algorithms

MDPI

Article

Machine Learning Analysis Using the Black Oil Model and Parallel Algorithms in Oil Recovery Forecasting

Bazargul Matkerim [1,2,3], Aksultan Mukhanbet [2,3], Nurislam Kassymbek [2,3,*], Beimbet Daribayev [1,2], Maksat Mustafin [2,3] and Timur Imankulov [1,2,3]

[1] National Engineering Academy of the Republic of Kazakhstan, Almaty 050010, Kazakhstan; bazargul.matkerim@kaznu.edu.kz (B.M.); beimbet.daribayev@kaznu.kz (B.D.); timur.imankulov@kaznu.kz (T.I.)
[2] Department of Computer Science, Al-Farabi Kazakh National University, Almaty 050040, Kazakhstan; mukhanbet_aksultan3@live.kaznu.kz (A.M.); mustafin.mb@gmail.com (M.M.)
[3] Joldasbekov Institute of Mechanics and Engineering, Almaty 050000, Kazakhstan
* Correspondence: nurislam.kassymbek@kaznu.edu.kz

Abstract: The accurate forecasting of oil recovery factors is crucial for the effective management and optimization of oil production processes. This study explores the application of machine learning methods, specifically focusing on parallel algorithms, to enhance traditional reservoir simulation frameworks using black oil models. This research involves four main steps: collecting a synthetic dataset, preprocessing it, modeling and predicting the oil recovery factors with various machine learning techniques, and evaluating the model's performance. The analysis was carried out on a synthetic dataset containing parameters such as porosity, pressure, and the viscosity of oil and gas. By utilizing parallel computing, particularly GPUs, this study demonstrates significant improvements in processing efficiency and prediction accuracy. While maintaining the value of the R^2 metric in the range of 0.97, using data parallelism sped up the learning process by, at best, 10.54 times. Neural network training was accelerated almost 8 times when running on a GPU. These findings underscore the potential of parallel machine learning algorithms to revolutionize the decision-making processes in reservoir management, offering faster and more precise predictive tools. This work not only contributes to computational sciences and reservoir engineering but also opens new avenues for the integration of advanced machine learning and parallel computing methods in optimizing oil recovery.

Keywords: distributed machine learning; HPC; artificial intelligence; cuML; enhanced oil recovery

Citation: Matkerim, B.; Mukhanbet, A.; Kassymbek, N.; Daribayev, B.; Mustafin, M.; Imankulov, T. Machine Learning Analysis Using the Black Oil Model and Parallel Algorithms in Oil Recovery Forecasting. *Algorithms* **2024**, *17*, 354. https://doi.org/10.3390/a17080354

Academic Editor: Steven Prestwich

Received: 16 July 2024
Revised: 3 August 2024
Accepted: 8 August 2024
Published: 14 August 2024

Correction Statement: This article has been republished with a minor change. The change does not affect the scientific content of the article and further details are available within the backmatter of the website version of this article.

1. Introduction

The black oil model is a simplified yet robust approach used in reservoir simulations to represent the phase behavior and flow of oil, water, and gas in subsurface formations. The black oil model plays a crucial role in oil production optimization by simulating reservoir behavior and aiding in decision-making processes [1]. It is widely used in the industry for predicting oil recovery and guiding management strategies [2]. The model's ability to accurately capture natural depletion scenarios makes it a valuable tool for reservoir management. The black oil model is essential for integrating optimization methods with simulation and prediction techniques, enabling the industry to make the optimal decisions regarding oil production [3]. The black oil model primarily simplifies the representation of reservoir hydrocarbons into three components—oil, gas, and water—making it a practical choice for many applications in reservoir engineering and simulation. This model is particularly favored for its computational efficiency and the reduced complexity it offers compared to compositional models, which require the detailed characterization of all hydrocarbon components and their interactions [4–6]. The black oil model is used to simulate the three-phase flow and predict the behavior of the oil, gas, and water phases under various

reservoir conditions [7], and its application ranges from primary and secondary recovery processes to enhanced oil recovery (EOR) operations. Although the black oil model has limitations in accurately capturing the phase behavior of more complex fluid systems, such as gas condensate reservoirs, it remains a cornerstone in reservoir simulations due to its ability to provide conservative calculations for compressible and incompressible multiphase flows [8]. Recent advancements have extended the capabilities of the black oil model to better represent the physical properties of oil–gas mixtures, such as density and viscosity, through the dynamic black oil properties that depend on the fraction of CO_2 in the cell, enhancing its applicability to CO_2 EOR operations [9]. Additionally, efforts to incorporate the effects of large gas–oil capillary pressures and compositional changes in gas injection scenarios have further improved the model's accuracy and robustness [10]. Moreover, the development of sophisticated black oil-based multi-component models for polymer flooding and the integration of black oil data with common equation of state (EOS) models for simulating fluid production in multi-reservoir systems with a common surface network demonstrate the model's adaptability and ongoing relevance in addressing contemporary challenges in reservoir management [11,12].

The traditional method to solve the black oil model in reservoir simulations is the fully implicit method, which is widely used due to its robustness in handling the complex interactions between different phases and components in the reservoir. This method involves the repeated linearization of large nonlinear systems, which often results in ill-conditioned linear systems that are computationally expensive to solve [5,13,14]. The fully implicit approach requires solving a coupled system of equations for pressure and saturation simultaneously, which can be challenging due to the nonlinearity and need for accurate phase behavior modeling, especially during phase transitions [15]. One significant drawback of the fully implicit method is its tendency to produce discontinuities in the discrete system when the phase transitions occur, leading to oscillations or even failure of the Newton iterations used for solving the nonlinear equations [15]. Additionally, the method's computational cost is high, because it necessitates solving large, sparse linear systems repeatedly, which can be particularly burdensome for large-scale reservoir models with high geological heterogeneity [16]. As reservoir complexity grows and data collection expands exponentially, there is an urgent need for more efficient computational techniques that not only accelerate processes but also enhance the simulation accuracy.

Machine learning (ML) has become an indispensable tool in earth sciences and represents a significant advancement, providing robust tools for the efficient and precise exploration of mineral resources. These methods are employed for tasks such as dimensionality reduction, classification, regression, and clustering, which are crucial for accurate mineral potential mapping [17,18]. ML has emerged as a fundamental instrument in the field of oil reservoir simulations, providing a wide array of applications that improve both efficiency and precision. One primary application is in the estimation of original oil in place (OOIP), where ML algorithms, particularly artificial neural networks (ANNs), are used to predict reserves with high accuracy, even when the data are insufficient [19]. Additionally, ML techniques are employed in reservoir production optimization, where advanced algorithms like ANNs optimize the cumulative oil recovery by evaluating various field development scenarios and well placements [20]. In geomechanical modeling, ML predicts the rock mechanical properties using conventional well logging data, thus reducing the need for costly dipole sonic logs [21]. ML also aids in optimizing the polymer injection processes by predicting the oil recovery factor using regression algorithms and ANNs, which are trained on extensive synthetic datasets [22]. In underground natural gas storage, ML models optimize parameters such as well positioning to maximize the gas delivery while minimizing the CO_2 production [23]. Surrogate proxy models created using ML techniques like XGBoost and MLP are used to estimate the net present value (NPV) of the reservoirs under various operating conditions, thereby aiding in decision-making for well placements and production strategies [24]. Furthermore, ML accelerates the reservoir simulation processes by providing fast and competent results that mimic traditional simulators,

thus reducing the computational time and costs [25]. The integration of ML with numerical methods enhances the precision of partial differential equation discretization in reservoir simulations, leading to rapid convergence and high computational efficiency [26]. ML also automates and accelerates reservoir characterization, production forecasting, and well test interpretation, making these processes more efficient and cost-effective [27]. Lastly, deep learning models with neural operators in Fourier space significantly reduce the computational time required for direct numerical simulations, enabling faster and more accurate predictions for reservoir management [28]. The potential of ML in reservoir management includes property prediction, oil recovery factor prediction, production optimization, enhanced oil recovery testing, and performance metrics analysis, demonstrating its versatility and effectiveness in the oil and gas industry. However, the application of parallel machine learning algorithms—those utilizing multiple processors simultaneously for enhanced efficiency—to tackle the black oil problem remains underexplored. This gap is significant, given parallel computing's potential to significantly reduce processing times and manage large-scale data more effectively, particularly in reservoir simulations. There are numerous parallelization methods for modern computing systems, including MPI, OpenMP, TBB (Threading Building Blocks), and LMK (Linux Kernel Module) for CPUs, as well as OpenCL and CUDA for GPUs. However, for our purposes, it was decided to focus on two well-established technologies: MPI and the RAPIDS cuML library based on Nvidia CUDA [29–35]. These tools were chosen for their high performance and ease of integration into our machine learning workflow. They allow for the efficient handling of large data volumes and accelerate computations, which is critical for our objectives.

This study aims to bridge the gap by leveraging parallel machine learning algorithms to enhance traditional reservoir simulation frameworks, specifically focusing on the black oil model dataset. This research focuses on exploring specific parallel machine learning algorithms, including random forest (RF), artificial neural networks (NNs), polynomial regression (PR), gradient boosting (GB), and decision trees (DTs) to predict oil recovery factors. By exploring the capabilities and advantages of parallel machine learning algorithms, this paper seeks to make significant contributions to computational sciences and reservoir engineering. It is anticipated that the findings will accelerate and optimize the decision-making processes within the industry by offering faster, more accurate predictive tools, ultimately facilitating more efficient reservoir management.

2. Methodology

In this study, we employed a structured approach to integrate parallel machine learning algorithms into traditional reservoir simulation frameworks. The methodology is designed to harness the computational power of parallel algorithms to enhance the prediction accuracy and efficiency.

Figure 1 shows the research process to predict the oil recovery factor based on parallel machine learning algorithms, which includes four main stages. The first stage is the collection of a synthetic dataset from the mathematical model. In the second stage, the preliminary processing and analysis of the resulting dataset is carried out. The third stage then involves parallel modeling and predicting the oil recovery factor using machine learning techniques. Finally, in the fourth step, the performance of the model is evaluated to determine its effectiveness.

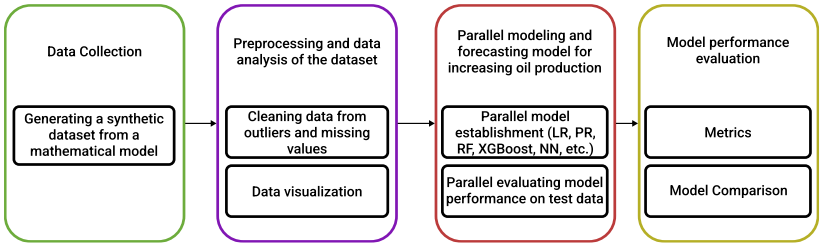

Figure 1. Flowchart for predicting oil recovery factor using machine learning.

2.1. Data Collecting

The black oil model, which is a cornerstone for synthesizing realistic simulation datasets, is used. A comprehensive synthetic dataset is generated through a numerical solution of this model, forming the foundation for the subsequent machine learning analysis. This ensures that the data embody realistic reservoir characteristics and are finely tuned for high fidelity in forecasting the oil recovery through this process.

The black oil model is a fundamental tool used in reservoir engineering to simulate the behavior of an oil reservoir. This model simplifies the complex fluid dynamics within a reservoir into a more manageable framework while still capturing the essential characteristics of the system. More details about the model can be found in [36].

Table 1 shows the input parameters of the black oil model used in the experiments.

Table 1. Parameters of model.

Parameters	Description	Value
k	Absolute permeability	0.001
μ_w	Water viscosity	0.09
μ_o	Oil viscosity	0.3
μ_g	Gas viscosity	0.01
$\varnothing$	Porosity	0.25
P_{inj}	Injection pressure	0.5
P_{init}	Initial pressure	0.3
P_{prod}	Produced pressure	0.1
S_{w_inj}	Water injection saturation	1.0
S_{w_init}	Water initial saturation	0.2
S_{o_init}	Oil initial saturation	0.7
S_{g_init}	Gas initial saturation	0.1
B_w	Water formation volume factor	1
B_o	Oil formation volume factor	1.2
B_g	Gas formation volume factor	0.5

Using various variations of the values of these parameters, a synthetic dataset was collected. The next sub-chapter explains the details of this dataset. Before diving into the data preprocessing details, it is important to note that each row in our dataset represents a distinct scenario within the reservoir. These scenarios are generated by varying key parameters such as the oil viscosity, gas viscosity, porosity, pressure, and saturation levels. This approach allows us to capture a wide range of conditions and behaviors within the reservoir, providing a comprehensive basis for training and testing our machine learning models. While this dataset does not represent the entire field in real time, it includes a broad spectrum of scenarios, allowing the models to learn from diverse conditions.

2.2. Data Preprocessing

This dataset includes 369,600 data points collected using the black oil mathematical model simulations. It contains parameters such as porosity, pressure, gas viscosity, oil viscosity, and average oil saturation. The initial dataset, as in Table 2, contains model parameters such as oil viscosity (VISC_OIL), gas viscosity (VISC_GAS), porosity (POROSITY), pressure (pressure), water saturation (Sw_avg), oil saturation (So avg), and oil recovery factor (oil_recovery_factor) values. These data are used to train the model, which is then used to predict the oil recovery factor.

Table 2. Dataset obtained using the black oil model.

k	Visc_oil	Visc_gas	Porosity	Pressure_avg	Oil_Recover_ Factor	Gas_Recovery_ Factor	Sw_avg	So_avg	Sg_avg	Pressure
0.0001	0.3	0.01	0.2	0.305602	0.305602	0.03125	0.0312501	0.225	0.678125	0.096875
0.0001	0.3	0.01	0.2	0.30562	0.30562	0.0316641	0.030819	0.225247	0.677835	0.096918
0.0001	0.3	0.01	0.2	0.305651	0.305651	0.032026	0.0304855	0.225467	0.677582	0.096951
0.0001	0.3	0.01	0.2	0.305689	0.305689	0.0323881	0.0301498	0.225687	0.677328	0.096985
0.0001	0.3	0.01	0.2	0.30573	0.30573	0.0327505	0.0298122	0.225907	0.677075	0.097018
0.0001	0.3	0.01	0.2	0.305774	0.305774	0.0331132	0.0294726	0.226126	0.676821	0.097052
0.0001	0.3	0.01	0.2	0.305822	0.305822	0.0334762	0.0291313	0.226346	0.676567	0.097086
0.0001	0.3	0.01	0.2	0.305874	0.305874	0.0338395	0.0287883	0.226566	0.676312	0.097121
0.0001	0.3	0.01	0.2	0.305928	0.305928	0.0342031	0.0284438	0.226787	0.676058	0.097155
0.0001	0.3	0.01	0.2	0.305986	0.305986	0.0345671	0.028098	0.227007	0.675803	0.097190
0.0001	0.3	0.01	0.2	0.306046	0.306046	0.0349314	0.0277509	0.227227	0.675548	0.097224
0.0001	0.3	0.01	0.2	0.30611	0.30611	0.035296	0.0274027	0.227447	0.675293	0.097259
0.0001	0.3	0.01	0.2	0.306176	0.306176	0.035661	0.0270535	0.227668	0.675037	0.097294
0.0001	0.3	0.01	0.2	0.306246	0.306246	0.0360263	0.0267035	0.227889	0.674782	0.097329
0.0001	0.3	0.01	0.2	0.306317	0.306317	0.036392	0.0263526	0.22811	0.674526	0.097364

There is a relevant correlation between the variables in the dataset. The correlation matrix in Figure 2 represents the relationships between the various parameters in the dataset. Each cell in the matrix indicates the correlation coefficient between the pairs of variables, where values close to 1 or −1 suggest strong positive or negative correlations, respectively, and values near 0 indicate a lack of correlation.

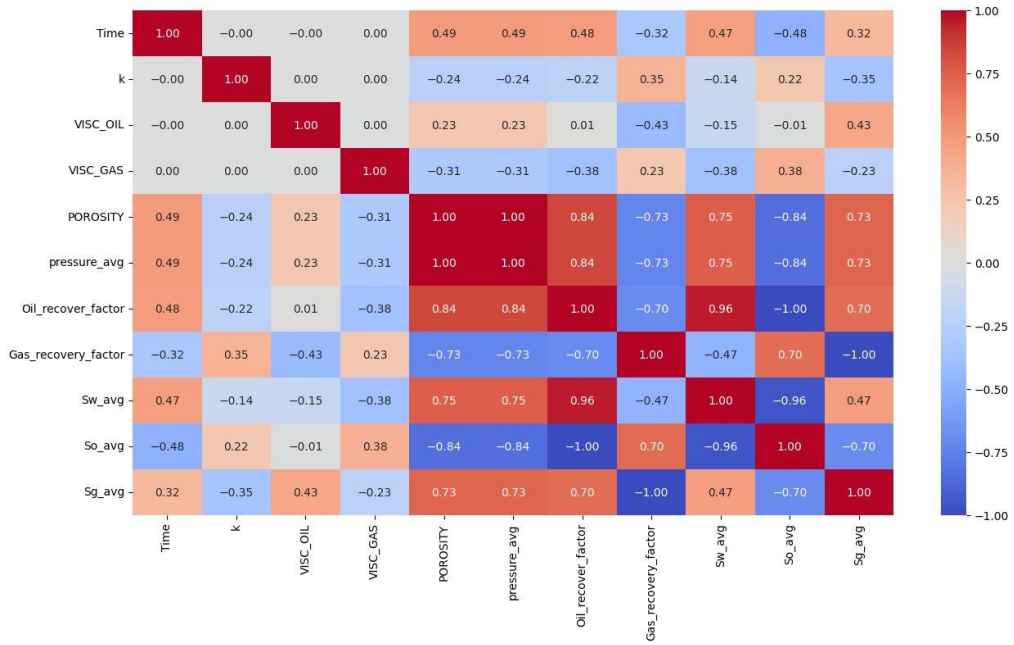

Figure 2. Correlation matrix.

There is a strong positive correlation between the pressure and both the oil recovery factor and porosity, with 0.84 and 1.00 Pearson correlation coefficients, respectively, which means that higher average reservoir pressures are associated with increased oil recovery and greater porosity.

The gas recovery factor exhibits a strong negative correlation (−0.73) with porosity and pressure, implying that in scenarios where porosity and pressure are high, the gas recovery factor tends to be lower. The average water saturation shows a near-perfect negative correlation (−0.96) with the average oil saturation, which is expected as increases in one typically result in decreases in the other within the reservoir.

The correlations between the viscosity of the oil and other parameters are generally weak, with the strongest negative correlation being −0.43 with the gas recovery

factor, suggesting the minimal influence of oil viscosity on the gas recovery under the conditions modeled.

The target variable, the oil recovery factor, exhibits strong correlations with several other features within the dataset, which is advantageous for the model training. This indicates a robust predictive foundation and increases the likelihood of developing an effective model that accurately forecasts the oil recovery based on these interrelated parameters.

In order to understand the models that will later be trained, it is important for the types of relationships exhibited between the variables to be understood. The scatterplots shown in Figure 3 help us to identify the relationships between multiple variables in the dataset.

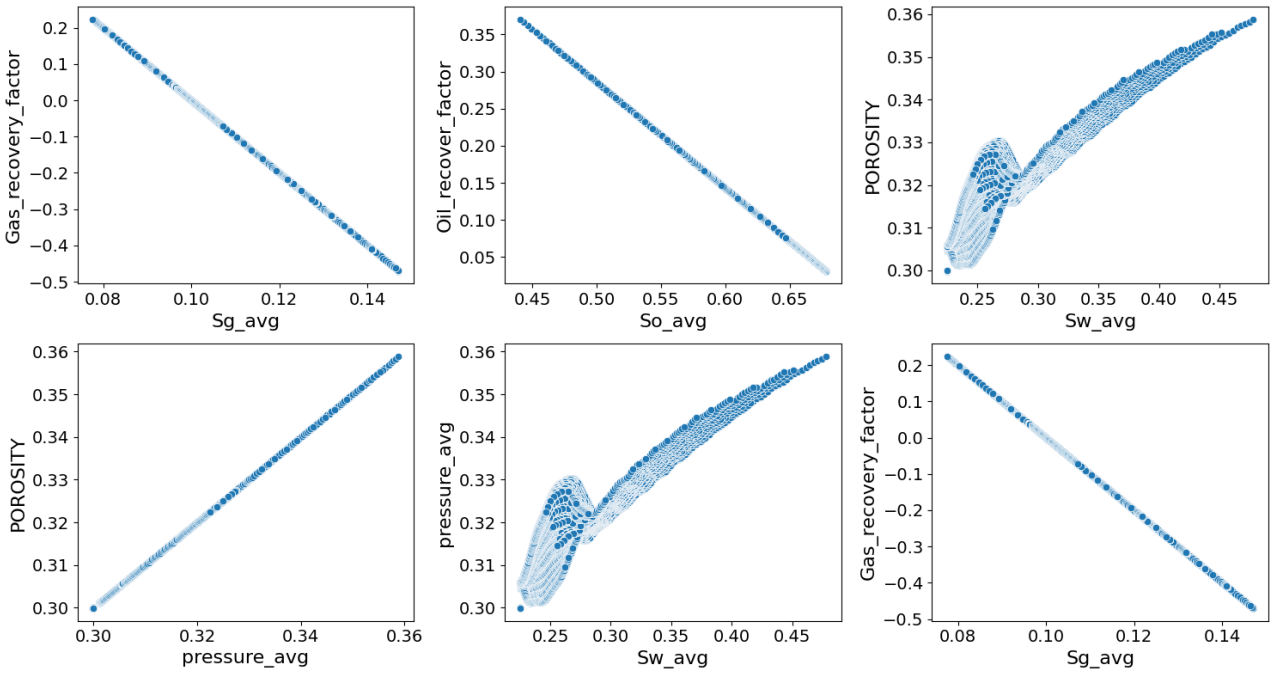

Figure 3. Dataset before adding noise.

From the scatterplots, some variables exhibit linear relationships. For instance, porosity has a positive linear relationship with pressure. Conversely, the oil recovery factor has a negative linear relationship with oil saturation and the gas recovery also has a negative linear relationship with gas saturation.

Noise Generation and Insertion

In this study, Laplace noise was incorporated into the simulated dataset to introduce stochastic variability, thus mirroring the inherent randomness and uncertainty present in natural reservoir properties. Laplace noise, also known as double exponential noise, is characterized by a probability density function that is exponentially decaying on either side of the mean (the location parameter).

The choice of Laplace noise was driven by its utility in scenarios where the data are expected to have heavier tails. This leads to a higher likelihood of observing extreme deviations from the mean. This feature is particularly advantageous in geological simulations where extreme values can represent rare but significant events, such as reservoir discontinuities or anomalous porosity and permeability zones. Additionally, the Laplace distribution's sharper peak enhances the model's sensitivity to subtle changes in the input parameters, providing a more robust framework for evaluating the impact of small fluctuations on the modeled reservoir properties.

The formula for the probability density function (PDF) of the Laplace distribution is given by the following:

$$f(x|u,b) = \frac{1}{2b} exp\left(-\frac{|x-\mu|}{b}\right), \quad (1)$$

where x represents the variable, μ is the location parameter, and b is the scale parameter, which determines the spread of the distribution.

In this work, b is set to 10% of the standard deviation of each variable in the dataset. This means that the noise added to each variable is scaled according to the variability (standard deviation) of that variable but at a reduced magnitude (only 10%). This approach ensures that the noise magnitude is proportionate to the inherent variability of each variable, thereby preserving the relative scale of fluctuations across different variables.

Figure 4 demonstrates that, despite the introduction of noise, the relationships between the variables are preserved and remain consistent with the original data scatterplots shown in Figure 3.

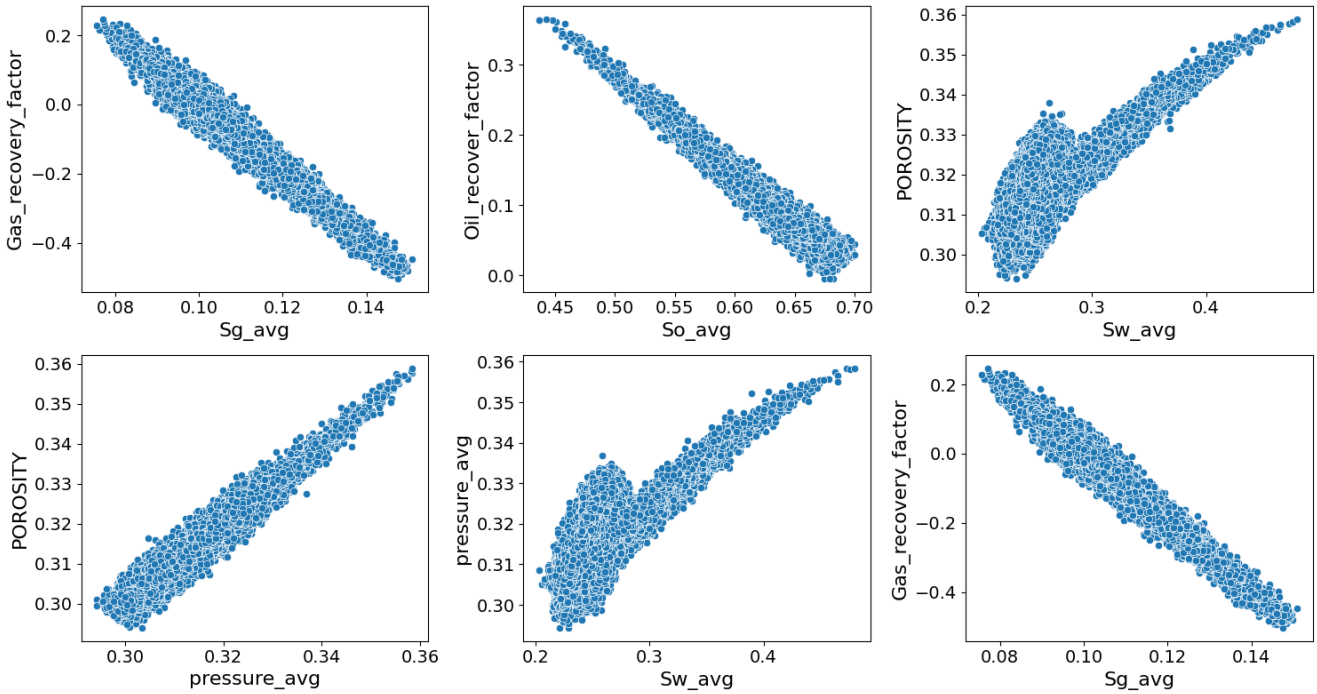

Figure 4. Dataset after adding noise.

2.3. Machine Learning Models

To determine the most accurate forecasting model on a given dataset, the performance of the different models is compared. After the existing literature on oil production forecasting and machine learning was reviewed, the following algorithms were chosen because they have shown an outstanding performance in regression problems: linear regression, polynomial regression, decision trees, random forest, gradient boosting, and artificial neural networks.

In addition to the above methods, extreme gradient boosting and a stacking regressor model were also trained. The basic principles and techniques for parallelizing these algorithms, as well as their associated evaluation metrics, are presented in the following sections.

2.3.1. Linear Models

(a) Linear regression assumes a linear relationship between the input features and the oil recovery factor. It seeks to find the best-fitting straight line (or hyperplane in higher di-

mensions) that minimizes the sum of squared errors between the predicted and actual values. This model serves as a baseline for comparison with more complex algorithms.

(b) Polynomial regression is an algorithm that addresses the nonlinear relationships in the data. This allows the model to fit a curved line (or surface) to the data, potentially improving the accuracy when the relationship between the parameters and oil recovery is not strictly linear. A polynomial regression of degree n is modeled as follows:

$$y = \beta_0 + \sum_{j=1}^{k} \beta_j x_j + \sum_{j=1}^{k} \sum_{i=j}^{k} \beta_{ij} x_i x_j + \sum_{j=1}^{k} \beta_{jj} x_j^2 + \dots , \quad (2)$$

where y is the target variable (the oil recovery factor), $x_1,\ x_2,\ \dots,\ x_k$ are the independent variables (input parameters), $\beta_1, \beta_j,\ \beta_{ij},\ \beta_{jj},\ \dots$ are the coefficients (weights) of the input parameters, and β_0 is the intercept term.

Specifically, PolynomialFeatures with a degree of 5 was implemented to transform the original feature set by introducing new features, such as x^2, x^3, and up to x^5 for each independent variable. This transformation of the feature space enabled the application of a linear regression model to the expanded set of features.

To enhance the model's accuracy and ensure optimal performance, grid search cross-validation was employed. This technique involves systematically searching through a specified hyperparameter space—in this case, the degree of the polynomial features—and evaluating the model's performance using cross-validation.

2.3.2. Tree-Based Algorithms

Beyond these parametric models, several tree-based algorithms were explored, including decision trees, random forests, gradient boosting, and extreme gradient boosting.

(a) Decision trees

Decision trees are a type of algorithm that predicts the target variable by learning simple decision rules inferred from the data features. The prediction of a given input is made by traversing the tree from the root node to a leaf node, where each internal node represents a decision based on a feature.

(b) Random forests

Random forests are an ensemble learning method that builds multiple decision trees and merges them to get a more accurate and stable prediction. Each tree in the forest is trained on a bootstrapped subset of the data, and the final prediction is made by averaging the predictions of all trees in the ensemble.

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} \hat{y}^t , \quad (3)$$

where T is the number of trees and $\hat{y}^t$ is the prediction from the t^{th} tree.

(c) Gradient boosting

Gradient boosting (GB) in an ensemble technique that builds models sequentially, with each new model attempting to correct the errors of the previous ones. The models are added until no further significant improvements can be made. It minimizes the differentiable loss function $L(y,\ \hat{y})$ using gradient descent.

(d) Extreme Gradient Boosting (XGBoost)

XGBoost is an optimized implementation of GB that incorporates regularization techniques to prevent overfitting and enhance performance. It also includes features for handling missing data and parallel computation, making it a robust and efficient algorithm.

For these models, hyperparameter tuning was crucial. Methods such as GridSearchCV with cross-validation were employed to determine the optimal hyperparameters, focusing

on parameters like the maximum tree depth, minimum number of samples required to split a node, and minimum number of samples required to be at a leaf node.

2.3.3. Stacking Machine Learning Models

Stacking regressor is an ensemble method that combines the predictions of multiple underlying regression models to improve the predictive power. It works by training a secondary model that uses the base models' predictions as the input. By learning to weigh the predictions of diverse models, the stacking regressor aims to achieve higher accuracy than any individual model. Figure 5 shows the architecture of the stacking regressor.

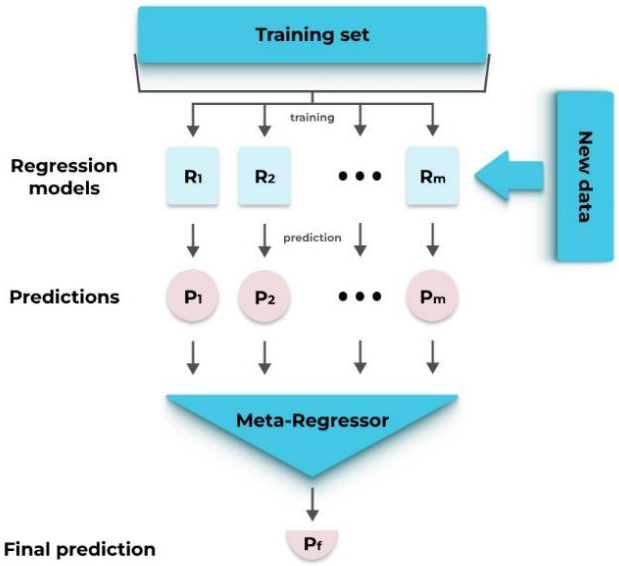

Figure 5. Stacking regressor architecture.

2.3.4. Neural Network

A neural network, which is by nature different from the above algorithms, was then utilized. A neural network uses interconnected neurons in a layered structure that resembles the brain.

To capture the complex, nonlinear relationships inherent in the oil reservoir data, this study utilizes neural networks (Figure 6). Unlike linear models or tree-based methods, ANNs excel at discerning the subtle patterns and interactions among multiple variables.

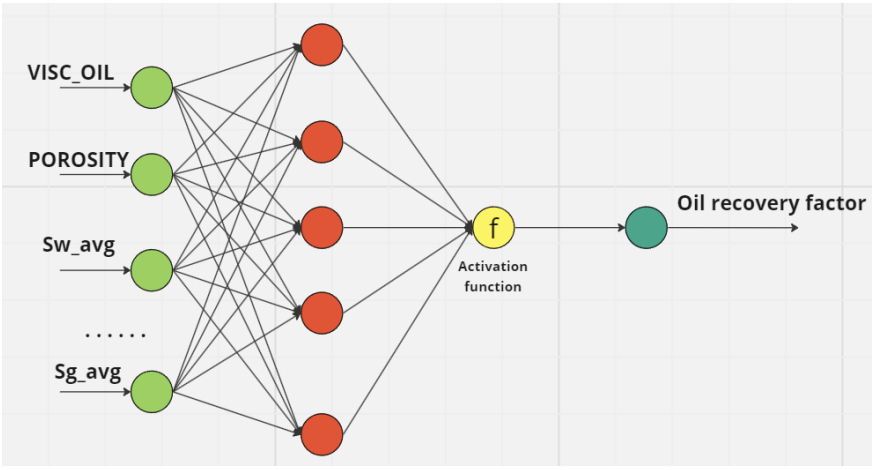

Figure 6. Neural network architecture for the black oil model.

Our model was constructed with five layers, including four fully connected (Dense) layers and one output layer. The ReLU activation function was used. The neural network was trained on input data that include various parameters related to oil wells, such as the depth of the well, its geological characteristics, characteristics of the oil reservoir, etc. After training, the neural network can predict the oil recovery factor based on the new input data.

Through a training process that minimizes the mean squared error between the predicted and actual values, the network learns to accurately forecast the oil recovery. This training utilizes the Adam optimizer, an efficient algorithm for adjusting the network's internal parameters to best fit the data.

Training regression models can take significant computational time, as shown in the Results and Discussion Section. To speed up this process, parallel computing is used.

The current implementation of our neural network is static, focusing on prediction of the oil recovery factors based on the snapshot data of the reservoir's characteristics. We recognize the dynamic nature of oil reservoirs and suggest that future research could explore dynamic neural network models that can incorporate temporal changes and production process data to better reflect the evolving conditions of oil reservoirs.

2.4. Parallel Learning

To speed up the learning process, launches on several parallel processes were implemented.

The parallel learning of machine learning models and neural networks can be divided into two main approaches: data parallelism and model parallelism. Data parallelism implies dividing the dataset into the nodes of a parallel computing system and training independent models on each node on its own sub-datasets. Each model independently performs forecasting, and the final solution is the average of all the models. Parallelizing a model involves distributing different computations across multiple nodes, with each node processing part of the overall task. Despite the distribution of tasks, all nodes contribute towards solving a unified model. The end goal is to efficiently solve a single, coherent model by leveraging the computational power of multiple nodes simultaneously. For small datasets, there is a risk of the loss of accuracy with data parallelism. However, by maintaining optimal accuracy and getting enough acceleration, using data parallelism can be useful. A visual diagram of data parallelism can be seen in Figure 7.

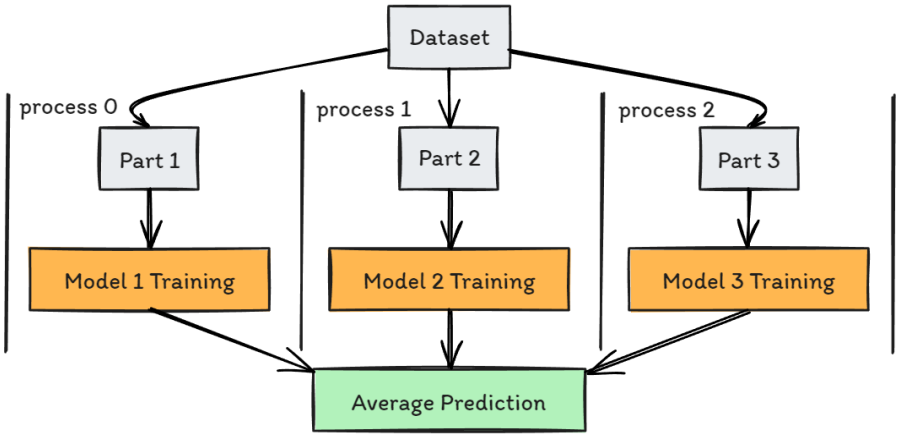

Figure 7. Data parallelism scheme.

In our approach to data parallelism, the same machine learning model, including neural network models, is applied to each subset of the data. This ensures consistency in model training and prediction across all subsets. Specifically, the data are divided into multiple subsets, and the same model (e.g., random forest or a neural network) is trained on each subset independently. The neural network model used on each subset has identical parameters and architecture, including the number of layers, learning rate, and other related

parameters. After training, the predictions from each model are aggregated to obtain the final result. This aggregation can be done by averaging the predictions or using another method to combine them effectively. This approach not only enhances computational efficiency but also maintains the accuracy and reliability of the predictive model.

Using GPUs for machine learning can significantly speed up the calculations, especially when working with large amounts of data due to parallel processing and high throughput, making them an ideal choice for resource-intensive tasks compared to traditional CPUs. To perform training on the GPU, the cuML library (cuml-cu11 23.4.1) included in RAPIDS was used. It provides the highly efficient implementation of various regression methods, such as linear regression, random forest, decision tree, and gradient boosting, using GPU architecture (Ampere) and NVIDIA's CUDA (Version: 11.8) platform. This allows the user to significantly speed up the calculations due to parallel data processing and the optimal use of GPU resources. The implementation of algorithms in cuML is adapted for execution on GPUs, which includes the parallelization of operations and efficient memory management. To speed up the training of the neural network, we used the TensorFlow functionality, which supports execution on the GPU through the CUDA library. This allows the user to efficiently distribute computational tasks across multiple cores, which significantly reduces the model training time. TensorFlow automatically detects available GPUs and distributes the workload to maximize performance and minimize latency, delivering fast and accurate results. While Figure 7 illustrates the use of classical parallel computing technology on a CPU, our implementation on a GPU using CUDA involves different mechanisms (Figure 8). We chose RAPIDS for our machine learning implementation due to its alignment with scikit-learn in terms of the method implementation. This consistency allows us to leverage the familiar scikit-learn API while taking advantage of the GPU acceleration provided by CUDA. The methods in RAPIDS are designed to be compatible with scikit-learn, making it straightforward to compare and validate the performance of models implemented in both libraries. Using CUDA, RAPIDS utilizes GPU threads to perform parallel computations, which significantly enhances the processing speed for large datasets. This approach ensures that the operations are efficient and scalable, enabling us to achieve a high performance in our machine learning tasks.

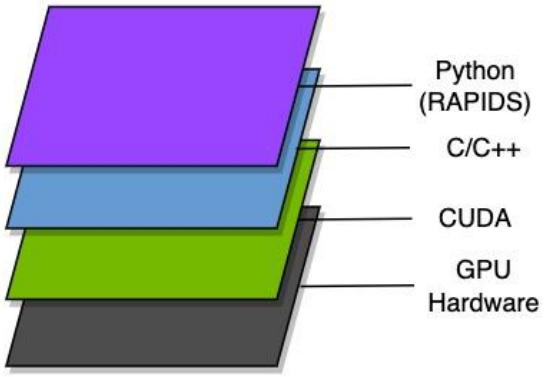

Figure 8. Parallelism on GPU.

The choice of operating system also plays a crucial role in the organization of parallel computing. We conducted our experiments on a Linux-based system, which is known for its robust support for parallel and multi-threaded computing. Linux provides efficient process scheduling and resource management, which are essential for achieving a high performance in parallel computations.

Next, the machine learning models described above are used on the dataset, and data parallelism is employed during training to speed up the process.

3. Results

In this study, the dataset is divided into training and test sets in a ratio of 80% to 20%. For the neural network, the validation sample was 20% of the training set. The models were evaluated using the key metrics: MSE, MAE, and R^2. The scaling of the training and test sets is presented in Table 3.

Table 3. The size of the training set and test set.

Data Type	Data Points
Training set	295,680
Test set	73,920
Total	369,600

Figure 8 shows a scatterplot comparing the training data and testing data for the models trained. Each graph displays a series of data points where the actual values of the dependent variable are compared with the predicted values obtained from the linear regression model.

To find the optimal parameters, a grid search was applied to optimize the hyperparameters of the tree-based models and artificial neural networks (ANNs). The optimal combinations of the hyperparameters were determined based on the accuracy of the test set used as the performance metric. A range of hyperparameters for these models was defined based on the dataset size and their characteristics. Within these given ranges, different combinations of parameters were systematically explored to find the optimal set. Table 4 shows the R^2 score using the optimized parameters.

Table 4. R^2 score of various algorithms on training and test sets after optimization.

	LR	PR	DT	RF	GB	XGBoost	NN	Stacking Regressor
Training set	0.97	0.98	0.98	0.99	0.99	0.98	0.97	0.97
Test set	0.97	0.98	0.97	0.98	0.98	0.98	0.97	0.97
Average	0.97	0.98	0.975	0.985	0.985	0.98	0.97	0.97

Analyzing the performance of various machine learning models on training datasets revealed the following key observations. First, most models, including linear regression (LR), polynomial regression (PR), decision tree (DT), random forest (RF), gradient boosting (GB), and extreme gradient boosting (XGBoost), demonstrate high accuracy in both the training and test data, exceeding the threshold value of 0.97. This indicates the high generalization ability of these models, which can be seen in Figures 9 and 10. The neural network (NN) also achieves high accuracy on both datasets, although it was slightly lower compared to the other models.

Figure 10 shows the actual and predicted results for the regression models based on the testing data. This graph clearly demonstrates the models' ability to generalize beyond the training set. The models show comparable coefficients of determination R^2, indicating that they are effective at capturing the underlying patterns in the data, even when exposed to new, unseen data. Remarkably, the PR model achieved a high R^2 value of 0.977, effectively capturing the nonlinear relationships among the variables. The remaining models showed results of about 0.974, which are also very high. These results highlight the ability of both traditional algorithms and advanced machine learning techniques, including neural networks, to achieve high prediction accuracy and maintain consistency across different pieces of data.

The second observation is that the stacking regressor shows a comparable performance to the other models on the test dataset but lags slightly on the training set. This may indicate that the stacking regressor is more robust to overfitting and is able to generalize better to new data.

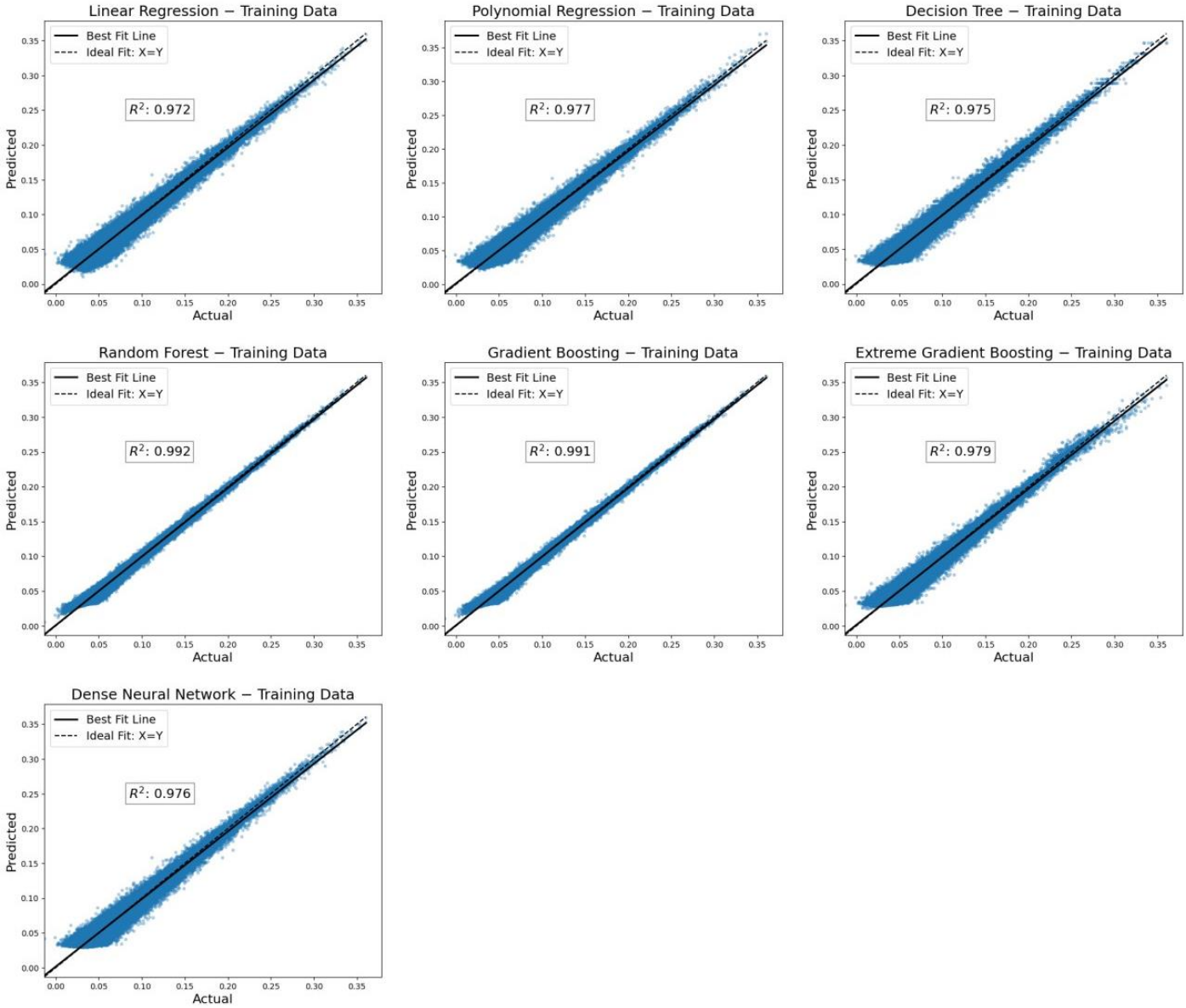

Figure 9. Comparison of predicted versus actual values on training set.

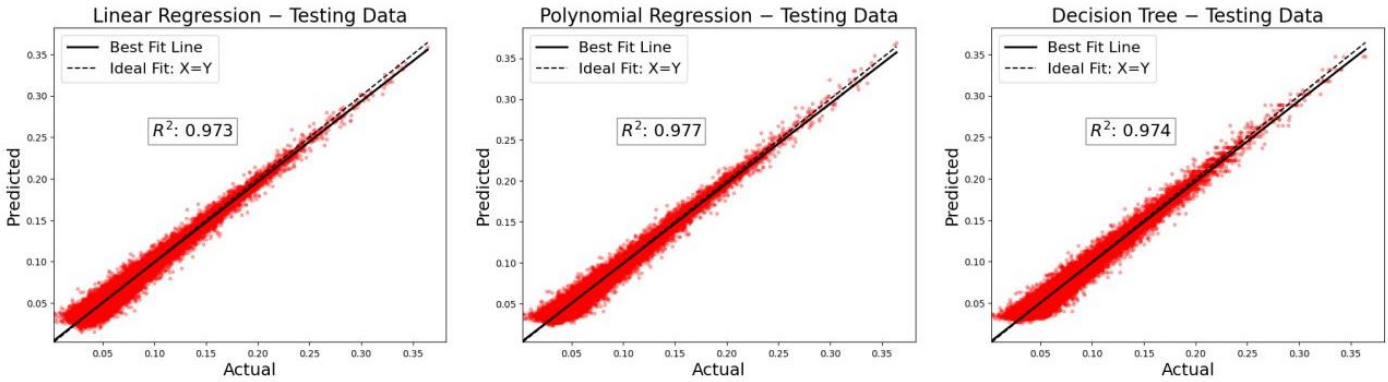

Figure 10. *Cont*.

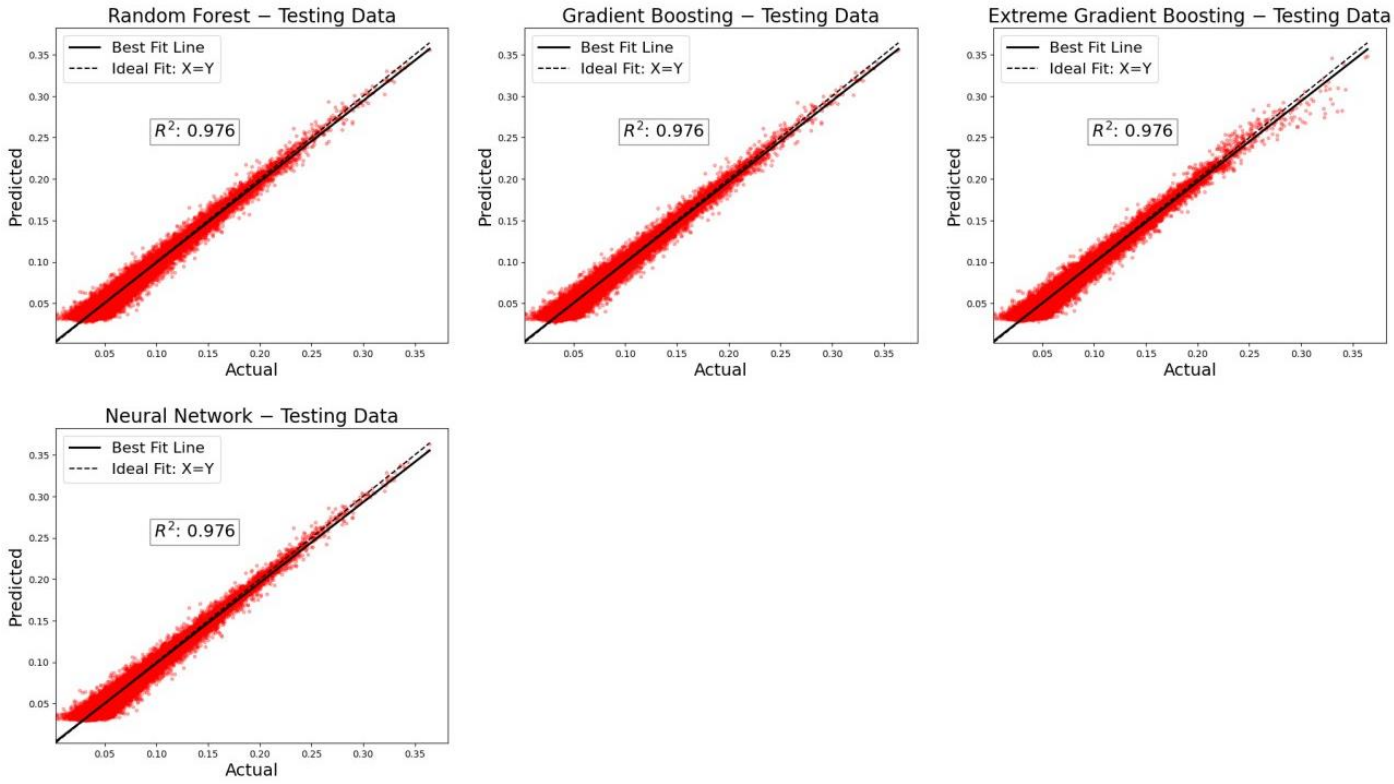

Figure 10. Comparison of predicted versus actual values on testing set.

A comparison of the R^2 scores before and after the model optimization, shown in Figure 11, is important for assessing the effectiveness of the improvements made to the algorithms.

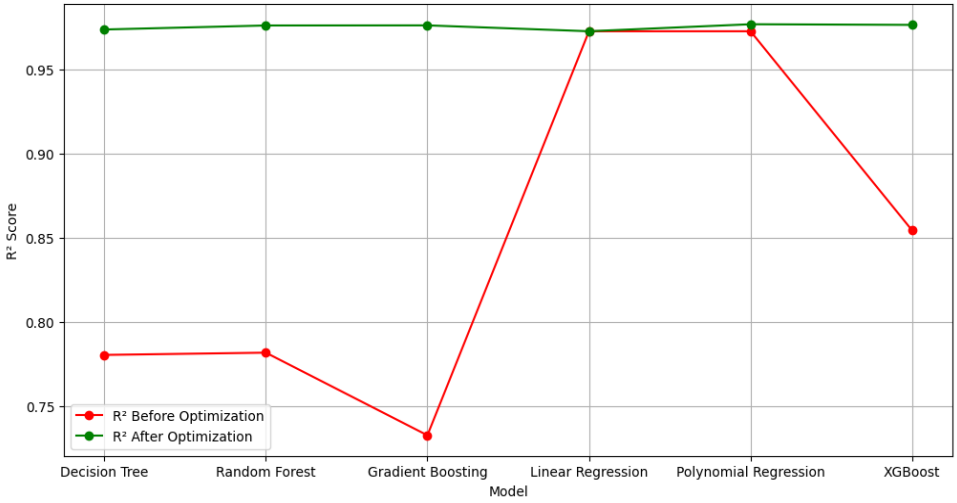

Figure 11. R^2 score of different algorithms on training set and test set.

It can be seen that decision trees, random forest, gradient boosting, and extreme gradient boosting showed a great improvement after being optimized, increasing their R^2 score by more than 10% at least. This means that, once optimized, these models are better at adjusting for changes in the data and can make more accurate predictions than the original values. These results confirm that optimizing the models can significantly improve their performance and make them more reliable for use in predictive models.

A comparison of the mean square error (MSE) results before and after optimizing the models is shown in Figure 12.

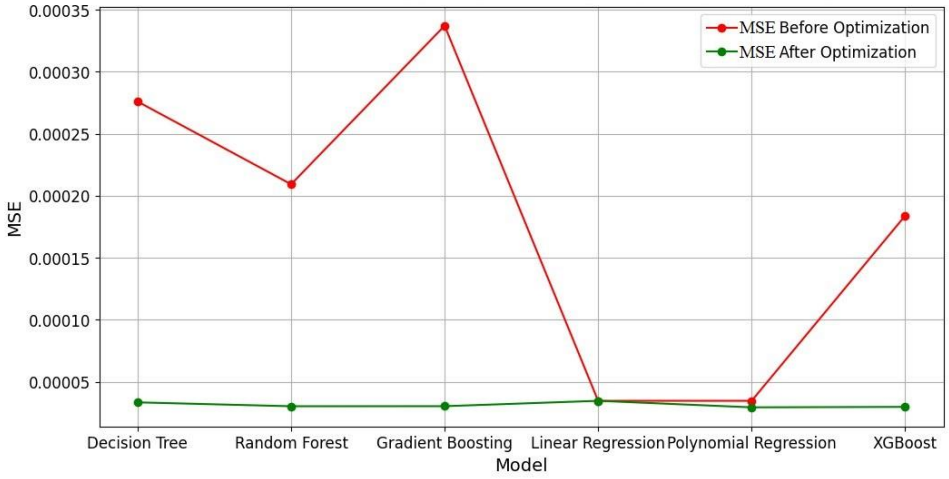

Figure 12. MSE of different algorithms on training set and test set.

Before the optimization, the MSE for the decision tree, random forest, gradient boosting and extreme gradient boosting was significantly higher. After optimization, the MSE significantly decreased.

The linear regression remained the same because it does not have any parameters to be tuned. For the polynomial regression, optimization means finding the best degree; however, this did not bring much change in the performance of the model. The predicted values for all models can be seen in Figure 13.

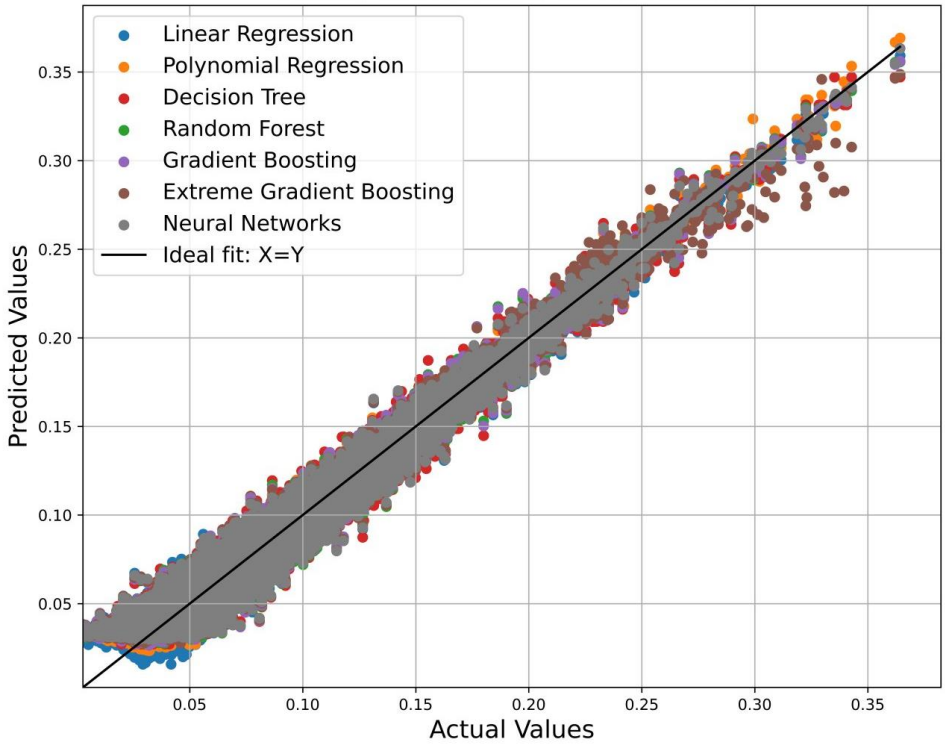

Figure 13. Actual vs. predicted for all models.

In order to see the impact of the features on the model output, SHAP (Shapley additive explanations) value plots were used, as seen in Figure 14. The SHAP values explain the output of a model by quantifying the contribution of each feature.

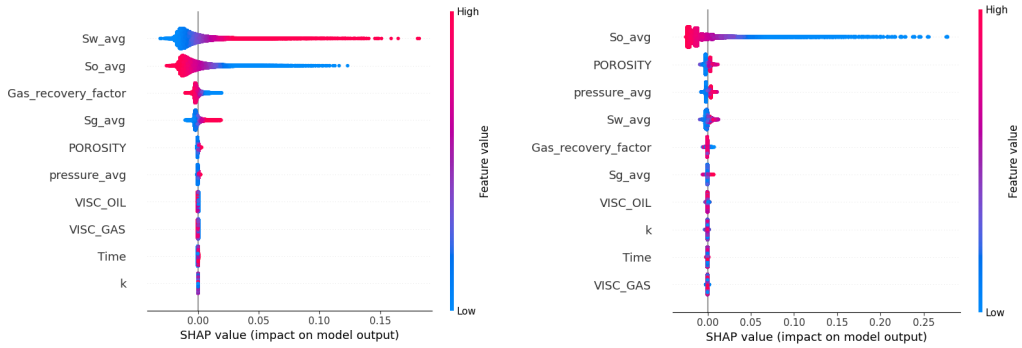

Figure 14. SHAP visualization for linear regression (**left**) and decision tree (**right**).

The features are ranked in order of the importance in influencing the model's output from top to bottom. The water saturation has the highest impact, followed by the oil saturation and gas recovery factor for the linear regression model, whereas, for the decision tree model, only the oil average shows a significant impact.

In summary, hyperparameter optimization resulted in significant improvements in the performance of all the models, reducing the mean squared error and producing more accurate predictions.

When training models on a conventional processor, the training time increases significantly. Now, the challenge is to speed up the training of the model.

Algorithms for the parallel learning of the regression models were developed and implemented using the mpi4py library. "mpi4py" is a Python package that provides bindings to the MPI (Message Passing Interface) library. MPI is a standard interface used for communication between processes running on different nodes within a computing cluster or parallel computing system. Using process ranks, the dataset was partitioned and distributed across different nodes. Each node then conducted model training in parallel, operating independently from the others. After the training, the performance (Tables 5 and 6) and accuracy (Table 7) of the model were collectively analyzed to assess the effectiveness of the distributed training approach.

Table 5. Time of parallel training of regression models, seconds.

Number of Processes	GB	DT	RF	LR	PR	NN
1	12,841.98	705.69	1054.32	0.026	585.15	124.97
2	6377.08	323.54	502.66	0.0194	381.87	62.47
4	3104.33	146.97	235.31	0.016	132.54	24.23
8	1508.85	66.95	106.15	0.023	66.45	16.76

Table 6. Speedup of parallel learning of regression models.

Number of Processes	GB	DT	RF	LR	PR	NN
1	1	1	1	1	1	1
2	2.013771	2.18115	2.097481	1.34020	1.53232	2.0004
4	4.136796	4.80159	4.480557	1.625	4.41489	5.1576
8	8.511104	10.5405	9.932359	1.13043	8.80586	7.4564

Table 7. R^2 score of different algorithms on different numbers of processes.

Number of Processes	GB	DT	RF	LR	PR	NN
1	0.9757	0.9735	0.975	0.9727	0.9771	0.9756
2	0.9758	0.9742	0.976	0.9727	0.9764	0.9758
4	0.9759	0.9753	0.976	0.9727	0.9765	0.9752
8	0.9758	0.975	0.975	0.9727	0.9764	0.9751

As seen in the results above, good acceleration has been obtained for almost all the models. At the same time, acceleration is achieved with almost no loss in accuracy. These results make it possible to consider MPI parallelization as a good opportunity to speed up the model learning.

The smallest acceleration was obtained for the linear regression, with the reason for this phenomenon being the short learning time even in one process.

In our study, we employed the Kendall concordance coefficient to evaluate the consistency of predictions from the machine learning models trained across eight parallel processes. This analysis aimed to assess the reliability of parallelized computations in maintaining uniform outcomes when employing different subsets of the data. Our findings (Table 8) indicate a high level of concordance among the outputs, suggesting that parallel training on multiple nodes does not compromise the predictive stability of the models. This underscores the effectiveness of parallel computational approaches in handling complex, large-scale machine learning tasks.

Table 8. Kendall concordance coefficient of the forecast on eight processes.

GB	DT	RF	LR	PR	NN
0.9978	0.9884	0.9981	0.9999	0.9995	0.9991

To determine the performance of parallel execution on the GPU, the tests were conducted on an Nvidia RTX 3070 graphics card. The results can be seen in Table 9.

Table 9. Time to train regression models on GPU, seconds.

GB	DT	RF	LR	NN
0.28	0.68	0.66	0.013	15.8

As can be seen from the results, the use of GPU significantly accelerated the learning process. However, the results of the polynomial regression are missing in Table 9, since this method is not directly supported in CuML.

For this dataset, using GPU is an optimal solution. Based on the results obtained, it can be concluded that with a large dataset and access to a system with multiple GPUs, it is possible to get even more acceleration using data parallelism on multiple GPUs.

4. Discussion

The high accuracy achieved by all the tested models, exceeding a R^2 score of 0.97, indicates the potential of machine learning for oil recovery forecasting. However, the relative performance and suitability of different algorithms varied. While the linear and polynomial regression were surprisingly effective, the success of the linear models suggests that the underlying relationships in the synthetic dataset, while potentially nonlinear, might be well approximated by simpler models. This finding warrants further investigation with real-world data, which often exhibits greater complexity.

Tree-based ensembles demonstrated an excellent performance with their ability to capture nonlinear interactions among the reservoir parameters. The significant improvement observed after the hyperparameter tuning demonstrates the importance of careful model configuration for the optimal results. The neural network also achieved high accuracy, suggesting its potential for uncovering subtle patterns and interactions within the reservoir data that might be missed by other methods.

Despite their promise, it is crucial to acknowledge the limitations of these models. All machine learning models are inherently limited by the data they are trained on. While our synthetic dataset encompassed a wide range of reservoir conditions, it may not fully represent the complexities and heterogeneity of real-world reservoirs. Moreover, machine learning models are generally poor at extrapolating beyond the range of data they were

trained on. If the model encounters reservoir conditions significantly different from those represented in the training data, its predictions may be unreliable.

Another crucial aspect to note is how parallelization has expedited training. Although the results indicate that parallel execution across multiple CPUs significantly accelerates the training, the speedup is not comparable to that achieved with GPUs. Utilizing GPUs allows for processing large datasets and accelerating computations, which is critical for real-time decision-making in oil reservoir management. However, the first type of parallelization remains vital when training on large datasets and cluster systems.

In future research, it might be beneficial to combine these approaches by segmenting the dataset into several parts and training each model on multiple GPUs. Such strategies are planned for our forthcoming studies.

5. Conclusions

This study is pivotal in advancing the application of machine learning to optimize oil recovery forecasts, aiming to significantly enhance the decision-making processes in oil production. By utilizing the black oil model, this research employed advanced machine learning algorithms to predict the oil recovery factor, with the RF and GB models achieving an R^2 score of 0.985. The PR and XGBoost models also performed well, each with R^2 scores of 0.98, while the LR, NN, and DT models showed slightly lower R^2 scores of 0.97, 0.97, and 0.975, respectively. Quantitative assessments were made on the key parameters that influence oil recovery, providing deeper insights into the production processes and optimization strategies. One of the notable computational advancements was the implementation of parallel computing techniques. In particular, the DT algorithm achieved a maximum speedup of 10.54 times when running on eight processes; other models, except LR, also showed speedups in the region of 8 times.

This research primarily utilized synthetic data, which facilitated controlled conditions but might limit the generalizability of the findings to real-world scenarios. Recognizing this limitation, this study highlights the necessity for further validation using real-world datasets to ensure the practical applicability and robustness of the developed models. Future research will apply these machine learning and parallel computing methods to larger datasets with real data encompassing multiple parameters, aiming to provide forecasts under actual operational conditions. This research not only demonstrates the efficacy of machine learning in enhancing oil recovery forecasts but also underscores the potential of integrating advanced computational techniques with traditional petroleum engineering practices.

Author Contributions: Conceptualization, B.M. and T.I.; methodology, B.D. and M.M.; software, A.M.; validation, A.M. and N.K.; formal analysis, B.M.; investigation, T.I.; resources, B.D.; data curation, N.K.; writing—original draft preparation, N.K.; writing—review and editing, T.I. and M.M.; visualization, A.M.; supervision, B.M.; project administration, B.M.; and funding acquisition, T.I. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan, grant numbers BR18574136 and AP14871644.

Data Availability Statement: The data presented in this paper are available upon request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Hørsholt, S.; Nick, H.M.; Jørgensen, J.B. Oil Production Optimization of Black-Oil Models by Integration of Matlab and Eclipse E300. *IFAC-PapersOnLine* **2018**, *51*, 88–93. [CrossRef]
2. Amooie, M.A.; Moortgat, J. Higher-Order Black-Oil and Compositional Modeling of Multiphase Compressible Flow in Porous Media. *Int. J. Multiph. Flow.* **2018**, *105*, 45–59. [CrossRef]
3. Nojabaei, B.; Siripatrachai, N.; Johns, R.T.; Ertekin, T. Effect of Large Gas-Oil Capillary Pressure on Production: A Compositionally-Extended Black Oil Formulation. *J. Pet. Sci. Eng.* **2016**, *147*, 317–329. [CrossRef]

4. Sandve, T.H.; Sævareid, O.; Aavatsmark, I. Dynamic PVT Model for CO_2-EOR Black-Oil Simulations. *Comput. Geosci.* **2022**, *26*, 1029–1043. [CrossRef]
5. Fioroni, S.; Larreteguy, A.E.; Savioli, G.B. An OpenFOAM Application for Solving the Black Oil Problem. *Math. Models Comput. Simul.* **2021**, *13*, 907–918. [CrossRef]
6. Ilyushin, Y.V.; Novozhilov, I.M. Temperature field control of a metal Oil-well tubing for producing of high-paraffin oil. In Proceedings of the 2020 XXIII International Conference on Soft Computing and Measurements (SCM), Petersburg, Russia, 1 May 2020. [CrossRef]
7. Burachok, O.V.; Pershyn, D.V.; Matkivskyi, S.V.; Kondrat, O.R. Evaluation of Black-Oil PVT-Model Applicability for Simulation of Gas-Condensate Reservoirs. *Miner. Resour. Ukr.* **2020**, *2*, 43–48. [CrossRef]
8. Mydland, S.; Whitson, C.H.; Carlsen, M.L.; Dahouk, M.M.; Yusra, I. *Black-Oil and Compositional Reservoir Simulation of Gas-Based EOR in Tight Unconventionals*; OnePetro: Richardson, TX, USA, 2020.
9. Du, F.; Nojabaei, B. A Black-Oil Approach to Model Produced Gas Injection in Both Conventional and Tight Oil-Rich Reservoirs to Enhance Oil Recovery. *Fuel* **2020**, *263*, 116680. [CrossRef]
10. Wei, Z.; Kang, X.; Zhang, X.; Zhang, J. A Black-Oil-Based Multi-Component Model for Polymer Flooding. *Sci. Sin. Technol.* **2018**, *48*, 415–423. [CrossRef]
11. Song, L.Y.; Wei, J.C.; Xu, X.M.; Gao, T. Optimization of Black Oil Model-CO_2 Flooding in Low Permeability Reservoir. *Appl. Mech. Mater.* **2014**, *580–583*, 2502–2507. [CrossRef]
12. Wong, T.; Fleming, G.C. Modified Black Oil Model for Calculating Mixing of Different Fluids in a Common Surface Network 2019. U.S. Patent No. 10,387,591, 20 August 2019.
13. Klemetsdal, Ø.S.; Rasmussen, A.F.; Møyner, O.; Lie, K.-A. Efficient reordered nonlinear Gauss–Seidel solvers with higher order for black-oil models. *Comput. Geosci.* **2020**, *24*, 593–607. [CrossRef]
14. Klemetsdal, Ø.S.; Flø Rasmussen, A.; Møyner, O.; Lie, K.-A. Nonlinear gauss-seidel solvers with higher order for black-oil models. In Proceedings of the ECMOR XVI-16th European Conference on the Mathematics of Oil Recovery, Barcelona, Spain, 3–6 September 2018. [CrossRef]
15. Jiang, J.; Wen, X.-H. Smooth Formulation for Three-Phase Black-Oil Simulation with Superior Nonlinear Convergence. In Proceedings of the SPE Reservoir Simulation Conference, Galveston, TX, USA, 21 March 2023. [CrossRef]
16. Møyner, O.; Lie, K.-A. A Multiscale Restriction-Smoothed Basis Method for Compressible Black-Oil Models. *SPE J.* **2016**, *21*, 2079–2096. [CrossRef]
17. Shirazi, A.; Hezarkhani, A.; Beiranvand Pour, A.; Shirazy, A.; Hashim, M. Neuro-Fuzzy-AHP (NFAHP) Technique for Copper Exploration Using Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER) and Geological Datasets in the Sahlabad Mining Area, East Iran. *Remote Sens.* **2022**, *14*, 5562. [CrossRef]
18. Shirazy, A.; Hezarkhani, A.; Timkin, T. Investigation of Magneto-/Radio-Metric Behavior in Order to Identify an Estimator Model Using K-Means Clustering and Artificial Neural Network (ANN) (Iron Ore Deposit, Yazd, IRAN). *Minerals* **2021**, *11*, 1304. [CrossRef]
19. Johnson, E.; Obot, O.; Attai, K.; Akpabio, J.; Inyang, U. The Use of Machine Learning in Oil Well Petrophysics and Original Oil in Place Estimation: A Systematic Literature Review Approach. *J. Eng. Res. Rep.* **2023**, *25*, 40–54. [CrossRef]
20. Koray, A.-M.; Bui, D.; Ampomah, W.; Appiah Kubi, E.; Klumpenhower, J. *Application of Machine Learning Optimization Workflow to Improve Oil Recovery*; OnePetro: Richardson, TX, USA, 2023.
21. Gulin, A.B.; Kerimov, A.-G.H. Application of Machine Learning Methods for Well Logging Data Interpretation. *GGDOGF* **2022**, *9*, 48–54. [CrossRef]
22. Kenzhebek, Y.; Imankulov, T.; Akhmed-Zaki, D.; Daribayev, B. Implementation of Regression Algorithms for Oil Recovery Prediction. *East. Eur. J. Enterp. Technol.* **2022**, *2*, 69–75. [CrossRef]
23. Helland, J.O.; Friis, H.A.; Assadi, M.; Nagy, S. Machine Learning for Underground Gas Storage with Cushion CO_2 Using Data from Reservoir Simulation. *IOP Conf. Ser. Mater. Sci. Eng.* **2023**, *1294*, 012058. [CrossRef]
24. Mousavi, S.M.; Bakhtiarimanesh, P.; Enzmann, F.; Kersten, M.; Sadeghnejad, S. Machine-Learned Surrogate Models for Efficient Oil Well Placement Under Operational Reservoir Constraints. *SPE J.* **2024**, *29*, 518–537. [CrossRef]
25. Samnioti, A.; Gaganis, V. Applications of Machine Learning in Subsurface Reservoir Simulation—A Review—Part II. *Energies* **2023**, *16*, 6727. [CrossRef]
26. Chen, X.; Zhang, K.; Ji, Z.; Shen, X.; Liu, P.; Zhang, L.; Wang, J.; Yao, J. Progress and Challenges of Integrated Machine Learning and Traditional Numerical Algorithms: Taking Reservoir Numerical Simulation as an Example. *Mathematics* **2023**, *11*, 4418. [CrossRef]
27. Wang, H.; Chen, S. Insights into the Application of Machine Learning in Reservoir Engineering: Current Developments and Future Trends. *Energies* **2023**, *16*, 1392. [CrossRef]
28. Kazemi, M.; Takbiri-Borujeni, A.; Nouroizeh, H.; Kazemi, A.; Takbiri, S.; Wallrich, C. A Novel Surrogate Model for Reservoir Simulations Using Fourier Neural Operators. In Proceedings of the SPE Annual Technical Conference and Exhibition? San Antonio TX, USA, 9 October 2023; OnePetro: Richardson, TX, USA, 2023.
29. MPI Forum. Available online: https://www.mpi-forum.org/ (accessed on 1 August 2024).
30. OpenMP. Available online: https://www.openmp.org/ (accessed on 1 August 2024).
31. Stone, J.E.; Gohara, D.; Shi, G. OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems. *Comput. Sci. Eng.* **2010**, *12*, 66–73. [CrossRef] [PubMed]

32. Reinders, J. *Intel Threading Building Blocks: Outfitting C++ for Multi-Core Processor Parallelism*, 1st ed.; O'Reilly: Beijing, China; Köln, Germany, 2007.
33. Pheatt, C. Intel® threading building blocks. *J. Comput. Sci. Coll.* **2008**, *23*, 298.
34. Salzman, P.J. *The Linux Kernel Module Programming Guide*; CreateSpace: Scotts Valley, CA, USA, 2009.
35. Richardson, B.; Rees, B.; Drabas, T.; Oldridge, E.; Bader, D.; Allen, R. Accelerating and Expanding End-to-End Data Science Workflows with DL/ML Interoperability Using RAPIDS. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, in KDD '20, New York, NY, USA, 6–10 July 2020; pp. 3503–3504. [CrossRef]
36. Chen, Z. *Reservoir Simulation: Mathematical Techniques in Oil Recovery*; Cambridge University Press: Cambridge, UK, 2007.