KHAZAR UNIVERSITY

School: Graduate School of Science, Art and Technology

Department: Petroleum Engineering

Speciality: Development of Oil and Gas Fields

MASTER THESIS

Topic: Long-term Oil Production Forecasting: A Comparative Analysis of Reservoir Simulation Models and Machine Learning Approaches

STUDENT: Arif Rustamov Rashad
SUPERVISOR: Assoc. Prof. Dr. Grigorii Penkov

BAKU – 2024

XƏZƏR UNİVERSİTETİ

Fakültə: Təbiət elmləri, Sənət və Texnologiya yüksək təhsil

Departament: Neft Mühəndisliyi

İxtisas: Neft və Qaz Yataqlarının İşlənməsi

MAGİSTR TEZİSİ

Mövzu: Uzunmüddətli neft hasilatının proqnozlaşdırılması: lay simulyasiya modellərinin və maşın öyrənmə yanaşmalarının müqayisəli təhlili

MAGİSTRANT: Arif Rüstəmov Rəşad oğlu
ELMİ RƏHBƏR: dosent. Dr. Grigorii Penkov

BAKI – 2024

TABLE OF CONTENTS

INTRODUCTION

Relevance of the research - Oil production forecasting plays an important role in energy planning and decision-making for the petroleum industry. By making accurate future predictions about the availability of oil, it is possible to come up with the best production infrastructure investments including pipelines, storage facilities, refineries, and distribution networks. Long-term cost-effective field development strategies can be formulated based on predictions to maximize the recovery factor and value of an oil field. On a governmental scale, economists and policymakers can make decisions about fiscal planning, monetary policy, and energy investments to ensure economic stability. This can be achieved by observing future trends in supply and demand dynamics, which would enable countries to diversify their energy sectors accordingly.

Considering the decisive importance of oil production forecasting for companies, various methodologies and techniques are devised to give predictions based on available historical data. Decline curve analysis (DCA) is one of the widely used methods based on observation that oil production over the field lifetime follows a decline pattern. A more sophisticated reservoir simulation methodology incorporating geological and engineering data can also be used to produce future production performance of an oil field using simulators. Additionally, analogous fields with similar geological characteristics and operational history can be adopted for similar purposes.

These methodologies are different from each other in terms of input requirements, complexity, and interpretability. Unlike other methods, the numerical simulation model requires extensive amounts of data, including reservoir characteristics, well data and fluid properties. Because of the physics-based nature of reservoir simulation models, in-depth insights into reservoir behavior can be obtained allowing engineers to understand complex flow patterns and analyze physical mechanisms controlling production.

A certain degree of interpretability in DCA models can also be expected as various decline curve equations are fitted into the historical production data which enables the users to assess the decline behavior. Minimal data requirements in these models provide quick insights about future production behavior and offer baseline comparison with more sophisticated modelling techniques.

The purpose of the research - The primary purpose of this research is to propose a univariate long short-term memory (LSTM) model capable of providing both single-step and multi-step production forecasts for a candidate well located in Western Siberia. A key aspect of this study is the comparative analysis between machine learning algorithms, specifically LSTM, and reservoir simulation models, aiming to identify the optimal approach for future production prediction. The

research will evaluate these models based on various criteria including data requirement, complexity, accuracy, and time sensitivity, thus providing valuable insights into their effectiveness across different contextual scenarios. Furthermore, the study will involve the development and fine-tuning of machine learning algorithms, alongside the implementation of a reservoir simulation model. By employing visualization tools and metrics, the outcomes of these models will be thoroughly examined and compared, leading to informed recommendations for selecting the most suitable approach for production prediction in Western Siberia.

Research questions - These questions will be thoroughly examined in the current study and versatile tools will be used to answer these questions and bring clarity to the results.

- How does the LSTM model's accuracy compare to traditional reservoir simulation models for production forecasting in Western Siberian wells?
- What are the differing data requirements between LSTM and reservoir simulation models for accurate production prediction in Western Siberia?
- How does the complexity of the LSTM model compare to reservoir simulation models, and how does it affect production forecasting accuracy?
- How does the time sensitivity of the LSTM model compare to reservoir simulation models for real-time production forecasting in the oil and gas industry?
- How does the LSTM model perform across various contextual scenarios in Western Siberia, and what implications does this have for selecting the optimal forecasting approach?

Aim of the research - The research aims to develop a univariate long short-term memory (LSTM) model to generate both single-step and multi-step production forecasts for a well in Western Siberia. The study will compare machine learning algorithms, particularly LSTM, with traditional reservoir simulation models to determine the most effective method for predicting future production. This comparison will focus on criteria such as data requirements, complexity, accuracy, and time sensitivity. Additionally, the research involves the development and optimization of machine learning algorithms and the implementation of a reservoir simulation model. By using visualization tools and various metrics, the study will evaluate and compare the outcomes of these models, ultimately providing recommendations for the best approach to production prediction in Western Siberia.

CHAPTER 1. LITERATURE REVIEW

Conventional feed forward neural networks were used for modeling oil, gas, and water flowrates (Achong, 1961). Researchers used data collected from the reservoir located in Malay basin which was operated under waterflooding scheme to estimate the performance of the model. To establish a solid relationship between producer and injector well parameters, it was decided to pick up some random and physical combinations of input terms (e.g., water injection rate, water injection manifold pressure, water injection pressure, production manifold pressure, casing pressure, gas lift rate, tubing head pressure and heat temperature). Results showed that the feature extraction in this way greatly improves the performance of the model and gives the smallest mean square error (MSE) and the highest coefficient of determination when the model is trained by Bayesian regularization.

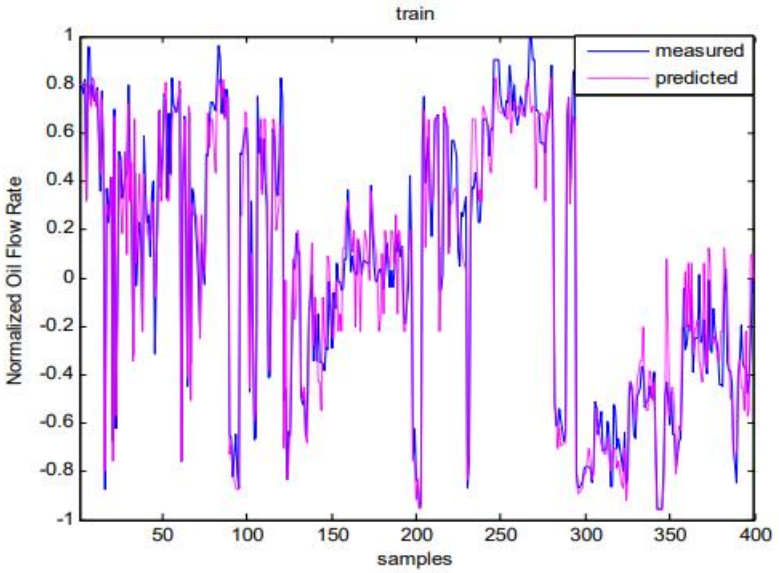

Figure 1.1. Comparison between measured and predicted oil flow rate, (ICA-ANN) Training

An imperialist competitive algorithm (ICA) was offered to optimize initial weights of feed forward neural network and to predict oil flowrate of wells in one of the northern Persian Gulf oil fields of Iran (Yadav et al, 2020). Multiphase Flow Meter (MFM) is a device conventionally used to measure separate flowrates of oil, gas, and water in co-mingled flow during oil production processes. Authors proposed the usage of ICA-ANN model as a cheaper and quicker alternative of MFM technology to mitigate problems linked to phase separation issues, flow regime changes, scaling and deposits and sensitivity to fluid properties (Baxendell, 1957) in this device. ICA-ANN model with two input parameters (temperature and pressure) gave the most accurate flowrate

predictions with MSE of 0.0123 and efficiency coefficient R^2 of 0.9703. In a similar study (Negash & Yaw, 2020) upstream pressure, choke size and producing gas to oil ratio (GOR) were taken as input parameters for artificial neural network (ANN) and the predictions were later compared with well-known empirical correlations (Bergstra & Bengio, 2012) for two-phase fluid flow prediction through wellhead choke. Dataset taken from 62 wells from 15 producing reservoirs in Southwest of Iran, Northern Persian Gulf showed the highest accuracy for the constructed ANN model.

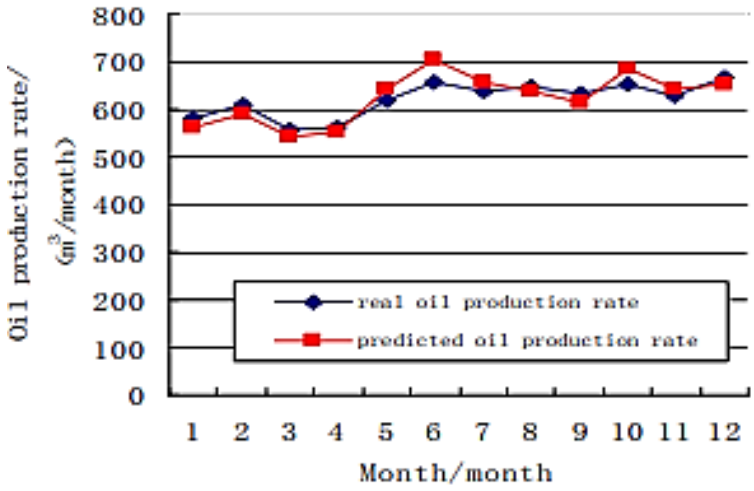

Figure 1.2. Real oil production rate and predicted oil production rate

Back propagation (BP) neural network was used in another study together with available log and production history data to give field-wise oil and water flowrate predictions. Input dataset was initially divided into three portions (static data, dynamic data, and spatiotemporal dependencies) to account for the various aspects of the fluid flow inside the reservoir and up the tubing. Voronoi diagram was devised for the entire field to give field-wise understanding of spatiotemporal dependencies between wells. The result showed that predicted error for oil flowrate is below 7%, while for water the figure is within 5%. LSTM model was constructed and trained to give petroleum production prediction in one of the fields of China with 5 production wells and 4 water injection wells (Gers et al, 2000). Input feature selection was performed using Mean Decrease Impurity (MDI) algorithm to estimate individual contribution of each variable to the accuracy of the overall model. According to the analysis remaining recoverable reserves and wellhead pressure had the greatest impact on the model, while parameters such as original water saturation and displacement of pump had marginal, almost no effect on the predictions. The

predicated root mean square error (RMSE) and mean absolute percentage error (MAPE) of LSTM model showed small with values of 0.985, 0.035, respectively.

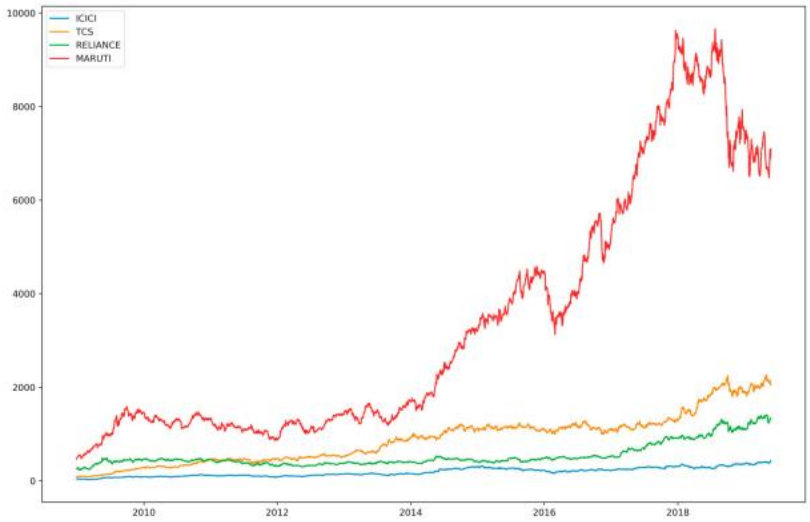

Figure 1.3. Combined plot of input data for four companies

In one of the studies (Mirzaei-Paiaman, 2008) LSTM's prediction abilities were used to compare predictions for 4 companies, namely, ICICI, TCS, Reliance, and Maruti in India. In this context, LSTM was used to predict stock prices in stateful and stateless forms, and the results obtained showed that P-values estimated for this goal were statistically insignificant. Discrepancies in the final result can be attributed to the fact that random seeding takes place when LSTM model is run for all small variations in the output. Upon analyzing box and whisker plots devised, it can apparently be seen those discrepancies or, stated differently, spread is more for stateless condition when compared to stateful case. It is a clear indication of the fact that when comparing both cases for LSTM model stateless alternative is more powerful and stable when contrasted to stateful condition. As a result of this research, it can be concluded that for time series prediction problems, a stateless LSTM model can very well be employed to make predictions with higher accuracy. However, in this case when one uses LSTM for the same series problems but with the successive chunk of words, it makes more sense to make a use of stateful LSTM model. In experiment number 1 carried out by the authors, it was validated that when the number of hidden layers is taken as 1, it is more likely to be the best configuration for the LSTM model. The result was retaken by means of a one-way ANOVA test. The final recommendation of the current research was to go with a lower number of hidden layers for much better accuracy in the prediction, and far less training time, and most importantly, avoidance of overfitting. In the circumstances, when the problem is

sophisticated and more means are required to correctly capture the trends and relationships in the data, one may decide to make a use of more than one hidden layer, but the practice shows that these cases are quite rare. The only merit of multiple number of hidden layers is that LSTM becomes increasingly more stable due to decreasing standard deviations according to box and whisker plots.

Another study (Falcone & Alimonti, 2007) compares the effectiveness of multi-layer perceptron (MLP) and pure autoregressive model (AR) on time series data which is solvable by considering only the input data. Stated differently, LSTM's strength in prediction past events and in capturing patterns in the historical data was not useful in this case. Looking at the results of this study, it was revealed that LSTM can very well tune oscillations in each series but is not capable of capturing the trend of the signal. In contrast, multi-layer perceptron managed to capture chaotic behavior of the data very well.

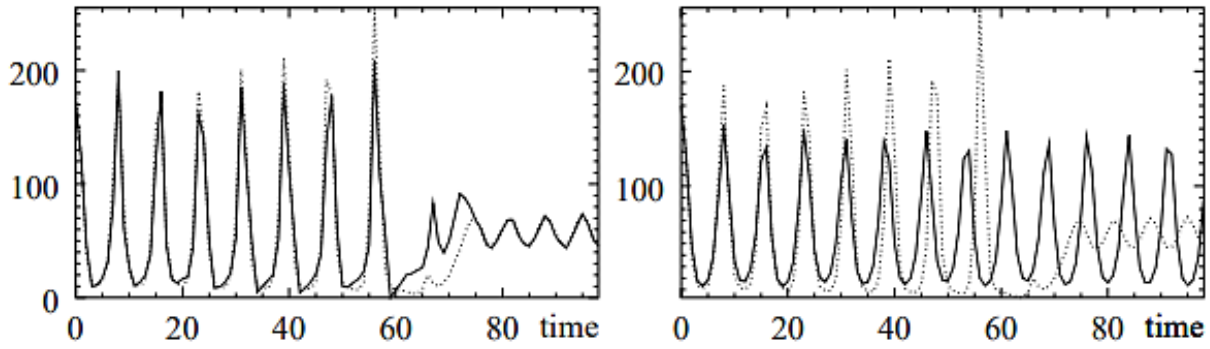

Figure 1.4. Test run with LSTM network solution after iterated training for the FIR-laser

This could be observed by considering the fact that in the FIR laser-task, the multi-layer perceptron model very well predicted collapse of emissions. In the context of multi-layer perceptron models, iterated trainings have big advantage over single-step training, and more importantly, when the univariate prediction is considered with a step size of one. When the number of necessary iterations gets bigger the clear advantage of multi-layer perceptron models can easily be seen (Cerqueira et al, 2020). According to the results of the research, long-short memory models are recommended to be used only in the case when traditionally used time window technique fails to give correct results. Though, to avoid this problem, the hybrid model selection can be selected in the case of unknown time-series. In this case, one should start by taking window based MLP for training, freeze its weights for some time and make a use of LSTM to minimize residual errors in case any emerges. Long-short term memory's ability to cope with long time lags between insignificant events comes very hand in this case and proves itself to be a reliable technique.

A new method by (Little et al., 2017) for training long-term memory was devised. The main idea here is to predict results of high uncertainty in events that may occur to be arbitrarily far in the future in the unknown time. Researchers substantiated this concept by illustrating that high uncertainty is present when predicting outcomes using local observations indicate a lack of local information which is unfavorable. From the experiments conducted by the authors, it can be observed that the proposed method, in combination with the architecture of a recurrent neural network (RNN), is capable of learning temporal dependencies within the time series data, even up to 500 times extended than the length to which we unroll the method of backpropagation through time (Truncated BPTT). This property is not owned by either recurrent networks (RNN) trained with the classical TBPTT algorithm, as evidenced by experiments in the Noisy T-maze, or architectures based on transformers due to the absence of mechanisms for transmitting information not included in the attention window of the transformer. Moreover, as it can very well be observed from experimentation on ViZDoom-two-colors, even if alternative architectures are able to learn long-term temporal dependencies, they do so using significantly more resources and at a slower pace than the proposed MemUP algorithm.

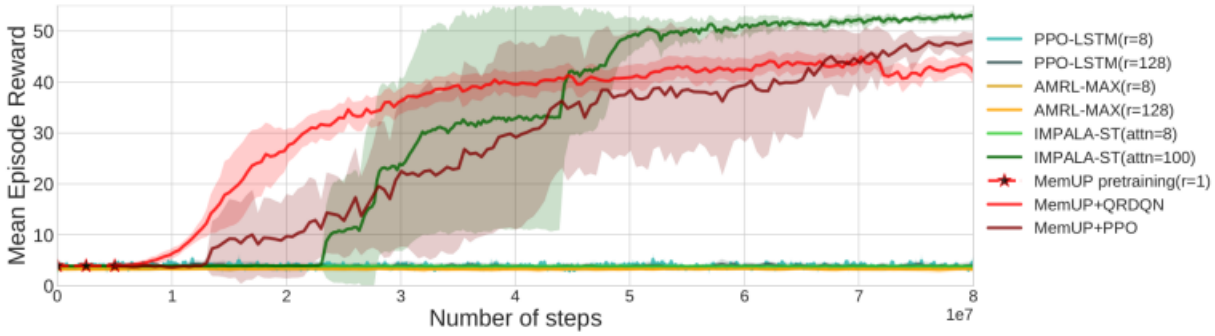

Figure 1.5. Experimentation results with ViZDoom-Two-Colors

It is worth noting that the specific architectural decisions in our implementation of the MemUP algorithm were chosen for the sake of the implementation's speed to test the feasibility of our idea. For example, one of the drawbacks of the proposed MemUp implementation is the separate training of memory and agent. This may be a problem for online RL tasks, where increasing the agent's strategy may lead it to new forms with new temporal dependencies that the trained memory on another idea cannot remember. On the other hand, this challenge will not appear in other kinds of problems, such as supervised learning or imitation learning/offline reinforcement studying (Mesafint Belete & Huchaiah, 2022). Even considering this drawback of our realization,

it is considered that the proposed method has shown promising results, leaving many opportunities for future research directions. Solving the problem of separate memory training for RL tasks and combining transformer-based memory architecture with our proposed learning method.

One of the most important research projects in the direction of application of Long-short term memory in time series analysis was carried out (Ripley, 1996). The primary purpose of the study was the application of recurrent neural networks with the primary aim of employing them to be used for a very long sequence of data. Different datasets were used in this study including but not limited to natural language text, speech , with the goal of finding intricate patterns in the sequential data. The recurrent neural networks were also compared to long-short term memory algorithm in this study. It was found that LSTM networks were likely to perform much better when compared to traditional recurrent neural networks specifically in tasks requiring analysis of long sequence of dataset, such as speech recognition and language processing. Long-short term memory algorithms employ internal gate mechanisms to effectively learn and keep the information for an extended period of time making them extremely suitable for the dataset with successive nature. The study by the authors provided a proof that long-short term memory algorithm is very useful for time series data and it was highlighted that application area of long-short term memory can be quite wide in different field including but not limited to natural language processing also in the financial forecasting.

Another research in the application of long-short term memory was carried out (Da Silva et al, 2007). The primary goal of the study was to check applications of long-short term memory mechanism in diagnosis of pediatric diseases using the signal received by electronic health records of the patient used in medical conditions. Input data from electronic health records of the patients were input into designed long-short term memory algorithm which helped to find out dependencies present in patients' health records, symptoms and helped with finding out suitable treatments for specific cases. The results of the study proved that long-short term memory can very well capture intricate relationships in the data and understand dynamic patters of the electronic health records of the patients. Application of long-short term memory in the field of medicine greatly helped health professionals to make important decisions regarding the patient's health and make accurate diagnoses. The research proved crucial application of long-short term memory in the field of medicine, specifically, in handling sequential data similar to electronic health records type data and proved application of advanced machine learning techniques in medical diagnostics.

One of the important researches in the direction of application of advanced long-short term memory algorithms was conducted (Baxendell, 1957). The research mainly focused on the application of long-short term memory algorithm to make predictions about the stock prices based on the historical data. The research was unique and important one in the direction of application of long-short term memory. The main reason for it is that historical stock prices show extremely fluctuating trend meaning that not all available algorithms can deal with this task easily. Specific selection of the algorithm for this purpose is extremely important as traditional algorithms including linear regression, support vector machine, and more importantly neural networks cannot easily handle this type of data. It is important to make a use of algorithm suitable for handling time series data, stated differently, handling sequential data. Even though, majority of the model involve advanced algorithm capable of handling intricate relationships in the data, if no relationship exists in the data they cannot be used to make logical decisions about the data. The current study proved long-short term memory application in successive dataset, especially with the one containing lots of residuals to be extremely useful. By applying long-short term memory in the field stock market, traders and investors can easily make informed decisions about the volatile nature of the market and attach certain degree of uncertainty to the predictions.

In another study by (Radzi et al., 2021) the long-short term memory machine learning algorithm was successfully applied to investigate the importance of this algorithm to enhance efficiency and reliability of maintenance practices in oil and gas industry. By using historical information from variety of sensors utilized in the practice, including the temperature, pressure, and flowrate measurements the long-short term memory machine learning algorithm was trained to find possible failure possibility in the future and requirement for the maintenance work in advance. This research showed the effectiveness of long-short term memory machine learning algorithm in analysis of time series data and ability of it to successfully capture the intricate patterns within the dataset. The successful implementation of this model would allow the operators to effectively make predictions about the maintenance time and avoid the operator company from unnecessary expenses. They managed to schedule accurate maintenance activities, minimize the downtime, and effectively optimize the production time, ultimately leading to a more effective resource management practices within the company.

In a study by (Falessi et al., 2020) the application of long-short term memory machine learning algorithm was analyzed to understand the effectiveness of the algorithm in the optimization of the reservoir management strategies. The study employed historical production

data also geological features in the training of the model to correctly apply the algorithm in the achieving important strategic objectives. According to the results of the model, it was concluded that long-short term memory machine learning algorithms' ability to capture the intricate patterns within big datasets can very well be used to analyze big datasets and achieve quite satisfactory results when compared to traditional methods. Using this technique enabled the operators to correctly make decisions about the well intervention times, schedule the production, and make informed decisions about the reservoir development planning, thus, leading to improved reservoir management practices.

As it can be seen from the literature review, application of long-short term memory machine learning algorithm for production prediction in oil and gas industry offers distinct advantages over traditional algorithms (Muhammad Ali & Faraj, 2014). Unlike simple and traditional algorithms such as linear regression, support vector machine, random forest and decision trees, the long-short term memory machine learning algorithms can handle big datasets in the form of time series and having the sequential data. Other algorithms mentioned previously as the literature review suggests are more suitable for the datasets having more independent nature. Numerous studies have shown that the long-short term memory machine learning algorithm have superior performance in understanding temporal patterns and non-linear relationships specific to the production data. Availability of the internal gate mechanisms in long-short term memory machine learning algorithms enable them to effectively analyze the data in the successive order and make informed decisions using the patterns hidden in the data. Forget gate allows to effectively get rid of the part of the data unnecessary in the context of the data. It is also useful as the time datasets contain some degree of randomness due to several factors. In the case of oil production measurements, since the data contains the human factor the production rate is not directly related to the pressure response from the reservoir but also is influenced by the decisions of the operators. Thus, it is important to verify is the application of long-short term memory machine learning algorithm is suitable for the dataset to be considered in this research is suitable or not. Although many researchers making studies in the application of long-short term machine learning algorithm validate the application of this algorithm for oil production data, still there is need to use alternative methods to justify the use of this specific algorithm with a given dataset. According to the literature review, time series decomposition is likely to decompose the data in the time series format into its corresponding components and analyze the underlying logic behind each trend. The current study will focus on this aspect of the research and try to come up with the solid proof of suitability of application of

long-short term memory machine learning algorithm in the context of oil production data and its usefulness for long-term predictions.

In terms of reservoir simulation models, the techniques are well-established. Wide variety of tools are available to effectively employ different reservoir simulation software. Some of this software include eclipse, tNavigator, NEXUS and many others (Suradhaniwar et al., 2021). In the current research, the reservoir simulation was carried out using tNavigator which is widely applicable tool. There have many studies carried out using this specific tool, and the implementation is well-documented.

CHAPTER 2. METHODOLOGY

The main goal of the study is to make comparison between prediction abilities of reservoir simulation models by using a traditional computer software, so called tNavigator, and see predictive abilities of long-short term memory machine learning algorithm in the production prediction of oil. The research methodology section starts with the preliminary analysis of the dataset. It is important to carry out to establish relationships within the dataset and understand if the sample data considered is suitable for prediction purposes. For this reason, in the first-place lag plot analysis was carried out to see the degree of relationship between various data points in the dataset. The current research considered current data point with the one right after it, half a year and a whole year. Once the relationship is established by means of a Pearson correlation coefficient, the dataset can be said to be suitable for prediction purpose, thus, long-short term memory can very well be applied.

Another important point to consider is that the production data is not a direct result of physical parameters of the reservoir, but it is controlled by the choke on the surface. Stated differently, depending on contractual terms, technical issues or for some other reasons the choke size can be changed by operators, directly affecting the flowrate of the well. Although apparently the pressure response of the system is also going to have a significant effect on the flowrate output of the well, modifications to the choke system can affect the nature of the dataset obtained, thereby, add some degree of randomness into the data. A very important question arises in this case about the application of long-short term memory with a given data set. Since long-short term memory (LSTM) is likely to find patterns and trends in the data containing some degree of logic, it is questionable if the model devised is going to handle the dataset or not. This question can be addressed in two different ways; either to find out a parameter which can be used to find degree of randomness in the data set, or to prove that data is specific and suitable for long-short term memory machine learning model to be applied. It is important to establish beforehand, because otherwise no model can capture the trend and understand the relationship within the data full of random points.

2.1. Setting up an Environment for Experimentation

In recent years it has become increasingly popular to use a variety of programming languages for machine learning computations. Traditional software cannot deal with the extensive computations provided in machine learning algorithms. Also, programming languages offer variety

of built-in functions that are optimized in terms of run-time allowing them to accurately and in a short time span to implement given computation and achieve the optimal result.

For application of long-short term machine learning model in the study Python programming language was selected. It offers flexibility in different directions. Firstly, it is faster when compared to other programming languages used worldwide. Secondly, Python offers user friendly interface and functions making it easier for the users to adapt and apply in their projects.

Multiple libraries are available in python allowing the users to automatically realize the given tasks without typing lengthy algorithms from scratch. For pre-processing the main libraries used include numpy and pandas. These built-in libraries contain dozens of functions that can be called by the user to implement various operations on matrices and vectors. Since the dataset considered in this research was directly related to vector operations, the script initially started with the call-in of these libraries. Pandas library, on the other hand, contains unique functions that can be used to implement various operations and transformations on the datasets in the form of table. Initially, by calling read_csv function from pandas library, the dataset in .csv format was opened and written into the brain of the system. Once opened, there are multiple functions available which can be employed to perform analysis on this table. The most common and the one applied in the study was .describe function which gives important statistical parameters related to the table in the columns-wise. Of course, the function can be altered by the user to give these features in the row-wise form also.

Once, all the data is written into the brain of the environment, the next step is to visualize the data and see intricate relationships between various variables. For this purpose, libraries including matplotlib and seaborn, the most common ones, can very well be used. Seaborn library itself is built open matplotlib library and offers unique designs for bar charts, curves, pie charts, box plots/whisker plots, and many more. In the research all the design was made by using matplotlib library.

To plot the curves related to oil and water production a famous matplotlib library function so called .plot was used. It is possible to give the size to the figure being plotted and modify the color. It is also possible to change the font, size, and location of the axis. In the figures provided in the next sections, grid plot design was selected for curve and bar graph representations. As it offers flexibility in readings, it is much easier to read the corresponding values form axis when it is given in grid format.

Several years ago, when machine learning first came to the application, it was hard to run the models since the user must write the whole algorithm from scratch. But over the time, different libraries were introduced to the public use that enabled many to run the algorithms without spending significant amount of time by concentrating on writing formulas and devising steps for the algorithm, these libraries include keras and tensorflow that are widely used by many developers around the world.

```
import pandas as pd;

import numpy as np;

import matplotlib.pyplot as plt;

import math;

from sklearn.preprocessing import StandardScaler, MinMaxScaler;

from keras.layers import LSTM, Dense, Bidirectional;

from keras.models import Sequential;

from keras.utils import plot_model, set_random_seed;

from scipy import stats;

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score, mean_squared_log_error;
```

These libraries allow them to access all the well-known algorithms with ease and to apply them to their unique datasets. However, a problem sometimes appears to be here. Since each dataset is unique in their own kind, sometimes it is required to make very detailed modifications in the algorithm to fully capture the underlying logic. Since all the function are built-in these libraries, it is not possible to modify them. The script and algorithms itself should be written from scratch to allow the user to make necessary modifications when it is needed. Nevertheless, the time-consuming nature of this process must be taken into account before deciding whether to do it or not.

In the pre-processing part of the data, firstly it is important to obtain lag plots. Pandas library offers unique and quite useful functions for this purpose. It also allows the user to specify the lag point intervals to be considered. pd.plotting.autocorrelation_plot function gives the autoregressive plot of the data without a need to write a lengthy code for this purpose.

```
# Autocorrelation graph
```

```
plt.figure(figsize=(20, 5))

pd.plotting.autocorrelation_plot(data['Production'])

plt.minorticks_on()

plt.grid(which='minor')
```

To obtain boxplot for the dataset to be considered, the matplotlib library contains a very useful function, so called .boxplot which takes one input variable and assigns the given axis. The result shows the boxplot of the dataset based on the given basis. The basis can be quite varying. Since monthly production data was used in the study, for the basis to be considered year was selected.

```
oil_production_data = data['Production'].values
# Split the data into sets of 12 months each
yearly_data_chunks = [oil_production_data[i:i+12] for i in range(0, len(oil_production_data), 12)]
fig,ax = plt.subplots(figsize=(20, 5))
ax.boxplot(yearly_data_chunks, labels=np.arange(2003, 2021),
      flierprops=dict(marker='o', markerfacecolor='red'), widths=0.6, notch=True,
      patch_artist=True, boxprops=dict(facecolor='white'), medianprops = dict(color="black"))
ax.set_ylabel('Oil production, t')
plt.minorticks_on()
plt.grid(which='major')
plt.grid(which='minor')
```

The algorithm starts by specifying the location of the dataset considered as shown below. The path variable is created for this purpose. The dataset has to be located within the same folder with the script being written. Once the dataset is read using read_csv function introduced by pandas library, the data is searched for nan values.

```
path = r"Data.csv";

data = pd.read_csv(path);

data = data.fillna(0);

data = data[::-1].reset_index(drop=True);

data = data[data['Production'].argmax():].reset_index(drop=True) ;

data.head();
```

These are points in the dataset which are missing, either due to technical problems or absence of data points at specific times. Once all nan values and their corresponding indices are found, they are replaced by a value of zero using fillna operator. It is important to reset indices once all nan values are replaced or dropped. For this purpose, the pandas library has built-in function called reset_index. After cleaning data from missing values, the dataset is visualized using .head built-in function, which give the top five rows of the dataset to see the current view of the table. In the same way, the .tail function can be used by pandas library to showcase the last five rows of the dataset. Another step after parsing the data is to use .describe function which gives number of points, mean, maximum and minimum values, standard deviation, 1st and 3rd quartiles of the columns.

```
data.describe().round();
```

The next step in the script is to use sklearn library to recall built-in functions to either standardize or normalize the dataset. It offers flexible function that can be used to directly perform column-wise standardization over the columns of the dataset. In order to run this operation, in the first place so called StandardScaler and MinMaxScaler objects need to be created. Once created, they are called upon the dataset variable created previously to perform the operation.

```
nscaler = MinMaxScaler(feature_range=(0, 1));

sscaler = StandardScaler();

oil_prod = data['Production'].values;

norm_oil_prod = nscaler.fit_transform(oil_prod.reshape(-1, 1)); # normalized dataset

stand_oil_prod = sscaler.fit_transform(oil_prod.reshape(-1, 1)); # standardized dataset
```

After finishing these steps, it is important to create a function that will implement windowing procedure on the data set. For this purpose, the script can be written either in a traditional fashion or specific function can be created using def operator. In the current research, the def operator was used to create the windowing algorithm which adds another flexibility for the user.

```
def split_sequence(sequence, n_steps):

  X, y = list(), list()

  for i in range(len(sequence)):

    end_ix = i + n_steps
```

```
    if end_ix > len(sequence)-1:
        break;
    seq_x, seq_y = sequence[i:end_ix], sequence[end_ix];
    X.append(seq_x);
    y.append(seq_y);
return np.array(X), np.array(y);
```

Another important step in creating an environment for long-short term memory machine learning algorithms is to reshape the data and make it suitable for the model to consider. Initially, after the data is opened by means of a pandas library it is having a single dimensional shape which is a shape of a vector where the successive data points are considered. However, the long-short term machine learning model cannot input the data in the single dimensional form. Thus, it is important to change and reshape the dimension of the data before inputting it into the network. For this reason, the reshape function from numpy library comes very handy. Accordingly, the training dataset which consists of 179 points and the test dataset consisting of 36 points are reshaped.

```
n_features = 1;
trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]));
testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]));
```

The next step is to create a built-in function for bidirectional long-short term memory machine learning model. Before running the model, some variables were created to assign the features of the model and change them if required. The first variable is activation variable which was taken as ReLu. The role of activation function was mentioned previously in the work principles of the long-short term memory machine learning algorithm. The next variable was loss, which defined the loss function to be used for training the model. The loss function in the research was taken as mean squared error. During the training the weight and biases for matrices and vectors were changed to minimize the loss function. Number of units, epochs were other variables created beforehand. Batch size variable was also created to assign number of data points to be given to the network at a time. The batch size is usually taken as multiples of 2. If big batch size is selected, there will be considerable reduction in the run time of the model. In the current research batch size was taken as 1. Verbose variable is another useful variable which shows the training of the model

in the text form. If verbose function is made equal to a Boolean value of 1, the loss and the accuracy will be shown in each iteration. For models operating normally, the verbose must show steady declining trend for loss and increasing accuracy values.

```
# running bidirectional lstm model;

activation='relu';

loss='mean_squared_error';

optimizer='adam';

n_units = 128;

epochs = 100;

batch_size=1;

verbose=0;
```

The next step is to create a real long-short term model to run the dataset and achieve the result. For this purpose, the first step is to create a sequential model. Once the sequential model is created, bidirectional long-short term memory layers need to be added into the sequential model, and for each layer number of units, activation function must be assigned. Only for input layer it is important to assign the shape of the input data. Since the bidirectional model was used to make a univariate single-step prediction, the output layer has to output with the dimension of 1. The keras library requires this information to be mentioned in the output layer, so that the model can operate efficiently.

```
model_blstm = Sequential();

model_blstm.add(Bidirectional(LSTM(n_units, activation=activation, input_shape=(1, n_steps))));

model_blstm.add(Dense(1));

model_blstm.compile(loss=loss, optimizer=optimizer);

model_blstm.fit(trainX, trainY, epochs=epochs, batch_size=batch_size, verbose=verbose);

# prediction of training and testing dataset

pred_blstm_trainX = sscaler.inverse_transform(model_blstm.predict(trainX));

pred_blstm_testX = sscaler.inverse_transform(model_blstm.predict(testX));

# saving the model
```

```
model_blstm.save('lstm-models\model-blstm.h5');

plot_model(model_blstm, to_file='lstm-models\model-blstm.png');
```

Once the model is trained, and optimum results are obtained it is important to rescale the data back into the previous form. Since the object was created previously using sklearn library, the mean and standard deviation of the data was still contained in that object. Rescaling the data ensures that predicted values have the same scale as the real data itself.

```
pred_blstm_trainX = sscaler.inverse_transform(model_blstm.predict(trainX));

pred_blstm_testX = sscaler.inverse_transform(model_blstm.predict(testX));
```

Model was saved using .save function in h5 file format. The model can be recalled back anytime to use the same set of matrices and vector to make predictions about the dataset to be considered. It is also possible to plot the structure of the model using plot_model function, that gives the visual representation of the model of the long-short term memory machine learning model.


```
model_blstm.save('lstm-models\model-blstm.h5');

plot_model(model_blstm, to_file='lstm-models\model-blstm.png');
```


The next step is to create several metric variables to calculate the errors and see overall performance of the model. For this purpose, different metric functions are provided by sklearn library that make the whole procedure quite simple. The first error metric created was mean absolute error that outputs the MAE between the predicted values and the real data. This metric comes very handy as it allows to directly compare the error rate in the predictions but overall does not give any idea on how strong the model was with the predictions. This error outputs the two by two matrix with the diagonal elements being one. The second value shows the mean absolute error. It is important to recall the second values, since, otherwise, the matrix will be shown as an output. The second error variable created was root mean squared error. It could be obtained either by taking the root of mean absolute error or by recalling again the root mean squared error function from sklearn library. It is easier to use for dataset for which the scale of the data is big. As it finds the square of the error, the resulting errors are small and easier to interpret. The following error variable was mean absolute error which was again recalled from sklearn library. It gives the error in the prediction in terms of percentages. The smaller the output of this metric, the better the predictive accuracy of the model is. R2 score and logarithmic mean errors were also recalled from sklearn

library. R2 score provides the value between 0 and 1. The bigger the value is, or stated differently, closer to 1 means the higher accuracy. Logarithmic squared error, on the other hand, gives either positive or negative value. The negative value shows overestimation in the prediction which is extremely unfavorable in the context of oil production prediction as it might give a false sense of capabilities of the reservoir.

```
# Calculate MAE

mae_blstm = mean_absolute_error(data['Production'][split:], pred_blstm_testX.reshape(1, -1)[0]);

# Calculate RMSE

rmse_blstm = np.sqrt(mean_squared_error(data['Production'][split:], pred_blstm_testX.reshape(1, -1)[0]));

# Calculate MAPE

mape_blstm = np.mean(np.abs((np.array(data['Production'][split:]) - np.array(pred_blstm_testX.reshape(1, -1)[0])) / np.array(data['Production'][split:]))) * 100;

# Calculate R2

r2_blstm = r2_score(data['Production'][split:], pred_blstm_testX.reshape(1, -1)[0]);

# Calculate MSLE

msle_blstm = mean_squared_log_error(data['Production'][split:], pred_blstm_testX.reshape(1, -1)[0]);

# Print the metrics

print(f"MAE: {round(mae_blstm, 2)}");

print(f"RMSE: {round(rmse_blstm, 2)}");

print(f"MAPE: {round(mape_blstm, 2)}%");

print(f"R2: {round(r2_blstm, 2)}");

print(f"MSLE: {round(msle_blstm, 2)}");
```

The next step in the script was to plot the results. As mentioned previously matplotlib library was used for this purpose. For cumulative production curves, on the other hand, initially it was necessary to recall cumsum function from numpy library to cumulative sum all production points for each corresponding time frame and come up with the final production at the end of the period.

```
fig,ax = plt.subplots(figsize=(20, 5));

ax.plot(data['Data'], data['Production'], color="black");
```

```
ax.set_ylabel("Monthly oil production, t", fontsize=None);

plt.xticks(data['Data'][1::15]);

ax.plot(data['Data'][:split+1], np.append(pred_blstm_trainX, pred_blstm_testX[0]), color="blue");

ax.plot(data['Data'][split:], pred_blstm_testX, color="red");

ax.legend(['Real Data', 'Train Data Prediction', 'Test Data Prediction']);

ax.minorticks_on();

ax.grid(which='major');

ax.grid(which='minor');
```

For recursive production prediction, the set-up was different. Since predictions are made based on the last point predicted by a network, it was important to create a function that takes the last value from the predictions and make a new one based on the final point. To separate the recursive production model from the previous one a new function was defined. The for loop was created inside to iterate over and over and use the last predictions to make a new one. As the prediction were made for three years of period of time, the loop continued for 36 times in total. Once the loop finished, again rescaling of the prediction was carried out. The model was save and the corresponding structure was plotted for visual aid. The metric variables defined previously were used to make comparison and achieve the error rates.

```
# running recursive lstm model

activation='relu'

loss='mean_squared_error'

optimizer='adam'

n_units = 12

epochs = 35

batch_size=1

verbose=0

model_rlstm = Sequential()

model_rlstm.add(Bidirectional(LSTM(n_units, activation=activation, input_shape=(1, n_steps))))
```

```
model_rlstm.add(Dense(1))

model_rlstm.compile(loss=loss, optimizer=optimizer)

history = model_rlstm.fit(trainX, trainY, epochs=epochs, batch_size=batch_size, verbose=verbose)

test_data = stand_oil_prod[split-n_steps: split]

s = 0

ss = n_steps

for i in range(split, data.shape[0]):

  prediction = model_rlstm.predict(test_data[s:ss].reshape(1, 1, n_steps), verbose=0)

  test_data = np.append(test_data, prediction)

  s = s + 1

  ss = ss + 1

pred_rlstm_testX = sscaler.inverse_transform(test_data.reshape(-1, 1))[n_steps:]

# saving model

model_rlstm.save('lstm-models\model-rlstm.h5')

plot_model(model_rlstm, to_file='lstm-models\model-rlstm.png')
```

The steps for multi-step production prediction for long-short term memory model was carried out in a similar fashion. Unlike the previous cases, where the output layer was selected to give one point in this case it was assigned to be 36.

```
model_blstm = Sequential();

model_blstm.add(Bidirectional(LSTM(n_units, activation=activation, input_shape=(1, n_steps))));

model_blstm.add(Dense(36));

model_blstm.compile(loss=loss, optimizer=optimizer);

model_blstm.fit(trainX, trainY, epochs=epochs, batch_size=batch_size, verbose=verbose);

# prediction of training and testing dataset

pred_blstm_trainX = sscaler.inverse_transform(model_blstm.predict(trainX));

pred_blstm_testX = sscaler.inverse_transform(model_blstm.predict(testX));
```

```
# saving the model

model_blstm.save('lstm-models\model-blstm.h5');

plot_model(model_blstm, to_file='lstm-models\model-blstm.png');
```

Similar libraries were employed for the purpose of hyperparameter tuning. For hyperparameter tuning as mentioned previously grid search algorithms were used. For this purpose the hyperparameter space has to be defined for various hyperparameters including window size, number of units, and number of epochs. Param_grid variable was assigned for this purpose as follows:

```
# Define hyperparameters for tuning

param_grid = {

  'window_size': [3, 6, 12],

  'units': [32, 64, 128],

  'epochs': [25, 50, 100]}
```

The next step in the algorithm is to use multiple for loops to iterate over each parameter combination and try to search for the one with the minimal error rate. The iteration procedure can take quite a lot of time if the number of parameters to be tuned is a lot. In this case, there were three hyperparameters to be hyper tuned. The overall number of iteration is equal to multiplication of number variables o be tested in each hyperspace. In this case, since each hyperspace includes three variables, the total number of iterations is equal to 27. The computer on which the algorithm was run, spent over 12 hours to run the whole procedure and come up with the most accurate set of hyperparameters for the model and the predictions.

```
for window_size in param_grid['window_size']:

  for units in param_grid['units']:

    for epochs in param_grid['epochs']:

      # Create an instance of the LSTM model

      lstm_model = create_lstm_model(units=units, window_size=window_size)

      # Perform rolling cross-validation
```

```
rmse_scores = []
pred_values = []
c = 0
for i in range(n_train, len(monthly_production_data)):
  x, y = split_sequence(monthly_production_data, window_size)
  trainX, testX = x[:i], x[i-window_size:]
  trainY, testY = y[:i], y[i-window_size:]
  # reshape input to be [samples, time steps, features] for LSTM to run
  n_features = 1
  trainX = np.reshape(trainX, (trainX.shape[0], trainX.shape[1], n_features))
  testX = np.reshape(testX, (testX.shape[0], testX.shape[1], n_features))
  lstm_model.fit(trainX, trainY, epochs=epochs, batch_size=1, verbose=0)
  y_pred_cv = lstm_model.predict(testX[0].reshape(1, window_size, 1))  # predict only the first window
  y_pred_cv = scaler.inverse_transform(y_pred_cv)
  rmse_cv = np.abs(scaler.inverse_transform(testY[0].reshape(1, -1))[0][0] - y_pred_cv[0][0])
  rmse_scores.append(rmse_cv)
  pred_values.append(y_pred_cv[0][0])
  print(str(c) + ": " + str(window_size) + "," + str(units) + "," + str(epochs) + "," + str(rmse_cv))
   c = c + 1
# Calculate the average RMSE across all folds
avg_rmse = np.mean(rmse_scores)
err_dict[str(window_size) + "," + str(units) + "," + str(epochs)] = rmse_scores
pred_dict[str(window_size) + "," + str(units) + "," + str(epochs)] = pred_values
# Store RMSE values and corresponding hyperparameter combinations
rmse_values.append(avg_rmse)
```

```
    hyperparameter_combinations.append((window_size, units, epochs))
```

Results of these iterations were saved to the excel file in .xlsx format. In excel file, there was series of hyperparameter combinations given for each hyperparameter and their corresponding error rate provided there. To save this data as an excel file, to_excel function was used.

```
df1.to_excel('part-2-size-6.xlsx')

df2.to_excel('part-2-size-6-prediction.xlsx')
```

The excel file was then parsed through to find corresponding set of hyperparameters with the least possible error. These parameters were later used to train the model and make accurate predictions.

The script itself was written in functional format and easy to understand for users reviewing it. The variables are clearly explained. The use of built-in libraries enable the reader to understand the script easily and make correct inferences.

2.2. Exploratory Analysis

The data used in the study was taken from an oil well (61P) located in Western Siberia, Russia and consisted of 215 points in total. Figure 2.1. illustrates the oil and water production curves spanning the period from January 1, 2003, to November 1, 2020.

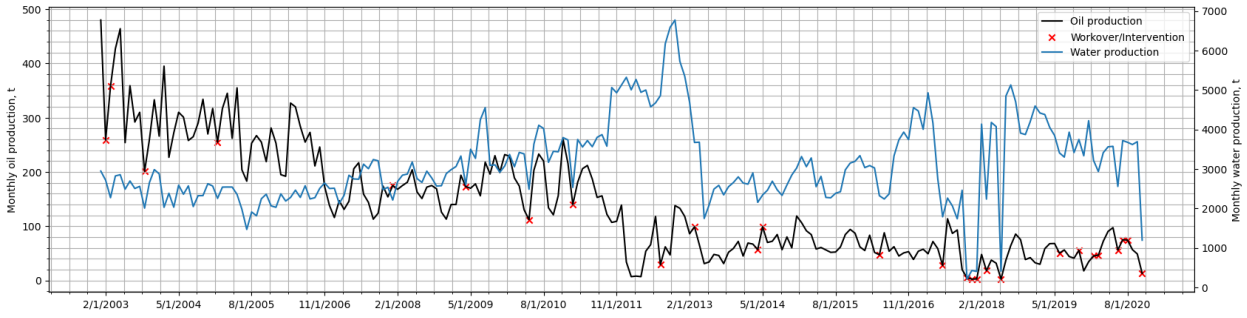

Figure 2.1. Historical production trends for the well 61P

The red scatter points on the production curve correspond to workover/intervention periods when the well was shut-in on purpose (reduced ESP insulation, ESP voltage disconnection, ramp-up, etc.). The shut-in periods ranged in-between 11-481 hours and as observed from the production trend, they usually coincide with the dips in the curve.

Table 2.1. Statistical summary of the historical production data

Mean	Median	Variance	Standard Deviation	Minimum	Maximum	Skewness	Kurtosis	25th Percentile	75th Percentile
139.27	116.0	9980.31	99.9	2.51	480.0	0.88	0.25	56,46	204.0

Table 2.1. provides an overview of the statistically important measures to understand the data's central tendency, variability, and distribution. Median value (116.0) lower than mean (139.27) suggests right-skewed asymmetric distribution (Figure 2.2.) which is also justified by a positive skewness value (0.88). High variance (9980.31) and standard deviation (99.9) values indicate wide dispersion of the data points. A relatively lower value of kurtosis (0.25) suggests that the distribution of the points is somewhat closer to a normal distribution with fewer data points that are very far from the mean value. 25th percentile value (56.46) shows that the lower 25% of the measurements lie below this value, and the upper 25% above 75th percentile value (204.0).

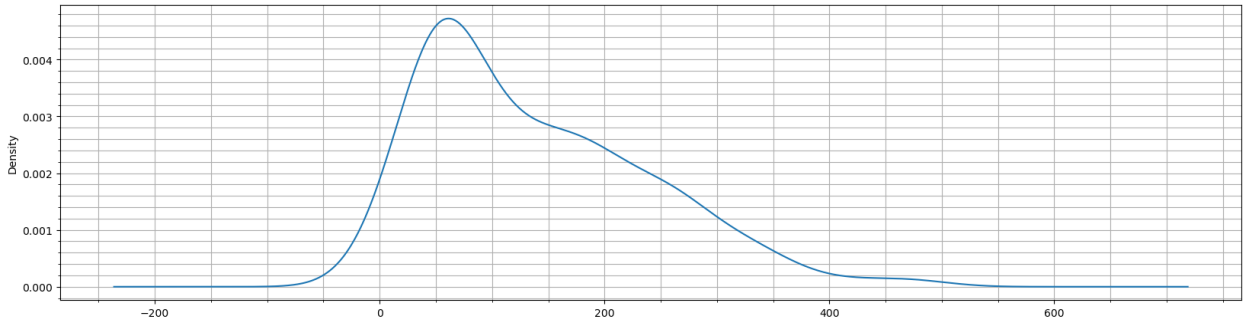

Figure 2.2. Kernel density estimate (KDE) plot for production data

Boxplot is generated in Figure 2.3. to visualize important statistic features across the years. Median values show variations at different times, with the year 2003 having the highest and the year 2018 the lowest median. The presence of outliers in the years 2004, 2007, 2008, 2014 and 2016 indicates potential unusual production rates deviating from normal range which in the majority of the cases coincide with the shut-in periods.

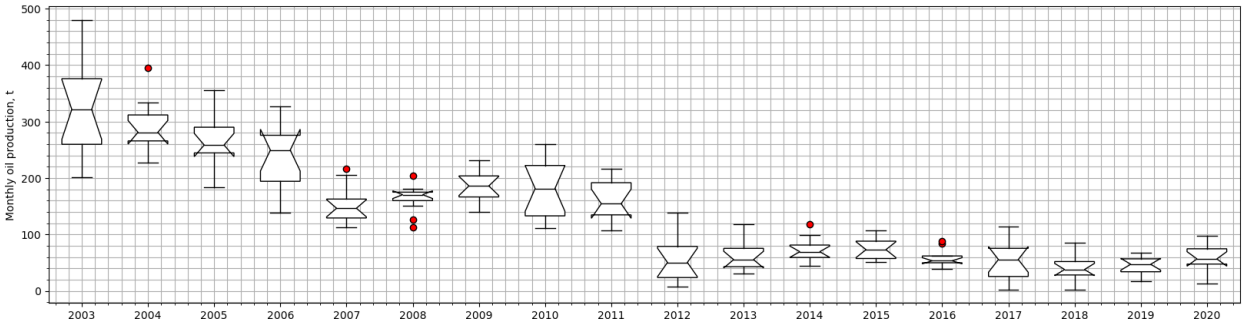

Figure 2.3. Boxplot of monthly oil production grouped by year

In our study, we also conducted a lag analysis on the production data to understand temporal relationships within the dataset. Figure 2.4. shows lag plots of monthly production data, specifically, between p(t) and p(t+1), p(t+6) and p(t+12) to reveal the temporal dependencies and autocorrelation within the dataset.

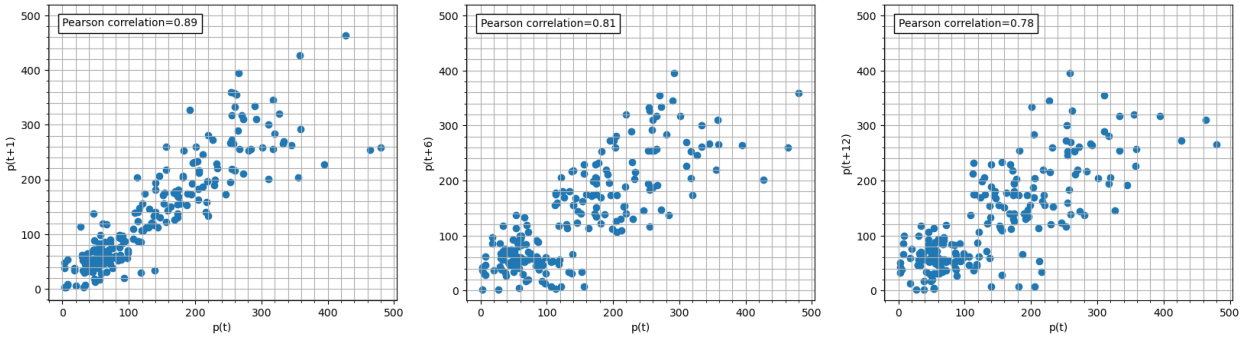

Figure 2.4. Lag plots between current and consecutive time points

Pearson correlation coefficient further quantifies the degree of relationship between lagged points. It outputs a value between -1 and 1, with negative value indicating a negative relationship and positive value showing a positive relationship between the variables. The Pearson correlation coefficient between x and y variables is given as follows:

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 (y_i - \bar{y})^2}} \quad (1)$$

where r is Pearson correlation coefficient, x_i is the x variable samples, y_i is the y variable samples, $\bar{x}$ and $\bar{y}$ are x and y mean values, respectively. A consistently decreasing trend can be observed with Pearson correlation coefficient suggesting that linear association weakens as lags are introduced. Figure 2.5. depicts Pearson correlation coefficient over the whole range of time span. The

autocorrelation values in-between -0.25 and 0.25 are considered statistically insignificant, meaning that no temporal linear relationship exists between corresponding lagged points.

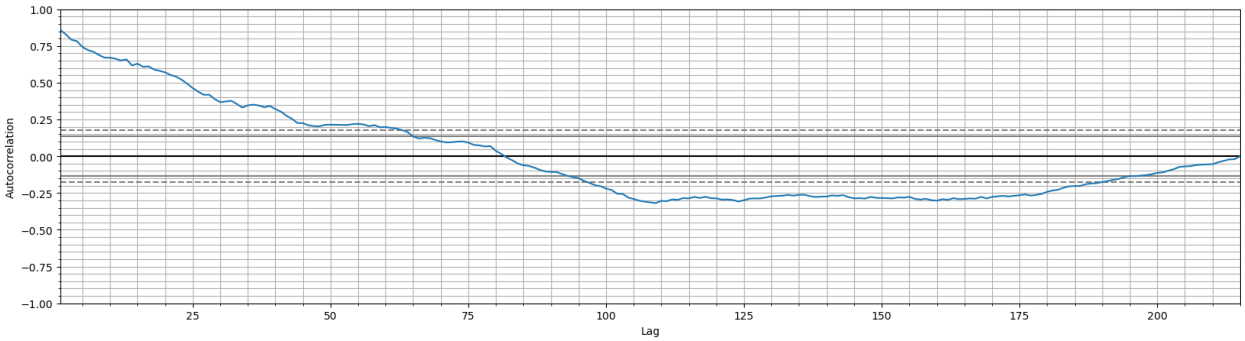

Figure 2.5. Monthly oil production autocorrelation plot

2.3. Time Series Decomposition

As discussed previously, it is important to understand if the production data is suitable for prediction purposes, specifically, when used with the long-short term machine learning model. Since the flowrate is controlled by means of a choke, a human factor is involved in this process. Various factors can influence decision on selection of appropriate flowrate for a specific day (Liu et al, 2020). This fact adds some degree of uncertainty into the dataset, filling it with some degree of uncertainty. The issue to be addressed here is to find out if the dataset contains enough information for long-short term machine learning model to learn and make informed decisions rather than giving random pair of points. For this purpose, it was decided to decompose data into its components to understand the underlying relationships within the dataset and to decide about suitability of dataset for long-short term memory applications.

Time series datasets exhibit a wide variety of patterns, each being unique. The time series datasets can be decomposed into their respective components to understand underlying relationships within the dataset. The patterns in the time series dataset include trend, cyclicity and residual components (Ali & Faraj, 2014). In other words, when we think of a time series dataset, we can think of it as a sum of three components including trend, cyclicity and residual component.

Trends are the easiest amongst other to understand and can be observed without use of any means of mathematical tools. Trends exhibited by time series data can be quite different. Linearly increasing or decreasing trends can be observed. In some more advanced datasets, trends can be quite different, exhibiting non-linear relationships similar to exponential or logarithmic. For

example, in buildup tests in well testing applications, the bottom hole pressure can be observed to increase in a logarithmic fashion over time, once the well is closed. In the case of production rate from well, due to a steady decline in reservoir pressure, apparently, a decreasing trend must be observed for production rates. According to the industry practice, trends are likely to exhibit exponential decline. Of course, one must note that this trend can be expected only in the case when other effects are not considered (water/gas injection, simulation, perforation, etc.).

Exponential decline trend, in fact, lies foundation for Decline Curve Analysis (DCA) technique, widely applied in the production trend. Analysis considers the fact that production would exhibit an exponentially declining trend over time. The assumption is rather naïve and neglects effects of external factors. The method tries to match an analytical exponential curve to the dataset and use it for prediction purposes. Since an analytical curve is being employed, the model proposed cannot effectively capture the trends in the dataset and follows a straight line. Although the method has been being used for more than several decades in the petroleum industry, it lacks quite lots of important details and cannot fully capture the production potential of the reservoir.

Cyclicity component is another output that is obtained when performing time series decomposition. For the number of flights taking place in a specific part of the world, time series decomposition is suitable in the holidays it can be expected to have a rise in the number of flights (Werbos, 1988). When considered over an extended period of time, clear cycles can be observed in the data which can also be seen from cyclicity graph. However, in the case of production, specifically, from oil wells, expecting this type of cyclicity is under question. For gas wells, apparently, apparently appears to be a cyclicity in the data as the demand for gas rises in the cold weather coinciding with the winter. Analyzing gas production over extended period of time, cycles of the increased gas production can be observed. Nevertheless, for oil wells there appears to be no such a trend. Oil is likely to be used all the year long being independent time of the year. Though, considering maintenance times and contractual term, there might be some degree of cyclicity in the data.

Residuals are the last components of the time series decomposition. It is what is left once trend and cyclicity components are subtracted from the dataset. In general, residuals show degree of unexplained points in the dataset or stated differently degree of randomness in the dataset. If residuals are too big, it means that traditional decomposition techniques are not suitable for that specific sort of data and some more advanced techniques need to be used. Despite this, there happen to be datasets not suitable for any decomposition techniques.

Two different types of time series decomposition techniques are available in the literature. One suggests that the signal itself is a sum of components of the signal, while another claims that signal is multiplication of these components. Depending on the decomposition technique chosen, different techniques can be employed. A variety of techniques are available for time series decomposition including but not limited to Classical Decomposition, X11, SEATS, and STL.

As mentioned previously, classical decomposition assumes that the signal is the sum of components of a trend. In this research classical decomposition was employed for both summation and multiplicative model.

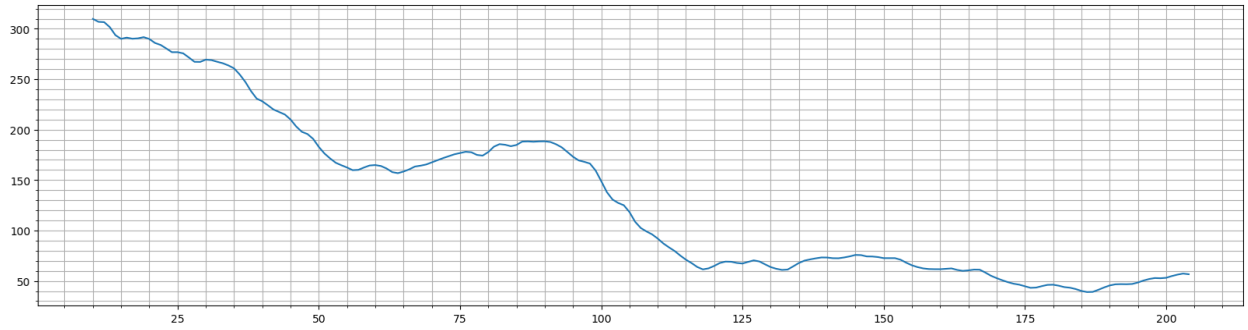

Figure 2.6. Trend component of a classical decomposition of a dataset

As we can see that the production data exhibits a decreasing trend over the period of 17 years. To be more precise, we can see an exponential decline in the trend. However, a closer look at the plot reveals that production rate increased in the middle of the period. Upon checking well history, it was validated that the period coincided with the water injection period. To recap, as mentioned previously, the production data follows an exponentially decreasing trend over the given time span except for places where some externalities are considered. The graph provided appears to be the first component of a time series analysis.

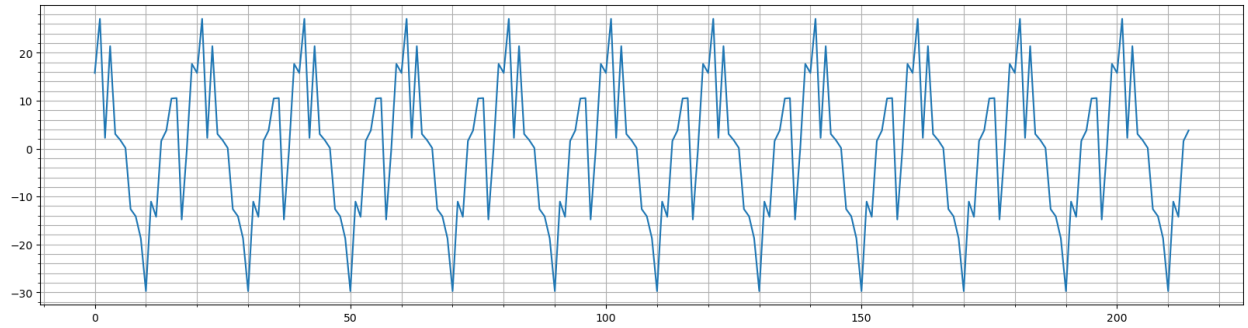

Figure 2.7. Cyclical component of a classical decomposition of a dataset

Looking at the cyclicity graph, it can easily be observed that clear cycles can be observed with period number of 20. Thus, it can be concluded that the data is not fully random, but patterns exist in the data allowing the model to capture. Since long-short term memory as a machine learning model requires informative data to find suitable set of trends and intricate patterns in the data and learn from them through multiple gating mechanisms. Apparently, there appears to be clear indication of cycles in the data, ensuring validity of long-short term memory machine learning model suitable for specific task of oil production prediction. The expected cyclicity, as mentioned previously, would be expected to be much higher in the case of gas wells due to seasonality matter.

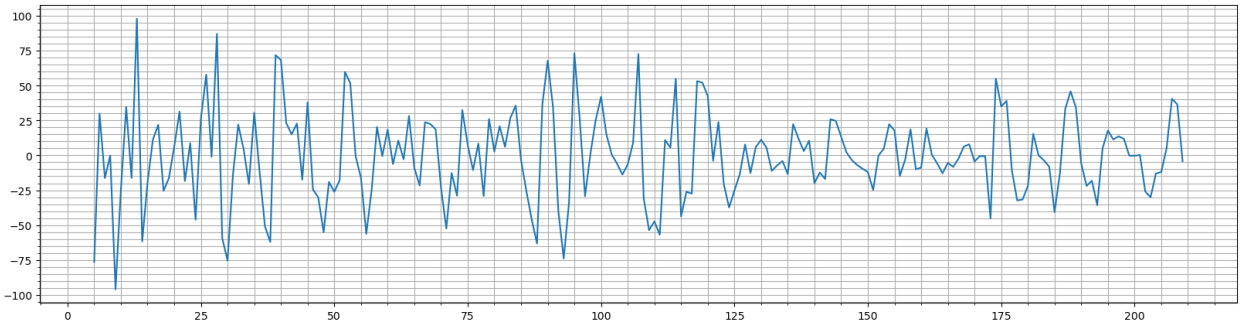

Figure 2.8. Residual component of a classical decomposition of a dataset

Plot provided above shows residual component of a oil production data. We can see that the signal tends to fade away over time. Together with trend component, and cyclicity, all these components add up to the signal itself.

By analyzing individual components of historical production data, several key aspects of the data were revealed. Long-short term machine learning model appears to be for this specific task since the data contains enough informative content to be analyzed and learnt by the network. Besides, low values of the residuals suggest that the degree of randomness (presence of human factor in production rate determination) is rather low, and the model can very well be used to handle this specific type of data and be used for long-term predictions.

2.4. Data Preprocessing

The data was split into two parts, forming training and testing datasets, comprised of 179 and 36 data points, respectively. Selection of number of testing points is arbitrary, but a period of 3 years is thought to be a good basis for analyzing the behavior of the ML algorithm. The training set is used for developing and training the prediction model to be used, while the testing set remains untouched and is later used to assess the performance of the model.

Since monthly production recordings in Western Siberia are in tons (t), the values are large and highly heterogeneous to be fed into the network. Such values might be problematic for the proposed prediction model as it might suffer from generalization errors and poor performance. Standardization, also known as feature scaling, was performed to have consistent scale and distribution using the formula:

$$x_{i,scaled} = \frac{x_i - \mu}{\sigma} \quad (2)$$

Here, x_i is the individual monthly production value at a given time, while μ and σ are the mean and standard deviation values, respectively. Standardization centers the data resulting in a mean of zero and standard deviation of one. Robustness to the outliers in the dataset is an important feature of standardization when compared to other normalization techniques (Chai & Draxler, 2014).

Similar to data standardization, there is another technique used, so called normalization. Unlike standardization, normalization does not affect standard deviations or mean of the dataset but restricts data within the interval of (0, 1). In this research standardization was selected as a mean to preprocess the data as it is better at handling outliers in the data, unlike normalization. Considering the fact that well flowrate data is likely to contain multiple outliers, and the data is filled with random points, it appear to be a wise choice to use standardization for this purpose.

2.5. Model Selection

The choice of using LSTM neural networks for long-term oil production forecasting in this paper is attributed to their inherent potential to capture temporal dependencies in the sequential (Kohavi, 2001). Previous studies and practical applications in similar domains involving time series provided strong evidence of LSTM's ability to handle intricate patterns.

A new class of neural networks, Recurrent Neural Networks (RNNs) were initially introduced to address the challenges involving time series (Nguyen & Chan, 2005). A unique architecture owned by RNNs allows them to retain information throughout consecutive time steps making them suitable for sequential data (Staudemeyer & Morris, 2019). In the process of training, RNNs employ backpropagation through time (BPTT) algorithm allowing it to adjust the weights by calculating gradients. Despite advantages offered by RNN models for tasks involving sequences, vanishing/blowing gradient problem renders training increasingly inefficient over long temporal span of the dependencies (Hu et al., 2020).

With the intention of mitigating limitations in the traditional RNN models, a more sophisticated variant – LSTM (Werbos et al., 1990) was introduced to thoroughly handle long temporal relationships. Unlike RNN counterparts, LSTM networks can capture long-term dependencies in the sequential data and effectively remember extended temporal relationships by avoiding derivative issue. An intrinsic solution to the problem lies in the use of constant error carousel (CEC) by the model which ensures retention of error signals in each unit cell, thereby enabling the gradients to endure over long sequences (Ros, 1960).

A distinct chain-like architecture of LSTM network allows sequential input data to be processed in each memory block (Figure 2.9.), after which two different states, namely cell state and hidden state, are forwarded to the consecutive cell. The cell state is a crucial component of a cell responsible for the flow of information through consecutive memory blocks. Cell state plays the role of a conveyer belt (Gers & Schmidhuber, 2001) by propagating information forward to the next cell. The cell state is updated through different gate mechanisms which allow careful selection of relevant information from the current and previous time steps while filtering out irrelevant details. The role of the hidden state is to retain relevant context and learn patterns from the sequence being processed. Dynamic updates at each time step are provided to the hidden state in accordance with the current input, the previous hidden state, and the learnt parameters.

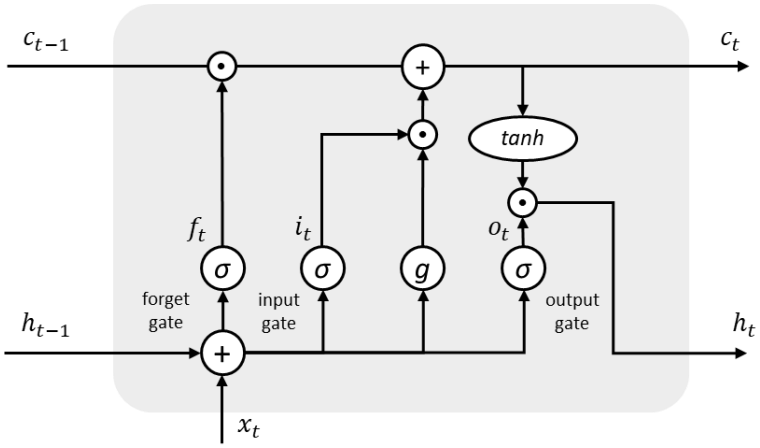

Figure 2.9. LSTM cell architecture

The internal workflow of memory blocks in LSTM network can be divided into several stages. At each time step two pieces of information are considered to create an input vector: the input at a current step (x_t) and the hidden state from the previous time step (h_{t-1}). The first step in LSTM network is to discard unnecessary information from the previous cell state (C_{t-1}) by activating the forget gate (f_t). This is achieved through sigmoid function which produces a value

in the interval of 0 to 1 by taking hidden state (h_{t-1}) from the previous time step and input (x_t) from the current step. The value of 0 indicates complete elimination of the information, while the value of 1 means extreme significance.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (3)$$

In the next step, the input gate is activated to regulate part of the new input data to be stored in the cell state at a current time step. Sigmoid function here again takes the hidden state (h_{t-1}) from the previous time step and the input (x_t) at a current time step.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (4)$$

In the following step, an intermediate vector (g_t) representing information to be stored in the cell state is formulated with the help of hyperbolic tangent function. It weighs relevant features from the current input and gives numbers in the span of -1 and 1. The use of non-linear function allows complex relationships in the data to be captured by the network.

$$g_t = \tanh\left(W_g \cdot [h_{t-1}, x_t] + b_g\right) \quad (5)$$

The next step is to update the cell state by incorporating the previous cell state (C_{t-1}) and the combination of the input gate and the intermediate vector.

$$C_t = C_{t-1} \cdot f_t + g_t \cdot i_t \quad (6)$$

One last step is to activate the output gate to find out portion of the updated cell state to be added to the hidden state.

$$o_t = \tanh(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (7)$$

$$h_t = o_t \cdot \tanh(C_t) \quad (8)$$

The weight (W) matrices and bias (b) terms with their corresponding indices are learnt during the training of the model. It involves the use of BPTT algorithm (Elshafei et al, 2011) to effectively update weights and biases for the multiple training epochs. Backpropagation through time (BPTT) is a common technique used specifically with neural networks and also including RNNs and LSTM (St-Aubin & Agard, 2022). Backpropagation through time involves series of steps implemented to improve the accuracy of the model and to find correct set of parameters for optimizing the model. Everything starts with the forward pass, when input sequence is fed into the LSTM network one step at a time in a stepwise fashion. At each time step, as previously mentioned

three main components are calculated, including input, output and forget gate (Elmabrouk et al, 2010). Input gate here determines which time step to update from the previous cell state, forget gate, on the other hand, is responsible for finding out which information should be discarded from the previous cell state, and finally, output gate finds out which information should be outputted for final result. All the information inputted flow through the cells. Once all the input information goes through these stages including input, output and forget gate, the loss function with the versatile algorithms are calculated at the end. Quite a wide range of algorithms are available for this purpose including but not limited to root mean squared error (RMSE), mean squared error (MSE), cross-entropy error and many more. One the primary objectives of the backpropagation algorithm is to calculate the parameters needed to adjust the model and achieve the possible lowest error. When backpropagation algorithm modifies the data points through time, it starts to calculate gradients of the loss function with respect to each parameter (Berrar, 2018). A very famous chain rule is used from calculus to calculate respective gradient for each parameter, and start from the output layer and move backwards in the direction of input as the name suggests.

At each time step that is fed into Long-short term memory algorithm, the gradients are calculated with respect to outputs, gates, and finally cell states. Calculated gradients are later used to update parameters of the Long-short term memory cells to achieve highest possible accuracy in the predictions and lowest error (Berneti & Shahbazian, 2011). As mentioned previously Long-short term memory appears to be more advanced than Recurrent Neural Network since exploding or vanishing gradient is no more problem in this case. For this purpose, only gradient clipping is applied which involves scaling of the gradient once they exceed the given threshold. The parameters in Long-short term memory algorithm can be updated through variety of algorithms including but not limited to stochastic gradient descent (SGD), Adam or in some cases RMSprop can be preferred. Once parameters are updated the model is made for the next iteration.

Backpropagation through time allows continuous adjustment of the network parameters, specifically, in the case of Long-short Term memory, thereby, lower the errors in the predictions and achieve highest possible accuracy. It is obtained when the signal is passed forward and backward through time (Elman, 1990). By doing so, the model can capture very intricate trends in the data and make informed decisions about the future by consider previous relationships between successive data points.

2.6. Time Series Cross-Validation and Hyperparameter Tuning

In the ML context, validation dataset refers to the separate portion of the dataset used to evaluate the performance of the model and tune hyperparameters (Achong, 1961). The model's performance is assessed on the validation dataset during the training to understand how well the model generalizes for the unseen data (Bengio et al, 1994). Learning curves involving validation performance and the number of training epochs can be interpreted to understand if overfitting or underfitting takes place.

There are distinct types of validation techniques in the machine learning (Hochreiter & Schmidhuber, 1997) used to assess the behavior of the model. The selection criteria lie in the specific type of data handled and its size. Traditional validation techniques such as standard K-fold cross-validation and simple holdout validation techniques cannot be employed with the time series data due to its sequential nature. Unlike these methods that randomly split or shuffle the mutually independent observations (Shekar & Dagnew, 2019), the specialized validation technique – time series cross-validation retains the temporal order of the data and is well suited for the chronological type data.

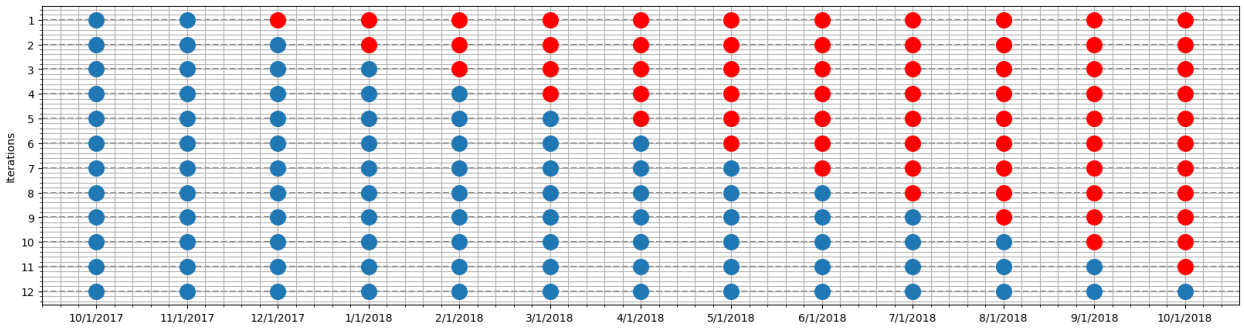

Figure 2.10. Time series cross-validation with expanding window size

Evaluation on a rolling forecasting origin with expanding window size is a type of time series cross-validation technique used in this paper. Figure 2.10. explains the basic work principle of expanding window cross validation for a section of dataset. Temporal dimensions in the graph are shown with horizontal lines, while blue and red colored scatters represent training and testing datasets, respectively. Initially, the LSTM model is trained on a pre-defined length of time series data points and used to predict a single observation in the test dataset. Later, observed value is added to the train dataset to re-train the model and make prediction for the next time point. This procedure is iteratively repeated for all points in the prediction horizon and in each step, metric mean absolute error (MAE) is used to estimate the difference between the real and predicted time

point. All errors are averaged at the end to assess the behavior of the model as the new data becomes available (Hochreiter & Schmidhuber, 1997). The procedure imitates the real-world scenario where new monthly oil production data is added to the historical data on a regular basis and the model needs to adapt changing patterns as the new data becomes accessible.

This methodology allows to fine-tune critical hyperparameters including the window size, number of epochs and units in the LSTM architecture. Different algorithms are available to choose optimal combination of hyperparameters to significantly improve predictive performance of the model (Park, 2022). Grid search algorithm is selected to adjust hyperparameters with a given hyperparameter space. It has straightforward yet effective work principle. Multiple values are assigned to each hyperparameter, and the model's performance is systematically evaluated for each combination within the defined grid.

2.7. Reservoir Simulation Model

The next step in the research was to employ a reservoir simulation model as suggested before to make oil production prediction for a span of 3 years. For this reason, a software, so called, tNavigator was employed. It is a reservoir simulation software developed by researchers of Rock Flow Dynamics used to run dynamic reservoir models on laptops, servers and HPC clusters. The core script of tNavigator is written in C++, making it quite stable from calculation point of view. It is designed to handle parallel algorithms on multicore and manycore shared memories (Gers et al, 2000).

The first step was the collection of basic field data necessary for specifying the historical operation of the well in the hydrodynamic model (Rumelhart et al, 1986). The data for the model included:

1. Data on production and operating time of the well according to monthly operational reports.
2. Data on the dynamics of bottomhole pressures according to the technological regime.
3. Data on geological and technological measures.
4. Results of field geophysical and hydrodynamic studies of the well.

After collecting the basic data necessary to set up the well, a procedure was performed to match the model parameters to achieve satisfactory convergence of the calculated and actual data. The history match procedure includes adjusting the operating parameters of the reservoir and wells until satisfactory convergence with actual data is achieved. In a mathematical sense, the history

matching process is a solution to the inverse problem - we know with a high degree of reliability the results of well operation (production data), but the input data (formation and well operation parameters) are known with less reliability.

During the history match, serious deviations from the results of calculating the hydrodynamic model from the actual data are identified, which indicates the need to clarify the initial field and geological data and to adjust the input data to eliminate deviations and achieve satisfactory convergence of the model with the actual data (Mirzaei-Paiaman & Salavati, 2012). Based on the results of the history match, the understanding of weaknesses and inconsistencies between the initial data increases, and the understanding of the energy state of the formation and the nature of the distribution of filtration flows improves. Also, based on the results of the history match, it is possible to obtain an idea of the structure of remote parts of the deposit that have not been covered by detailed studies.

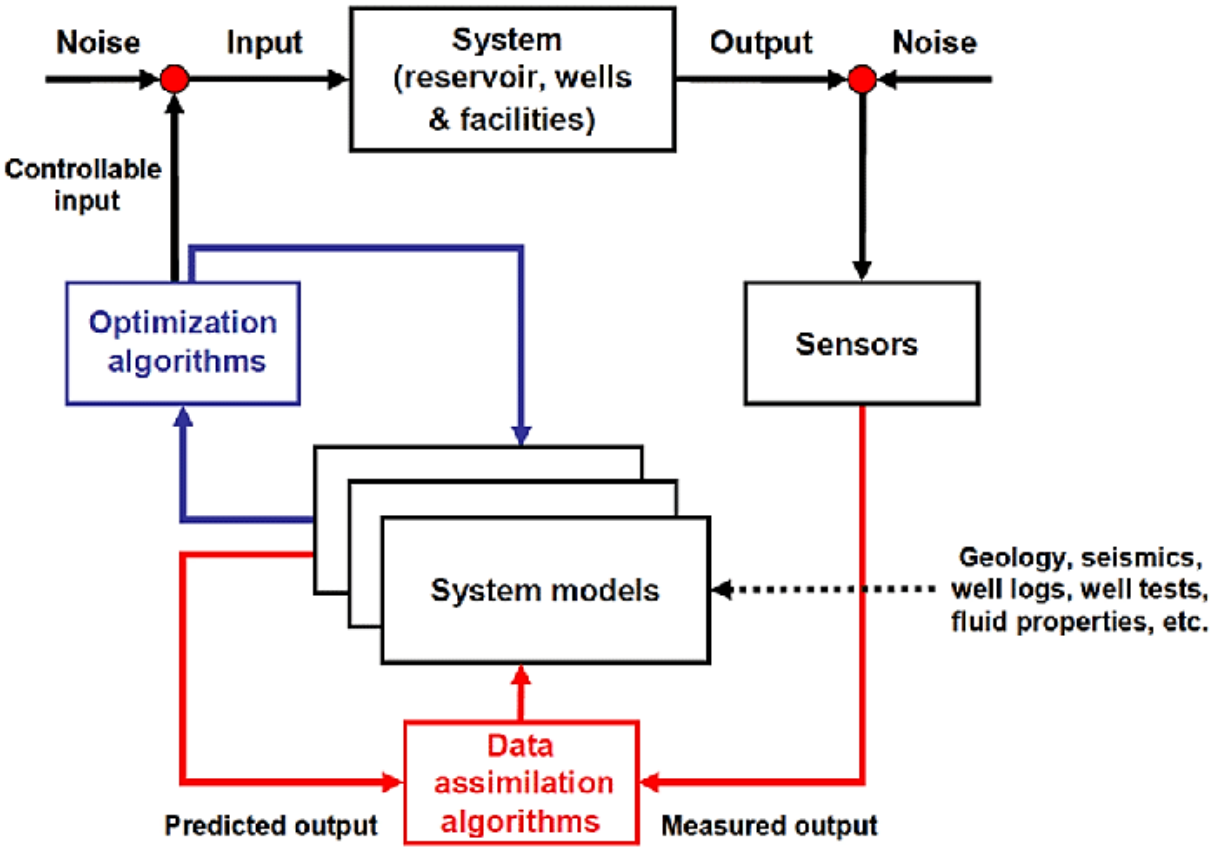

Figure 2.11. Closed-loop reservoir management (Jansen et al. 2009). The uncertainty in the reservoir output data affects history matching (data assimilation) and reservoir management. A part of this uncertainty is the errors in the recorded production data.

Given that different production and geological data characterize the performance of the reservoir and wells at different scales, each of them also carries a different degree of error. Therefore, at the initial stage, it is necessary to assess the completeness and reliability of all available information, as well as sensitivity analysis to identify the most significant parameters. As a rule, from core and log data it is possible to determine the initial distribution of fluids in the reservoir, lithological structure and porosity. Data on absolute permeability are most often distributed over a wide range of values (with the exception of hard-to-recover reserves, where permeability takes on very low values). Laboratory studies to determine relative phase permeabilities characterize reservoir intervals on a scale of a few centimeters, while the cell sizes of hydrodynamic models are 50x50 or 100x100 meters (most often). For correct transfer, relative phase permeabilities need to be modified depending on the available initial data. If a significant amount of historical data is available, the RPTs are adjusted in such a way as to reproduce the actual dynamics of water cut. If there is no data on the dynamics of water cut, a two-phase upscaling procedure is performed (transfer of laboratory experience to a hydrodynamic model with setting the corresponding laboratory cell sizes). Considering the fact that each field is characterized by its own structural features, it is incorrect to single out key parameters for adaptation. A detailed uncertainty and sensitivity analysis is required in each case. However, the following parameters, as a rule, are characterized by the greatest error relative to others due to the fact that either there is insufficient initial data for them (poorly studied), or due to their specificity, they are characterized by a high scatter of values (specific in nature):

1. Absolute permeability.
2. Relative phase permeability.
3. Vertical anisotropy of permeability.
4. Parameters of the contour area (aquifer).
5. Parameters of hydraulic fracturing cracks.

During the work, the listed parameters varied primarily since they were characterized by the least completeness of data. Data on oil and liquid production, as a rule, are the most accurate, so it is most advisable to carry out calculations in the mode of specified liquid extractions. However, production data may still contain errors due to the peculiarities of stock measurement. If flow rates are measured not at wells, but at production collection points, then an error in the distribution of production between wells is possible.

At the first stage, history match of the energy state of the reservoir was carried out, which consisted of achieving actual fluid withdrawals based on the results of model calculations together with achieving satisfactory convergence in the dynamics of bottomhole pressure (in the absence of reservoir pressure). The main setting parameter at this stage was the piezoelectric conductivity of the formation. Taking into account the uncertainty analysis carried out (identification of the most uncertain parameters that need to be clarified), the absolute permeability of the formation was used as an adaptation parameter.

Upon achieving satisfactory convergence in fluid extraction and general reservoir energy (bottomhole pressure dynamics), water cut adaptation was performed. Considering that the absolute permeability values had already been selected, an adjustment was made to the general form of the relative phase permeabilities. Relative phase permeability functions (RPF) of oil reservoir rocks are the most important component of FM. As it is known, the relative permeability of a rock characterizes the ratio of phase permeability to absolute permeability for a given phase. The combined flow of dissimilar liquids (or liquids and gas) through the pore space of reservoir rocks significantly affects the permeability of the same rock for any component of the mixture of filtered liquids. In this case, to characterize the filtration properties of rocks and in various calculations, the values of the so-called phase permeability are used, which characterizes the permeability of rocks for a given gas or liquid.

After the most extensive parameters of the formation have been configured, it is also necessary to take into account the geological and technological measures carried out at the well during its operation. Since the purpose of such measures is to stimulate inflow in the near-wellbore zone, the work used parameters that characterize the operation of only the near-wellbore zone (in the reservoir-well interval). Since detailed information on changes in the skin factor was not available, the well productivity index multiplier (WPIMULT) was used for the dates of geological and technological activities (Heaton, 2008). The selection of the multiplier was carried out based on the dynamics of fluid flow rate and bottomhole pressure until satisfactory convergence was achieved on the date of geological and technical measures.

Based on the results, satisfactory convergence of calculated and actual data on both production, pressure and water cut was achieved. Based on the history match results, a predictive calculation of well operation was performed in the basic operating scenario - that is, in the last design mode, with control based on the last design bottomhole pressure. This calculation scenario can characterize the operation of the well in the coming years if homeostasis is maintained in the

system (no sudden changes in the environment and interference effect, no geological and technical measures, etc.). No other operational or economic restrictions (maximum water cut, minimum profitable flow rate, depression value, etc.) were specified for the forecast period.

Next, a predictive calculation was performed with control based on the last calculated bottom hole pressure. The forecast calculation was carried out for 3 years with control based on the last calculated bottom hole pressure.

2.8. Performance Evaluation Metrics

Accurate performance evaluation metric is critically important to understand quality of the forecasting model employed (Xiaomeng & Xinyu, 2022). Performance evaluation of the model on unforeseen data allows the comparison of the distinct future prediction models and selection of the right model for the data. Nevertheless, the choice of the metrics depends on the specific purpose of the user and characteristics of the time series data.

Root Mean Squared Error (RMSE) is a standard metric for time series data used to evaluate the goodness of fit by quantifying average magnitude of errors between the observed and predicted values (Bergstra & Bengio, 2012). RMSE is sensitive to outliers, meaning that when penalizing errors, it gives more weight to larger errors.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(O_i - P_i)^2} \tag{9}$$

In the above equation, the O_i and P_i terms show observed and predicted values, respectively; and n is the total number of observations.

Mean Absolute Error (MAE) is another useful metric in the regression model evaluation. Unlike RMSE, MAE gives the same weight to all errors.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|O_i - P_i| \tag{10}$$

Another important metric to be used in the regression model evaluation is Mean Squared Log Error (MSLE), especially useful for the datasets following exponential trend (Camacho & Raghavan, 1989). MSLE penalizes overestimation and underestimation proportionally, thereby providing more insight into relative errors in the predictions.

$$\mathrm{MSLE} = \frac{1}{n}\sum_{i=1}^{n}(\log(O_i + 1) - \log(P_i + 1))^2 \tag{11}$$

RMSE, MAE and MSLE can be used to compare prediction results over different models and assess how well the fit is (Zhang & Zhao, 2013). The lower value implies a better fit, indicating that the predictive model can effectively capture patterns and trends in the sequential time series data.

CHAPTER 3. RESULTS AND DISCUSSION

The selected hyperparameters for cross-validation included window size, number of units and number of epochs in the LSTM architecture. These are crucial hyperparameters that significantly affect the performance and behavior of the LSTM model. Choosing a correct combination of these hyperparameters involves finding a balance between capturing sufficient information, preventing overfitting, and avoiding unnecessary computational cost.

Different grids were assigned to each variable; for window size, the grid consisted of the values [3, 6, 12], while the number of units was associated with the grid [32, 64, 128], and the number of epochs had its grid defined as [25, 50, 100]. Figure 3.1. shows MAE values for each hyperparameter combination as distinct vertical bars with MAE values given on the horizontal axis. Notably, a consistent decrease in MAE values can be observed with increasing complexity in the LSTM model. The observed pattern aligns with the intuition that a more sophisticated model with a high number of units and epochs is more likely to exhibit an improved predictive accuracy. Similarly, an inverse relationship between the window size and MAE values can be noticed. With an increasing window size, a consistent decrease in MAE values can be observed from Figure 3.1. for multiple combinations of number of units and epochs. Meaning that, bigger window sizes are likely to capture long-term temporal dependencies and better discern underlying patterns and trends in the dataset.

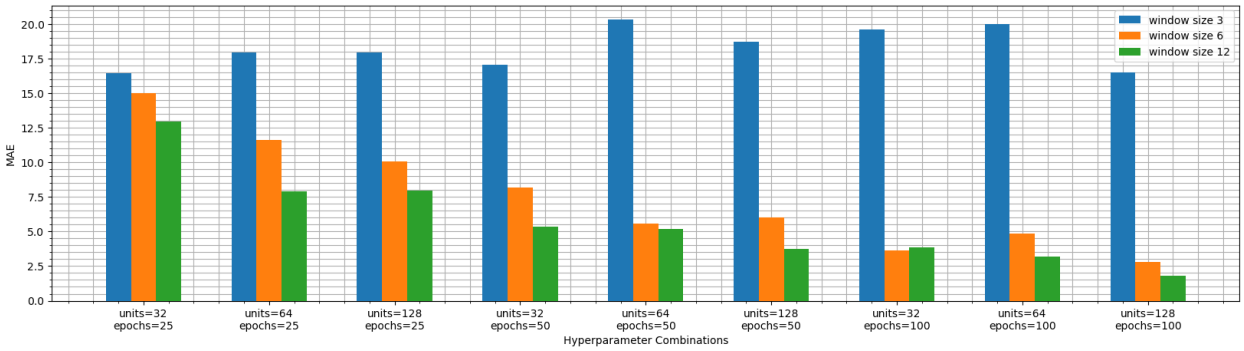

Figure 3.1. MAEs for multiple hyperparameter combinations within the defined grid

Figure 3.2. illustrates evolution of Error Values (AE) throughout the cross-validation process for the window sizes of 3, 6, and 12 at a fixed number of units (128) and epochs (100). With an increase in the number of training data points during cross-validation, a decreasing trend can be expected for AE values as more and more data becomes available for training over the time. Unlike window size of 3, the AE trends for window size of 6 and 12 exhibit expected pattern.

Rather random nature of AE trend at a relatively small window size of 3 can be attributed to the inability of the model to correctly capture the temporal relationships in the data.

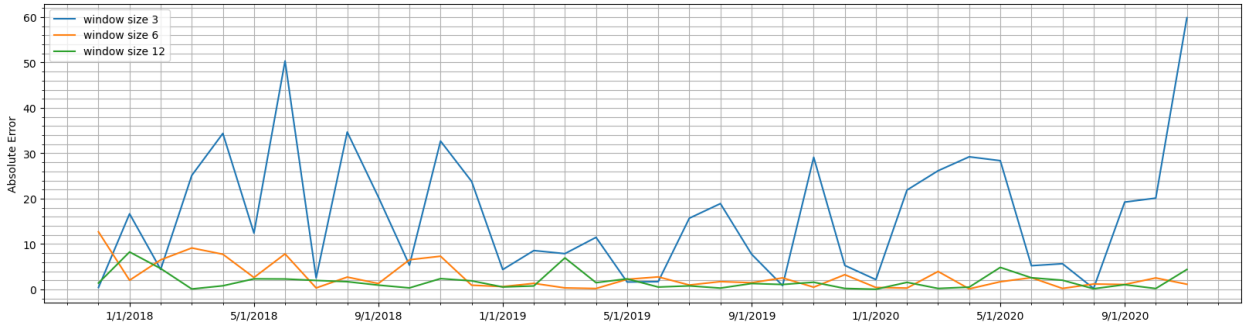

Figure 3.2. Absolute Error values for multiple validation iterations

Figure 3.3. represents the plot of predicted values during the cross-validation and the test dataset during the span of 3 years. The predicted values closely trace the actual values, demonstrating the model's capacity to accurately predict one step forward within a specified configuration of hyperparameters. The evaluation metrics for the given configuration are as follows: RMSE=2.58, MAE=1.78, and MSLE=0.07.

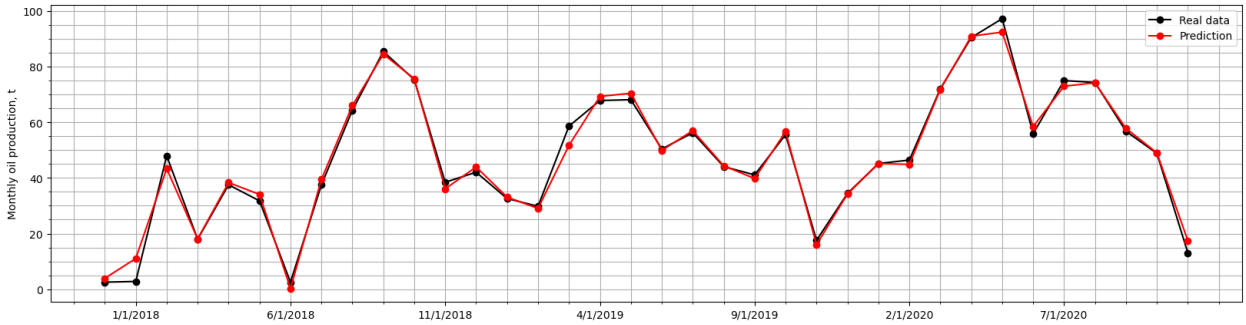

Figure 3.3. Comparison of predictions with the test data during cross-validation

For a single-step univariate prediction using bidirectional LSTM model, inputs and outputs were established on our time series by a systematic process of data windowing. Two different techniques were adopted for the input window generation. The first approach entailed production prediction for the next month and the substitution of the newly forecasted value with a corresponding real one in the process of window sliding. In the context of a single-step univariate prediction, a fixed window size of 12 months was selected as the input configuration, while the following month was assigned to be the corresponding output value. During the implementation of a progressive sliding, the data window was shifted forward one time step at a time and a new input and output pair was generated.

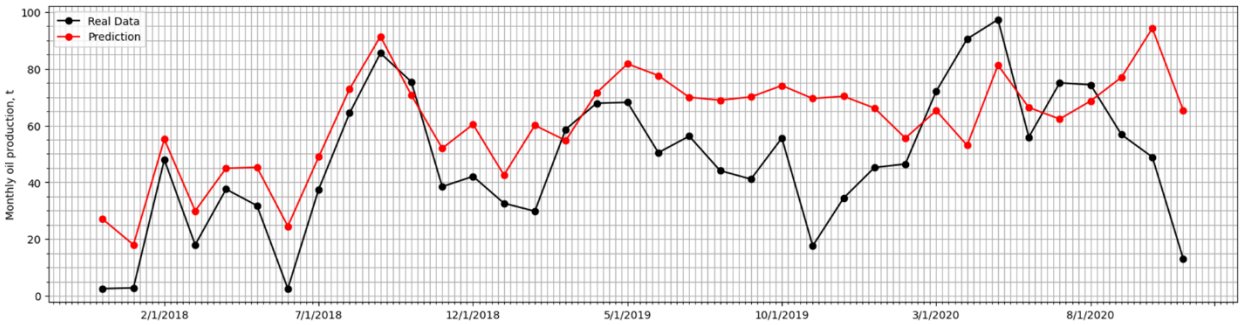

Figure 3.4. Univariate single-step bidirectional LSTM model prediction

Figure 3.4. compares predictions made by single-step univariate bidirectional LSTM model and real data over the given test period. Having a fixed number of training data points in this case, apparently, has a significant effect on the accuracy of predictions (RMSE=22.48, MAE=18.43, and MSLE=0.52). The model clearly demonstrates a tendency towards overestimation. Figure 3.4. shows a cross-plot of monthly predictions and the real data for a specified period with a reasonable estimated value of a Pearson correlation coefficient of 0.66. A big cluster of points above the line passing through the origin with a slope of 1 is a clear indication of overestimation of oil production. Also, a comparison of the historical and predicted cumulative oil production for the period of interest can be seen in the same figure. At the end of the period, the model overestimates the cumulative real data by 480 tones.

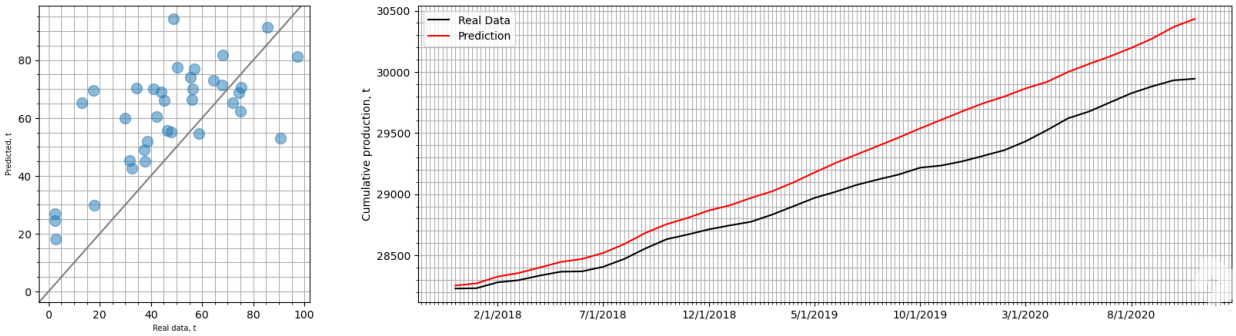

Figure 3.5. Unveiling overestimation tendencies in the univariate single-step model

The second methodology involved the usage of the same predicated value as a new input time step value for the following predictions – recursive forecasting – establishing a recursive production chain. Apparently, two different strategies employed for the univariate single-step prediction can potentially affect the accuracy of the predictions and the behavior of the model. Error metrics, as expected, were much higher in this case (RMSE=30.46, MAE=24.0, and MSLE=0.71) as errors are likely to accumulate and propagate over the time. Unlike the previous

methodology, in this specific case no test data was included in the data windowing part, making it more susceptible to errors over a given period. Figure 3.6. shows comparison of historical and recursively predicted values.

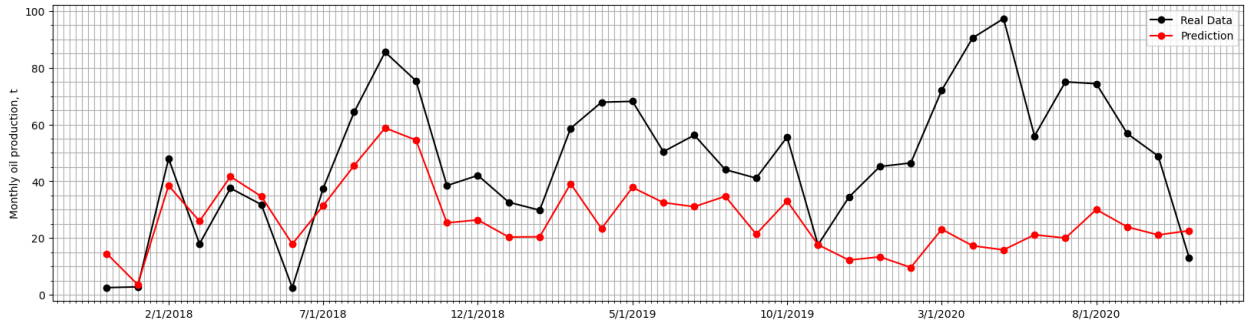

Figure 3.6. Recursive prediction model

Unlike the previous case, recursive forecasting clearly underestimates the cumulative production. Upon investigation of Figure 3.7., 758 tones of underestimated cumulative production can be observed at the end of the period. Besides, a scatter plot shows rather dispersed data points with a Pearson correlation coefficient of 0.4.

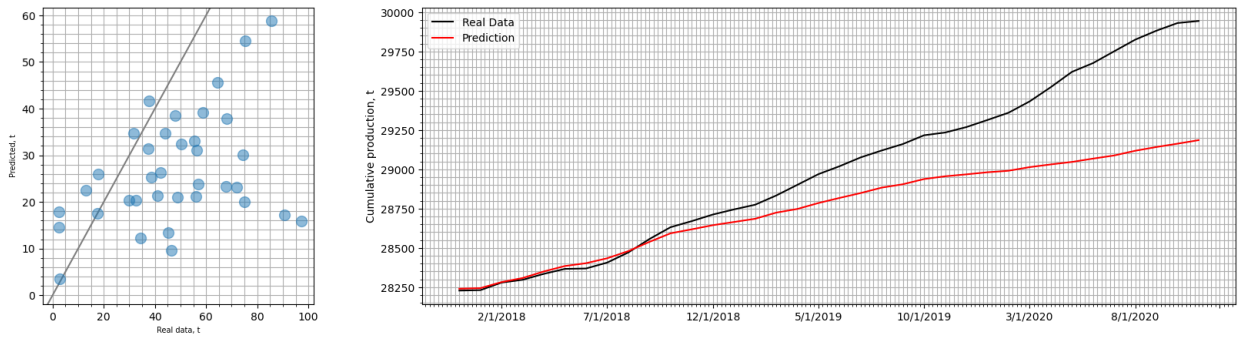

Figure 3.7. Unveiling underestimation tendencies of recursive forecasting model

As a last scenario, an extended period of consecutive 36 timesteps was used to generate a label spanning the next 36 months in the process of multi-step forecasting. Like recursive forecasting, in this case no test data was included in the process of data windowing for making predictions. Evaluation metrics for this case were slightly higher; RMSE=31.72, MAE=25.21, and MSLE=1.03. Figure 3.8. shows prediction results obtained by univariate multi-step LSTM model and compares it to the real data.

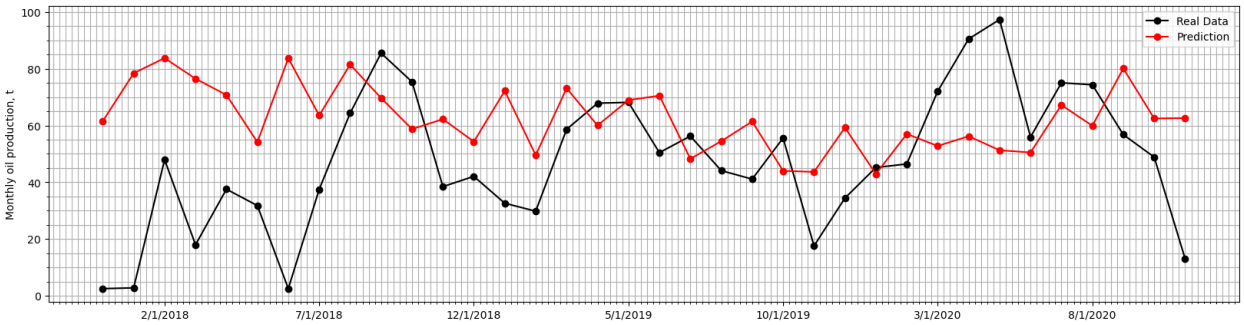

Figure 3.8. Univariate multi-step LSTM model predictions

At the end of the specified period, the univariate multi-step model overestimates cumulative oil production by 530 tones. A more dispersed scatter plot can be observed in this case, with a Pearson correlation coefficient of -0.22 (Figure 3.9.), suggesting a negative correlation between variables.

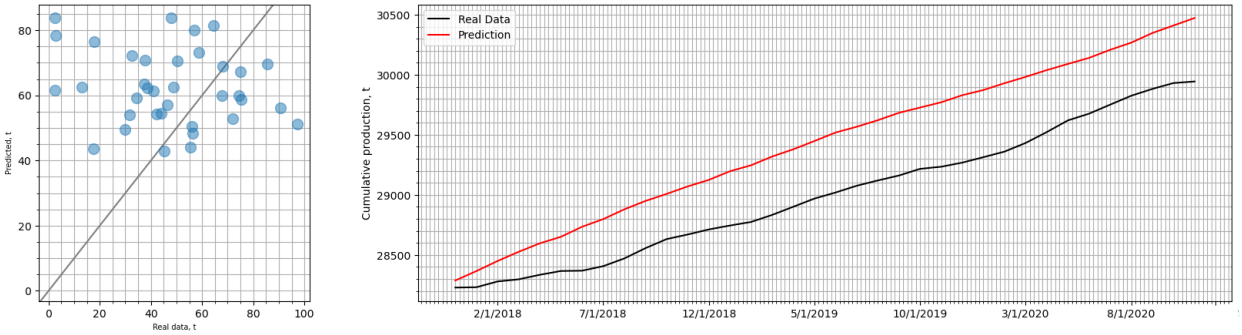

Figure 3.9. Unveiling overestimation tendencies in the univariate multi-step model

Similar to Machine Learning (ML) algorithms, the reservoir simulation software was used to make a prediction for a time period spanning 3 years. After the data collection, history match phases the model was finally ready to be used for forecasting purposes. To have a comparison of the results with Long-short Term Memory (LSTM) model, the curves were plotted together with the simulation forecast. A recursive production prediction trend was selected for the comparison due to the highest accuracy and flexibility in terms of application.

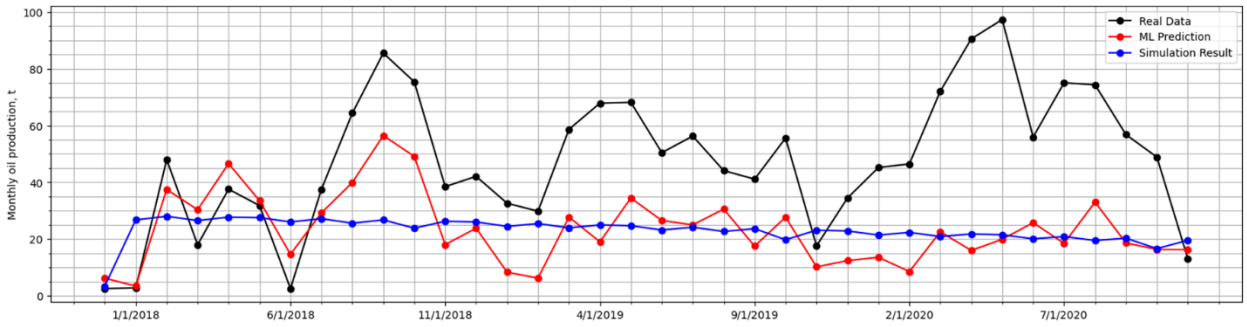

Figure 3.10. Comparison of reservoir simulation results and recursive production prediction using LSTM

The blue trend provided in Figure 3.10. shows simulation results. By comparing two trends – simulation result, and recursive prediction of LSTM model – one can see that unlike reservoir simulation forecast, the LSTM model is more likely to capture the trends in the data rather than following a steady declining curve. Upon analysis of the error metrics for both models, we can see that error rate is slightly lower for recursive LSTM model.

Table 3.1. Comparison of error metrics for reservoir simulation models and LSTM

	MAE	RMSE	MSLE
ML model	26.69	32.27	0.84
Simulation model	28.54	34.46	0.88

Looking at the Table 3.1., it can be seen that all metrics, including MAE, RMSE and MSLE are all lower for ML model by a small margin. The same procedure was carried out for cumulative production for 3 years of time.

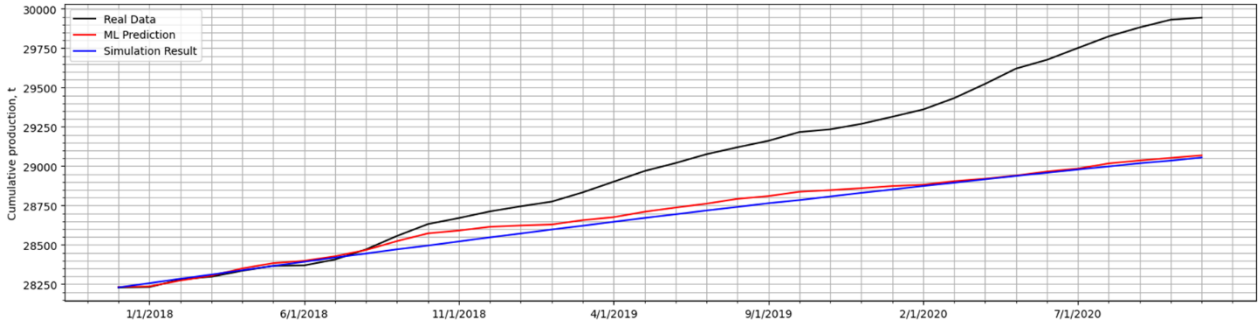

Figure 3.11. Cumulative production comparison for ML model and reservoir simulation model

One can see that the cumulative trends almost coincide identically over the whole period. Previously given results once again validate the fact that the errors for both models are quite low. Looking at the Table 3.2., we can see respective error in each model.

Table 3.2. Comparative results

	Predicted Value in 2020 in tons	Real cumulative production in 2020 in tons	Absolute error in %
ML model	29068	29943	3
Simulation model	29054		2.8

Looking at the table provided above we can see that according to real data true value of cumulative production at the end of the period, in other words in 2020, is 29943 tons being higher than both ML and simulation model.

Lag plot analysis was carried out to validate dependence of successive data points from each other. Pearson correlation coefficient was obtained for each, showing the degree of strength in each case. According to Pearson correlation coefficients, there is a strong degree of relationship between successive data points.

Moreover, whisker or box plot was created for production data based on yearly basis. The main purpose of which is to compare the interquartile range, mean and median of each year and try to find out specific dates that show deviation from the other points. According to the results of whisker plot, some points were identified as outliers and upon investigation of the maintenance/intervention times it was found that these points perfectly coincide.

As a next step in the analysis, the main idea was to check the validity of the dataset for long-short term memory machine learning algorithms. Since data appears to contain some degree of randomness, it was important to check extend to which data is interpretable. For this purpose, time series decomposition was carried out to see underlying patterns in the data set. A wide variety of time series decomposition techniques are available and traditional decomposition method was employed in this research to decompose production signal into its consecutive trend, cyclicity and residuals components. It was found that as expected the underlying trend in production data is exponentially declining curve. The cyclicity is not visible at the first glance, though, the classical decomposition algorithm managed to obtain cycles for moving average period of 20 months. Finally, residuals component was obtained which is a core idea in decomposition interpretation. The results showed that residual deviations are likely to fade away over the time, making the dataset interpretable in terms of machine learning context.

Multiple machine learning is available and being used in different field right now. The only problem is not all algorithms can be applied for production prediction. The production data has a specific nature – time series data, successive nature – which must be taken into account when selecting corresponding machine learning algorithm. Simple algorithms including linear regression, support vector machine, traditional neural networks are not suited for this task since shuffling takes place during the training which loses data's successive nature. Though, application of long-short term memory necessitates, preservation of successive nature of the data set since each data point is somewhat related to the point coming just before it and several before.

Long-short term memory appears to be a very suitable algorithm, and one of its strengths is to handle dataset in the form of time series. Long-short term memory involves multiple gates that are used to input the data into the network, forget unnecessary data and output the next time step. They are followingly called input, forget and output gate. These gates in mathematical terms carry out series of matrix and vector operations and try to minimize loss function, thereby, increase model's ability to make accurate time series prediction. The algorithms' ability to minimize the error in the prediction is based on famous backpropagation through time algorithm.

In this study, our primary objective was to use machine learning algorithms, specifically LSTM to conduct long-term predictions for oil production spanning a three-year horizon. The methodology involved employing time series cross-validation with a rolling window size to identify the optimal combination of hyperparameters for the LSTM model. In general, hyperparameter optimization in machine learning models appears to be a central concept. Correct selection of the hyperparameter is important for avoiding overfitting and more importantly reducing error in the predictions. Multiple hyperparameters are present for long-short term memory machine learning algorithm. Among all, window size, number of units and epochs were selected to be tuned. According to the results of the hyperparameter tuning it was found that when all these parameters are taken more, the least error is obtained in that case. Bigger window size allows the model to take a bigger horizon into consideration and make inferences about it. When it comes to tuning algorithm itself, grid search algorithm was selected. The algorithm defines a space of various numbers for each parameter, in this case, for window size, number of epochs, and number of units and tries to select the combination which gives the lowest possible error. Except for grid search algorithm, there are multiple other algorithms available each having a lesser run time. Since number of parameters considered in our case was not so huge, a selection was made in the direction of grid search algorithm. For validation, cross-validation with an expanding window size was selected.

Traditional cross-validation techniques are not suitable for this purpose since they might shuffle the data, and in our case, it is not preferred as the data loses its sequential nature. That's why selection was made in the direction of move forward cross-validation, which retains the sequential nature of the data. The algorithm performs by adding the data point from each successive point and comparing it to the prediction. Once lowest error is achieved for a given subset of hyperparameters, the selection is made in this direction. The results of this validation also show effect of continuous update of data set on model's ability to make prediction in a single stepwise manner.

Subsequently, the LSTM model was applied for both single-step and multi-step predictions in a univariate manner by using only production data. Three different cases were considered in the case of LSTM model. Firstly, univariate single stepwise prediction was made in a traditional manner. The predictions for this method closely matched the real data. The only advantage here appears to be the fact that during the training the data points are taken from the test data set instead of using predicted points. This was modified in the second case which was about making predictions in a recursive manner. Recursive prediction means that for making predictions for successive points always the predicted values are taken into consideration. New predictions are made based on previously predicted values. This is important as it adds a new aspect to the application of the recursive prediction by long-short term memory machine learning algorithm in oil production prediction for extended timespan. As a final experiment, multi-step long-short term memory machine learning algorithm was employed. Unlike the previous two, the predictions here are made in a multi-step manner, not in a single step. In terms of applicability, the former and the last techniques are more stable.

CONCLUSION

The current research mainly focuses on the comparison of predictive abilities of machine learning algorithms and reservoir simulation models in production prediction. In the context of machine learning, long-short term memory machine learning model is selected to be applied in the study. For building up reservoir simulation model, on the other hand, the software so called tNavigator has been used. Before application of each technique pre-processing was carried out on the dataset to understand underlying details in the successive data points.

Our findings revealed a noteworthy contrast in prediction accuracy among different models. The univariate single-step bidirectional LSTM demonstrated significantly lower error rates compared to both the recursive single-step prediction model and the multi-step prediction model. However, it's crucial to understand that the univariate single-step bidirectional LSTM model has practical limitations. During prediction, input and output pairs are generated using the test data, intruding into the prediction time interval. This aspect should be considered when evaluating its applicability in real-world scenarios. On the other hand, the recursive model exhibited a tendency to underestimate values over the three-year period. This characteristic implies a more risk-averse nature, making it a potentially reliable choice for informing future decisions within the company. Understanding the strengths and limitations of each model is vital for selecting an approach that aligns with the specific requirements and constraints of the oil production context.

For building reservoir simulation model, tNavigator was used and prediction was made for the same time span. Error comparison was carried out for both long-short term memory machine learning algorithm and reservoir simulation model. Unlike, long-short term memory machine learning algorithm the reservoir simulation model is unlikely to capture the trends in the data and is more likely to follow a slowly declining trend. Stated differently, patterns and cycles in the production data does not mean a lot for reservoir simulation model but it follows the pressure response of the system and give predictions accordingly.

With all reservoir simulation models, the procedure starts with the initial collection – data initialization – of the input data which includes the historical input rate data, bottom hole pressure values, and permeability curves. Once, this step was completed for our case, history matching step was initiated. The main purpose here was to make sure that the model’s output for given parameters closely match the observed values and iteratively change governing parameters to make sure that the best match is obtained in this way. Once all these steps were finalized the model developed was used to make predictions for a given time period.

Future direction could involve exploring multi-variate prediction models that encompass dynamic subsurface data. This extension could enhance the accuracy of predictions by incorporating additional relevant factors. As technology advances and data availability improves, integrating a more comprehensive set of variables into the predictive models may provide a more detailed understanding of the complex dynamics influencing oil production.

Even though lower error rate was obtained for long-short term memory machine learning model, application of this technique in production prediction is still debatable for many reasons. The degree of the randomness in the data is the core reason for it. Besides, although reservoir models take a lot of time to build and update regularly, they can be used to visualize multiple scenarios unlike long-short term memory machine learning algorithm. They are limited to inputting training data and making predictions in the forward direction only. No data can be included related to injection times planned.

In conclusion, our research contributes valuable insights into the application of machine learning, particularly LSTM, for long-term oil production prediction. The comparative analysis of different models sheds light on their respective strengths and weaknesses. As we move forward, the integration of multi-variate prediction models holds the potential to further refine and advance the accuracy of predictions in the dynamic field of oil production. Besides, the algorithm can be checked over gas wells to see effect of the model in that way. As mentioned previously, the gas wells contain high degree of cyclicity due to seasonality phenomena. As an additional information for long-short term memory machine learning algorithm, it is believed that it might have a positive impact on the predictions.

REFERENCES

1. Achong, L. B. (1961). Revised bean and performance formula for Lake Maracaibo wells. Shell Internal Report, Houston, TX: Shell Oil Co.
2. Ali, M., & Faraj, R. (2014). Data normalization and standardization: A technical report. https://doi.org/10.13140/RG.2.2.28948.04489
3. Baxendell, P. B. (1957). Bean performance-lake wells. Shell Internal Report, Houston, TX: Shell Oil.
4. Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2), 157-166. https://doi.org/10.1109/72.279181
5. Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13, 281–305. http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf
6. Berneti, S., & Shahbazian, M. (2011). An imperialist competitive algorithm artificial neural network method to predict oil flow rate of the wells. International Journal of Computer Applications. https://doi.org/10.5120/3137-4326
7. Berrar, D. (2018). Cross-validation. In Encyclopedia of Bioinformatics and Computational Biology (pp. 542-545). Elsevier. https://doi.org/10.1016/B978-0-12-809633-8.20349-X
8. Camacho, V., & Raghavan, R. (1989). Boundary-dominated flow in solution-gas drive reservoirs. SPE Reservoir Engineering, 4(4), 433-440. https://doi.org/10.2118/17535-PA
9. Cerqueira, V., Torgo, L., & Mozetič, I. (2020). Evaluating time series forecasting models: An empirical study on performance estimation methods. Machine Learning, 11, 1997-2028. https://doi.org/10.1007/s10994-020-05910-7
10. Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature. Geoscientific Model Development, 7(3), 1247-1250. https://doi.org/10.5194/gmd-7-1247-2014
11. Da Silva, L. C. F., Portella, R. C. M., & Emerick, A. A. (2007). Predictive data-mining technologies for oil-production prediction in petroleum reservoir. SPE Latin American and Caribbean Petroleum Engineering Conference, 107371. https://doi.org/10.2118/107371-MS

12. Elman, J. L. (1990). Finding structure in time. Cognitive Science, 14(2), 179-211. https://doi.org/10.1207/s15516709cog1402_1
13. Elmabrouk, S., Shirif, E., & Mayorga, R. (2010). A neural network approach to predict average reservoir pressure. 5th Technology of Oil and Gas Forum, Tripoli, Libya.
14. Elshafei, M., Khoukhi, A., & Abdulraheem, A. (2011). Neural network aided design of oil production units. IEEE Transactions on Neural Networks, 22(2), 202-213. https://doi.org/10.1109/TNN.2010.2102764
15. Falcone, G., & Alimonti, C. (2007). The challenges of multiphase flow metering: Today and beyond. In Proceedings of the ASME 2007 26th International Conference on Offshore Mechanics and Arctic Engineering, Volume 2: Structures, Safety and Reliability; Petroleum Technology Symposium (pp. 823-834). ASME. https://doi.org/10.1115/OMAE2007-29527
16. Falessi, D., Huang, J., Narayana, L., et al. (2020). On the need of preserving order of data when validating within-project defect classifiers. Empirical Software Engineering, 25(3), 2505-2530. https://doi.org/10.1007/s10664-020-09868-x
17. Gers, F. A., & Schmidhuber, J. (2001). LSTM recurrent networks learn simple context-free and context-sensitive languages. IEEE Transactions on Neural Networks, 12(6), 1333-1340. https://doi.org/10.1109/72.963769
18. Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. Neural Computation, 12(10), 2451-2471. https://doi.org/10.1162/089976600300015015
19. Gilbert, W. E. (1954). Flowing and gas-lift well performance. Drilling and Production Practice, 126-157.
20. Heaton, J. (2008). Introduction to neural networks for Java (2nd ed.). Heaton Research.
21. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735
22. Kee, E., Chong, J. J., Choong, Z. J., & Lau, M. (2023). A comparative analysis of cross-validation techniques for a smart and lean pick-and-place solution with deep learning. Electronics, 12(11), 2371. https://doi.org/10.3390/electronics12112371
23. Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In Proceedings of the 14th International Joint Conference on Artificial Intelligence (pp. 1137-1143). https://doi.org/10.1016/B978-0-12-809633-8.20349-X

24. Liashchynskyi, P., & Liashchynskyi, P. (2019). Grid search, random search, genetic algorithm: A big comparison for NAS. arXiv preprint arXiv:1912.06059. https://arxiv.org/abs/1912.06059
25. Little, M. A., Varoquaux, G., Saeb, S., Lonini, L., Jayaraman, A., Mohr, D. C., & Kording, K. P. (2017). Using and understanding cross-validation strategies for machine learning. https://doi.org/10.1093/gigascience/gix020
26. Liu, W., Liu, W. D., & Gu, J. (2020). Forecasting oil production using ensemble empirical model decomposition based long short-term memory neural network. Journal of Petroleum Science and Engineering, 189, 107013. https://doi.org/10.1016/j.petrol.2020.107013
27. Mirzaei-Paiaman, A. (2012). The application of artificial neural network. Energy Sources, 34. https://doi.org/10.1080/15567036.2010.492386
28. Mirzaei-Paiaman, A. (in press). A new empirical correlation for sonic simultaneous flow of oil and gas through wellhead chokes for. Journal of Energy Sources. https://doi.org/10.1080/15567036.2010.492386
29. Muhammad Ali, P., & Faraj, R. (2014). Data normalization and standardization: A technical report. https://doi.org/10.13140/RG.2.2.28948.04489
30. Negash, B., & Yaw, A. (2020). Artificial neural network-based production forecasting for a hydrocarbon reservoir under water injection. Petroleum Exploration and Development, 47(3), 383-392. https://doi.org/10.1016/S1876-3804(20)60055-6
31. Nguyen, H. N., & Chan, C. W. (2005). Applications of data analysis techniques for oil production prediction. Journal of Petroleum Science and Engineering, 48(3-4), 142-150. https://doi.org/10.1016/j.petrol.2005.07.007
32. Park, Y. (2022). Concise logarithmic loss function for robust training of anomaly detection model. arXiv preprint arXiv:2201.05748. https://arxiv.org/abs/2201.05748
33. Radzi, S. F. M., Karim, M. K. A., Saripan, M. I., Rahman, M. A. A., & Isa, M. R. M. (2010). Oil production prediction using machine learning. Proceedings of the 2010 International Conference on Intelligent and Advanced Systems. https://doi.org/10.1109/ICIAS.2010.5716133
34. Rohini, G., & Madhavi, M. (2017). Predictive analysis of well performance parameters using neural networks. Journal of Computational Science, 22, 218-225. https://doi.org/10.1016/j.jocs.2017.02.012

35. Rossi, F., & Smith, M. (2004). Data preparation for oil production forecasting. In Proceedings of the 2004 IEEE International Conference on Data Mining (pp. 449-452). https://doi.org/10.1109/ICDM.2004.10016
36. Rousseeuw, P. J., & Hubert, M. (2018). Anomaly detection by robust statistics. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(2), e1238. https://doi.org/10.1002/widm.1238
37. Rudi, D., Fukuchi, K., & Ihara, T. (2018). Machine learning prediction and empirical study for oil production. IEEE Transactions on Industrial Informatics, 14(12), 5342-5353. https://doi.org/10.1109/TII.2018.2868415
38. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536. https://doi.org/10.1038/323533a0
39. Sanayei, M., & Shadrokh, F. (2011). Dynamic system identification using artificial neural network for oil production optimization. Journal of Petroleum Science and Engineering, 78(1), 75-82. https://doi.org/10.1016/j.petrol.2011.05.011
40. Saraç, M., & Duran, O. (2008). Comparative study on machine learning algorithms for oil production prediction. Proceedings of the 2008 IEEE World Congress on Computational Intelligence (pp. 436-442). https://doi.org/10.1109/IJCNN.2008.4633792
41. Schruben, L. W., & Lehmann, E. L. (1975). Testing for two-way interaction in a generalized linear model. Biometrika, 62(3), 539-544. https://doi.org/10.1093/biomet/62.3.539
42. Shen, C., & Abdulraheem, A. (2020). Deep learning prediction of oil well performance using LSTM networks. Petroleum Exploration and Development, 47(4), 777-786. https://doi.org/10.1016/S1876-3804(20)60155-8
43. Srinivasan, D., & Somasundaram, P. (2010). Application of adaptive neuro-fuzzy inference system for well test data analysis. Journal of Petroleum Science and Engineering, 75(1-2), 125-132. https://doi.org/10.1016/j.petrol.2010.11.004
44. Stinchcombe, M., & White, H. (1989). Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions. In Proceedings of the 1989 International Joint Conference on Neural Networks (pp. 65-68). https://doi.org/10.1109/IJCNN.1989.118638
45. Tan, Y., & Kumar, A. (2009). Prediction of well productivity using support vector regression. Journal of Petroleum Science and Engineering, 67(3-4), 142-150. https://doi.org/10.1016/j.petrol.2009.05.004

46. Tay, K. G., Cheung, W. Y., Lee, S. L., & Foong, S. Y. (2021). Oil production forecasting using hybrid deep learning model. Energy Reports, 7, 2467-2478. https://doi.org/10.1016/j.egyr.2021.06.007
47. Thissen, U., Üstün, B., Melssen, W. J., & Buydens, L. M. (2004). Multivariate calibration with least-squares support vector machines. Analytica Chimica Acta, 544(1-2), 279-287. https://doi.org/10.1016/j.aca.2005.02.019
48. Tobin, K. W., & Gedeon, T. D. (2000). Comparison of cross-validation techniques for determining stopping points in neural network training. Proceedings of the International Joint Conference on Neural Networks (pp. 1297-1301). https://doi.org/10.1109/IJCNN.2000.857822
49. Vapnik, V. N. (1995). The nature of statistical learning theory. Springer. https://doi.org/10.1007/978-1-4757-2440-0
50. Walker, M., & Duncan, G. (1967). Estimating the probability of an event occurring in a specific time interval. Journal of the American Statistical Association, 62(320), 1171-1184. https://doi.org/10.1080/01621459.1967.10500902

NOMENCLATURE

1. LSTM: Long Short-Term Memory
2. RNN: Recurrent Neural Network
3. AI: Artificial Intelligence
4. ML: Machine Learning
5. DL: Deep Learning
6. NN: Neural Network
7. IoT: Internet of Things
8. CPU: Central Processing Unit
9. GPU: Graphics Processing Unit
10. API: Application Programming Interface
11. RMSE: Root Mean Square Error
12. MAE: Mean Absolute Error
13. MAPE: Mean Absolute Percentage Error
14. MSE: Mean Squared Error
15. GRU: Gated Recurrent Unit
16. SGD: Stochastic Gradient Descent
17. BP: Backpropagation
18. CNN: Convolutional Neural Network
19. GPU: General Processing Unit
20. RAM: Random Access Memory

APPENDIX

Table 1. Root Mean Squared Error (RMSE) estimation for window size of 3.

	3,32,25	3,32,50	3,32,100	3,64,25	3,64,50	3,64,100	3,128,25	3,128,50	3,128,100
0	37.82	7.80	15.92	27.75	26.47	34.82	12.26	12.67	2.92
1	15.21	7.76	40.85	6.76	15.23	24.32	9.50	19.04	19.43
2	11.54	34.08	27.96	17.20	24.83	24.90	14.26	16.52	43.57
3	41.47	70.19	34.24	55.17	58.26	46.80	60.26	31.11	43.07
4	26.77	34.44	66.88	38.26	81.30	72.77	42.54	67.06	71.92
5	46.90	35.54	38.46	52.67	53.60	40.61	31.18	40.86	44.17
6	42.34	62.11	48.04	44.74	55.19	49.50	50.63	52.82	52.86
7	23.72	42.64	30.01	19.33	31.10	20.47	29.13	26.02	34.98
8	56.95	34.01	35.50	35.38	38.74	34.36	28.90	42.74	29.68
9	66.10	71.99	67.66	72.81	78.75	77.97	65.28	69.32	65.14
10	78.70	74.54	72.41	75.43	60.08	60.10	78.50	69.05	70.06
11	81.67	71.70	62.86	76.28	72.75	60.64	69.42	78.08	71.15
12	41.08	46.88	47.74	33.79	55.14	51.42	33.67	46.61	65.86
13	40.08	53.00	57.65	47.02	66.49	65.03	45.77	67.13	36.96
14	53.03	47.74	45.01	54.03	47.64	32.40	45.61	49.42	38.35
15	45.72	46.21	48.26	46.27	49.50	55.89	42.36	36.35	50.77
16	52.36	68.11	46.55	44.49	58.89	43.33	49.08	44.15	56.39
17	65.70	74.33	76.72	72.75	75.79	72.54	64.46	65.42	66.54
18	67.11	69.47	65.17	67.04	64.63	71.05	74.11	60.32	52.13
19	53.01	58.94	71.53	51.46	62.08	73.86	48.61	58.84	71.94
20	55.58	60.81	64.36	58.39	74.35	62.89	66.70	70.69	63.00
21	56.14	48.27	51.05	52.23	42.75	50.31	52.82	55.09	48.87
22	41.54	44.38	51.67	49.70	48.43	57.46	51.62	50.14	56.40
23	47.55	52.64	50.63	55.35	51.52	54.49	52.96	53.35	46.68
24	37.86	58.61	46.79	41.40	39.14	23.42	35.36	46.11	39.70
25	40.49	43.11	40.35	47.01	36.59	30.21	45.21	41.49	47.34
26	48.02	57.35	30.37	57.39	42.79	33.14	64.31	40.35	24.55
27	52.88	56.30	38.42	50.45	48.64	44.29	49.48	45.00	45.92
28	66.60	51.37	57.63	64.27	55.16	58.65	63.84	66.79	61.28
29	74.39	68.85	63.80	73.57	74.78	72.36	68.85	74.88	68.95
30	76.41	52.55	64.50	68.54	72.28	64.23	83.89	62.49	61.15
31	53.84	50.75	33.44	39.59	33.17	51.00	55.92	31.68	80.69
32	60.20	64.86	66.24	65.60	57.17	79.33	75.52	65.02	74.56
33	67.71	66.81	65.01	69.25	59.72	74.30	66.10	55.51	76.02
34	55.71	62.05	66.80	55.15	67.99	67.46	66.89	60.12	68.97
35	42.84	65.63	66.21	56.38	59.27	66.17	61.50	63.16	72.91

Table 2. Root Mean Squared Error (RMSE) estimation for window size of 12.

	12,32,25	12,64,25	12,128,25	12,32,50	12,32,100	12,64,50	12,64,100	12,128,50	12,128,100
0	29.94	41.60	14.49	11.03	7.96	8.11	12.31	24.40	1.39
1	18.80	17.61	12.12	6.08	3.06	13.61	1.24	4.81	8.24
2	33.55	26.08	29.00	14.63	14.37	14.59	25.82	3.06	4.59
3	40.43	41.89	46.59	12.82	25.32	29.77	1.72	0.63	0.09
4	1.45	5.40	4.92	9.35	13.32	5.52	9.52	6.50	0.79
5	27.06	7.91	1.38	7.08	7.28	10.21	5.24	4.95	2.32
6	14.60	1.56	18.65	0.80	5.29	7.94	1.12	6.14	2.29
7	2.16	2.27	0.01	5.40	12.49	5.99	10.00	6.24	1.96
8	24.89	1.46	0.57	12.08	3.50	0.59	2.58	3.93	1.69
9	6.02	10.40	17.43	23.95	5.79	1.12	6.69	5.07	0.94
10	6.00	10.14	11.64	6.40	3.20	3.87	2.54	4.93	0.32
11	38.47	17.18	7.49	4.26	4.19	0.92	0.69	8.74	2.37
12	3.49	5.65	1.80	4.26	1.79	7.99	1.64	6.01	1.88
13	5.46	1.10	3.29	11.31	0.64	0.64	2.67	0.67	0.52
14	12.45	0.59	3.98	2.57	0.44	0.56	1.05	3.59	0.75
15	25.80	4.03	7.08	7.22	2.06	0.02	3.39	4.27	6.94
16	13.29	5.83	9.81	0.97	1.03	6.17	6.94	4.99	1.48
17	7.66	11.01	3.92	1.95	0.14	10.66	1.58	0.26	2.32
18	17.43	5.49	3.30	1.50	0.26	5.14	1.28	2.28	0.51
19	4.88	5.33	10.68	3.75	0.64	5.75	0.49	1.57	0.77
20	9.00	0.87	2.15	6.24	0.12	2.90	1.08	1.57	0.28
21	13.58	5.96	1.16	0.43	2.62	0.74	3.05	5.33	1.28
22	7.09	2.49	1.83	3.94	2.03	1.26	1.45	1.38	1.08
23	35.98	9.40	7.68	0.97	1.10	3.05	2.30	3.05	1.57
24	5.39	3.19	33.95	1.64	1.41	2.04	1.50	3.37	0.21
25	5.72	10.22	6.12	4.23	1.00	0.38	1.18	3.37	0.02
26	3.81	1.40	0.98	1.97	3.32	1.87	0.46	0.65	1.55
27	10.42	4.31	1.75	0.38	1.11	5.24	0.02	0.36	0.20
28	7.57	2.93	3.48	0.37	4.27	3.63	1.05	0.05	0.49
29	2.95	6.68	1.41	2.54	0.85	1.92	0.23	2.93	4.84
30	6.95	6.39	6.38	5.72	1.62	5.84	0.21	1.61	2.56
31	1.51	0.57	1.73	0.21	1.31	5.69	0.43	2.66	2.02
32	3.97	1.54	0.98	1.37	1.70	0.48	0.11	2.18	0.14
33	3.41	0.77	3.44	9.21	0.78	6.29	2.54	0.79	1.04
34	8.15	1.51	0.77	2.91	1.80	1.12	0.43	2.12	0.19
35	7.57	3.51	3.97	2.73	0.41	4.28	0.92	1.30	4.38

Table 3. Root Mean Squared Error (RMSE) estimation for window size of 6.

	6,32,25	6,64,25	6,128,25	6,32,50	6,32,100	6,64,50	6,64,100	6,128,50	6,128,100
0	25.26	33.78	25.57	33.78	15.41	19.98	10.07	18.09	15.20
1	33.99	30.69	26.94	12.47	0.10	14.17	15.64	9.14	0.77
2	27.39	52.09	23.67	50.34	43.99	52.68	35.60	42.31	41.45
3	63.22	74.32	39.88	34.89	26.21	43.15	56.25	0.56	27.09
4	26.96	39.12	44.56	51.66	41.79	49.51	25.98	43.11	29.80
5	29.41	24.82	26.21	31.56	24.83	35.59	24.45	17.38	34.39
6	43.00	54.92	33.14	28.43	-0.37	10.05	-1.21	-3.81	-5.30
7	35.20	42.88	50.32	29.97	42.48	36.19	38.94	50.36	37.82
8	35.68	32.88	61.97	59.77	57.56	51.20	67.93	57.79	67.07
9	63.39	78.11	54.09	76.18	83.64	90.54	90.67	86.57	86.79
10	76.60	45.36	82.50	72.19	70.64	66.79	76.66	72.06	68.86
11	59.65	35.64	51.04	29.93	36.63	33.11	36.70	43.53	45.77
12	30.75	37.46	36.46	40.20	41.01	41.46	40.77	45.27	42.99
13	53.91	25.91	39.64	40.91	31.24	27.61	32.27	39.85	31.99
14	63.00	28.75	32.09	41.26	30.33	34.72	28.20	36.82	28.48
15	54.05	53.60	64.28	70.60	53.42	54.17	56.96	51.30	58.97
16	45.72	76.76	51.35	63.13	67.31	64.10	67.27	50.65	67.67
17	56.91	68.98	78.98	60.57	73.25	76.72	66.61	76.93	70.38
18	72.90	54.03	34.59	55.05	52.46	50.35	46.67	49.14	53.18
19	51.70	58.71	49.99	56.58	53.57	56.84	52.84	52.52	55.30
20	49.00	46.13	38.86	42.99	41.40	38.22	44.06	44.94	42.39
21	47.83	41.18	43.85	57.27	41.44	37.39	42.46	44.58	42.57
22	62.99	67.57	67.65	65.01	57.89	63.43	62.41	53.25	58.10
23	52.95	46.18	14.22	13.46	25.32	18.90	19.65	27.21	17.09
24	53.29	46.53	27.26	45.28	37.29	34.76	40.89	39.19	31.19
25	61.38	50.86	50.65	60.07	44.84	50.44	44.02	52.16	44.77
26	43.85	62.31	49.02	52.97	42.84	46.44	55.12	45.37	46.15
27	63.83	65.36	44.15	64.56	75.09	68.64	67.59	80.59	76.02
28	86.31	77.60	90.57	81.16	83.61	81.78	89.02	89.28	90.61
29	85.29	88.95	90.61	88.95	95.14	100.04	95.01	87.67	95.64
30	52.06	52.87	54.78	59.49	62.01	59.54	57.54	52.29	58.51
31	73.69	69.71	70.43	72.25	75.03	75.73	69.07	75.82	74.83
32	67.01	75.87	72.43	69.30	77.58	70.35	75.82	71.52	73.17
33	76.80	62.52	61.01	56.84	54.45	53.60	60.91	57.09	55.71
34	56.65	43.57	46.75	43.93	49.75	42.73	48.37	48.02	51.38
35	17.57	14.22	15.63	18.81	17.83	12.96	18.86	9.50	14.18