Synthetic Well Log Generation Software

Daniel E. Keller
West Virginia University, dekeller@mix.wvu.edu

Synthetic Well Log Generation Software

Daniel Keller

A Thesis submitted to
Benjamin M. Statler College of Engineering and Mineral Resources
West Virginia University

In partial fulfillment of the requirements for the degree of
Master of Science
In Petroleum and Natural Gas Engineering

Department of Petroleum and Natural Gas Engineering
Morgantown, West Virginia
2023

Keywords: Synthetic Well Log, Artificial Intelligence, Neural Network

Abstract

Synthetic Well Log Generation Software

Daniel Keller

In this study, we developed a novel approach to generate synthetic well logs using backpropagation neural networks through the use of an open source software development tool. Our method predicts essential well logs such as neutron porosity, sonic, photoelectric, and resistivity, which are crucial in various stages of oil and gas exploration and development, as they help determine reservoir characteristics. Our approach involves sequentially predicting well logs, using the outputs of one prediction model as inputs for subsequent models to generate comprehensive and coherent sets of well logs. We trained and tested our models using 16 wells from a single field, and the resulting synthetic well logs demonstrated an acceptable degree of accuracy and consistency with the actual logs, thus supporting the efficacy of our approach. This research not only opens up new avenues for enhancing the efficiency of hydrocarbon exploration but also contributes to the growing body of knowledge in the field of AI and ML applications in the oil and gas industry. This work also demonstrates the capabilities of open source tools for developing software and for oil and gas applications.

Contents

List of Figures

1. Introduction

1.1 Background and Context

Introduction to Well Logging

Well logging is a fundamental process in the petroleum and geosciences industries, aimed at evaluating subsurface formations. This process involves recording various physical, chemical, and geological properties of the earth's subsurface through indirect measurements made from boreholes. Well logs are crucial for understanding the geology and hydrocarbon potential of an area, providing invaluable data for exploration, production, and development activities in the oil and gas industry.

Key Well Logs In This Study

Neutron Porosity Logs are essential in determining the porosity of formations, providing insights into the presence and quantity of hydrocarbons. These logs measure the hydrogen atom concentration in formations, which is directly related to fluid content. Bulk Density Logs offer critical information about the density of subsurface formations. This parameter is vital for hydrocarbon identification and for understanding rock composition and structure. Gamma Ray Logs are used to evaluate the natural radioactivity of formations. These logs help in identifying different rock types and stratigraphic boundaries, playing a crucial role in lithology differentiation and correlation. Sonic Logs measure the travel time of acoustic waves in formations. They provide essential data on the rock's elastic properties and pore spaces, aiding in the evaluation of formation lithology and fluid content. Photoelectric Logs (PEF) measure the photoelectric absorption index of formations. This data is crucial for mineral identification, particularly in distinguishing between different types of carbonate rocks. Resistivity Logs assess the electrical resistivity of formations. They are vital for identifying hydrocarbon-bearing zones by measuring the electrical resistance, which is generally lower in hydrocarbon-saturated rocks compared to water-saturated rocks. Caliper Logs are used to measure the borehole diameter. This information is crucial for identifying borehole conditions and detecting zones of washouts or cavings, which can affect the interpretation of other log data. Andersen, The Defining Series: Basic Well Log Interpretation

Artificial Neural Networks in Geoscience and Well Log Analysis

Artificial intelligence (AI) and machine learning (ML) have made significant strides in recent years, and their applications have expanded across various industries. In the context of synthetic well logging, these advanced technologies offer promising solutions for enhanced subsurface characterization, reservoir modeling, and improved decision-making processes.

Synthetic well logs are essential in the oil and gas industry, as they provide valuable data about rock properties, fluid content, and reservoir conditions. Traditional methods of generating synthetic logs rely on expert knowledge and deterministic models, which may not always be accurate or consistent. AI and ML techniques, such as deep learning and data-driven modeling, offer a more robust and reliable approach for generating these logs by leveraging vast amounts of available data to learn complex patterns and relationships.

One of the key applications of AI and ML in synthetic well logging is the prediction of petrophysical properties from seismic data. By using neural networks and other machine learning algorithms, researchers can develop accurate models that predict properties like porosity, permeability, and fluid saturation from seismic attributes. These predictive models enable geoscientists to make more informed decisions about reservoir development, drilling locations, and production strategies, ultimately leading to reduced costs and increased efficiency.

Another crucial application of AI and ML in synthetic well logging is automated formation evaluation. By incorporating machine learning algorithms into the interpretation process, geoscientists can identify different rock types, lithofacies, and fluid content with higher accuracy and speed. This automated approach allows for a more detailed understanding of the subsurface, enabling better reservoir management and exploitation.

In conclusion, artificial intelligence and machine learning have revolutionized the field of synthetic well logging by providing more accurate and reliable predictions of subsurface properties. As AI and ML technologies continue to advance, they will play an increasingly vital role in the oil and gas industry, enabling more efficient resource extraction and improved decision-making processes. By integrating these innovative technologies, companies can optimize their operations and minimize the risks associated with exploration and production.

The Relevance of Santa Fe Area, Kansas Data

The Santa Fe area in Kansas provides a unique geological setting, rich in hydrocarbon resources and diverse stratigraphy. The data from this region is integral for training and testing the neural network models, ensuring that the generated synthetic logs accurately reflect the complex geological characteristics of the area. This region's data offers a valuable case study for applying ANNs in well log analysis due to its varied lithologies and reservoir properties.

1.2 Research Objectives

Objective 1: To Develop an Artificial Neural Network Model for Synthetic Well Log Generation

The primary objective of this research is to develop an efficient and accurate artificial neural network (ANN) model capable of generating synthetic well logs. This model will be designed to assimilate and process key well log data - specifically, neutron porosity, bulk density, and gamma ray logs - to synthesize new log profiles. The aim is to address the common challenge of incomplete or missing log data in subsurface studies, thus enhancing geological and petrophysical interpretation accuracy.

Objective 2: To Validate the ANN Model Using Data from the Santa Fe Area, Kansas

A critical objective is the validation of the ANN model using extensive well log data from the Santa Fe area in Kansas. This area is chosen due to its complex geological structure and diverse lithological features, providing a robust testing ground for the ANN model. Validation involves comparing synthetic logs generated by the ANN with actual well log data, assessing the model's accuracy in replicating key geological features and its reliability in various geological settings.

Objective 3: To Evaluate the Efficacy of ANN in Enhancing Subsurface Interpretation

This research aims to evaluate the efficacy of the ANN model in enhancing the interpretation of subsurface geological formations. By generating synthetic logs where data is missing or sparse, the research seeks to demonstrate how ANNs can contribute to more comprehensive geological models, improved reservoir characterization, and better-informed decision-making in exploration and production activities.

Objective 4: To Explore the Potential of ANN Models in Predictive Analysis

An ancillary objective is to explore the potential of the developed ANN model in predictive analysis. This involves using the model to predict key geological and petrophysical properties in areas where well log data is unavailable or limited, thereby assessing the model's utility in guiding exploratory drilling and production strategies.

Objective 5: To Contribute to the Body of Knowledge in Applied Geosciences

Finally, this thesis aims to contribute significantly to the existing body of knowledge in applied geosciences, particularly in the area of artificial intelligence applications. By documenting the process, challenges, and outcomes of using ANNs in well log analysis, this research seeks to provide valuable insights and methodologies that can be adopted and adapted in similar geoscientific applications.

1.3 Thesis Structure

This thesis is structured to provide a comprehensive understanding of the application of artificial neural networks (ANNs) in generating synthetic well logs, with a focus on neutron porosity, bulk density, and gamma ray logs in the Santa Fe area of Kansas. The structure is as follows:

1. **Introduction:** This chapter sets the stage for the thesis, introducing the topic, its significance in the field of geosciences, and an overview of the primary objectives and scope of the research.

2. **Literature Review:**

- *Fundamentals of Well Logging:* Provides a detailed background on well logging, its importance, and the types of logs focused on in this study.
- *Artificial Neural Networks:* Discusses the basics of ANNs, their principles, and functionalities.
- *Previous Applications of ANN in Well Logging:* Reviews past research and applications of ANNs in well log analysis, highlighting various approaches and findings.

3. **Methodology:**
    - *Data Collection:* Details the process of data gathering, including the source and nature of the well log data used.
    - *Data Exploration:* Details the process of visualizing the data.
    - *Data Processing:* Details the process of preparing the data to be used to train the network.
    - *Training the ANN:* Outlines the training process, including the algorithms used and the training dataset.
    - *Deploying the Model:* Outlines the process of deploying the model on new data.
4. **Results and Discussion:**
    - *Analysis of Results:* Presents the findings from the application of the ANN model to the well log data.
5. **Conclusions and Future Work:**
    - *Summary of Findings:* Summarizes the key findings and contributions of the research.
    - *Implications of Research:* Explores the broader implications of the research findings for the field of geosciences and well log analysis.
    - *Limitations:* Acknowledges the limitations of the current study.
    - *Recommendations for Future Research:* Suggests potential areas for future research building upon the findings of this thesis.

2. Literature Review

2.1 Fundamentals of Well Logging

Well log interpretation is a crucial process in the oil and gas industry, as it provides valuable information about subsurface formations and their properties. By analyzing well logs, geoscientists can identify potential hydrocarbon reservoirs, assess their quality, and estimate their production potential. The interpretation typically involves the analysis of several types of logs, including neutron porosity, sonic (Delta T), bulk density, photoelectric, and resistivity logs. The combination of these logs helps in determining the lithology, porosity, fluid content, and other key characteristics of the rock formations. (The Defining Series: Basic Well Log Interpretation)

Neutron porosity logs measure the hydrogen content of the subsurface formations. Since hydrogen is a primary component of fluids (water, oil, and gas) within the rock pores, the neutron porosity log provides an indirect measure of porosity. Neutrons emitted from a source in the logging tool interact with the formation, and the tool measures the number of neutrons that return to the detector. A higher concentration of hydrogen in the formation causes more neutron scattering and absorption, resulting in a lower neutron count rate. By comparing the measured neutron count rate with a reference value from a known porosity material, the porosity of the formation can be calculated.(Neutron porosity logs)

The sonic log, also known as the Delta T or acoustic log, measures the travel time of sound waves through rock formations. The travel time, expressed in microseconds per foot (µs/ft) or microseconds per meter (µs/m), is related to the rock's mechanical properties, such as porosity, lithology, and permeability. In general, higher porosity formations have longer travel times, as sound waves travel slower through fluids-filled pores compared to solid rock. The sonic log can be used in combination with other logs to derive additional information about the formation, such as estimating rock mechanical properties or identifying fractures and faults.(Sonic logging)

The bulk density log measures the bulk density of subsurface formations, which is the combined density of the rock matrix and the fluids within its pores. Bulk density is measured by a tool that emits gamma rays, which are scattered and absorbed by the formation. The tool then detects the remaining gamma rays, and the attenuation of the gamma rays is related to the formation's bulk density. The bulk density log is valuable for determining porosity, lithology, and estimating hydrocarbon saturation. Lower bulk density values often indicate higher porosity or the presence of lighter hydrocarbons, such as gas.(Andersen, The Defining Series: Introduction to Wireline Logging)

The photoelectric log measures the photoelectric absorption index (Pe) of a formation, which is related to the atomic number of the elements in the rock. The logging tool emits low-energy gamma rays, which interact with the formation, causing photoelectric absorption. The absorbed gamma rays produce photoelectrons, and the rate at which these photoelectrons are emitted is proportional to the formation's Pe value. The photoelectric log is particularly useful for identifying different lithologies, as different rock types have distinct Pe values. In combination with other logs, the photoelectric log can help in determining mineral composition, porosity, and fluid content.("The Photoelectric Index")

Resistivity logs measure the electrical resistivity of subsurface formations, which is related to the rock's porosity, fluid content, and saturation. The logging tool sends an electric current into the formation, and the tool measures the voltage drop caused by the formation's resistance to current flow. In general, hydrocarbons are more resistive than water, and higher resistivity values can indicate the presence of hydrocarbon-bearing formations. Resistivity logs are essential for identifying potential reservoirs.(Andersen, The Defining Series: Basic Well Log Interpretation)

In well log interpretation, crossplotting different log measurements can provide valuable insights into the subsurface formations. By comparing the relationships between neutron porosity, sonic, bulk density, photoelectric, and resistivity logs, geoscientists can identify patterns and trends that are indicative of specific rock types, fluid content, and reservoir properties. Combining the information from multiple logs can enhance the accuracy and reliability of the interpretation, enabling more precise determination of lithology, porosity, and hydrocarbon saturation.

Formation evaluation is a key aspect of well log interpretation, as it provides information about the reservoir's potential to contain and produce hydrocarbons. By analyzing the various log measurements, geoscientists can estimate the volume of shale, porosity, and water saturation in the formation. The Archie equation, a widely used empirical relationship, can be employed to estimate hydrocarbon saturation using resistivity, porosity, and water saturation data. Understanding the distribution and saturation of hydrocarbons within the reservoir is essential for making informed decisions about drilling, completion, and production strategies.

Well log interpretation can be further enhanced by integrating log data with other subsurface information, such as core data, seismic data, and production data. Core data, obtained from rock samples extracted from the subsurface, provide a direct measurement of rock properties and can be used to calibrate and validate log-derived interpretations. Seismic data, which provide a detailed image of the subsurface structure, can be integrated with well log data to develop more accurate geological models and improve reservoir characterization. Production data, such as pressure and fluid production rates, can help validate the accuracy of the reservoir properties estimated from well logs.

Despite the advances in well log interpretation techniques, challenges still exist in accurately characterizing complex subsurface formations, particularly in the presence of mixed lithologies, fractures, or thin beds. Machine learning and artificial intelligence (AI) algorithms are increasingly being applied to well log interpretation, offering the potential for more accurate and efficient analysis of complex subsurface environments. By leveraging the power of AI and machine learning, geoscientists can develop more sophisticated models that can better predict reservoir properties, optimize hydrocarbon exploration, and improve overall decision-making in the oil and gas industry.

2.1.1 Bulk Density Log (RHOB)

Limestone Formations: The bulk density of pure limestone is typically around 2.71 g/cm3. Hence, in a bulk density log, you would expect a reading around this value for a clean, non-porous limestone. However, most limestone formations in the subsurface are not pure and contain some porosity. The bulk density reading will decrease as porosity increases, because the pore space is filled with fluids (oil, gas, water) that have a lower density than the rock matrix. The bulk density log can help identify zones of higher porosity (potential reservoir zones) within the limestone, indicated by lower density values. Furthermore, by comparing the bulk density log with other logs such as the neutron log, it's possible to estimate the types of fluid present in the pore space. Shale Formations : In a bulk density log, you would typically see higher readings in shale layers than in adjacent sandstone or limestone layers due to their higher clay content. Anomalously low density in a shale layer could potentially indicate a high organic content, which might be of interest in shale gas or shale oil exploration.

2.1.2 Photoelectric Effect Log (PE)

Limestone Formations: The calcium in limestone gives it a relatively high photoelectric factor. A common value for the photoelectric factor of pure limestone is about 5.0 barns/electron. lower Pe values might indicate the presence of shale or sand interbedded with the limestone.("The Photoelectric Index")

2.1.3 Neutron Porosity Log (NPHI)

The neutron tool used emits high-energy neutrons into the formation. These neutrons collide with hydrogen atoms, slowing them down, and the tool measures the count of these slowed neutrons. The measurement is crucial for porosity estimation and can help in identifying gas-bearing formations.(Neutron porosity logs)

2.1.4 Gamma Ray Log (GR)

Limestone Formations: Limestones are typically formed in low energy depositional environments and usually contain less clay and radioactive material, resulting in lower gamma ray log readings. Limestone formations will usually appear as zones of low gamma ray values on a log. The purity of the limestone can often be inferred from the gamma ray log - a very clean (low clay content) limestone will have a very low gamma ray reading.(Gamma ray logs)

2.1.5 Caliper Log (CALI)

Limestone Formations: Limestone is a dense, compact sedimentary rock composed mostly of calcite. When logging tools pass through these formations, the caliper measurements tend to be quite stable and match the planned borehole diameter if there are no significant issues like washouts, fractures, or major bedding planes. These features can be indicated by sudden changes or spikes in the caliper log values. Shale Formations: Shale formations, composed primarily of clay minerals, tend to be softer than limestone formations and may react with drilling fluids leading to borehole instability. As a result, caliper logs in shale zones often show larger borehole diameters due to washouts. In addition, shale formations with high clay content may swell when exposed to water-based drilling fluids, causing the borehole to constrict. This would be reflected as smaller caliper readings. (Openhole caliper logs)

2.1.6 Sonic Log (DT)

This log is used to determine rock properties such as porosity, lithology, and mechanical properties of the rock. (Sonic logging)

2.1.7 Resistivity Logs (RT10 to RT90)

Resistivity Logs, ranging from RT10 to RT90, measure the electrical resistivity of the formation at various depths. These tools use electrodes to emit currents and measure the voltage drop, which is related to the formation's resistivity. Resistivity is key in identifying hydrocarbon-bearing formations, as hydrocarbons are more resistive than water-filled formations.(Andersen, The Defining Series: Basic Well Log Interpretation)

Each of these logs provides unique insights into the subsurface geological formations, and when combined, they offer a comprehensive understanding of the lithology, porosity, fluid content, and other critical properties of the formations encountered in a borehole.

2.2 Artificial Neural Networks

2.2.1 Introduction to Backpropagation

Back-propagation neural networks, also known as back-propagation or simply back-prop, are a type of feed-forward artificial neural network widely used in machine learning and artificial intelligence applications. Back-propagation is an efficient supervised learning algorithm that adjusts the weights of the network to minimize the error between the predicted output and the actual target values. The key concept behind back-propagation is the use of gradient descent to optimize the weights of the neural network, which allows it to learn complex relationships and generalize from training data to make predictions on unseen data.(Géron)

In a back-propagation neural network, information flows forward from the input layer through one or more hidden layers to the output layer. Each layer consists of interconnected nodes or neurons, which process the incoming information using an activation function. The activation function determines the output of a neuron based on its weighted inputs and biases. Common activation functions include the sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU) functions. These nonlinear activation functions enable the neural network to learn and model complex, non-linear relationships between inputs and outputs.(Géron)

After the feed-forward process, the network's output is compared to the target values, and an error function is calculated. The error function, also known as the loss function, quantifies the difference between the predicted outputs and the actual target values. The goal of the back-propagation algorithm is to minimize this error by adjusting the weights of the network. Common loss functions include mean squared error (MSE), cross-entropy loss, and mean absolute error (MAE). The choice of the loss function depends on the specific problem being addressed and the desired properties of the error measurement.(Géron)

The back-propagation algorithm starts by calculating the gradient of the error function with respect to each weight in the network. This is done by applying the chain rule of calculus to compute the derivatives of the error function with respect to the weights. The calculated gradients are then used to update the weights using the gradient descent optimization algorithm. During each iteration of the learning process, the weights are adjusted in the direction of the negative gradient, which reduces the error function's value. The learning rate, a hyper-parameter, determines the size of the steps taken in the direction of the negative gradient during weight updates.(Géron)

Back-propagation neural networks have been successfully applied in various domains, including image recognition, natural language processing, speech recognition, and control systems. Their ability to learn complex patterns and relationships in data makes them a popular choice for solving challenging machine learning problems. However, back-propagation networks have some limitations, such as the possibility of getting stuck in local minima during optimization, slow convergence, and the need for a large amount of training data. In recent years, advancements in deep learning techniques and architectures, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have addressed some of these limitations and expanded the applicability of neural networks in various fields.(Géron)

2.2.2 Theoretical Foundations

Conceptual Framework

At its core, back-propagation is a method for computing the gradient of the loss function with respect to the weights of the network. This computation is critical for optimizing the weights using gradient-based optimization techniques, notably gradient descent.

Mathematical Underpinnings

The algorithm utilizes the chain rule from calculus to iteratively compute gradients for each layer in the network, starting from the output layer and moving backward towards the input layer. This process involves partial derivatives, reflecting how changes in weights affect the overall error.(Géron)

2.2.3 Network Architecture and Data Flow

Layered Structure

A typical neural network consists of an input layer, several hidden layers, and an output layer. Each layer comprises numerous neurons, and the connections between neurons are associated with weights.

Forward Propagation

Initially, input data is fed forward through the network. Each neuron processes the input by calculating a weighted sum and then applying an activation function to produce an output. This output becomes the input for the next layer.

Error Computation

At the output layer, the network's prediction is compared with the actual target values to calculate the error using a predefined loss function.

2.2.4 The Backpropagation Algorithm

Backward Error Propagation

The calculated error is propagated back through the network. This step involves calculating the gradient of the error with respect to each weight in the network by applying the chain rule.

Gradient Calculation

The gradient, representing the partial derivative of the error with respect to each weight, indicates how much a change in the weight will impact the overall error.

Weight Update

The weights are then adjusted in the opposite direction of the gradient, with the magnitude of adjustment governed by the learning rate. This process is aimed at reducing the error in subsequent iterations.

2.2.5 Optimization and Learning Rate

Gradient Descent Variants

Various forms of gradient descent, like stochastic gradient descent (SGD), batch gradient descent, and mini-batch gradient descent, are employed, each with its characteristics and suitability depending on the size and nature of the dataset.(Keras)

Learning Rate Considerations

The learning rate is a crucial hyperparameter that determines the size of the step taken in the weight space. A rate too high can lead to overshooting the minimum, while a rate too low results in slow convergence.(Keras)

2.2.6 Challenges and Solutions

Vanishing and Exploding Gradients

In deep networks, gradients can become excessively small (vanish) or large (explode), which complicates the training process. Techniques like weight initialization strategies, normalized initialization, and the use of specific activation functions (e.g., ReLU) are employed to mitigate these issues.(Keras)

Overfitting

Overfitting is a common challenge where the model performs well on training data but poorly on unseen data. Regularization techniques, dropout, and early stopping are strategies used to prevent overfitting.

2.2.7 Advanced Considerations

Batch Normalization

Batch normalization is a technique to standardize the inputs to a layer for each mini-batch. This standardization helps combat the internal covariate shift problem, making the network more stable and faster in training.(Keras)

Activation Functions

The choice of activation functions, like sigmoid, tanh, and ReLU, plays a significant role in the effectiveness of backpropagation, influencing how well gradients are propagated through the network.Keras

2.3 Previous Applications of ANN in Well Logging

2.3.1 Using Artificial Intelligence And Machine Learning To Develop Synthetic Well Logs

Alnuaimi, Marwan Mohammed Alnuaimi

Introduction

This research explores the use of artificial intelligence (AI) technologies in oil and gas, particularly for creating synthetic well logs. These logs are crucial in petroleum engineering for evaluating hydrocarbon formations but are often limited by practical difficulties in field data collection. AI, specifically artificial neural networks (ANN), is presented as a solution to supplement missing or incomplete well data. The development of synthetic well logs is significant in analyzing reservoir characteristics in areas lacking complete log sets.

Additionally, the research also touches on the concept of Petroleum Data Analytics in the gas and oil industry, aimed at optimizing production processes. It mentions that mature oil fields often face challenges with missing or incomplete well logs, due to technological changes or equipment failures.

Methodology Used in the Source

This study selects a number of wells that have available logs via LAS files and initially performs an analysis on the location of the reservoirs to gain an understanding on the log measurements and their properties. The wells are then divided into two groups, the first group having all data available, the second having missing data. A set of wells from the first group was used as blind validation. A software was then used to train a neural network using the remaining wells in the first group. Once a model was developed from the first group, this model was then used to generate synthetic logs for the second group which had some values missing.

Key Findings or Contributions

The research highlights the limitations of obtaining data directly from the field in petroleum engineering. It emphasizes the value of artificial intelligence in generating synthetic well logs to analyze reservoir characteristics, especially in regions with incomplete or absent log sets. The study reveals that using neural networks to generate resistivity logs for producer wells based on injector well data is not always effective, particularly when injection is for pressure maintenance in aquifers. It identifies the importance of domain expertise in AI applications for petroleum engineering, especially in selecting relevant attributes and parameters.

2.3.2 Synthetic well logs generation via Recurrent Neural Networks

Zhang, Dongxiao and Chen, Yuntian and Meng, Jin Zhang et al.

Introduction

This section discusses the importance of well logging in reservoir description and evaluation, highlighting the challenges of missing or incomplete log information due to various factors. It introduces the use of machine learning, specifically Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks, as a method for generating synthetic well logs to address these challenges.

Methodology Used in the Source

The paper presents a methodology that employs LSTM networks for synthetic well log generation. It compares this approach with traditional Fully Connected Neural Networks (FCNN), discussing the limitations of FCNN in handling spatial dependencies and their inability to effectively predict long-term series data. The methodology includes network architecture, specific settings, and the experimental application of the LSTM model to both horizontal and vertical well log data.

Key Findings or Contributions

The key findings reveal that LSTM networks provide more accurate synthetic well log generation compared to traditional FCNNs. The study also introduces a cascaded LSTM approach, combining standard LSTM with a cascade system, which shows improved suitability for problems involving multiple series data. The results from real well log data indicate the superior accuracy and feasibility of LSTM methods in synthetic well log generation.

2.3.3 Developing Intelligent Synthetic Logs: Application to Upper Devonian Units in PA

Rolon, Luisa F. Rolon

Introduction

The paper focuses on well logging, a crucial technique in hydrocarbon reservoir production potential analysis. It highlights the problem of incomplete log data and introduces a methodology using artificial neural networks for generating synthetic wireline logs. The study uses data from southern Pennsylvania, aiming to predict reservoir properties where actual logs are absent or incomplete.

Methodology Used in the Source

The methodology involves using artificial neural networks, specifically a General Regression Neural Network, to generate synthetic wireline logs. The study uses data from four wells, including gamma-ray, density, neutron, and resistivity logs. Two exercises were conducted: one using data from all four wells and another using three wells for training and one for validation.

Key Findings or Contributions

The study successfully generated synthetic logs with reasonable accuracy. It found that the best performance was achieved with specific combinations of inputs and outputs, emphasizing the importance of data quality in neural network model development. The study also noted that lithologic heterogeneities in reservoirs do not significantly affect the model's performance.

2.3.4 Applications of Artificial Intelligence (AI) in Petroleum Engineering Problems

Lashari, Shan e Zehra Lashari

Introduction

This section introduces the application of Artificial Intelligence (AI) in various petroleum engineering problems, highlighting the advancement of AI in different sectors, including the oil and gas industry. It outlines the objective of the study, which is to explore different horizons of petroleum engineering using various AI tools.

Methodology Used in the Source

The methodology section describes the use of AI techniques such as fuzzy logic, machine learning, and clustering in petroleum engineering problems. It details the specific AI tools and approaches used in the study, including the application of supervised and unsupervised machine learning for well log analysis and regression analysis for drilling performance prediction.

Key Findings or Contributions

The key findings section summarizes the outcomes of applying AI methods in petroleum engineering. This includes the effectiveness of fuzzy logic in reservoir fluid characterization, the ability of machine learning to automate well log analysis, and the use of regression algorithms for predicting drilling performance.

2.3.5 A Machine Learning and Data-Driven Prediction and Inversion of Reservoir Brittleness from Geophysical Logs and Seismic Signals: A Case Study in Southwest Pennsylvania, Central Appalachian Basin

Ore, Tobi Micheal Ore

Introduction

This section introduces the significance of brittleness in unconventional reservoir exploration, emphasizing the challenges in accurately estimating it. It describes the machine learning approach adopted to predict brittleness using geophysical logs and seismic data, with a focus on the Middle Devonian Marcellus shale in the Appalachian basin.

Methodology Used in the Source

The paper outlines the use of machine learning algorithms, including Support Vector Regression, Gradient Boosting, and Artificial Neural Networks, for predicting brittleness from geophysical logs. It also discusses the application of Texture Model Regression for brittleness inversion from seismic data.

Key Findings or Contributions

The study demonstrates the effectiveness of machine learning models in predicting reservoir brittleness, with Gradient Boosting showing superior performance. It also explores the potential of seismic inversion for brittleness prediction, providing insights into the applicability of these techniques in geophysical problem-solving.

3. Methodology

3.1 Data Collection

The software allows the user to select and import data using a FileBrowser class. The class "FileBrowser" represents a graphical user interface (GUI) component that is designed to facilitate the browsing of files within a directory on a computer, the selection of these files, and their subsequent loading into a database. Built on the QtWidgets.QDialog class of the Qt library, it provides a highly interactive way for users to perform file management tasks. Upon instantiation, the FileBrowser object creates a layout consisting of several interactive widgets. These include two buttons for selecting a folder and toggling the selection of all files within the folder, a text input field for the database name, a table for displaying file information, a button to load the selected data into the database, and a progress bar to track the loading process. Upon selecting a folder, the files within the directory are displayed in the table widget, where each file has an associated checkbox for selection purposes. Users can select or deselect all files simultaneously with the "Select All" button. Once the necessary files are selected and a database name has been provided, the user can load the data into the database by clicking on the "Load Data" button. This action triggers a process in which each selected file's data is read, supporting '.csv', '.xls', and '.las' formats, and then concatenated into a single database. The new database is then saved as a '.csv' file in a "Database Folder" within the current working directory. Progress towards completion of the data loading process is visually represented through updates to the progress bar widget. This process provides the user with real-time feedback on the system's status and task completion.

3.2 Data Exploration

The software provides various data visualization tools through the use of methods, or functions, built into the application's class. The process for viewing data begins by creating a QFileDialog instance with the DontUseNativeDialog option. This flag makes the file dialog use a non-native, and perhaps more versatile, Qt dialog. The dialog prompts the user to "Select File" and allows any file type to be selected, but particularly suggests CSV and Excel files. A series of methods then follows for processing the data to be viewed. The first method modifies a table widget to filter data. It sets the row count to 1 and adds a combo box to each cell in the row, each containing different filtering options. For example, one combo box contains column names from the data, another contains operators for filtering ("<", ">", "==", etc.), and another is a text input for user-specified values. The second method adds a new row to the filter table, identical to the initial row, but only if a certain attribute, presumably a DataFrame, is present in the class instance. The third method clears the filter table by setting its row count to 0, effectively removing all filter options currently present. The fourth method applies the filter to the data. It iterates through the filter table, grabbing the selected options from each cell in each row and constructing a filter from them. If the filter is valid (all necessary parts are present), it adds it to a list of filters. After constructing the list of filters, it checks if the aforementioned DataFrame attribute exists and is not empty. If this is true, it processes the list of filters and applies it to the DataFrame, and then updates the data visualization. The final method processes a list of filter instructions into a valid pandas query, applies it to the DataFrame, and returns the filtered DataFrame. It supports both simple comparison queries (e.g., 'value $>$ 3') and complex queries involving multiple conditions for a single attribute, formatted as a string and passed to the pandas query method. If the condition involves a numerical value, it ensures that the value is treated as a float. It also handles the case where multiple conditions for a single attribute are combined with an OR operation ('|'). Once the data is processed for visualization, another set of methods are used to generate the plots. A scatter plot and log plot will be described as examples.

The first method takes a dataset and updates a variety of GUI elements with the dataset's information. This includes populating a table with the data, and updating several drop-down menus with the dataset's column names. The method also clears any old data from these GUI elements before updating them. The table is populated by first setting its row and column counts to match the dimensions of the dataset. It then iterates over each cell, filling it with the corresponding data value. In the case of the drop-down menus, the method simply clears them and then adds each column name from the dataset as an option. The second method is focused on generating a scatter plot from selected data. It first retrieves the column names chosen for the x and y axes of the scatter plot from the relevant drop-down menus. It checks if these columns exist and if there's data available to plot. If everything is ready, the method proceeds to create a DataFrame for plotting, which excludes any missing values. It then uses this to generate a scatter plot, setting the chosen column names as the x and y axis labels. The created plot is then wrapped in a canvas, which in turn is placed inside a new window. A toolbar is also created for the window, providing navigation controls for the plot. If the required data isn't available or the column names haven't been selected, the method prints a message to the console alerting the user to these issues.

The log plot is generated by a method with the following description. To start, it retrieves the names of two specific columns from combo boxes. These names are used as identifiers in the incoming data. A plot setup is also generated, and the unique track names from the setup are extracted. A figure for the plot is then initiated, and the plotting begins. For each track, a subplot grid is created, with its parameters being set for optimal visualization. The y-axis is inverted, aligning with the typical display for log plots. For each track, the corresponding attributes are obtained. These attributes are used to draw multiple twinned x-axes on the same subplot, each corresponding to a different attribute. The method iterates over these attributes and plots each one on a separate twinned x-axis, with customized color, label, and position. The scale for each plotted line can also be set to logarithmic based on user input. Furthermore, if an annotation is chosen, the method goes on to annotate the plot at specific points with labels, each label being placed at a specific position related to the y-value of the data point it annotates. Once the plot is created, it's encapsulated within a canvas and placed inside a new window. A navigation toolbar is also created for the window, providing controls for the plot. This window is then displayed, presenting the log plot to the user.

3.3 Data Processing

The software allows for data to be processed in various ways via the use of classes. The class "AttributeManager" is a graphical user interface (GUI) component, structured as a QDialog from the Qt library, specifically designed for managing attributes of a database stored as a CSV file. The class provides an intuitive and interactive way for users to modify attributes within a database. Upon instantiation, an AttributeManager object constructs a vertical layout which contains several interactive widgets. These include a button for database selection, a table widget for displaying and editing attribute information, and a button for saving changes made to the attributes. The user begins by selecting a CSV database using the 'Select Database' button. The class loads the selected database into a pandas DataFrame, which is used to display the database's attributes within the table widget. The table presents each attribute in a separate row, and each row contains three cells. The first cell displays the attribute's current name, which is non-editable. The second cell contains a dropdown menu with options to edit the attribute's name or to remove the attribute. The third cell is editable and is used to input a new name for the attribute, if the 'Edit Name' option is selected. Upon clicking the 'Save Changes' button, the changes specified in the table are applied to the DataFrame. This involves iterating over each row in the table, checking the specified action, and applying the corresponding operation to the DataFrame, such as dropping a column or renaming it. If the 'Edit Name' action is chosen and the new name already exists in the DataFrame, the values of the original and renamed attributes are combined, favoring the values from the renamed attribute. Finally, the modified DataFrame is saved back to its original CSV file, and a message box is displayed to confirm the successful saving of the changes. This class therefore provides a powerful tool for users to interactively manage their database attributes in a streamlined and user-friendly manner.

The "AppendData" class is a QDialog subclass from the Qt library that creates a GUI for users to append additional datasets to a main database, with options for selecting specific columns and methods of merging the data. The class also allows users to forward-fill missing values. Upon initialization, an AppendData object creates a vertical layout, housing various interactive elements. These include two buttons for selecting a database and the data to append, dropdown menus for choosing specific columns from the database and dataset, a checkbox for toggling forward-fill, and a button to execute the merge operation. The user starts by selecting a CSV or Excel file as the main database. After reading the selected file into a pandas DataFrame, the class populates the dropdown menus in the "Database Column Section" with column names from the database. The user then selects another CSV or Excel file to append to the main database. Similarly, this file is read into a DataFrame and its column names are used to populate the dropdown menus in the "Dataset Column Section".

The "merge data" method merges the main database and the additional dataset based on user-selected ID and Reference columns from both the database and dataset. The method also ensures that the dataset's target column does not already exist in the main database. If it does, the column in the dataset is renamed before merging. The merging operation is performed using pandas' merge as-of function. This is a type of merge where rows from the left DataFrame (database) are matched to the nearest equal or earlier row from the right DataFrame (dataset), based on the specified merge key (Reference column). This operation sorts both DataFrames by the Reference column and merges on this column. If the "Forward Fill" option is checked, the method forward-fills missing values in the target column. It also creates a factorized column that represents the filled data as integer factors. Finally, the updated database DataFrame is saved back to its original CSV file, and a message box is shown to confirm the successful merge. This class provides a handy and intuitive GUI for users to interactively append data to their main database and perform advanced merge operations.

Data processing also includes the process of partitioning data. This suite of methods helps with selecting, partitioning, and filtering data for a particular use-case. The first method prompts the user to choose a file from their local system through a file dialog. It supports files of different types such as CSV and Excel. Once a file is selected, the method reads the file into a DataFrame and extracts the column names. It then calls another method to set up a filter table with these column names. The second method prepares a filter table with one row initially. The columns in this row are combo boxes and a line edit, which allow the user to select a column name, comparison operator, input a value, and choose a data type, respectively. Another method adds a new row to the filter table with the same configuration as the initial row. It can be called multiple times to add multiple filter conditions. There's also a method to clear the filter table when needed, removing all filter conditions. The next method applies the filters based on the conditions set in the filter table. It extracts the condition from each row and builds a query string. This query string is then used to filter the DataFrame. The filtered data is then passed to another method to update the visual elements, such as plots or tables, based on the filtered data. Finally, there's a method that takes a list of filter conditions and generates a query string that can be used to filter a DataFrame. The method constructs a separate condition for each row in the filter table and then combines all the conditions to form a single query string. This query string is then used to filter the DataFrame and return the filtered data.

Once the data is prepared to be partitioned, a set of methods performs the partitioning. This group of functions work collectively to update UI elements and save partitioned data into different datasets for machine learning purposes. The first function updates a table widget with the filtered data received. The data is presented row by row in a tabular format with the table headers set to the column names of the data. The following three functions update labels of training, calibration, and validation sliders respectively. Whenever a slider's value is changed, the corresponding function gets called to update the label showing the current slider's value. The main function in this group is responsible for saving data partitions. It reads the input values for the partition names and the percentages of data to be used for training, calibration, and validation. It then creates quantile-based bins in the data for stratification. Depending on the percentages specified, it uses stratified splitting if possible to create training, calibration, and validation datasets. The datasets are saved in their respective directories as CSV files. If stratified splitting is not possible due to insufficient bins, it uses simple random sampling to split the data. Each time a partition is saved, it updates the table widgets showing the list of datasets in each category (training, calibration, validation). Finally, there's a function that updates the tables displaying the names of training, calibration, and validation datasets. It does so by listing the CSV files present in each category's directory and updating the corresponding table widget with this list. Once the data is processed and partitioned, it will then be ready to be used for training.

3.4 Training the ANN

The data flows through the network starting from the input layer, passes through the hidden layers, and ends up in the output layer. This is called the feedforward process. At each neuron, an operation called the "activation function" is applied to the incoming data. The value computed for a given neuron is a weighted sum of the outputs of the neurons in the previous layer, plus the bias term, passed through this activation function. Common choices for the activation function include the sigmoid function, the hyperbolic tangent function, the rectified linear unit (ReLU) function, and others. The essence of learning in an ANN involves iteratively adjusting the weights and biases to minimize the difference between the network's predictions and the actual target values. This difference is quantified by a loss function, which the network seeks to minimize. The most common technique for this is called "backpropagation". It starts with a forward pass where input data is passed through the network to compute the output. Then, the loss (or error) is calculated by comparing this output to the target values. In the backward pass, the derivative of the loss with respect to each weight and bias in the network is calculated using the chain rule for differentiation. This process effectively determines how much each weight and bias contributes to the total error. This information, commonly referred to as the gradient, is then used in an optimization algorithm (often some variant of gradient descent) to adjust the weights and biases. The adjustments are made in such a way as to reduce the error: parameters contributing more to the error are adjusted more heavily. This process of forward pass, calculating loss, backpropagation, and adjusting weights and biases is typically repeated for many iterations. An iteration where the network has processed all training samples once is known as an epoch. The network usually trains over multiple epochs, gradually learning the underlying patterns in the data. (Géron)

Input Layer:

The input data is first passed to the input layer of the neural network. The input layer doesn't do any processing; its job is to distribute the input data to the neurons in the next layer. Each neuron in the input layer represents a feature of the data. For instance, in an image recognition task, a neural network might have one input neuron for each pixel in the input image. Géron

Weighted Summation:

The data from the input layer is then passed to the first hidden layer. For each neuron in this layer, a weighted sum is computed. This involves multiplying each input value by the corresponding weight, and then summing these products together. An additional bias term is then added to this sum. The weights and biases are parameters that the network has learned during training. Géron

Weights:

The weights are parameters associated with the connections between neurons in the network. Each connection from one neuron to another has a corresponding weight. These weights effectively determine how strongly the output of one neuron influences the input of the next.

In the context of the network's structure, each layer has its own weight matrix. The size of this matrix is determined by the number of neurons in the current layer and the number of neurons in the previous layer. If layer 'm' has 'n' neurons and layer 'm+1' has 'p' neurons, then the weight matrix for layer 'm+1' will be of size 'p x n'. Each entry in this matrix corresponds to the weight for the connection between a neuron in layer 'm' and a neuron in layer 'm+1'.

During the forward pass, the weighted sum of the inputs is computed for each neuron, which is then passed through the activation function. The weights play a vital role in this computation. During backpropagation, these weights are updated to minimize the loss. (Géron)

Biases:

The bias parameters are an additional set of parameters that are added to the weighted sum of inputs for each neuron. Each neuron has one bias associated with it.

Biases are crucial because they allow the activation function to be shifted to the left or right, which can be critical for successful learning. This shift can help the network model patterns that would not be possible if the activation function were centered at zero.

Like the weights, the biases are learned during training and updated during backpropagation to reduce the loss. The bias can be thought of as the parameter that 'activates' a neuron even when all its inputs are zero. It helps each neuron make more complex and non-linear decisions. (Géron)

Activation Function:

The output of each neuron is then passed through an activation function. The activation function is a non-linear function that helps the neural network learn complex patterns. It's chosen at the time of designing the neural network and commonly used activation functions include the ReLU (Rectified Linear Unit), sigmoid, and tanh. (Géron)

Propagation Through the Network:

This process of weighted summation and applying the activation function is then repeated for every layer in the network, from the first hidden layer to the output layer. The data flows through the network, being transformed at each layer. This is why it's called a "feedforward" process. (Géron)

Output Layer:

Once the data reaches the output layer, it may be transformed one last time depending on the task the network is designed to perform. For a binary classification task, for example, a sigmoid activation function might be used to ensure the output is between 0 and 1, representing a probability. For a multi-class classification task, a softmax function might be used, which gives a set of outputs that sum to 1, representing a probability distribution over the classes. (Géron)

Generating Predictions:

The final output of the network represents its prediction given the input data. For a regression task, this output is directly the predicted value. For a classification task, the output neuron or neurons with the highest activation determines the predicted class or classes. (Géron)

For this case study, the overall goal is to be able to use a small number of seed logs, those being caliper and gamma ray, along with depth and location, to generate a predicted suite of logs. This process is done by developing models for each log that needs to be predicted. In this case, a model was developed for neutron porosity, sonic, bulk density, photoelectric, and resistivity.

For the neutron porosity model, depth, location, caliper, and gamma ray logs were used as inputs. The lithology consists of shale and limestone, so for this model, we are relying heavily on the relationships between neutron porosity and gamma ray.

For the sonic log, while there is a relationship between it and caliper and gamma ray, it is more abstract. The measurements of these log themselves are not indicative of a particular formation or formation characteristic. In other words, the same measurements are possible for different lithologies, but with the inclusion of neutron porosity, we are able to narrow the possibilities.

The subsequent logs to be generate for this general logical premise. In general, all the logs are needed to make accurate determinations of the formation, but since the logs measure different physical properties of the formation, the patterns between them will vary significantly. This more or less determines the order in which the logs are generated.

3.4.1 Final Model Design

The model used to generate the synthetic logs has the following hyper-parameters.

- Loss Function: huber (MAE and MSE)
- Loss Function Delta: 0.64
- Input Activation Function: relu
- Output Activation Function: linear
- Initial Learning Rate: 0.001
- Reduce On Plateau: TRUE
- Reduction Factor: 0.95
- Minimum Learning Rate: 1.00E-05

- Learning Rate Patience: 100
- Batch Size: 64
- L1 Factor: 0
- L2 Factor: 0
- Early Stop Patience: 250
- Number of Epochs: 2000
- Structure Name: 2 x 100
- Optimizer: Adam
- Weight Initialization: GlorotNormal
- Batch Normalization: TRUE
- Bias: FALSE

3.5 Deploying the Model

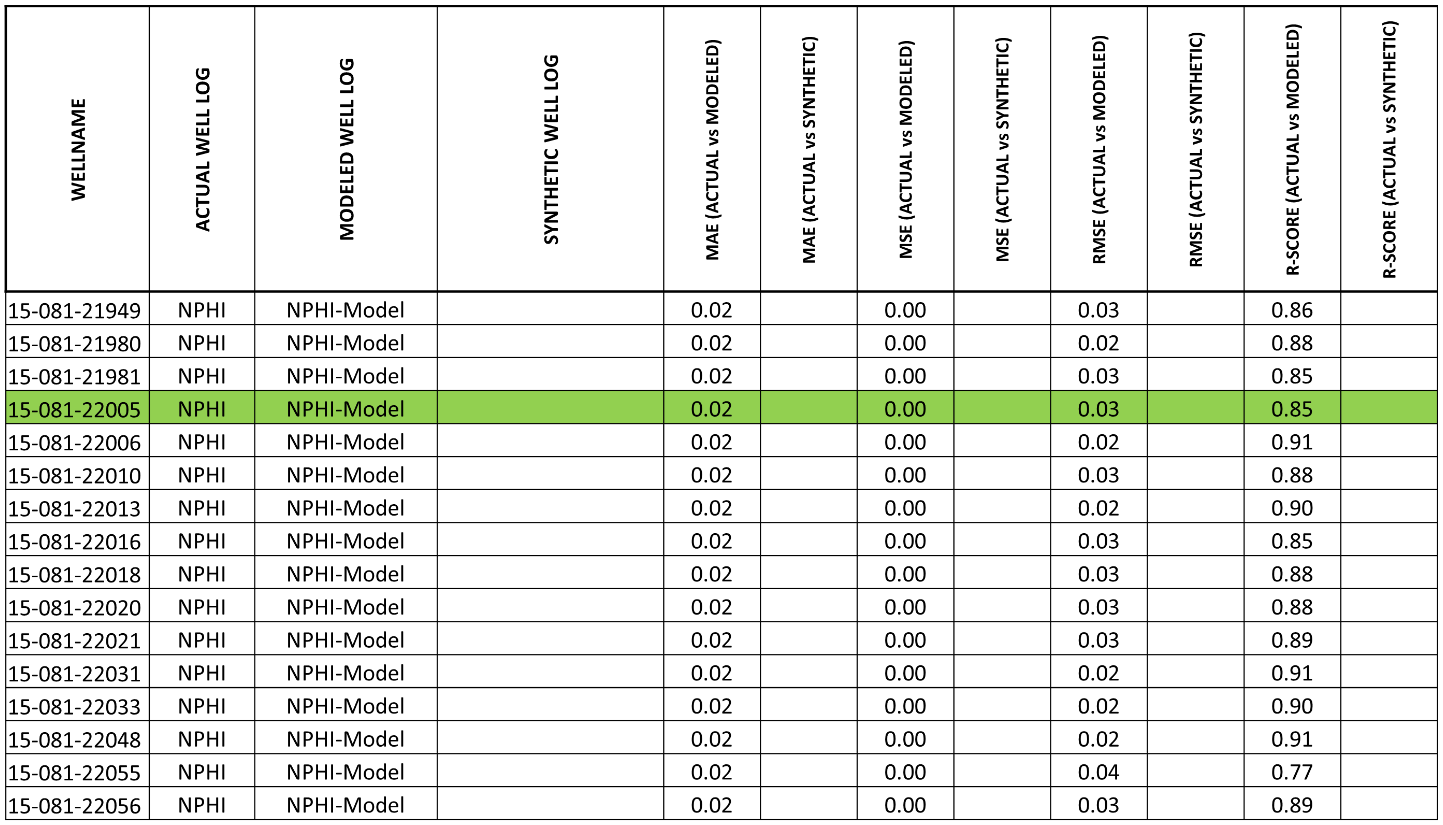
The process for deploying a model begins with a user interface that allows the user to select a data file. This file can be in various formats, such as CSV or Excel. Once a file is selected, it's loaded into the system. The content and structure of the file are then read, specifically focusing on the column names and data contained within. A user interface component, a table with dropdown menus, is presented. This component allows the user to match or map the columns from the loaded data to expected data fields or formats. The user selects corresponding columns from their data that align with the expected format or requirements of an existing system or model. The system processes the user's selections, mapping the original data to a new structure that aligns with the predefined requirements. This newly structured data is then run through a machine learning model. This involves transforming the data as required by the model, making predictions, and then converting the predictions into a usable format. The results from the machine learning model, which are predictions or processed data, are integrated back into the original data set, adding a new layer of information. This updated dataset, now enriched with the model's outputs, is then saved as a new file and available for further use or analysis.

4. Results and Discussion

4.1 Analysis of Results

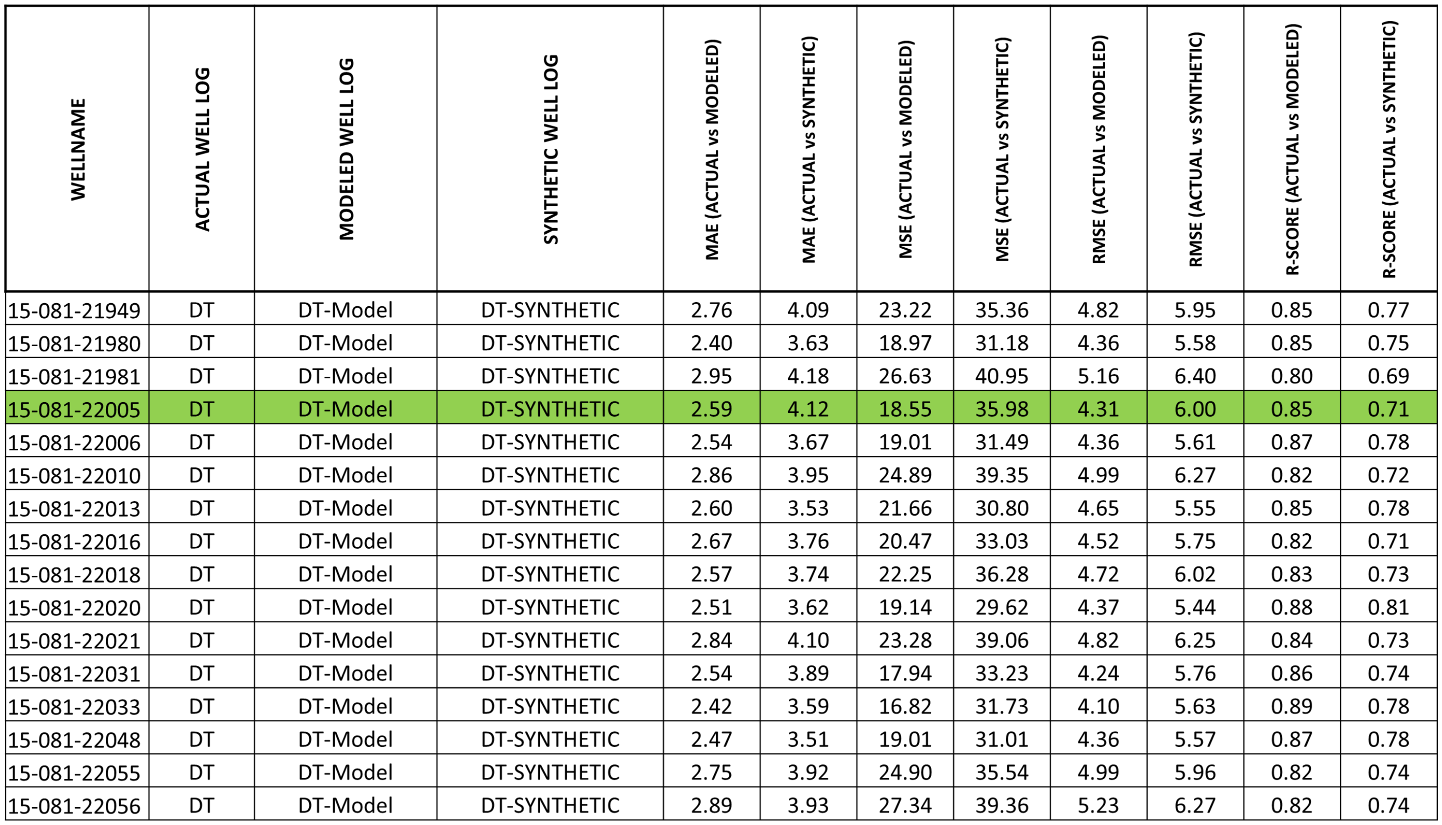
The results show that given a model that is accurate enough, well logs can be sequentially generated with enough accuracy to capture formation characteristics. What we can see from various metrics regarding the accuracy of the modeled logs and synthetic logs is that there is error propagation that occurs. The models are relatively accurate, so the first two generated logs follow the actual log closely, but as each subsequent log is generated, its accuracy decreases with respect to its model counterpart. We also see that the software provides the user with all the necessary tools to accomplish the task. The software allows the user to import data from a variety of formats, visualize the data and save images, train and deploy the model, and finally visualize the results of the model.

In the figures below, the green higlighted well represents the blind test well.



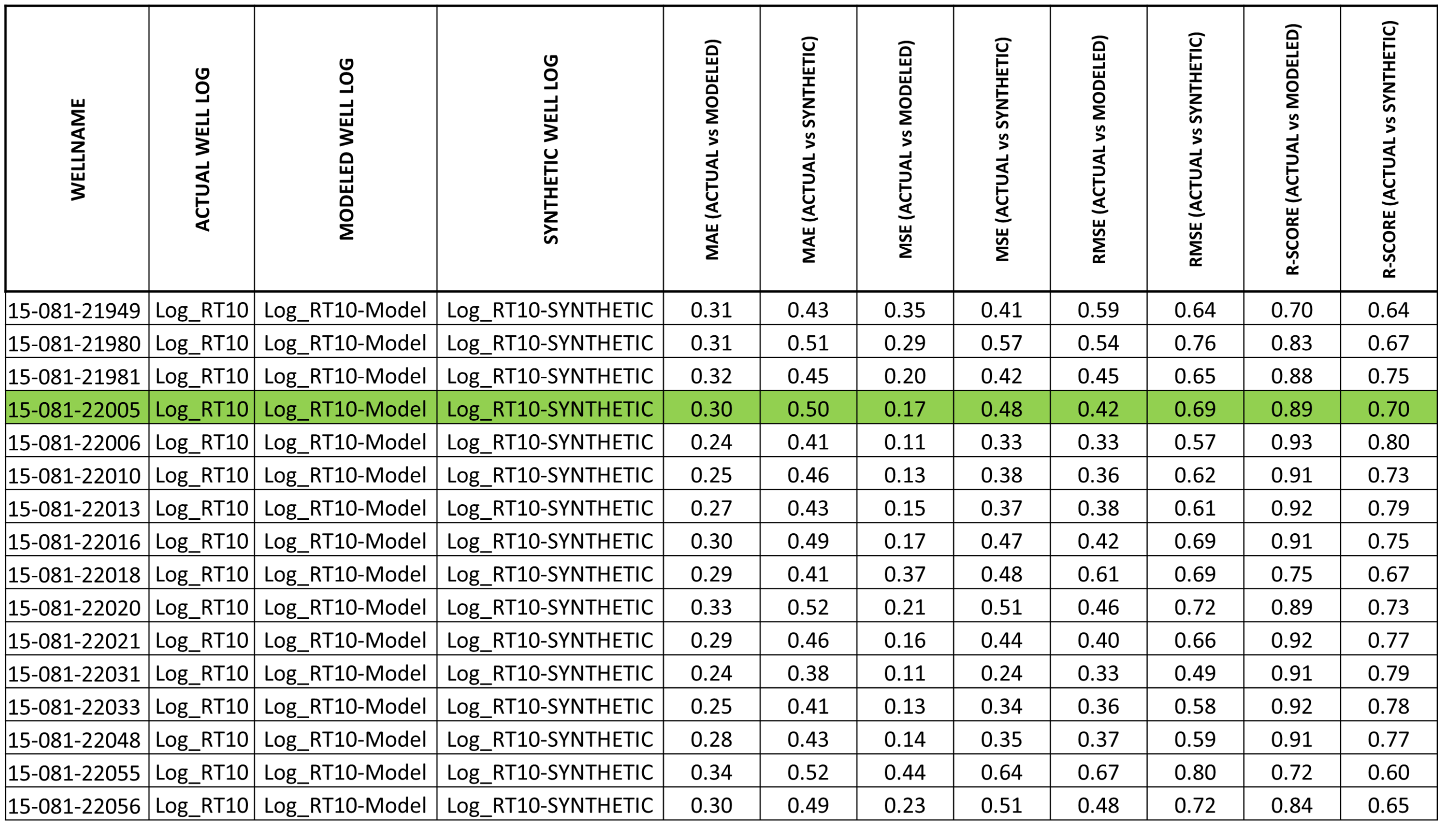
WELLNAME	ACTUAL WELL LOG	MODELED WELL LOG	SYNTHETIC WELL LOG	MAE (ACTUAL vs MODELED)	MAE (ACTUAL vs SYNTHETIC)	MSE (ACTUAL vs MODELED)	MSE (ACTUAL vs SYNTHETIC)	RMSE (ACTUAL vs MODELED)	RMSE (ACTUAL vs SYNTHETIC)	R-SCORE (ACTUAL vs MODELED)	R-SCORE (ACTUAL vs SYNTHETIC)
15-081-21949	NPHI	NPHI-Model		0.02		0.00		0.03		0.86	
15-081-21980	NPHI	NPHI-Model		0.02		0.00		0.02		0.88	
15-081-21981	NPHI	NPHI-Model		0.02		0.00		0.03		0.85	
15-081-22005	NPHI	NPHI-Model		0.02		0.00		0.03		0.85	
15-081-22006	NPHI	NPHI-Model		0.02		0.00		0.02		0.91	
15-081-22010	NPHI	NPHI-Model		0.02		0.00		0.03		0.88	
15-081-22013	NPHI	NPHI-Model		0.02		0.00		0.02		0.90	
15-081-22016	NPHI	NPHI-Model		0.02		0.00		0.03		0.85	
15-081-22018	NPHI	NPHI-Model		0.02		0.00		0.03		0.88	
15-081-22020	NPHI	NPHI-Model		0.02		0.00		0.03		0.88	
15-081-22021	NPHI	NPHI-Model		0.02		0.00		0.03		0.89	
15-081-22031	NPHI	NPHI-Model		0.02		0.00		0.02		0.91	
15-081-22033	NPHI	NPHI-Model		0.02		0.00		0.02		0.90	
15-081-22048	NPHI	NPHI-Model		0.02		0.00		0.02		0.91	
15-081-22055	NPHI	NPHI-Model		0.02		0.00		0.04		0.77	
15-081-22056	NPHI	NPHI-Model		0.02		0.00		0.03		0.89	

Figure 4.1: The overall results of the NPHI Model.



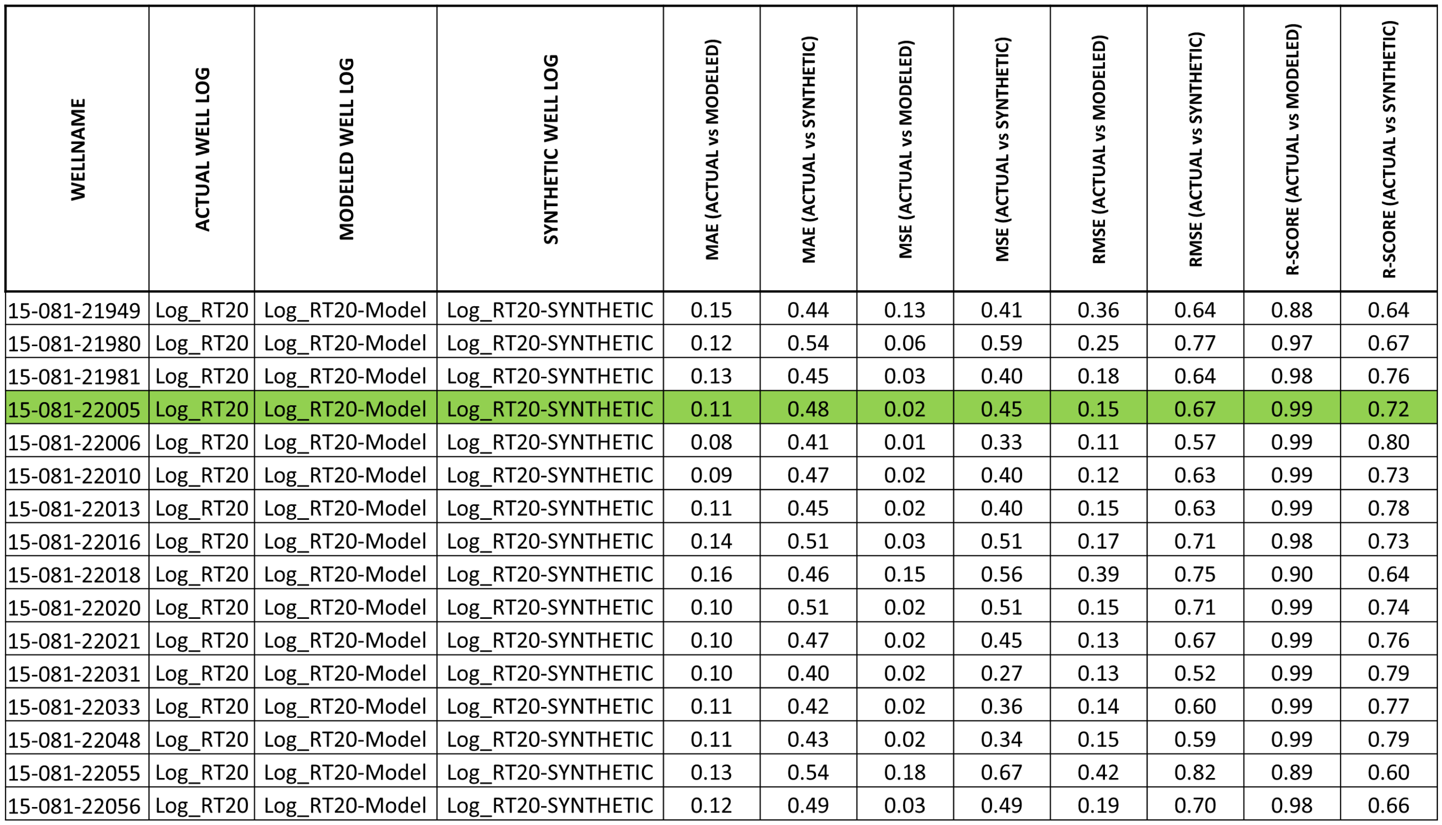
WELLNAME	ACTUAL WELL LOG	MODELED WELL LOG	SYNTHETIC WELL LOG	MAE (ACTUAL vs MODELED)	MAE (ACTUAL vs SYNTHETIC)	MSE (ACTUAL vs MODELED)	MSE (ACTUAL vs SYNTHETIC)	RMSE (ACTUAL vs MODELED)	RMSE (ACTUAL vs SYNTHETIC)	R-SCORE (ACTUAL vs MODELED)	R-SCORE (ACTUAL vs SYNTHETIC)
15-081-21949	DT	DT-Model	DT-SYNTHETIC	2.76	4.09	23.22	35.36	4.82	5.95	0.85	0.77
15-081-21980	DT	DT-Model	DT-SYNTHETIC	2.40	3.63	18.97	31.18	4.36	5.58	0.85	0.75
15-081-21981	DT	DT-Model	DT-SYNTHETIC	2.95	4.18	26.63	40.95	5.16	6.40	0.80	0.69
15-081-22005	DT	DT-Model	DT-SYNTHETIC	2.59	4.12	18.55	35.98	4.31	6.00	0.85	0.71
15-081-22006	DT	DT-Model	DT-SYNTHETIC	2.54	3.67	19.01	31.49	4.36	5.61	0.87	0.78
15-081-22010	DT	DT-Model	DT-SYNTHETIC	2.86	3.95	24.89	39.35	4.99	6.27	0.82	0.72
15-081-22013	DT	DT-Model	DT-SYNTHETIC	2.60	3.53	21.66	30.80	4.65	5.55	0.85	0.78
15-081-22016	DT	DT-Model	DT-SYNTHETIC	2.67	3.76	20.47	33.03	4.52	5.75	0.82	0.71
15-081-22018	DT	DT-Model	DT-SYNTHETIC	2.57	3.74	22.25	36.28	4.72	6.02	0.83	0.73
15-081-22020	DT	DT-Model	DT-SYNTHETIC	2.51	3.62	19.14	29.62	4.37	5.44	0.88	0.81
15-081-22021	DT	DT-Model	DT-SYNTHETIC	2.84	4.10	23.28	39.06	4.82	6.25	0.84	0.73
15-081-22031	DT	DT-Model	DT-SYNTHETIC	2.54	3.89	17.94	33.23	4.24	5.76	0.86	0.74
15-081-22033	DT	DT-Model	DT-SYNTHETIC	2.42	3.59	16.82	31.73	4.10	5.63	0.89	0.78
15-081-22048	DT	DT-Model	DT-SYNTHETIC	2.47	3.51	19.01	31.01	4.36	5.57	0.87	0.78
15-081-22055	DT	DT-Model	DT-SYNTHETIC	2.75	3.92	24.90	35.54	4.99	5.96	0.82	0.74
15-081-22056	DT	DT-Model	DT-SYNTHETIC	2.89	3.93	27.34	39.36	5.23	6.27	0.82	0.74

Figure 4.2: The overall results of the DT Model.



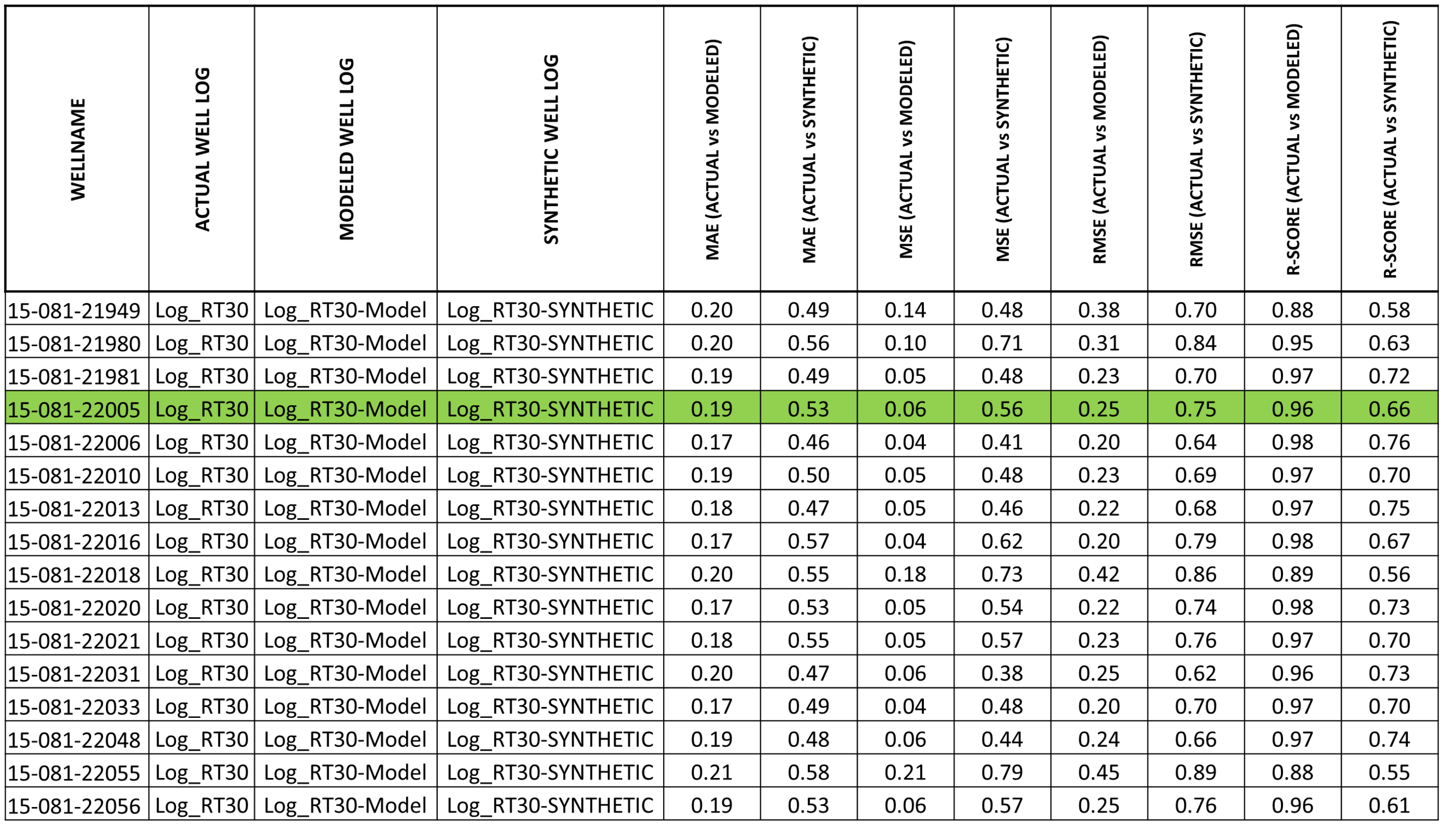
WELLNAME	ACTUAL WELL LOG	MODELED WELL LOG	SYNTHETIC WELL LOG	MAE (ACTUAL vs MODELED)	MAE (ACTUAL vs SYNTHETIC)	MSE (ACTUAL vs MODELED)	MSE (ACTUAL vs SYNTHETIC)	RMSE (ACTUAL vs MODELED)	RMSE (ACTUAL vs SYNTHETIC)	R-SCORE (ACTUAL vs MODELED)	R-SCORE (ACTUAL vs SYNTHETIC)
15-081-21949	Log_RT10	Log_RT10-Model	Log_RT10-SYNTHETIC	0.31	0.43	0.35	0.41	0.59	0.64	0.70	0.64
15-081-21980	Log_RT10	Log_RT10-Model	Log_RT10-SYNTHETIC	0.31	0.51	0.29	0.57	0.54	0.76	0.83	0.67
15-081-21981	Log_RT10	Log_RT10-Model	Log_RT10-SYNTHETIC	0.32	0.45	0.20	0.42	0.45	0.65	0.88	0.75
15-081-22005	Log_RT10	Log_RT10-Model	Log_RT10-SYNTHETIC	0.30	0.50	0.17	0.48	0.42	0.69	0.89	0.70
15-081-22006	Log_RT10	Log_RT10-Model	Log_RT10-SYNTHETIC	0.24	0.41	0.11	0.33	0.33	0.57	0.93	0.80
15-081-22010	Log_RT10	Log_RT10-Model	Log_RT10-SYNTHETIC	0.25	0.46	0.13	0.38	0.36	0.62	0.91	0.73
15-081-22013	Log_RT10	Log_RT10-Model	Log_RT10-SYNTHETIC	0.27	0.43	0.15	0.37	0.38	0.61	0.92	0.79
15-081-22016	Log_RT10	Log_RT10-Model	Log_RT10-SYNTHETIC	0.30	0.49	0.17	0.47	0.42	0.69	0.91	0.75
15-081-22018	Log_RT10	Log_RT10-Model	Log_RT10-SYNTHETIC	0.29	0.41	0.37	0.48	0.61	0.69	0.75	0.67
15-081-22020	Log_RT10	Log_RT10-Model	Log_RT10-SYNTHETIC	0.33	0.52	0.21	0.51	0.46	0.72	0.89	0.73
15-081-22021	Log_RT10	Log_RT10-Model	Log_RT10-SYNTHETIC	0.29	0.46	0.16	0.44	0.40	0.66	0.92	0.77
15-081-22031	Log_RT10	Log_RT10-Model	Log_RT10-SYNTHETIC	0.24	0.38	0.11	0.24	0.33	0.49	0.91	0.79
15-081-22033	Log_RT10	Log_RT10-Model	Log_RT10-SYNTHETIC	0.25	0.41	0.13	0.34	0.36	0.58	0.92	0.78
15-081-22048	Log_RT10	Log_RT10-Model	Log_RT10-SYNTHETIC	0.28	0.43	0.14	0.35	0.37	0.59	0.91	0.77
15-081-22055	Log_RT10	Log_RT10-Model	Log_RT10-SYNTHETIC	0.34	0.52	0.44	0.64	0.67	0.80	0.72	0.60
15-081-22056	Log_RT10	Log_RT10-Model	Log_RT10-SYNTHETIC	0.30	0.49	0.23	0.51	0.48	0.72	0.84	0.65

Figure 4.3: The overall results of the RT10 Model.



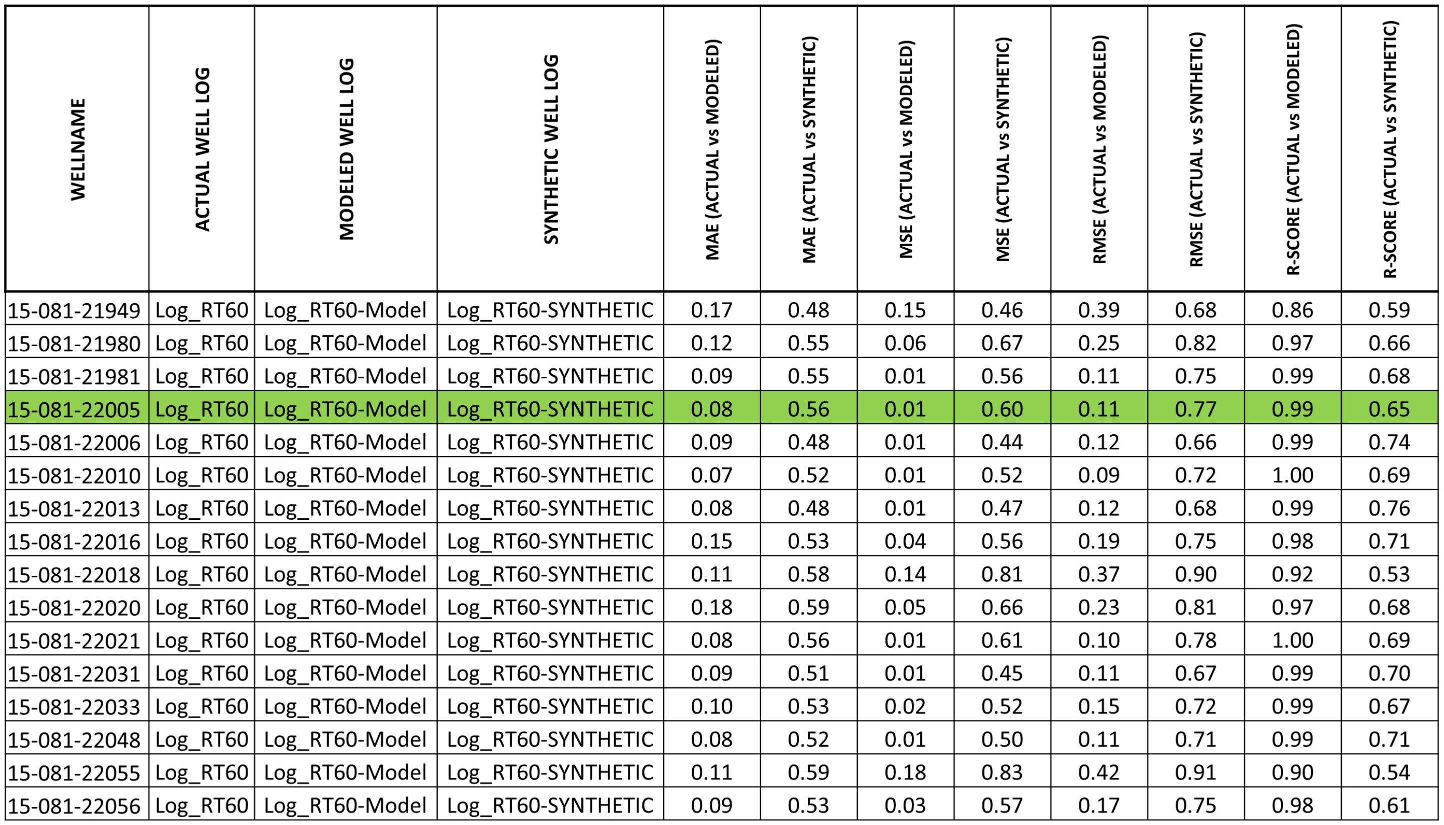
WELLNAME	ACTUAL WELL LOG	MODELED WELL LOG	SYNTHETIC WELL LOG	MAE (ACTUAL vs MODELED)	MAE (ACTUAL vs SYNTHETIC)	MSE (ACTUAL vs MODELED)	MSE (ACTUAL vs SYNTHETIC)	RMSE (ACTUAL vs MODELED)	RMSE (ACTUAL vs SYNTHETIC)	R-SCORE (ACTUAL vs MODELED)	R-SCORE (ACTUAL vs SYNTHETIC)
15-081-21949	Log_RT20	Log_RT20-Model	Log_RT20-SYNTHETIC	0.15	0.44	0.13	0.41	0.36	0.64	0.88	0.64
15-081-21980	Log_RT20	Log_RT20-Model	Log_RT20-SYNTHETIC	0.12	0.54	0.06	0.59	0.25	0.77	0.97	0.67
15-081-21981	Log_RT20	Log_RT20-Model	Log_RT20-SYNTHETIC	0.13	0.45	0.03	0.40	0.18	0.64	0.98	0.76
15-081-22005	Log_RT20	Log_RT20-Model	Log_RT20-SYNTHETIC	0.11	0.48	0.02	0.45	0.15	0.67	0.99	0.72
15-081-22006	Log_RT20	Log_RT20-Model	Log_RT20-SYNTHETIC	0.08	0.41	0.01	0.33	0.11	0.57	0.99	0.80
15-081-22010	Log_RT20	Log_RT20-Model	Log_RT20-SYNTHETIC	0.09	0.47	0.02	0.40	0.12	0.63	0.99	0.73
15-081-22013	Log_RT20	Log_RT20-Model	Log_RT20-SYNTHETIC	0.11	0.45	0.02	0.40	0.15	0.63	0.99	0.78
15-081-22016	Log_RT20	Log_RT20-Model	Log_RT20-SYNTHETIC	0.14	0.51	0.03	0.51	0.17	0.71	0.98	0.73
15-081-22018	Log_RT20	Log_RT20-Model	Log_RT20-SYNTHETIC	0.16	0.46	0.15	0.56	0.39	0.75	0.90	0.64
15-081-22020	Log_RT20	Log_RT20-Model	Log_RT20-SYNTHETIC	0.10	0.51	0.02	0.51	0.15	0.71	0.99	0.74
15-081-22021	Log_RT20	Log_RT20-Model	Log_RT20-SYNTHETIC	0.10	0.47	0.02	0.45	0.13	0.67	0.99	0.76
15-081-22031	Log_RT20	Log_RT20-Model	Log_RT20-SYNTHETIC	0.10	0.40	0.02	0.27	0.13	0.52	0.99	0.79
15-081-22033	Log_RT20	Log_RT20-Model	Log_RT20-SYNTHETIC	0.11	0.42	0.02	0.36	0.14	0.60	0.99	0.77
15-081-22048	Log_RT20	Log_RT20-Model	Log_RT20-SYNTHETIC	0.11	0.43	0.02	0.34	0.15	0.59	0.99	0.79
15-081-22055	Log_RT20	Log_RT20-Model	Log_RT20-SYNTHETIC	0.13	0.54	0.18	0.67	0.42	0.82	0.89	0.60
15-081-22056	Log_RT20	Log_RT20-Model	Log_RT20-SYNTHETIC	0.12	0.49	0.03	0.49	0.19	0.70	0.98	0.66

Figure 4.4: The overall results of the RT20 Model.



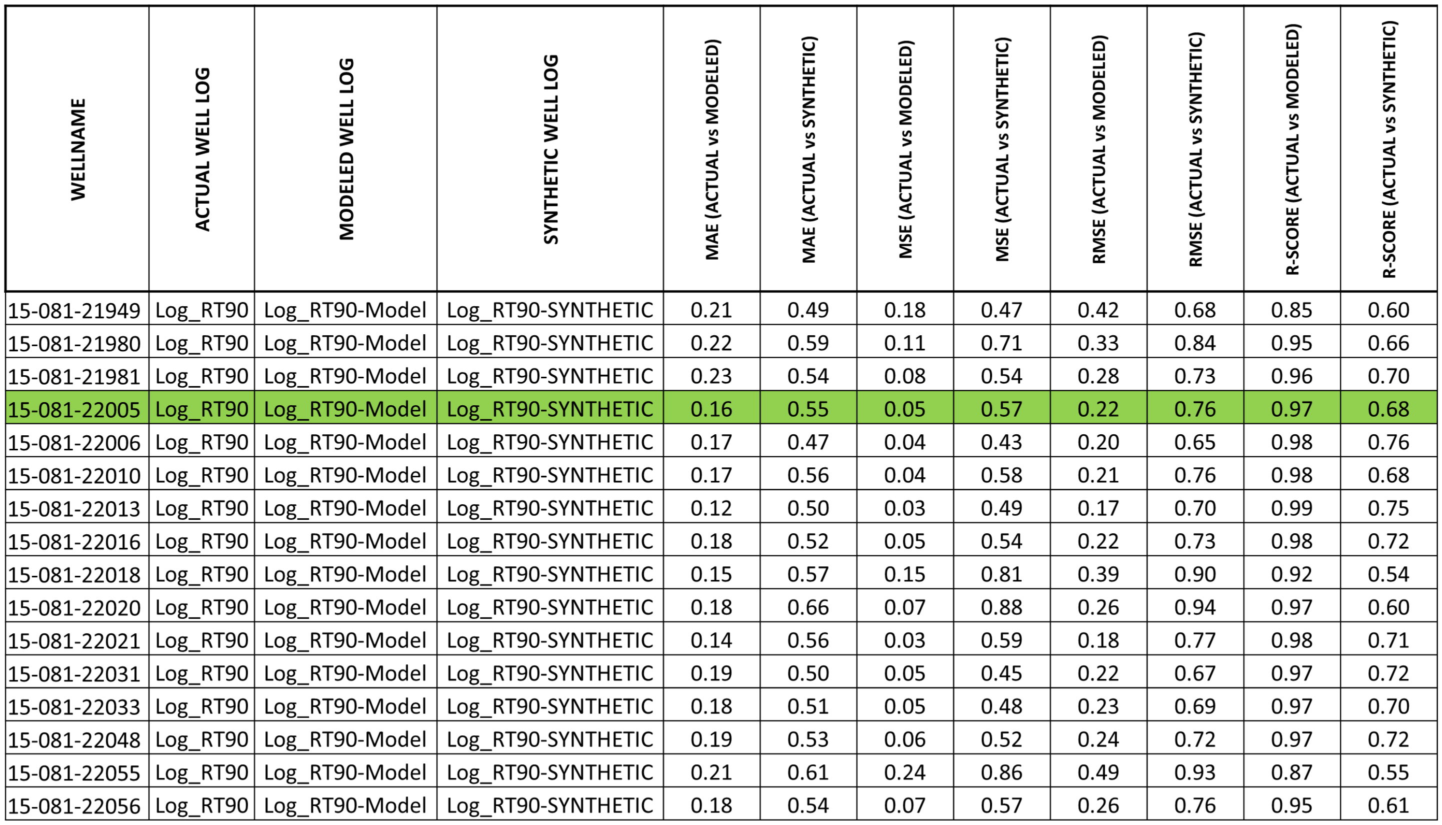
WELLNAME	ACTUAL WELL LOG	MODELED WELL LOG	SYNTHETIC WELL LOG	MAE (ACTUAL vs MODELED)	MAE (ACTUAL vs SYNTHETIC)	MSE (ACTUAL vs MODELED)	MSE (ACTUAL vs SYNTHETIC)	RMSE (ACTUAL vs MODELED)	RMSE (ACTUAL vs SYNTHETIC)	R-SCORE (ACTUAL vs MODELED)	R-SCORE (ACTUAL vs SYNTHETIC)
15-081-21949	Log_RT30	Log_RT30-Model	Log_RT30-SYNTHETIC	0.20	0.49	0.14	0.48	0.38	0.70	0.88	0.58
15-081-21980	Log_RT30	Log_RT30-Model	Log_RT30-SYNTHETIC	0.20	0.56	0.10	0.71	0.31	0.84	0.95	0.63
15-081-21981	Log_RT30	Log_RT30-Model	Log_RT30-SYNTHETIC	0.19	0.49	0.05	0.48	0.23	0.70	0.97	0.72
15-081-22005	Log_RT30	Log_RT30-Model	Log_RT30-SYNTHETIC	0.19	0.53	0.06	0.56	0.25	0.75	0.96	0.66
15-081-22006	Log_RT30	Log_RT30-Model	Log_RT30-SYNTHETIC	0.17	0.46	0.04	0.41	0.20	0.64	0.98	0.76
15-081-22010	Log_RT30	Log_RT30-Model	Log_RT30-SYNTHETIC	0.19	0.50	0.05	0.48	0.23	0.69	0.97	0.70
15-081-22013	Log_RT30	Log_RT30-Model	Log_RT30-SYNTHETIC	0.18	0.47	0.05	0.46	0.22	0.68	0.97	0.75
15-081-22016	Log_RT30	Log_RT30-Model	Log_RT30-SYNTHETIC	0.17	0.57	0.04	0.62	0.20	0.79	0.98	0.67
15-081-22018	Log_RT30	Log_RT30-Model	Log_RT30-SYNTHETIC	0.20	0.55	0.18	0.73	0.42	0.86	0.89	0.56
15-081-22020	Log_RT30	Log_RT30-Model	Log_RT30-SYNTHETIC	0.17	0.53	0.05	0.54	0.22	0.74	0.98	0.73
15-081-22021	Log_RT30	Log_RT30-Model	Log_RT30-SYNTHETIC	0.18	0.55	0.05	0.57	0.23	0.76	0.97	0.70
15-081-22031	Log_RT30	Log_RT30-Model	Log_RT30-SYNTHETIC	0.20	0.47	0.06	0.38	0.25	0.62	0.96	0.73
15-081-22033	Log_RT30	Log_RT30-Model	Log_RT30-SYNTHETIC	0.17	0.49	0.04	0.48	0.20	0.70	0.97	0.70
15-081-22048	Log_RT30	Log_RT30-Model	Log_RT30-SYNTHETIC	0.19	0.48	0.06	0.44	0.24	0.66	0.97	0.74
15-081-22055	Log_RT30	Log_RT30-Model	Log_RT30-SYNTHETIC	0.21	0.58	0.21	0.79	0.45	0.89	0.88	0.55
15-081-22056	Log_RT30	Log_RT30-Model	Log_RT30-SYNTHETIC	0.19	0.53	0.06	0.57	0.25	0.76	0.96	0.61

Figure 4.5: The overall results of the RT30 Model.



WELLNAME	ACTUAL WELL LOG	MODELED WELL LOG	SYNTHETIC WELL LOG	MAE (ACTUAL vs MODELED)	MAE (ACTUAL vs SYNTHETIC)	MSE (ACTUAL vs MODELED)	MSE (ACTUAL vs SYNTHETIC)	RMSE (ACTUAL vs MODELED)	RMSE (ACTUAL vs SYNTHETIC)	R-SCORE (ACTUAL vs MODELED)	R-SCORE (ACTUAL vs SYNTHETIC)
15-081-21949	Log_RT60	Log_RT60-Model	Log_RT60-SYNTHETIC	0.17	0.48	0.15	0.46	0.39	0.68	0.86	0.59
15-081-21980	Log_RT60	Log_RT60-Model	Log_RT60-SYNTHETIC	0.12	0.55	0.06	0.67	0.25	0.82	0.97	0.66
15-081-21981	Log_RT60	Log_RT60-Model	Log_RT60-SYNTHETIC	0.09	0.55	0.01	0.56	0.11	0.75	0.99	0.68
15-081-22005	Log_RT60	Log_RT60-Model	Log_RT60-SYNTHETIC	0.08	0.56	0.01	0.60	0.11	0.77	0.99	0.65
15-081-22006	Log_RT60	Log_RT60-Model	Log_RT60-SYNTHETIC	0.09	0.48	0.01	0.44	0.12	0.66	0.99	0.74
15-081-22010	Log_RT60	Log_RT60-Model	Log_RT60-SYNTHETIC	0.07	0.52	0.01	0.52	0.09	0.72	1.00	0.69
15-081-22013	Log_RT60	Log_RT60-Model	Log_RT60-SYNTHETIC	0.08	0.48	0.01	0.47	0.12	0.68	0.99	0.76
15-081-22016	Log_RT60	Log_RT60-Model	Log_RT60-SYNTHETIC	0.15	0.53	0.04	0.56	0.19	0.75	0.98	0.71
15-081-22018	Log_RT60	Log_RT60-Model	Log_RT60-SYNTHETIC	0.11	0.58	0.14	0.81	0.37	0.90	0.92	0.53
15-081-22020	Log_RT60	Log_RT60-Model	Log_RT60-SYNTHETIC	0.18	0.59	0.05	0.66	0.23	0.81	0.97	0.68
15-081-22021	Log_RT60	Log_RT60-Model	Log_RT60-SYNTHETIC	0.08	0.56	0.01	0.61	0.10	0.78	1.00	0.69
15-081-22031	Log_RT60	Log_RT60-Model	Log_RT60-SYNTHETIC	0.09	0.51	0.01	0.45	0.11	0.67	0.99	0.70
15-081-22033	Log_RT60	Log_RT60-Model	Log_RT60-SYNTHETIC	0.10	0.53	0.02	0.52	0.15	0.72	0.99	0.67
15-081-22048	Log_RT60	Log_RT60-Model	Log_RT60-SYNTHETIC	0.08	0.52	0.01	0.50	0.11	0.71	0.99	0.71
15-081-22055	Log_RT60	Log_RT60-Model	Log_RT60-SYNTHETIC	0.11	0.59	0.18	0.83	0.42	0.91	0.90	0.54
15-081-22056	Log_RT60	Log_RT60-Model	Log_RT60-SYNTHETIC	0.09	0.53	0.03	0.57	0.17	0.75	0.98	0.61

Figure 4.6: The overall results of the RT60 Model.



WELLNAME	ACTUAL WELL LOG	MODELED WELL LOG	SYNTHETIC WELL LOG	MAE (ACTUAL vs MODELED)	MAE (ACTUAL vs SYNTHETIC)	MSE (ACTUAL vs MODELED)	MSE (ACTUAL vs SYNTHETIC)	RMSE (ACTUAL vs MODELED)	RMSE (ACTUAL vs SYNTHETIC)	R-SCORE (ACTUAL vs MODELED)	R-SCORE (ACTUAL vs SYNTHETIC)
15-081-21949	Log_RT90	Log_RT90-Model	Log_RT90-SYNTHETIC	0.21	0.49	0.18	0.47	0.42	0.68	0.85	0.60
15-081-21980	Log_RT90	Log_RT90-Model	Log_RT90-SYNTHETIC	0.22	0.59	0.11	0.71	0.33	0.84	0.95	0.66
15-081-21981	Log_RT90	Log_RT90-Model	Log_RT90-SYNTHETIC	0.23	0.54	0.08	0.54	0.28	0.73	0.96	0.70
15-081-22005	Log_RT90	Log_RT90-Model	Log_RT90-SYNTHETIC	0.16	0.55	0.05	0.57	0.22	0.76	0.97	0.68
15-081-22006	Log_RT90	Log_RT90-Model	Log_RT90-SYNTHETIC	0.17	0.47	0.04	0.43	0.20	0.65	0.98	0.76
15-081-22010	Log_RT90	Log_RT90-Model	Log_RT90-SYNTHETIC	0.17	0.56	0.04	0.58	0.21	0.76	0.98	0.68
15-081-22013	Log_RT90	Log_RT90-Model	Log_RT90-SYNTHETIC	0.12	0.50	0.03	0.49	0.17	0.70	0.99	0.75
15-081-22016	Log_RT90	Log_RT90-Model	Log_RT90-SYNTHETIC	0.18	0.52	0.05	0.54	0.22	0.73	0.98	0.72
15-081-22018	Log_RT90	Log_RT90-Model	Log_RT90-SYNTHETIC	0.15	0.57	0.15	0.81	0.39	0.90	0.92	0.54
15-081-22020	Log_RT90	Log_RT90-Model	Log_RT90-SYNTHETIC	0.18	0.66	0.07	0.88	0.26	0.94	0.97	0.60
15-081-22021	Log_RT90	Log_RT90-Model	Log_RT90-SYNTHETIC	0.14	0.56	0.03	0.59	0.18	0.77	0.98	0.71
15-081-22031	Log_RT90	Log_RT90-Model	Log_RT90-SYNTHETIC	0.19	0.50	0.05	0.45	0.22	0.67	0.97	0.72
15-081-22033	Log_RT90	Log_RT90-Model	Log_RT90-SYNTHETIC	0.18	0.51	0.05	0.48	0.23	0.69	0.97	0.70
15-081-22048	Log_RT90	Log_RT90-Model	Log_RT90-SYNTHETIC	0.19	0.53	0.06	0.52	0.24	0.72	0.97	0.72
15-081-22055	Log_RT90	Log_RT90-Model	Log_RT90-SYNTHETIC	0.21	0.61	0.24	0.86	0.49	0.93	0.87	0.55
15-081-22056	Log_RT90	Log_RT90-Model	Log_RT90-SYNTHETIC	0.18	0.54	0.07	0.57	0.26	0.76	0.95	0.61

Figure 4.7: The overall results of the RT90 Model.

5. Conclusions and Future Work

5.1 Summary of Findings

In this study, we have successfully created a software that allows the user to import well logs, filter and process the well logs, build a artificial neural network, train the network, and then deploy the model to view the results.

5.2 Limitations of Synthetic Well Logging

1. **Data Quality and Availability:** The accuracy of synthetic well logs heavily relies on the quality and quantity of the training data. Incomplete, sparse, or noisy data can lead to inaccurate predictions.

2. **Model Complexity:** Complex models might overfit to the training data, making them less generalizable to new, unseen data. In this study, the location and proximity of the wells could disguise an over fitting model.

3. **Computational Requirements:** High computational resources might be required, especially for complex models and large datasets. While using multiple layers in a neural network can be seen has being efficient, the lower batch sizes and potential use of regularization could cause the computation resources to dramatically increase.

4. **Generalization:** Models trained on data from specific geological formations might not perform well on data from different formations.

5.3 Recommendations for Further Research

1. **Data Augmentation:** Explore methods to augment sparse well log data to improve model training.

2. **Model Explainability:** Research into explainable AI can help in understanding the decision-making process of the models.

3. **Hybrid Models:** Combining machine learning with traditional physics-based models might improve accuracy and generalizability.

4. **Uncertainty Analysis:** Develop methods to quantify uncertainty in predictions, possibly using Bayesian approaches or ensemble methods.

5. **Cross-Formation Training:** Experiment with training models across various geological formations to improve robustness and generalizability.

6. **Real-time Application:** Investigate the feasibility and efficiency of applying these models in real-time well logging scenarios.

7. **Comparative Studies:** Conduct comparative studies with traditional well logging methods to validate the efficiency and accuracy of synthetic logs.

8. **Feature Engineering:** Explore advanced feature engineering techniques to better capture the geological characteristics relevant to well logging.

Works Cited

Alnuaimi, Marwan Mohammed. “Using Artificial Intelligence And Machine Learning To Develop Synthetic Well Logs”. 2018. West Virginia U, Graduate Theses, Dissertations, and Problem Reports. researchrepository.wvu.edu/etd/3747.

Andersen, Mark A. The Defining Series: Basic Well Log Interpretation. Available at www.slb.com. Publication: Oilfield Review.

———. The Defining Series: Introduction to Wireline Logging. Available at www.slb.com. Publication: Oilfield Review.

Gamma ray logs. Available at petrowiki.spe.org.

Géron, Aurélien. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow.* O’Reilly Media, 2019.

Keras. Keras: The Python Deep Learning API. 2023, Accessed: 2023-12-08. keras.io.

Lashari, Shan e Zehra. “Applications of Artificial Intelligence (AI) in Petroleum Engineering Problems”. 2018. West Virginia U, Graduate Theses, Dissertations, and Problem Reports. researchrepository.wvu.edu/etd/6041.

Neutron porosity logs. Available at petrowiki.spe.org.

Openhole caliper logs. Available at petrowiki.spe.org.

Ore, Tobi Micheal. “A Machine Learning and Data-Driven Prediction and Inversion of Reservoir Brittleness from Geophysical Logs and Seismic Signals: A Case Study in Southwest Pennsylvania, Central Appalachian Basin”. 2020. West Virginia U, Graduate Theses, Dissertations, and Problem Reports. researchrepository.wvu.edu/etd/7714.

Rolon, Luisa F. Developing Intelligent Synthetic Logs: Application to Upper Devonian Units in PA. 2004, Morgantown, West Virginia.

Sonic logging. Available at en.wikipedia.org/wiki/Sonic$_{l}$*ogging*.

“The Photoelectric Index”. *Geological Log Interpretation*, Available at pubs.geoscienceworld.org.

Zhang, Dongxiao, et al. “Synthetic well logs generation via Recurrent Neural Networks”. *Petroleum Exploration and Development*, vol. 45, no. 4, 2018, Available online at www.sciencedirect.com, pp. 629–39.

6. Appendices

6.1 Sequentially Generated Logs

6.1.1 Actual vs Synthetic Well Logs for Training and Validation Wells

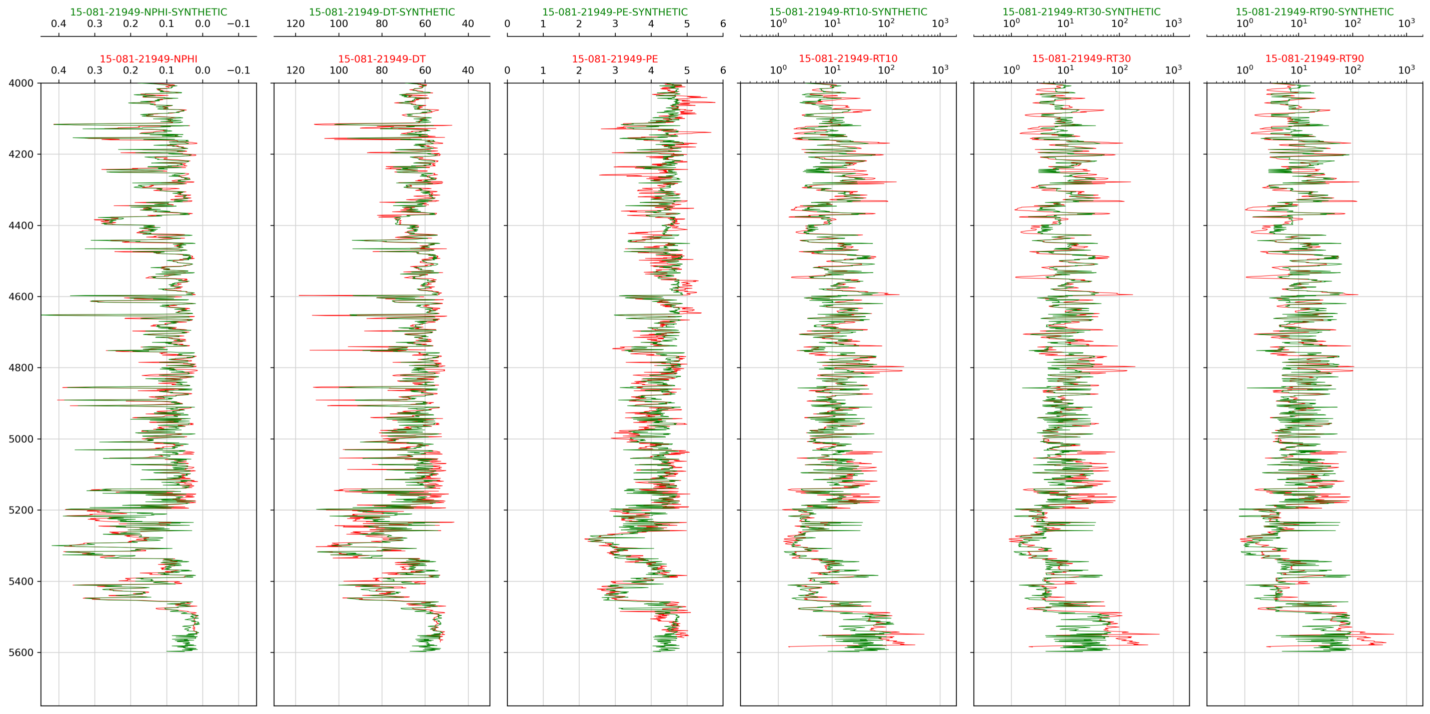


Figure 6.1: Comparison of model performance for a well used in training.

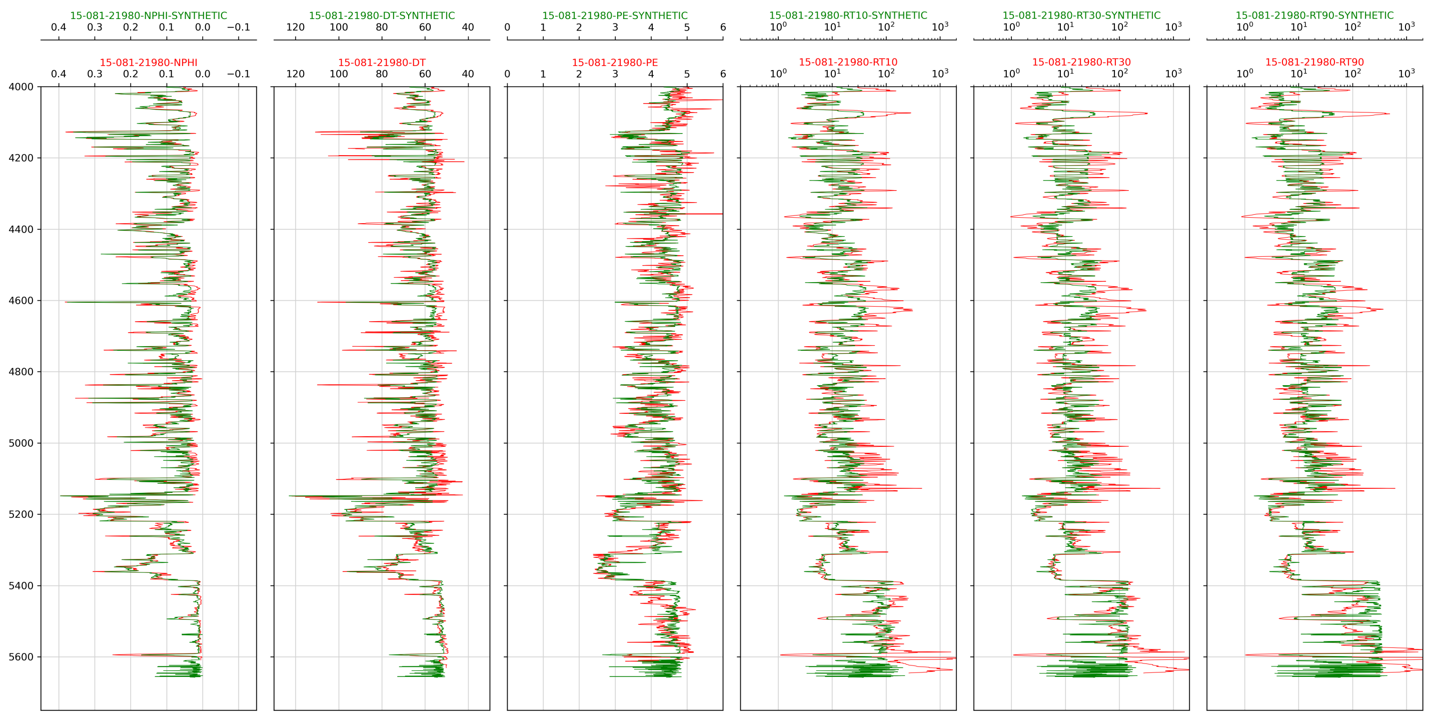


Figure 6.2: Comparison of model performance for a well used in training.

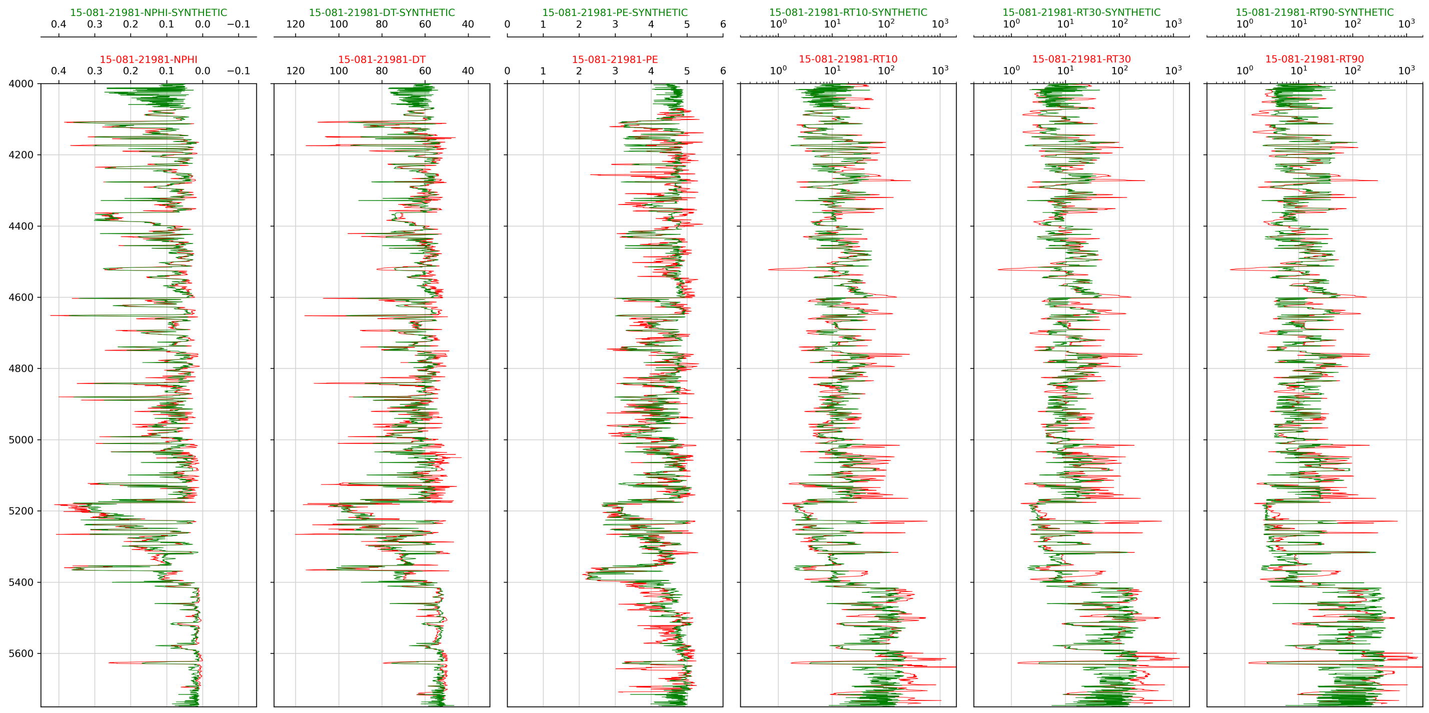


Figure 6.3: Comparison of model performance for a well used in training.

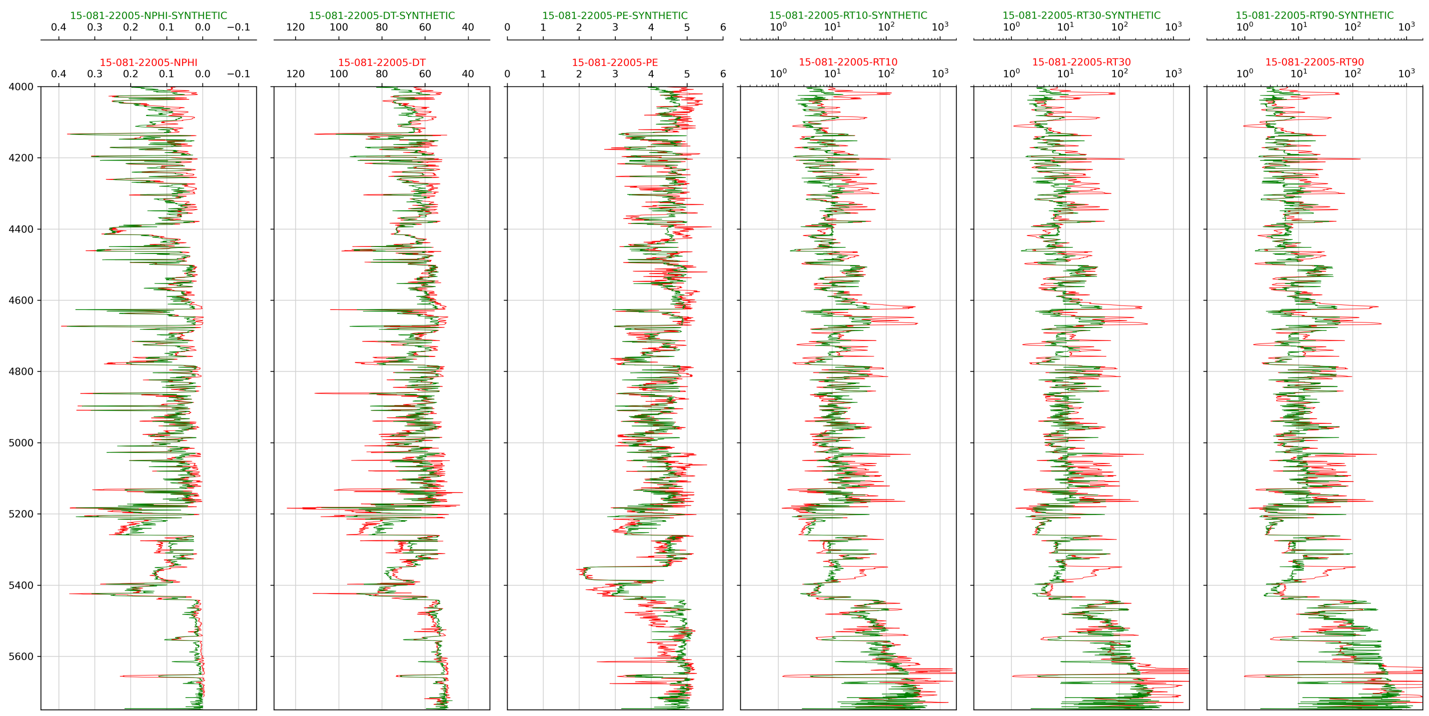


Figure 6.4: Comparison of model performance for a well used in validation.

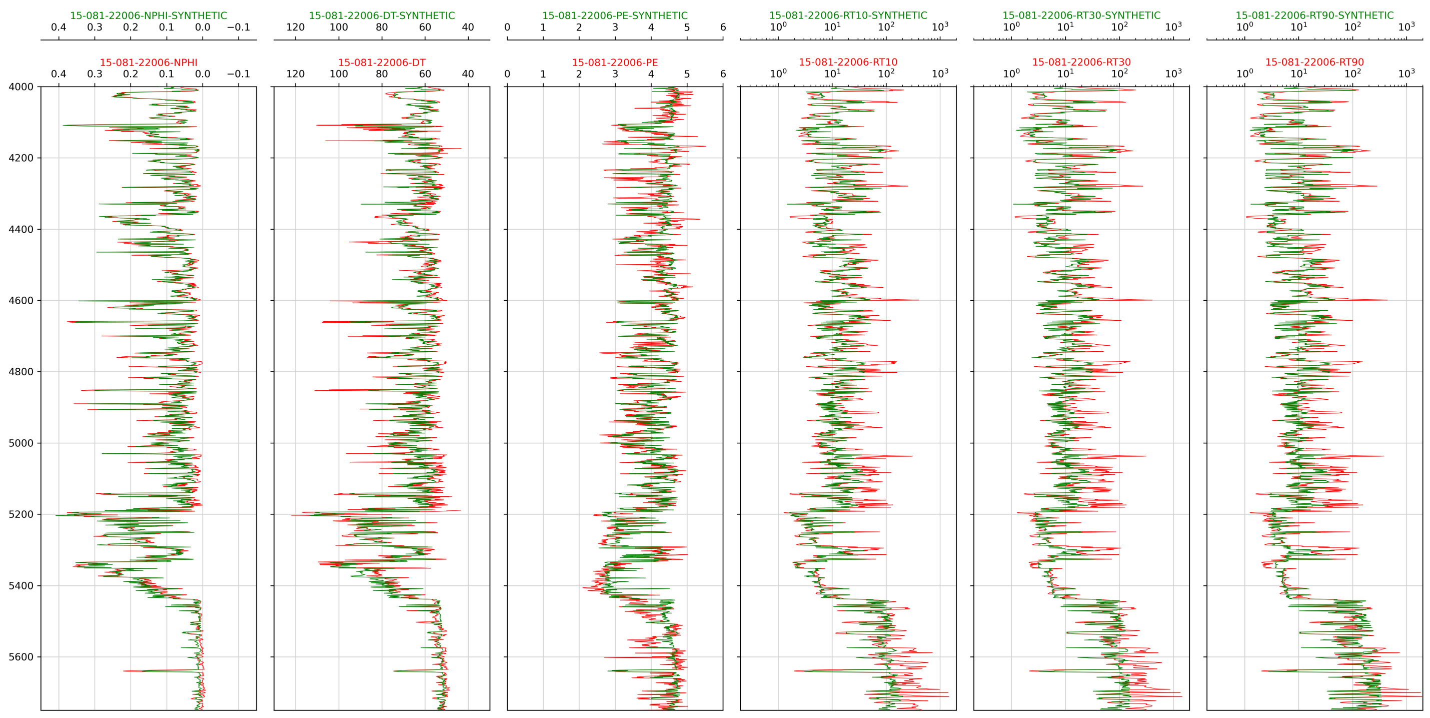


Figure 6.5: Comparison of model performance for a well used in validation.

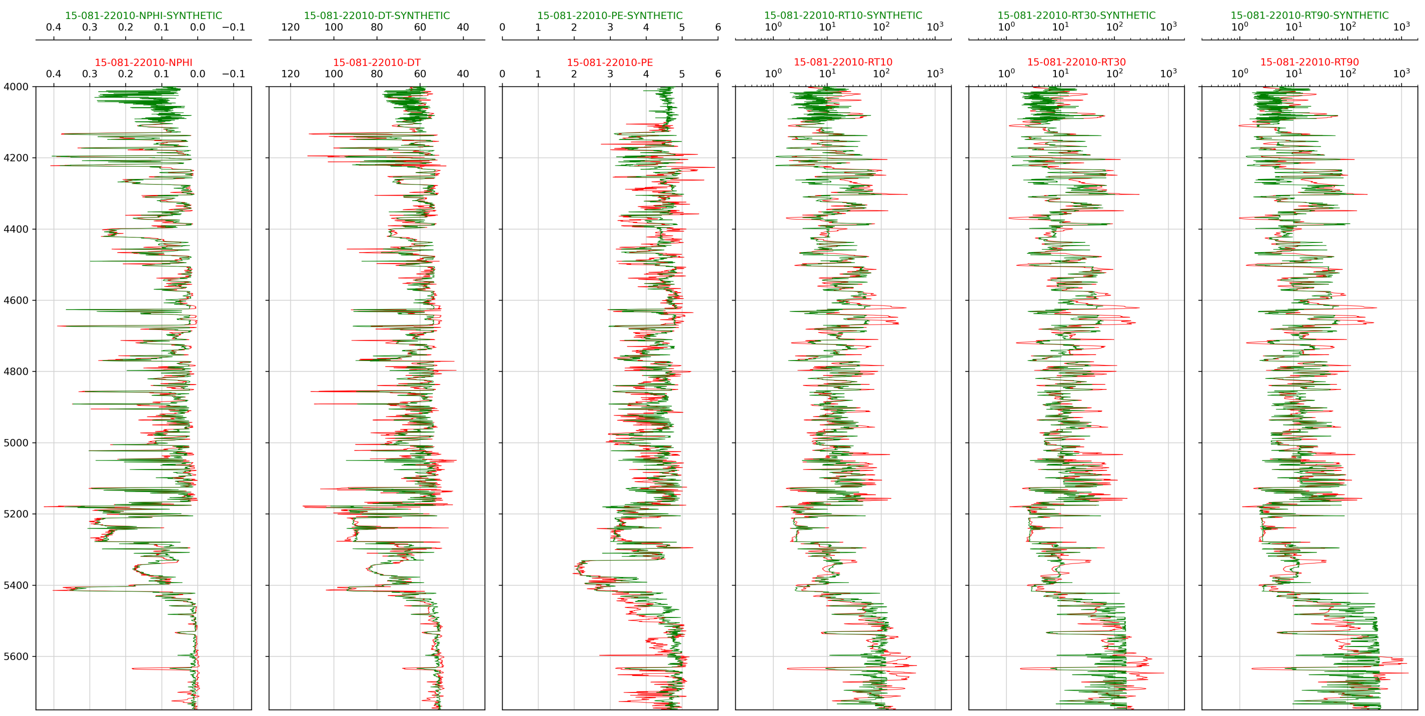


Figure 6.6: Comparison of model performance for a well used in training.

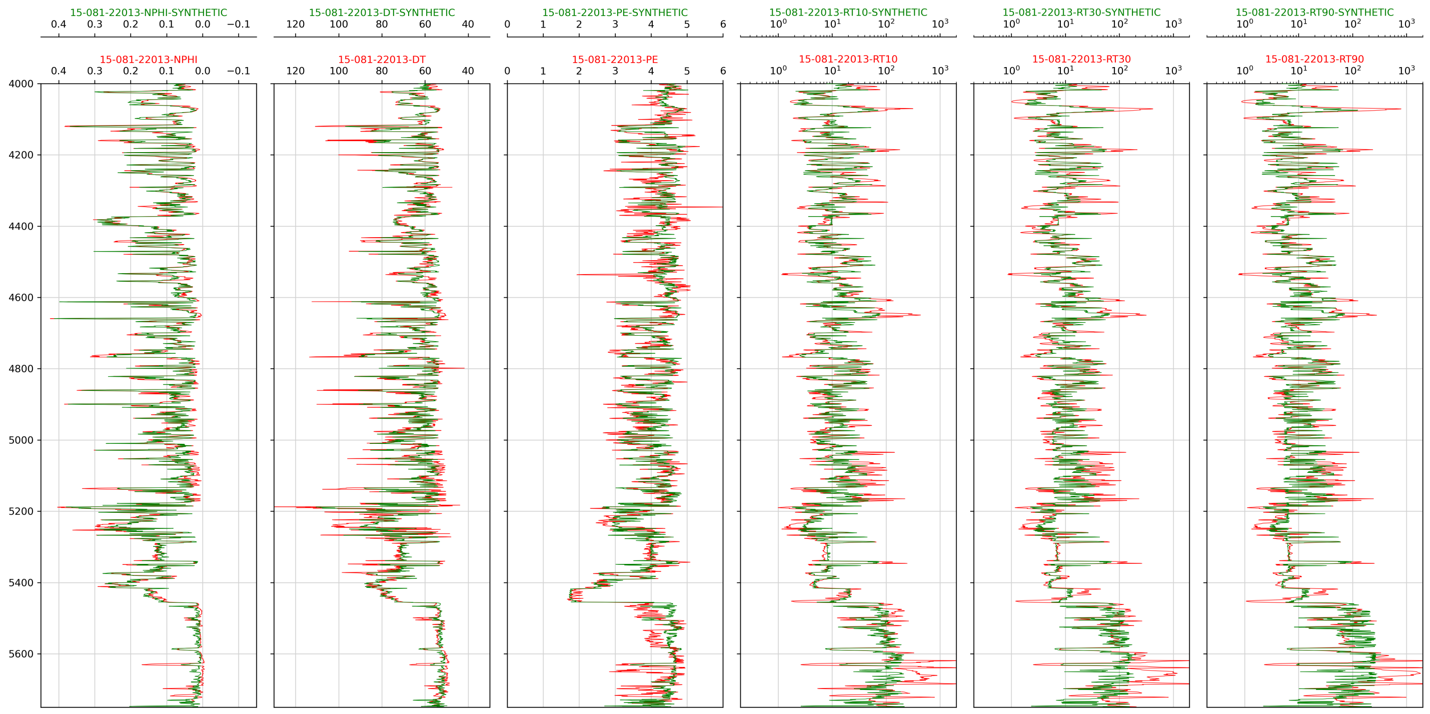


Figure 6.7: Comparison of model performance for a well used in training.

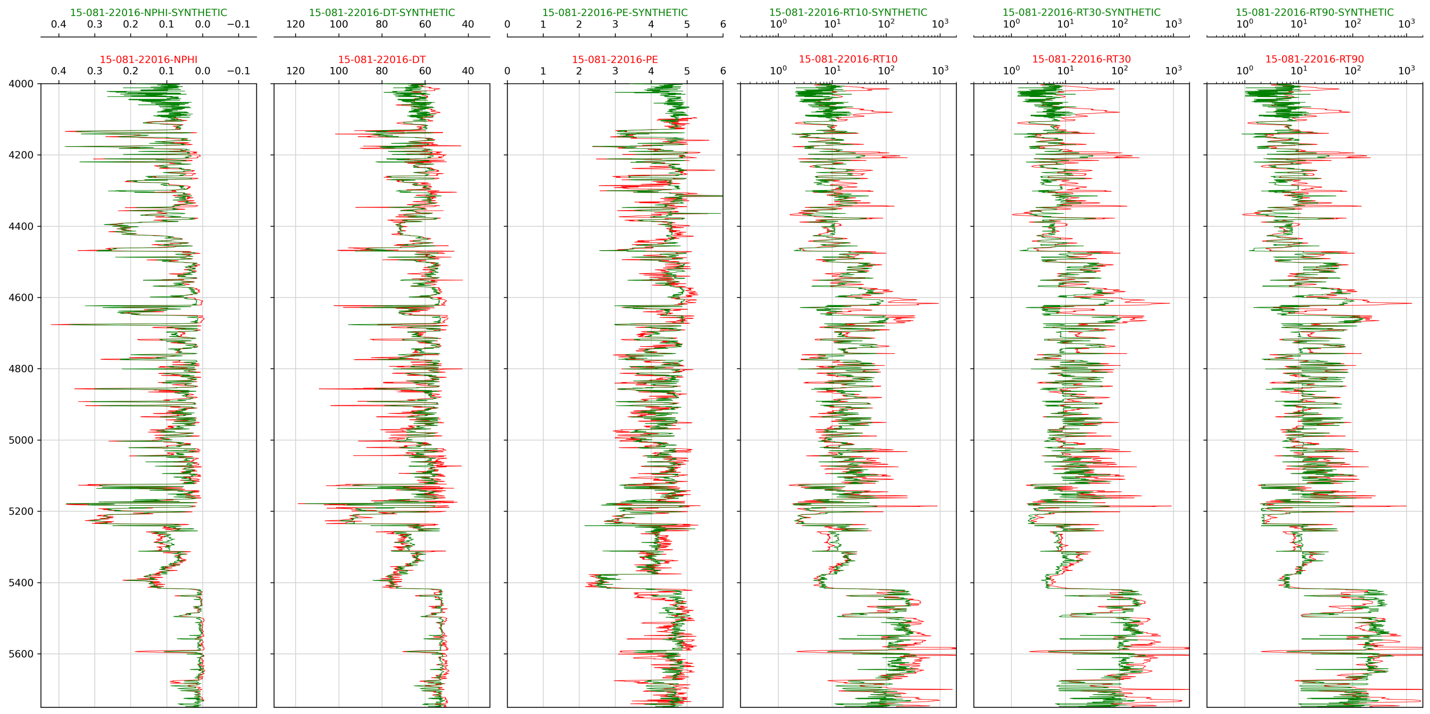


Figure 6.8: Comparison of model performance for a well used in training.

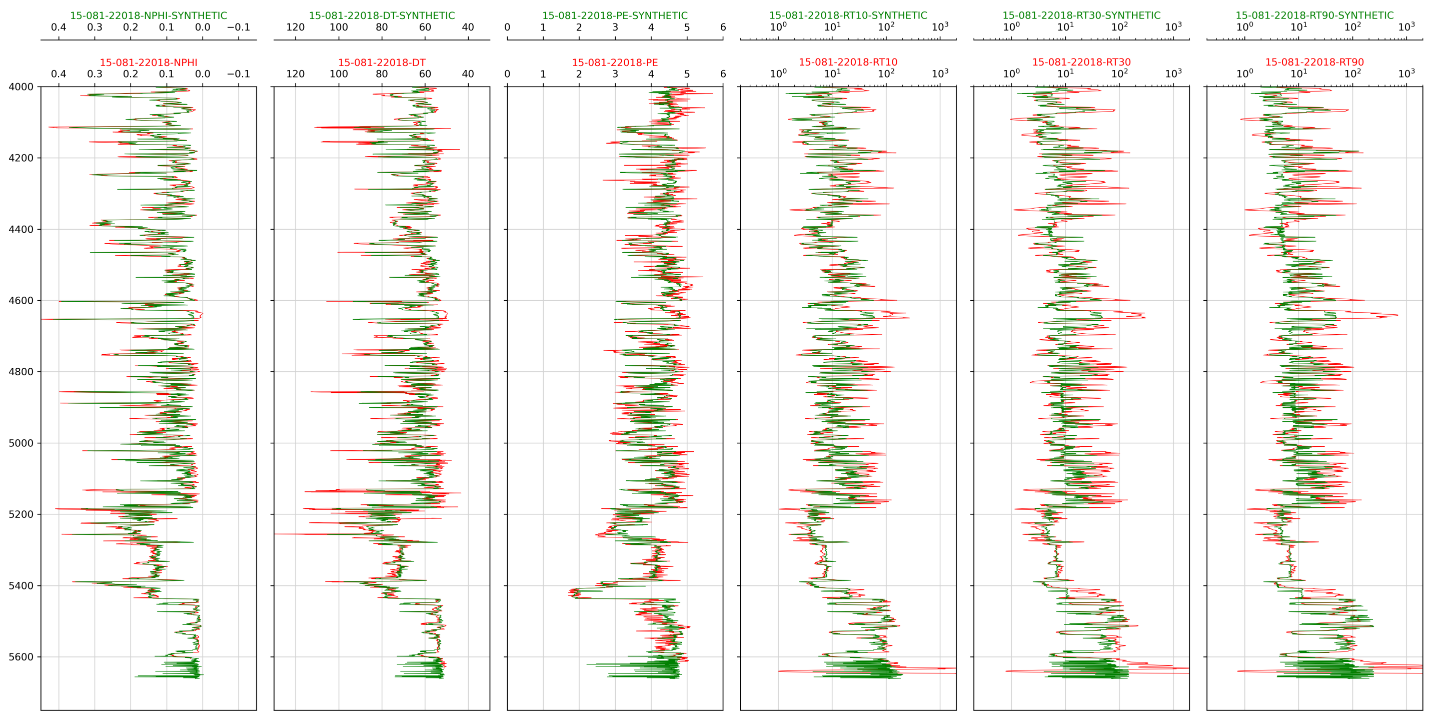


Figure 6.9: Comparison of model performance for a well used in training.

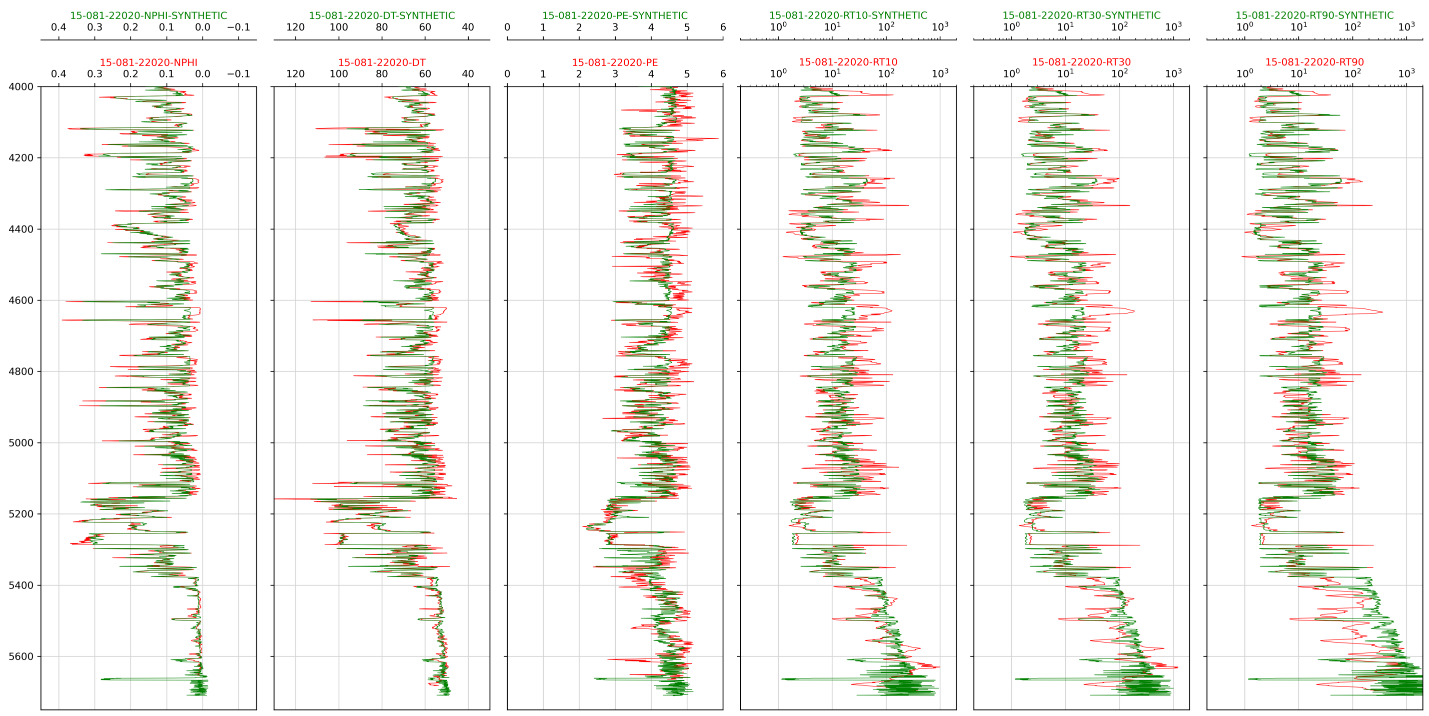


Figure 6.10: Comparison of model performance for a well used in training.

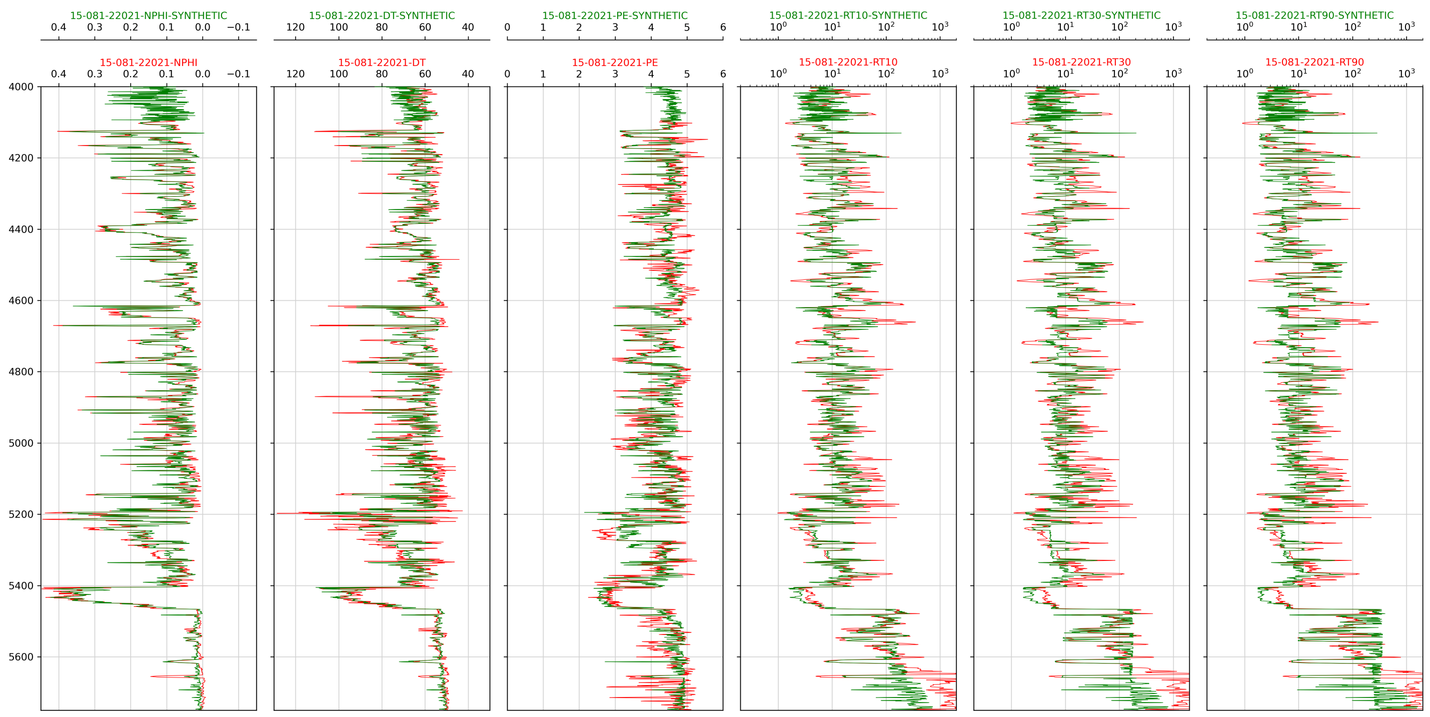


Figure 6.11: Comparison of model performance for a well used in training.

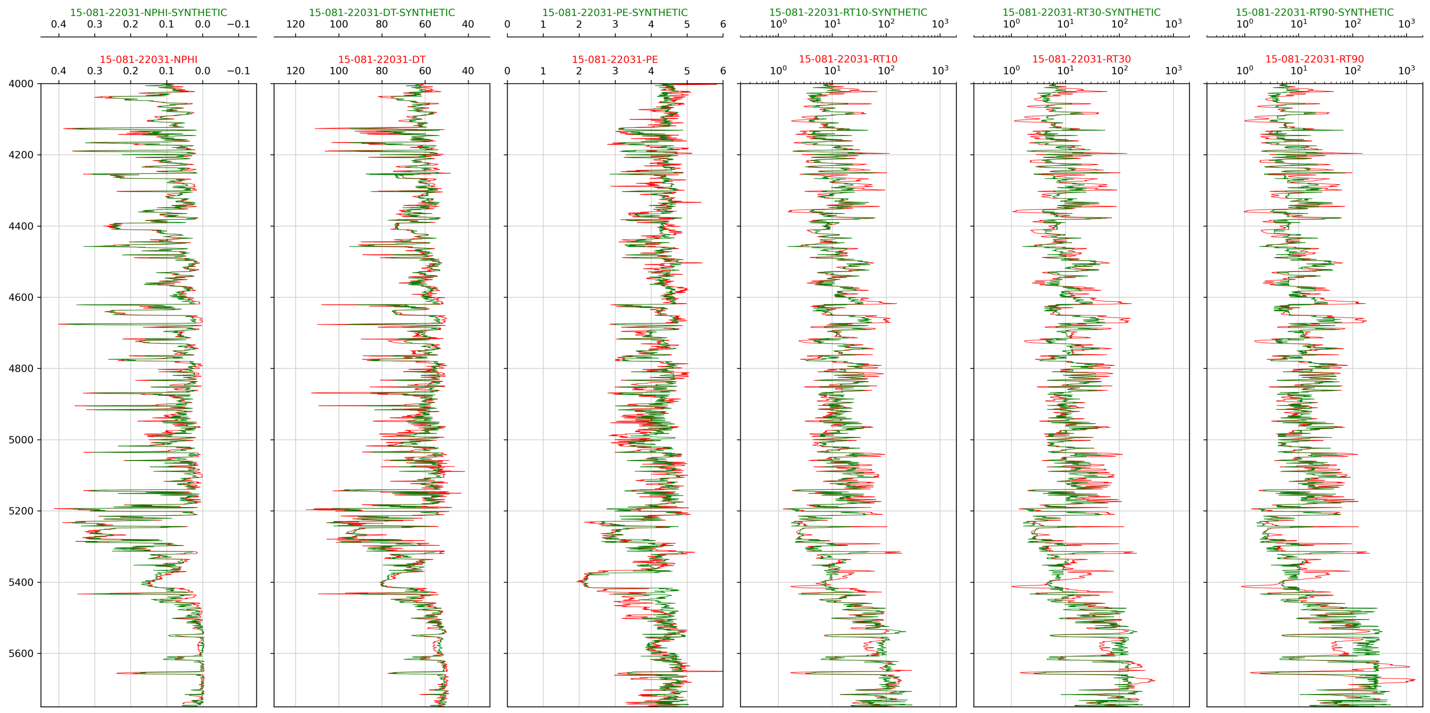


Figure 6.12: Comparison of model performance for a well used in training.

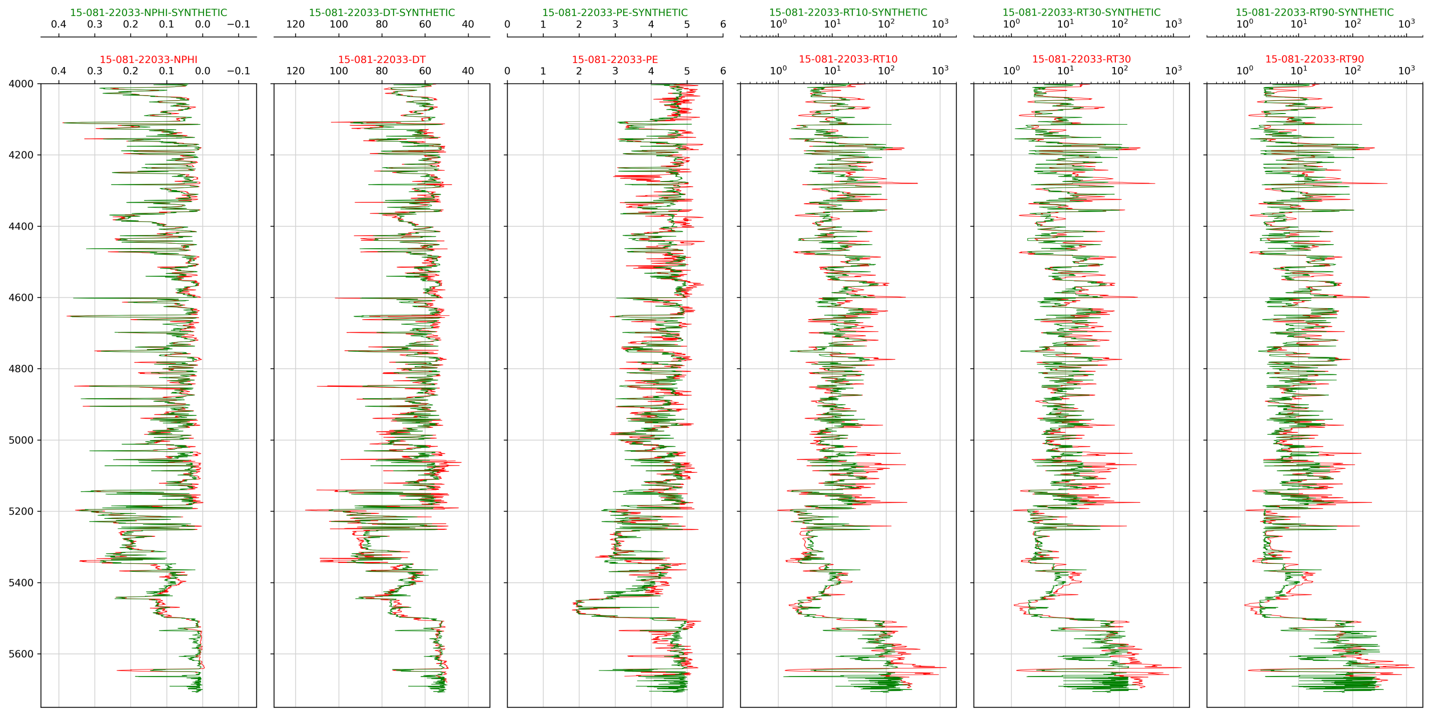


Figure 6.13: Comparison of model performance for a well used in training.

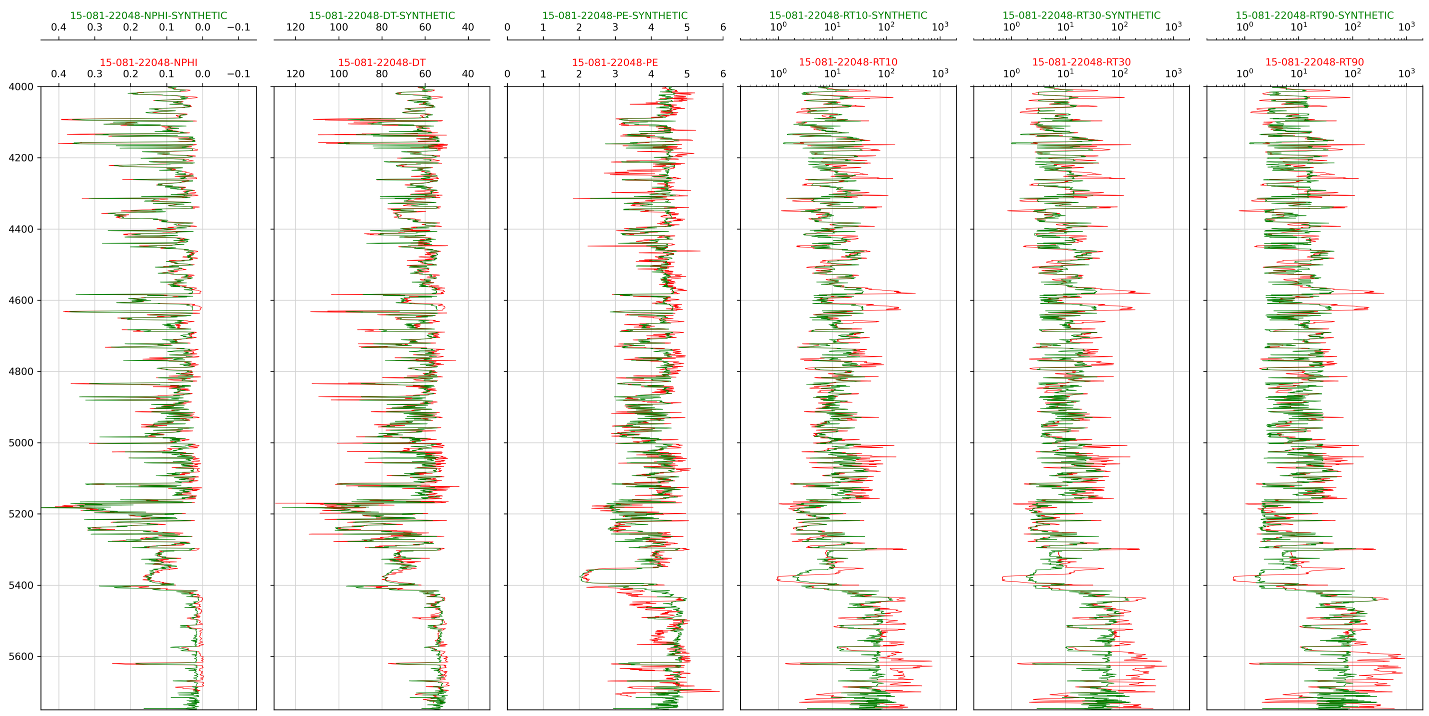


Figure 6.14: Comparison of model performance for a well used in training.

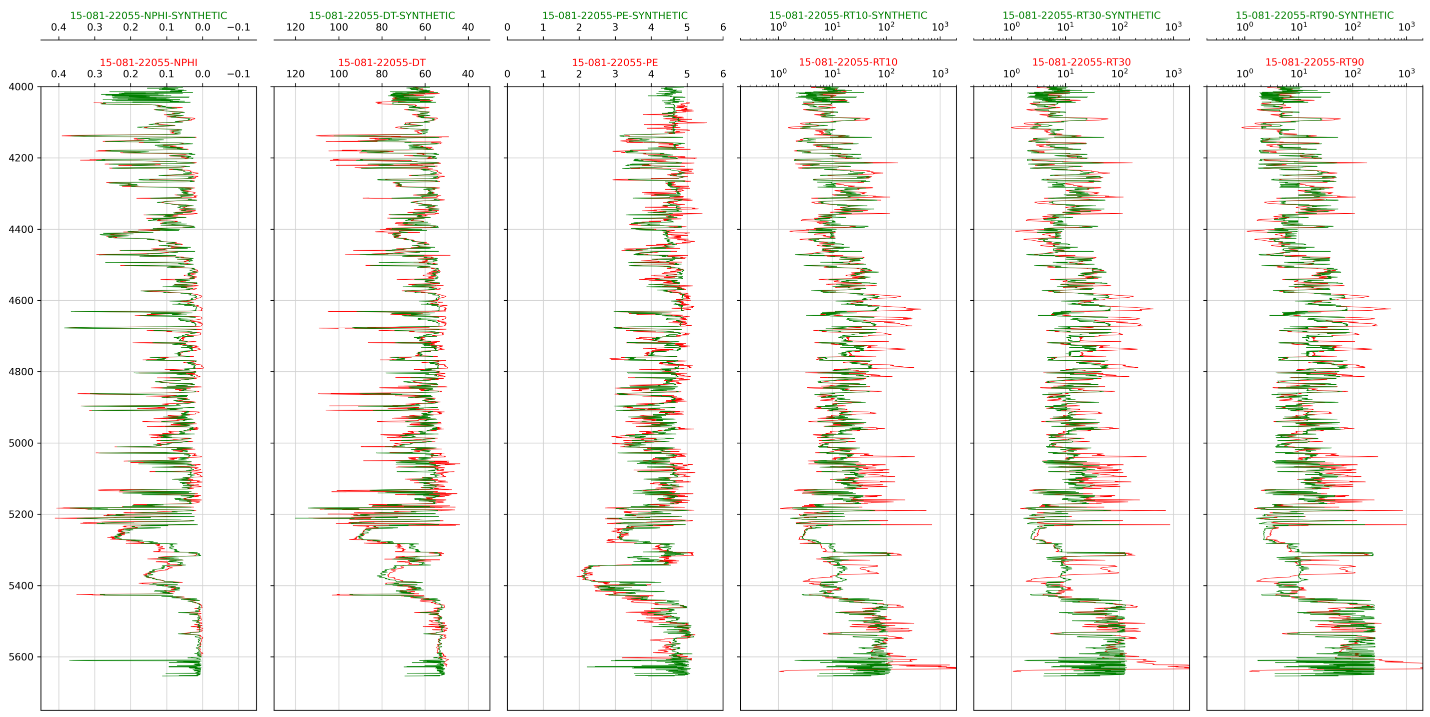


Figure 6.15: Comparison of model performance for a well used in training.

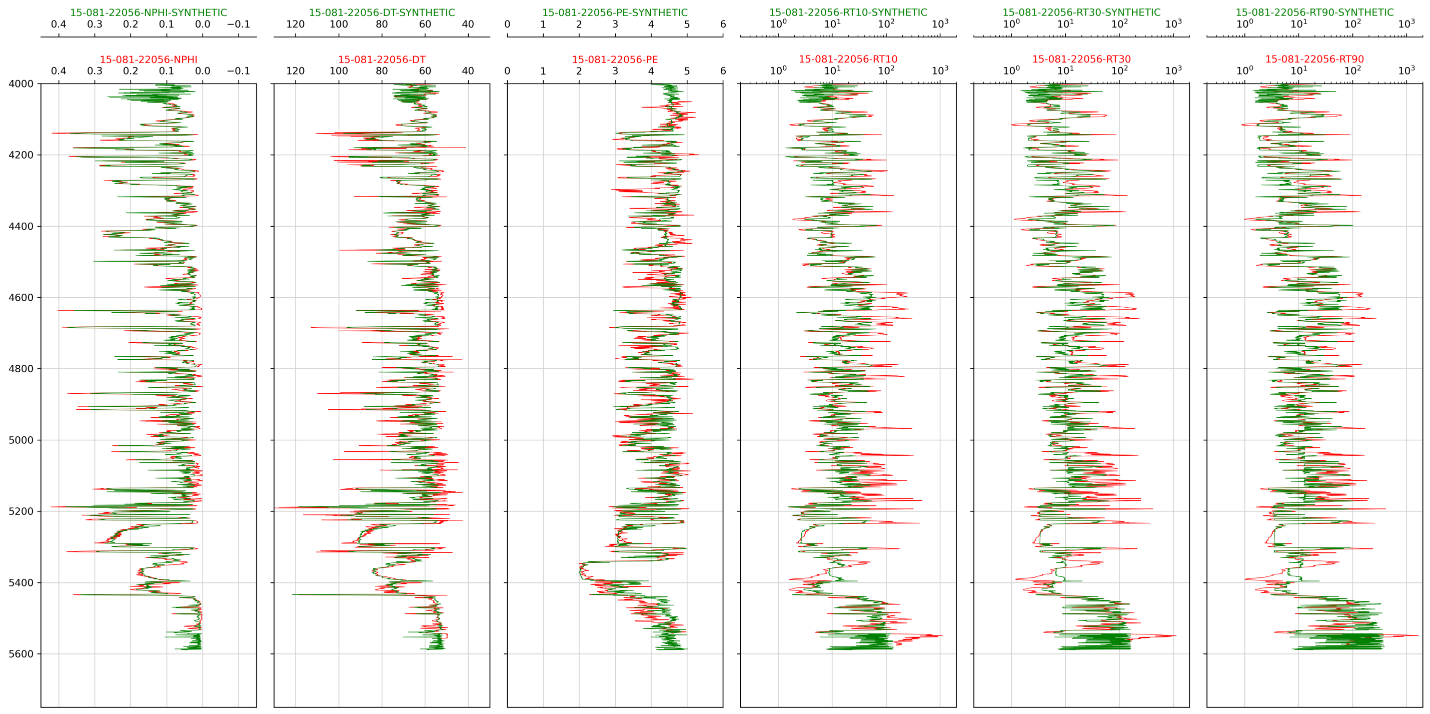


Figure 6.16: Comparison of model performance for a well used in training.

6.1.2 Actual vs Modeled Well Logs for Training and Validation Wells

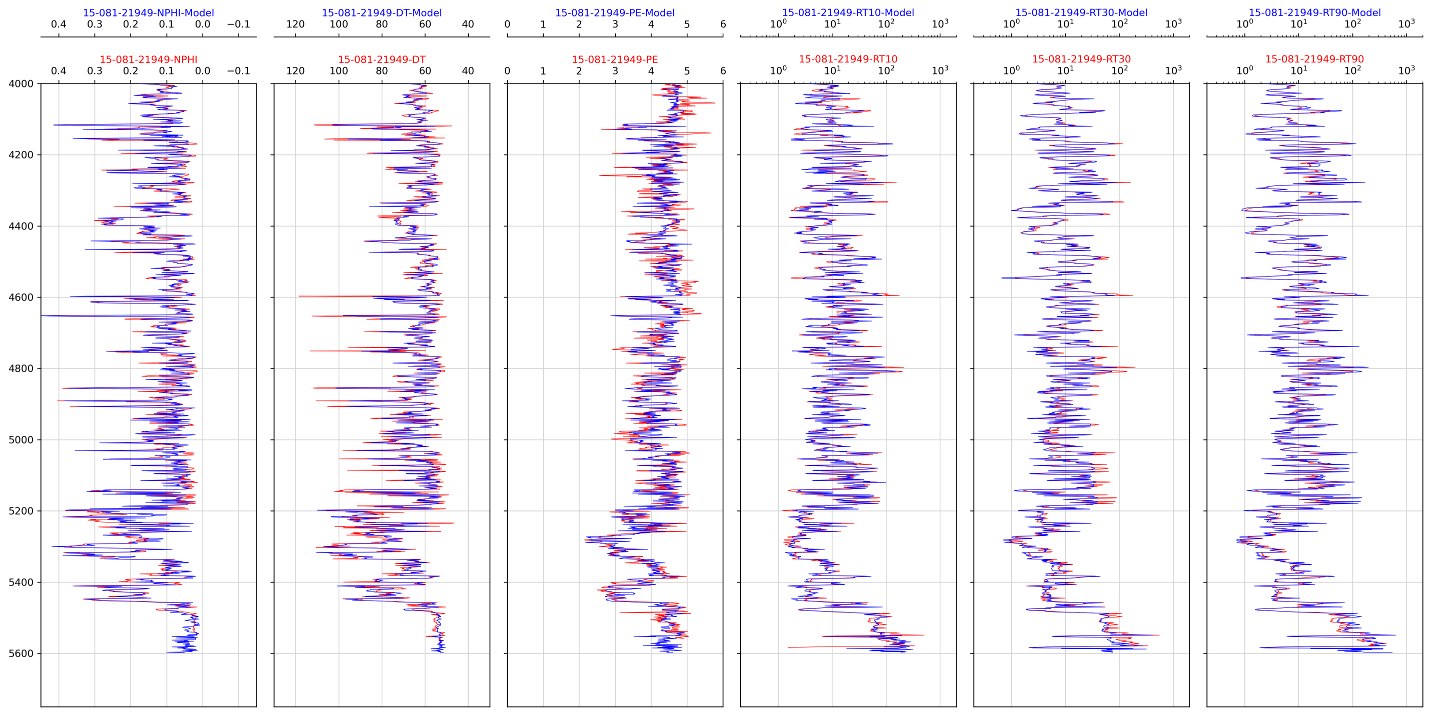


Figure 6.17: Comparison of model performance for a well used in training.

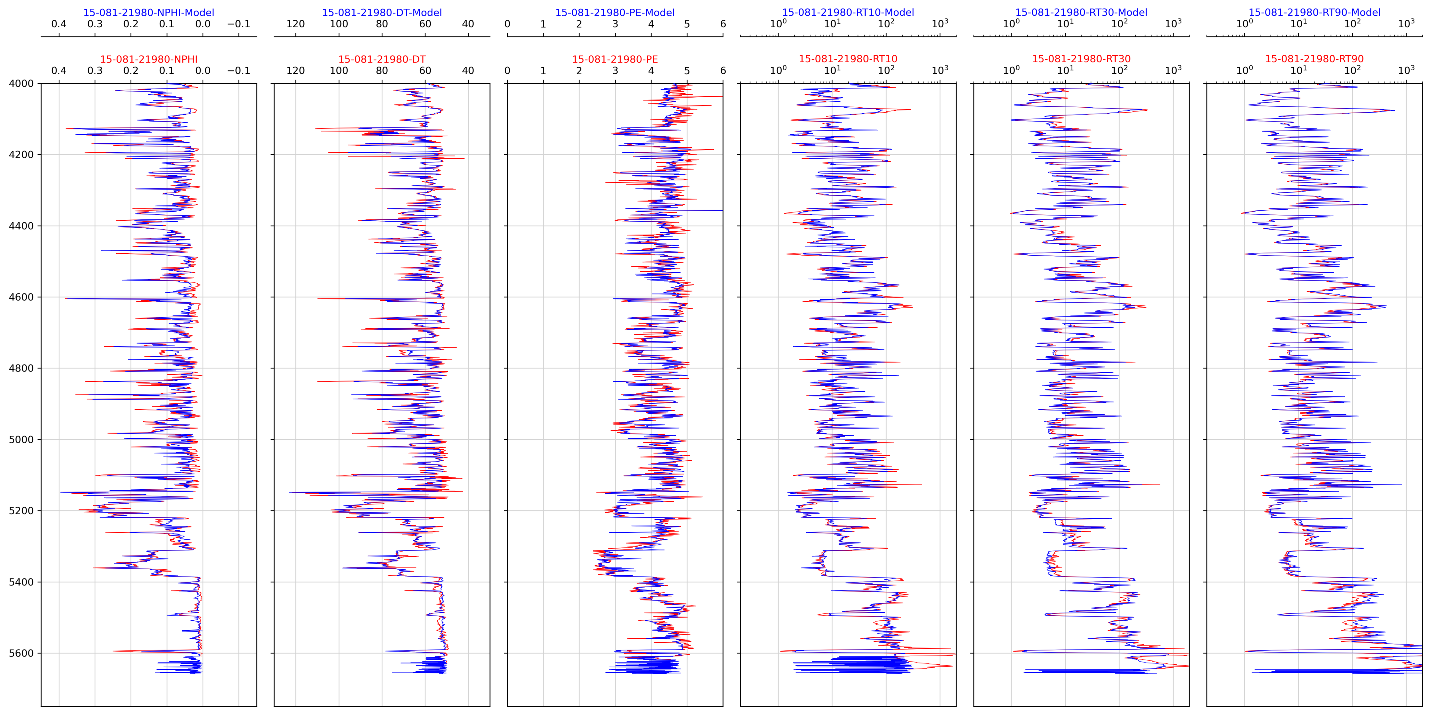


Figure 6.18: Comparison of model performance for a well used in training.

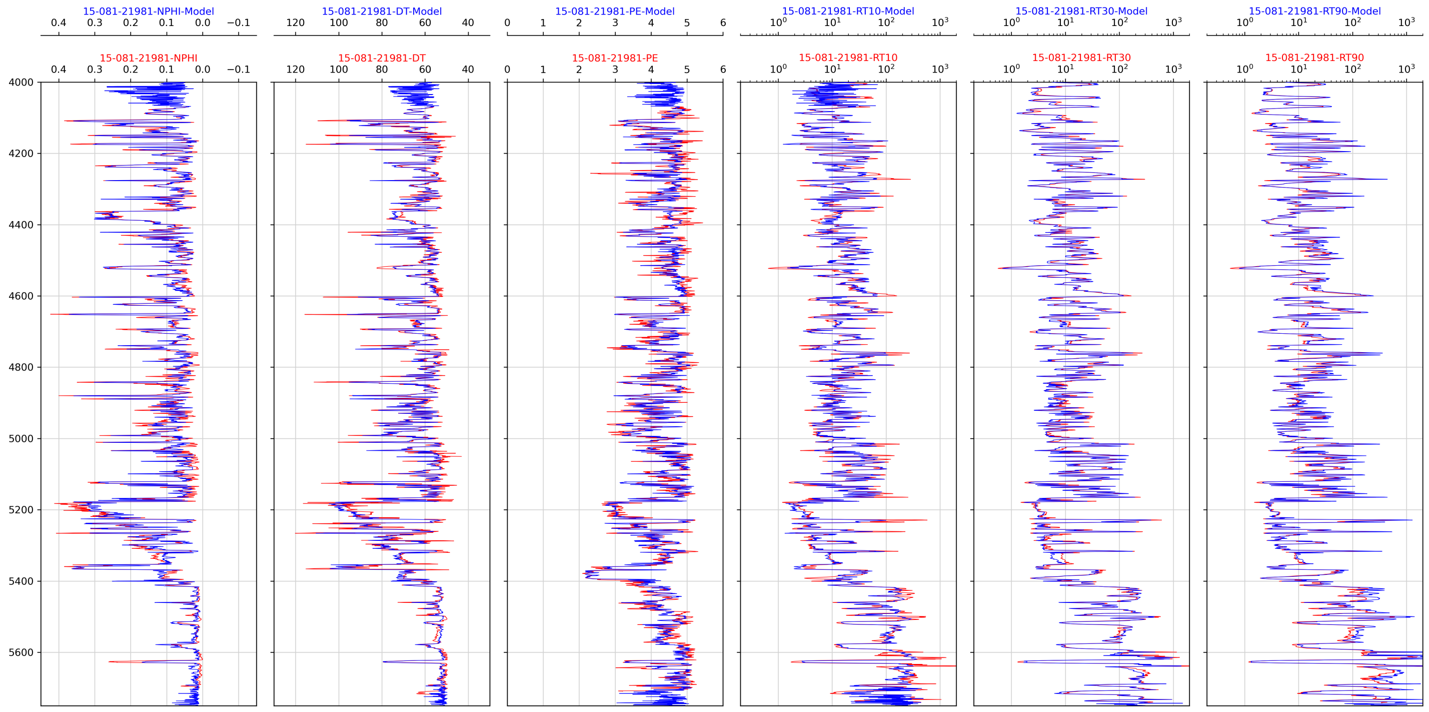


Figure 6.19: Comparison of model performance for a well used in training.

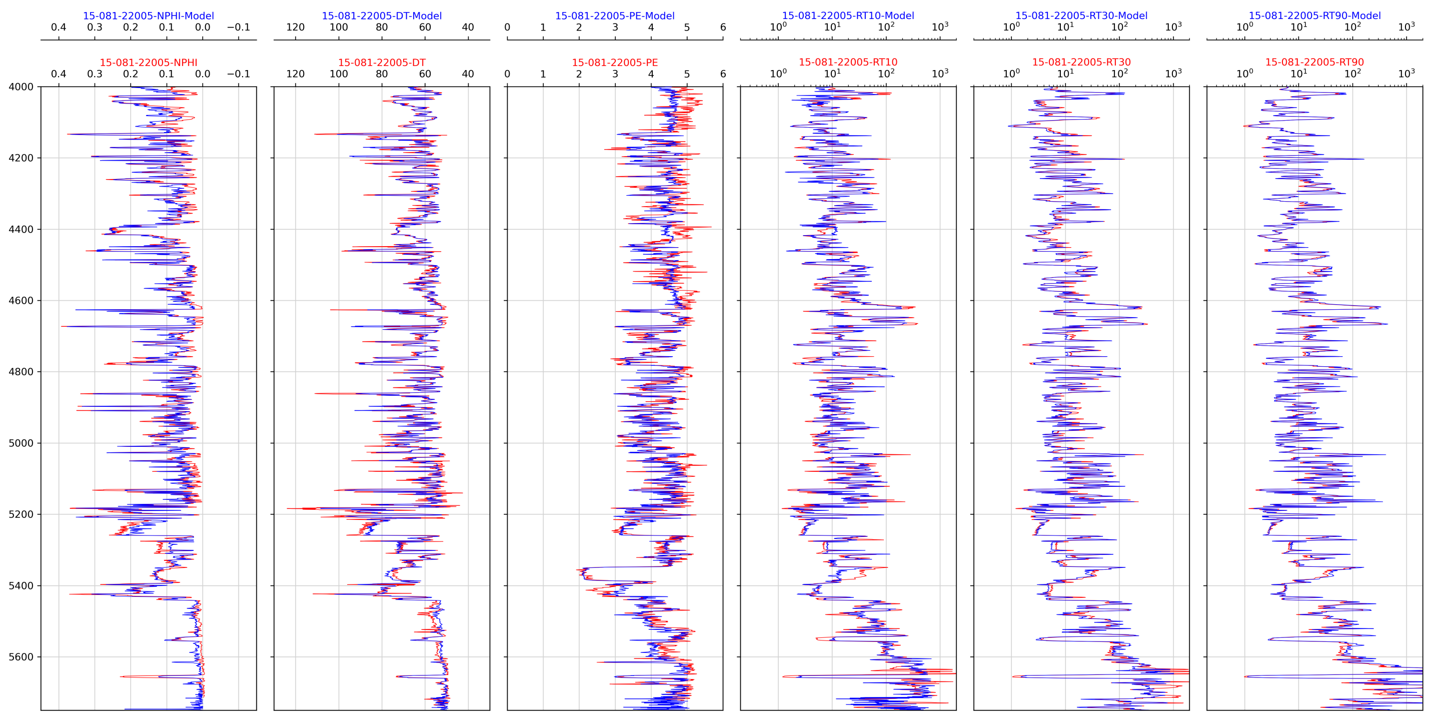


Figure 6.20: Comparison of model performance for a well used in validation.

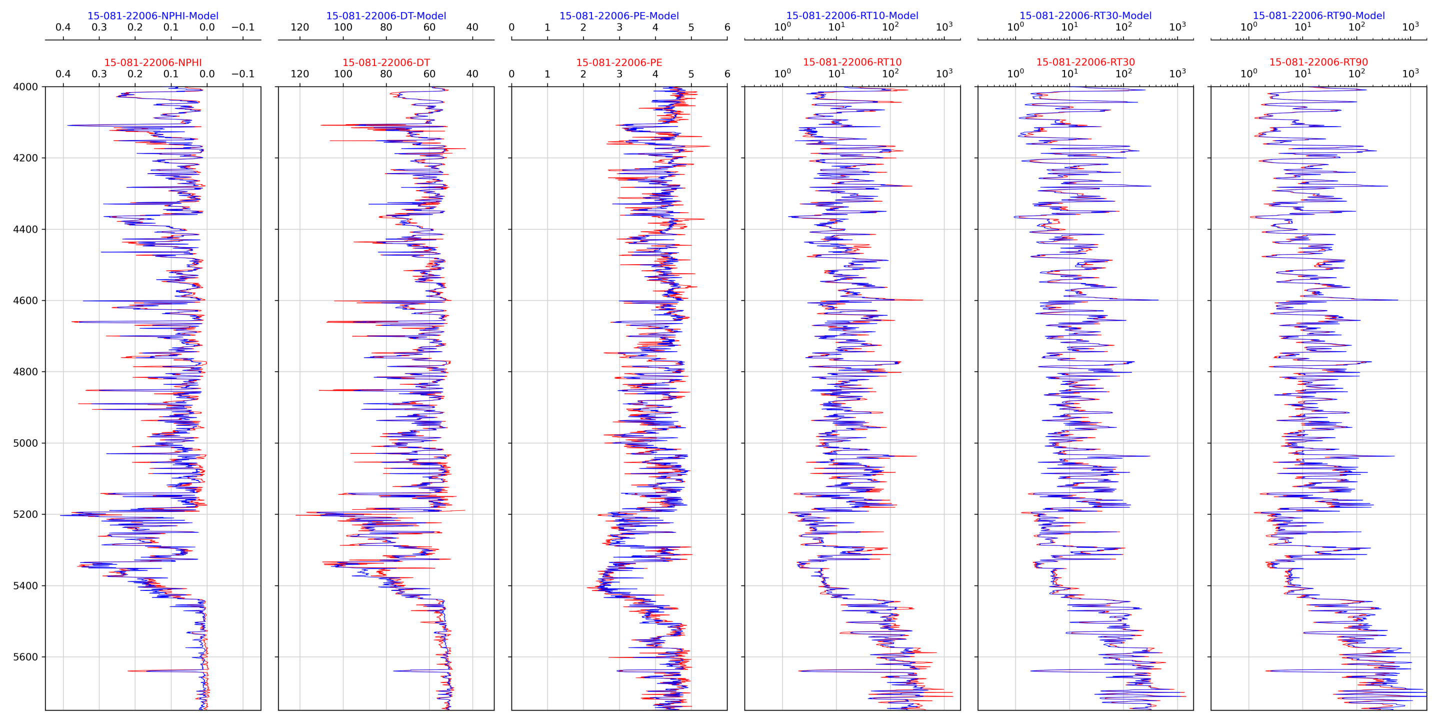


Figure 6.21: Comparison of model performance for a well used in validation.

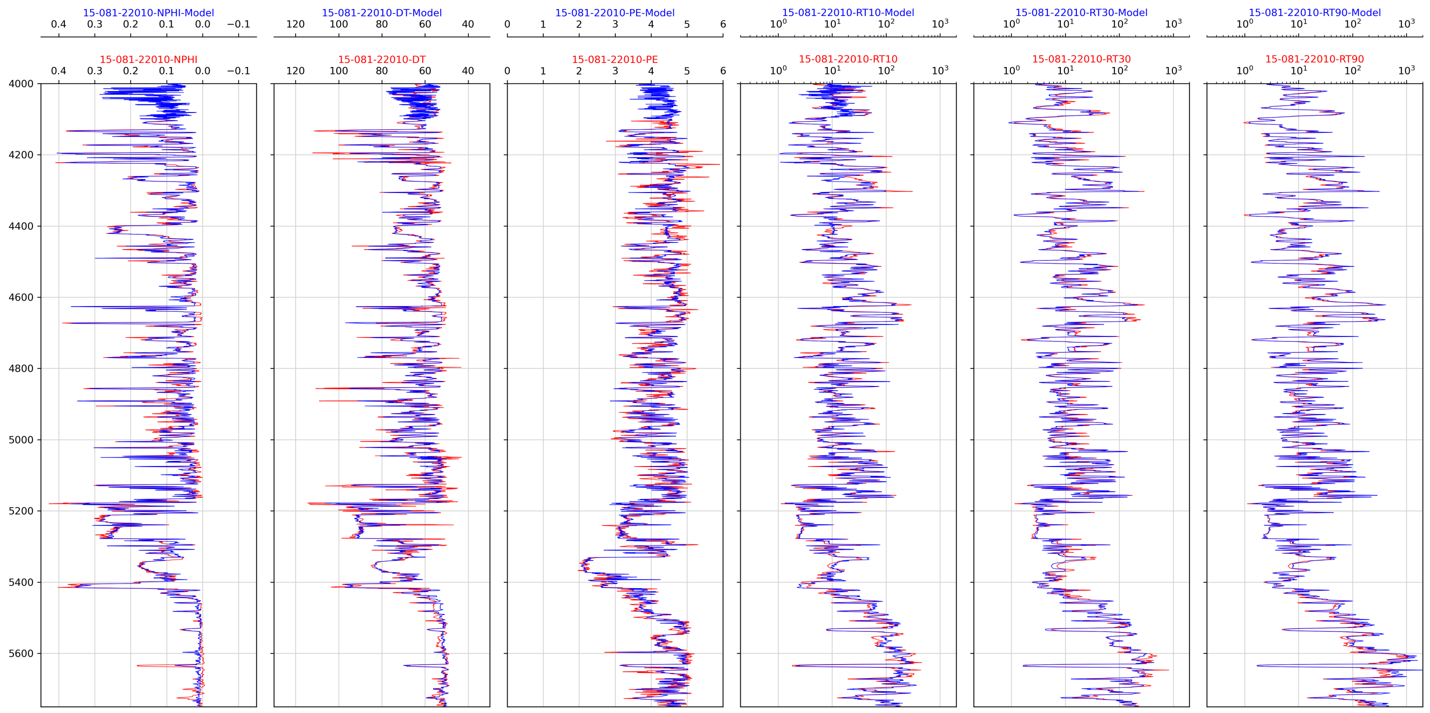


Figure 6.22: Comparison of model performance for a well used in training.

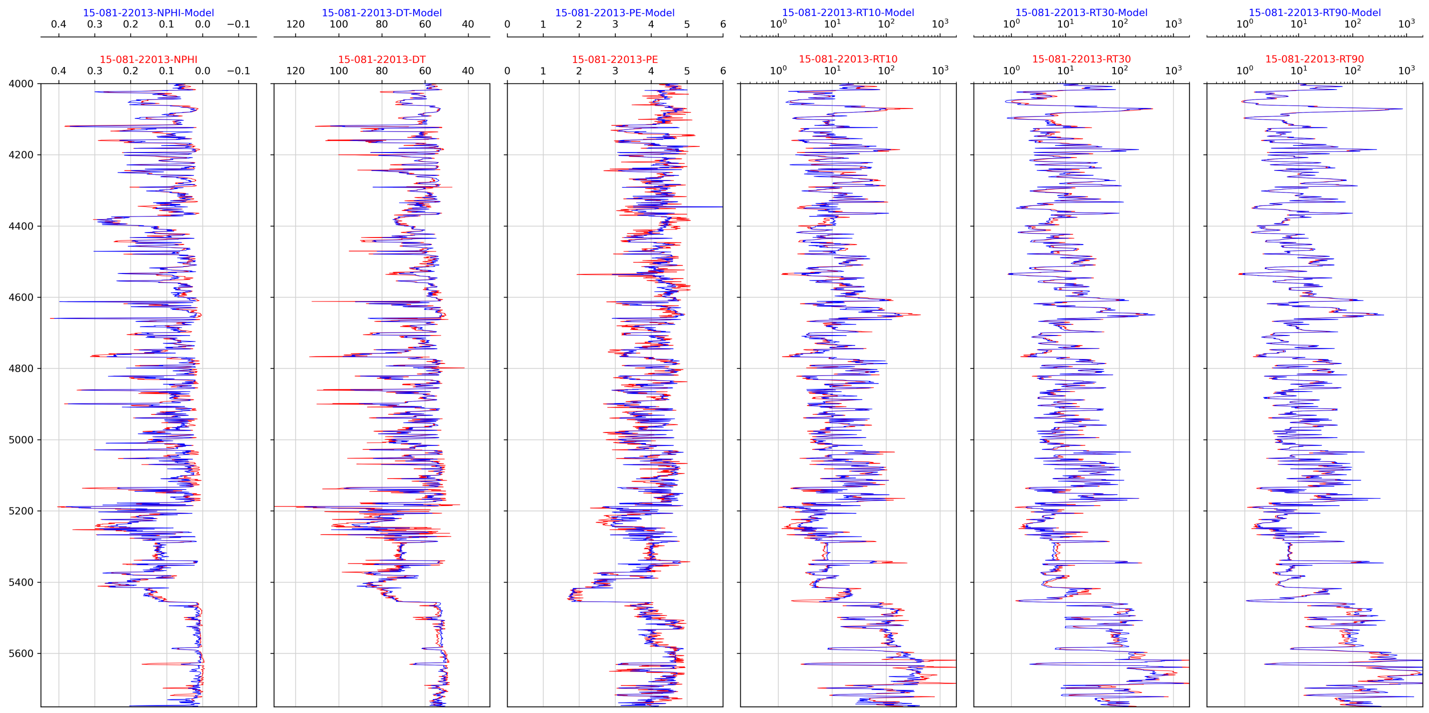


Figure 6.23: Comparison of model performance for a well used in training.

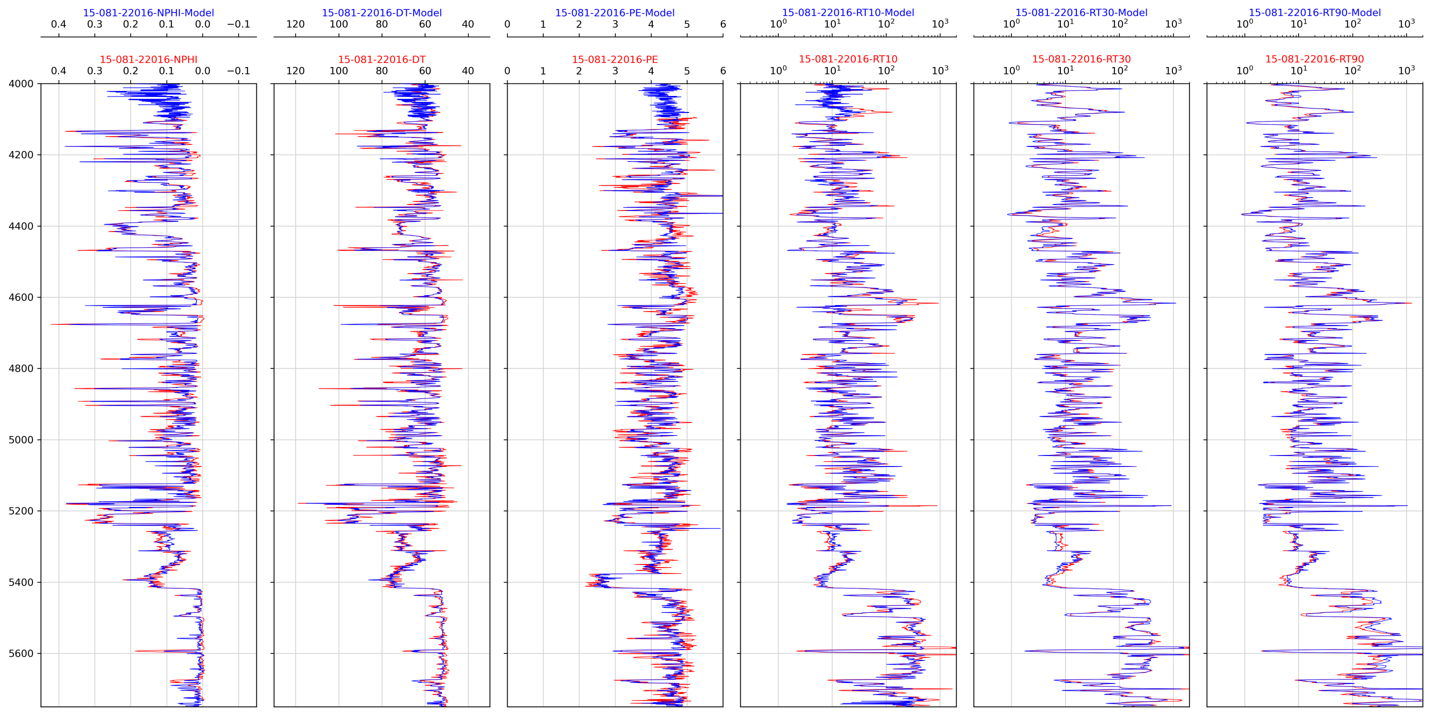


Figure 6.24: Comparison of model performance for a well used in training.

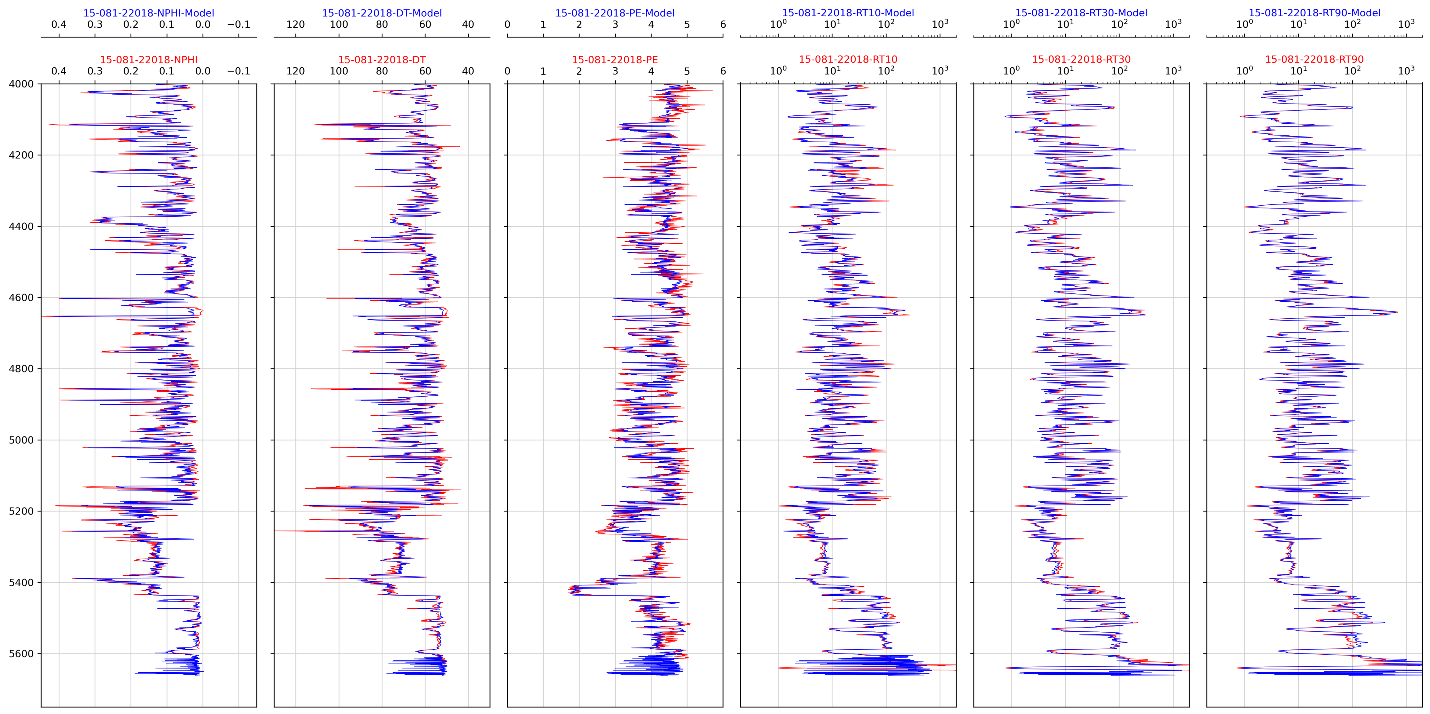


Figure 6.25: Comparison of model performance for a well used in training.

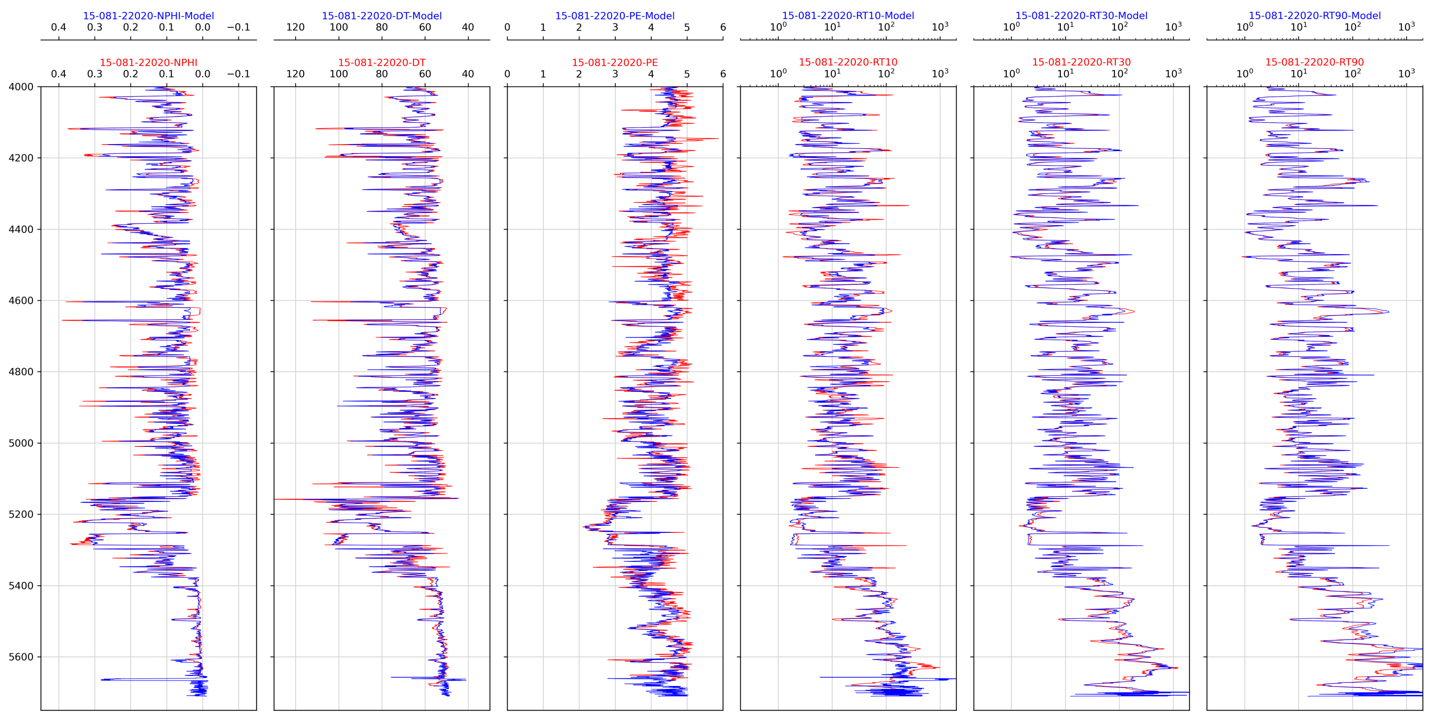


Figure 6.26: Comparison of model performance for a well used in training.

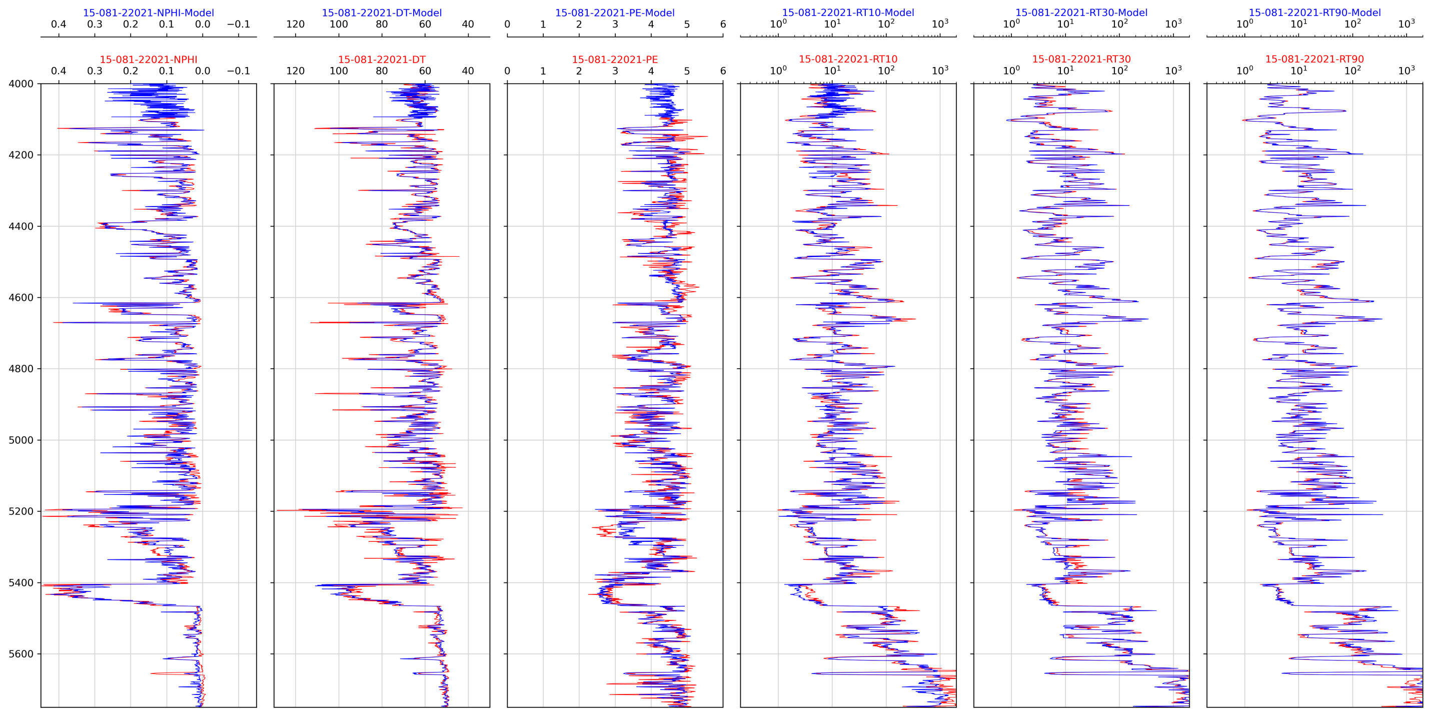


Figure 6.27: Comparison of model performance for a well used in training.

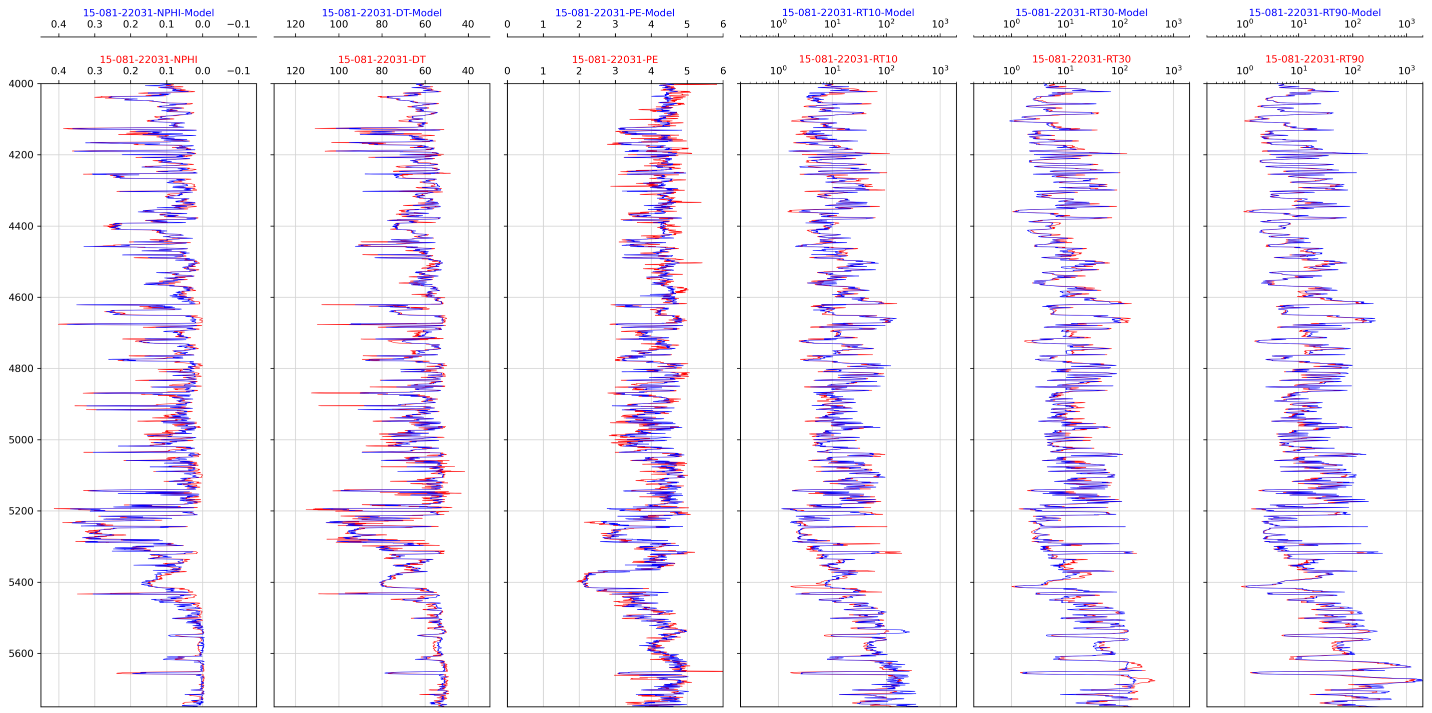


Figure 6.28: Comparison of model performance for a well used in training.

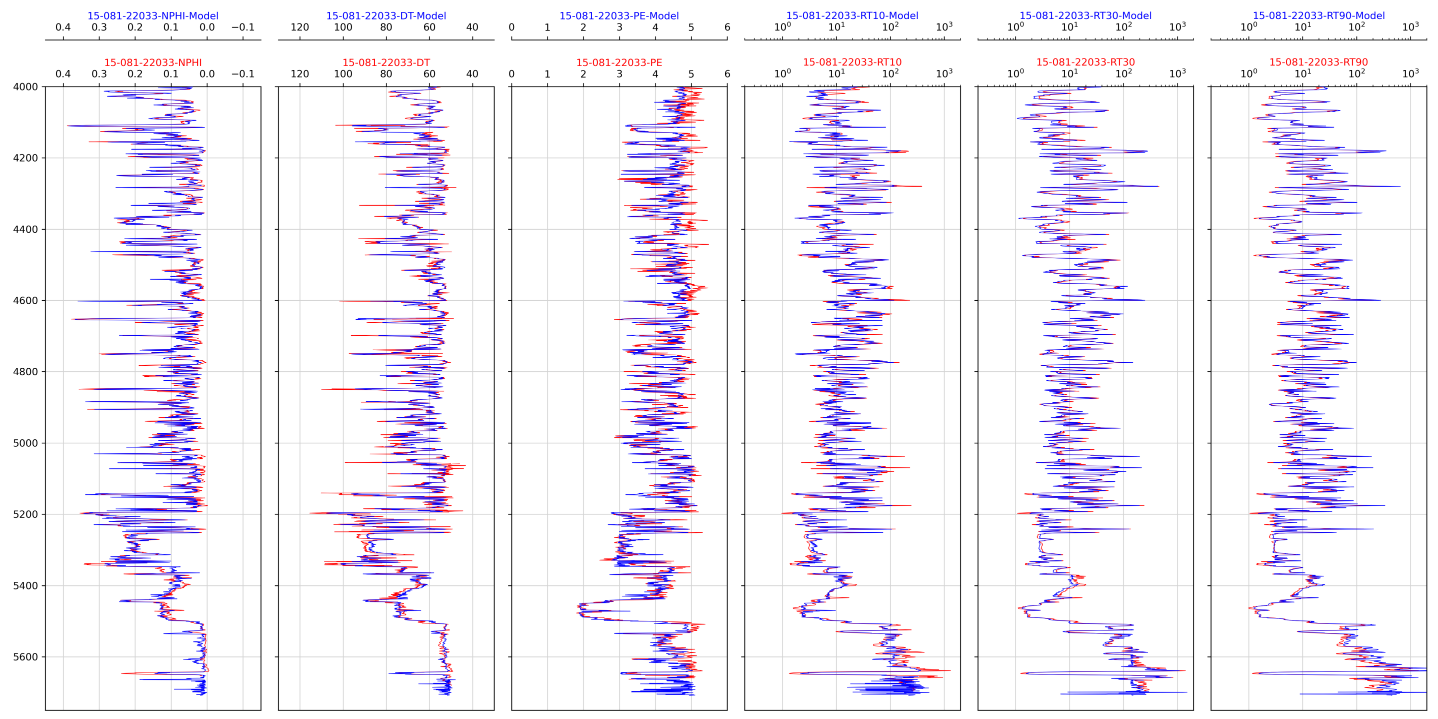


Figure 6.29: Comparison of model performance for a well used in training.

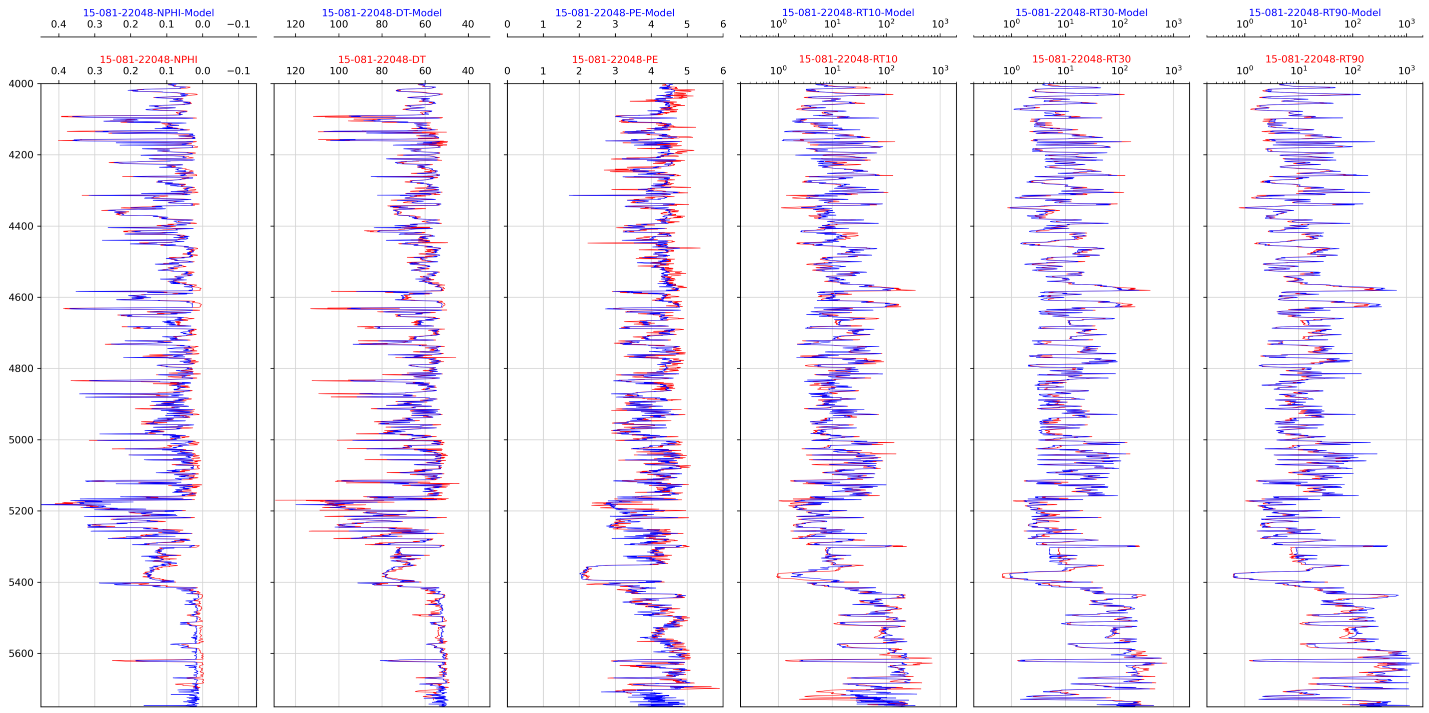


Figure 6.30: Comparison of model performance for a well used in training.

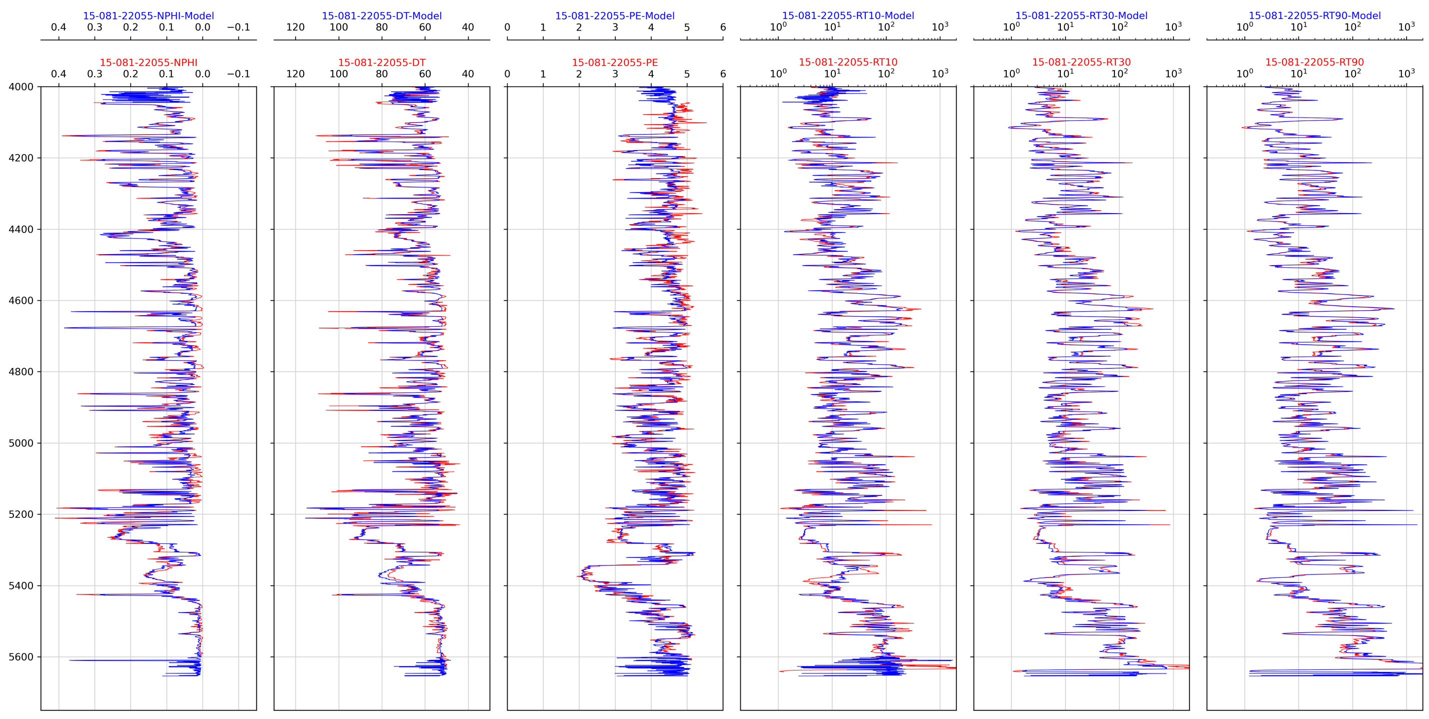


Figure 6.31: Comparison of model performance for a well used in training.

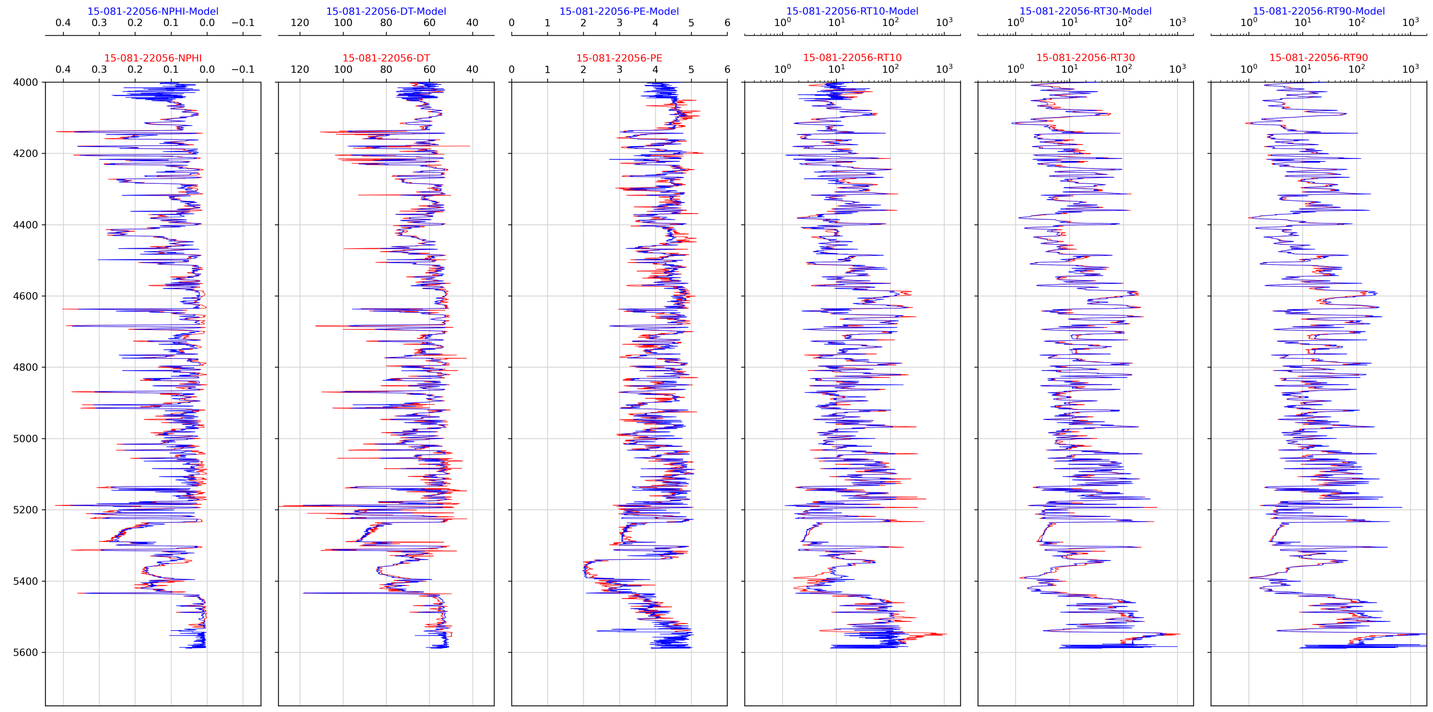


Figure 6.32: Comparison of model performance for a well used in training.