	90.61	74.01	19.46	58.81	42.51	48.70
	60	90.61	102.90	25.19	61.56	38.10	56.94
	90	90.61	47.90	24.85	26.99	29.41	32.29
60	**60**	117.10	67.02	43.93	36.35	53.91	50.30
	90	117.10	42.94	60.04	51.10	52.21	51.57
90	**90**	142.56	54.67	30.60	24.35	54.35	40.99
	avg	83.20	56.56	26.78	41.72	47.94	
	best avg	83.20	37.05	23.02	27.98	37.92	

Table 4.4 – RMSE errors for 1h frequency

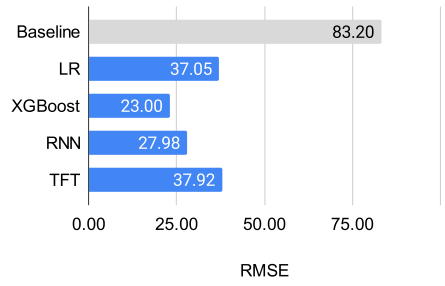
horizon	looback	BASELINE	RL	XGBOOST	RNN	TFT	avg
1	**1**	5.93	9.08	6.56	10.42	18.63	11.17
	6	5.93	7.68	8.12	8.30	23.59	11.92
	12	5.93	7.91	9.28	9.38	26.04	13.15
	24	5.93	8.21	10.18	12.68	21.96	13.26
	168	5.93	11.84	22.16	11.85	21.58	16.86
6	**6**	12.15	12.47	12.75	14.23	34.40	18.46
	12	12.15	12.65	12.69	49.10	26.71	25.29
	24	12.15	12.99	13.68	55.08	27.56	27.33
	168	12.15	18.37	16.68	14.40	30.97	20.11
12	**12**	17.27	16.19	14.88	15.86	22.09	17.26
	24	17.27	16.43	25.95	18.63	25.74	21.69
	168	17.27	22.54	29.74	25.47	24.47	25.56
24	**24**	24.75	20.84	17.04	20.85	27.52	21.56
	168	24.75	28.16	24.70	24.40	33.26	27.63
168	**168**	61.46	46.83	55.10	38.03	40.73	45.17
	avg	24.31	22.56	24.94	24.87	29.5	
	best avg	24.31	20.80	21.27	19.45	27.14	

Table 4.5 – RMSE errors for 10-min frequency

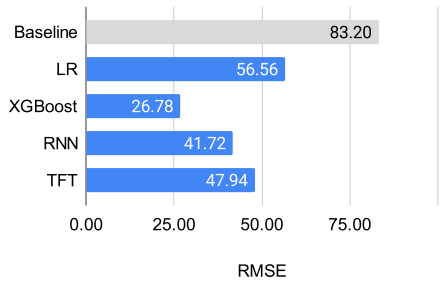
horizon	looback	BASELINE	RL	XGBOOST	RNN	TFT	avg
1	1	5.48	8.31	8.66	30.47	31.03	19.62
	6	5.48	7.57	11.34	10.34	53.36	20.65
	12	5.48	7.49	6.91	12.51	30.48	14.35
	24	5.48	6.70	9.24	13.43	9.87	9.81
	168	5.48	7.15	5.86	12.07	39.43	16.13
6	6	9.04	11.31	8.14	16.15	26.10	15.43
	12	9.04	11.22	12.02	29.59	33.61	21.61
	24	9.04	10.17	20.14	9.67	14.43	13.60
	168	9.04	11.14	24.14	14.08	27.30	19.17
12	12	11.34	12.97	14.21	36.32	50.79	28.57
	24	11.34	12.12	25.81	42.00	55.33	33.82
	168	11.34	13.24	10.38	16.07	15.44	13.78
24	24	14.46	14.56	12.06	21.64	22.74	17.75
	168	14.46	15.69	12.70	16.34	16.12	15.21
168	168	33.60	28.74	28.44	28.24	43.23	32.16
	avg	14.78	15.01	16.43	22.37	32.27	
	best avg	14.78	14.46	12.98	16.13	19.82	

Figures 4.8, 4.9, and 4.10 compare the total average error (*best avg* of previous tables) for each technique over the different sampling frequencies.

Figure 4.8 – Average RMSE of each technique for 1-day frequency

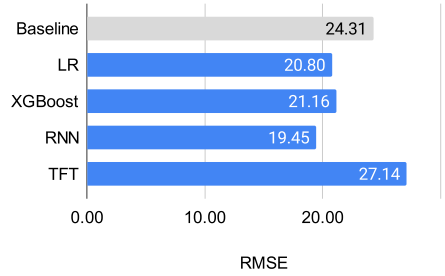


(a) *best avg* (average of only the best candidates of each horizon)

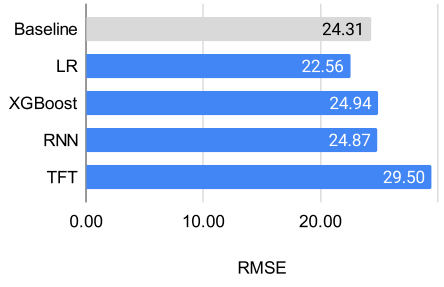


(b) *avg* (average of all candidates)

Figure 4.9 – Average RMSE of each technique for 1h frequency

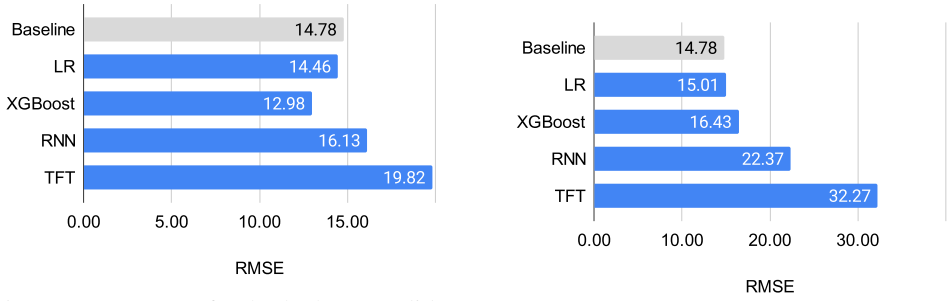


(a) *best avg* (average of only the best candidates of each horizon)



(b) *avg* (average of all candidates)

Figure 4.10 – Average RMSE of each technique for 10-min frequency



(a) *best avg* (average of only the best candidates of each horizon)

(b) *avg* (average of all candidates)

By evaluating the average RMSE between the splits and various combinations of lookback and horizon values, we determined that the XGBoost ensemble tree model has the best overall performance, coming on top on the 10-min and 1-day sampling frequencies. Tree ensemble methods show very good performance with tabular data and, in most cases, outperformed more complex methods. XGBoost showed an overall 32.34% decrease in prediction error over the baseline approach, compared to the second-best method, RNN, which showed a 25.74% decrease over the baseline. LR came third with a 24.02% reduction, followed by TFT's 2.89%.

Notably, TFT and RNN failed to beat the baseline, and LR was just slightly below the threshold in 10-min frequency. TFT was the only method that failed to beat the baseline in 1h frequency. XGBoost and LR were the only methods to beat the baseline in all sampling frequencies. In the 1-day frequency, every method beat the baseline by long margins ($\approx$ 62.14% error decrease). If we consider the performance of all methods for the 10-min and 1h frequency, error reduction over the baseline was more modest, at 12.34%, 5.43%, and 8.30% for XGBoost, RNN, and LR, respectively. TFT, on the other hand, showed a 22.87% increase of errors in these sampling frequencies.

Figures 4.11, 4.12, and 4.13 evaluate the average RMSE over all lookback sizes, making it clear that lookback size affects model prediction errors.

For all cases, errors tend to increase as the horizon value increases. For the 10-min frequency, errors tend to decrease when a longer lookback is used but sometimes can reach a local minimum before the biggest lookback. This behavior happens (but to a lesser extent) in the 1-day sampling frequency. For the 1h frequency, the smallest lookback size is mostly optimal.

Figure 4.11 – 1d average RMSE error across all lookback sizes

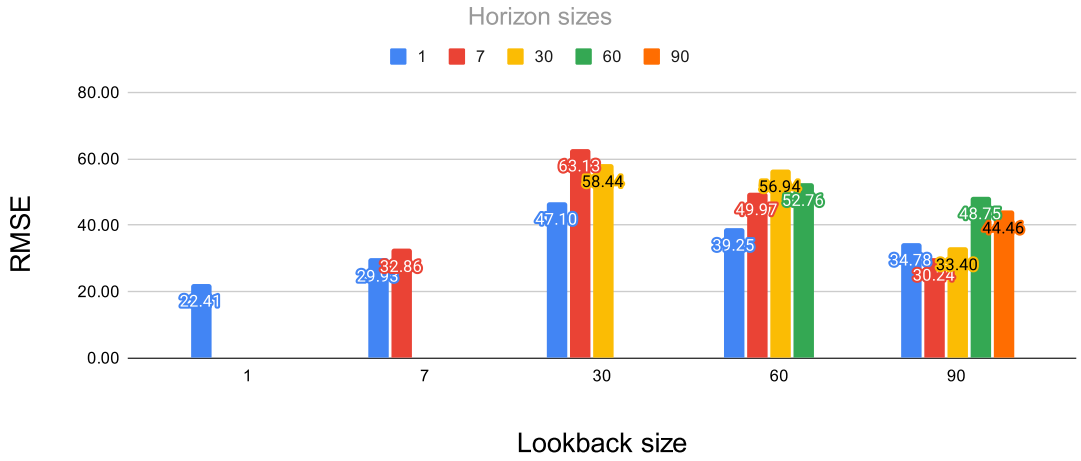


Figure 4.12 – 1h average RMSE error across all lookback sizes

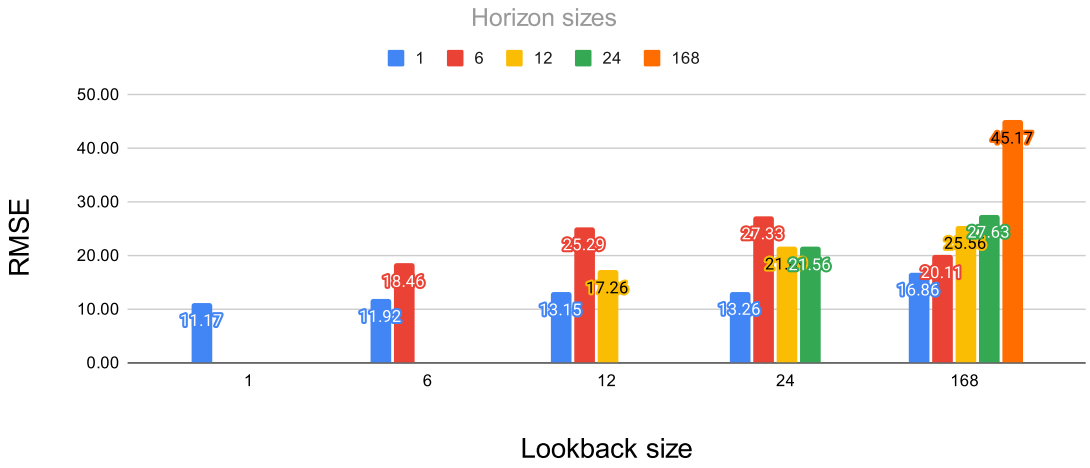
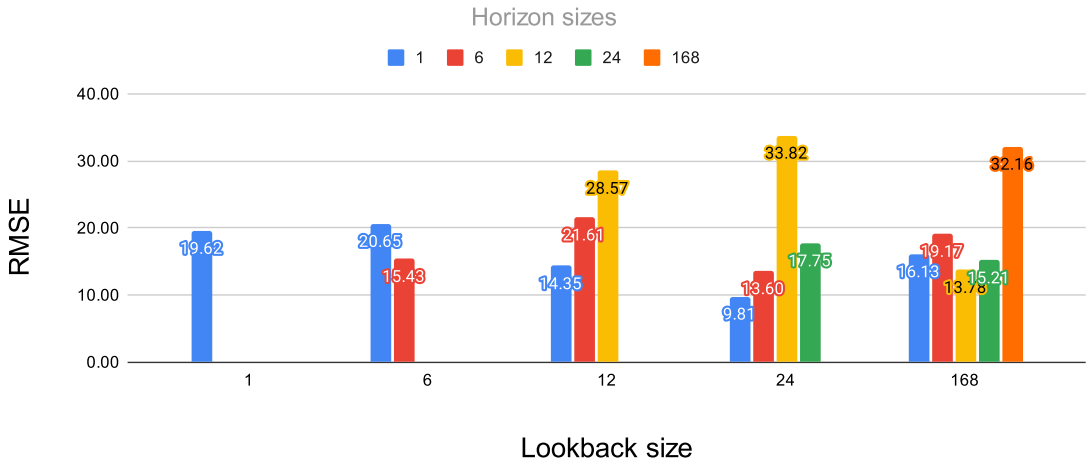


Figure 4.13 – 10-min average RMSE error across all lookback sizes



For comparing the impact of the future covariate, we repeated the same training procedure but without considering the future choke position % variable. We re-trained the models in only the worst overall lookback for each horizon. The results (Tables 4.6, 4.7 and 4.8) were compared to previously trained models in the same window setting.

Table 4.6 – RMSE errors for 1-day frequency without considering future choke opening %

horizon	lookback	LR	XGBoost	LSTM	TFT	avg	choke avg	% difference
1	30	176.2	40.18	31.69	65.32	78.35	**47.10**	-39.89%
7	30	433	83.94	74.52	116.1	176.89	**63.13**	-64.31%
30	60	260.6	85.07	90.49	128.7	141.22	**56.94**	-59.68%
60	90	309.1	91.95	126.8	120.4	162.06	**51.57**	-68.18%
90	90	239.4	90.03	87.39	117.9	133.68	**40.99**	-69.34%
								-60.28%

Table 4.7 – RMSE errors for 1h frequency without considering future choke opening %

horizon	lookback	LR	XGBoost	LSTM	TFT	avg	choke avg	% difference
1	168	12.27	13.92	7.959	22.19	**14.08**	16.86	19.74%
6	24	16.76	33.69	14.87	41.42	**26.69**	27.33	2.40%
12	168	29.28	30.88	30.31	51.44	35.48	**25.56**	-27.96%
24	168	41.82	54.25	43.9	91.42	57.85	**27.63**	-52.24%
168	168	94.23	87.99	121	125.7	107.23	**45.17**	-57.88%
								-23.19%

Table 4.8 – RMSE errors for 10-min frequency without considering future choke opening %

horizon	lookback	LR	XGBoost	LSTM	TFT	avg	choke avg	% difference
1	6	7.74	8.94	7.50	39.58	**15.94**	20.65	29.55%
6	12	11.6	22.04	40.52	30.12	26.07	**21.61**	-17.11%
12	24	16.49	16.78	14.43	77.74	**31.36**	33.82	7.84%
24	24	12.9	19.42	28.5	56.73	29.39	**17.75**	-39.61%
168	168	42.79	44.83	73.04	86.13	61.70	**32.16**	-47.89%
								-13.44%

As for consideration of the future variable, there was an overall 32.30% decrease in error when models were trained with the future covariate, as opposed to without it. Even in cases where RMSE didn't improve with the inclusion of the future variable, the difference was insignificant.

4.1.3 Discussion

Through the comparison of error metrics, it was determined that XGBoost, a tree-ensemble ML method, exhibited the most favorable overall performance, even in the absence of dedicated mechanisms for handling time series sequences, such as attention layers and gating found in TFT and LSTM. We hypothesize that the challenges faced

by deep learning models may be attributed to the limited size and variability of the observed data, comprising only two years' worth of observations from a specific geological and production environment involving a producer and two injector wells. LR performed pretty well for its simplicity, even beating the baseline in all sampling frequencies. This evidences the importance of considering simpler time series forecasting techniques before parameter-heavy ones.

Additionally, our findings indicate that the effectiveness of prediction models relies on the careful selection of lookback and horizon size. The accurate prediction of oil production necessitates the use of historical data to anticipate future production, considering that reservoir and well pressure transients unfold gradually over time. The lookback window influences the historical information available to the model. Large lookback windows result in more adjustable parameters and higher training costs, while models with limited past information tend to perform poorly but are computationally cheap. Our proposed model evaluation strategy, employing appropriate nested CV for time series, has proven to be a concise technique for selecting horizon size and lookback. Results indicate that the sampling frequency affects the choice of horizon size, with 1h frequency models not improving with increased lookback, while 10-min and 1-day ones do. This could be due to the past 1h frequency data points not having the right granularity to provide the models with new information about oil flow dynamics. Another complementary explanation is that the naive prediction of the baseline makes it very close to the true value because in this sampling frequency the values don't change as much. Nonetheless, we acknowledge that the analysis of the lookback impact for the biggest horizon sizes (90 for 1-day frequency and 168 for 1h and 10-min) is hindered by the fact that we adopted the rule that horizon size is never greater than the lookback size.

We also stress the significance of integrating planned future operational data to improve production forecasting. The results highlight an important influence of operational variables in accounting for transient phenomena within the well and reservoir.

The feature selection step for time series forecasting presents unique challenges due to the temporal nature of the data. Unlike traditional feature selection problems, where static features can be assessed independently, time series data introduces dependencies across observations. While cross-correlation is a valid metric to determine the target response to a system variable, it might not be the best for this scenario. Using feature importance or other wrapper methods for selecting features proved unpractical due to the sheer number of variables present at the start.

4.2 A digital twin module for production forecasting

This microservice-based architecture marks our proposal for a component that can be integrated into a DT service system layer, previously described in Section 2.1. From meetings with the stakeholders, we devised a list of functional and non-functional requirements specialists saw as important to add to our solution. These are:

1. To enable training of different types of time series forecasting models and enable the deployment of the best ones.

2. To create production forecasts, what-if scenarios, and backcasting based on historical data and future planned data from O&G production.

3. To take into account various time horizons (future periods) for forecasting and hypothetical scenarios development.

4. To consider different sampling frequencies (days, hours, minutes, etc.) for production forecasting.

5. To enable both forecasting and hypothetical scenarios creation efficiently (utilizing fewer computational resources) and swiftly (providing near real-time responses). It is noteworthy that efficiency and response time, in this context, are evaluated in relation to other existing solutions at Libra, such as numerical simulation. While the latter offers satisfactory precision in predictions, it demands a considerable amount of computational resources and significant time to deliver responses.

6. To handle constant data flow, given that the DT's state is continuously updated based on sensor data monitoring various properties in the production plant.

We proposed and implemented the microservice architecture shown in Figure 4.14 to fulfill these requirements. It illustrates (1) the parts of the DT component and (2) the use cases the two actors can partake in. Essentially, the DT component is made up of self-contained units called microservices. The data source feeds the data needed for model training and inference. Another microservice deals with experiment tracking, the persistence of artifacts (generated models and other objects), and metrics (training errors, training parameters, hyperparameters, etc.). Finally, a user interface is also provided, which takes the required parameters for inference and can draw the forecasts for the end-user. The full implementation [2] for the DT component is shown in Appendix C. We

[2] https://github.com/BDI-UFRGS/MLFlow-TimeSeries-Oil-Stack

present two actors for the use cases. Actor 1 is in charge of training models and deciding which ones to use for inference later, while Actor 2 executes the saved models to generate forecasts.

The possible use cases for Actor 1 within this architecture are:

1. Train model

   A model is trained after Actor 1 chooses all possible training parameters: type of ML model, number of cross-validation splits, sampling frequency, size of lookback and horizon, hyperparameter intervals, selected features, preprocessing strategy, and feature scaling strategy. The model is then saved in the experiments and model tracking service for later use.

2. Register model

   Actor 1 can visualize metrics relating to training and decide what models will be chosen (or registered) for inference.

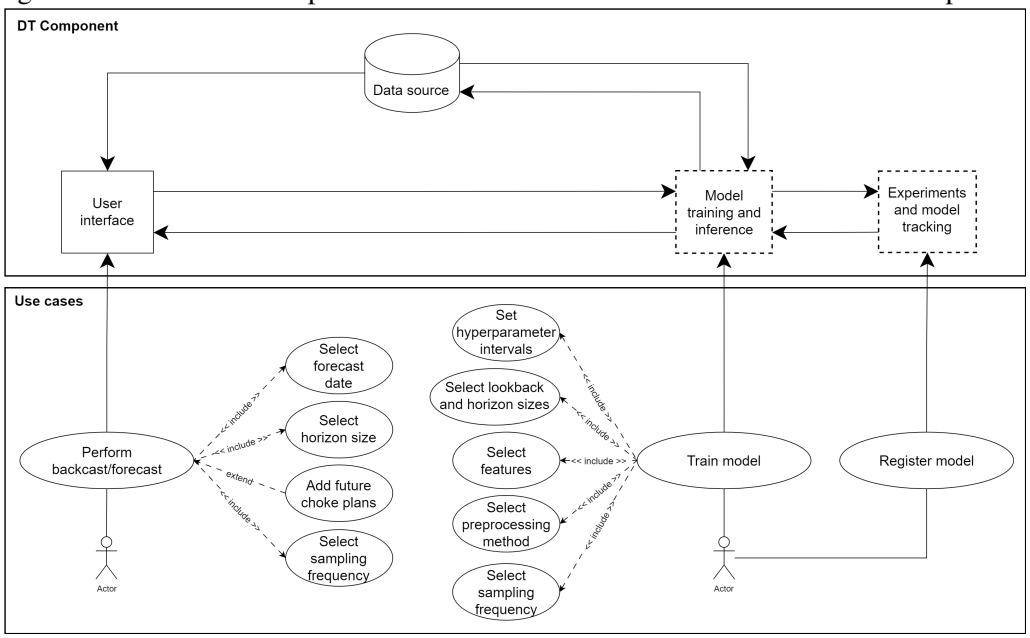
The possible use case for Actor 2 within this architecture is:

1. Perform backcast/forecast

   A dedicated user interface is provided to them, which allows the input of mandatory parameters: the date from which the forecast will start, the size of the horizon, and sampling frequency. The optional parameter is the operational variable. The actor will either use the existing historical values for the time they selected or input the future plans. In the case of forecasts, where the origin date is in the future, the input of future operational features is mandatory.

Appendix D shows the practical execution of these use cases.

Figure 4.14 – Schematic representation of use cases and abstract architecture components



5 CONCLUSION

Establishing a standard pipeline for training purely data-driven petroleum forecast models is difficult due to numerous physical, geophysical, and operational factors. These factors include complex dynamics within the reservoir rock, different types of reservoirs, equipment malfunction, and the need for domain knowledge. Creating standards for training data-driven models in this task has, therefore, proven to be a challenging deal. In this work, we employ a set of novel methodological steps to compare data-driven models to forecast production, considering future covariates, multiple combinations of window sample sizes, and various data sampling frequencies.

We compared error metrics and determined that XGBoost, a tree-ensemble ML method, had the best overall performance, even without dedicated mechanisms to handle time series sequences, such as attention layers and gating present in TFT and LSTM. We believe these deep learning models may have struggled due to the limited size and variability of the observed data, which included only two years' observations from a producer and an injector well in a specific geological and production environment.

Our results also suggest that model performance depends on the appropriate selection of lookback and horizon size. Accurate prediction of oil production relies on historical data to anticipate future production, as reservoir and well pressure transients occur over time and not instantaneously. The lookback window influences the amount of historical information the model has from the past; larger windows allow models to understand long-term behaviors better, while smaller windows make recent past changes have a more significant influence on the final prediction. Therefore, it is crucial to design models that receive just enough information about the past. Models with a very large lookback window have more adjustable parameters and are more expensive to train. On the other hand, models less informed about the past typically perform poorly. Our proposed model evaluation strategy using appropriate CV for time series has proven to be a concise technique for selecting horizon size and lookback in this domain, and results show that increased lookback size can improve model accuracy.

We emphasize the importance of incorporating planned future operational data to enhance production forecasting. Our findings demonstrate that this approach is crucial for modeling the change in production rate caused by the opening and closing of flow restriction valves. Our study shows that operational variables play a significant role and are an important way to account for transient phenomena within the well and reservoir.

5.1 Limitations

Developing forecasting models for predicting petroleum production using ML methods represents a complex problem encompassing the multiple systems involved in oil flow. We identified some obstacles and limitations to our work, which include:

1. Data size, variability, and bias. ML techniques, especially those that rely on many neural layers (deep NNs), must have enough input data to work as a robust predictor that can address variations in production behavior in multiple physical settings. The data used for training in this work is considerably small and biased as it relates to a production system with only one production and one injector well. Moreover, it only comprises about two years of production data in a novel reservoir submitted to EWT.

2. Not using data generated by numerical simulators. While there are reasons to explore exclusive ML-based approaches, we acknowledge the importance of augmenting the training dataset with synthetic data. The lack of data augmentation is due to a lack of system knowledge and data to build said simulators.

3. The decision not to predict gas or water and oil separately. Likewise, the decision not to train a model that can provide an output for each fluid production. Water production was only measured daily.

4. Not considering integrating numerical models or a hybrid approach involving numerical models and ML. While numerical models offer detailed physics-based simulations, combining them with ML techniques can enhance predictive capabilities, leveraging the strengths of both approaches for a more comprehensive understanding of petroleum reservoir behavior. Works in literature have shown the usefulness of hybrid models when modeling long-term production behavior. Due to data restraints and the scope of this work, we decide not to develop a hybrid approach.

5.2 Future work

1. Evaluate the accuracy of a single “global model” that’s trained with data from all wells.

2. Consider training a weak model in one or multiple wells from one reservoir and transfer learning its parameters to another model to be further trained by data from wells in another reservoir. Similarly, do the same procedure but for wells in the same reservoir.

3. Forecasting models could extend to predicting well behavior under specific recovery technologies, such as cyclic steam injection. This application is particularly relevant in the oil industry, where recovery techniques are prominent. ML algorithms can analyze the response of wells to different recovery technologies, offering valuable insights for optimizing production processes. We could consider steam allocation plans and other advanced recovery values in future works.

4. Consider lookback window size not as predefined values but as a hyperparameter that's chosen from a range of values via Bayesian hyperparameter tuning.

5. Include geographical location as a static feature, thus creating a spatiotemporal aware model, a concept present in other papers in the literature.

6. Include reservoir static features to consider the subsurface geological aspects the well is in when constructing the model.

7. Use a reservoir model, configure multiple scenarios of well placement and geophysical attributes of the subsurface, and generate data that serves as input to a data-driven model.

8. Consider modifying the objective function to penalize model prediction overestimation.

REFERENCES

ABDRAKHMANOV, I. R. et al. Development of Deep Transformer-Based Models for Long-Term Prediction of Transient Production of Oil Wells. Society of Petroleum Engineers (SPE), 10 2021.

AIZENBERG, I. et al. Multilayer Neural Network with Multi-Valued Neurons in time series forecasting of oil production. **Neurocomputing**, v. 175, p. 980–989, 1 2016. ISSN 09252312.

AL-ALI, Z. A. A. H.; HORNE, R. Meta Learning Using Deep N-BEATS Model for Production Forecasting with Limited History. **Society of Petroleum Engineers - Gas and Oil Technology Showcase and Conference, GOTS 2023**, OnePetro, 3 2023. Available from Internet: <https://dx.doi.org/10.2118/214214-MS>.

AL-ALI, Z. A.-A. H.; HORNE, R. Probabilistic Well Production Forecasting in Volve Field Using Temporal Fusion Transformer Deep Learning Models. OnePetro, 3 2023. Available from Internet: </SPEGOTS/proceedings-abstract/23GOTS/3-23GOTS/517872>.

AL-JASMI, A. et al. Short-Term Production Prediction in Real Time Using Intelligent Techniques. In: **All Days**. [S.l.]: SPE, 2013.

ALALI, Z. H.; HORNE, R. N. A Comparative Study of Deep Learning Models and Traditional Methods in Forecasting Oil Production in the Volve Field. In: **Day 3 Wed, October 18, 2023**. [S.l.]: SPE, 2023.

American Petroleum Institute. **Glossary of Oilfield Production Terminology (GOT)**. 1. ed. [S.l.]: American Petroleum Institute (API), 1988.

ANDRIANOV, N. A Machine Learning Approach for Virtual Flow Metering and Forecasting. **IFAC-PapersOnLine**, Elsevier, v. 51, n. 8, p. 191–196, 1 2018. ISSN 2405-8963.

ANJOS, S. M. et al. Libra: Applied technologies adding value to a giant ultra deep water pre-salt field - Santos Basin, Brazil. In: **Offshore Technology Conference Brasil 2019, OTCB 2019**. [S.l.]: Offshore Technology Conference, 2020. ISBN 9781613996713.

ARPS, J. Analysis of Decline Curves. **Transactions of the AIME**, v. 160, n. 01, 12 1945. ISSN 0081-1696.

AVEVA Group Limited and its subsidiaries. **Compression testing**. 2023. Available from Internet: <https://docs.aveva.com/bundle/pi-server-da-admin/page/1021696.html>.

BACCIU, D. Unsupervised feature selection for sensor time-series in pervasive computing applications. **Neural Computing and Applications**, v. 27, n. 5, p. 1077–1091, 7 2016. ISSN 0941-0643.

BALAJI, K. et al. Status of Data-Driven Methods and their Applications in Oil and Gas Industry. In: **Day 3 Wed, June 13, 2018**. [S.l.]: SPE, 2018.

BARRICELLI, B. R.; FOGLI, D. Digital Twins in Human-Computer Interaction: A Systematic Review. **International Journal of Human–Computer Interaction**, p. 1–19, 9 2022. ISSN 1044-7318.

BEAR, J. **Dynamics of fluids in porous media**. [S.l.]: Courier Corporation, 1988.

BIKMUKHAMETOV, T.; JÄSCHKE, J. First Principles and Machine Learning Virtual Flow Metering: A Literature Review. **Journal of Petroleum Science and Engineering**, Elsevier, v. 184, p. 106487, 1 2020. ISSN 0920-4105.

BONTEMPI, G. Long Term Time Series Prediction with Multi-Input Multi-Output Local Learning. In: . [s.n.], 2008. Available from Internet: <https://api.semanticscholar.org/CorpusID:137250>.

BOX, G. E. P. et al. **Time series analysis: forecasting and control**. [S.l.]: John Wiley & Sons, 2015.

BROCKWELL, P. J.; DAVIS, R. A. **Introduction to time series and forecasting**. [S.l.]: Springer, 2002.

CHEN, T.; GUESTRIN, C. XGBoost: A Scalable Tree Boosting System. In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2016. p. 785–794. ISBN 9781450342322.

CLARK, N. **Elements of petroleum reservoirs**. Dallas: Society of Petroleum Engineers of AIME, 1969. 66–84 p. ISBN 978-0895202093.

COCCOLI, F. C. et al. Intelligent completion in extended well test. In: **Proceedings of the Annual Offshore Technology Conference**. [S.l.]: Offshore Technology Conference, 2019. v. 2019-May. ISBN 9781613996416. ISSN 01603663.

COSTA, F. F. et al. EWT program - Enabling optimization and speed up for Libra block production systems development in ultra-deepwater. In: **Proceedings of the Annual Offshore Technology Conference**. [S.l.]: Offshore Technology Conference, 2019. v. 2019-May. ISBN 9781613996416. ISSN 01603663.

CUKUR, H. et al. Cross correlation based clustering for feature selection in hyperspectral imagery. In: **2015 9th International Conference on Electrical and Electronics Engineering (ELECO)**. [S.l.]: IEEE, 2015. p. 232–236. ISBN 978-6-0501-0737-1.

DARCY, H. **Les fontaines publiques de la ville de Dijon: exposition et application...** [S.l.]: Victor Dalmont, 1856.

FULLER, A. et al. Digital Twin: Enabling Technologies, Challenges and Open Research. **IEEE Access**, Institute of Electrical and Electronics Engineers Inc., v. 8, p. 108952–108971, 2020. ISSN 21693536. Available from Internet: <https://ieeexplore.ieee.org/document/9103025/>.

GAO, C.; RAJESWARAN, T.; NAKAGAWA, E. A Literature Review on Smart-Well Technology. In: **All Days**. [S.l.]: SPE, 2007.

GEP. **The Increasing Popularity of Digital Twins in Oil and Gas**. 2020. Available from Internet: <https://www.gep.com/blog/mind/the-increasing-popularity-of-digital-twins-in-oil-and-gas>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. [S.l.]: MIT press, 2016.

GOVINDARAJAN, S. K. An overview on extension and limitations of macroscopic Darcy's law for a single and multi-phase fluid flow through a porous medium. **International Journal of Mining Science (IJMS) Volume**, v. 5, p. 1–21, 2019.

GRANGER, C. W. J. Investigating Causal Relations by Econometric Models and Cross-spectral Methods. **Econometrica**, v. 37, n. 3, p. 424, 8 1969. ISSN 00129682.

GRIEVES, M. Digital Twin: Manufacturing Excellence through Virtual Factory Replication. 4 2015.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. **Neural Computation**, v. 9, n. 8, p. 1735–1780, 11 1997. ISSN 0899-7667.

Holmes E. E.; Scheuerell M. D.; Ward E. J. **Applied Time Series Analysis for Fisheries and Environmental Sciences**. 2021. Available from Internet: <https://atsa-es.github.io/atsa-labs/>.

Honeywell. Essential Digital Twins in Upstream Oil & Gas Production Operations. 2020. Available from Internet: <https://process.honeywell.com/content/dam/process/en/documents/document-lists/doc-list-onshore-production/WhitePaper_EssentialDigitalTwinsForUpstreamOilAndGas_APC.pdf>.

HUBBERT, M. K. The theory of ground-water motion. **The Journal of Geology**, University of Chicago Press, v. 48, n. 8, Part 1, p. 785–944, 1940.

HUBBERT, M. K. Darcy's law and the field equations of the flow of underground fluids. **Transactions of the AIME**, OnePetro, v. 207, n. 01, p. 222–239, 1956.

JADON, A.; PATIL, A.; JADON, S. A Comprehensive Survey of Regression Based Loss Functions for Time Series Forecasting. **arXiv preprint arXiv:2211.02989**, 2022.

JIMÉNEZ, F. et al. Feature selection based multivariate time series forecasting: An application to antibiotic resistance outbreaks prediction. **Artificial Intelligence in Medicine**, v. 104, p. 101818, 4 2020. ISSN 09333657.

LAKE, L. **Enhanced oil recovery**. Englewood Cliffs, N.J: Prentice Hall, 1989. ISBN 978-0132816014.

LEE, T.-H. Loss functions in time series forecasting. **International encyclopedia of the social sciences**, Macmillan Thomson Gale Publishers Detroit, p. 495–502, 2008.

LI, X. et al. Time-series production forecasting method based on the integration of Bidirectional Gated Recurrent Unit (Bi-GRU) network and Sparrow Search Algorithm (SSA). **Journal of Petroleum Science and Engineering**, v. 208, 1 2022. ISSN 09204105.

LI, X. et al. A well rate prediction method based on LSTM algorithm considering manual operations. **Journal of Petroleum Science and Engineering**, Elsevier, v. 210, p. 110047, 3 2022. ISSN 0920-4105.

LIM, B. et al. Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. **International Journal of Forecasting**, Elsevier B.V., v. 37, n. 4, p. 1748–1764, 12 2019. ISSN 01692070. Available from Internet: <https://arxiv.org/abs/1912.09363v3>.

LIU, Y. et al. **Non-stationary Transformers: Exploring the Stationarity in Time Series Forecasting**. 2023.

LOH, K. K. L.; OMRANI, P. S.; LINDEN, R. V. D. Deep Learning and Data Assimilation for Real-Time Production Prediction in Natural Gas Wells. **ArXiv**, abs/1802.05141, 2018.

MARCELLINO, M.; STOCK, J. H.; WATSON, M. W. A comparison of direct and iterated multistep AR methods for forecasting macroeconomic time series. **Journal of Econometrics**, v. 135, n. 1-2, p. 499–526, 11 2006. ISSN 03044076.

MATTAX, C. C.; DALTON, R. L. Reservoir Simulation. **Journal of Petroleum Technology**, v. 42, n. 06, 6 1990. ISSN 0149-2136.

MLflow Project. **Multistep Workflow Example**. 2023. Available from Internet: <https://github.com/mlflow/mlflow/tree/7b80e443774ba80246cb8c45774992b0b62b241b/examples/multistep_workflow#multistep-workflow-example>.

MOCZYDLOWER, B.; FIGUEIREDO, F. P.; PIZARRO, J. O. S. A. Libra extended well test - An innovative approach to de-risk a complex field development. In: **Proceedings of the Annual Offshore Technology Conference**. [S.l.]: Offshore Technology Conference, 2019. v. 2019-May. ISBN 9781613996416. ISSN 01603663.

MUSKAT, M. The flow of homogeneous fluids through porous media. **International series in physics, York The Mapple Press Company**, 1946.

NETER, J.; WASSERMAN, W.; KUTNER, M. H. **Applied linear regression models**. [S.l.]: Richard D. Irwin, 1983.

NGUYEN, T.; GOSINE, R. G.; WARRIAN, P. A Systematic Review of Big Data Analytics for Oil and Gas Industry 4.0. **IEEE Access**, v. 8, p. 61183–61201, 2020. ISSN 2169-3536.

NING, Y.; KAZEMI, H.; TAHMASEBI, P. A comparative machine learning study for time series oil production forecasting: ARIMA, LSTM, and Prophet. **Computers & Geosciences**, Pergamon, v. 164, p. 105126, 7 2022. ISSN 0098-3004.

NNANNA, E. J.; AJIENKA, J. A. Critical Success Factors for Well Stimulation. In: **All Days**. [S.l.]: SPE, 2005.

OMRANI, S. P. et al. Deep Learning and Hybrid Approaches Applied to Production Forecasting. In: **Abu Dhabi International Petroleum Exhibition & Conference**. [S.l.: s.n.], 2019.

PAL, M. On application of machine learning method for history matching and forecasting of times series data from hydrocarbon recovery process using water flooding. **Petroleum Science and Technology**, 7 2021. ISSN 1091-6466.

PELEKIS, S. et al. DeepTSF: Codeless machine learning operations for time series forecasting. 7 2023. Available from Internet: <https://arxiv.org/abs/2308.00709v1>.

PIVETTA, M. V. L. et al. A Systematic Evaluation of Machine Learning Approaches for Petroleum Production Forecasting. In: **2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI)**. [S.l.]: IEEE, 2023. ISBN 979-8-3503-9744-4.

RASHEED, A.; SAN, O.; KVAMSDAL, T. Digital twin: Values, challenges and enablers from a modeling perspective. **IEEE Access**, Institute of Electrical and Electronics Engineers Inc., v. 8, p. 21980–22012, 2020. ISSN 21693536.

RAZAK, S. M. et al. Transfer Learning with Recurrent Neural Networks for Long-term Production Forecasting in Unconventional Reservoirs. In: **Proceedings of the 9th Unconventional Resources Technology Conference**. Tulsa, OK, USA: American Association of Petroleum Geologists, 2021. ISBN 978-0-9912144-9-5.

ROVINA, P. S. et al. Extend well test EWT libra project overview and technological highlights. In: **Proceedings of the Annual Offshore Technology Conference**. [S.l.]: Offshore Technology Conference, 2019. v. 2019-May. ISBN 9781613996416. ISSN 01603663.

RWECHUNGURA, R.; DADASHPOUR, M.; KLEPPE, J. Advanced history matching techniques reviewed. **SPE Middle East Oil and Gas Show and Conference, MEOS, Proceedings**, Society of Petroleum Engineers (SPE), v. 3, p. 1729–1747, 2011.

SAGHEER, A.; KOTB, M. Time series forecasting of petroleum production using deep LSTM recurrent networks. **Neurocomputing**, v. 323, 1 2019. ISSN 09252312.

SAIKHU, A.; ARIFIN, A.; FATICHAH, C. Correlation and Symmetrical Uncertainty-Based Feature Selection for Multivariate Time Series Classification. **International Journal of Intelligent Engineering and Systems**, v. 12, n. 3, p. 129–137, 6 2019. ISSN 21853118.

SEMERARO, C. et al. Digital twin paradigm: A systematic literature review. **Computers in Industry**, v. 130, p. 103469, 9 2021. ISSN 01663615.

SILVA, F. P. et al. Libra digital: An integrated view. In: **Offshore Technology Conference Brasil 2019, OTCB 2019**. [S.l.]: Offshore Technology Conference, 2020. ISBN 9781613996713.

SILVA, L. M. D.; AVANSI, G. D.; SCHIOZER, D. J. Development of proxy models for petroleum reservoir simulation: a systematic literature review and state-of-the-art. **International Journal of Advanced Engineering Research and Science**, v. 7, n. 10, 2020. ISSN 23496495.

SIRCAR, A. et al. Digital twin in hydrocarbon industry. **Petroleum Research**, v. 8, n. 2, p. 270–278, 6 2023. ISSN 20962495.

STEFANI, J. D. **Towards multivariate multi-step-ahead time series forecasting : A machine learning perspective**. Thesis (PhD), 2 2022.

SUN, Y. et al. Using causal discovery for feature selection in multivariate numerical time series. **Machine Learning**, v. 101, n. 1-3, p. 377–395, 10 2015. ISSN 0885-6125.

TAO, F.; ZHANG, M. Digital Twin Shop-Floor: A New Shop-Floor Paradigm Towards Smart Manufacturing. **IEEE Access**, v. 5, p. 20418–20427, 2017. ISSN 2169-3536.

TARIQ, Z. et al. A systematic review of data science and machine learning applications to the oil and gas industry. **Journal of Petroleum Exploration and Production Technology**, v. 11, n. 12, p. 4339–4374, 12 2021. ISSN 2190-0558.

TASHMAN, L. J. Out-of-sample tests of forecasting accuracy: an analysis and review. **International Journal of Forecasting**, v. 16, n. 4, p. 437–450, 10 2000. ISSN 01692070.

THAVARAJAH, R. et al. A Deep Learning Framework for Multi-Horizon Probabilistic Production Forecasting in Unconventional Reservoirs. OnePetro, 6 2022. Available from Internet: </URTECONF/proceedings-abstract/22URTC/3-22URTC/489224>.

WANASINGHE, T. R. et al. **Digital Twin for the Oil and Gas Industry: Overview, Research Trends, Opportunities, and Challenges**. Institute of Electrical and Electronics Engineers Inc., 2020. 104175–104197 p. Available from Internet: <https://ieeexplore.ieee.org/document/9104682/>.

WANG, F. et al. Field Application of Deep Learning for Flow Rate Prediction with Downhole Temperature and Pressure. In: **International Petroleum Technology Conference**. [S.l.]: IPTC, 2021.

WANG, Q.-G.; LI, X.; QIN, Q. Feature Selection for Time Series Modeling. **Journal of Intelligent Learning Systems and Applications**, v. 05, n. 03, p. 152–164, 2013. ISSN 2150-8402.

WANG, W. et al. A review of analytical and semi-analytical fluid flow models for ultra-tight hydrocarbon reservoirs. **Fuel**, v. 256, 11 2019. ISSN 00162361.

WATANABE, S. **Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance**. 2023.

WERNECK, R. d. O. et al. Data-driven deep-learning forecasting for oil production and pressure. **Journal of Petroleum Science and Engineering**, Elsevier, v. 210, p. 109937, 3 2022. ISSN 0920-4105.

YAN, M. et al. Machine Learning and the Internet of Things Enable Steam Flood Optimization for Improved Oil Production. **arXiv preprint arXiv:1908.11319**, 2019.

YETEN, B. et al. Decision analysis under uncertainty for smart well deployment. **Journal of Petroleum Science and Engineering**, v. 44, n. 1-2, p. 175–191, 10 2004. ISSN 09204105.

APPENDIX A — RESUMO EXPANDIDO EM PORTUGUÊS

A.1 Introdução

A adoção da mentalidade da Indústria 4.0 na indústria de Petróleo e Gás (O&G) e os avanços em tecnologia de aquisição, transferência e armazenamento de dados têm levado a um aumento na instrumentação de poços e instalações de produção (NGUYEN; GOSINE; WARRIAN, 2020), resultando em uma maior disponibilidade de medições em tempo real das condições de reservatório e poço, como pressão e temperatura em diferentes segmentos de poço, operações de válvulas e taxas de fluidos. Estimar o desempenho futuro de produção de poços de petróleo, uma tarefa crucial no gerenciamento de campos de petróleo (TARIQ et al., 2021), pode ser alcançado tirando proveito de todos esses dados, já que seu valor é importante para tomada de decisões, como mudanças nas configurações de ativos para maximizar a produção e minimizar os riscos.

Os Gêmeos Digitais (GDs), uma tecnologia de ponta da indústria 4.0, têm uma ampla gama de aplicações no domínio de O&G, incluindo o gerenciamento de desempenho de ativos (WANASINGHE et al., 2020). Os GDs oferecem uma maneira concisa de lidar com o problema da comunicação de dados de sensores entre ativos, manutenção de histórico e execução de cenários hipotéticos em grandes instalações de produção de petróleo. Eles proporcionam às empresas um ambiente digital onde diversos cenários e estratégias de resolução de problemas podem ser explorados sem interromper as operações de reservatório, poço ou plataforma. Os GDs têm como objetivo criar uma réplica virtual de um ativo real usando múltiplos dispositivos de aquisição de dados e modelos computacionais que geram dados e ajudam a orientar as decisões sobre o ativo físico. Um sistema de previsão orientado por dados atuaria como um componente dessa camada de GD, alimentando-se de uma camada de dados e produzindo previsões a partir de cenários reais ou hipotéticos, com base nas configurações atuais e futuras planejadas da instalação.

A estimativa da produção futura de petróleo em poços é principalmente fundamentada na modelagem mecanicista do escoamento do fluido através da rocha do reservatório e das tubulações do poço. Tradicionalmente, diferentes tipos de métodos baseados em modelo e "simuladores de reservatório", são comumente empregados para simular dinâmicas de longo prazo de um reservatório completo e estimar a produção nos poços. Estes incluem métodos *numéricos*, *analíticos* e de *curva de declínio de produção*, sendo

os modelos numéricos os mais comuns (LI et al., 2022a). Além da simulação de todo o reservatório, podemos estimar o volume de produção considerando as especificações dos equipamentos da instalação e as medidas relacionadas ao poço. Os chamados "simuladores de linhas de fluxo" são projetados para entender a dinâmica do fluxo dentro do poço e da infraestrutura de superfície. Eles consideram temperaturas, pressões, medidas de abertura de válvulas e dados de conclusão próximos a ou no poço e inferem taxas de produção usando modelos mecanicistas do fluxo de fluido do reservatório, a dinâmica termo-hidráulica do fluido dentro das linhas de fluxo da instalação, as propriedades do fluido e as operações de válvulas restritivas de fluxo (BIKMUKHAMETOV; JÄSCHKE, 2020).

Os métodos descritos acima têm desvantagens, como dependência da experiência profissional, disponibilidade de dados, custos de aquisição de dados e custos computacionais. Além disso, eles são baseados em suposições sobre a física dos fluidos ou simplificações excessivas (BALAJI et al., 2018). Essas desvantagens limitam sua aplicabilidade em certos contextos. Por exemplo, na geração rápida de cenários hipotéticos de produção a curto prazo. É improvável que se encontre a resposta em um curto espaço de tempo usando modelos numéricos mais complexos. Além disso, as restrições de aquisição de dados podem ser tais que nenhum modelo numérico confiável exista para aquele campo específico.

Muitos algoritmos foram propostos para simplificar a estimativa da produção de petróleo sem depender de métodos complexos. Métodos de *soft computing*, particularmente aqueles que se baseiam em algoritmos de Aprendizado de Máquina (AM), podem usar dados do sistema de produção para inferir comportamentos complexos do sistema. Esses métodos aproveitam dados de séries temporais de sensores de poços, planos de produção futuros e outros dados estáticos complementares relacionados à geologia do reservatório para treinar um modelo matemático que estima a produção atual e prevê a produção futura de petróleo. Essa abordagem orientada por dados, em vez da orientada por modelo, remove ou reduz a necessidade de experiência profissional, aquisição custosa de dados geológicos, custos computacionais e paradas de poços necessárias para calibrar Medidores de Fluxo Multifásicos (MFMs).

Tendo em mente a disponibilidade crescente de dados históricos de sensores, a necessidade de serviços que permitam treinamento, execução de modelos e visualização analítica no conceito de GD, e as vantagens de modelos de AM para previsão de produção, formulamos os seguintes objetivos para este trabalho:

1. revisar e discutir estudos que propõem métodos de previsão de produção que dependem de características temporais dos dados.

2. propor e disponibilizar uma arquitetura Prova de Conceito (POC) implantável que atue como um componente de GD para previsão da produção de petróleo, e que supra os requisitos das partes interessadas.

3. propor uma metodologia de seleção de modelos concebida a partir de lacunas da literatura, e evalia-la em dados reais de produção de petróleo com modelos *off-the-shelf*.

4. usando a metodologia de seleção, comparar performance dos diferentes modelos e levantar conhecimento acerca do impacto do uso de variáveis futuras, diferentes combinações de janelas e frequências de amostragem.

A.2 Metodologia

A.2.1 Abordagem de Avaliação para Modelos de Previsão

Usando dados reais de produção de um campo petrolífero, aplicamos uma abordagem de seleção de modelo para avaliar e comparar o erro de previsão de quatro técnicas de aprendizado de máquina de séries temporais considerando ou não variáveis futuras e com várias combinações de tamanhos de janela e frequências de amostragem. Os modelos são: Regressão Linear (RL), eXtreme Gradient Boosting (XGBoost), Long Short-Term Memory network (LSTM) e Temporal Fusion Transformers (TFT).

Input: Dataset de séries temporais multivariadas D com n pontos no tempo
Require: $nested\_cross\_val$, uma função que descreve um procedimento de *day-chaining nested Cross-Validation*

```
for method in [baseline, lr, xgb, lstm, tft] do
    for freq in [1day, 1h, 10min] do
        for l, k in window_sizes do
            if l ≥k then
                avg_pred_error ← nested_cross_val(method, freq, l, k)
```

A.2.2 Arquitetura de Previsão de Petróleo baseada em Microsserviços para um Gêmeo Digital

A arquitetura proposta baseada em microsserviços marca nossa proposta para um componente que pode ser integrado em um GD. Ela consiste em um componente *back-end* para treinamento e inferência de modelos, assim como um componente *front-end* para visualização de resultados e exploração da série temporal de produção. A arquitetura foi estabelecida após reuniões com a parte interessada, e basicamente supre os requisitos:

1. Permitir o treinamento de diferentes tipos de modelos de previsão de séries temporais e possibilitar a implantação dos melhores.

2. Criar previsões de produção, cenários hipotéticos e retroanálises com base em dados históricos e dados planejados futuros da produção de petróleo e gás.

3. Considerar vários horizontes temporais (períodos futuros) para previsão e desenvolvimento de cenários hipotéticos.

4. Levar em conta diferentes frequências de amostragem (dias, horas, minutos, etc.) para previsão de produção.

5. Possibilitar tanto a previsão quanto a criação de cenários hipotéticos de forma eficiente (utilizando menos recursos computacionais) e rápida (fornecendo respostas quase em tempo real). Vale ressaltar que a eficiência e o tempo de resposta, nesse contexto, são avaliados em relação a outras soluções existentes na Libra, como a simulação numérica. Enquanto esta última oferece uma precisão satisfatória nas previsões, exige uma quantidade considerável de recursos computacionais e tempo significativo para fornecer respostas.

6. Lidar com o fluxo constante de dados, considerando que o estado do GD é continuamente atualizado com base em dados de sensores que monitoram várias propriedades na planta de produção.

A.3 Resultados e conclusão

A arquitetura proposta baseada em microsserviços consegue suprir as necessidades da parte interessada. Isto é, uma ferramenta de treinamento, monitoramento de

modelos, inferência com base em cenários hipotéticos e visualização das previsões. Ao passo que utiliza dados do sistema de produção, e de alguma forma retorna uma saída que age no processo de tomada de decisão do ativo, mostra-se também um componente adequado para GDs do setor upstream. Quanto aos resultados da metodologia de comparação entre modelos ML, a comparação das métricas de erro revelou que o XGBoost, apesar da ausência de mecanismos dedicados para lidar com sequências de séries temporais como TFT e LSTM, demonstrou desempenho geral superior na previsão da produção de petróleo. O tamanho limitado e a variabilidade dos dados podem ter contribuído para os desafios enfrentados pelos modelos de aprendizado profundo, enquanto o LR teve um bom desempenho devido à sua simplicidade. A seleção cuidadosa dos tamanhos de janela emergiu como crucial para previsões precisas, com dados históricos informando as estimativas futuras de produção. A integração de dados operacionais planejados mostrou-se essencial para a previsão, destacando o impacto das variáveis operacionais nos fenômenos transitórios nos poços e reservatórios. A seleção de características para previsão de séries temporais apresentou desafios únicos devido à temporalidade dos dados, dificultando os métodos tradicionais de seleção de características. No geral, uma abordagem meticulosa para a avaliação de modelos, considerando retrocesso, tamanho do horizonte e integração de dados operacionais, é fundamental para uma previsão eficaz da produção na indústria de petróleo e gás.

APPENDIX B — ADDITIONAL DATA

Figure B presents the statistical description of each variable that was selected from the original dataset for supporting the training of the models produced in this work.

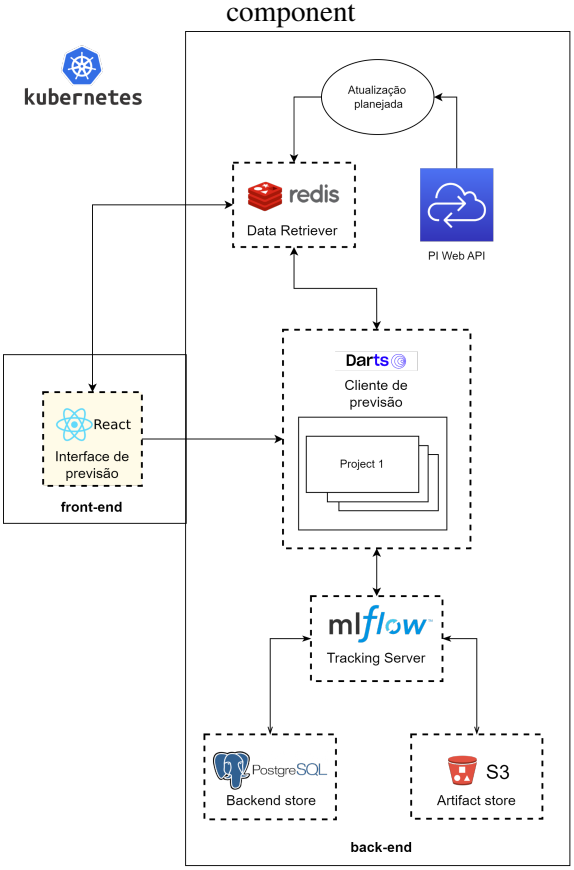
Table B.1 – Statistical description of each selected variable

	non_null_count	count	mean	std	min	25%	50%	75%	max	variance	skew	kurt
Choke Position		873960.00	63.61	46.02	0.00	9.59	99.83	99.90	100.06	2117.87	-0.50	-1.73
Choke Position 1	587675	873960.00	30.10	44.32	0.00	0.00	0.00	99.89	100.06	1964.07	0.92	-1.12
Choke Position 2	36650	873960.00	30.18	44.32	0.00	0.00	0.00	100.00	100.00	1964.01	0.93	-1.11
Choke Position 3	546156	873960.00	33.49	46.62	0.00	0.00	0.00	99.82	100.05	2173.63	0.71	-1.48
Well Oil Rate		873960.00	199.06	190.69	0.00	109.66	278.83	286.76	53135.63	36361.49	146.68	37350.39
Well Oil Rate 1	661830	873960.00	51.44	73.37	0.00	0.00	0.00	138.30	26696.72	5382.56	55.29	19905.48
Well Oil Rate 2	1023207	873960.00	132.75	190.29	0.00	0.00	142.40	279.96	53135.63	36211.83	145.35	37431.67
Well Oil Rate 3	237026	873960.00	7.08	19.09	0.00	0.00	0.00	0.00	197.07	364.58	2.39	3.92
Well Oil Rate 4	245518	873960.00	7.79	20.39	0.00	0.00	0.00	0.00	187.31	415.67	2.32	3.77
Well Oil Rate 5	23376	873960.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Well Gas Rate		873960.00	83308.73	50316.97	0.00	45419.76	117785.25	121197.15	288815.78	2531797124.48	-0.76	-1.17
Well Gas Rate 1	707356	873960.00	38742.27	50472.91	0.00	0.00	0.00	107804.35	142688.75	2547515097.79	0.64	-1.49
Well Gas Rate 2	565238	873960.00	32747.27	49811.99	0.00	0.00	0.00	104037.58	147364.85	2481233978.42	0.88	-1.22
Well Gas Rate 5	1253611	873960.00	7097.34	4813.71	0.00	0.00	9835.32	10776.14	18178.08	23171843.87	-0.66	-1.40
Well Gas Rate 4	1590852	873960.00	1321.50	4286.29	0.00	112.02	201.26	333.00	196777.45	18372294.83	9.69	201.79
Well Gas Rate 3	1408688	873960.00	3568.91	11152.77	0.00	88.49	240.12	382.95	205028.75	124384362.17	3.58	14.13
Top Annular Pressure	1702992	873960.00	607.51	42.98	0.00	586.77	595.79	638.43	2263.35	1847.53	-8.26	136.10
Top Tubing Pressure	1702963	873960.00	601.52	43.80	0.00	584.73	588.43	634.69	2454.39	1918.85	-7.65	110.84
Downstream temperature of choke	1604506	873960.00	29.64	6.40	-20.00	25.69	29.83	35.54	119.04	41.02	-1.36	8.76
Upstream Production Temperature	717691	873960.00	60.43	28.38	0.00	73.05	75.38	75.62	80.65	805.66	-1.44	0.12

APPENDIX C — IMPLEMENTATION OF THE ARCHITECTURE

The architecture is implemented into two parts: (I) the *backend*, responsible for the inference and model training endpoint, and (II) the *frontend*, which allows the user to input parameters for inference and visualize predicted values. Each rectangle in Figure C.1 represents a container running on a Kubernetes orchestration platform. Containers requiring data persistence (backend and artifact storage) have attached a Persistent Volume Claim (PVC).

Figure C.1 – Architectural diagram of the solution consisting of a frontend and a backend component



C.1 Training Backend

The training backend is based on a multi-step workflow (MLflow Project, 2023) in which each step of the training pipeline (from loading the dataset to inference) is treated as its own encapsulated execution. It's divided up into Forecasting Client and Tracking Server.

C.1.1 Forecasting Client

This container performs model training and inference, with the main dependencies being the Python libraries **Darts** and **MLFlow**. The Darts library was chosen because it provides a ready-made framework for time series forecasting tasks, including constructing input datasets and pre-built forecasting models using *PyTorch* and *scikit-learn*. We adopted the *Projects* of MLFlow. This code packaging format allows for the creation of *pipelines*, execution by the command line, and better control of dependencies for each task. Two *Projects* are currently implemented: training and inference.

1. **Model Training**

   We use a command-line interface (CLI), similar to (PELEKIS et al., 2023), where model training parameters are concatenated into a `mlflow run` command to be executed in the container.

   Model training involves:

   (a) Loading training data

   The container fetches training data (time series) from another service, called *Data Retriever*, which stores the latest time series data in the PI data management system.

   (b) Applying variable standardization method (scaling)

   Optionally, it deals with normalizing or standardizing the variables in the training dataset. Any *scaler* from the *sklearn.preprocessing* library can be used.

   (c) Preprocessing training data

   Optionally, it handles the removal of outliers, filling empty spaces, and resampling to other frequencies of the input series.

   (d) Training and evaluating the model using Cross-Validation

   Model training is done via the command line within the *Darts Client* container. The command follows the template below:

```
$ mlflow run -e train . -P darts_model={
    ↪ NaiveMovingAverage | TFT} -P num_splits=int -P
    ↪ optuna_trials=int -P freq={10min | 1h | 1d} -P
    ↪ input_chunk_length=int -P output_chunk_length=int
```

```
        ↪ -P hyperparams_entrypoint=str -P device={gpu | cpu
        ↪ } -P targets=list -P future_cov=list -P past_cov=
        ↪ list -P fill={linear_interpol | ffill} -P
        ↪ outlier_removal={std_dev_12h | std_dev_global} -P
        ↪ scaler={MinMaxScaler} --env-manager=local --
        ↪ experiment-name=str
```

(e) Recording metrics and artifacts in the *tracking service*

MLFlow, during the training execution, stores metrics, training parameters, and artifacts in the relational database or the *artifact store*. These records are later loaded during inference.

2. **Prediction**

The container serves a RESTful API with FastAPI that receives requests with parameters:

origin_timestamp: datetime ISO 8601
forecast_length: integer (1, 168)
frequency: [10min | 1h | 1d]
choke: integer (0,100)

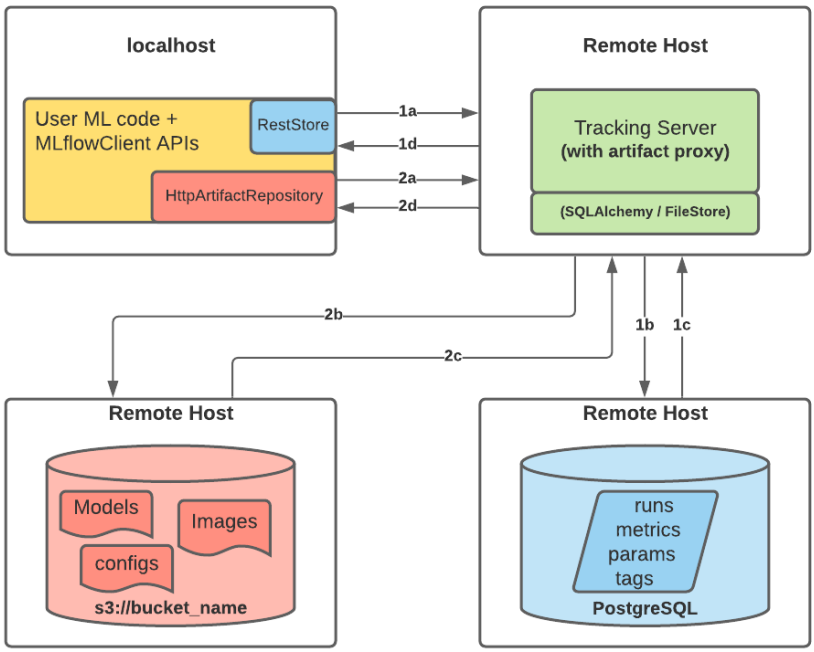
After receiving the request, the MLFlow Forecasting Project is executed, and an array in JSON format with the predicted values is returned to the frontend. It is also possible to make inferences via the command line using the following template:

```
$ mlflow run -e backcast . -P pyfunc_model_folder=str -P
    ↪ forecast_horizon=int -P forecast_date=datetime -P freq
    ↪ ={10min | 1h | 1d} -P choke=str --env-manager=local --
    ↪ experiment-name=str
```

C.1.2 Tracking Server

The MLflow Tracking Server is a proxy server to load and save model metrics and other artifacts. MLFlow uses a database and an *artifact store* to handle persistent files. The MySQL database stores execution information, metrics, parameters, and tags of the models. The artifact storage, an S3-compatible storage, stores artifacts, including model files, configurations, and images. The administrator can configure the access policy for each storage type. Figure C.2 diagrammatically shows the communication between the forecasting client, *tracking server*, and databases.

Figure C.2 – Communication between MLFlow client, database, and artifact store



Source: (MLflow Project, 2023)

The tracking server also serves a web interface where each model training execution, called *runs*, can be viewed. In addition to maintaining the record of each execution, the interface allows us to view metrics and build comparative charts. Figure C.3 shows the *runs* made for each model training. The hierarchical model of *runs* follows this format:

- Final Model
  - Best model from split 1
    - Hyperparameter trial 1
    - Hyperparameter trial 2
    - Hyperparameter trial 3
    - ...
  - Best model from split 2
    - ...
    - ...
  - Best model from split 3
    - ...
  - ...

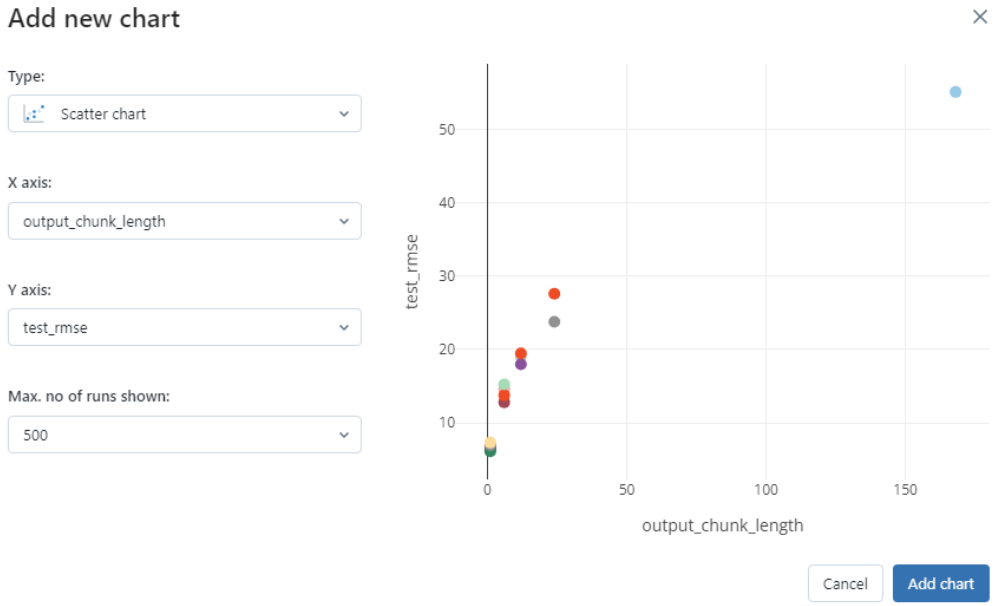
Figure C.4 illustrates the creation in the MLFlow interface of a scatter plot that shows the change in model accuracy with the increase in the *lookback* window (out-

Figure C.3 – MLFlow Interface Showing *Runs* from a Specific Training

			Metrics		
Run Name	Created	Duration	test_rmse	train_loss	val_loss
lyrical-dog-517	1 day ago	4.8h	71.17	0.367	1.039
agreeable-wasp-27	22 hours ago	2.9h	91.13	0.202	0.406
worried-squid-129	20 hours ago	17.3min	84.06	0.133	0.285
unleashed-cod-79	20 hours ago	2.1min	-	-	1.929
aged-pug-587	20 hours ago	19.8min	90.43	0.149	0.238
ambitious-ray-566	20 hours ago	3.1min	-	0.449	0.418
amazing-grouse-783	20 hours ago	51.5s	-	-	4.258
silent-finch-715	21 hours ago	33.6min	101.7	0.164	0.258
magnificent-sow-807	21 hours ago	14.2min	77.42	0.192	0.3
masked-chimp-610	22 hours ago	39.1min	95.94	0.182	0.291
glamorous-hen-238	22 hours ago	17.8min	93.23	0.184	0.383
upbeat-duck-169	22 hours ago	18.3min	175.3	1	0.981
bold-bass-43	23 hours ago	1.2h	81.52	0.421	1.482
stately-boar-933	1 day ago	36.5min	40.85	0.479	1.228

put_chunk_length).

Figure C.4 – Creation of a Scatter Plot: Error vs. Lookback Size



C.2 Prediction Front-end

The prediction frontend provides the end user with a prediction interface for models loaded in the backend. The user can (I) select a sampling frequency (10 minutes, 1 hour, or 1 day), (II) choose the starting time on the graph for the prediction, (III) select

the horizon size to be predicted, and (IV) choose the planned *choke* value for the future period. Currently, the prediction task in the interface also works as a “backcaster”, allowing the user to click on a point in the past on the timeline and either use the data from that historical period as input to the model or consider the future *choke* value determined by the user.

Figure C.5 – Front-end Interface

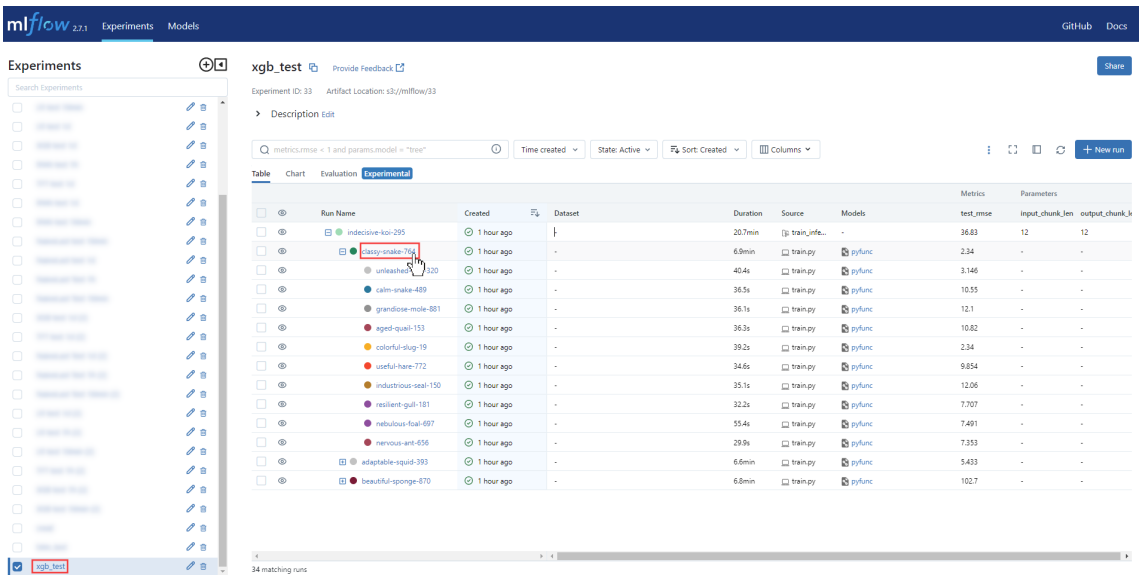
APPENDIX D — TRAINING AND VISUALIZATION USE CASE

In this section, we describe an example of using the application, from model training to visualization of predictions by the end user.

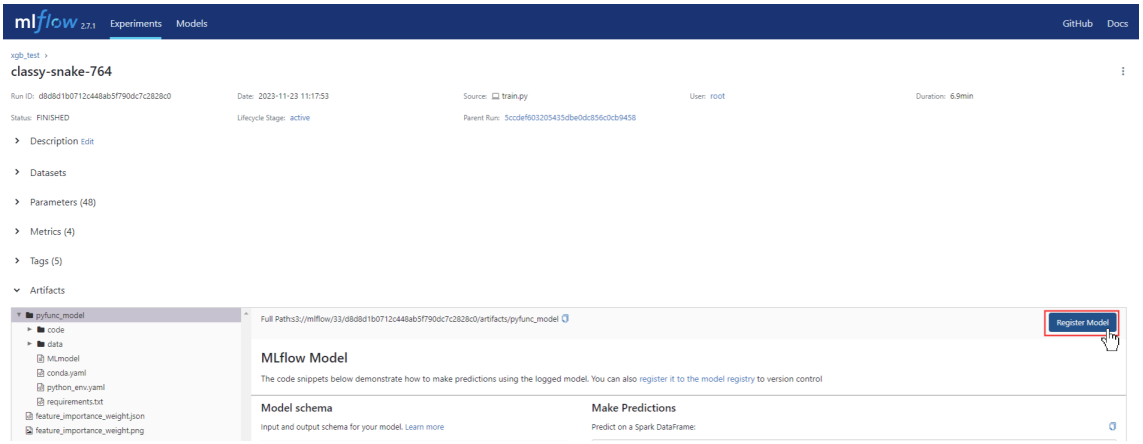
1. *Actor 1* accesses the backend machine running the prediction client and executes the following command:

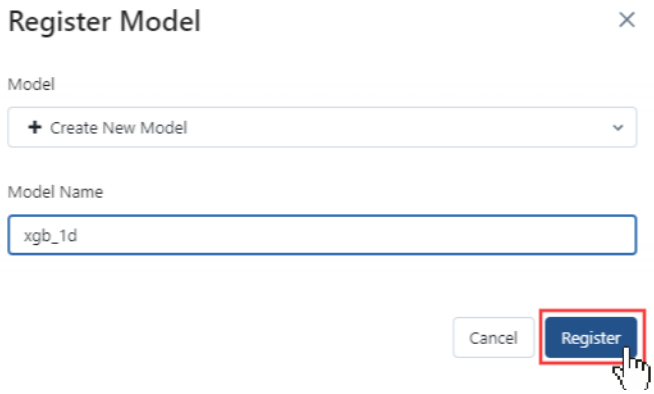
```
$ mlflow run -e train . -P darts_model=XGB -P num_splits=3 -P
    ↪ optuna_trials=10 -P freq=1d -P input_chunk_length=12 -P
    ↪ output_chunk_length=24 -P hyperparams_entrypoint=xgb_hourly
    ↪  -P device=gpu -P targets=RATE_OIL_PROD -P future_cov=CHOKE
    ↪  -P past_cov=DOWNSTREAM_TEMP_CHOKE, TOP_ANNULAR_PRESSURE,
    ↪ TOP_TUBING_PRESSURE, UPSTREAM_PRODUCTION_TEMPERATURE -P
    ↪ fill=linear_interpol -P outlier_removal=std_dev_12h -P
    ↪ scaler=MinMaxScaler --env-manager=local --experiment-name=
    ↪ xgb_test
```

2. *Actor 1* checks the training metrics for each run, the average of all splits, and generates comparative graphs.
3. *Actor 1* clicks on the best model.

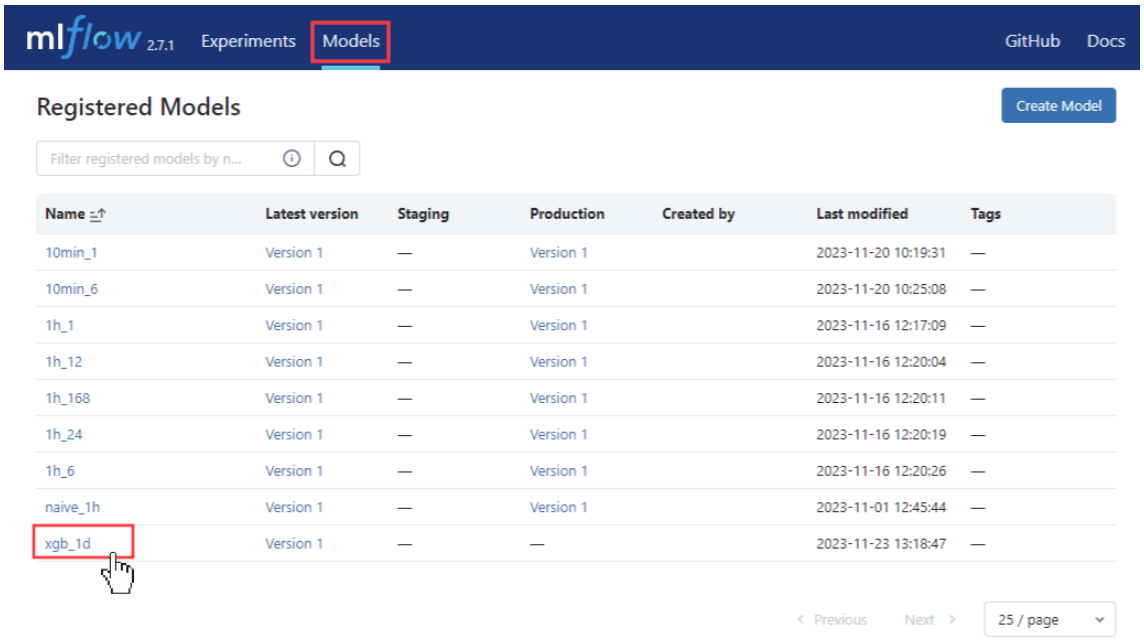


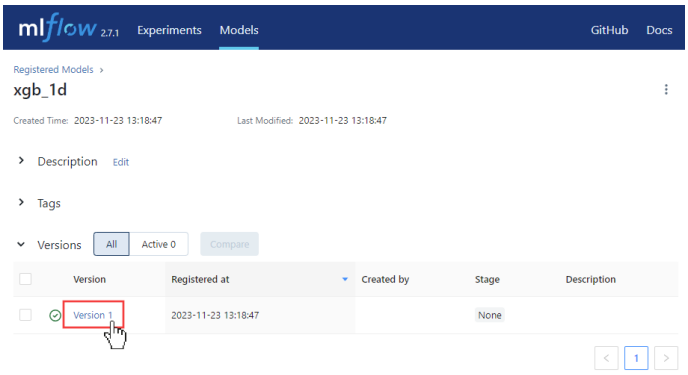
4. *Actor 1*, within the best model, clicks the "Register Model" button, names the model as xgb_1d, and registers it for later inference.

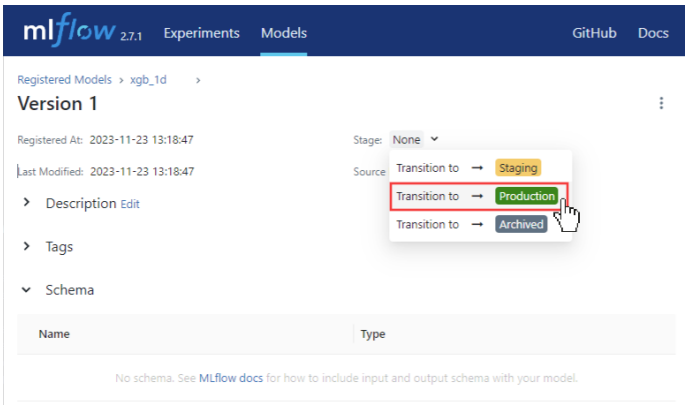




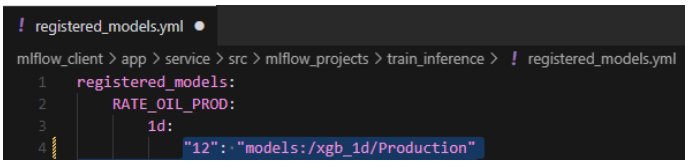
5. *Actor 1* selects this newly registered model in the "Models" tab and puts it into production.







6. *Actor 1* inserts the model descriptor into the *registered_models.yml* file.



7. *Actor 2* uses the model for inference by selecting, in the interface, the sampling frequency "1d" and the horizon size of 12.

Figure D.1 – Front-end Interface