Pontifícia Universidade Católica
do Rio de Janeiro

Isabel Figueira de Abreu Gonçalves

Machine learning strategies to predict oil field performance as time-series forecasting

Dissertação de Mestrado

Thesis presented to the Programa de Pós–graduação em Matemática, do Departamento de Matemática da PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Matemática.

Advisor : Prof. Sinesio Pesco
Co-advisor: Dr Thiago de Menezes Duarte e Silva

Rio de Janeiro
April 2023

Pontifícia Universidade Católica
do Rio de Janeiro

Isabel Figueira de Abreu Gonçalves

Machine learning strategies to predict oil field performance as time-series forecasting

Thesis presented to the Programa de Pós–graduação em Matemática da PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Matemática. Approved by the Examination Committee:

Prof. Sinesio Pesco
Advisor
Departamento de Matemática – PUC-Rio

Dr Thiago de Menezes Duarte e Silva
Schlumberger Serviços de Petróleo - Matriz

Prof. Abelardo Borges Barreto Jr.
Departamento de Matemática – PUC-Rio

Dr. Emilio Jose Rocha Coutinho
Petróleo Brasileiro - Rio de Janeiro - Matriz

Dr. Angelica Nardo Caseri
Energisa

Dr. Orlando Fonseca Guilarte
Departamento de Matemática – PUC-Rio

Rio de Janeiro, April the 28th, 2023

Isabel Figueira de Abreu Gonçalves

Majored in mathematics by Fluminense Federal University in 2021.

Bibliographic data

F A Gonçalves, Isabel

Machine learning strategies to predict oil field performance as time-series forecasting / Isabel Figueira de Abreu Gonçalves; advisor: Sinesio Pesco; co-advisor: Thiago de Menezes Duarte e Silva. – 2023.

100 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Matemática, 2023.

Inclui bibliografia

1. Matemática – Teses. 2. LSTM. 3. Machine Learning. 4. Production prediction. 5. Neural networks. I. Pesco, Sinesio. II. Silva, Thiago M. D.. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Matemática. IV. Título.

CDD: 510

Acknowledgments

I would like to thank my parents Marcos and Vanda for all support and love. I am blessed to have you as my parents.

To my brother Vinicius, for all the faith you had in me. I do not know a better person in this world.

To my advisor Professor Sinesio Pesco, my co-advisor Thiago Silva and Professor Abelardo Barreto Jr. for all their guideness, patience and understanding.

To Professor Simone Dantas, my first and always advisor. Thank you for everything.

To my friends, specially Thais, Braian, Guilherme and Alice for just being there. Life is better with you around.

I would like to thank the CENPES reservoir research team for all discussion and suggestion that enriched this work.
I thank Petrobras for the financial support.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001

Abstract

F A Gonçalves, Isabel; Pesco, Sinesio (Advisor); Silva, Thiago M. D. (Co-Advisor). **Machine learning strategies to predict oil field performance as time-series forecasting**. Rio de Janeiro, 2023. 100p. Dissertação de Mestrado – Departamento de Matemática, Pontifícia Universidade Católica do Rio de Janeiro.

Precisely forecasting oil field performance is essential in oil reservoir planning and management. Nevertheless, forecasting oil production is a complex nonlinear problem due to all geophysical and petrophysical properties that may result in different effects with a bit of change. Thus, all decisions to be made during an exploitation project must consider different efficient algorithms to simulate data, providing robust scenarios to lead to the best deductions. To reduce the uncertainty in the simulation process, recent studies have efficiently introduced machine learning algorithms for solving reservoir engineering problems since they can extract the maximum information from the dataset. This thesis proposes using two machine learning techniques to predict the daily oil production of an offshore reservoir. Initially, the oil rate production is considered a time series and is pre-processed and restructured to fit a supervised learning problem. The Random Forest model is used to forecast a one-time step, which is an extension of decision tree learning, widely used in regression and classification problems for supervised machine learning. Regardless, the restrictions of this approach lead us to a more robust model, the LSTM RNN's, which are proposed by several studies as a suitable deep learning technique for time series modeling. Various configurations of LSTM RNN's were constructed to implement single-step and multi-step oil rate forecasting and down-hole pressure was incorporated to the inputs. For testing the robustness of the proposed models, we use four different datasets, three of them synthetically generated and one from a public real dataset, the Volve oil field, as a case study to conduct the experiments. The results indicate that the Random Forest model could sufficiently estimate the one-time step of the oil field production, and LSTM could handle more inputs and adequately estimate multiple-time steps of oil production.

Keywords

LSTM; Machine Learning; Production prediction; Neural network.

Resumo

F A Gonçalves, Isabel; Pesco, Sinesio; Silva, Thiago M. D.. **Predição da performance de reservatórios de petróleo utilizando estratégias de aprendizado de máquina para séries temporais**. Rio de Janeiro, 2023. 100p. Dissertação de Mestrado – Departamento de Matemática, Pontifícia Universidade Católica do Rio de Janeiro.

Prever precisamente a produção de óleo é essencial para o planejamento e administração de um reservatório. Entretanto, prever a produção de óleo é um problema complexo e não linear, devido a todas as propriedades geofísicas que com pequenas variações podem resultar em differentes cenários. Além disso, todas as decisões tomadas durante a exploração do projeto devem considerar diferentes algoritmos para simular dados, fornecer cenários e conduzir a boas deduções. Para reduzir as incertezas nas simulações, estudos recentes propuseram o uso de algoritmos de aprendizado de maquina para solução de problemas da engenharia de reservatórios, devido a capacidade desses modelos de extrair o maxiomo de informações de um conjunto de dados. Essa tese propôe o uso ed duas tecnicas de machine learning para prever a produção diaria de óleo de um reservatório. Inicialmente, a produção diária de óleo é considerada uma série temporal, é pré-processada e reestruturada como um problema de aprendizado supervisionado. O modelo Random Forest, uma extensão das arvores de decisão muito utilizado em problemas de regressão e classificação, é utilizado para predizer um passo de tempo a frente. Entretanto, as restrições dessa abordagem nos conduziram a um modelo mais robusto, as redes neurais recorrentes LSTM, que são utilizadas em varios estudos como uma ferramenta dee aprendizado profundo adequada para modelagem de séries temporais. Várias configurações de redes LSTM foram construidas para implementar a previsão de um passo de tempo e de multiplos passos de tempo, a pressão do fundo de poço foi incorporada aos dados de entrada. Para testar a eficacia dos modelos propostos, foram usados quatro conjunto de dados diferentes, três gerados sintéticamente e um conjunto de dados reais do campo de produção VOlve, como casos de estudo para conduzir os experimentos. Os resultados indicam que o Random Forest é suficiente para previsões de um passo de tempo da produção de óleo e o LSTM é capaz de lidar com mais dados de entrada e estimar multiplos passos de tempo da produção de óleo.

Palavras-chave

LSTM; Machine Learning; Production prediction; Neural networks.

Table of contents

List of figures

List of tables

List of Abreviations

ARIMA – Autoregressive Integrated Moving Average

CART – Classification and Regression Trees

GN – Gauss-Newton

GPU – Graphics processing unit

LSTM – Long Short Term Memory

LN – Levenberg-Marquardt

LS – Least Squares

MA – Moving Average

MAE – Mean Absolut Error

PCA – Principal component analysis

RF – Random Forest

RNN – recurrent neural network

1
Introduction

Oil field operation is a critical aspect of the energy industry that involves the exploration, production, and management of oil and gas resources. It plays a crucial role in the world's energy challenges, powering everything from transportation to electricity generation [1]. Despite efforts to transition to renewable energy sources, oil and gas remain essential for the foreseeable future. The industry is vital in global economy and provides feedstock for numerous products, including plastics, fertilizers, and pharmaceuticals.

With the ever-increasing demand for energy, the oil and gas industry faces significant challenges and has become a multidisciplinary field that incorporates a wide range of technologies. With the complexity of oil field operations, predicting future oil production is essential to optimize production, minimize costs and make decisions to develop and manage the reservoir. The risks involved are considerably high, demanding proper uncertainty administration during an exploitation project.

Moreover, reservoir characterization is essential when forecasting an oil deposit's performance and handling uncertainty. As a result, constructing a robust reservoir model is, therefore, an important task. The process of characterizing the reservoir may be executed by incorporating observed dynamic data from a real field in a model, which is a popular technique called history matching.

The primary tool for history matching algorithms is reservoir simulation, which demands the creation of a theoretical reservoir numerical model in which the user inputs the static properties and the simulation process computes the dynamical data as output. Reservoir simulation processes are crucial for reservoir management, which enables testing of several different production plans for forecasting. At the end of the simulation and characterization process, the model is expected to compute output dynamical data similar to the observed data. Nevertheless, reservoir characterization using history matching procedures requires many reservoir simulations and adjustments in the reservoir model properties until the output data match the observed one.

History matching algorithms [2] are widely known to be efficient in predicting reservoir dynamical properties and oil field production. Furthermore,

many studies prove that using optimization algorithms may obtain good results. We can mention the nonlinear least square methods, e.g., the Gauss-Newton and Levenberg-Marquardt algorithms [3], and the ensemble-based methods, e.g., the ensemble Kalman filter [2] and the ensemble smoother with multiple data assimilation [4]. The study of Shirangi and Emerick [5] compares the results obtained by applying the Levenberg-Marquardt (LM) and the Gauss-Newton (GN) method. Their results suggest that the LM approach could bring better results when predicting reservoir properties due to the more negligible influence of minimal singular values in the computation of the update vector compared to the GN application. Considering ensemble-based methods, the study of Silva *et al.* [6] offers a good characterization of the damage zone in a multilayered reservoir using the ensemble smoother with multiple data assimilation. Although, the algorithms often used in these studies demand complex mathematical or statistical backgrounds and advanced computational knowledge.

Time series prediction is also a challenging task, since it involves dynamical on linear data. Some algorithms are known for beeing efficient in time series forecast. ARIMA [7], which stands for Autoregressive Integrated Moving Average, is a popular time series forecasting model that has three components: the autoregressive (AR) component, the integrated (I) component, and the moving average (MA) component. Although ARIMA models are flexible and can be applied to a wide range of time series data, it assumes that the relationships between the variables are linear and stationary, which may not always be the case in real-world applications. Another technique widely used for time series predicton is Moving Average (MA)[8], that tries to predict the value of the target variable based on the average of its past values. The MA model has only one parameter, which is the number of past observations used to compute the moving average. It is a simple model, that requires little statistical knowladge to understand the process and the outcomes. However, since it is based only on the past values of the target variable, it does not take into account any external factors or trends that may influence the future values. Also, alike ARIMA, the MA model assumes that the data is stationary, which may not always be the case in practice.

A powerful forecasting tool involves using machine learning algorithms. This technique has become very popular in the last few years due to the easy manipulation and understanding of the mathematical formulation of such algorithms. Moreover, the statistical background of machine learning enables the algorithm to identify complex patterns in the dataset, which may be unfeasible when not using data-driven procedures. These algorithms are

split into two ample categories: supervised and unsupervised machine learning. We can also mention the algorithms based on reinforcement learning, which recently gained much attention.

As a result, there has been a growing interest in integrating machine learning techniques into oil field management to provide proxy models to replace the necessity of simulators in some steps of the history matching problems [9]. Machine learning algorithms can analyze large amounts of data from various sources such as production logs, sensor data, and historical records, to identify patterns and make predictions. The integration of machine learning techniques into oil field management has the potential to improve the efficiency of operations and provide more accurate information to decision-makers.

The Random Forest is a machine learning algorithm that has gained popularity in various industries, including the oil and gas. It is a type of ensemble learning, which combines multiple decision trees to produce more accurate results. In petroleum exploration, random forest has been used in various applications, including Well test planning [10], reservoir facies classification [11] and prediction of solid particle erosion [12]. Although the traditional Random Forest algorithm was not designed to handle time series data, it can be adapted for time series prediction by transforming the time series data into a set of features that can be used to train the Random Forest model.

Moreover, there are more sophisticated and complex models to handle history matching problems. The deep learning area, a subfiled of Machine Learning, presents the neural networks as a versatile and potent approach to handling complex and nonlinear relations in datasets. The Long Short-Term Memory (LSTM) neural network is a deep learning model specifically designed for modeling sequential data, capable of selectively remembering or forgetting information over time. Thus, it has been used in speech generation [13], text summarizing [14], music generation [15] and time series forecasting [16] and anomaly detection[17]. Hence, the LSTM neural networks are a promissing approach for history matching problems.

Therefore, this work presents two different data-driven solutions for daily oil production forecasting, one using the random forest and other using Long Short-Term Memory neural networks.

This thesis is segmented as follows: Section 2 supplies a summary of Machine Learning concepts and how Random Forest and LSTM works; Section 3 displays how we can adapt the time series problem to fit the desired inputs and the models constructed in this work. Moreover, we introduce four different

datasets used to evaluate the models. Finally, section 4 presents and compares the results of the proposed methods.

2
Machine Learning

The term "machine learning" was coined in 1959 by Arthur Samuel, who is known for developing the first computer program to play checkers using machine learning techniques [18]. To him, is also attributed the statement that machine learning is a *"field of study that gives computers the ability to learn without being explicitly programmed"* [19].

The beginning of Machine Learning in the 1950s was inspired by neurophysiological, biological, and psychological investigation [18]. During this time, researchers began to develop and test algorithms analogues of neurons and the first concept of a neural network, the Perceptron, was proposed in 1958 [20].

Although the advances in Machine Learning, neural networks were set aside in the 1970s due to the perceived limitations of the algorithms [21] and computing resources at the time. It was not until the 1980s, with the development of new algorithms and the availability of more powerful computing resources, that neural networks began to see a resurgence in popularity.

During the 1980s, there was a vast growth in the field and an explosion of interest in the area. The development of deep learning [22], a subset of machine learning that involves more complex architectures of neural networks, has led to breakthrough results. The term "deep" refers to the depth of the neural network since it has multiple layers. Deep neural networks are capable of processing large amounts of data and extracting more complex patterns and relations. The accessibility of powerful Graphics processing units (GPUs) and large datasets, combined with advances in neural network architectures, has enabled deep learning algorithms to achieve higher levels of accuracy.

Among several advances that came with the rise of deep learning, the development of backpropagation [23], a method for training multi-layer neural networks, was a key breakthrough that helped make neural networks a more practical approach to machine learning. This led to the development of new neural network architectures, such as the multi-layer perceptron, and the application of neural networks to a wide range of areas.

Throughout the 1990's, the advancements in computing technology lead to significant developments of Machine Learning in parallel areas of research.

The emergence of ensemble methods, an approach that combines multiple models to improve prediction accuracy, began to gain popularity. There are several types of ensemble methods, including the Random Forest [24], a popular ensemble method that combines several decision trees.

Moreover, there were advances in the neural networks models. More complex and sophisticated neural units were built to handle more complex tasks, such as the Long-Short Term Memory [25], a neural network capable of retaining information over long periods of time, improving the performance on sequential data.

Therefore, with the wide range of applications of machine learning and deep learning techniques available nowadays, there are several types of machine learning. In this chapter we describe the main concepts of Machine Learning, including Supervised Learning, Unsupervised Learning and Reinforcement Learning. Furthermore, we describe the Random Forest Algorithm and the LSTM neural network which are the techniques used in this work.

2.1 Supervised Learning

Supervised learning is a type of machine learning algorithm in which the model learns from labeled data, that is, data that has already been labeled with the correct answers. In such problems, its given a set of data $S = \{(x_i, y_i)\}_{i=1}^{n}$, where y_i is the label corresponding to x_i. The key idea behind supervised learning is to learn a function f such as $y_i = f(x_i) + \varepsilon_i$, that maps the input x_i to the output y_i and minimizes the error ε_i between the predicted and true outputs by adjusting its internal parameters.

To accomplish this, the labeled dataset S is split into two parts: a training set and a test set. The algorithm is trained on the training set, and the performance of the constructed function f is evaluated on the test set. The goal is to find the parameters that minimize the error on the test set while still generalizing well to unseen data. Supervised learning can be further divided into two categories: regression and classification. In regression, the target y_i is a continuous output variable, and in classification, the target y_i is a discrete output variable.

Some popular algorithms used in supervised learning include linear regression, logistic regression, decision trees, random forests, and neural networks. Each algorithm has its own strengths and weaknesses and is suitable for different types of problems. One of the advantages of supervised learning is that it can be used for both prediction and explanation. For example, a linear regression model can be used to predict the price of a house based on its fea-

tures, such as the number of bedrooms, bathrooms, and square footage. At the same time, it can also provide insight into which features are most important in determining the price.

However, supervised learning also has some limitations. One of the main challenges is the need for labeled data, which can be time-consuming and expensive to obtain. Additionally, the model's performance is highly dependent on the quality and representativeness of the labeled data. If the dataset is biased or incomplete, the model's predictions may be inaccurate or even harmful.

In conclusion, supervised learning is a powerful and versatile machine learning technique that has been used to solve a wide range of problems. It requires labeled data, which can be a limitation, but it also provides insight into the relationship.

2.2 Unsupervised Learning

Unsupervised learning is a type of machine learning that involves finding patterns or structure in unlabeled data without explicit supervision from a human. Unlike supervised learning, where its provided labeled data to train the algorithm, unsupervised learning involves discovering patterns and relationships within data on its own. Unsupervised learning is a crucial part of modern artificial intelligence since its a potential technique to work with great volumes of unlabeled data and discover novel patterns and relationships that may not be apparent with tradional statistics methods. Although it is a potential technique with several applications to unlabeled data, unsupervised learning can be challenging due to the lack of ground truth to compare the results to.

A common type of unsupervised learning is clustering, where data points are grouped into clusters based on their similarity. One of the most used criterion for clustering is the k-means criterion [26]. It is based on minimizing the distance of data points within the same cluster. Formally, given $\{x_1, \cdots, x_n\} \in \mathbb{R}$, we partion this points in k clusters $C_1, \cdots, C_k$. The total sum of distances

$$f(C_j) := \sum_{i=1}^{n} \min_{i \leq j \leq k} d(x_i, C_j) \tag{2-1}$$

it is called k-means loss function, and $d(x_i, C_j)$ is what characterizes the similarity, between x_i and C_j. Mathematically, this similarity is a distance, defined by a norm, usually, the Euclidean norm:

$$||x|| = \sqrt{\sum_{i=1}^{n} x_i^2} \tag{2-2}$$

Another type of unsupervised learning is dimensionality reduction, which involves reducing the number of variables or features in a dataset while still retaining as much information as possible. Dimensionality reduction is often used in data visualization and compression to help humans understand and work with large datasets. Principal component analysis (PCA) [27] is one of the most popular dimensionality reduction techniques and involves finding the directions of maximum variance in the data and projecting the data onto these directions.

Unsupervised learning can be also used for anomaly detection, where the algorithm learns to identify data points that are significantly different from the rest of the data. This can be useful in many applications, including fraud detection [28], cybersecurity [29], and predictive maintenance [30]. Anomaly detection algorithms can use clustering, density estimation, or neural networks to detect unusual patterns in the data.

Clustering, dimensionality reduction, and anomaly detection are just a few examples of the types of problems that can be solved using unsupervised learning techniques. As the amount of data generated by humans and machines continues to grow, unsupervised learning will become even more important for discovering patterns and relationships in this data.

2.3 Reinforcement Learning

Reinforcement learning [31] is a type of machine learning that involves training agents to learn optimal decision-making strategies in a particular environment. This approach, which focuses on how agents can learn through feedback, has gained popularity in recent years due to its success in various applications [32], including robotics, game theory, and recommendation systems.

Reinforcement learning involves an agent that interacts with an environment to learn a specific task or goal. The environment provides feedback in the form of rewards or punishments for the actions taken by the agent. The agent's objective is to maximize the rewards it receives by taking the optimal actions in the given environment.

The key components of a reinforcement learning system include the agent, environment, and reward system. The agent is the learning entity that takes actions based on its current state and the information it has learned from the environment. The environment is the external system in which the agent operates and receives feedback from. The reward system is the mechanism that provides the agent with feedback on the quality of its actions.

The process begins with the agent selecting an action based on its current state. The environment then provides feedback to the agent in the form of a reward or punishment, depending on the quality of the action taken. The agent then updates its policy based on the feedback received and selects the next action based on the updated policy. This process continues until the agent learns the optimal policy for the given environment.

One of the most significant advantages of reinforcement learning is that it does not require labeled data to train an agent. Instead, it relies on the feedback provided by the environment to learn the optimal policy. This makes it particularly useful in situations where it may be challenging to obtain labeled data, such as in robotics or game theory.

2.4 Random Forest Regressor

Random Forest [24] is a supervised learning algorithm with an ensemble of size N decision trees built and trained with the input training dataset. Each tree in the forest is build using the CART criterion. CART stands for Classification and Regression Trees [33] and it is a machine learning algorithm that generates binary trees, i.e., a popular data structure in computer science in which each node has exactly two children. These two edges (children) are defined as the left and right children. The splitting decision is made by using an appropriate impurity criterion. The most popular ones are defined as *Gini* or *entropy*. For regression, CART introduced variance reduction using least squares LS (Equation (2-3)) and Mean Absolute Error MAE (Equation (2-4)). In Equations (2-3) and (2-4) y_i, refers to the prediction for an instance, N is the number of instances and μ is the mean given by $\frac{1}{N}\sum_{i=1}^{N} y_i$.

$$LS = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mu)^2 \tag{2-3}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \mu| \tag{2-4}$$

One can apply this model to classification and regression problems. The main concern when using decision trees to solve real-case issues is the low variance problem [34] widely described in the machine learning literature.It means that if one slightly changes the training set input data, the output may change substantially. Moreover, it is reasonable to expect the model to present a common issue in machine learning applications called overfitting.

The overfitting [35] indicates that the model learned too much from the training data but could not generalize the result for other datasets. A traditional procedure that alleviates the low-variance problem of decision trees

is called *bagging* [36], which builds a forest of decision trees and trains each one with the input training dataset. However, instead of training all trees with the whole dataset, it draws a sample of the entire dataset and determines this small sample as the tree's input. This procedure is executed for each tree in the forest. More precisely, given N cases in the training dataset, it samples, with replacement, $k < N$ subsets among all possible cases.

In addition, another crucial technique is implemented in each tree of the forest to reduce the problems of low variance and increase the model's generalization capability, which is called *randomized subspace*. The randomized subspace is also applied to the bagging strategy for each tree in the forest. A random sample of the input features and training data is employed in each tree. It is straightforward to expect that each tree in the forest captures slightly different information from the whole dataset, constructing a forest that could provide more variability and robustness to the model. For classification problems, the result of a random forest would be the class with the most appearance in the forest. For regression, the result would be the mean of the outcome of each forest. Figure 2.1 shows a simple diagram explaining how the random forest algorithm works.

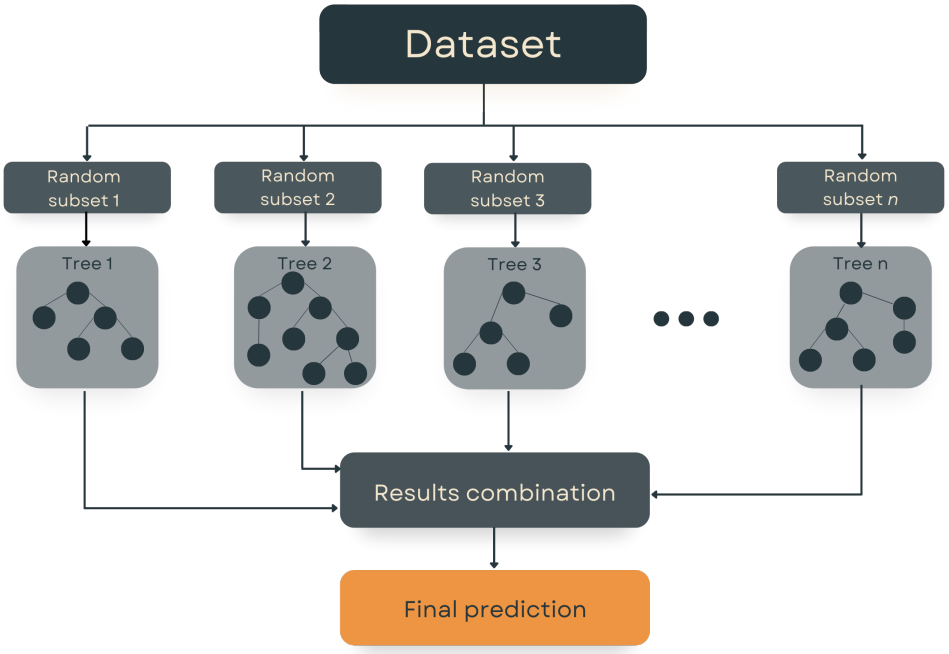

Figure 2.1: Random Forest diagram.

2.5 Neural Networks

Artificial neural networks are inspired by the functioning of the human brain and its smallest building parts, the biological neurons. A neuron, also called unity, node or cell, is an information-processing unity [37] that is essential to a neural network.

The Perceptron is the simplest neural network, and consists of one or more inputs, a neuron and only one output. Mathematically, a Perceptron takes numeric inputs $\{x_1, x_2, \cdots, x_n\}$, multiplies them with the respective weights $\{w_1, w_2, \cdots, w_n\}$ and finally add them together with the bias b, producing a result that is the input to an activation function $f(x)$, as it shows equation 2-5.

$$f(\sum_{i=1}^{n} w_i \times x_i + b) \tag{2-5}$$

The values for the weights w_i and the bias b are randomly initiated and than adjusted by a process named supervised training.

Perceptron are the building components of neural networks. The Perceptron can be understood as a neural network of one neuron. When several neurons are combined, they form the architecture of neural network. Formally, the architecture is the way the units are connected and arranged.

These neural networks can be used to handle time series predictions, but they may not always be the best choice. One of the main limitations of simple neural networks is that they may struggle to capture the complex patterns and dependencies present in time series data, especially when the data is non-linear or non-stationary. Additionally, simple neural networks are susceptible to overfitting, which can occur when the model is too complex and fits the training data too closely, leading to poor generalization performance on unseen data.

Another problem faced by neural networks is the vanishing gradient problem, specially when working with long-term dependency sequences. This is because during the backpropagation process, the gradients computed at each layer of the network are multiplied together as they are propagated backwards towards the input layer. When the gradients are very small, this multiplication results in a gradient that approaches zero, making it difficult for the model to learn the relevant features and parameters.

This problem is particularly acute in feedforward neural networks because they lack the ability to store information about previous inputs. In contrast, recurrent neural networks (RNNs) address this issue by using a recurrent connection that allows the network to maintain a "memory" of past inputs.

However, even RNNs can still face the vanishing gradient problem when the sequence of inputs is very long.

On the other hand, there are other machine learning algorithms that are specifically designed for time series analysis and can outperform simple neural networks. For example, the long short-term memory (LSTM) networks are more suitable for capturing the temporal dependencies present in time series data.

2.6 LSTM

Figure 2.2 shows the LSTM neural network architecture unrolled over time. The current time step is the cell in the middle. We use t to refer to actual time step. So to its left is the LSTM cell at the previous time step $t-1$ while the one to the right is the cell at one step ahead, $t+1$. In the illustration below, each line carries a vector from the output of one node to the inputs of others. LSTMs have a structure similar to others recurrent neural networks, but the repeating cell has a different layout. Instead of a single neural network layer, there are four gates, interacting to produce the output.

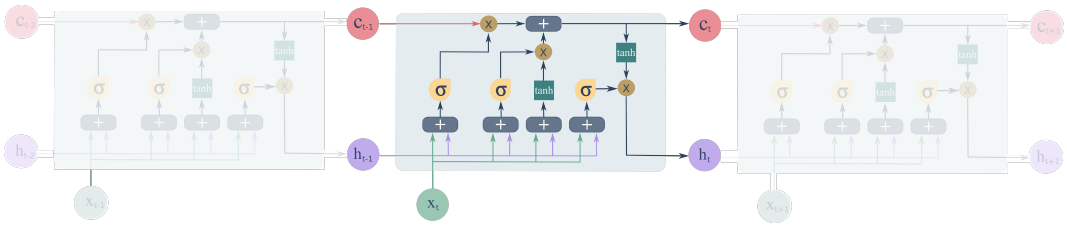

Figure 2.2: LSTM architecture unrolled over time.

Figure 2.3 shows the LSTM cell. The gates are a way to optionally let information through. They are composed out of a activation function neural net layer and a pointwise multiplication operation. The sigmoid layer output is a number between zero and one, describing how much of each information should be let through. If the output is zero, nothing is passed beyond; if the output is one, all information is passed beyond. In all equations below, the lowercase variables represent vectors, and uppercase variables represent matrices.

Each cell receives three information. In the bottom left corner, it receives the input X_t and the output from the previous time step h_{t-1}. This information are run into the four gates. The third input c_{t-1} is represented as a straight arrow through the upper part of the cells. This is the cell state and enables the LSTM to remember long term dependencies with a considerably smaller

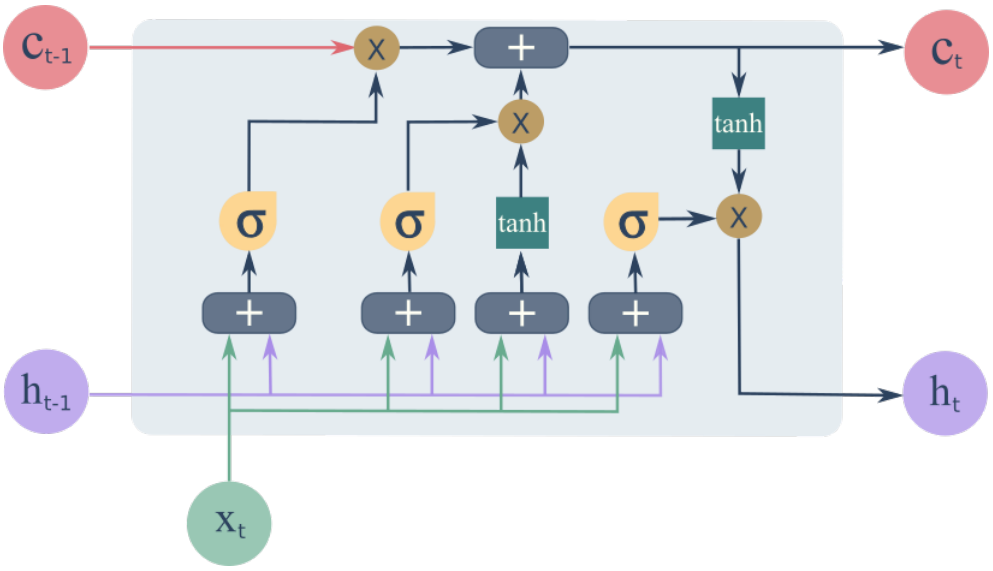

Figure 2.3: LSTM architecture unrolled over time.

chance for the vanishing and exploding gradient problems seen in traditional RNN [38].

The first gate is the forget gate, showed in Figure 2.4. It considers the current inputs X_t and the output from the previous time step h_{t-1}. The product of the current input and the weights are linear transformed by sigmoid function (Equation 2-6 to a matrix with values between 1 and 0 (Equation 2-7. Matrix W_f contain the weights of X_t and h_{t-1}. From here the cell state from the previous cell is multiplied elementwise with the forget gate. One may think of the forget gate as a filter, that erases or decreases values that we want to delete or degrade from previous cell state, the memory of LSTM.

$$\sigma(x) = \frac{1}{1+e^x} \tag{2-6}$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2-7}$$

The input gate decide how much of the new information will be included in the memory of LSTM. In Equation 2-8, the new inputs of the cell are concatenated and processed by a Sigmoid function, while in Equation 2-9 they are processed by a hyperbolic tangent function. The results of each equation are multiplied point by point, and the new information are ready to update the memory of LSTM.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2-8}$$

$$\tilde{C} = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2-9}$$

The cell state is not more than a vector in a mathematical sense. It can

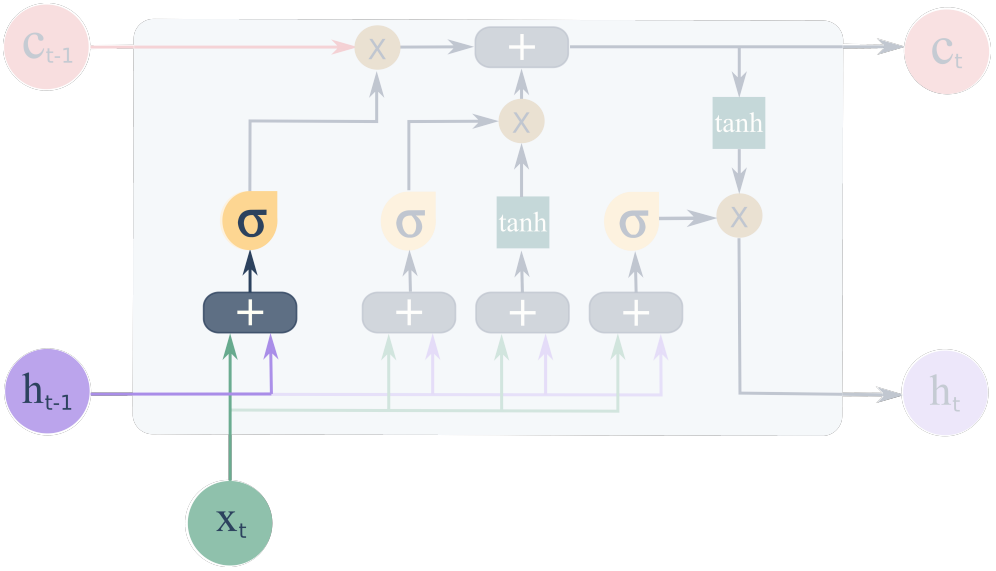

Figure 2.4: The Forget gate.

be thought of as way for information that runs through the whole chain of cells with only some linear interactions, as shows Figure 2.6. It allows the LSTM to remember long term input dependencies. It is possible to read, write and delete information to this internal memory. The key to solving the vanishing gradient problem is that the new information is added and not multiplied to the cell state, as shows Equation 2-10 Addition distribute gradients equally and the chain rule does not apply within the backpropagation.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C} \tag{2-10}$$

The "Cell State" and the "Hidden State" have different functions. The cell state is meant to aggregate data from all previous time-steps that have been processed, while the hidden state is meant to encode some characterization of the most recent time-step's data. It is important to note that the hidden state is not equal the output or prediction of the previous time-step.

Finally it is time to produce a output h_t. The output gate will run a sigmoid layer to the inputs (Equation 2-11) and an hyperbolic tangent into the cell state (Equation 2-12). Then, both results are multiplied point y point and the output is passed ahead.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2-11}$$

$$h_t = o_t \cdot tanh(C_t) \tag{2-12}$$

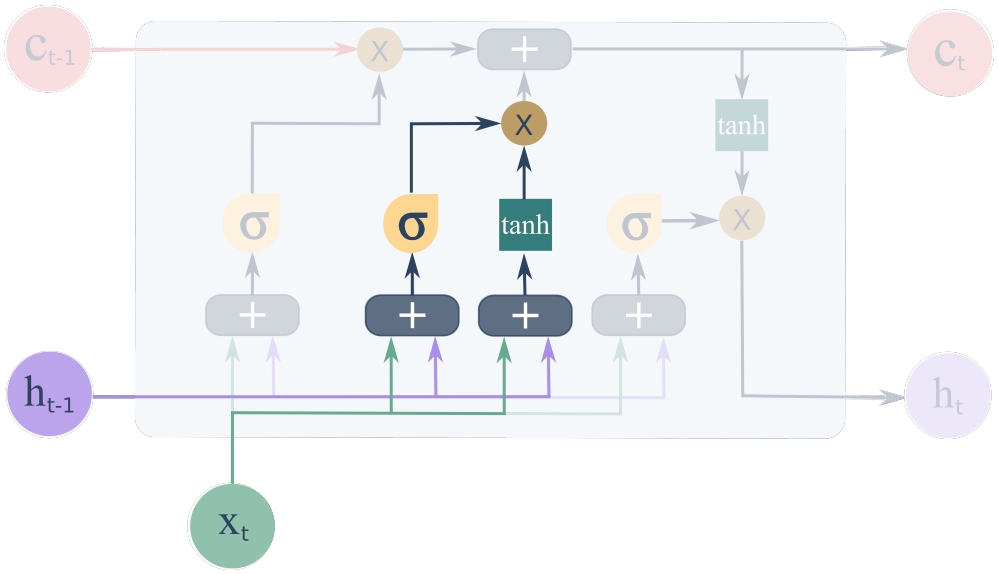

Figure 2.5: The Input gate.

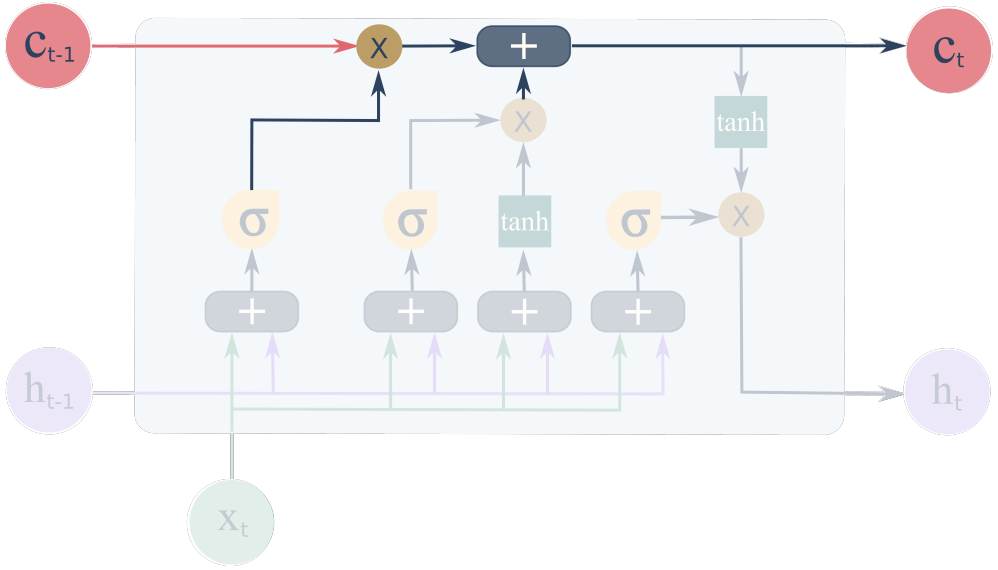

Figure 2.6: The cell state.

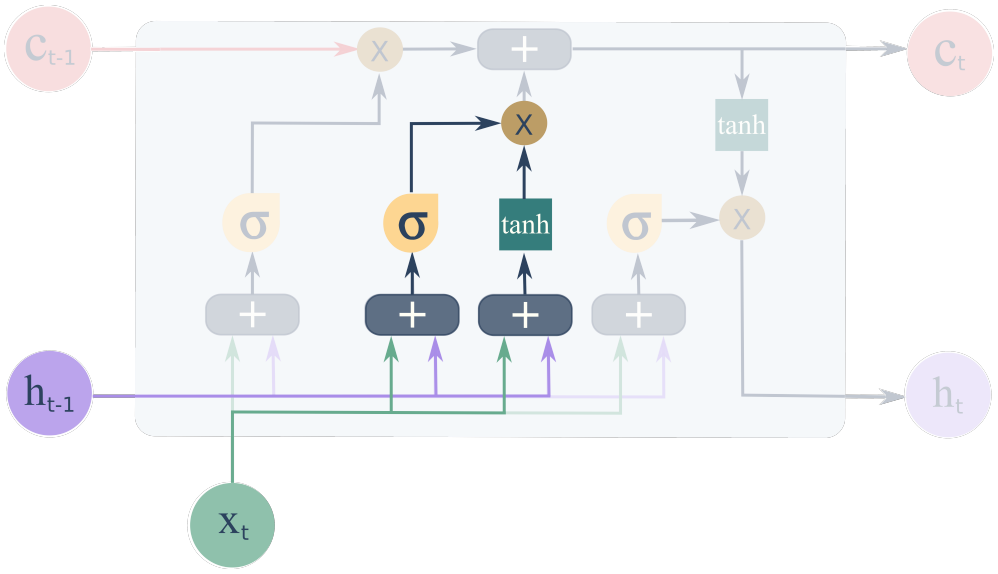

Figure 2.7: The Output gate.

3
Proposed Methodology

This chapter outlines the approach taken to evaluate the performance of two machine learning algorithms, Random Forest and Long Short-Term Memory (LSTM), in predicting oil production as an outcome variable. This study aimed to evaluate the effectiveness of Random Forest and LSTM in predicting time series, based on different reservoir datasets. This chapter details the workflow of this research, the methodology used for data preparation and the algorithm selection criteria, and the parameter tuning process for each machine learning technique used.

3.1
Problem description

Managing reservoir information plays a vital position in oil industry. The decisions to develop and manage the facilities are based on this kind of analysis chasing good oil production results. With the progress in technology and data acquisition sensors, the exploration, drilling, and production operations have become abundant static data generators. For a single well, there are several pieces of data available, including downhole pressure, downhole temperature, average tubing information, annular pressure, gas rate, oil rate, water rate for injection and production and geological characteristics.

The oil field production is non-linearly connected to all this data, so the attempts to predict the oil production using machine learning techniques usually use some of this data. Lu et all [39] used machine learning to predict oil production from Shale reservoirs using geological information and technical information such as the horizontal well length. Elmabrouk, Shirif and Mayorga [40] use neural networks to predict oil production with five inputs: oil production rate, gas production rate, future water injection rate and the future number of oil wells in production and injection. Cheng and Yang [41] feed neural networks with oil pressure, casing pressure, water content, production days, and monthly oil production to forecast the oil production of a well.

In this work, we investigate the performance of Random Forest algorithm and the LSTM neural networks in predicting oil rate daily production of a single well. All the models were tested in four different datasets. Initially,

we consider the daily oil rate as a time series dataset and uses the Random Forest algorithm to make one time step ahead predictions. Besides the daily oil production, no other input was given to the model.

After obtaining favorable outcomes forecasting one time step ahead, we started using the model to predict multiple time steps ahead, with acceptable results. At this point, we decided to incorporate the downhole pressure into the inputs. Although the Random Forest had shown satisfactory results with two or more features in the inputs, the time series dependence could be affected. For that reason, we decided to move on to a more complex machine learning algorithm, capable of handling multiple features and more adequate to work with time series datasets, the LSTM neural networks.

Initially, we built a simple LSTM architecture to compare the performance with the previous results from Random Forest algorithm. Therefore, in this first LSTM model, the only input was the daily oil production. After that, we built a more sophisticated architecture to handle the oil rate production and the downhole pressure as inputs. The model was used to forecast single and multiple time steps ahead, proving to be more adequate than the Random Forest algorithm to forecast multiple time steps ahead.

Finally, we decided to include the downhole pressure from the days we were about to predict as input. More specifically, if the model is about to predict x time steps ahead, the x downhole pressure information associated to each of these time steps are incorporated into the inputs. In this case, the model receives three inputs: the oil rate production, the past downhole pressure and the future pressure.

3.2 Random Forest Methodology

The first step in building a machine learning model is to preproces the datasets to ensure that they are properly formatted and ready for use in a machine learning algorithm. Before working on the data organization, it is essential to ensure that the data is properly cleaned, removing any missing or erroneous values. For synthetically generated datasets, this is not a problem, For real datasets, it is common to find missing or corrupted data points, since oil fields are complex systems with many variables and data sources that can fail or malfunction, leading to gaps in the data. After cleaning datasets, the next step is to restructure a time series dataset into inputs for the Random Forest Algorithm.

Consider a time series dataset with n time steps observations. We create a subset X containing vectors X_i, $i \in \mathbb{N}$, of size $L > 0$ corresponding to the

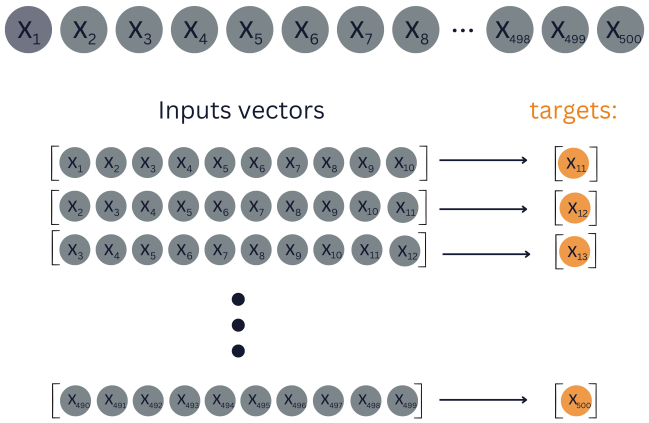

Figure 3.1: Example of 500 days observations dataset organization

past L observations of the time series. For the target (or labels) set Y, we create vectors Y_i, $i \in \mathbb{N}$, of length $F > 0$, corresponding to the immediate F forward time steps of the time series related to vector X_i. In this study, we refer to the length of each vector X_i, denoted by L, as *look-back* and to the length of vector Y_i, denoted by F, as *forward-days*.

For example, consider a dataset with 500 daily observations of oil production. Lets restructure it with *look-back*= 10 and *forward-days*= 1. In other words, the model will predict a single time step ahead ($F = 1$ day) and the inputs to make this prediction are the immediate previous ten days $L = 10$ observations of oil production. Since $L = 10$, the input vectors have length 10. Moreover, since forward-days = 1, the target vectors have length = 1 and are, therefore, single data points. Figure 3.1 displays a representation of the input vectors and the target data points for a 500 days example with *look-back* = 10 and *forward-days* = 1.

After restructuring the dataset into inputs and labels to fit the random forest algorithm, the data needs to be split into training and testing sets to evaluate the performance of the model accurately. Since time series data is dependent on time, we decided to split the data in chronological order, with the training set containing data from earlier time periods and the testing set containing data from later periods. In the four datasets used to evaluate the model, we consider, approximately, the first 70% of data for the training set and the remaining 30% as test set.

The Random Forest Regression algorithm was implemented using the Scikit-Learn package, which includes some default parameters. If not specified, the number of trees in the forest, usually known as *n_estimators*, is 100. The function to measure the quality of a split is the Squared error shown

in equation 3-1, where N is the number of samples being tested, y_i is the model prediction, and $\hat{y}$ is the actual value.

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

By default, the minimum number of samples required to split an internal node is two, and all the nodes are expanded until all leaves are pure or until all leaves contain less than two samples. The minimum number of samples required at a leaf node is 1. The minimum weighted fraction of the total sum of weights (of all the input samples) required at a leaf node is 0. *look-back* is the number of time steps to consider when looking for the best split. If not specified, there is no limit for the number of leaf nodes. A node will be split if this split induces a decrease of the impurity greater than ni that is calculated as shown in equation 3-2, where ni_j is the importance of node j, W_j is weighted number of samples reaching node j, C_j is the impurity value of node j, $W_{left(j)}$ (resp. $W_{right(j)}$) is the weighted number of samples reaching child node from left (resp. right split) on node j and $C_{left(j)}$ (resp. $C_{right(j)}$) is is the impurity value of child node from left (resp. right split) on node j. By default, bootstrap samples are used when building trees, and the number of samples in a subset, usually called *max_samples*, is equal to the number of samples in the original dataset.

$$ni_j = W_j C_j - W_{left(j)} C_{left(j)} - W_{right(j)} C_{right(j)} \tag{3-2}$$

Scikit-learn package also includes GridSearch, a tuning technique that finds optimum parameters. It is an exhaustive search performed with specific sets of parameter values. GridSearch builds a model for every combination of parameters specified in the collection and evaluates each model, returning the set of values with best results. In this work, we use GridSearch to optimize the n_estimators and max_samples values, and all other parameters are set as default.

Although the Random Forest algorithm showed good results in forecasting the oil daily production, especially when forward-days = 1, the model has some limitations. Since it was not planned for handling time series data, the Random forest does not take into account the temporal dependencies between the data points. The time series observations are restructured into linear sequences (vectors X_i) in an attempt to preserve the time information of datasets.

Nevertheless, since the input of the Random Forest is arranged as vectors, to insert a second feature (pressure, for example) in the inputs, it would be necessary to merge the sequence of oil production observations and the sequence of pressure observations into one single sequence. However, this would

mess the sequence organization, which is the only thing that carries the time information on it. Moreover, oil reservoir datasets can admit a large number of features, as they often include information about properties of the well and production data such as flow rates, pressures, and temperatures.

Considering these limitations, we choose to follow the study with a machine learning model that was designed to deal with time series datasets, the LSTM neural network.

3.3 LSTM Methodology

Likewise the Random Forest, the first step to build an LSTM model is to preprocess the datasets. The procedure is very similar to the one used in the Random Forest, the only difference is that now, the inputs are organized as matrix, and not vectors.

Consider a time series dataset with n time steps and k features. We create a subset $X := \{X_i\}$ containing matrix (X_{ab}), $a, b \in \mathbb{N}$, $0 \leq a \leq k$, $0 \leq b \leq L$, $L > 0$, corresponding to the past L observations of k features in the time series. For the target (or labels) set Y, we create vectors Y_i, $i \in \mathbb{N}$, of length $F > 0$, corresponding to the immediate F forward time steps of the time series related to the matrix X_i. Analog to the Random Forest model, we refer to the number of columns in each matrix X_i, denoted by L, as *look-back* and to the length of vector Y_i, denoted by F, as *forward-days*.

For example, consider a dataset with 500 daily observations of oil production and pressure. Lets restructure it with *look-back*= 10 and *forward-days*= 1. In other words, the model will predict a single time step ahead ($F = 1$ day) and the inputs to make this prediction are the immediate previous ten days $L = 10$ observations of oil production and pressure. Since we have two features and $L = 10$, the inputs matrix has two rows and ten columns. Moreover, since forward-days = 1, the target vectors have size = 1 and are, therefore, single data points. Figure 3.2 displays a representation of the input matrix and the target data points for a 500 days example, with *look-back* = 10 and *forward-days* = 1.

After restructuring the dataset into inputs and labels to fit the random forest algorithm, the data needs to be split into training and testing sets to evaluate the performance of the model accurately. Likewise the Random Forest, in the four datasets used to evaluate the model, we consider, approximately, the first 70% of data for the training set and the remaining 30% as test set.

The LSTM algorithm was implemented using the Keras package, which includes some default parameters. If not specified, the activation function is

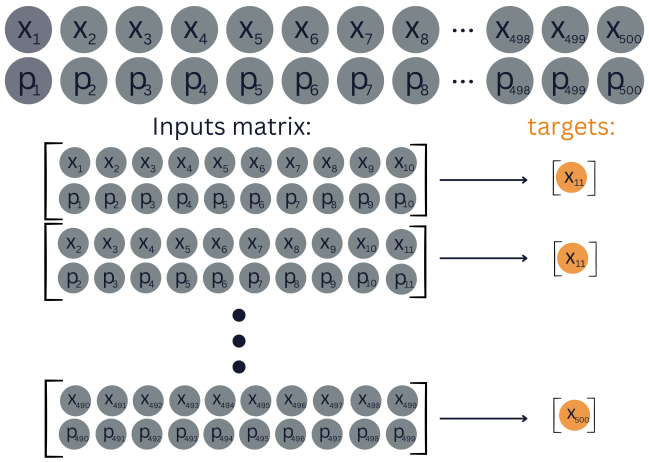

Figure 3.2: Example of 500 days observations dataset organization.

the hyperbolic tangent, and recurrent activation function is Sigmoid. There is also a default weights matrix, used for the linear transformation of the inputs. The default batch-size, the number of samples that are processed together in a single forward/backward pass of a neural network during training, is 32. The loss function, the number of neurons in each layer, the number of epochs and the optimization algorithm for training must always be specified.

Throughout the study, we built different LSTM architectures that received as input the most recent data of x days and predicted as outputs y days ahead. As said before, We refer to x and y as *look-back* and *forward-days*, respectively. At first, the information given as input was only oil rate production (bbl/day). Later, the downhole pressure (kgf/cm^2) data was incorporated into the inputs. To build the model, we use different configurations of layers and neurons, but the loss function is always MSE (Equation 3-1), the optimization algorithm is the stochastic gradient descent, and batch size is always two.

3.4 Datasets

For experiments with LSTM and random forest regressor, we investigate four different datasets, three of them synthetically generated and one from a public real dataset, in order to provide controlled results for the architectures under analysis.

3.4.1 Dataset 1, Dataset 2 and Dataset 3

The first three datasets are synthetically generated and consider a reservoir with dimensions of $5000m \times 5000m \times 15m$, permeability of $500mD$,

		Dataset 1	Dataset 2	Dataset 3
Pressure	$275kg/cm^2$	day 1 to 731	day 1 to 731	day 1 to 731
	$250kg/cm^2$	day 732 to 1461	day 732 to 1826	day 732 to 1826
	$260kg/cm^2$			day 1827 to 2373
	$225kg/cm^2$	day 1462 to 2192	day 1827 to 2373	
	$200kg/cm^2$	day 2193 to 2922	day 2374 to 2738	day 2374 to 2738
	$175kg/cm^2$	day 2923 to 3653	day 2739 to 3653	day 739 to 3653

Table 3.1: Variation of pressure in Dataset 1, Dataset 2, and Dataset 3.

initial pressure of $300kg/cm^2$, final pressure of $175kg/cm^2$, and ten years of operation, which results in 3653 days of data. The model used to genarete theese datsets considers aspecified pressure, which means that an human operator will decide the pressure that the field will be operating. All three datasets admit the same initial and final pressire, but each one considers a different pressure variation over time, as shown in table 3.1. Figures 3.3 and 3.4 show Dataset 1 and Dataset 2, respectively. Dataset 3 includes 546 days of well closure, and its presented in Figure 3.5.

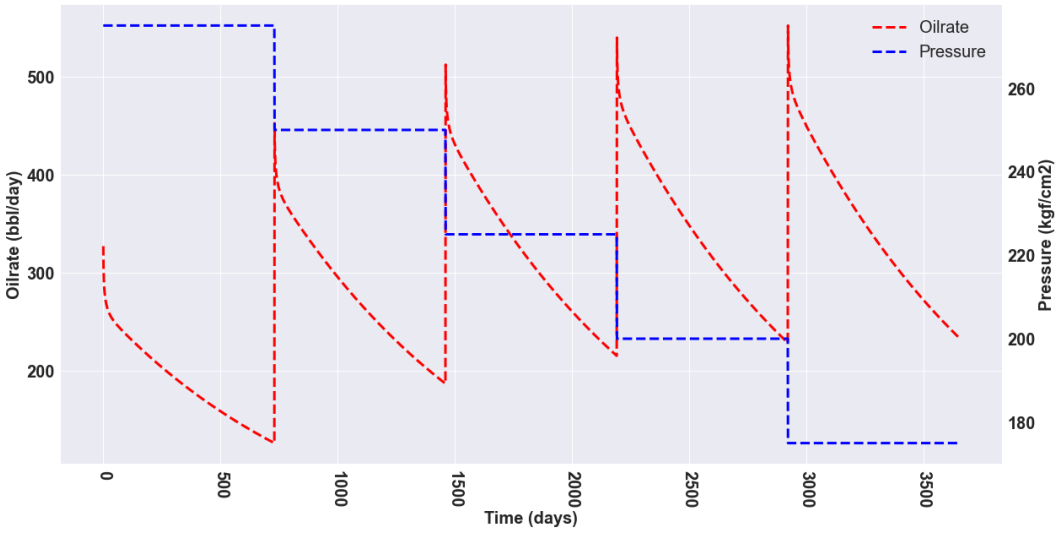

Figure 3.3: Dataset 1.

3.4.2 Dataset 4: Volve

The offshore Volve reservoir [42], located in the Norwegian North Sea, was discovered in 1993. The field is in the sandstone of the Middle Jurassic age at a depth around 2900m. The development plan was approved in 2005 and productions started in 2008, achieving a peak oil rate of 56,000 bbl/day.

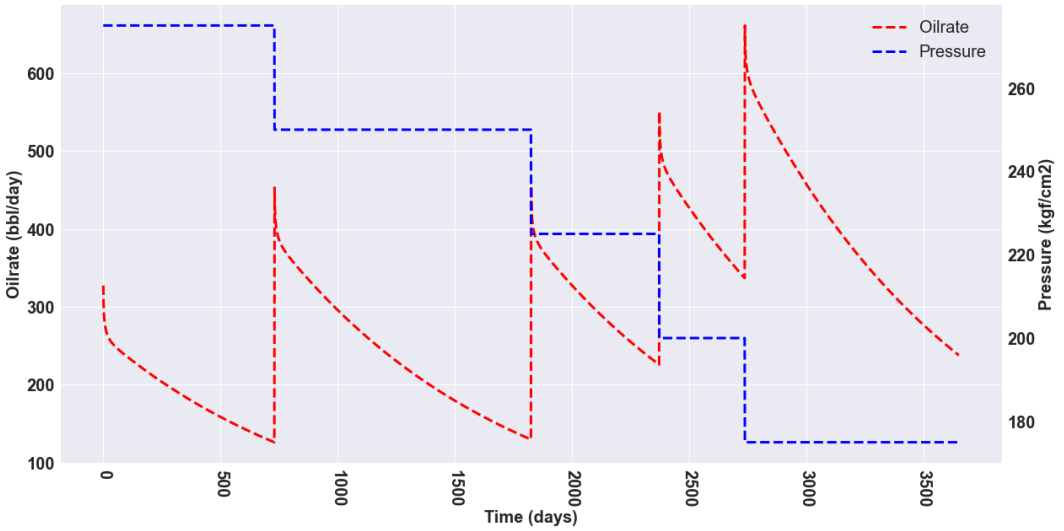

Figure 3.4: Dataset 2.

The field was decommissioned in 2016 with a cumulative oil production of 63 million barrels.

Equinor and the Volve license partners, ExxonMobil and Bayerngas, have disclosed all seismic records and oil production data from this reservoir in an open repository [42]. Since real data are often prohibitive or challenging to be obtained, the multi-terabyte Volve dataset, containing lots of information on a complete lifetime of a reservoir exploitation project, has been widely used by data scientists and reservoir engineers in their work involving the oil and gas industry. Moreover, academic researchers could test the different complex models they develop in a real field case using the Volve dataset. It has been a substance for research in drilling data, geometric modeling, scientific visualization, and Petroleum reservoir modeling. Tunkiel *et al.* [43] explored the dataset, described common obstacles found in the Volve dataset, and presented approaches for overcoming all the issues. Gupta *et al.* [44] developed a complex workflow to identify the formation type around the bit from surface drilling data. Sun *et al.* [45] build a 3D mechanical earth model of the Volve field.

This study uses only the data from well 15/9-F-1 C, contained in the Volve Field which is an uninterrupted 746 days sequence of the oil production and pressure information, shown in Figure 3.6.

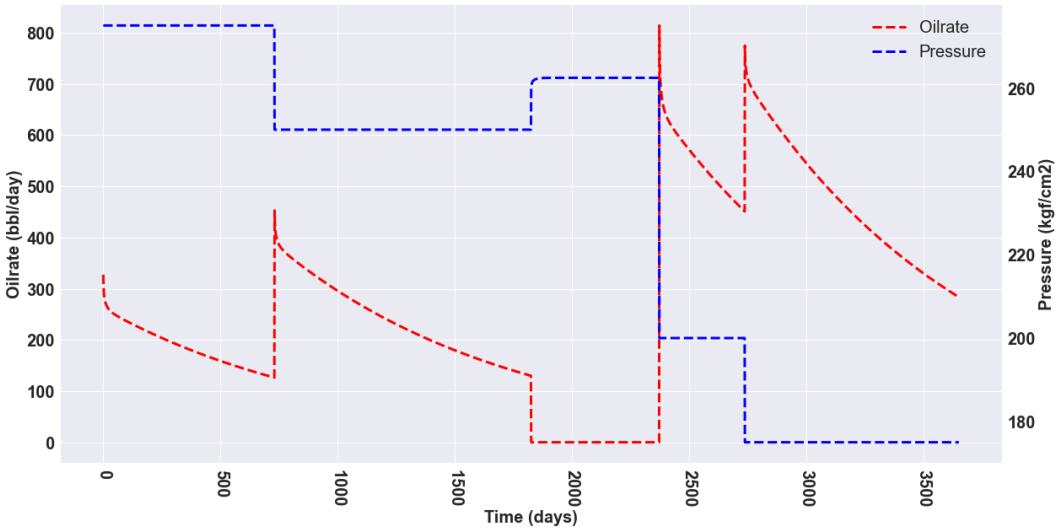

Figure 3.5: Dataset 3.

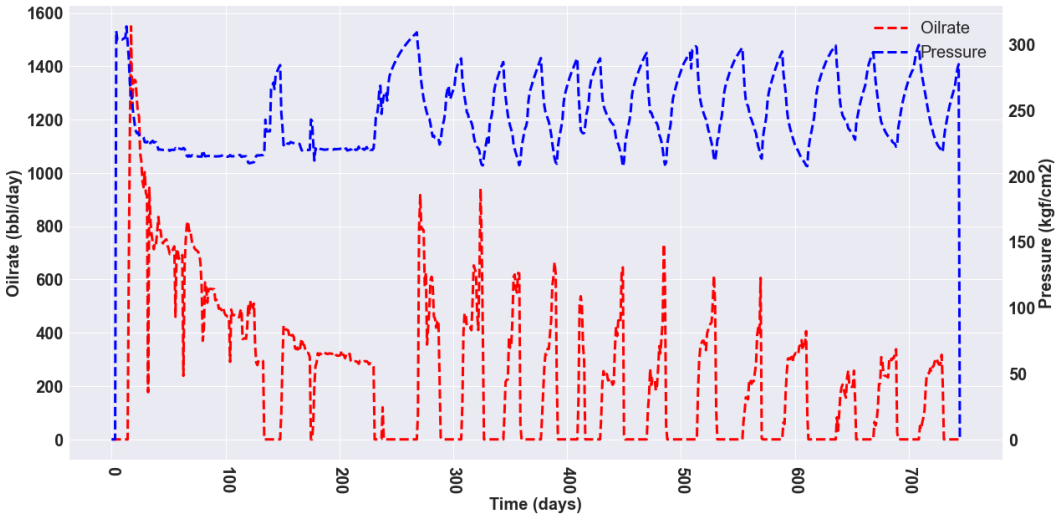

Figure 3.6: Dataset 4: Well "15/9-F-1 C" daily oil production data contained in the Volve dataset.

4
Results

In this chapter, we describe the performance achieved by each Random Forest algorithm and LSTM architecture used in this study. To measure the accuracy of the proposed models, we use Root Mean Square Error (RMSE), a classical way to evaluate the error of a forecasting model. It represented the square root of the mean of the differences between predicted and observed values, as shown in equation 4-1.

In all following results, for Datasets 1, 2 and 3, the first 2500 days are used as a training set, and the remaining 1153 days as a test set. In the Dataset 4, the Volve field, the first 500 days are used as a training set and the remaining 246 days as a test set.

$$RMSE = \left(\frac{1}{N_m} \sum_{k=1}^{N_m} \left(m_{true,k} - m_{j,k} \right)^2 \right)^{1/2}, \tag{4-1}$$

4.1
Random Forest Prediction

In this study, we settle the Random Forest parameters n_estimators as 100, 500, or 1000 and max_samples as 50%, 80%, or 100% of the original dataset. All other parameters are used as available in the Python package Scickit-Learn. Besides, we structure the dataset with three different values for *look-back*, for each value of *forward-days*. Hence, for each *look-back*, we use GridSearch to find the best n_estimators and max_samples. Thus, for each *look-back* value, GridSearch builds different models and returns the best one, according to RMSE metrics. Initially, the Random Forest Algorithm was used for single time steps prediction, wich means $forward\text{-}days = 1$. Than, we use the algorithm to forecast 10, 50 and 100 time steps ahead.

Note that when *look-back* is settled as t, it is necessary t time steps to forecast *forward-days* time steps ahed, so different values for *look-back* generate different amounts of predictions even if the test set always has the same size.

4.1.1 Forward-days = 1

The daily oil rate production was the only input for the Random Forest algorithm. In each dataset, the original time series oil production is restructured into a sliding window dataset, where *look-back* time steps are used to predict the immediately next time step. For *forward-days*, we set *look-back* $= \{10, 25, 50\}$. Table 4.1 summarizes the results presented in this section. We present the best parameters selected by GridSearch and the computed value for the RMSE metric for each look-back, in each dataset.

	look-back	max_samples	n_estimators	RMSE
Dataset 1	10	50%	1000	09.64
	25	50%	500	09.84
	50	50%	100	10.21
Dataset 2	10	50%	500	22.87
	25	50%	100	25.07
	50	50%	1000	24.85
Dataset 3	10	50%	100	36.37
	25	50%	100	42.04
	50	80%	100	43.97
Dataset 4	10	80%	100	76.65
	25	80%	100	81.10
	50	50%	1000	73.07

Table 4.1: Summary of the Random Forest results with *look-back* $= \{10, 25, 50\}$ and forward-days = 1.

For Dataset 1, the best outcome was with look-back= 10, with n_estimators as 1000, max_samples as 50% and RMSE = 09.64. Figure 4.1 shows the results in test set for each look-back and Figure 4.2 displays the best result for all Dataset 1. Although the training dataset provides physical anomalies due to the well opening and shut, after the well is open, around day 2900, the algorithm found troubles when forecasting. It might happen because of the considerable difference in the values transition in the dataset. Hence, it is reasonable to expect some mismatch in prediction and observations when

physical phenomena cause data anomalies. Remark that no physics information is provided for the algorithm. One may conclude that the Random Forest algorithm might be sensitive to huge data oddities.

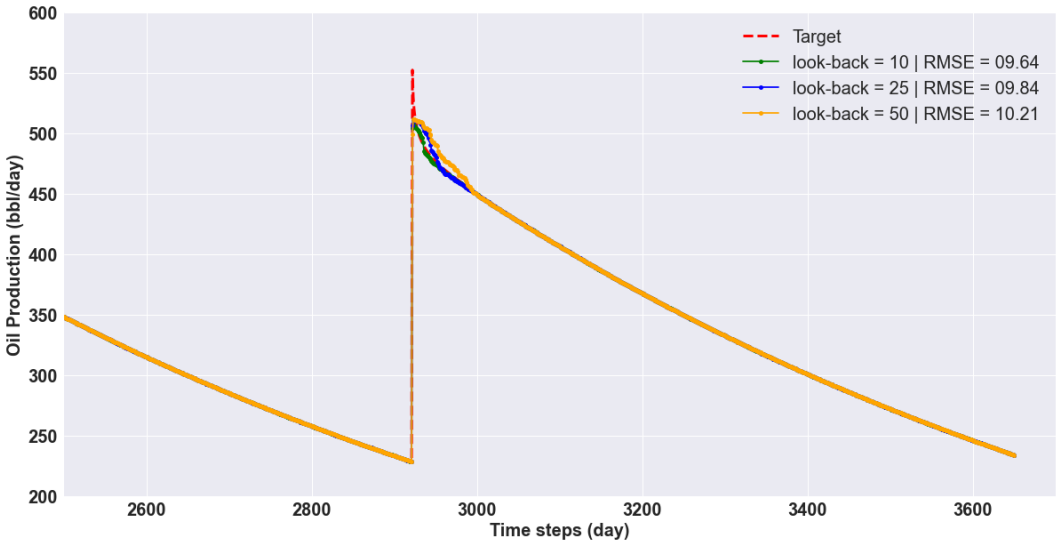

Figure 4.1: Random Forest oil production predictions in test set from Dataset 1 with *look-back* $= \{10, 25, 50\}$.

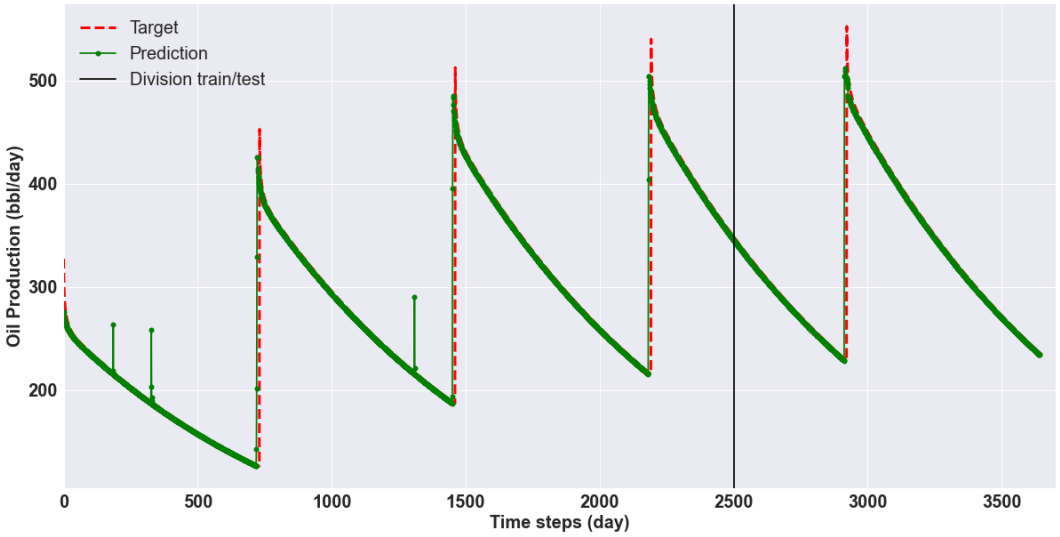

Figure 4.2: Oil production forecast with Random Forest in Dataset 1 with *look-back* =10, n_estimators=1000, max_samples as 50% and RMSE= 09.64

For Dataset 2, the best outcome was with look-back= 10, with n_estimators as 500, max_samples as 50% and RMSE = 22.87. Figure 4.3 shows the results in test set for each look-back and Figure 4.4 displays the

best result for all Dataset 2. Note that dataset 2 presents higher complexities than the previous one. In this case, the oil rate and time separating well opening and shut are discontinuous. In that case, the oil rate is maintained continuously uniformly in time. Although the prediction is not as good as the one presented before, one may note the consistency in forecasting the decay in oil production. However, when the well is open, the pike presents higher turbulence than in the previous experiment. It may indicate that the Random Forest algorithm could not capture the physical information of the problem when joining with reservoir management decisions, which is expected since well opening and shutting are decided by a human.

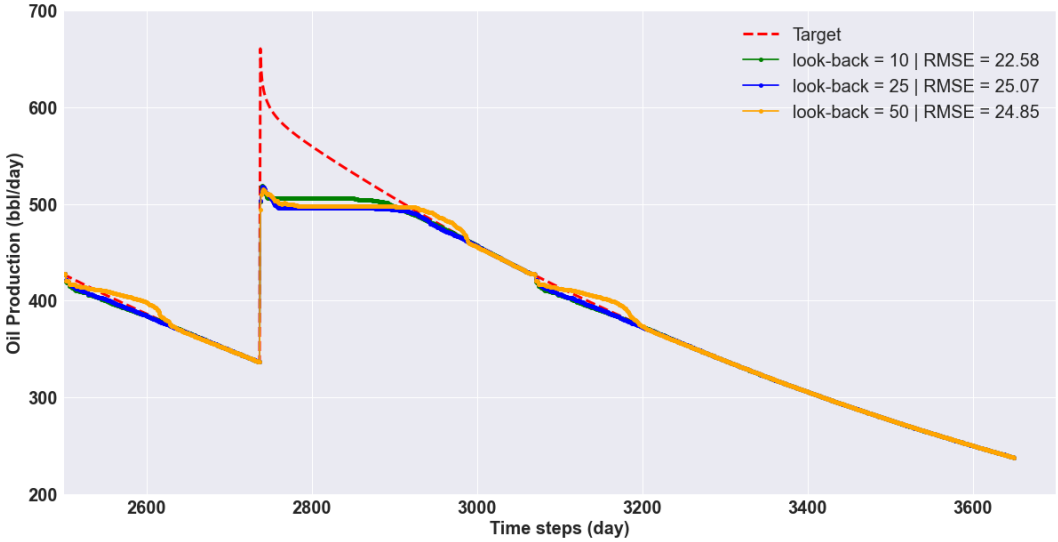

Figure 4.3: Random Forest oil production predictions in test set from Dataset 2 with *look-back* $= \{10, 25, 50\}$.

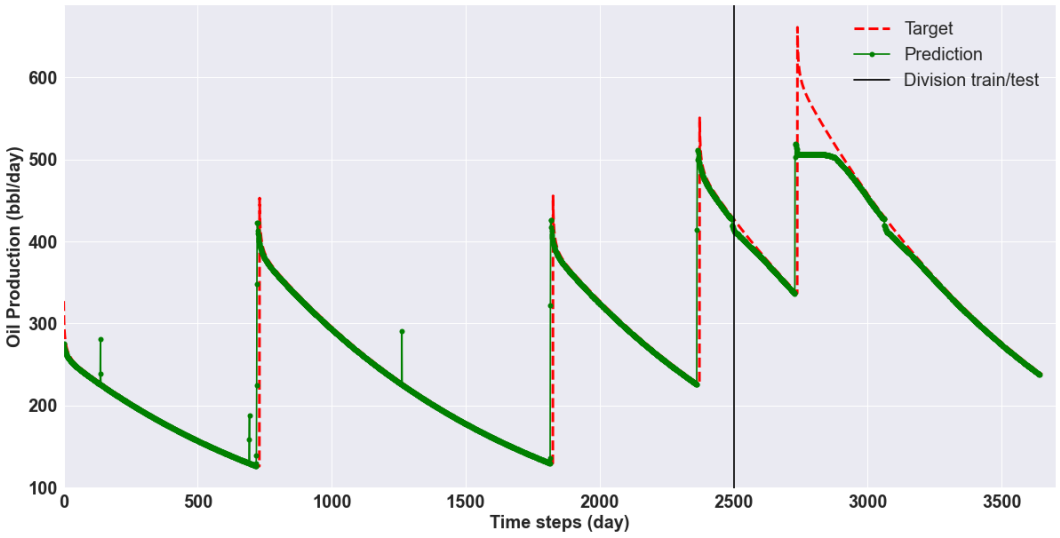

Figure 4.4: Oil production forecast with Random Forest in Dataset 2 with *look-back* =10, n_estimators=500, max_samples as 50% and RMSE= 22.87

For Dataset 3, the best outcome was with look-back= 10, with n_estimators as 100, max_samples as 50% and RMSE = 36.37. Figure 4.5 shows the results in test set for each look-back and Figure 4.6 displays the best result for all Dataset 3. In this case, in contrast with what happens in dataset 2, one may observe that the model outcomes are not sharp. Remark that dataset 2 provides more reservoir management and data physics information. Therefore it is expected that dataset 3 presents low performance compared to the previous ones.

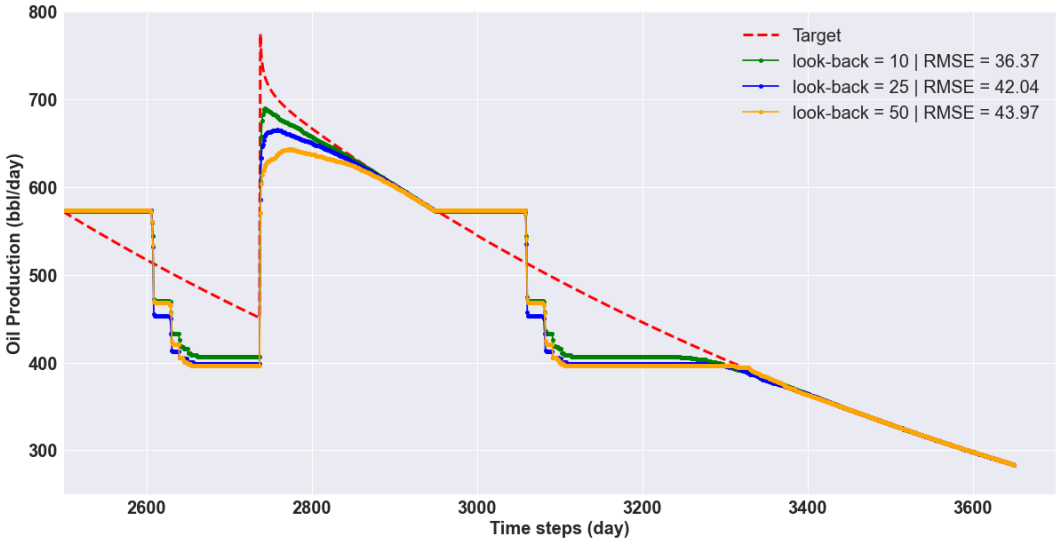

Figure 4.5: Random Forest oil production predictions in test set from Dataset 3 with *look-back* $= \{10, 25, 50\}$.

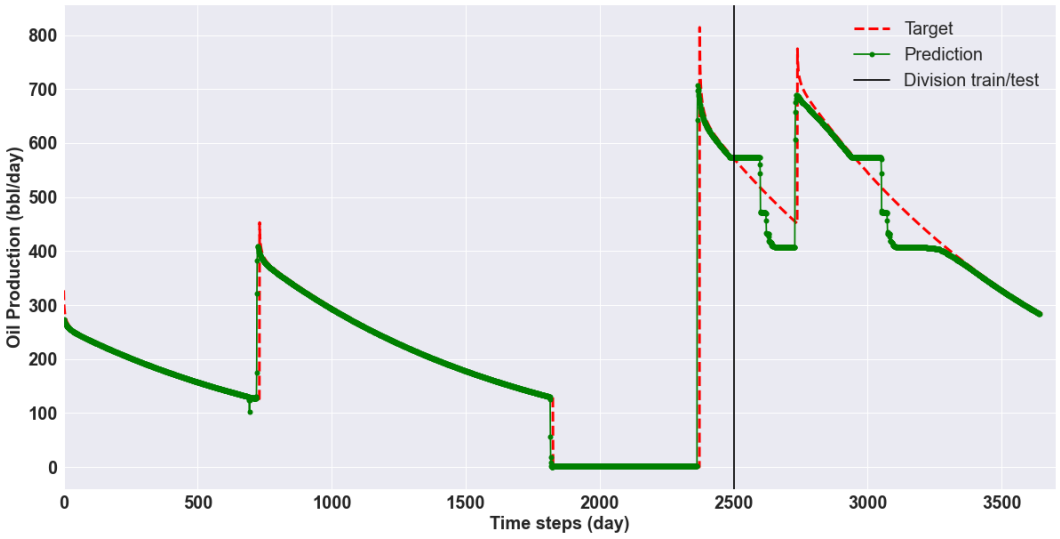

Figure 4.6: Oil production forecast with Random Forest in Dataset 3 with *look-back* =10, n_estimators=100, max_samples as 50% and RMSE= 36.37

For Dataset 4, the best outcome was with look-back= 50, with n_estimators as 1000, max_samples as 50% and RMSE = 73.07. Figure 4.7 shows the results in test set for each look-back and Figure 4.8 displays the best result for all Dataset 4. This dataset is the most challenging because it relates to a real field case. It is expected that the predictions are not sharply matched with observations. Moreover, predicting oil field production with many system perturbations is difficult. Nevertheless, the Random Forest algorithm could

reasonably catch the necessary information to predict and reproduce the oil field performance.

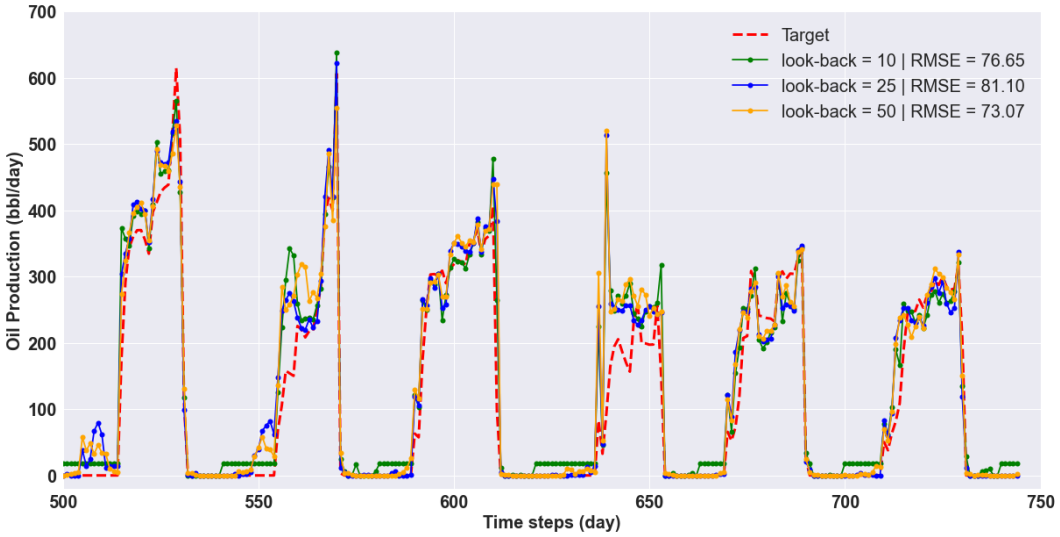

Figure 4.7: Random Forest oil production predictions in test set from Dataset 4 with *look-back* $= \{10, 25, 50\}$.

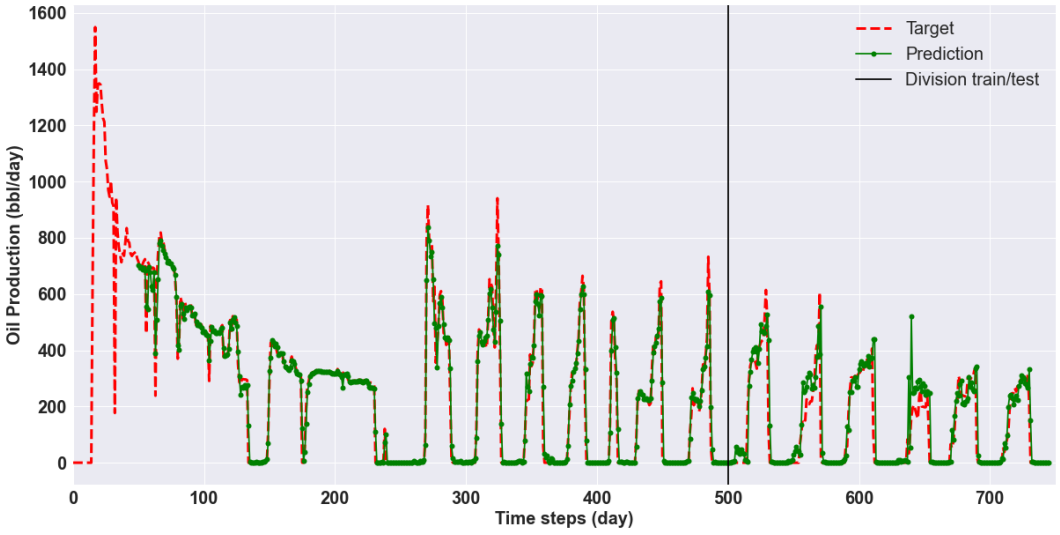

Figure 4.8: Oil production forecast with Random Forest in Dataset 4 with *look-back* = 50, n_estimators=1000, max_samples as 50% and RMSE= 73.07

The approach presented in this section aims to verify if the suggested approach is suitable for predicting oil field performance. We tested different values for look-back and *forward-days* and simulated different values for the Random Forest algorithm. The presented results state that the Random Forest

algorithm, although simple, was a good ally when predicting oil production. It could learn from the training dataset relevant features and physics from the problem, which can be noticed in datasets 1, 2, and 3. The most complex was dataset 4, using a real field case to train and test the algorithm. Even with limited and perturbated data, it was possible to reproduce similar results to the observations.

4.1.2 Forward-days = 10

The primary objective of this section is to compare the previous approach of predicting one value to predicting a sequence. In the previous approach, we concluded that the Random Forest is suitable for predicting one time step ahead in the presented datasets. On the other hand, adequate reservoir management might need forecasting multiple time steps in the future. As a result, we intend to verify if the suggested approach is suitable for this new problem. Note that the previous technique trained the model to produce one output. Currently, it is trained to offer a succession of values that must have sequence features, as expected in a time-series application.

As in all other Random Forest results of this work, the daily oil rate production was the only input for the algorithm. For In each dataset, the original time series oil production is restructured into a sliding window dataset, where *look-back* time steps are used to predict the immediately 10 next time steps. For *forward-days* = 10, we maintain *look-back* = $\{10, 25, 50\}$. Table 4.1 shows the best values for max_samples and n_estimators returned by GridSearch and the respective RMSE for each look-back, in each dataset.

	look-back	max_samples	n_estimators	RMSE
Dataset 1	10	50%	500	4.21
	25	50%	1000	02.48
	50	50%	100	04.21
Dataset 2	10	50%	500	28.38
	25	50%	100	31.57
	50	50%	1000	32.06
Dataset 3	10	50%	100	43.49
	25	50%	100	49.57
	50	80%	100	52.92
Dataset 4	10	50%	100	167.65
	25	50%	100	225.72
	50	50%	1000	137.20

Table 4.2: Summary of the Random Forest results with *look-back* = $\{10, 25, 50\}$ and forward-days = 10.

For Dataset 1, the best outcome was with look-back= 25, with n_estimators as 1000, max_samples as 50% and RMSE = 02.48. Figure 4.9 shows the results in test set for each look-back and Figure 4.10 displays the best result for all Dataset 1. As presented in the previous section, one may observe similar turbulence in this approach when predicting right after the well is open again. The turbulence is alleviated when time moves forward. Thus, for dataset 1, increasing *forward-days* from 1 to 10 did not cause any problems. Increasing the data available for training might improve the presented results for dataset 1.

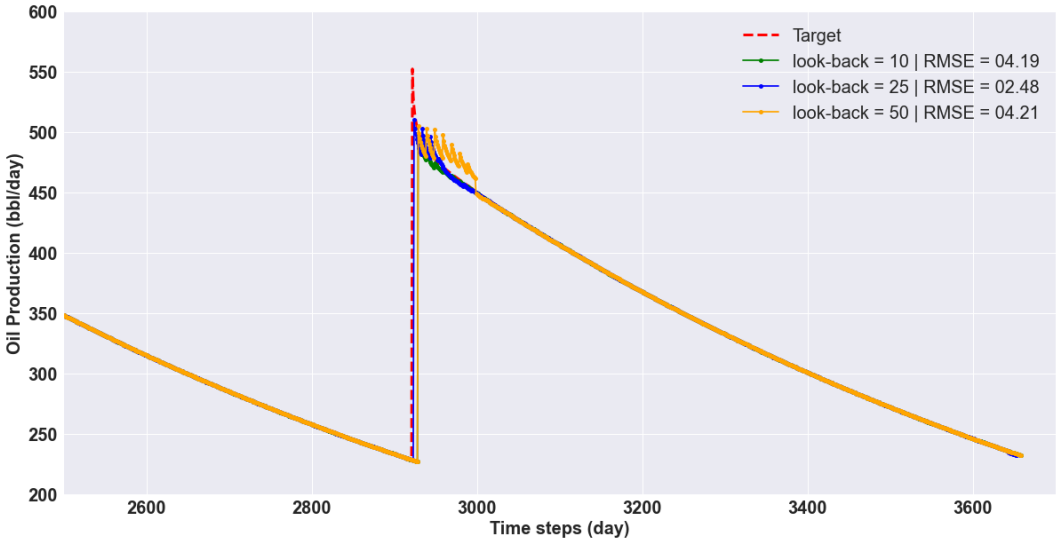

Figure 4.9: Random Forest oil production predictions in test set from Dataset 1 with *look-back* $= \{10, 25, 50\}$ and *forward-days* $= 10$.

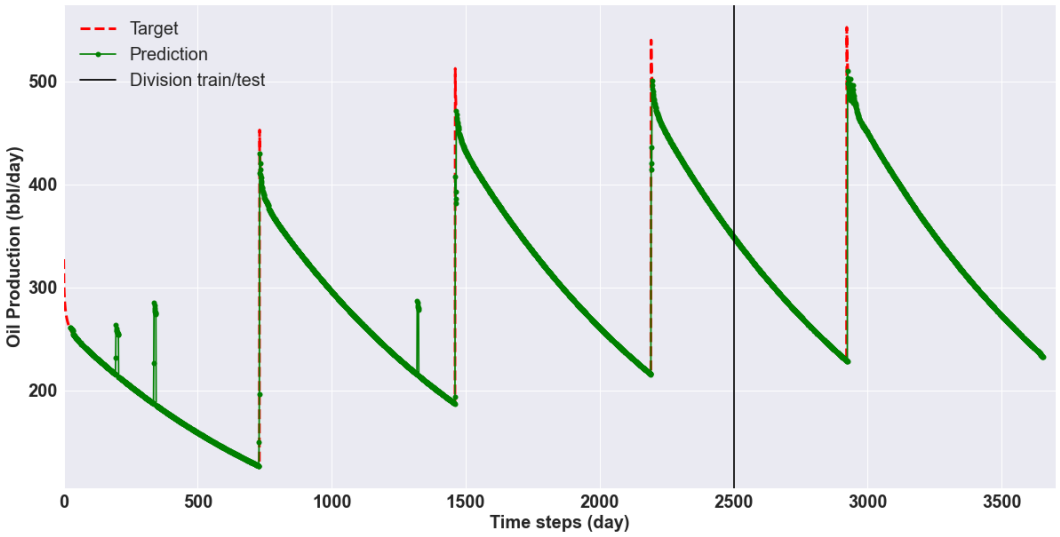

Figure 4.10: Oil production forecast with Random Forest in Dataset 1 with *look-back* =25, n_estimators=1000, max_samples as 50% and RMSE= 2.48

For Dataset 2, the best outcome was with look-back= 10, with n_estimators as 1000, max_samples as 50% and RMSE = 28.38. Figure 4.11 shows the results in test set for each look-back and Figure 4.12 displays the best result for all Dataset 2. In this case, similar to the conclusion of the experiment of dataset 1, increasing the *forward-days* to ten did not cause much trouble in predicting oil field production. Again, the suggested application to

alleviate the problems of lack of physics informed in the training data, one must increase the available data, including more possible conditions.

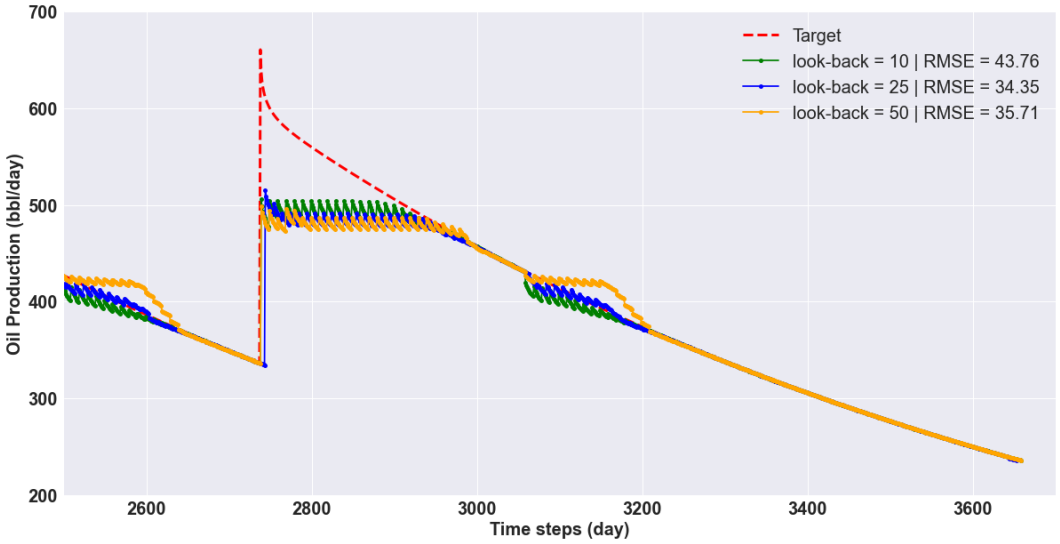

Figure 4.11: Random Forest oil production predictions in test set from Dataset 2 with *look-back* $= \{10, 25, 50\}$ and *forward-days* $= 10$.

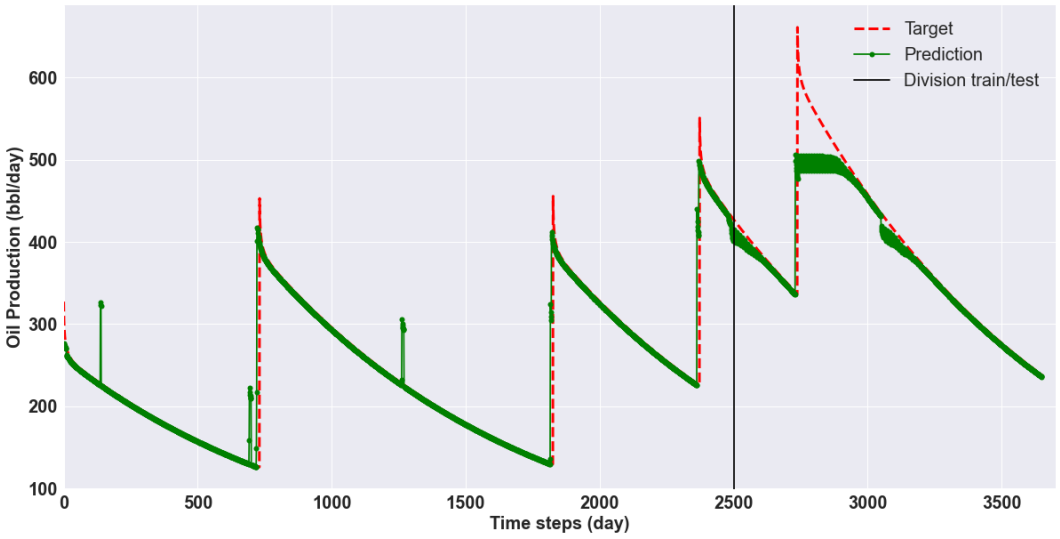

Figure 4.12: Oil production forecast with Random Forest in Dataset 2 with *look-back* =10, *forward-days* = 10, n_estimators=1000, max_samples as 50% and RMSE= 28.38

For Dataset 3, the best outcome was with look-back= 10, with n_estimators as 500, max_samples as 50% and RMSE = 43.49. Figure 4.13 shows the results in test set for each look-back and Figure 4.14 displays the

best result for all Dataset 3. As evaluated in the simulations presented in this section, the realization for dataset 3 is not different from the one observed in the previous section.

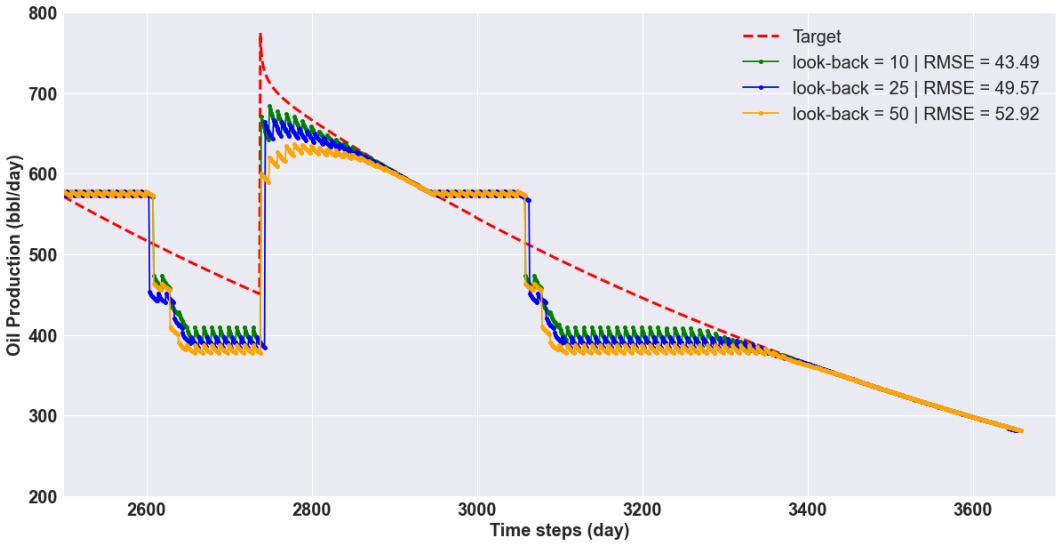

Figure 4.13: Random Forest oil production predictions in test set from Dataset 3 with *look-back* $= \{10, 25, 50\}$ and *forward-days* $= 10$.

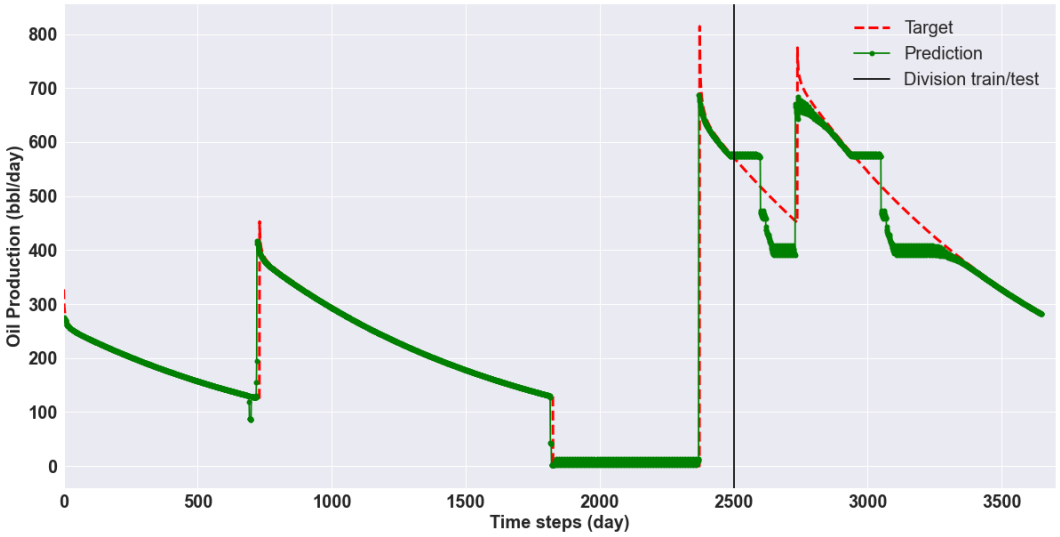

Figure 4.14: Oil production forecast with Random Forest in Dataset 3 with *look-back* =10, *forward-days* = 10, n_estimators=1000, max_samples as 50% and RMSE= 43.49

For Dataset 4, the best outcome was with look-back= 50, with n_estimators as 1000, max_samples as 50% and RMSE = 137.20. Figure 4.15

shows the results in test set for each look-back and Figure 4.16 displays the best result for all Dataset 4.

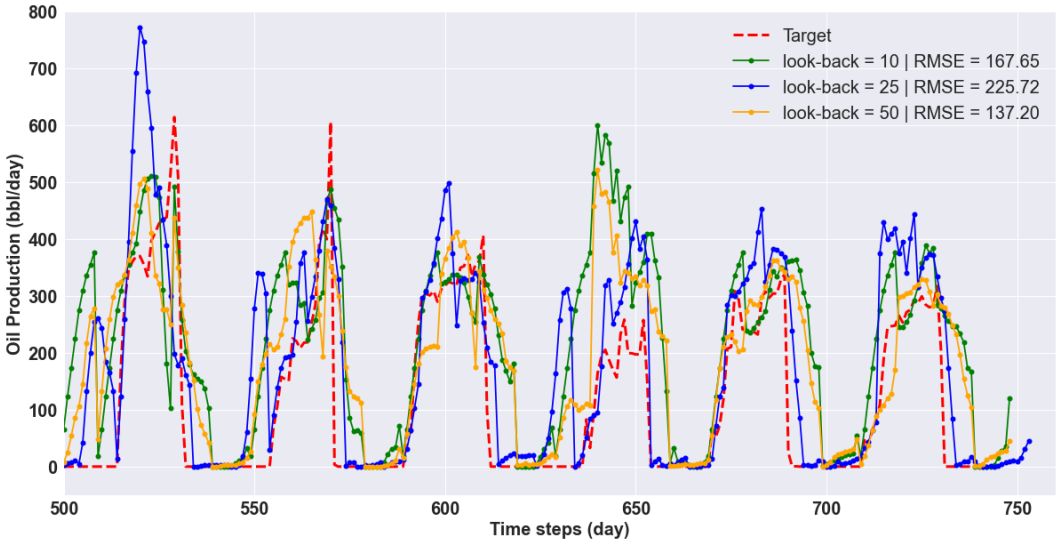

Figure 4.15: Random Forest oil production predictions in test set from Dataset 4 with *look-back* $= \{10, 25, 50\}$ and *forward-days* $= 10$.

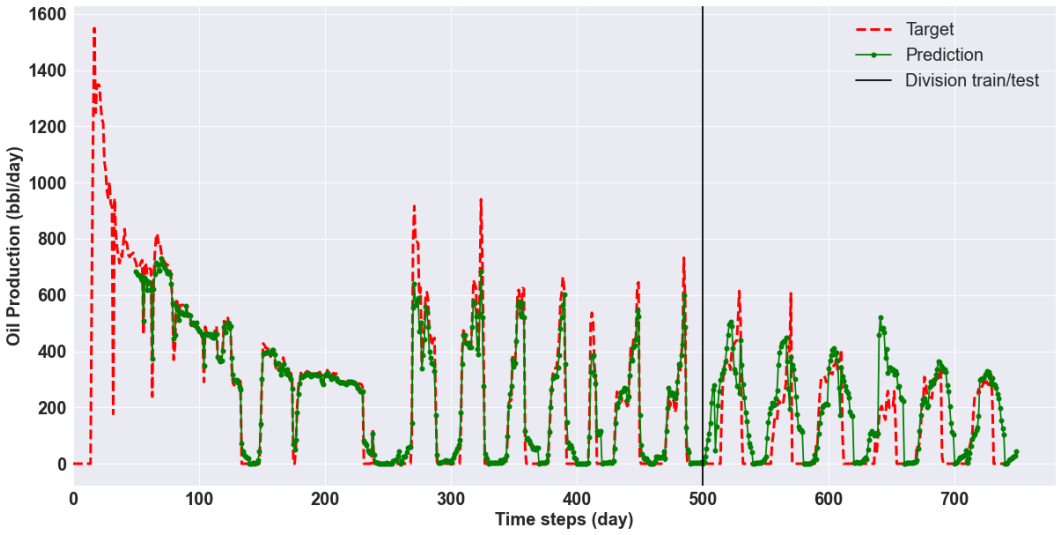

Figure 4.16: Oil production forecast with Random Forest in Dataset 4 with *look-back* =10, *forward-days* = 10, n_estimators=1000, max_samples as 50% and RMSE= 137.20

Again, the results are similar to those presented in this section. We may conclude that the forward-days is not a sensitive parameter in the proposed strategy to predict oil production. However, it is reasonable to expect that the as bigger the value set for forward-days gets, the worse model outputs quality.

This conjecture could not be proved by increasing forward-days from 1 to 10. In the following section, we will increase forward-days to 50, then to 100, and check the model results with different values for look-back.

4.1.3 Forward-days = 50

To predict 50 time steps ahead, we decided to use *look-back* = $\{25, 50, 100\}$. For each dataset, the original time series oil production is re-structured into a sliding window dataset, where *look-back* time steps are used to predict the immediately 50 next time steps. Table 4.3 shows the best values for max_samples and n_estimators returned by GridSearch and the respective RMSE for each look-back, in each dataset.

	look-back	max_samples	n_estimators	RMSE
Dataset 1	25	50%	100	34.35
	50	50%	1000	35.71
	100	50%	100	64.00
Dataset 2	25	50%	100	45.19
	50	50%	1000	48.44
	100	50%	100	69.03
Dataset 3	25	50%	500	71.73
	50	50%	100	72.06
	100	80%	100	100.37
Dataset 4	25	50%	1000	203.41
	50	50%	100	208.31
	100	50%	1000	259.83

Table 4.3: Summary of the Random Forest results with *look-back* = $\{25, 50, 100\}$ and forward-days = 50.

For Dataset 1, the best outcome was with look-back= 25, with n_estimators as 100, max_samples as 50% and RMSE = 34.35. Figure 4.17 shows the results in test set for each look-back and Figure 4.18 displays the best result for all Dataset 1.

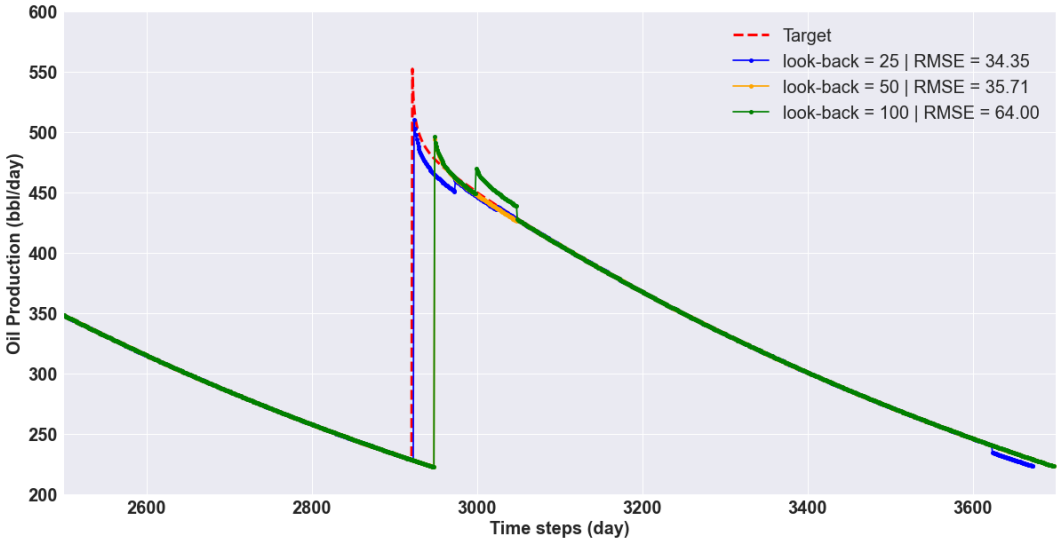

Figure 4.17: Random Forest oil production predictions in test set from Dataset 1 with *look-back* $= \{25, 50, 100\}$ and *forward-days* $= 50$.

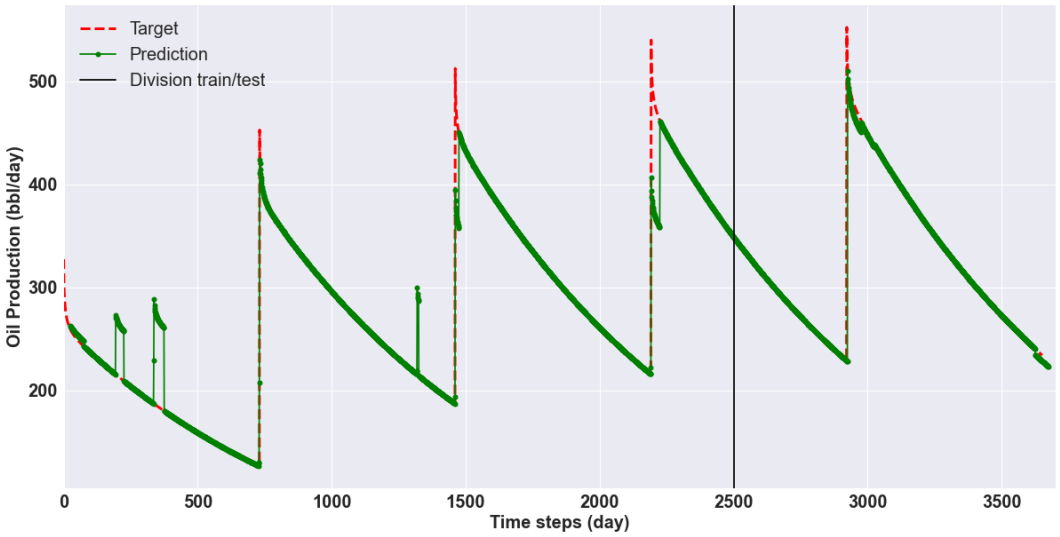

Figure 4.18: Oil production forecast with Random Forest in Dataset 1 with *look-back* =25, *forward-days* = 50, n_estimators=100, max_samples as 50% and RMSE= 34.35

For Dataset 2, the best outcome was with look-back= 25, with n_estimators as 100, max_samples as 50% and RMSE = 45.19. Figure 4.19 shows the results in test set for each look-back and Figure 4.20 displays the best result for all Dataset 2.

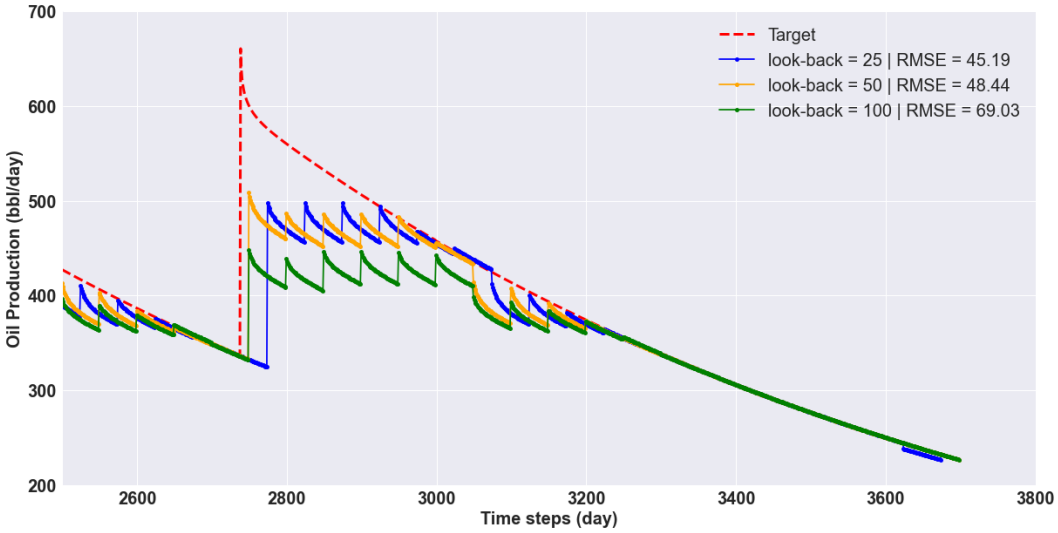

Figure 4.19: Random Forest oil production predictions in test set from Dataset 2 with *look-back* $= \{25, 50, 100\}$ and *forward-days* $= 50$.

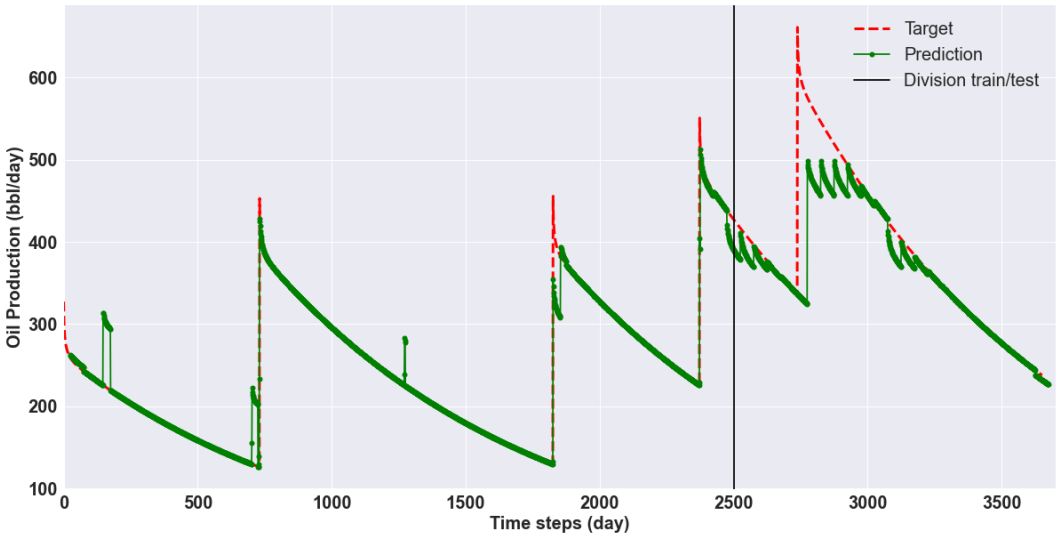

Figure 4.20: Oil production forecast with Random Forest in Dataset 2 with *look-back* =25, *forward-days* = 50, n_estimators=100, max_samples as 50% and RMSE= 45.19

For Dataset 3, the best outcome was with look-back= 25, with n_estimators as 500, max_samples as 50% and RMSE = 71.73. Figure 4.21 shows the results in test set for each look-back and Figure 4.22 displays the best result for all Dataset 3.

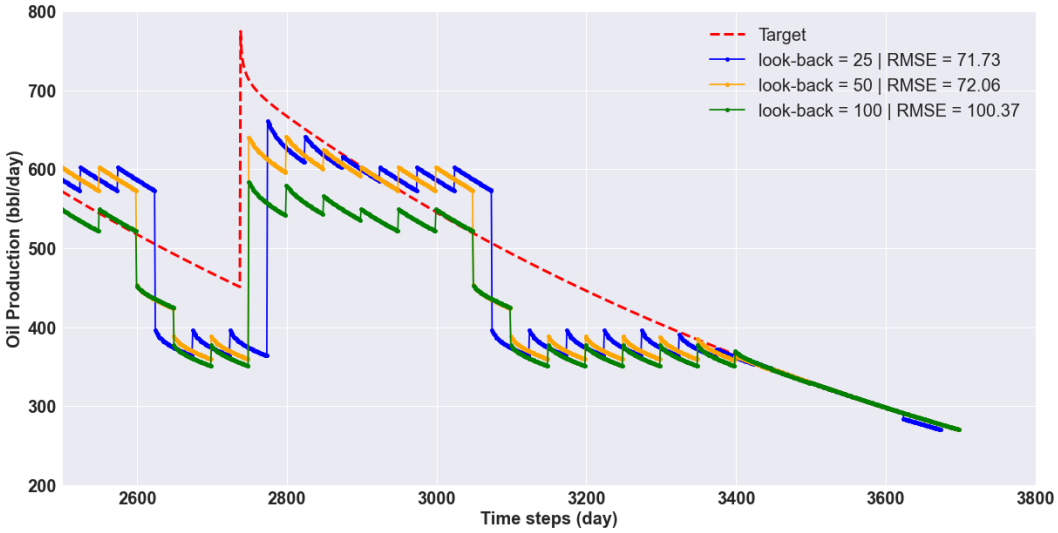

Figure 4.21: Random Forest oil production predictions in test set from Dataset 3 with *look-back* $= \{25, 50, 100\}$ and *forward-days* $= 50$.

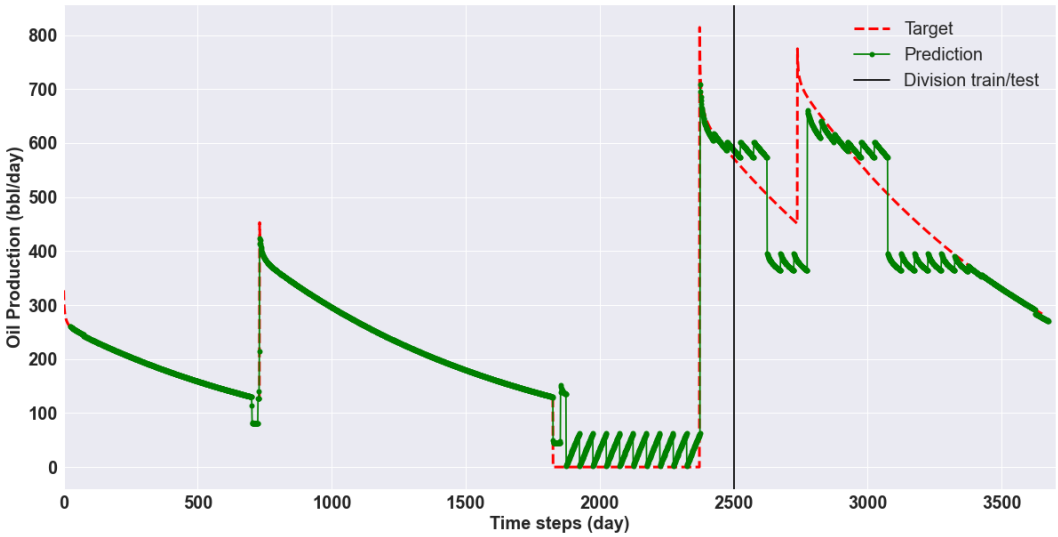

Figure 4.22: Oil production forecast with Random Forest in Dataset 3 with *look-back* =25, *forward-days* = 50, n_estimators=500, max_samples as 50% and RMSE= 71.73

For Dataset 4, the best outcome was with look-back= 25, with n_estimators as 1000, max_samples as 80% and RMSE = 203.41. Figure 4.21 shows the results in test set for each look-back and Figure 4.24 displays the best result for all Dataset 4.

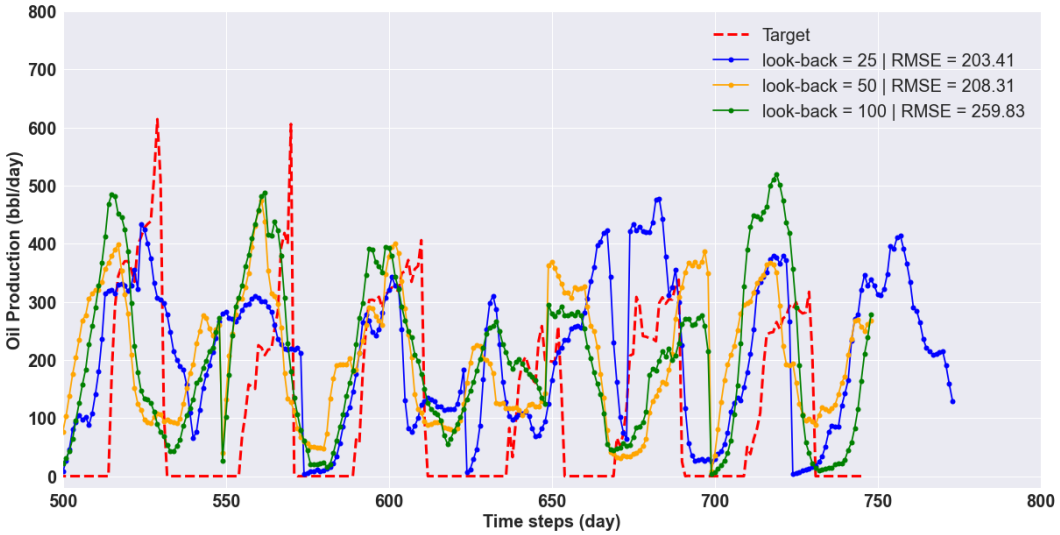

Figure 4.23: Random Forest oil production predictions in test set from Dataset 4 with *look-back* $= \{10, 25, 50\}$ and *forward-days* $= 50$.

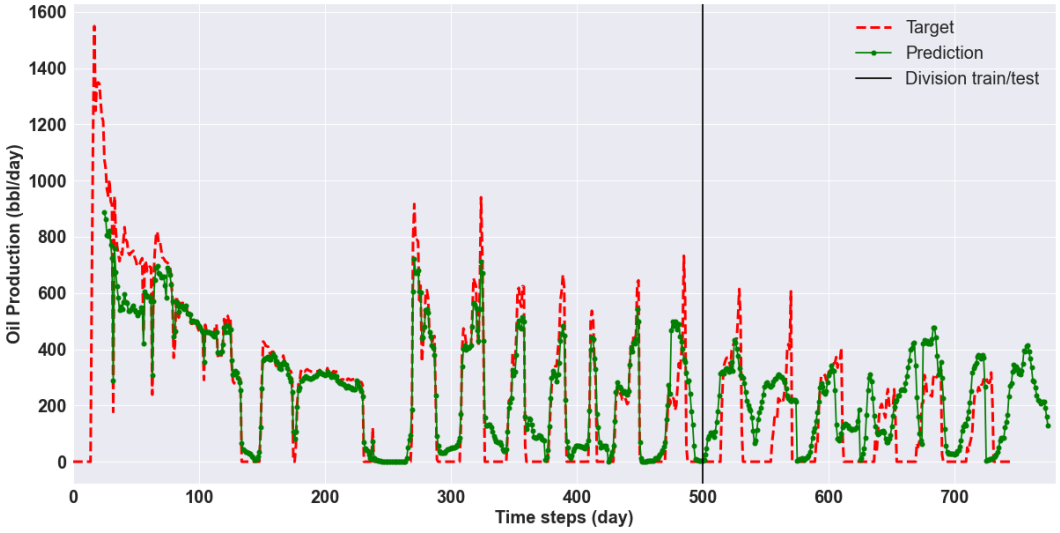

Figure 4.24: Oil production forecast with Random Forest in Dataset 4 with *look-back* =25, *forward-days* = 50, n_estimators=1000, max_samples as 80% and RMSE= 203.41

4.1.4 Forward-days = 100

To predict 100 time steps ahead, we decided to use *look-back* $= \{50, 100, 200\}$. For each dataset, the original time series oil production is re-structured into a sliding window dataset, where *look-back* time steps are used

to predict the immediately 100 next time steps. Note that, for dataset 4, it was not possible to structure the data with *look-back* = 200. Since the test set has only 246 days, it is not possible to use 200 days to predict the next 100 days ahead. So for this dataset, it was only tested *look-back* = $\{50, 100\}$. Table 4.4 shows the best values for max_samples and n_estimators returned by GridSearch and the respective RMSE for each look-back, in each dataset.

	look-back	max_samples	n_estimators	RMSE
Dataset 1	50	50%	100	60.72
	100	50%	100	45.67
	200	50%	100	87.97
Dataset 2	50	50%	100	63.42
	100	50%	100	69.25
	200	50%	100	133.92
Dataset 3	50	80%	100	112.25
	100	80%	100	135.11
	200	50%	500	158.43
Dataset 4	50	50%	100	213.99
	100	50%	100	213.91

Table 4.4: Summary of the Random Forest results with *look-back* = $\{10, 25, 50\}$ and forward-days = 100.

For Dataset 1, the best outcome was with look-back = 100, with n_estimators as 100, max_samples as 50% and RMSE = 45.67. Figure 4.13 shows the results in test set for each look-back and Figure 4.14 displays the best result for all Dataset 1. Note that the results presented for this dataset differ from those presented in the previous sections, with *forward-days* equal to 1 and 10. In this case, the expected fall after opening the well is delayed in the model predictions. We observe this happening due to the model predicting when the well will be shut or open, which is very unlikely to reproduce real observations because this is not physics information, which is a human decision.

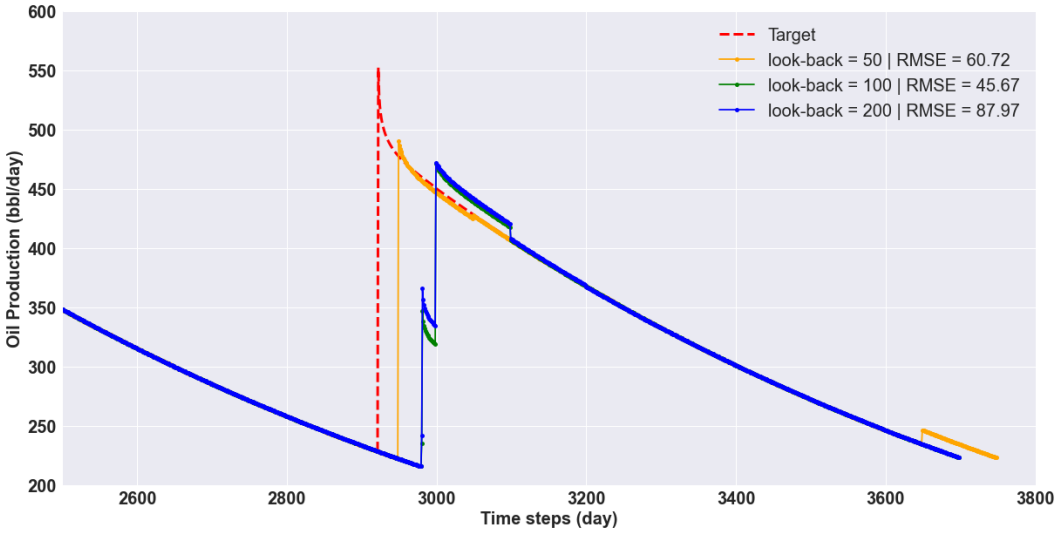

Figure 4.25: Random Forest oil production predictions in test set from Dataset 1 with *look-back* = $\{50, 100, 200\}$ and *forward-days* = 100.

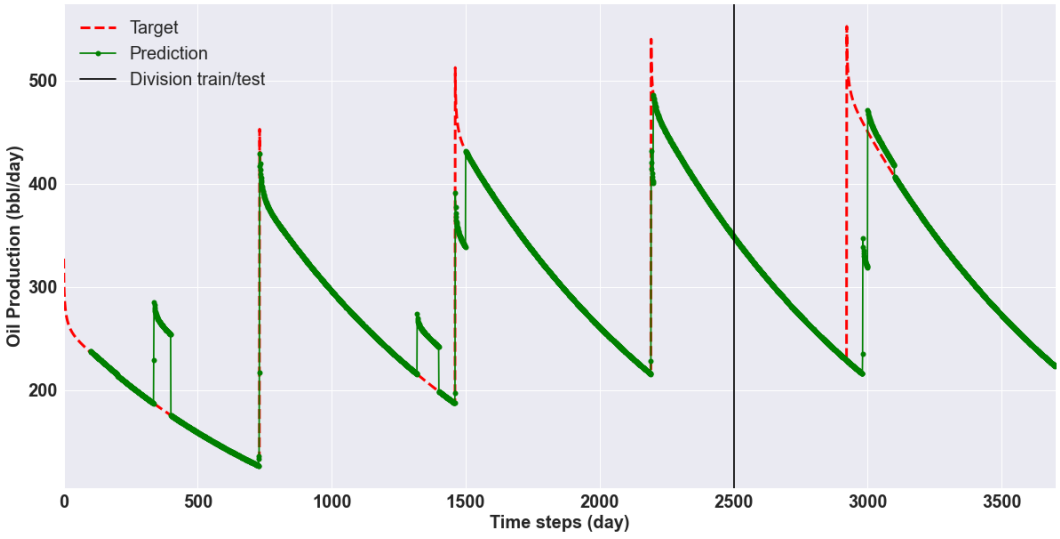

Figure 4.26: Oil production forecast with Random Forest in Dataset 1 with *look-back* = 100, *forward-days* = 100, n_estimators=100, max_samples as 50% and RMSE= 45.67

For Dataset 2, the best outcome was with look-back = 50, with n_estimators as 100, max_samples as 50% and RMSE = 63.42. Figure 4.27 shows the results in test set for each look-back and Figure 4.28 displays the best result for all Dataset 2.

As observed in the previous test case, when we select *forward-days* equal to 100, the model could not present proper forecasting for oil production.

The reason might be the same as discussed before, such as joining highly complex physics information with the human decision to open and shut the well. In addition, predicting 100 days might need the model to learn the best from the data, which must provide all possible information to guarantee good training. Nevertheless, providing such a quality dataset might be unfeasible for the energy industry, where data acquisition might be costly and complex.

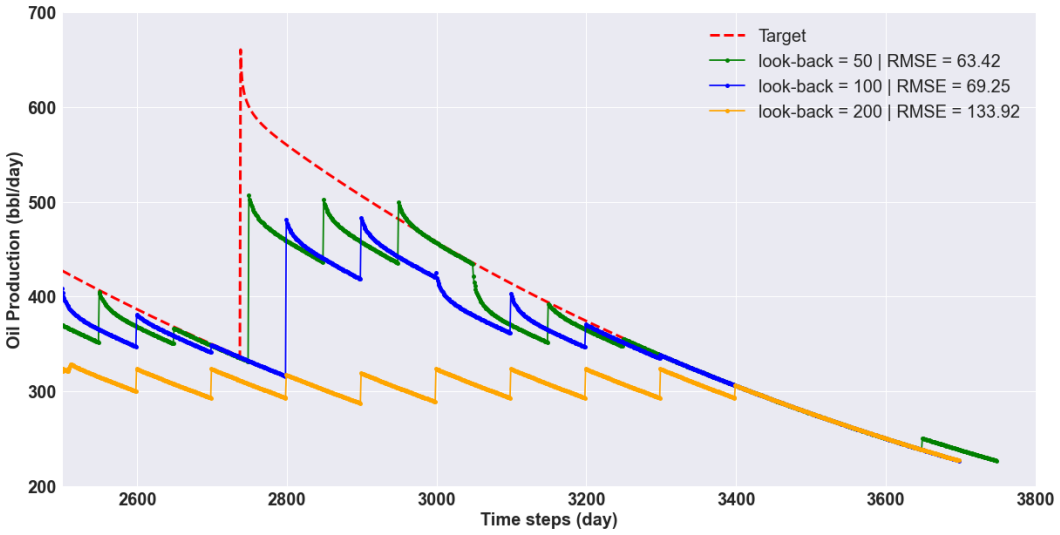

Figure 4.27: Random Forest oil production predictions in test set from Dataset 2 with *look-back* $= \{50, 100, 200\}$ and *forward-days* $= 100$.

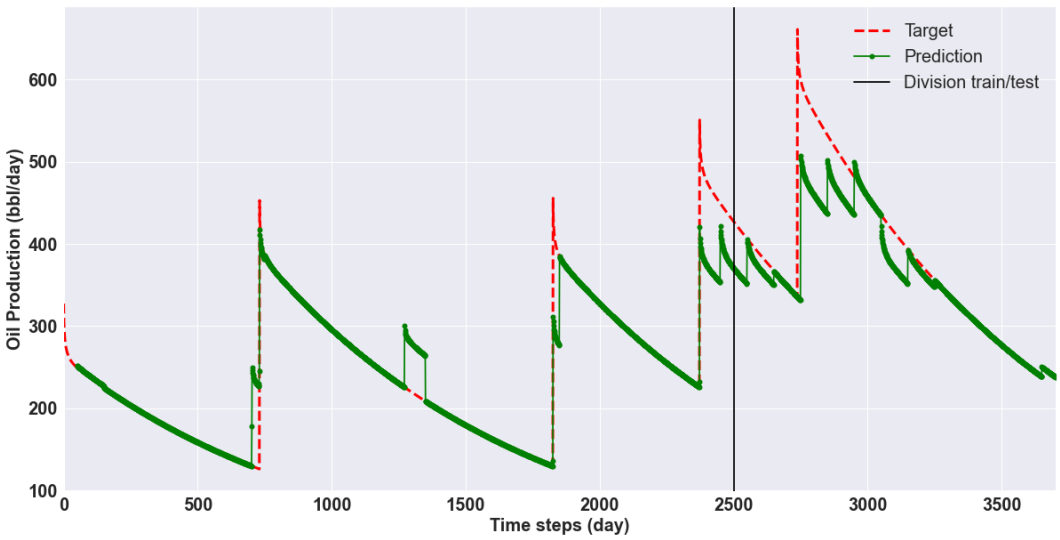

Figure 4.28: Oil production forecast with Random Forest in Dataset 2 with *look-back* $= 50$, *forward-days* $= 100$, n_estimators=100, max_samples as 50% and RMSE= 63.42

For Dataset 3, the best outcome was with look-back= 50, with n_estimators as 100, max_samples as 50% and RMSE = 63.42. Figure 4.27 shows the results in test set for each look-back and Figure 4.28 displays the best result for all Dataset 3.

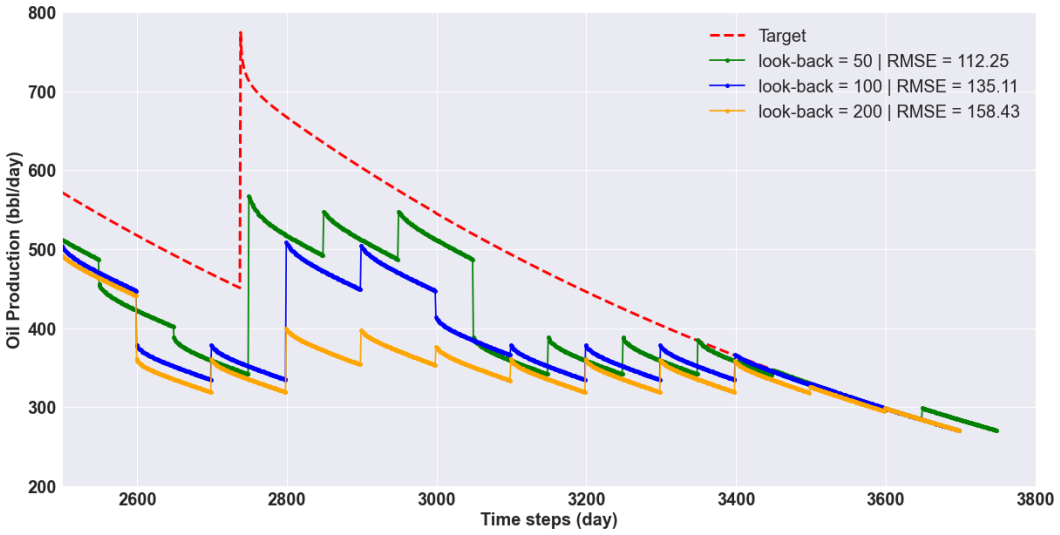

Figure 4.29: Random Forest oil production predictions in test set from Dataset 3 with *look-back* = $\{10, 25, 50\}$ and *forward-days* = 100.

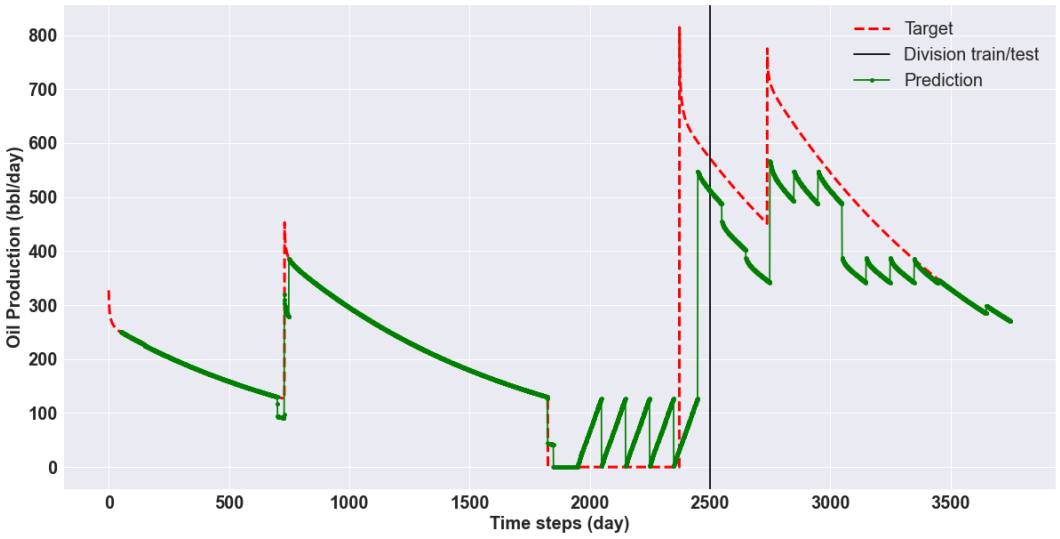

Figure 4.30: Oil production forecast with Random Forest in Dataset 3 with *look-back* = 50, *forward-days* = 100, n_estimators=500, max_samples as 50% and RMSE= 158.43

For Dataset 4, the best outcome was with look-back= 100, with n_estimators as 1000, max_samples as 50% and RMSE = 213.91. Figure 4.31

shows the results in test set for each look-back and Figure 4.32 displays the best result for all Dataset 4.

The experiment run for dataset 4 displays the trouble found when setting *forward-days* equal to 100. Many errors are involved in real field case problems, such as measurement inaccuracies related to the sensors or the physics problem itself. As a result, it is not expected that machine learning models, such as the Random Forest, to reproduce reliable predictions. Adding all these inaccuracies to the nonlinear application of increasing the *forward-days* to a considerable value resulted in poor oil production forecasting. It is possible to note a sequence characteristic in the prediction data, exposing the power of the model and how it could detect features that happened in the look-back input data and try to reproduce to the *forward-days* output data. However, the results could be more reliable for accurate and precise reservoir management applications. We suggest using deep learning to predict oil field performance to escape this problem. Moreover, the long short-term memory model, commonly known as LSTM, may greatly help due to its "memory" approach. Therefore, the following sections expose the results obtained using the LSTM method.

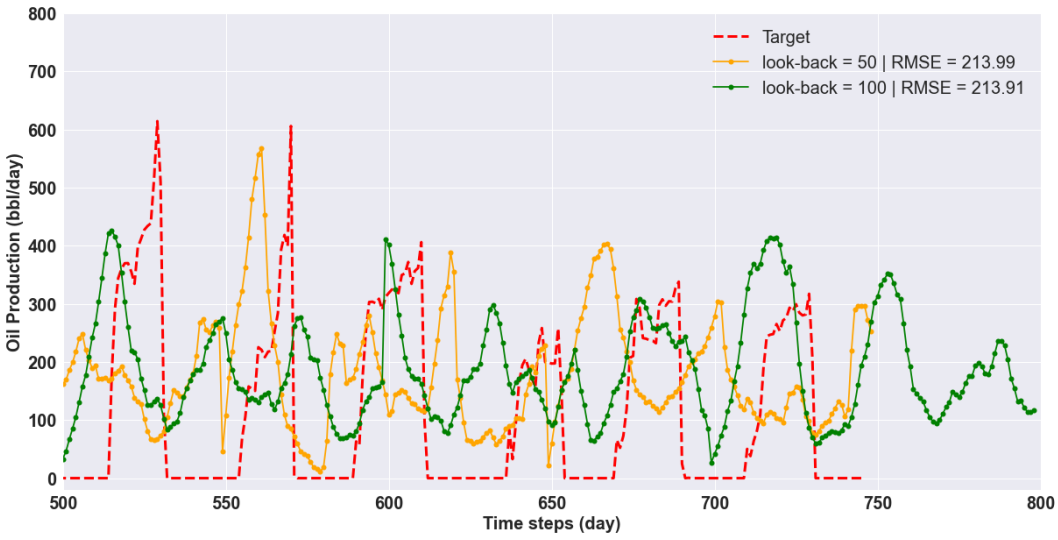

Figure 4.31: Random Forest oil production predictions in test set from Dataset 4 with *look-back* $= \{50, 100\}$ and *forward-days* $= 100$.

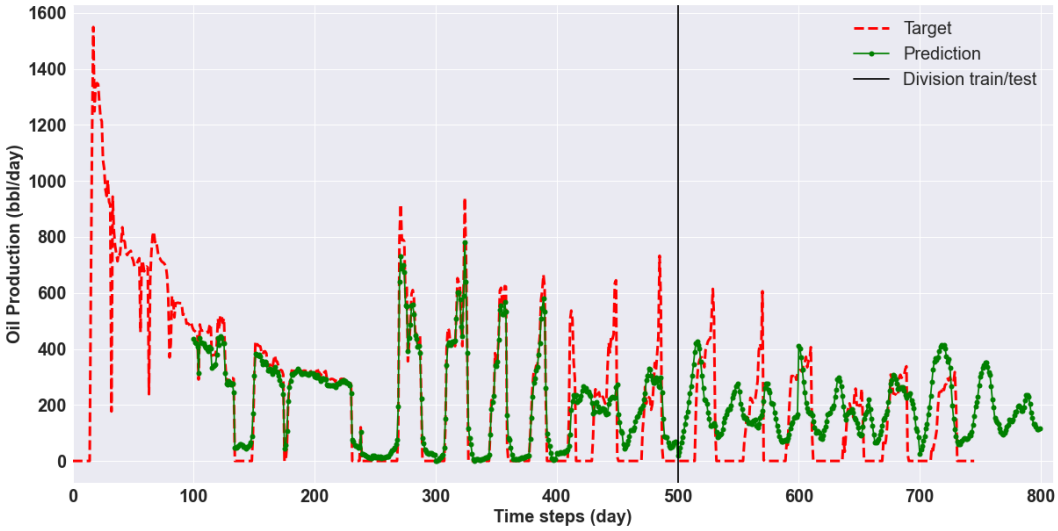

Figure 4.32: Oil production forecast with Random Forest in Dataset 4 with *look-back* = 100, *forward-days* = 100, n_estimators=1000, max_samples as 80% and RMSE= 213.91

4.2 LSTM predictions

The results in this section are organized according to the inputs given to the model. Initally, we decided to build a simple LSTM model to compare the outcomes with Random Forest results. Therefore, at first, the only input was the oil rate production, just as in the Random Forest algorithm. Then, the down hole pressure was incorporated to the inputs and finally, we decided to give as an input, the down hole pressure from the days we were about to predict.

During this work, we built two different LSTM models. The first is a simple model, constructed with 1 LSTM layer with 8 neurons, activation function Sigmoid, and 1 dense layer with forward-days neurons. Once this model could not handle anymore the complexity of results we were trying to achieve, we start using the second model, constructed with 1 LSTM layer with 64 neurons, 1 LSTM layer with 32 neurons, 1 dense layer with 16 neurons and 1 dense layer with *forward-days* neurons. In both LSTM layers, the Hyperbolic Tangent was used as activation function. We refer to the first model as Model 1, and to the second model, we refer as Model 2. Although the tuning technique GridSearch is also available for neural networks, we decided not to use it, since it is an exhaustive search, it takes too much time for more complex algorithms, such as the LSTM.

The LSTM models were build using Python 3 with packages Scikit-learn version 0.24.2, and Keras version 2.7.0. The default paramenters for LSTM layers are activation function hyperbolic tangent (tanh), recurrent activation function Sigmoid. $glorot_uniformtoinitializetheweightmatrix.Allinputswerescaledbetween0and1, andeachfeaturewas$

With LSTM models we make predictions with forward-days $= \{1, 10, 50, 100\}$ *For forward-days* $= \{1, 10\}$ *we restructure datasets with look-back*$=\{10, 25, 50\}$*, just as in Random Forest results. For forward-days* $= \{50, 100\}$ *we restructure datasets with look-back*$=\{\frac{x}{2}, x, 2x\}$*, where* x *refers to forward-days.*

4.2.1
Input: Oil rate

Initially we use Model 1 to forecast forward-days $= \{1, 10, 50, 100\}$ *and the only input given to the model was the oil rate production.*

4.2.1.1
Forward-days = 1

Table 4.5 shows the RMSE for each look-back, in each dataset for forward-days $= 1$*. For all datasets, the best RMSE was with look-back* $= 10$*. Figures 4.33, 4.34, 4.35, 4.36 shows the best results for Datasets 1, 2, 3 and 4, respectively. Figure 4.37 shows the results in test set of Dataset 4 for each look-back. For Datasets 1, 2 and 3, the results were so similar, that there is no visual difference in test sets.*

	look-back	RMSE
Dataset 1	10	1.34
	25	6.83
	50	12.36
Dataset 2	10	3.75
	25	06.82
	50	12.89
Dataset 3	10	1.71
	25	5.90
	50	12.51
Dataset 4	10	12.64
	25	67.69
	50	64.81

Table 4.5: Summary of the LSTM Model 1 results when the input was the oil rate production, with *look-back* $= \{10, 25, 50\}$ and forward-days $= 1$.

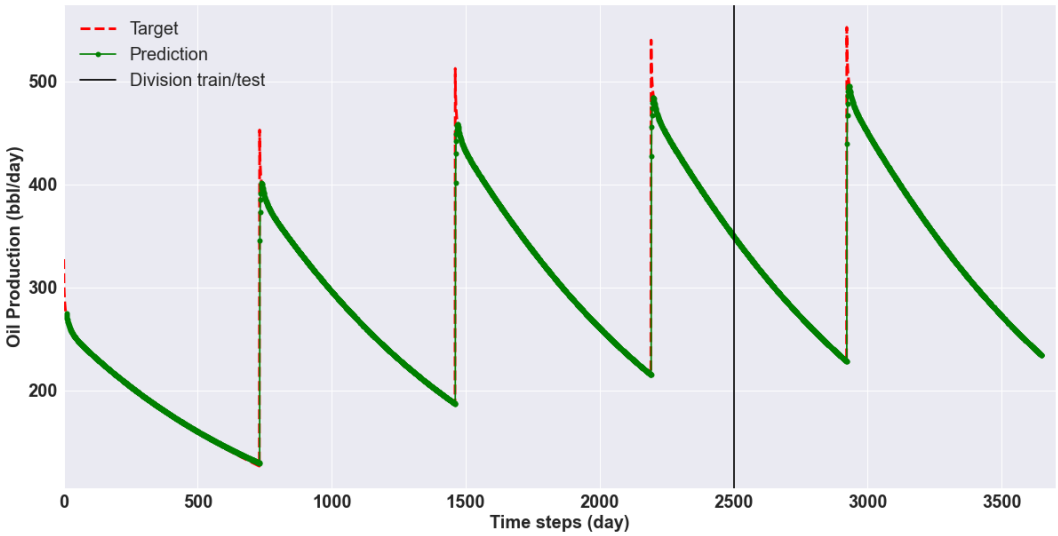

Figure 4.33: LSTM Model 1 oil production forecast in Dataset 1, with *forward-days* $= 1$, *look-back* $= 10$ and RMSE= 1.34.

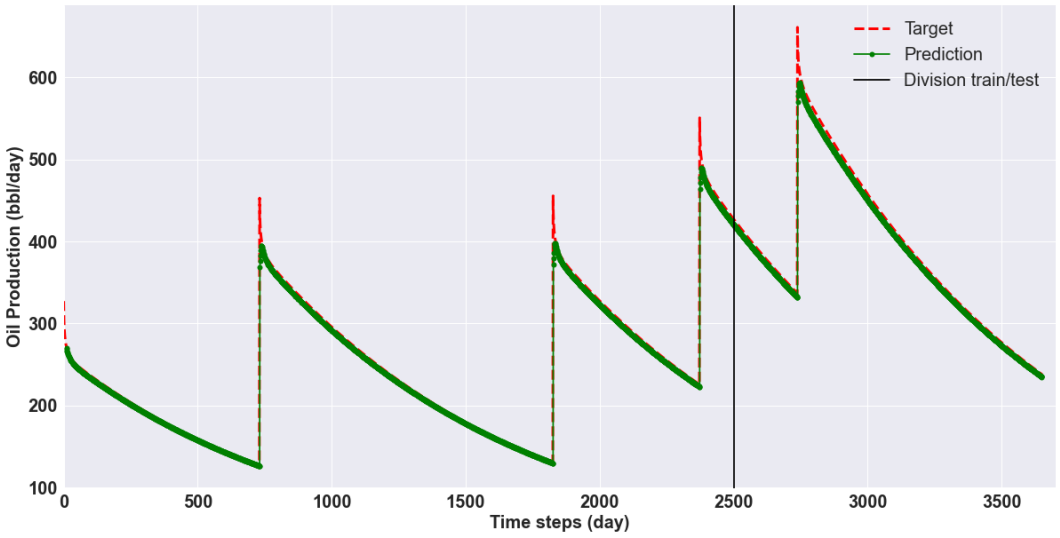

Figure 4.34: LSTM Model 1 oil production forecast in Dataset 2, with *forward-days* = 1, *look-back* = 10 and RMSE= 03.73.

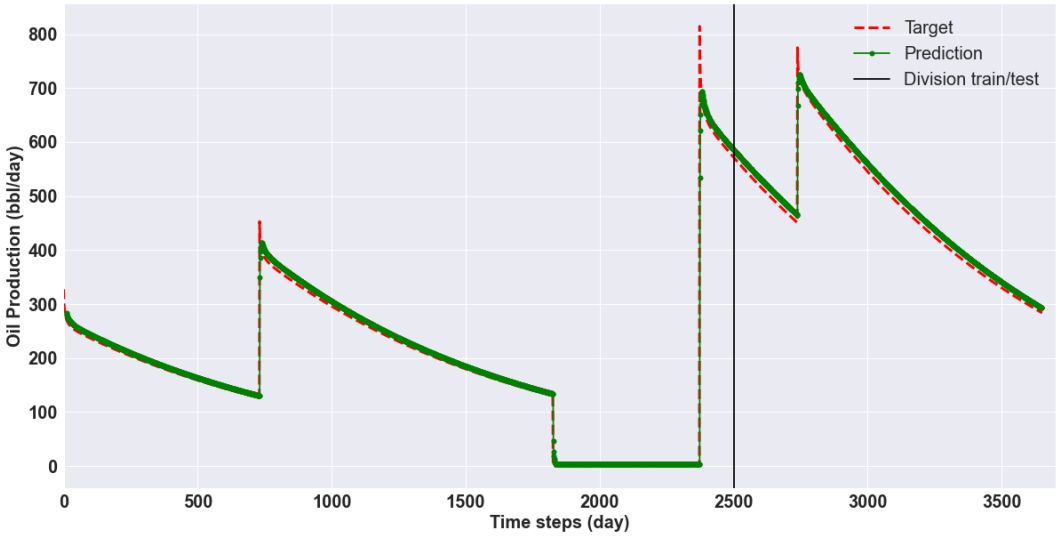

Figure 4.35: LSTM Model 1 oil production forecast in Dataset 3, with *forward-days* = 1, *look-back* = 10 and RMSE= 1.71.

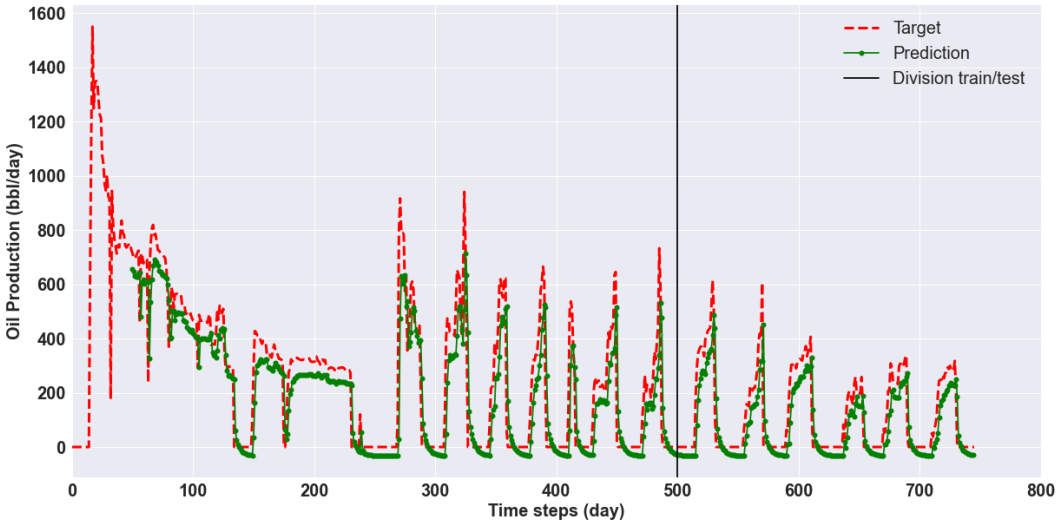

Figure 4.36: LSTM Model 1 oil production forecast in Dataset 4, with *forward-days* = 1, *look-back* = 10 and RMSE= 12.64.

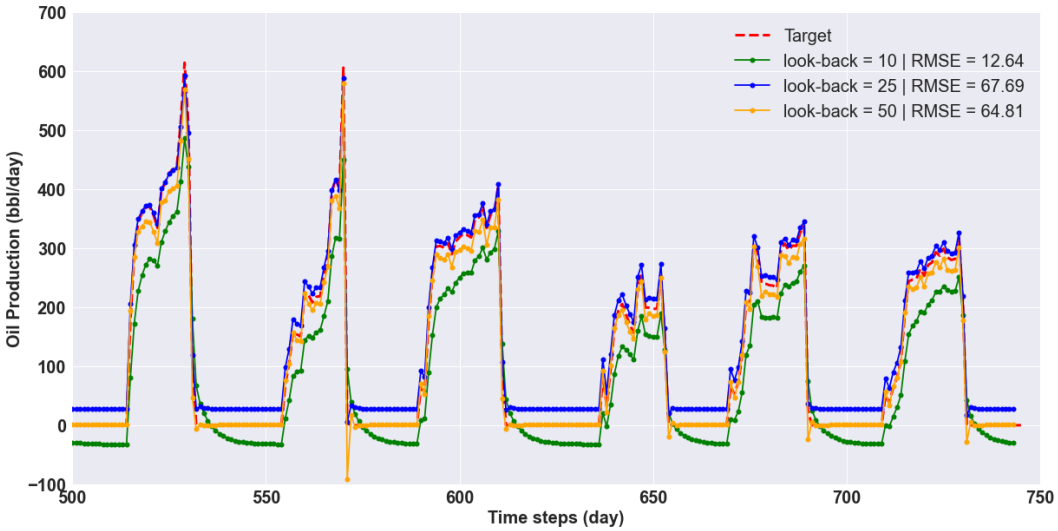

Figure 4.37: LSTM Model 1 oil production forecast in Dataset 4, with *forward-days* = 1, *look-back*=$\{10, 25, 50\}$.

Using forward-days = 1 *we observe a low RMSE. This result is expected since the LSTM can adjust the error at each time step.*

4.2.1.2 Forward-days = 10

Table 4.6 shows the RMSE for each look-back, in each dataset for forward-days = 10. For Datasets 1, the best RMSE was with look-back = 50. For Datasets 2, 3 and 4, the best RMSE was with look-back = 10. Figures 4.38, 4.39, 4.40, 4.41 shows the best results for Datasets 1, 2, 3 and 4, respectively. Figure 4.42 shows the results in test set of Dataset 4 for each look-back. For Datasets 1, 2 and 3, the results were so similar, that there is no visual difference in test sets.

	look-back	RMSE
Dataset 1	10	3.98
	25	4.85
	50	1.00
Dataset 2	10	3.12
	25	06.21
	50	09.71
Dataset 3	10	2.81
	25	6.31
	50	12.31
Dataset 4	10	102.66
	25	105.55
	50	125.97

Table 4.6: Summary of the LSTM Model 1 results when the input was the oil rate production, with *look-back* = $\{10, 25, 50\}$ and forward-days = 10.

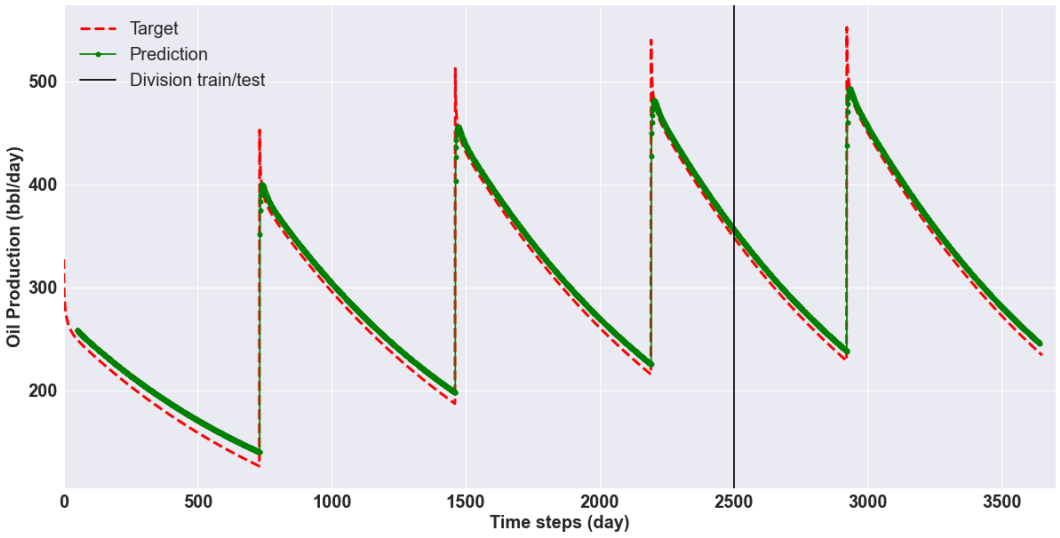

Figure 4.38: LSTM Model 1 oil production forecast in Dataset 1, with *forward-days* = 10, *look-back* = 50 and RMSE= 1.00.

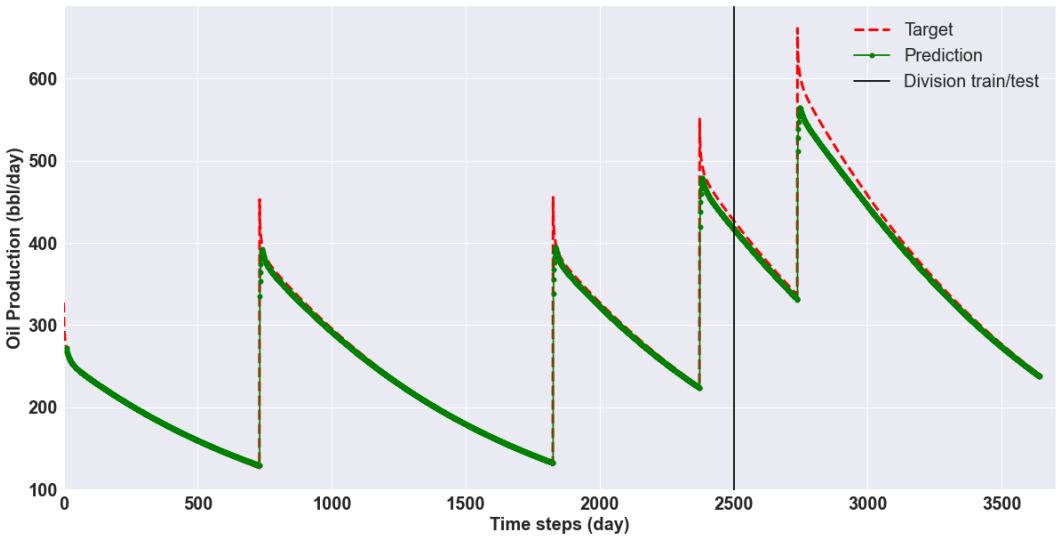

Figure 4.39: LSTM Model 1 oil production forecast in Dataset 2, with *forward-days* = 10, *look-back* = 10 and RMSE= 03.12.

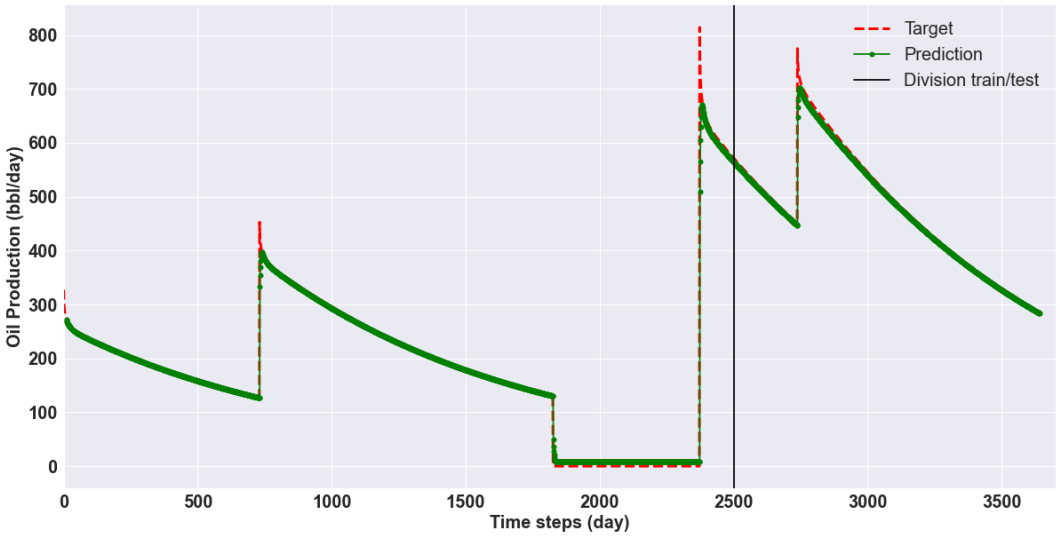

Figure 4.40: LSTM Model 1 oil production forecast in Dataset 3, with *forward-days* = 1, *look-back* = 10 and RMSE= 1.71.

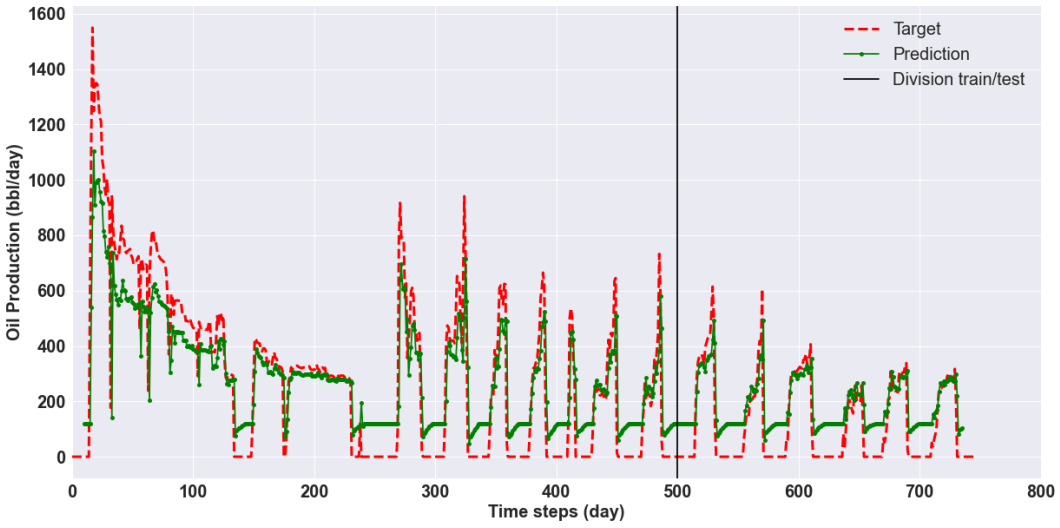

Figure 4.41: LSTM Model 1 oil production forecast in Dataset 4, with *forward-days* = 1, *look-back* = 10 and RMSE= 102.66.

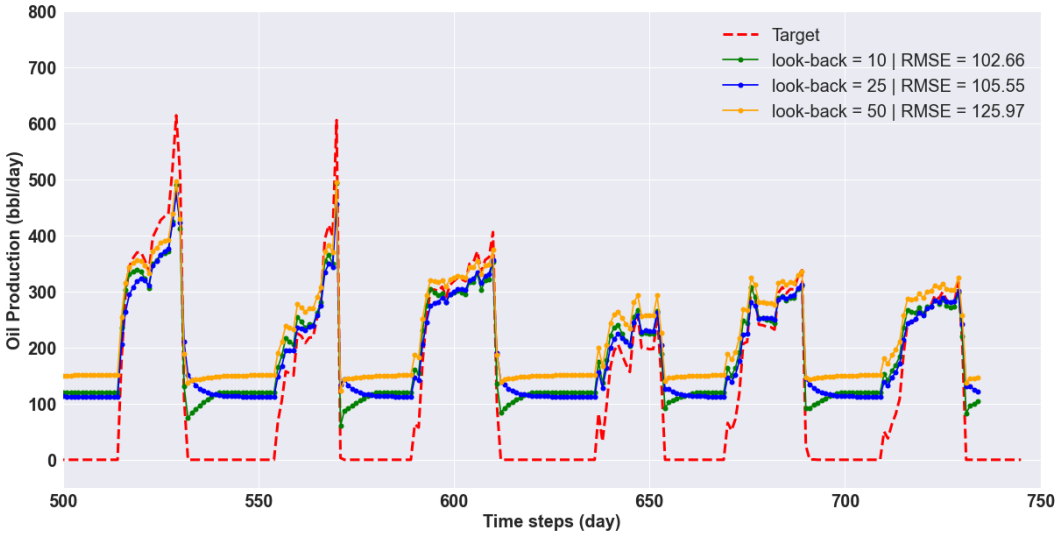

Figure 4.42: LSTM Model 1 oil production forecast in Dataset 4, with *forward-days* $= 10$, *look-back*$=\{10, 25, 50\}$.

Using forward-days $= 10$ *we note a sensible increment in RMSE, comparing with the first cases using forward-days* $= 1$*, mainly in Dataset 4. For Dataset 1, 2 and 3, the results are similar. However since Dataset 4 correspond to a real data with more complex behavior, it is more sensible in prediction.*

4.2.1.3 Forward-days = 50

Table 4.7 shows the RMSE for each look-back, in each dataset for forward-days $= 50$.

Figure 4.43 shows the results of Model 1 for each look-back in test set of Dataset 1 and Figure 4.44 shows the best outcome for all Dataset 1, with look-back $= 200$. *Figure 4.45 shows the results of Model 1 for each look-back in test set of Dataset 2 and Figure 4.46 shows the best result for all Dataset 2, with look-back* $= 25$. *Figure 4.47 shows the results of Model 1 for each look-back in test set of Dataset 3 and Figure 4.48 shows the best outcome for all Dataset 3, with look-back* $= 100$. *Figure 4.49 shows the best outcome for all Dataset 4, with look-back* $= 50$.

	look-back	RMSE
Dataset 1	25	12.99
	50	10.92
	100	15.86
Dataset 2	25	12.20
	50	19.01
	100	30.79
Dataset 3	25	11.97
	50	23.04
	100	13.53
Dataset 4	25	172.87
	50	161.67
	100	170.73

Table 4.7: Summary of the LSTM Model 1 results when the input was the oil rate production, with *look-back* $= \{25, 50, 100\}$ and forward-days $= 50$.

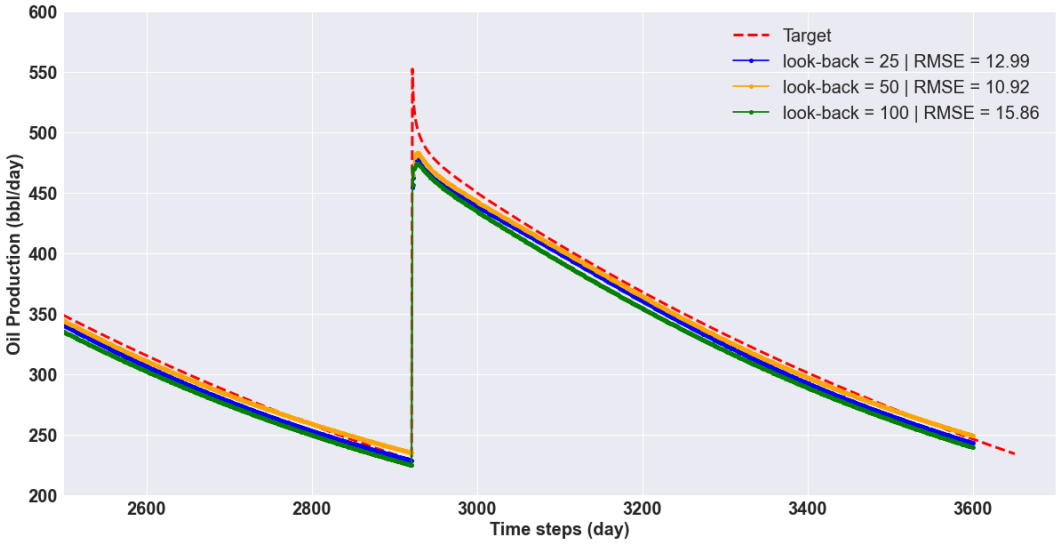

Figure 4.43: LSTM Model 1 oil production forecast in Dataset 1, with *forward-days* $= 50$, *look-back*$=\{25, 50, 100\}$.

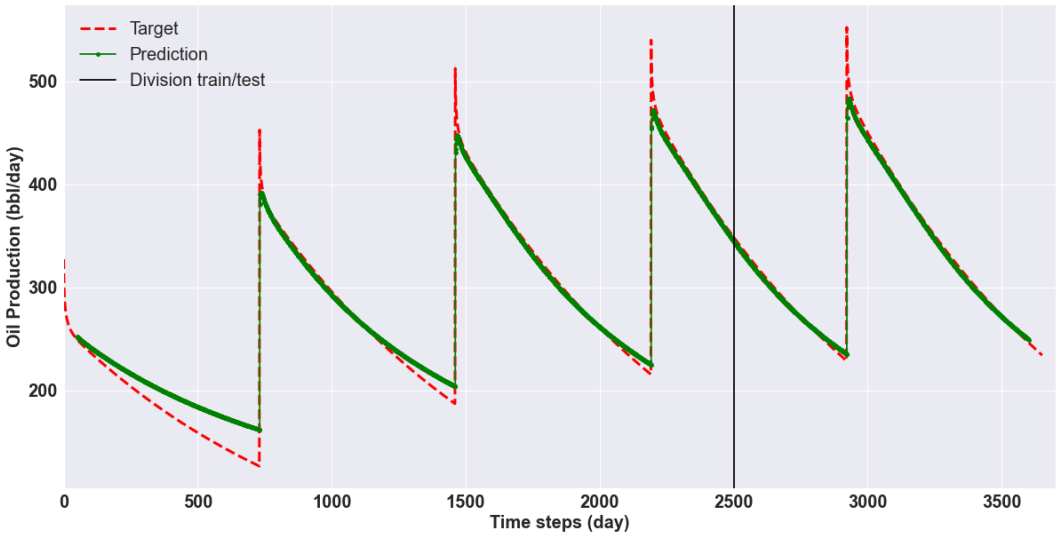

Figure 4.44: LSTM Model 1 oil production forecast in Dataset 1, with *forward-days* = 50, *look-back* = 50 and RMSE= 10.96.

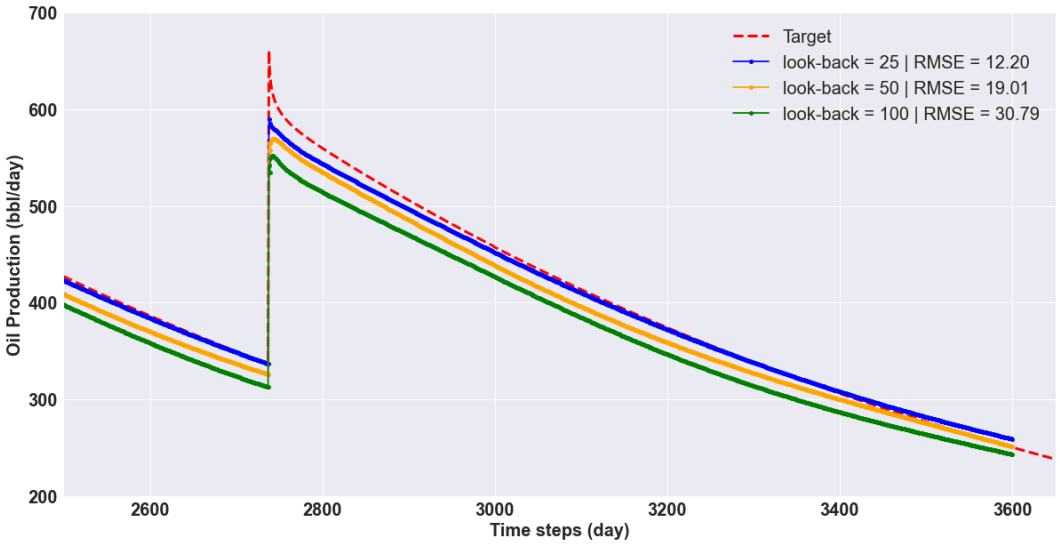

Figure 4.45: LSTM Model 1 oil production forecast in Dataset 2, with *forward-days* = 50, *look-back*=$\{25, 50, 100\}$.

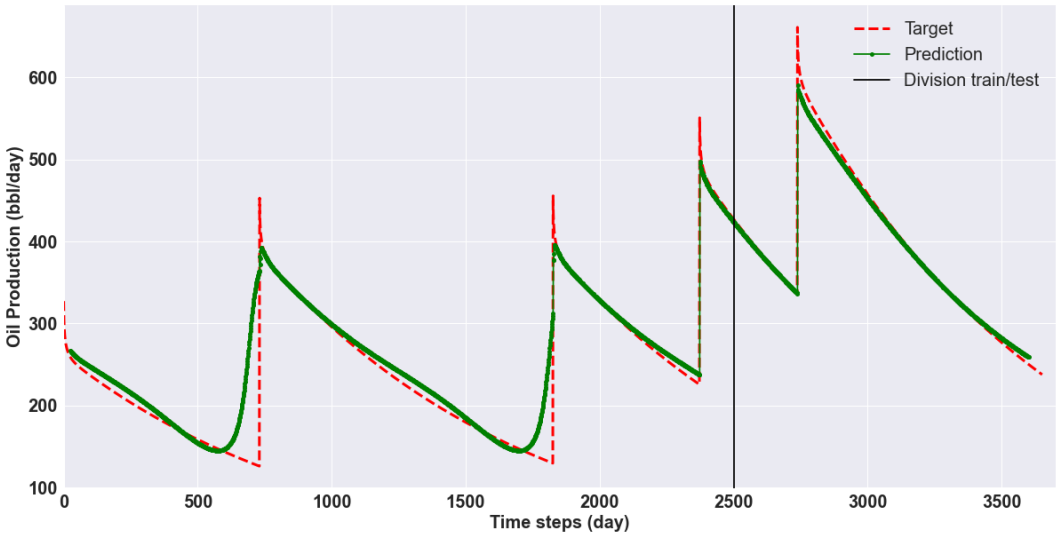

Figure 4.46: LSTM Model 1 oil production forecast in Dataset 2, with *forward-days* = 50, *look-back* = 25 and RMSE= 12.20.

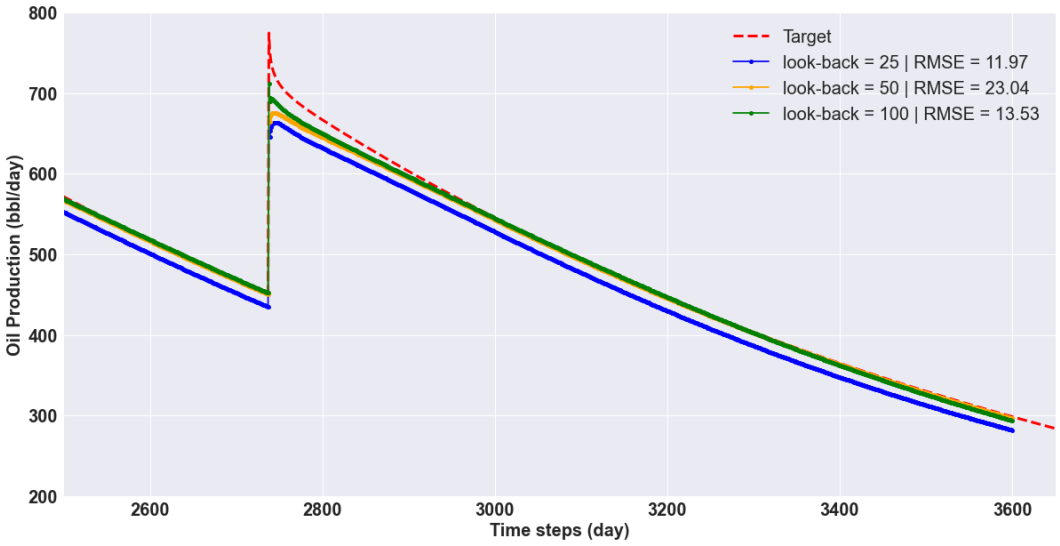

Figure 4.47: LSTM Model 1 oil production forecast in Dataset 3, with *forward-days* = 50, *look-back*=$\{25, 50, 100\}$.

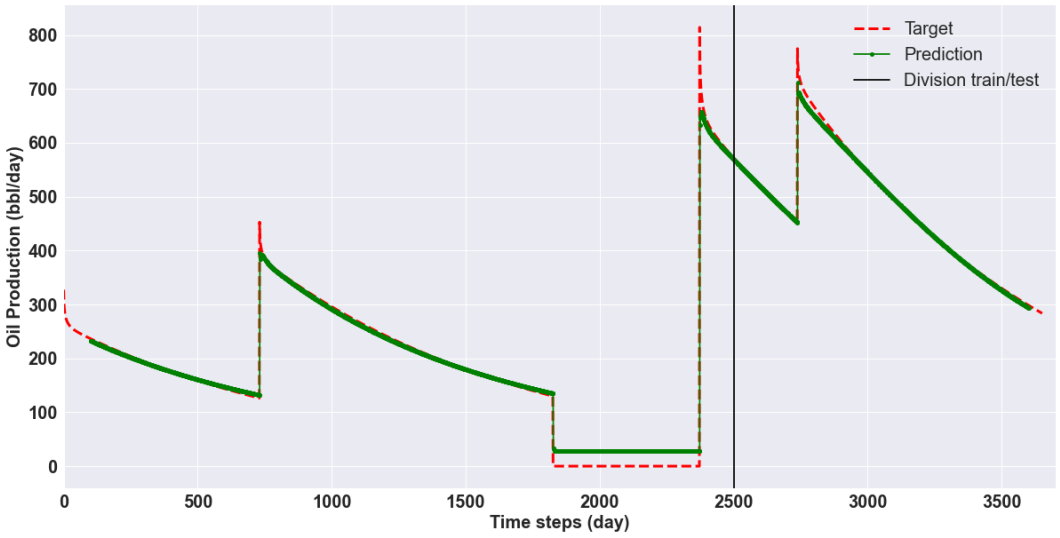

Figure 4.48: LSTM Model 1 oil production forecast in Dataset 3, with *forward-days* = 50, *look-back* = 100 and RMSE= 11.97.

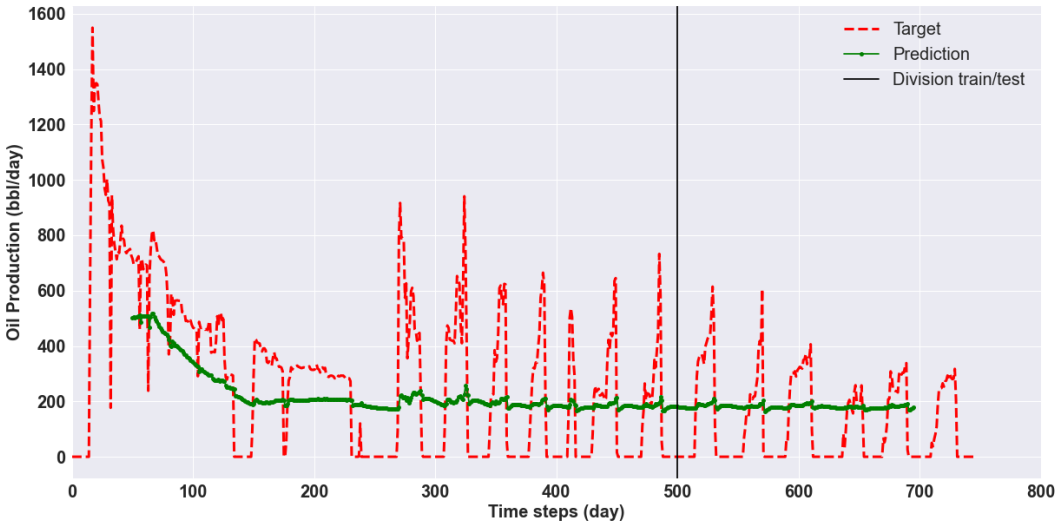

Figure 4.49: LSTM Model 1 oil production forecast in Dataset 4, with *forward-days* = 50, *look-back* = 10 and RMSE= 161.67.

Note that the results for Dataset 4 was not satisfactory. Figure 4.49 shows the Model 1 results for forward-days = 50 predictions. For that reason, we decided to use Model 2 to forecast forward-days = 50 in Dataset 4. Table 4.8 shows the RMSE of Model 2 for each look-back, in Dataset 4 with forward-days = 50. Figure 4.50 shows the results with best RMSE in Dataset 4, for Model 2

forward-days = 50 *predictions. Figure 4.51 shows the results in Dataset 4 test set for each look back and forward-days* = 50.

	look-back	RMSE
Dataset 4	25	284.89
	50	271.32
	100	290.13

Table 4.8: Summary of the LSTM Model 2 results when the input was the oil rate production, with *look-back* = $\{25, 50, 100\}$ and forward-days = 50 for Dataset 4.

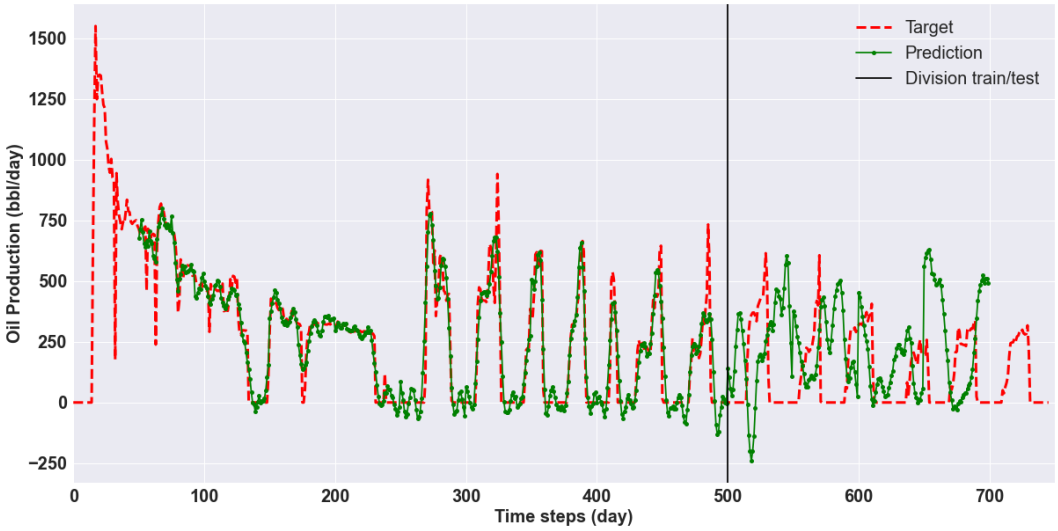

Figure 4.50: LSTM Model 1 oil production forecast in Dataset 4, with *forward-days* = 50, *look-back* = 50 and RMSE= 271.32.

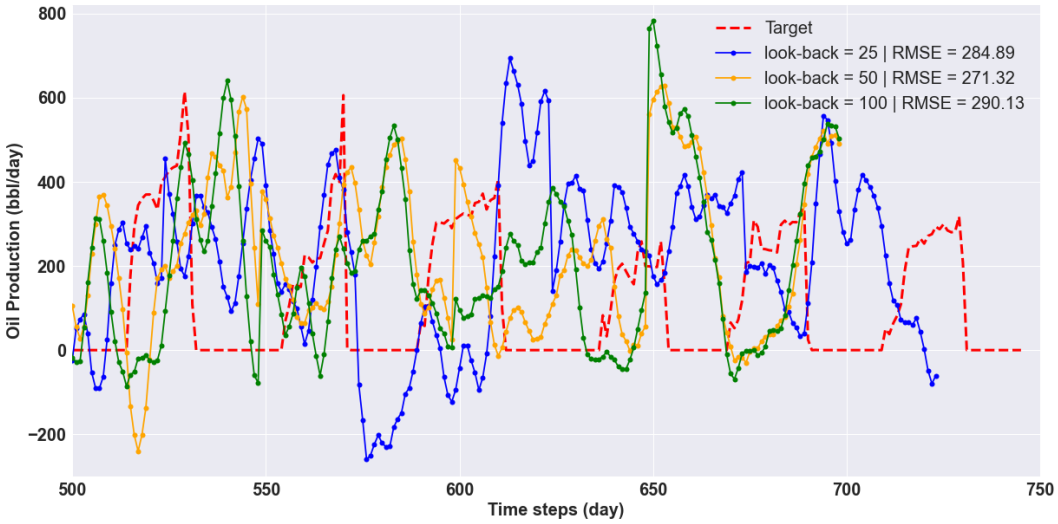

Figure 4.51: LSTM Model 2 oil production forecast in Dataset 4, with *forward-days* = 50, *look-back*=$\{10, 25, 50\}$.

The results indicate that datasets 1, 2 and 3 reasonably tolerate forecasts up to 50 days. They are synthetic data with a certain periodicity behavior that the network captures with reasonable quality. For the case of example 4, the prediction was not acceptable, due to its greater complexity.

4.2.1.4 Forward-days = 100

Table 4.12 shows the Model 1 RMSE for each look-back, For Datasets 1, 2 and 3 and forward-days = 100. *Since Dataset 4 has only 746 days, there is not enough data to structure with look-back* = 200 *and forward-days* = 100. *For that reason, we only use look-back* = $\{50, 100\}$. *Table 4.10 displays the Model 2 RMSW for each look-back in Dataset 4.*

Figure 4.52 shows the results of Model 1 for each look-back in test set of Dataset 1 and Figure 4.53 shows the best outcome for all Dataset 1, with look-back = 200. *Figure 4.54 shows the results of Model 1 for each look-back in test set of Dataset 2 and Figure 4.55 shows the best result for all Dataset 2, with look-back* = 100. *Figure 4.56 shows the results of Model 1 for each look-back in test set of Dataset 3 and Figure 4.57 shows the best outcome for all Dataset 3, with look-back* = 100. *Figure 4.58 shows the results of Model 2 for each look-back in test set of Dataset 4 and Figure 4.59 shows the best outcome for all Dataset 1, with look-back* = 200.

	look-back	RMSE
Dataset 1	50	23.93
	100	20.07
	200	17.09
Dataset 2	50	39.17
	100	19.24
	200	28.80
Dataset 3	50	20.23
	100	17.99
	200	26.87

Table 4.9: Summary of the LSTM Model 1 results when the input was the oil rate production, with *look-back* $= \{25, 50, 100\}$ and forward-days $= 100$.

	look-back	RMSE
Dataset 4	50	237.97
	100	209.66

Table 4.10: LSTM Model 2 results when the input was the oil rate production, with *look-back* $= \{50, 100\}$ and forward-days $= 100$ for Dataset 4.

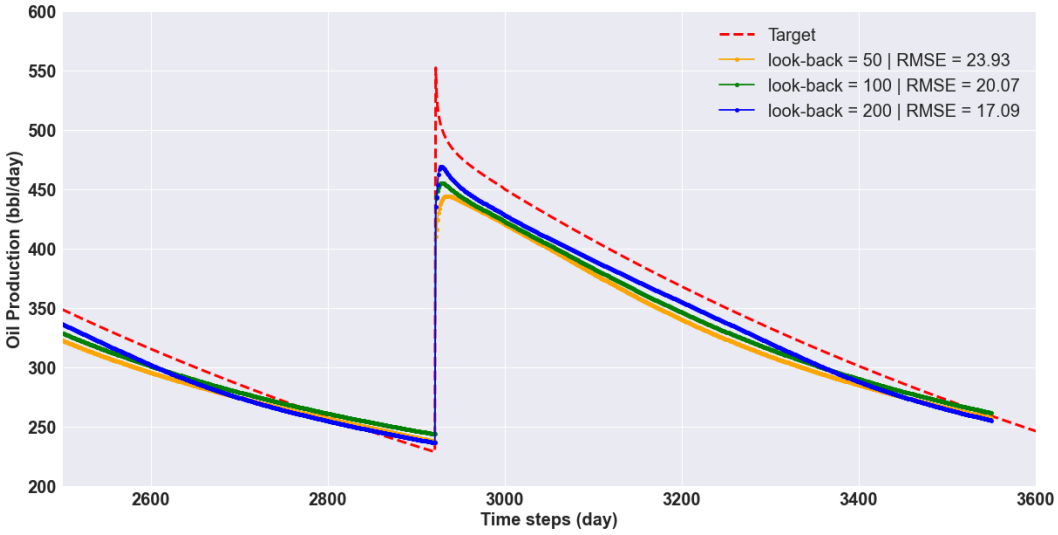

Figure 4.52: LSTM Model 1 oil production forecast in Dataset 1, with *forward-days* $= 100$, *look-back*$=\{50, 100, 200\}$.

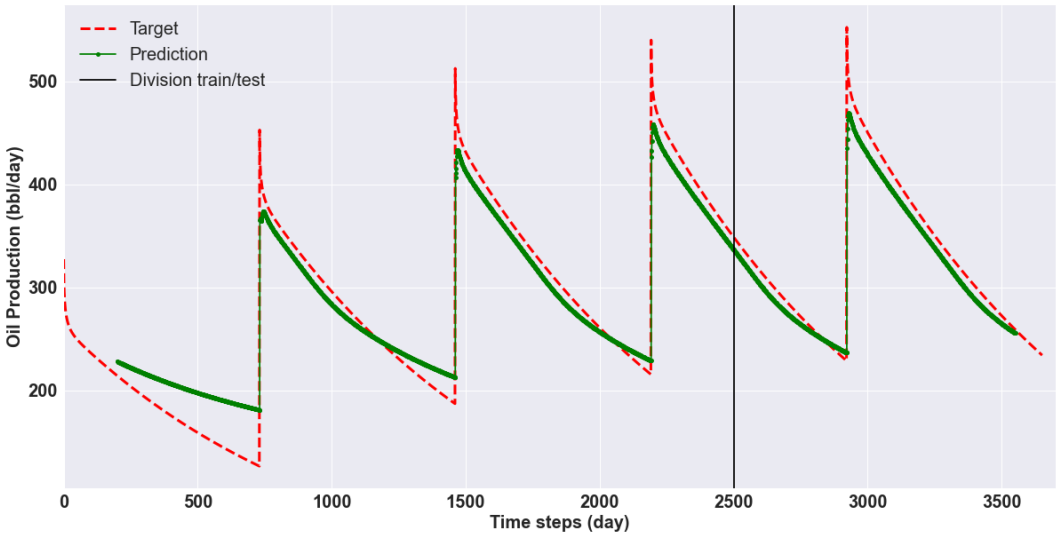

Figure 4.53: LSTM Model 1 oil production forecast in Dataset 1, with *forward-days* = 100, *look-back* = 200 and RMSE= 17.09.

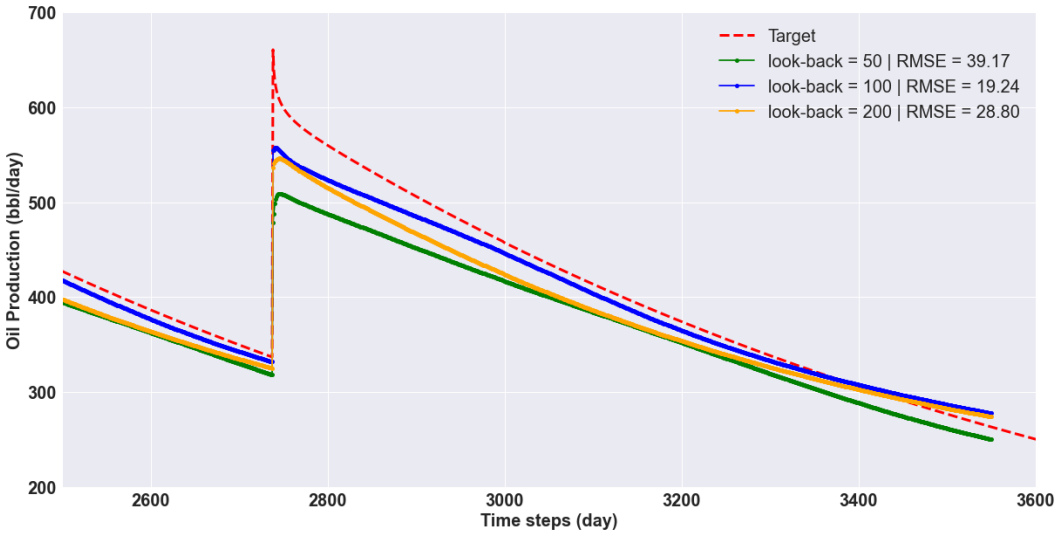

Figure 4.54: LSTM Model 1 oil production forecast in test set of Dataset 2, with *forward-days* = 100, *look-back*=$\{50, 100, 200\}$.

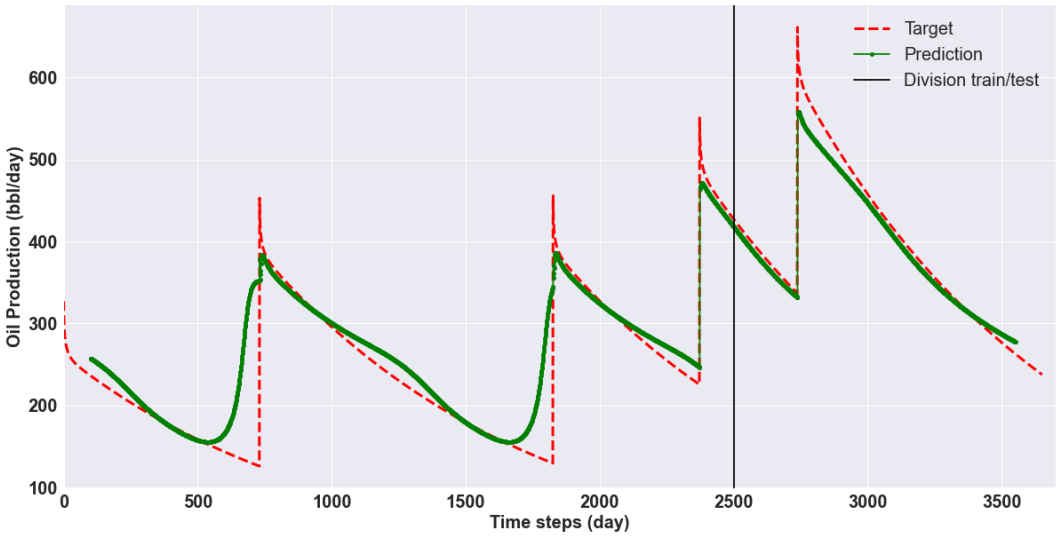

Figure 4.55: LSTM Model 1 oil production forecast in Dataset 2, with *forward-days* = 100, *look-back* = 100 and RMSE= 19.24.

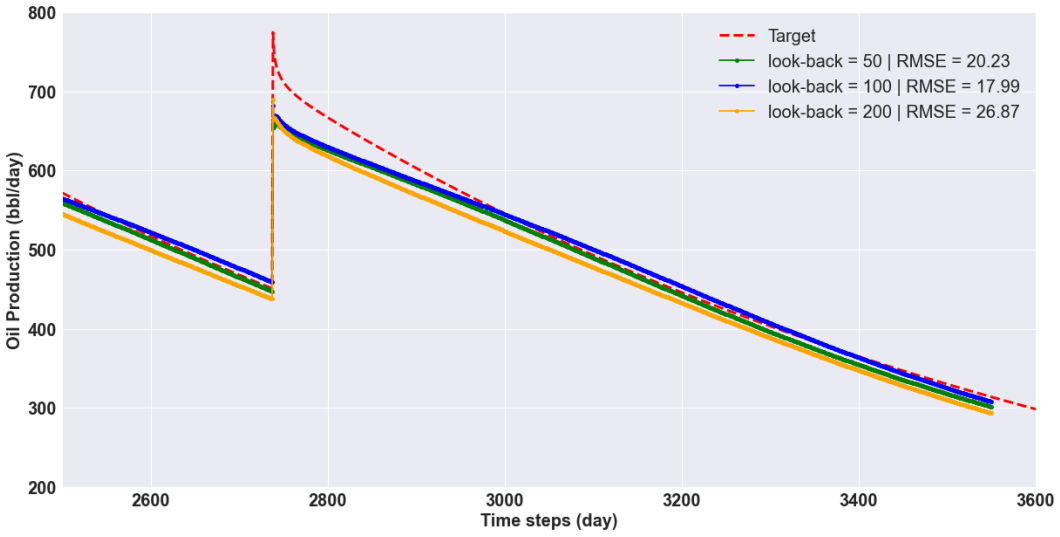

Figure 4.56: LSTM Model 1 oil production forecast in Dataset 3, with *forward-days* = 100, *look-back*=$\{50, 100, 200\}$.

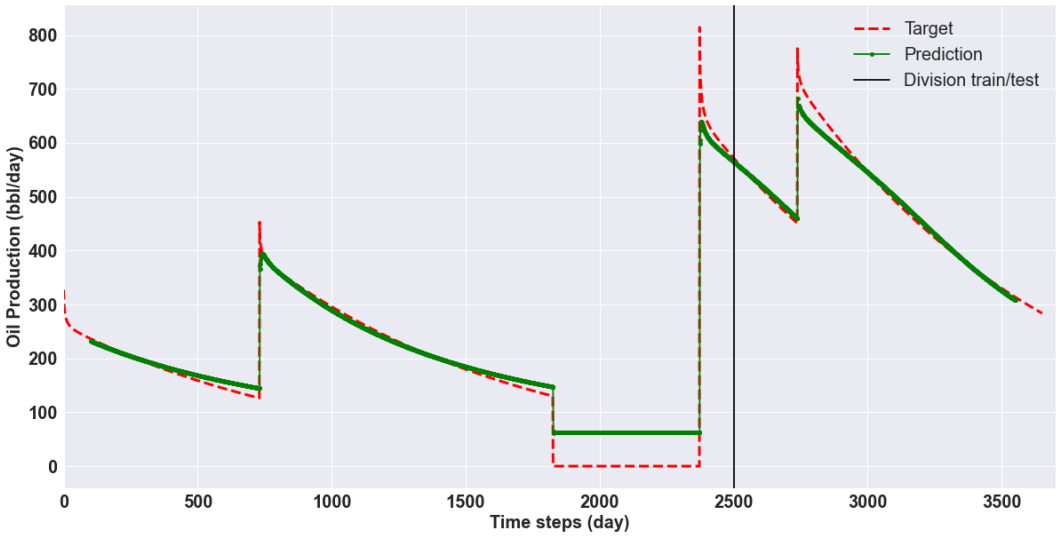

Figure 4.57: LSTM Model 1 oil production forecast in Dataset 3, with *forward-days* = 100, *look-back* = 100 and RMSE= 17.99.

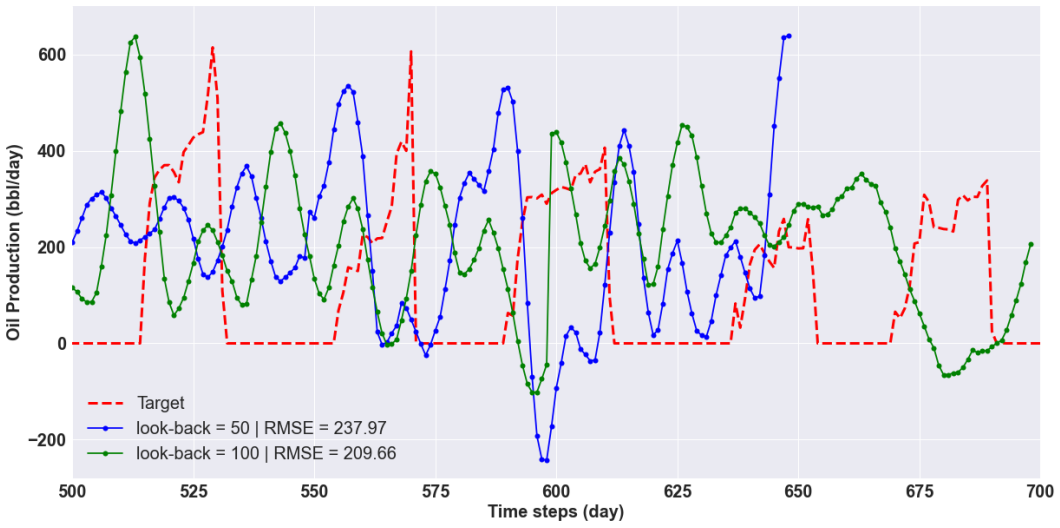

Figure 4.58: LSTM Model 2 oil production forecast in Dataset 4, with *forward-days* = 100, *look-back*=$\{50, 100\}$.

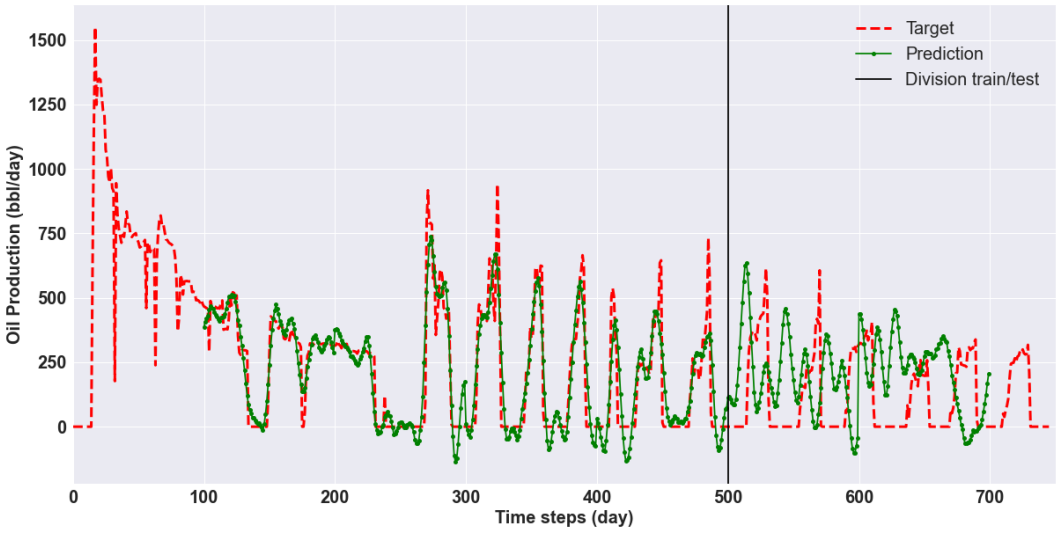

Figure 4.59: LSTM Model 1 oil production forecast in Dataset 4, with *forward-days* = 100, *look-back* = 100 and RMSE= 209.66.

4.2.2 Input: Oil rate and pressure

As already observed in the previous section, for relatively high values of forward-days, the predictions are not satisfactory for dataset 4. In the cases of datasets 1, 2 and 3, due to their periodic characteristics, the results seems acceptable considering the long period of prediction. The results of this section reinforce the behavior already observed in the case forward-days = 50. If we look for predictions for longer periods such as forward-days = 50 or 100, the results indicate that we should insert other information in the system. In the next section, we add pressure data as a second feature.

The daily oil rate production and the down hole pressure were the inputs for LSTM model, to forecast forward-days = 50. From now on, since there is more then one feature in the inputs, we decided to use Model 2 to all predictions.

4.2.2.1 Forward-days = 50

Table 4.11 shows the RMSE for each look-back, in each dataset for forward-days = 50.

Figure 4.60 shows the results of Model 2 for each look-back in test set of Dataset 1 and Figure 4.61 shows the best outcome for all Dataset 1, with look-back = 50. Figure 4.62 shows the results of Model 2 for each look-back in

test set of Dataset 2 and Figure 4.63 shows the best result for all Dataset 2, with look-back = 50. *Figure 4.64 shows the results of Model 2 for each look-back in test set of Dataset 3 and Figure 4.65 shows the best outcome for all Dataset 3, with look-back* = 50. *Figure 4.66 shows the results of Model 2 for each look-back in test set of Dataset 4 and Figure 4.67 shows the best outcome for all Dataset 4, with look-back* = 25.

	look-back	RMSE
Dataset 1	25	57.34
	50	42.37
	100	55.48
Dataset 2	25	50.77
	50	35.31
	100	41.48
Dataset 3	25	63.71
	50	34.09
	100	45.77
Dataset 4	25	269.56
	50	224.79
	100	302.00

Table 4.11: Summary of the LSTM Model 2 results when the input was the oil rate production and the down hole pressure, with *look-back* = $\{25, 50, 100\}$ and forward-days = 50.

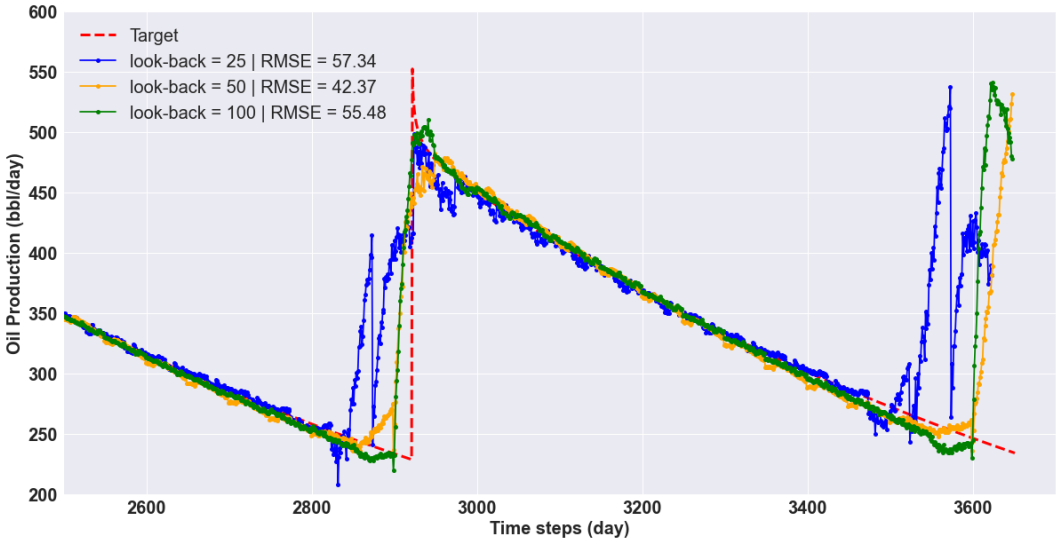

Figure 4.60: LSTM Model 2 oil production forecast in Dataset 1, with *forward-days* = 50, *look-back*=$\{25, 50, 100\}$.

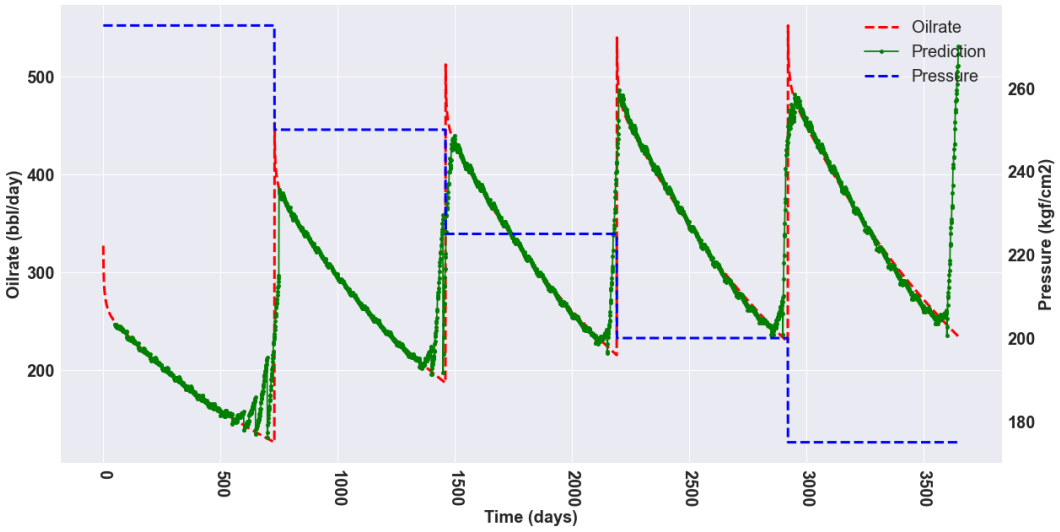

Figure 4.61: LSTM Model 2 oil production forecast in Dataset 1, with *forward-days* = 50, *look-back* = 50 and RMSE= 42.37.

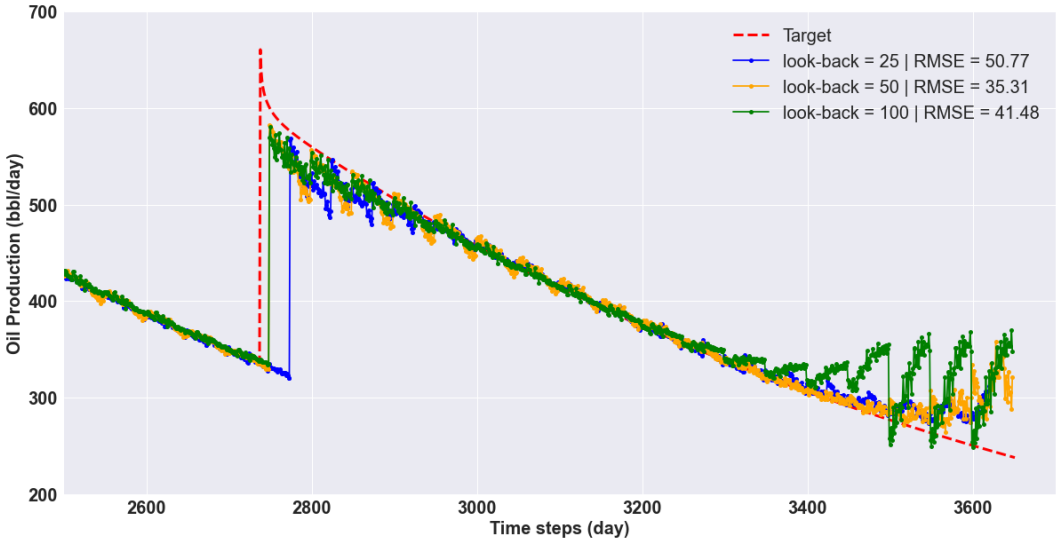

Figure 4.62: LSTM Model 2 oil production forecast in Dataset 2, with *forward-days* $= 50$, *look-back*$=\{25, 50, 100\}$.

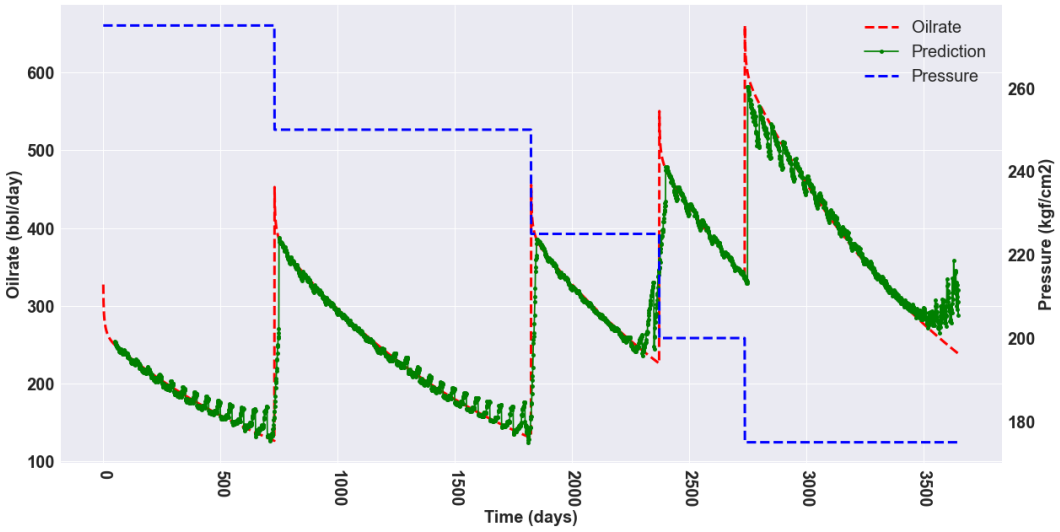

Figure 4.63: LSTM Model 2 oil production forecast in Dataset 2, with *forward-days* $= 50$, *look-back* $= 50$ and RMSE$= 35.31$.

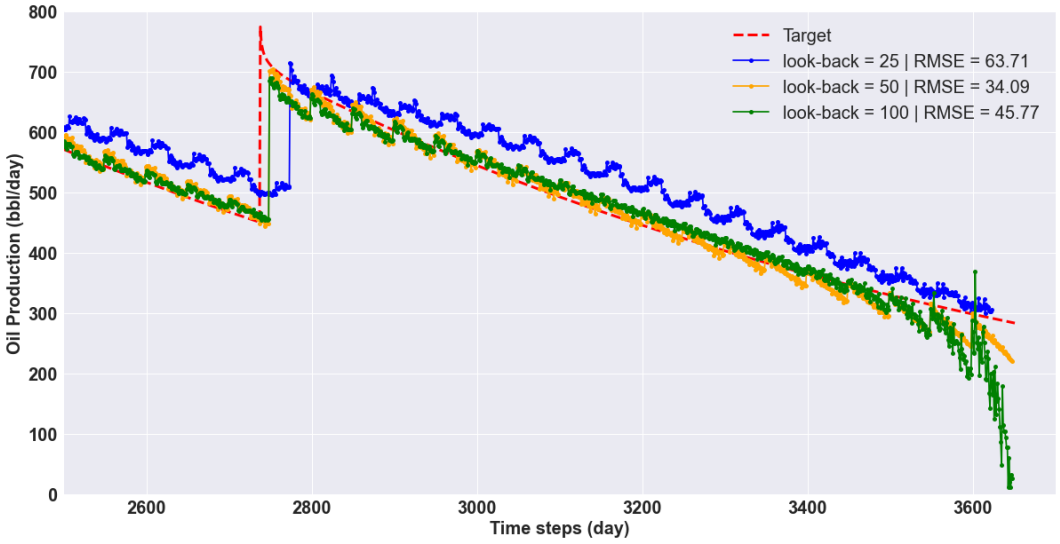

Figure 4.64: LSTM Model 2 oil production forecast in test set of Dataset 3, with *forward-days* = 50, *look-back*=$\{25, 50, 100\}$.

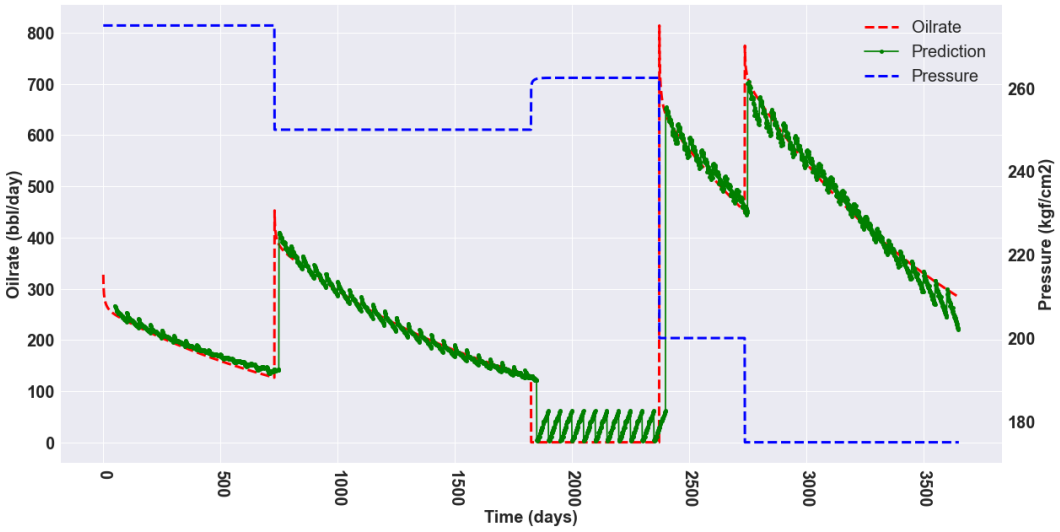

Figure 4.65: LSTM Model 2 oil production forecast in Dataset 3, with *forward-days* = 50, *look-back* = 50 and RMSE = 34.09.

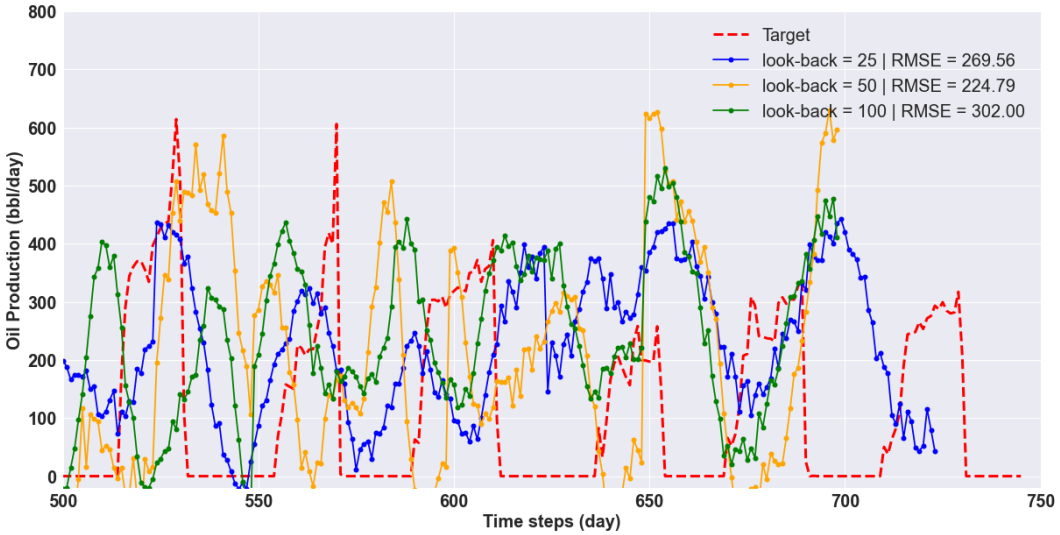

Figure 4.66: LSTM Model 2 oil production forecast in test set of Dataset 4, with *forward-days* = 50, *look-back*=$\{25, 50, 100\}$.

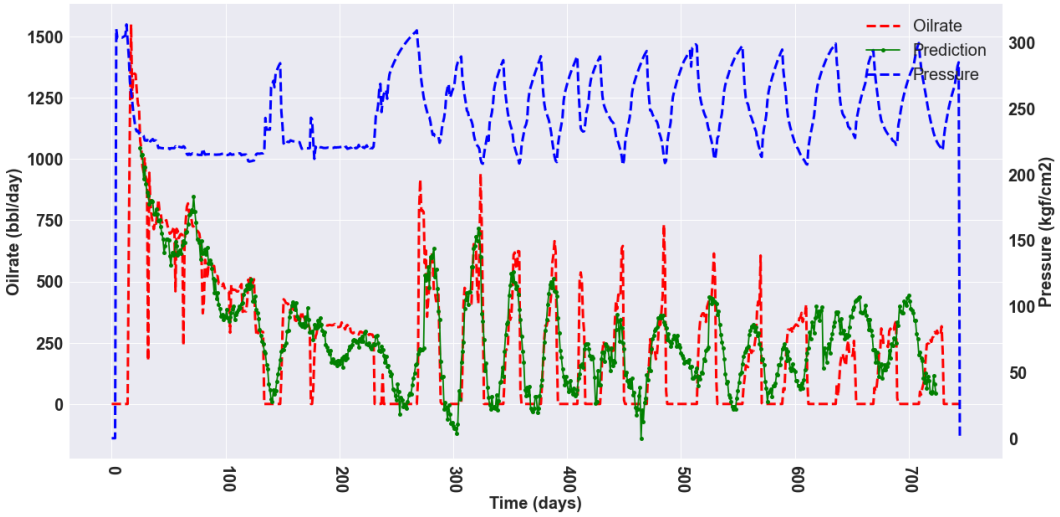

Figure 4.67: LSTM Model 2 oil production forecast in Dataset 4, with *forward-days* = 50, *look-back* = 25 and RMSE= 224.79.

The results indicate that datasets 1, 2 and 3 can still reasonably tolerate forecasts up to 50 days. The incorporation of pressure into the inputs, introduced some noise into the results, that could be seen int the graphics and in the RMSE's. For the case of example 4, the prediction was still not acceptable, and incorporating the pressure to the inputs did not bring better outcomes.

4.2.3 Input: Oil rate, pressure and future pressure

The results obtained in previous section, showed that including the pressure to the inputs, could bring some noise to the results.Therefore, we decided to include the pressure from the days we were about to predict as an input. The idea behind this atempt is that if the network could reasonable learn the relation between oil production and pressure, it could produce better predictions of oil production if it knows the pressure from these days. So, the LSTM neural network has 3 inputs: the oil rate production of look-back days, the down hole pressure of look-back days, and the down hole pressure of forward-back days.

According to the method adopted to transform a time series information to the inputs of an LSTM neural network, it is necessary that all inputs has the same dimensions. Hence, the three inputs used in this section must be sequences of the same size, requiring that look-back and forward-back assume the same value.

In this section, we use the LSTM Model 2 to forecast forward-days $= \{50, 100\}$*. Table summarizes the results for each dataset.*

	look-back = *forward-days*	RMSE
Dataset 1	50	07.51
	100	13.61
Dataset 2	50	39.01
	100	21.36
Dataset 3	50	45.89
	100	102.93
Dataset 4	50	147.11
	100	174.11

Table 4.12: LSTM Model 2 results when the input was the oil rate production, the past and the future down hole pressure with forward-days $= \{50, 100\}$.

Figure 4.68 shows the results of Model 2 for each forward-days in test set of Dataset 1 and Figure 4.69 shows the best outcome for all Dataset 1, with forward-days $= 50$*. Figure 4.70 shows the results of Model 2 for each look-back in test set of Dataset 2 and Figure 4.71 shows the best result for all Dataset 2, with forward-days* $= 100$*. Figure 4.72 shows the results of Model 2 for each look-back in test set of Dataset 3 and Figure 4.57 shows the best outcome for all Dataset 3, with look-back* $= 100$*. Figure 4.58 shows the results of Model*

2 for each look-back in test set of Dataset 4 and Figure 4.75 shows the best outcome for all Dataset 4, with look-back = 50.

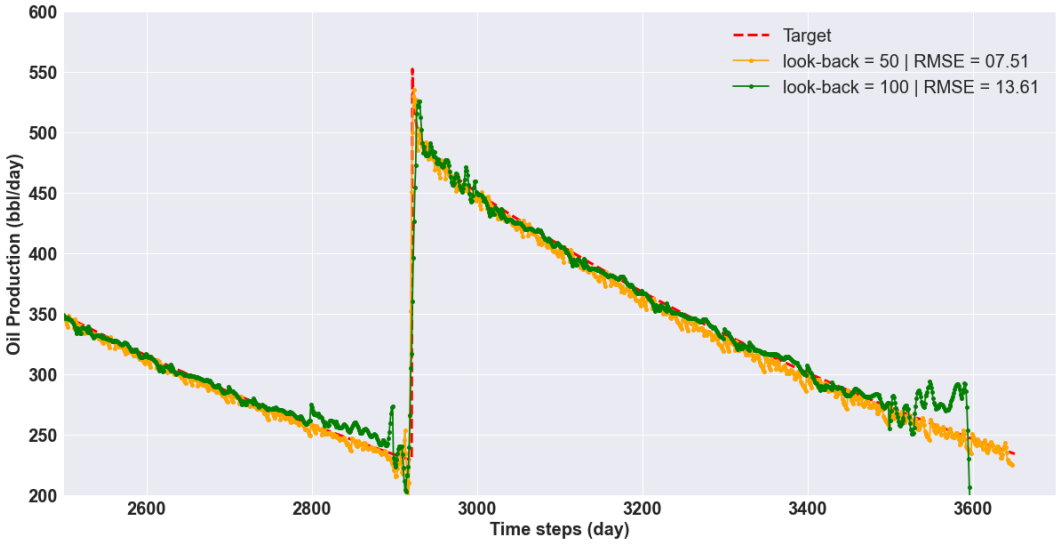

Figure 4.68: LSTM Model 2 oil production forecast in Dataset 1, with 3 inputs and *forward-days* = $\{50, 100\}$.

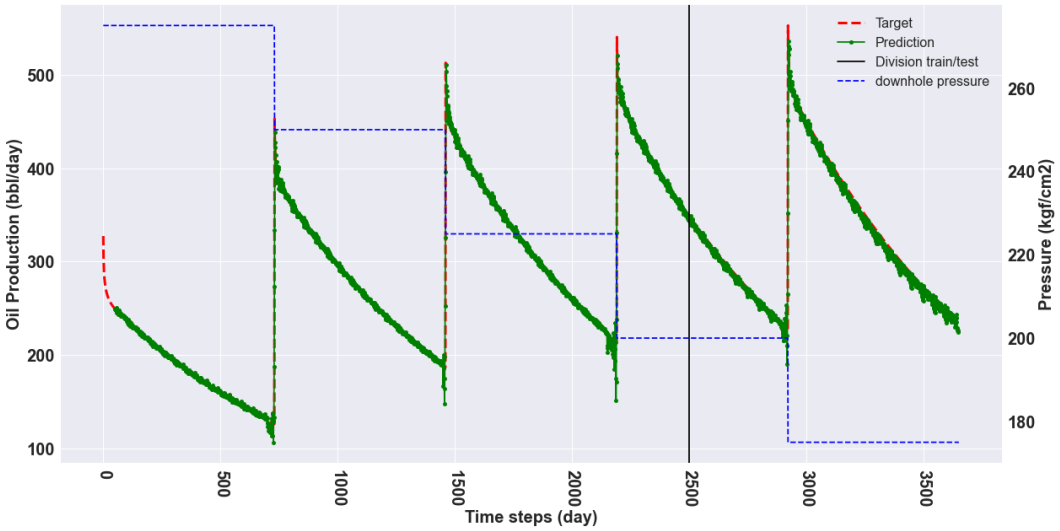

Figure 4.69: LSTM Model 2 oil production forecast in Dataset 1, with 3 inputs and *forward-days* = 100, *look-back* = 100 and RMSE= 07.51.

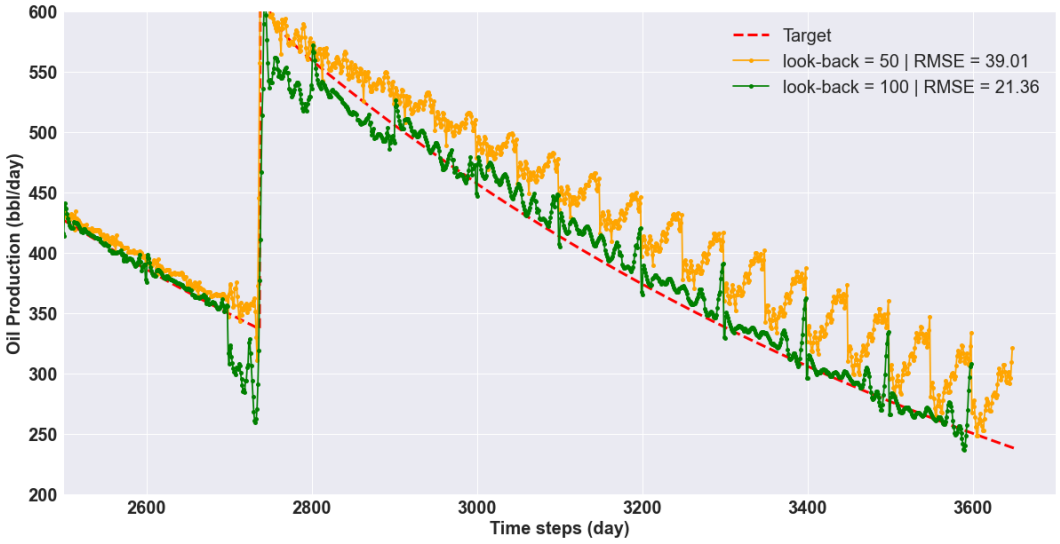

Figure 4.70: LSTM Model 2 oil production forecast in Dataset 2, with 3 inputs and *forward-days* $= \{50, 100\}$.

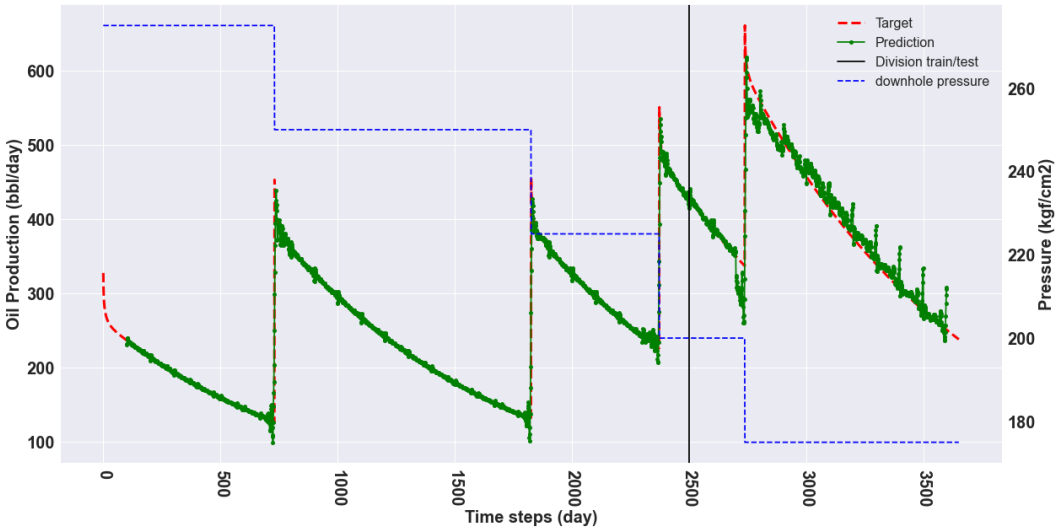

Figure 4.71: LSTM Model 2 oil production forecast in Dataset 2, with 3 inputs and *forward-days* $= 50$, *look-back* $= 50$ and RMSE$= 21.36$.

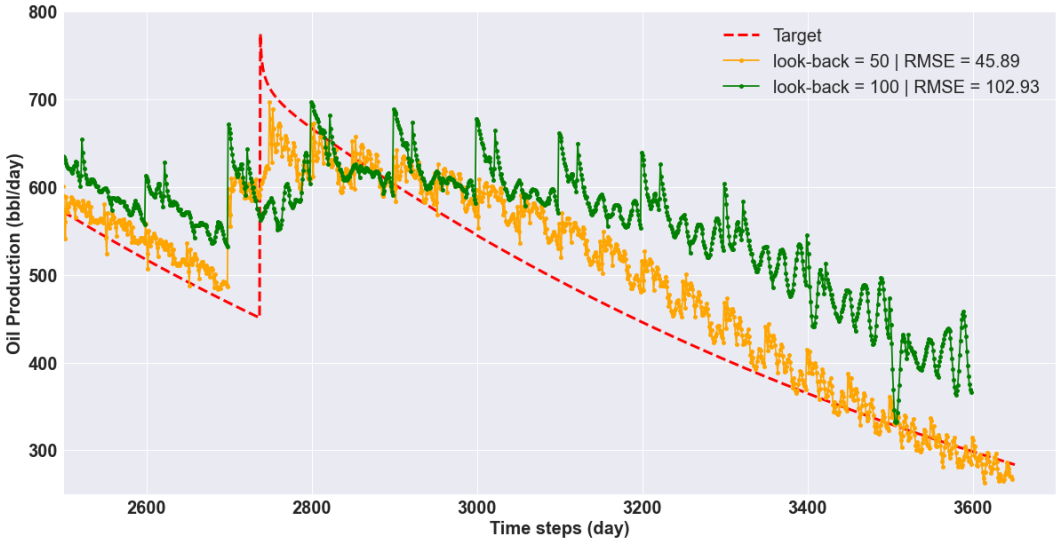

Figure 4.72: LSTM Model 2 oil production forecast in Dataset 3, with 3 inputs and *forward-days* $= \{50, 100\}$.

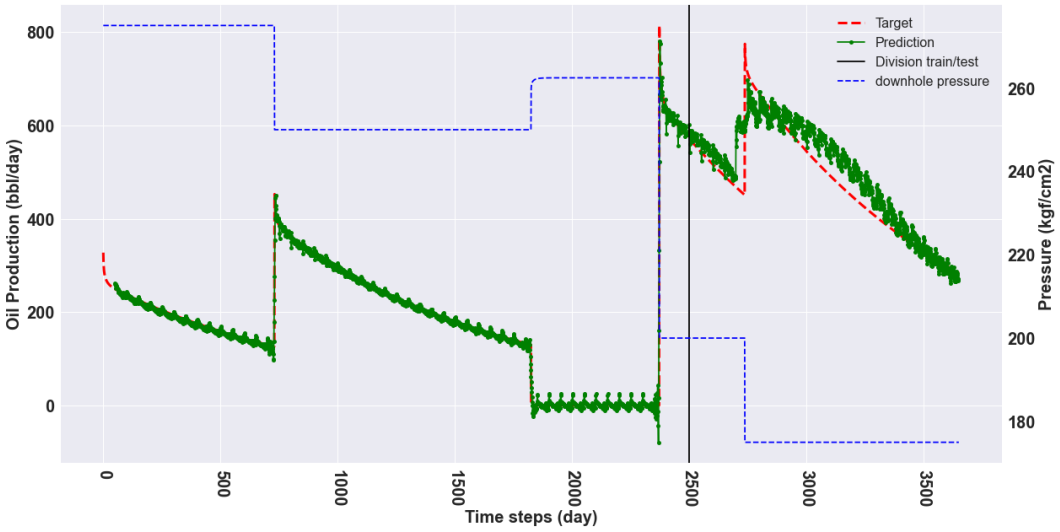

Figure 4.73: LSTM Model 2 oil production forecast in Dataset 3, with 3 inputs and *forward-days* $= 50$, *look-back* $= 50$ and RMSE $= 45.89$.

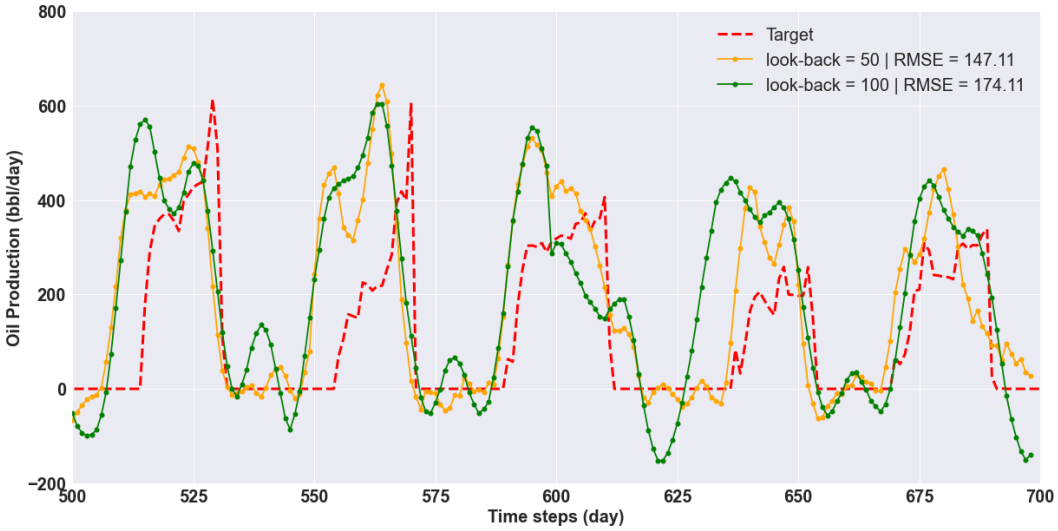

Figure 4.74: LSTM Model 2 oil production forecast in Dataset 4, with 3 inputs and *forward-days* $= \{50, 100\}$.

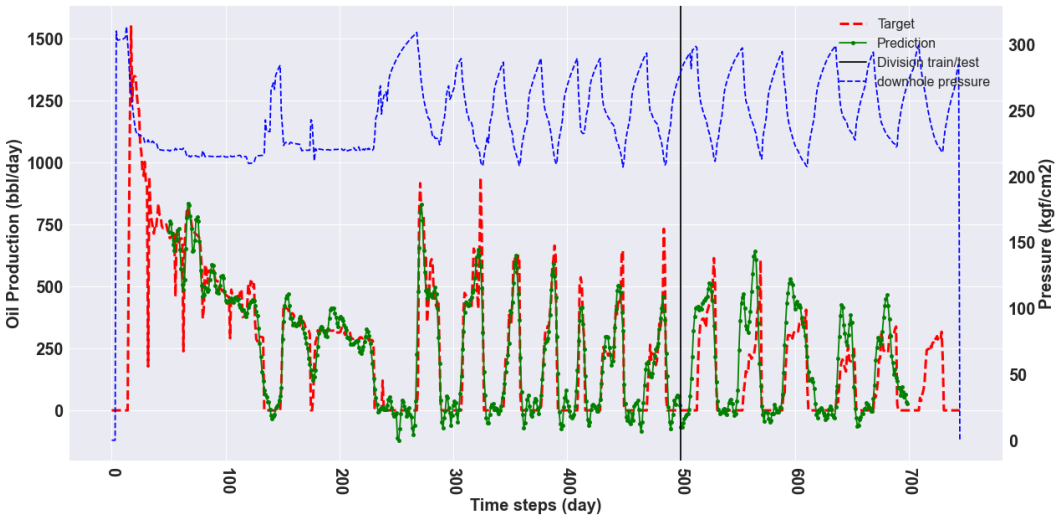

Figure 4.75: LSTM Model 2 oil production forecast in Dataset 4, with 3 inputs and *forward-days* $= 50$, *look-back* $= 50$ and RMSE $= 147.11$.

The results indicate that datasets 1, 2 3 and 4 can reasonably tolerate forecasts up to 100 days. The incorporation of future pressure into the inputs, improved a lot the results from previous section, suggesting that the network could learn some of the relation between oil production and pressure.

4.3 Results Comparison

In this section, we present the comparison of the models used in this work to forecast the oil production of a well. In most predictions shown in last sections, we tested three different values for look-back. In this section, we consider only the best outcome, wich means the result with lower RMSE. For that reason, we only consider one value of look-back for each prediction.

4.3.1 Random Forest x LSTM

Table 4.13 summarizes the results of LSTM Model 1 and Random Forest algorithms for Datasets 1, 2, 3 and 4 and forward-days $= \{1, 10, 50, 100\}$. *The prediction of forward-days = 10 in Dataset 1, was the only time Random Forest presented an slightly lower RMSE then LSTM. In all other predictions, LSTM had a better performance, proving to be a more appropriate technique to time series prediction.*

	forward-days	RF RMSE	LSTM RMSE
Dataset 1	1	9.64	1.34
	10	2.48	1.00
	50	34.35	10.92
	100	45.67	17.09
Dataset 2	1	22.58	3.75
	10	28.38	3.12
	50	45.19	12.20
	100	63.42	19.24
Dataset 3	1	36.37	1.71
	10	43.49	2.81
	50	71.73	11.97
	100	112.25	12.99
Dataset 4	1	73.07	12.64
	10	137.20	102.66
	50	203.41	161.67
	100	213.91	209.66

Table 4.13: Summary of the LSTM Model 1 and Random Forest results for forward-days $= \{1, 10, 50, 100\}$.

4.3.2
1 input x 3 inputs

In this work, we tested the LSTM neural network with different inputs. Initially, the network had 1 input: the oil rate production of look-back days. Then, we incorporated the pressure into the inputs, and the network had 2 inputs: the oil rate production and the down hole pressure of look-back days. Finally, we added the pressure from the days we were about to predict to the inputs so the network had 3 inputs: the oil rate production and the down hole pressure of look-back days and the down hole pressure of forward-days. The results obtained with three inputs were much superior to the ones generated with two inputs. For that reason, in this section, we compare the results of compare the results of one and three inputs for forward-days $= 100$.

Table 4.14 shows the RMSE's for each dataset in each input given to the network. For Datasets 1, 2 and 3, both inputs could genarate reasanable results. For Dataset 4, a real dataset with more complex information, the pressure brought a slightly improvement to the outcomes, producing a better RMSE.

	1 input RMSE	3 inputs RMSE
Dataset 1	17.09	13.61
Dataset 2	19.24	21.36
Dataset 3	12.99	17.99
Dataset 4	209.66	174.11

Table 4.14: Summary of the LSTM results with 1 and 3 inputs, for forward-days = 100.

5
Conclusions

This study proposes applying the random forest algorithm and the LSTM neural network for oil production forecasting. The strategy used was similar to the one presented in the time series forecasting, where the whole dataset is considered time-dependent, and the training dataset is split into small pieces of the entire data. The size of these series pieces is referred to as look-back. The chosen techniques were used to predict one and multiple tim steps ahead, and we refer to the amount of time steops we were about to predict as forward-days.

The target data is the oil production related to that series piece at the immediate forward-days time steps. We use 3 different sythtetic datasets and 1 real dataset to evaluate the robustness and the efficiency of the method in predicting high nonlinear physical dynamic data. Moreover, we test the method's performance by applying different values for the look-back and forward-days parameteres. We also experiment with some hyperparameters of the theoretical background of the random forest algorithms, such as the forest's number of trees and the bootstrap sampling ratio, to proceed with the bagging technique. We use the root mean squared error (RMSE), a classical error measuring practice, to assess the results.

The results obtained by this study suggest that the random forest algorithm could get accurate results when predicting oil field performance with an acceptable RMSE for forward-days $= 1$*. The bootstrap sampling ratio of 50% showed to be enough to acquire good final results. Considering the number of trees in the forest, we tested 100, 500, and 1000. However, the result was inconclusive because some experiments found the best fit for using 100 trees and others using 1000 trees. Therefore, we leave this hyperparameter open for further research.*

The outcomes for higher values of forward-days suggest that the LSTM neural network is more adequated to predict time series data. For Datasets 1, 2 and 3, the results for forward-days $= \{10, 50, 100\}$ *had acceptable RMSE values. We understand that these datasets, due to its periodicity behavior, the network could capture information with reasonable quality. For Dataset 4, due to its greater complexity, the network could only produce admissible predictions when the inputs included the pressure of the days we were about to predict.*

Even so, the outcomes were not accurate, they only reproduced the order of magnitude in Dataset.

For future works, we plan to continue this work by implementing more features to the LSTM neural network. Also, we plan to implement other machine learning techniques to make more accurate predictions considering additional information from a reservoir as inputs.

Bibliography

[1] PAIK, K.. **Energy cooperation in sino-russian relations: The importance of oil and gas**. The Pacific Review, 9(1):77–95, 1996.

[2] D. S. OLIVER, A. C. REYNOLDS, N. L.. **Inverse Theory for Petroleum Reservoir Characterization and History Matching**. Cambridge University Press.

[3] J. NOCEDAL, S. J. W.. **Numerical Optimization**. Springer New York.

[4] SILVA, T. M.; PESCO, S. ; BARRETO, A.. **Influences of the inflation factors generation in the main parameters of the ensemble smoother with multiple data assimilation**. Journal of Petroleum Science and Engineering, 203:108648, 2021.

[5] M. SHIRANGI, A. E.. **An improved tsvd-based levenberg-marquardt algorithm for history matching and comparison with gauss-newton**. Journal of Petroleum Science and Engineering, 143:258–271, 02 2016.

[6] SILVA, T. M.; BELA, R. V.; PESCO, S. ; BARRETO, A.. **Es-mda applied to estimate skin zone properties from injectivity tests data in multilayer reservoirs**. Computers Geosciences, 146:104635, 2021.

[7] TUNNICLIFFE WILSON, G.. **Time series analysis: Forecasting and control,5th edition, by george e. p. box, gwilym m. jenkins, gregory c. reinsel and greta m. ljung, 2015. published by john wiley and sons inc., hoboken, new jersey, pp. 712. isbn: 978-1-118-67502-1**. Journal of Time Series Analysis, 37:n/a–n/a, 03 2016.

[8] CHATFIELD, C.. **The Analysis of Time Series: An Introduction, Sixth Edition**. Chapman & Hall/CRC Texts in Statistical Science. CRC Press, 2016.

[9] L. A. N. COSTA, C. MASCHIO, D. J.. **Application of artificial neural networks in a history matching process**. Journal of Petroleum Science and Engineering, 123:30–45, 2014.

[10] **Strategic Well Test Planning Using Random Forest**, volumen All Days de **SPE Intelligent Energy International Conference and Exhibition**, 2014.

[11] M. RAHIMI, M. A. R.. **Reservoir facies classification based on random forest and geostatistics methods in an offshore oilfield**. Journal of Applied Geophysics, 201:104640, 2022.

[12] P. ZAHEDI, S. PARVANDEH, A. A. B. S. M. S. A. S. B. A. M.. **Random forest regression prediction of solid particle erosion in elbows**. Powder Technology, 338:983–992, 2018.

[13] G. TIAN, Y. YUAN, Y. L.. **Audio2face: Generating speech/face animation from single audio with attention-based bidirectional lstm networks**. In: 2019 IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA EXPO WORKSHOPS (ICMEW), p. 366–371.

[14] P. M. HANUNGGUL, S. S.. **The impact of local attention in lstm for abstractive text summarization**. In: 2019 INTERNATIONAL SEMINAR ON RESEARCH OF INFORMATION TECHNOLOGY AND INTELLIGENT SYSTEMS (ISRITI), p. 54–57, 2019.

[15] J. WANG, X. WANG, J. C.. **Jazz music generation based on grammar and lstm**. In: 2019 11TH INTERNATIONAL CONFERENCE ON INTELLIGENT HUMAN-MACHINE SYSTEMS AND CYBERNETICS (IHMSC), volumen 1, p. 115–120, 2019.

[16] V. K. R. CHIMMULA, L. Z.. **Time series forecasting of covid-19 transmission in canada using lstm networks**. Chaos, Solitons Fractals, 135:109864, 2020.

[17] Y. WANG, X. DU, Z. L. Q. D. J. W.. **Improved lstm-based time-series anomaly detection in rail transit operation environments**. IEEE Transactions on Industrial Informatics, 18(12):9027–9036, 2022.

[18] J. W. SHAVLIK, J. W.; DEITTERICH, T. E. ; DIETTERICH, T.. **Readings in Machine Learning**. Morgan Kaufmann Publishers Inc., 1st edition, 1991.

[19] SAMUEL, A. L.. **Some studies in machine learning using the game of checkers**. IBM Journal of Research and Development, 3(3):210–229, 1959.

[20] ROSENBLATT, F.. **The perceptron: A probabilistic model for information storage and organization in the brain**. Psychological Review, 65(6):386–408, 1958.

[21] MINSKY, M.; PAPERT, S.. **Perceptrons; an Introduction to Computational Geometry**. MIT Press, 1969.

[22] I. GOODFELLOW, Y. BENGIO, A. C. A.. **Deep Learning**. Adaptive Computation and Machine Learning series. MIT Press, 2016.

[23] D. E. RUMELHART, G. E. HINTON, J. R. W.. **Learning representations by back-propagating errors**. Nature, 323:533—-536, 1986.

[24] BREIMAN, L.. **Random forests**. Machine Learning, 45:5–32, 10 2001.

[25] S. HOCHREITER, J. S.. **Long short-term memory**. Neural computation, 9:1735–80, 12 1997.

[26] J. A. HARTIGAN, M. A. W.. **Algorithm as 136: A k-means clustering algorithm**. Journal of the Royal Statistical Society. Series C (Applied Statistics), 28(1):100–108, 1979.

[27] JOLLIFFE, I.. **Principal Component Analysis**. Springer Series in Statistics. Springer, 2002.

[28] T. POURHABIBI, K. ONG, B. H. K. Y. L. B.. **Fraud detection: A systematic literature review of graph-based anomaly detection approaches**. Decision Support Systems, 133:113303, 2020.

[29] TEN, C.-W.; HONG, J. ; LIU, C.-C.. **Anomaly detection for cybersecurity of the substations**. IEEE Transactions on Smart Grid, 2(4):865–873, 2011.

[30] K. POOJA, S. R.. **Anomaly detection for predictive maintenance in industry 4.0- a survey**. E3S Web Conf., 170:02007, 2020.

[31] R. S. SUTTON, A. G. B.. **Reinforcement Learning, second edition: An Introduction**. Adaptive Computation and Machine Learning series. MIT Press, 2018.

[32] LI, Y.. **Reinforcement learning applications**. CoRR, abs/1908.06973, 2019.

[33] D. T. LAROSE, C. D. L.. **Decision Trees**, p. 165–186. 2014.

[34] T. G. DIETTERICH, E. B. K.. **Machine learning bias, statistical bias, and statistical variance of decision tree**. 1995.

[35] HAWKINS, D. M.. **The problem of overfitting**. Journal of Chemical Information and Computer Sciences, 44(1):1–12, 2004.

[36] BREIMAN, L.. **Bagging predictors**. Machine Learning, 24(2):123–140, 1996.

[37] HAYKIN, S.. **Neural networks: A comprehensive foundation**. The Knowledge Engineering Review, 13(4) edition, 1994.

[38] HOCHREITERP, S.. **The vanishing gradient problem during learning recurrent neural nets and problem solutions**. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 06(02):107–116, 1998.

[39] C. LU, H. JIANG, J. Y. Z. W. M. Z. J. L.. **Shale oil production prediction and fracturing optimization based on machine learning**. Journal of Petroleum Science and Engineering, 217:110900, 2022.

[40] S. ELMABROUK, E. SHIRIF, R. M.. **Artificial neural network modeling for the prediction of oil production**. Petroleum Science and Technology, 32(9):1123–1130, 2014.

[41] Y. CHENG, Y. Y.. **Prediction of oil well production based on the time series model of optimized recursive neural network**. Petroleum Science and Technology, 39(9-10):303–312, 2021.

[42] EQUINOR. **Volve dataset**. `https://www.equinor.com/en/news/14jun2018-disclosing-volve-data.html`, 2008. [Online; accessed 13-April-2022].

[43] A. T. TUNKIEL, T. WIKTORSKI, D. S.. **Drilling dataset exploration, processing and interpretation using volve field data**. In: ASME 2020 39TH INTERNATIONAL CONFERENCE ON OCEAN, OFFSHORE AND ARCTIC ENGINEERING, 2020.

[44] I. GUPTA, D. DEVEGOWDA, N. T. V. J.. **Looking ahead of the bit using surface drilling and petrophysical data: Machine-learning-based real-time geosteering in volve field**. SPE Journal, 25(2):386–408, 2020.

[45] **A Novel 3D Mechanical Earth Modeling of the Volve Field and Its Application to Fault Stability Analysis**, volumen All Days de **U.S. Rock Mechanics/Geomechanics Symposium**, 06 2021.