Click to view slide presentation

# EA Supervised Learning Applied to Rock Type Classification in Sandstone Based on Wireline Formation Pressure Data*

**Jose Victor Contreras**[1]

Search and Discovery Article #42539 (2020)**
Posted June 8, 2020

*Adapted from extended abstract prepared in conjunction with oral presentation given at 2019 AAPG Geoscience Technology Workshop, Boosting Reserves and Recovery Using Machine Learning and Analytics, Houston, Texas, January 16-17, 2019
**Datapages © 2020 Serial rights given by author. For all other rights contact author directly. DOI:10.1306/42539Contreras2020

[1]Baker Hughes Company, Houston, TX, United States (jose.contreras@bakerhughes.com)

## Abstract

Wireline formation testing (WFT) tools provide important information about formation pressure, which is the main objective for this kind of log, however, WFT tools deliver a large amount of pressure information with time whose behavior can be related qualitative with rock type, but it is impractical and inefficient to analyze it manually. In reservoirs with little changes in fluid viscosity, it is possible to develop a pattern recognition workflow, based on supervised learning, to identify pressure behavior in different type of rocks and associate this behavior to the corresponding rock type.

A workflow for pattern recognition based on supervised learning was developed to classify the different pressure behaviors and assign those comportment to the corresponding rock type. A supervised Neural Network algorithm was used to train and automatically identify and classify pressure behaviors. This workflow based on supervised learning, assigned type of rocks to each pressure station, according to pressure behavior classification.

During the workflow development, first, data set was processed to extract different pressure behavior characteristics, which were captured in the form of mathematical indicators, such as, pressure stabilization, pressure repeatability, pressure drop, withdrawal rate, etc. Then, a supervised learning algorithm was applied to these indicators, in order to classify wireline pressure responses according to a previously known pattern. Finally, these classification results were associated to the corresponding type of rocks.

This effort showed that supervised learning is a potential tool to identify and classify rock types, based on wireline formation pressure data, due to the tremendous capacity for pattern classification. This workflow makes information obtained from WFT tools more efficient, in terms of

additional information linked directly to type of rocks, and in some cases, could result in cost reductions compared with core analysis once this workflow has been tuned in a reservoir.

## Introduction

Rock type identification is an important process developed in reservoir characterization studies that help to identify and estimate different petrophysical and geological parameters to describe the storage capacity and transmissibility characteristics in porous media. This assessment is particularly useful at the time of studying and assigning rock properties along different geological units. After, these characteristics are established, it is simple to classify geological units that share similar characteristics.

Wireline formation testing tools provide some information that can qualitatively be related with petrophysical characteristics. For example, pressure behavior characteristics for each depth, such as, pressure stabilization, pressure repeatability, pressure drop, withdrawal rate, etc., could be captured in the form of mathematical indicators and be related to a previous known pattern, where each pattern represents a specific rock type.

Rock-type identification assessment is commonly performed by hand for experts that use their knowledge and expertise to classify geological units that share similar characteristics by using cross plot techniques, histograms plots, etc. In the case of wireline formation testing data, experts have to deal with large amounts of observations and data consisting of several curves from hundreds of wells and several characteristics per curves, which turns this task in a large multidimensional problem. This assessment is very difficult to be performed with simple plots and with traditional methods.

One method that can be used to overcome this sort of limitation is a pattern recognition technique based on supervised learning algorithms. These kind of algorithms makes it possible to classify large amounts of observations (i.e. pressure stations) in certain numbers of classes with a previously known pattern, where each class is comprised of observations that share similar attributes, but different compared with observations from other classes. Supervised learning algorithms have been applied effectively in many other Oil and Gas disciplines such as geophysics, petrophysics, reservoir and production engineering. Successful utilizations have been encountered in seismic facies classification (Hami-Eddine et al., 2009 and Roy et al., 2013) and shale discrimination (Hoeink and Zambrano, 2017). Among supervised learning algorithms, artificial neural network is frequently used to accomplish this task.

This paper proposes a workflow to identify and automatically classify rock types based on supervised learning and WFT data attributes. The theory of supervised learning algorithms and details of the workflow construction will be discussed. The results of two WFT data sets will be presented and compared with results from traditional processes to classify different rock types. This comparison demonstrates the potential of supervised learning algorithm to classify rock types.

## Supervised Learning

Supervised learning is a machine learning algorithm that learn from input and output pair of data. These algorithms are called supervised learning because a “teacher” provides supervision to the algorithms in the form of the desired outputs that they have to learn for each example (Müller and Guido, 2017). The teacher has the knowledge of the environment, and the knowledge is represented by a set of input–output examples (Haykin, 2008). Supervised learning algorithms learn to associate some input with some output, given a training set of examples of inputs $x$ and outputs $y$. In many cases the outputs $y$ may be difficult to collect automatically and must be provided by a human “supervisor,” but the term still applies even when the training set targets were collected automatically (Goodfellow et al., 2016). While creating a dataset of inputs and outputs is often a laborious manual process, supervised learning algorithms are well understood and their performance is easy to measure (Müller and Guido, 2017).

Supervised learning relies on the availability of a training sample of labeled examples, with each example consisting of an input signal (stimulus) and the corresponding desired (target) response. In practice, the collection of labeled examples is a time-consuming and expensive task, especially when dealing with large-scale learning problems (Haykin, 2008).

Supervised learning can be explained as follows. In the most general scenario, it is assumed that examples are independently and identically distributed (i.i.d.) according to some fixed but unknown distribution $D$. The learning problem is then formulated in the way that, the learner considers a fixed set of possible concepts $H$, called a hypothesis set, which might not necessarily coincide with a concept class $C$, which is a set of concepts we may wish to learn. It receives a sample $S = (x_1, \ldots, x_m)$ drawn i.i.d. according to $D$, as well as the labels $(c(x_1), \ldots, c(x_m))$, which are based on a specific target concept $c \in C$ to learn. The task is to use the labeled sample $S$ to select a hypothesis $h_S \in H$ that has a small generalization error with respect to the concept $c$. The generalization error of a hypothesis $h \in H$, also referred to as the risk or true error of $h$ is denoted by $R(h)$ and defined as follows (Mohri et al., 2018).

$$R(h) = \underbrace{\mathbb{P}}_{(x,y)\sim\mathcal{D}} [h(x) \neq y] = \underbrace{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[1_{h(x)\neq y}\right]$$

This more general scenario is referred to as the stochastic scenario. Within this setting, the output label is a probabilistic function of the input. In a simpler explanation, supervised learning scenario consists of using a finite sample of labeled examples as training data to make accurate predictions about unseen examples.

Supervised learning is a closed-loop feedback system, where the error is the feedback signal. The error measure, which shows the difference between the network output and the output from the training samples, is used to guide the learning process. The error measure is usually defined by the mean squared error (MSE):

$$E = \frac{1}{N}\sum_{p=1}^{N}\left\|y_p - \hat{y}_p\right\|^2$$

Where $N$ is the number of pattern pairs in the sample set, $y_p$ is the output part of the $p_{\text{th}}$ pattern pair, and $\hat{y}_p$ is the network output corresponding to the pattern pair $p$. The error $E$ is calculated after each epoch. The learning process is terminated when $E$ is sufficiently small, or a failure criterion is met. To decrease $E$ toward zero, a gradient-descent procedure is usually applied (Du and Swamy, 2013).

There are two major types of supervised machine learning problems, called classification and regression. In classification, the goal is to predict a class label, which is a choice from a predefined list of possibilities. Classification is sometimes separated into binary classification, which is the special case of distinguishing between exactly two classes, and multiclass classification, which is classification between more than two classes. In the case of regression, the goal is to predict a continuous number or a floating-point number in programming terms (Müller and Guido, 2017).

There are several types of algorithms used to perform supervised learning tasks, but the most important algorithms are (Géron, 2017): k-Nearest Neighbors, Linear Regression, Logistic Regression, Support Vector Machines (SVMs), Decision Trees and Random Forests, and Artificial Neural networks.

## Capacity, Overfitting, and Underfitting

The main challenge in machine learning is to create a model that make accurate predictions on new unseen inputs, that mean, not just those data on which the model was trained. Unseen data must have the same characteristics as the training set that was used. The ability to perform well on unobserved inputs is called generalization. When a machine learning model is being trained the training set is available, then it is possible to compute some error measure on the training set, called the training error, and minimize this training error. Additionally, it is important to estimate the generalization error, or test error, as the objective in machine learning is to reduce the training error and the generalization error at the same time. The generalization error estimation in a machine learning model measures its performance on a test set of examples that were collected separately from the training set. The generalization error is defined as the expected value of the error on a new input (Goodfellow et al., 2016).

When building a machine learning model, it is expected that simple models may generalize better to new data. Therefore, it is wanted to find the simplest model. Building a model that is too complex for the amount of available information, is called overfitting. Overfitting occurs when a model is fitted too closely to the particularities of the training set, obtaining a model that works well on the training set, but it is not able to generalize to new data. On the other hand, if the model is too simple, it might not be able to capture all the aspects of and variability in the data, and the model will perform badly even on the training set. Choosing a very simple model is called underfitting (Müller and Guido, 2017).

There is a sweet spot between overfitting model and underfitting model that yields the best generalization performance. This is the model that it is wanted to build. The trade-off between overfitting zone and underfitting zone is illustrated in Figure 1 (Müller and Guido, 2017).

## The Bias - Variance Tradeoff

The field of statistics gives several tools to achieve the machine learning goal of solving a task not only on the training set but also to generalize. An important theory from statistics and Machine Learning fields is that a model’s generalization error can be expressed as the sum of three different errors. One error is the bias, which measures the expected deviation from the true value of the function or parameter and is the part of the generalization error due to wrong assumptions. A high-bias model is most likely to underfit the training data. The other error is variance that provides a measure of the deviation from the expected estimator value that any particular sampling of the data is likely to cause. This error is generated due to the model’s excessive sensitivity to small variations in the training data. A model with many degrees of freedom is likely to have high variance, and thus to overfit the training data. The third error is the irreducible error, this one appears as a consequence of noise in the data itself. The only way to reduce this part of the error is by cleaning up the data (Géron, 2017).

Increasing a model’s complexity will typically increase its variance and reduce its bias. Conversely, reducing a model’s complexity increases its bias and reduces its variance. This is why it is called a tradeoff (Géron, 2017). The most common way to negotiate this trade-off is by using cross-validation technique. Empirically, cross-validation is highly successful on many real-world tasks (Goodfellow et al., 2016).

## Evaluating Machine-Learning Models

Typically, model’s performance on never-before-seen data stalled or worsened, compared to model’s performance on the training data, which always improves as training progresses. This effect is called overfit. In machine learning, the objective is to build models that perform well on never-before-seen data (generalization), and overfitting is the central obstacle (Chollet, 2018).

The only way to know how well a model will generalize to new cases is to try it out on new cases. The best strategy to mitigate overfitting and maximize generalization is by splitting the data into three different sets: the training set, the validation set, and the test set. This implies that the model has to be trained by using the training set, make a validation process with validation set, and once the model is ready, test it using the test set. The error rate on new cases is called the generalization error, and by evaluating the model on the test set, it is possible to get an estimation of this error. This value indicates how well the model will perform on new instances it has never seen before. It is common to use 80% of the data for training and hold out 20% for testing (Géron, 2017).

## Splitting Data Techniques

There are several strategies commonly utilized to split data sets in different groups like training set, validation set, and test set. In the following paragraph, two of the most important techniques will be discussed.

### Simple Hold-Out Validation

This strategy sets apart some fraction of the data as test set (Chollet, 2018). The remaining data is used as the train set and validation set. It is very important to prevent information leaks from the test set; the model must not be tuned based on the test set. For tune purposes reserve the validation set, on which the model must be tuned. It is common to use 60% of the data for training, 20% for validating, and reserves 20% for testing. Figure 2 depicts a simple scheme of implementation for hold-out validation

**K-Fold Validation**

This strategy is frequently used when, so few data points are available to build a machine learning model (for example, 100 data points). In this case, the validation set would end up being very small, in consequence, the validation error might change a lot depending on which data points are chosen for training and validation purpose. This situation could prevent a reliable model evaluation, and the best practice is to use K-fold cross-validation approach.

K-fold cross-validation approach consists of splitting the available data into K partitions, creating K identical models (Figure 3), and training each one on K–1 partitions while evaluating on the remaining partition. The validation score for the model used is then the average of the K validation scores obtained (Chollet, 2018). Like hold-out validation approach, this method does not exempt from using a distinct validation set for model calibration.

**Confusion Matrix**

Confusion matrix is a tool widely used to evaluate the model or classifier's performance. The general idea is to count the number of times a specific class is classified correctly or incorrectly. To compute the confusion matrix, a set of prediction instances have to be compared to the actual targets in a matrix representation. In a confusion matrix, each row represents an actual class, while each column represents a predicted class (Géron, 2017).

The confusion matrix gives a lot of information, nevertheless, it is preferred to have more concise performance information in the form of metrics. There are several types of metrics that can be calculated from confusion matrix and give quantitative indictors about models or classifiers' performance. One of the most used indicator is precision, which is the accuracy of the positive predictions, and can be calculated as follow (Géron, 2017).

$$Precision = \frac{TP}{TP + FP}$$

Where TP is the number of true positives, and FP is the number of false positives.

Precision is typically used along with another metric named recall or sensitivity or true positive rate (TPR). This is the ratio of positive instances that are correctly detected by the classifier (Géron, 2017).

$$Recall = \frac{TP}{TP + FN}$$

Where FN is the number of false negatives.

## Artificial Neural Network

Inspired by animal brain's architecture (Géron, 2017), artificial neural network (ANN) is a collection of connected nodes (artificial neurons), which try to model the neurons in a biological brain. Each connection can transmit a signal from one neuron to other neurons. An artificial neuron that receives a signal processes it and sends the signal to other neurons connected to it.

An ANN is composed of input, hidden, and output layers, which are chained together and where each layer has several neurons. The loss function, which defines the feedback signal used for learning, and the optimizer, which determines how learning proceeds are part of the ANN anatomy. The network maps the input data to predictions. The loss function compares these predictions to the targets, producing a loss value (a measure of how well the network's predictions match what it is expected), and then the optimizer uses this loss value to update the network's weights (Chollet, 2018).

The fundamental data structure in neural networks is the layer. A layer is a data-processing module that takes as input one or more tensors and that outputs one or more tensors. Some layers are stateless, but more frequently layers have a state. The layer's weights, one or several tensors learned with stochastic gradient descent, which together contain the network's knowledge. Loss functions and optimizers are very important parameters to configure during the learning process. Loss function, which is the objective function, is the quantity that will be minimized during training and represents a measure of success for the task performed. The optimizer determines how the network will be updated based on the loss function. It implements a specific variant of stochastic gradient descent (SGD) (Chollet, 2018).

A neural network that has multiple outputs may have multiple loss functions. But the gradient-descent process must be based on a single scalar loss value; so, for multi-loss networks, all losses are combined (averaging) into a single scalar quantity.

### The Perceptron

The Perceptron is one of the simplest ANN architectures, invented in 1957 by Frank Rosenblatt. In this network inputs and outputs are numbers and each input connection is associated with a weight. Each unit or neuron computes a weighted sum of its inputs, then applies an activation or step function to that sum and outputs the result (Figure 4). A Perceptron is composed of a single layer of linear threshold unit (LTU), with each neuron connected to all the inputs through a weight. These connections are often represented by neurons called input neurons, and they just output whatever information they are fed. Additionally, each neuron has a weighted sum of inputs and an extra bias feature to form its own scalar. The bias feature is represented by a special type of neuron called a bias neuron, which just outputs 1 all the time (Géron, 2017). Then, the scalar passes to the step or activation function to get another modified scalar. Finally, the neuron layer outputs form a column vector (Figure 5).

In a mathematical standpoint, each neural layer transforms its input data as follows:

$$h_w(x) = step(W \cdot x + b)$$

Where *x* is the input, *W* and *b* are called the weights and bias, respectively, and they are tensors that are attributes of the layer. These weights contain the information learned by the network from exposure to training data. Initially, these weight matrices are filled with small random values, however, during the learning process weights are gradually adjusted, based on a feedback signal. This gradual adjustment or training is basically the core of machine learning processes (Géron, 2017).

This process takes place in a training loop, which works as follows. The input signal (stimulus or *x*) comes in the network and propagates forward through the network, emerging at the output end of the network to obtain predictions. This signal is calculated as a function of the inputs and associated weights applied to neurons. Afterwards, the loss of the network on the batch is computed, which is the measure of the mismatch between predictions (*y_pred*) and real values (*y*). For every output neuron that produced a wrong prediction, it reinforces the connection weights from the inputs that would have contributed to the correct prediction. Repeating these steps in a loop, as long as necessary, will eventually end up with a network that has a very low loss on its training data. Finally, the network will learn to map its inputs to correct targets (Chollet, 2018).

## Multilayer Perceptron

Multilayer perceptron neural network consists of one input layer, one or more hidden layers, and one final output layer. Every layer except the output layer includes a bias neuron which is fully connected to the next layer. When an ANN has two or more hidden layers, it is called a deep neural network (DNN) (Géron, 2017). The most common multilayer perceptrons have a single hidden layer, containing two layers of neurons not counting the inputs (MacKay, 2005). The number of hidden layers and neurons per layer vary according to the complexity of the situation that is been modeled. Nevertheless, it is recommended to keep the number of layers and neurons as low as possible. Figure 6 shows the architecture of a multilayer perceptron with multiple layers and an output layer, and the network fully connected.

In 1986, (Rumelhart et al., 1986) published an article introducing the backpropagation training algorithm. In this algorithm each hidden or output neuron of a multilayer perceptron perform the following computations. First, in this algorithm the network feeds on training instance and computes the output of every neuron in each consecutive layer (forward pass, making predictions). Then it measures the network's output error (for example, the difference between the desired output and the actual output of the network), and it computes how much each neuron in the last hidden layer contributed to each output neuron's error. Afterwards, it proceeds to measure how much of these error contributions came from each neuron in the previous hidden layer, and so on until the algorithm reaches the input layer (forward pass). This reverse pass efficiently measures the error gradient across all the connection weights in the network by propagating the error gradient backward in the network (Figure 7). Finally, the algorithm slightly tweaks the connection weights to reduce the error (Gradient Descent step) (Géron, 2017).

The backpropagation algorithm is required to use a well-defined nonzero derivative function, to allow the Gradient Descent to make some progress at every step and work properly. Some of the most popular activation functions are the hyperbolic tangent function, sigmoid function, linear function, etc.

## Activation Function

Activation functions are mathematical equations attached to each neuron in the network, which help to determine the output of a neural network or indicate if a neuron should be activated or not.

There are several activation functions (Figure 8), and the most popular are described below (MacKay, 2005):

Linear

$$y(a) = a$$

Sigmoid (logistic function)

$$y(a) = \frac{1}{1 + e^{-a}} \qquad (y \in (0,1))$$

Sigmoid (tanh)

$$y(a) = tanh(a) \qquad (y \in (-1,1))$$

Hard-limit function

$$y(a) = \text{hardlim}(a) \equiv \begin{cases} 1 & a > b \\ 0 & a \leq b \end{cases}$$

## Building the Workflow

### Data Set

The wireline pressure information used in this study belongs to several vertical wells from the same reservoir that were drilled in a sequence of interbedded shale sandstone formations. This data set comprises depth, formation pressure and drawdown rate information available for each pressure station. A quality control process was performed on the raw data.

## Data Preparation

The data set was processed to extract different pressure behavior characteristics, which were captured in the form of mathematical indicators, such as, pressure stabilization, pressure repeatability, pressure drop, withdrawal rate, etc. (Figure 9). Then, the generated information was organized in an input matrix in order to be used in the supervised learning algorithm. In the input matrix, each *ith* column has the elements that represent the measurements taken from each pressure station, as well as the output indicator associated with the corresponding type of rock. In the same manner, each *ith* row represents the number of observations. In this case, the data set used to train the model produced a matrix that has nine (9) columns, in which the first five (5) columns represent the characteristics from each rock type and the last four (4) columns represent the corresponding type of rocks tied to those characteristics (Figure 10). Additionally, there are 170 rows containing all observations available in all reservoir.

An independent study was previously carried out on this reservoir, in order to associate the different type of rocks with the corresponding characteristics extracted from WFT data.

## Applying the Supervised Learning Algorithm

### Creating the ANN Model

The model was created with a neural network that will learn how to classify pressure stations and assign those results to the corresponding rock type. The first step is to initialize the ANN, where it specified the number of layers, number of neurons per layer, transfer function, weight/bias learning functions, and the type of training algorithm. In this case, the ANN was initialized with one (1) hidden layer, ten (10) neurons in the hidden layer, and LOGSIG as transfer function.

### Validation Process

The data set was subdivided into three different groups in order to carry out the training and the validation process to build the model. In this work, the data set was subdivided in 60% of data for training, 20% for validation, and 20% for testing process.

### Building the Model

Model construction process consists of the selection of the optimum number of layers, number of neurons per layer, and transfer functions, by carrying out a sensitivity analysis using only the training and validation data. The selection of the right ANN architecture minimizes the balanced error rate (BER) and increases the model's accuracy. A sensitivity analysis was performed with just one hidden layer, varying the number of neurons in the hidden layer 10, 12, 15, and 20, and transfer functions, in order to study and analyze the model performance of different ANN arrangements in the classification process. The models' performance evaluation was done by using confusion matrix (Figure 11), estimating the accuracy in the classification process and the precision to classify rock type 3.

Table 1 summarizes different architectures of neural networks (number of layers, neurons, transfer functions, etc.) and their performance during the training process.

Table 1 shows that improvements were obtained in model results when the number of neurons in the hidden layer was increased from 10 to 12 and then from 12 to 15. Furthermore, for the same number of neurons in the hidden layer, additional improvements were observed when the transfer function in the hidden layer was changed from TANSIG to LOGSIG.

From models studied in Table 1, model #5 was selected, since it is the model that has the highest accuracy and precision. Although the model #7 has the lowest BER value, the small improvement in BER value does not justify selecting a more complex model, and it is truer when these models (5 and 7) have the same ability to classify observations.

## Identification of Type of Rocks

Once the supervised neural network model has been trained, it can be used to carry out the rock type classification based on pressure station.

### Example 1

The available wireline pressure information for well 1 includes depth, formation pressure, and drawdown rate curves for 6 pressure stations. The supervised algorithm was applied to this information and results are shown in Table 2.

The supervised learning algorithm identified two (02) different type of rocks, which are rock type 2 and rock type 3. It is worth noting that rock type 2 is predominant in this well. Figure 12 shows the rock type distribution along well 1.

### Example 2

For well 2, the wireline pressure information includes depth, reservoir pressure, and drawdown rate curves for 7 pressure stations. The supervised algorithm was applied to this information and results are shown in Table 3.

In this case, the supervised learning algorithm also identified two (02) different type of rocks, which are rock type 2 and rock type 3. Nevertheless, in this example rock type 3 is mainly found in the well. Figure 13 depicts the rock type distribution along well 2.

## Discussion

Results presented in this work show that supervised learning is a potential tool to identify and classify rock types, based on WFT data. The workflow followed makes information obtained from WFT tools more efficient, in terms of additional information linked directly to type of rocks, and in some cases, could result in cost reductions compared with core analysis once this workflow has been tuned in a reservoir.

In this work, the classification process relies on differences in mathematical indicators extracted mainly from pressure and drawdown rate curves, reason for which special attention should be paid during the data preparation process.

## Conclusions

Rock types classified with supervised learning algorithm obtained in this work have good results (Accuracy is 94%).

Precision to classify rock types 1, 2, and 4 is 100 %, while precision in the rock type 3 is 87 %.

Pressure behavior characteristics chosen to classify pressure tests in different rock types gave good results.

Supervised learning is a powerful tool to classify efficiently different type of rocks based on wireline formation pressure.

Once the trained supervised learning model is available, the classification process is performed very fast and unifying the criteria, when the job is done by several interpreters.

## References Cited

Chollet, F., 2018, Deep Learning with Python: Manning Publications Co., Shelter Island, New York, USA, p. 29 and 94.

Du, K.-L., and M.N.S. Swamy, 2013, Neural Networks and Statistical Learning: Enjoyor Labs, Enjoyor Inc., Concordia University, Canada, p. 17 and 31.

Géron, A., 2017, Hands-On Machine Learning with Scikit-Learn and TensorFlow – Concepts, Tools, and Techniques to Build Intelligent Systems: O’Reilly Media, Inc., Sebastopol, California, USA, p. 8, 9, 29, 84, 85, and 126.

Goodfellow, I., Y. Bengio, and A. Courville, 2016, Deep Learning: The MIT Press, Massachusetts, USA, p. 100, 111, 119, 122, 123, and 124.

Hami-Eddine, K., P. Klein, and L. Richard, 2009, Well Facies Based Supervised Classification of Prestack Seismic: Application to a Turbidite Field: Abstract, SEG Houston International Exposition and Annual Meeting.

Haykin, S., 2008, Neural Networks and Learning Machines: 3rd ed., Pearson-Prentice Hall, Ontario, Canada, p. 34, 35, 36, 45, 171, 209, and 210.

Hoeink, T., and C. Zambrano, 2017, Shale Discrimination with Machine Learning Methods: 51st US Rock Mechanics/Geomechanics Symposium, San Francisco, California, USA, 25-28 June 2017, ARMA-2017-0769, 6 p.

MacKay, D.J.C., 2005, Information Theory, Inference, and Learning Algorithms: Cambridge University Press, Edinburgh, Cambridge, United Kingdom, p. 471, 472, 473, and 527.

Mohri, M., A. Rostamizadeh, and A. Talwalkar, 2018, Foundations of Machine Learning: 2nd ed., The MIT Press, Massachusetts, USA, p. 6, 7, 9, 10, and 11.

Müller, A.C., and S. Guido, 2017, Introduction to Machine Learning with Python - A Guide for Data Scientists: O'Reilly Media, Inc., Sebastopol, California, USA, p. 2, 25, 27, 28, and 29.

Roy, A., V. Jayaram, and K.J. Marfurt, 2013, Active Learning Algorithms in Seismic Facies Classification: SEG Houston 2013 Annual Meeting, p. 1467-1471.  doi.org/10.1190/segam2013-0769.1

Rumelhart, D.E., G.E. Hinton, and R.J. Williams, 1986, Learning Representations by Back-Propagating Errors: Nature, v. 323, p. 533-536. doi.org/10.1038/323533a.

Russell, S., and P. Norvig, 2016, Artificial Intelligence: A Modern Approach: 3rd ed., Pearson Education, Edinburgh, Harlow, England, p. 695-696.

Smola, A., and S.V.N. Vishwanathan, 2008, Introduction to Machine Learning: Cambridge University Press, Edinburgh, Cambridge, United Kingdom, p. 6.

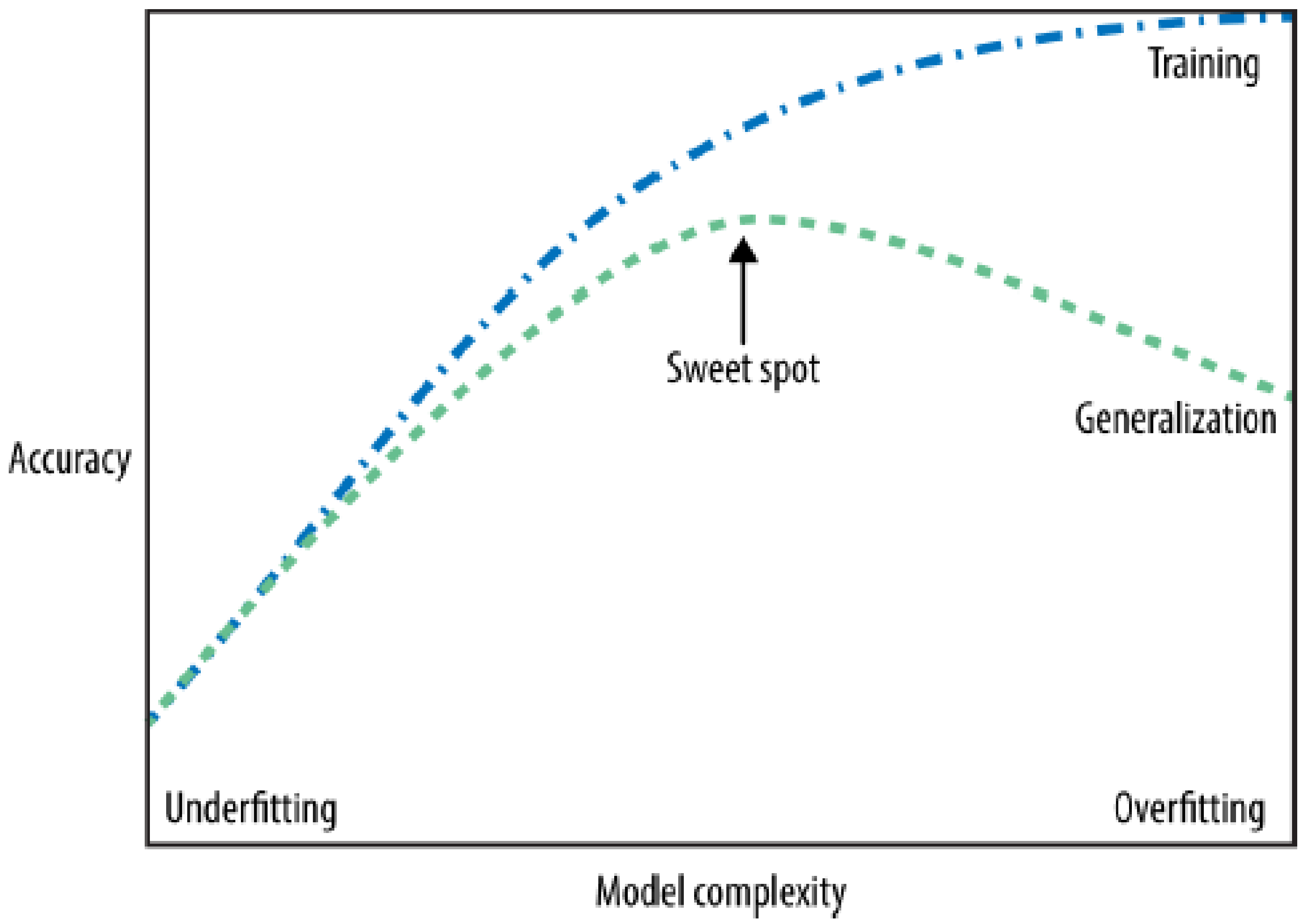


Figure 1. Trade-off of model complexity against training and test accuracy (Müller and Guido, 2017).

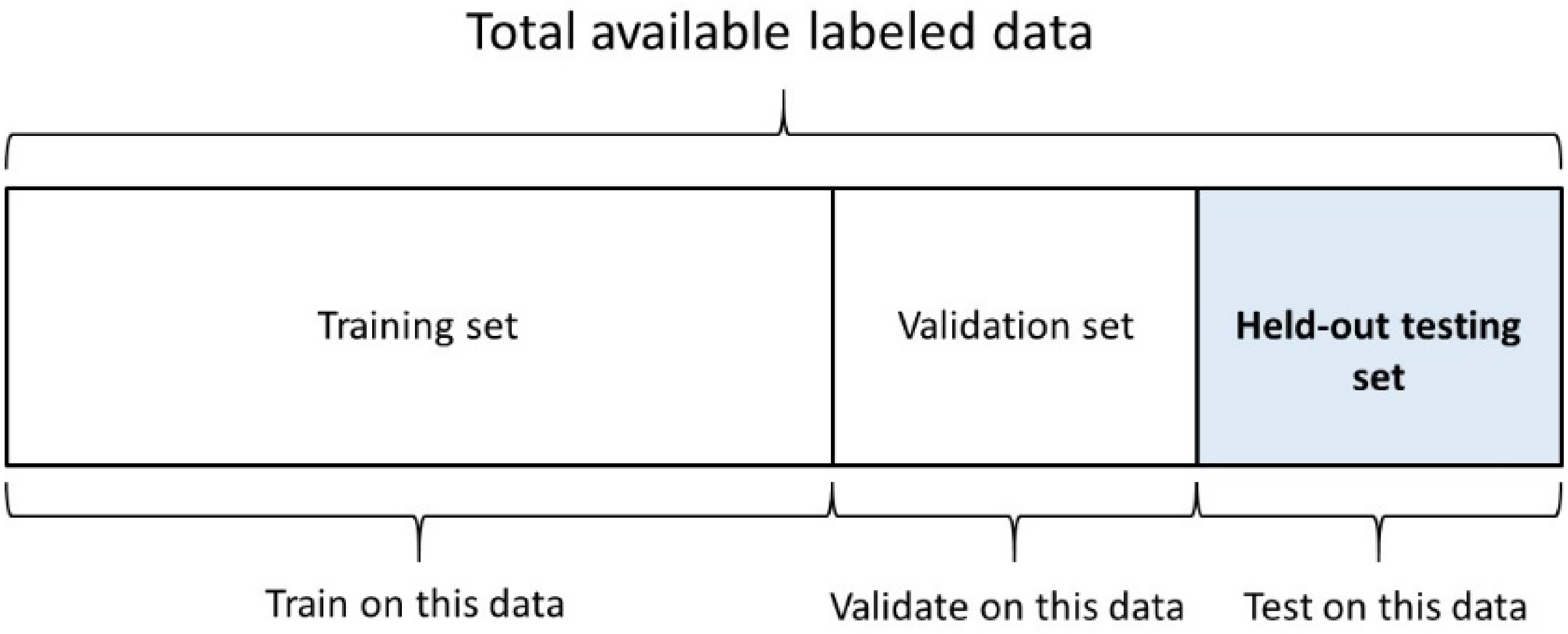


Figure 2. Simple hold-out validation split.

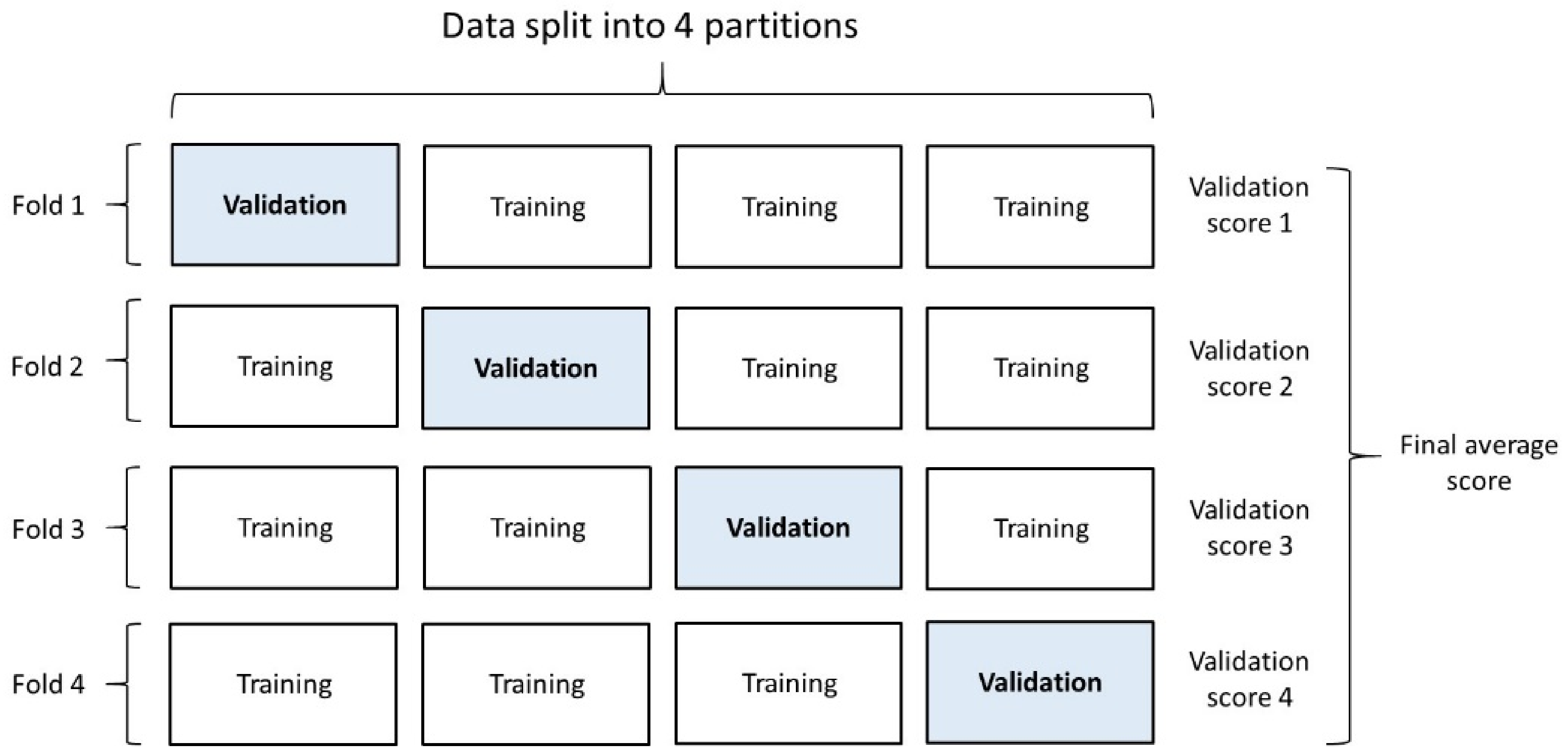


Figure 3. 4-fold cross-validation strategy.

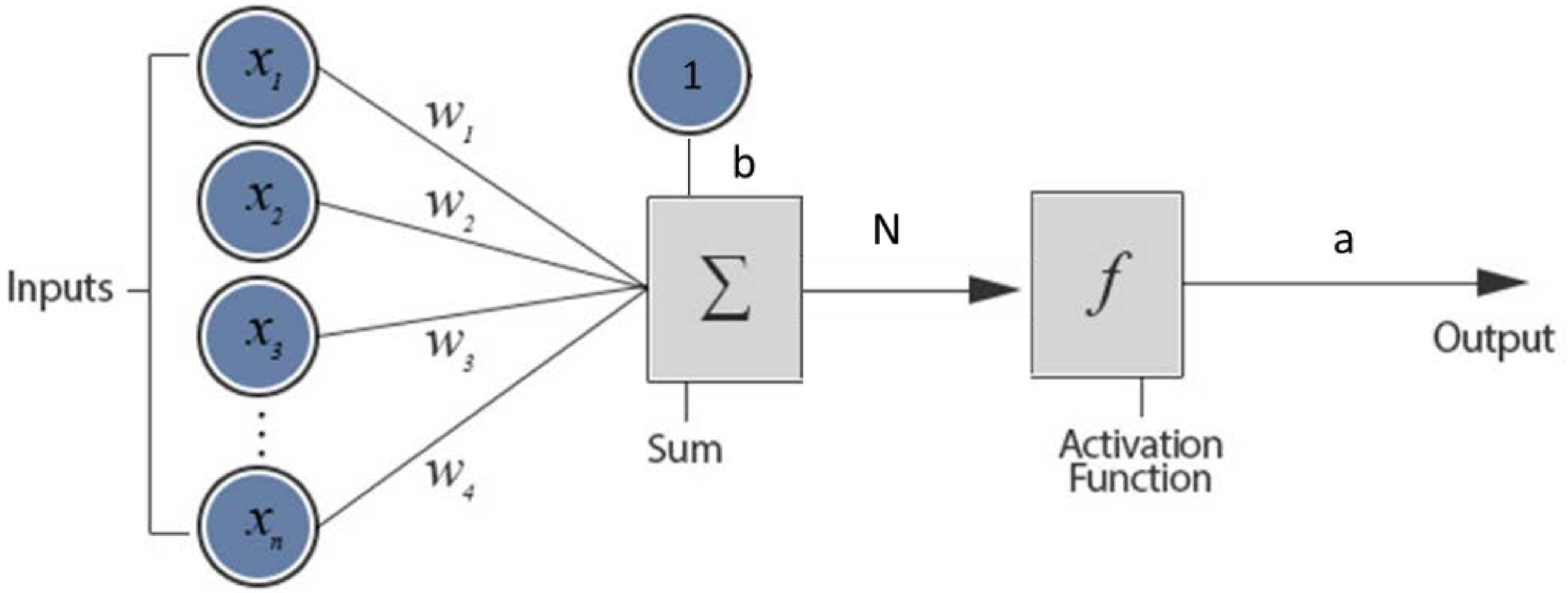


Figure 4. Neuron model.

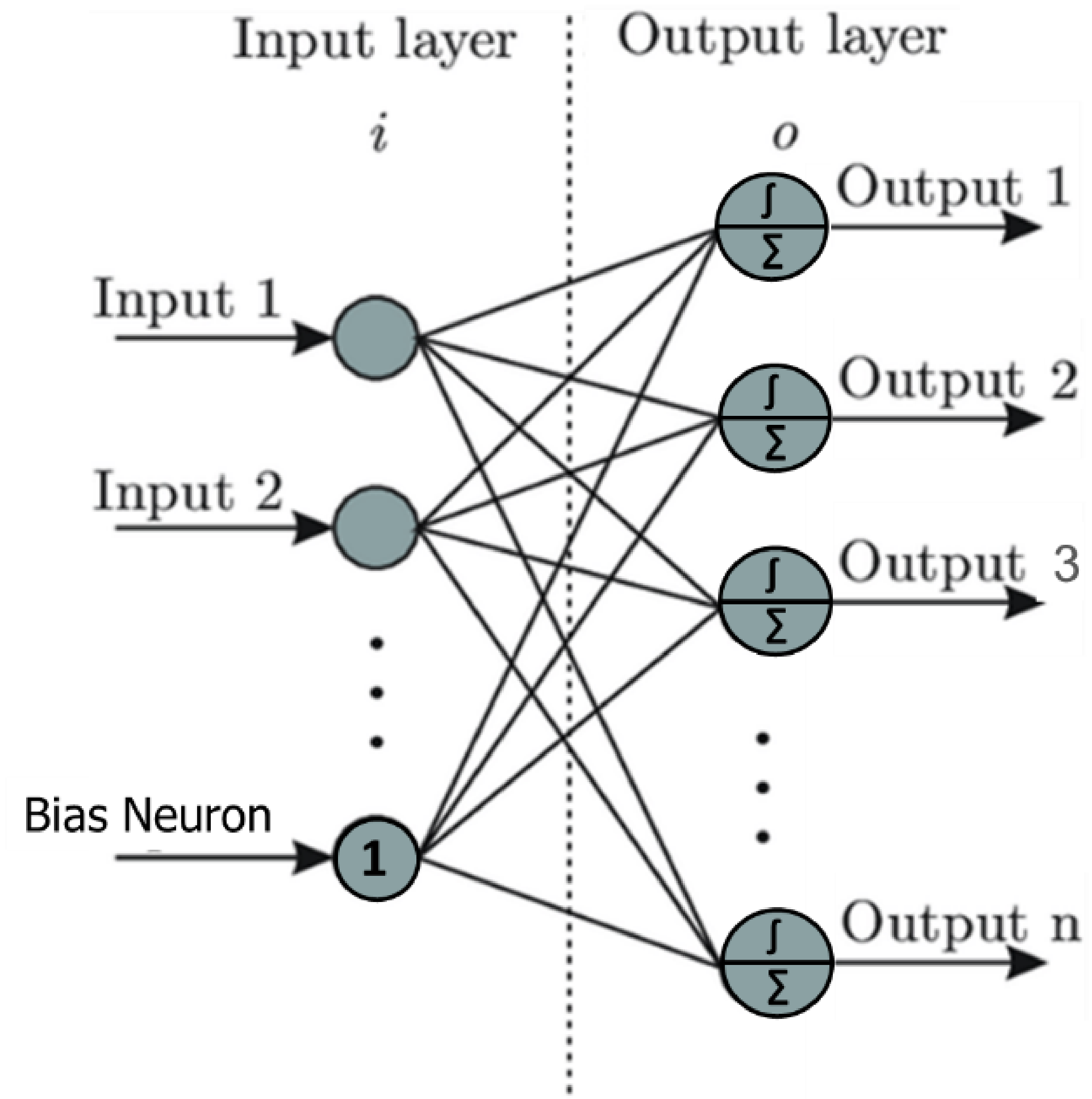


Figure 5. Perceptron Artificial Neural Network architecture.

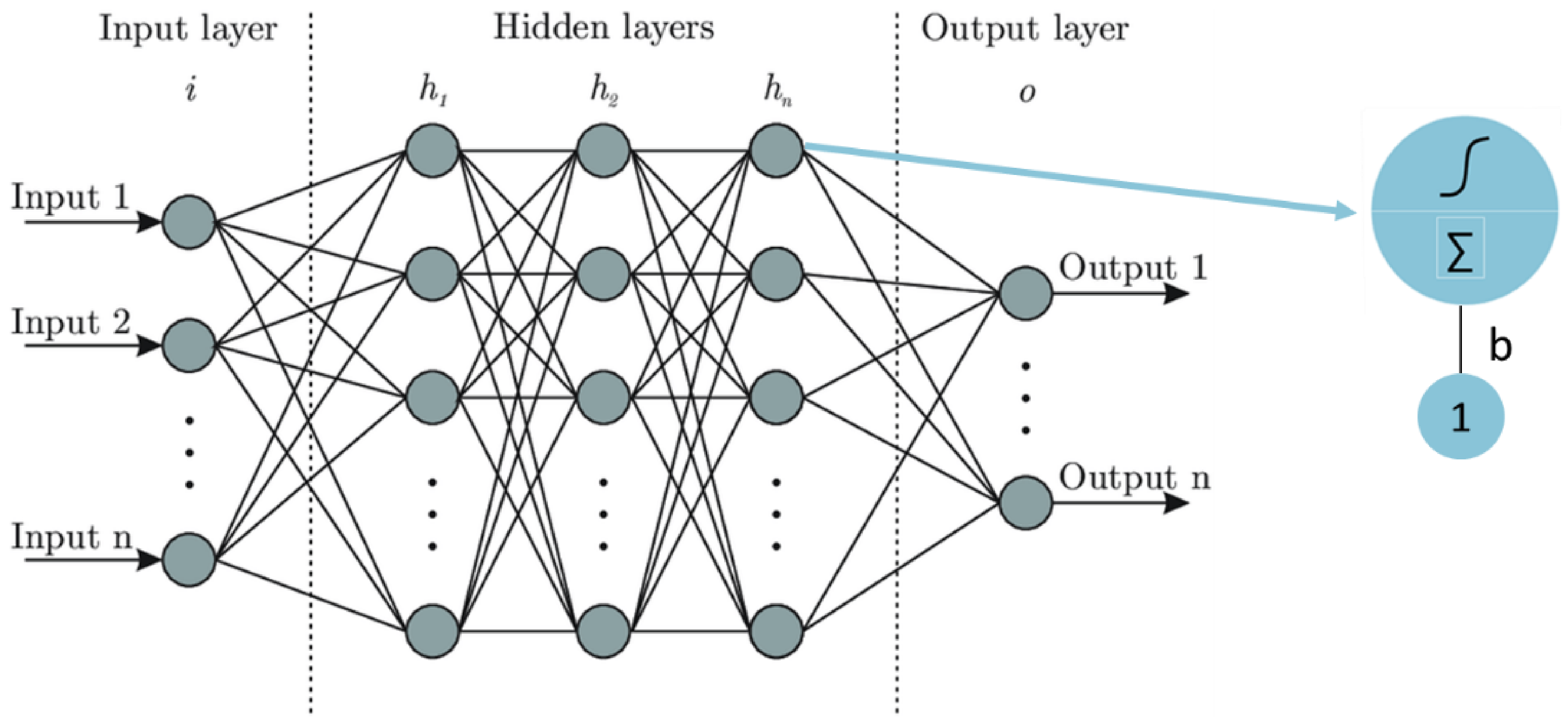


Figure 6. Artificial Neural Network architecture with multiple layers.

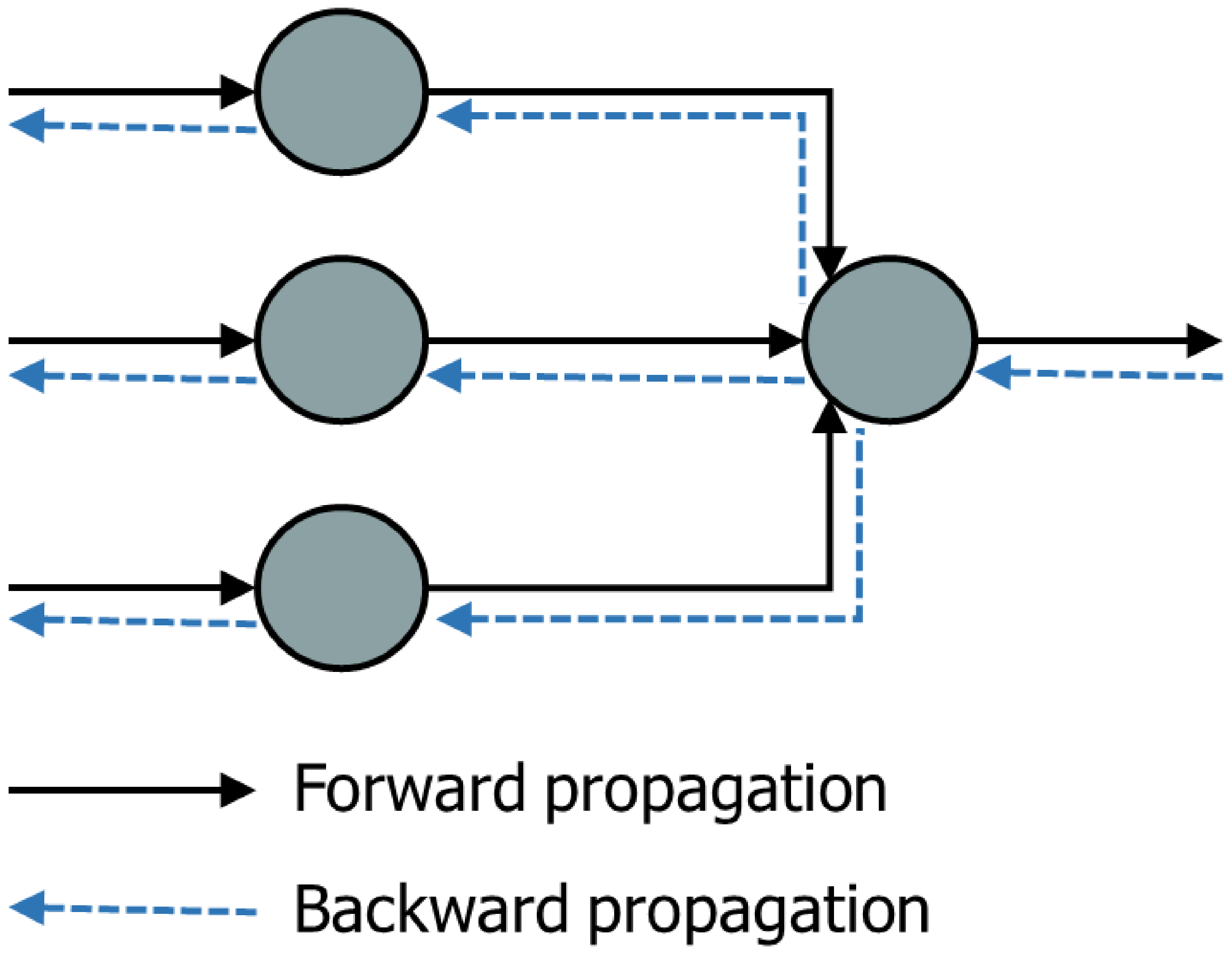


Figure 7. Direction of signal flows in a multilayer perceptron: forward propagation of function signals and back propagation of error signals.

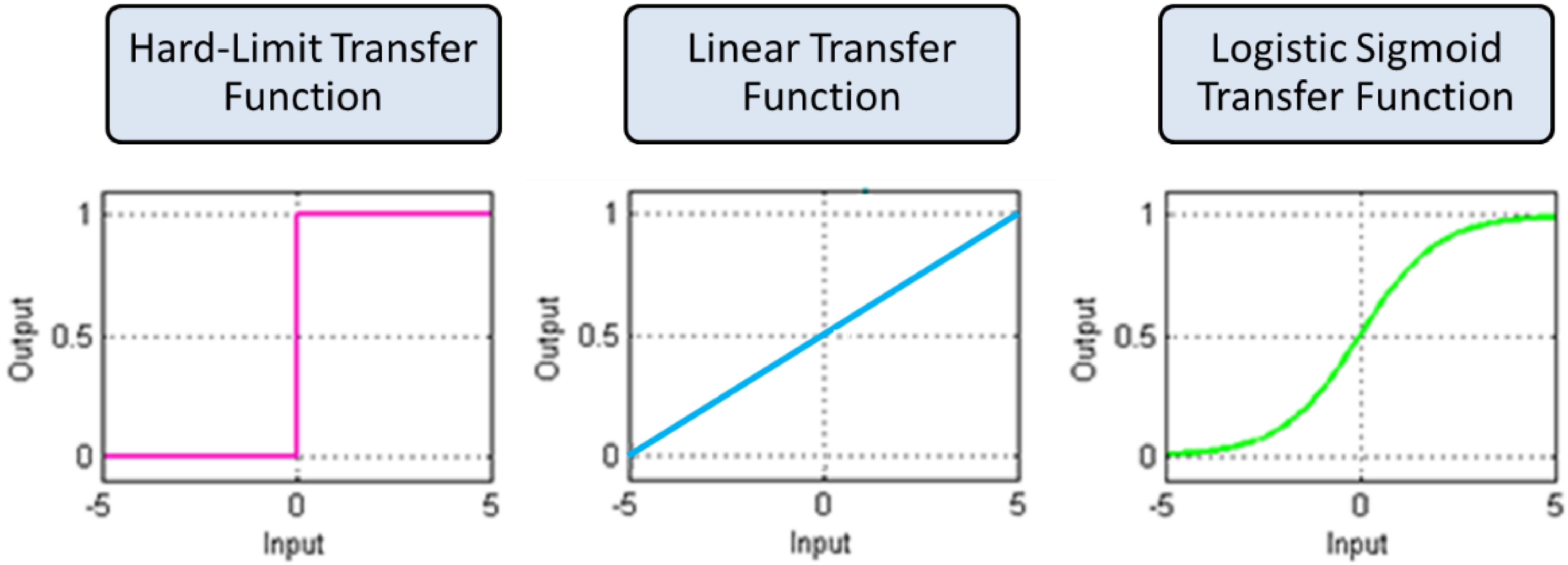


Figure 8. Activation or transfer function.

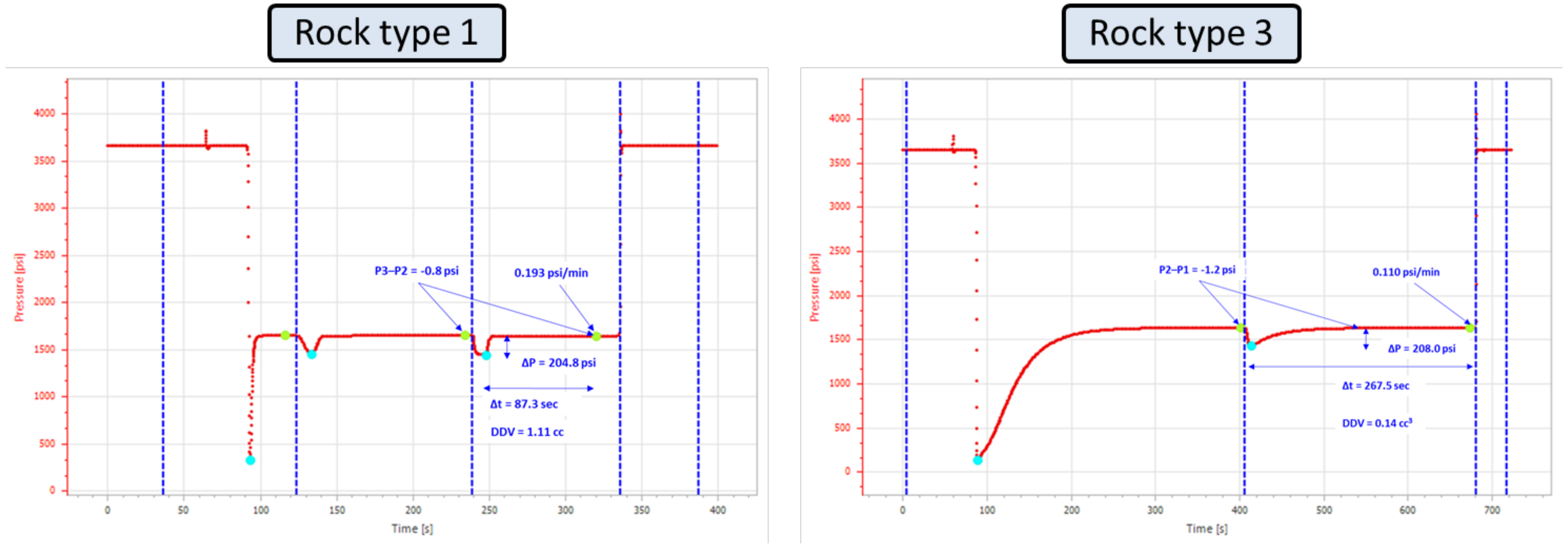


Figure 9. Mathematical indicators captured from pressure and volume curves.

$$input = \begin{pmatrix} S_1 & R_1 & \Delta P_1 & DD_1 & t_1 \\ S_2 & R_2 & \Delta P_2 & DD_2 & t_2 \\ S_3 & R_3 & \Delta P_3 & DD_3 & t_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ S_n & R_n & \Delta P_n & DD_n & t_n \end{pmatrix} \qquad output = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Figure 10. Input and Output matrix generated from data set.

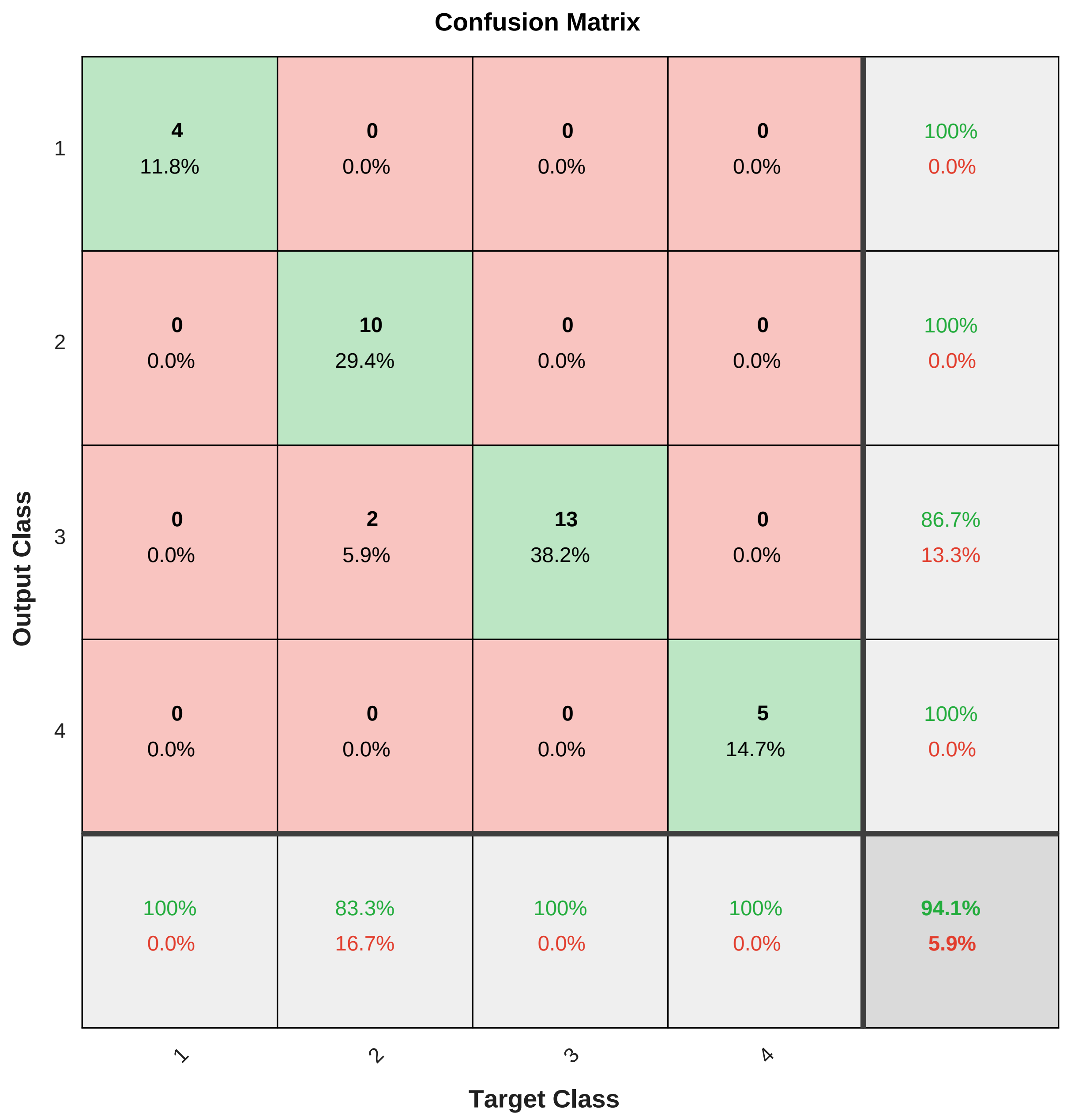


Figure 11. Confusion matrix for the selected model's performance.

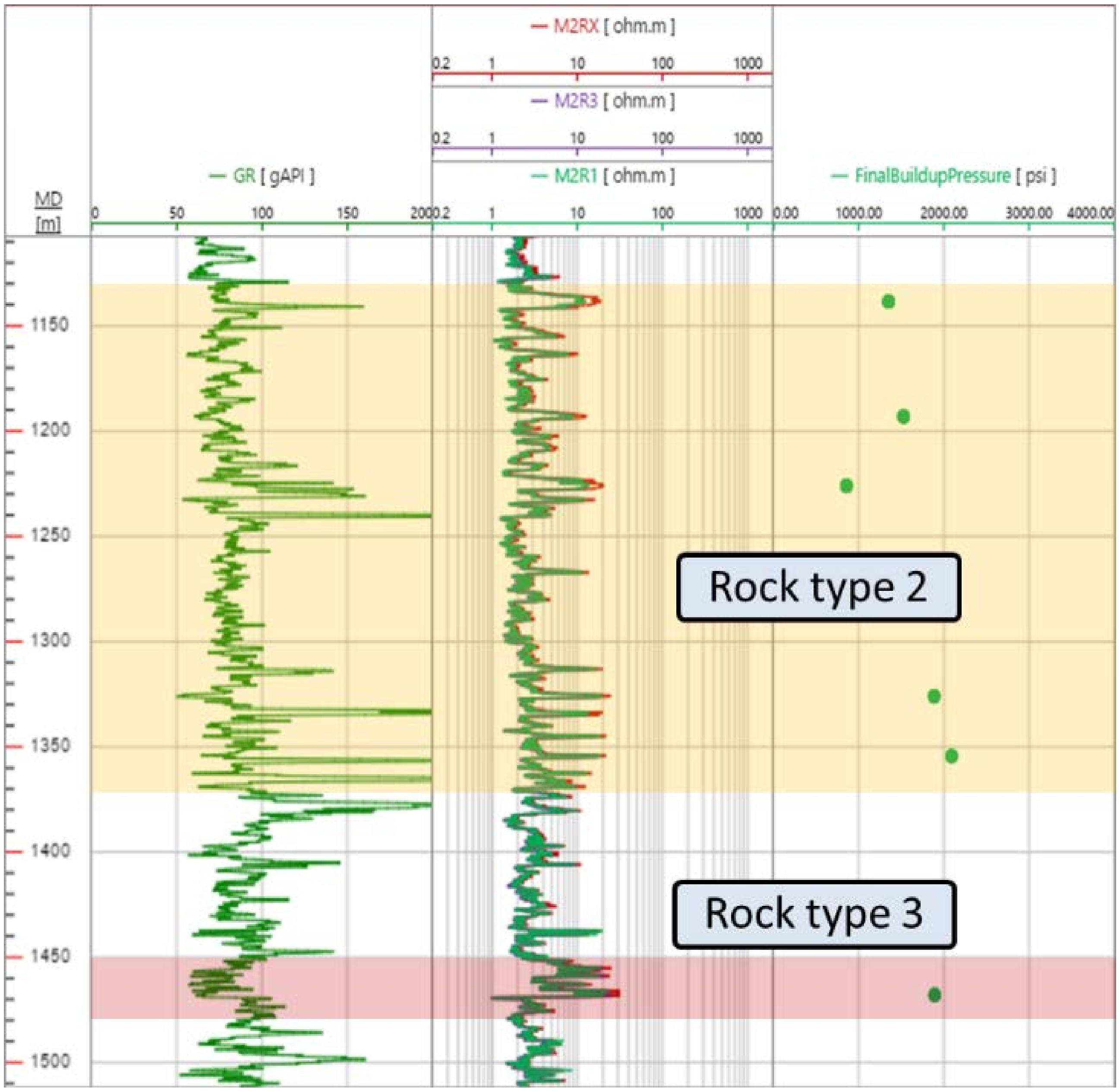


Figure 12. Pressure profile and the rock type classification in example 1.

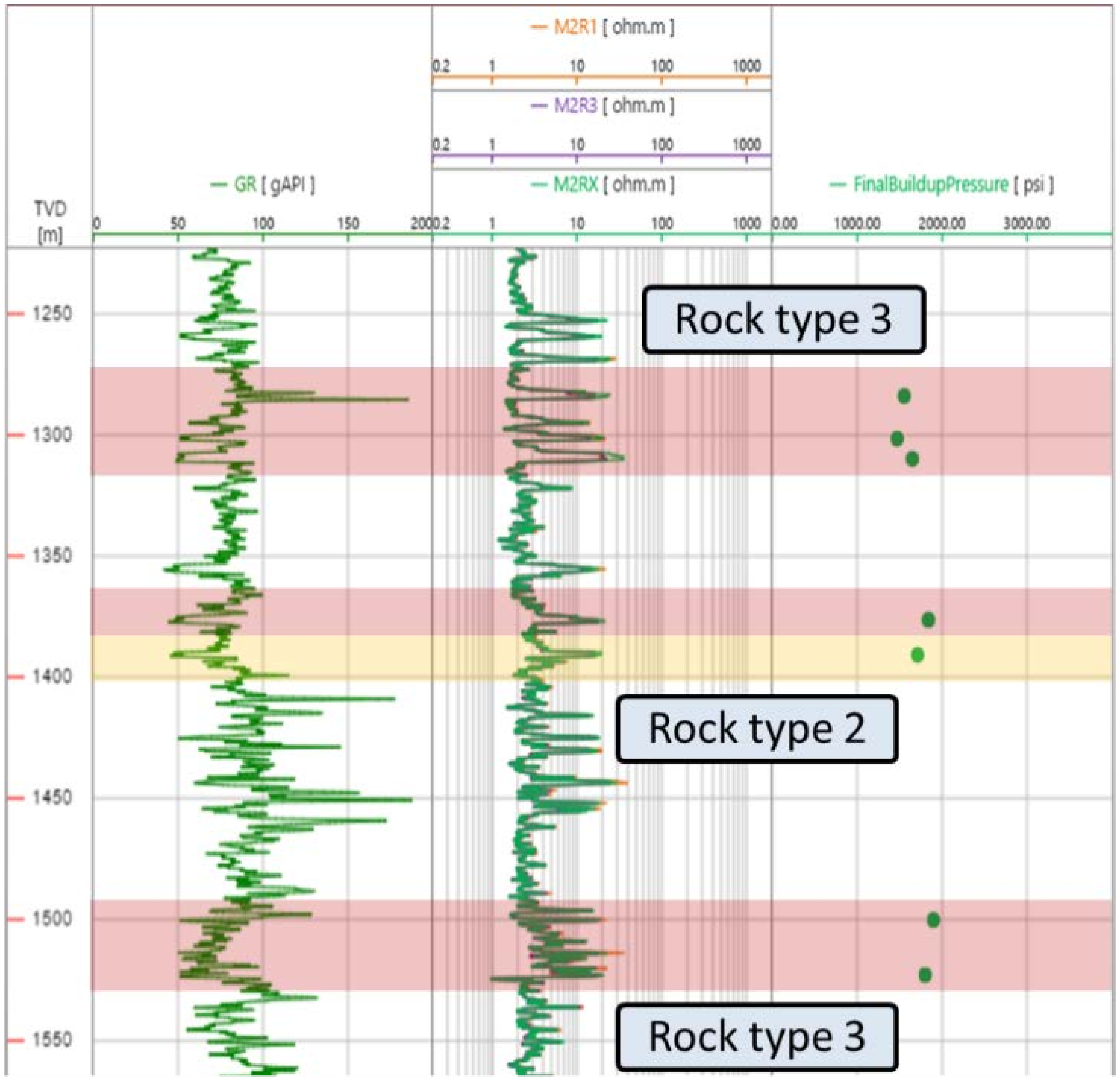
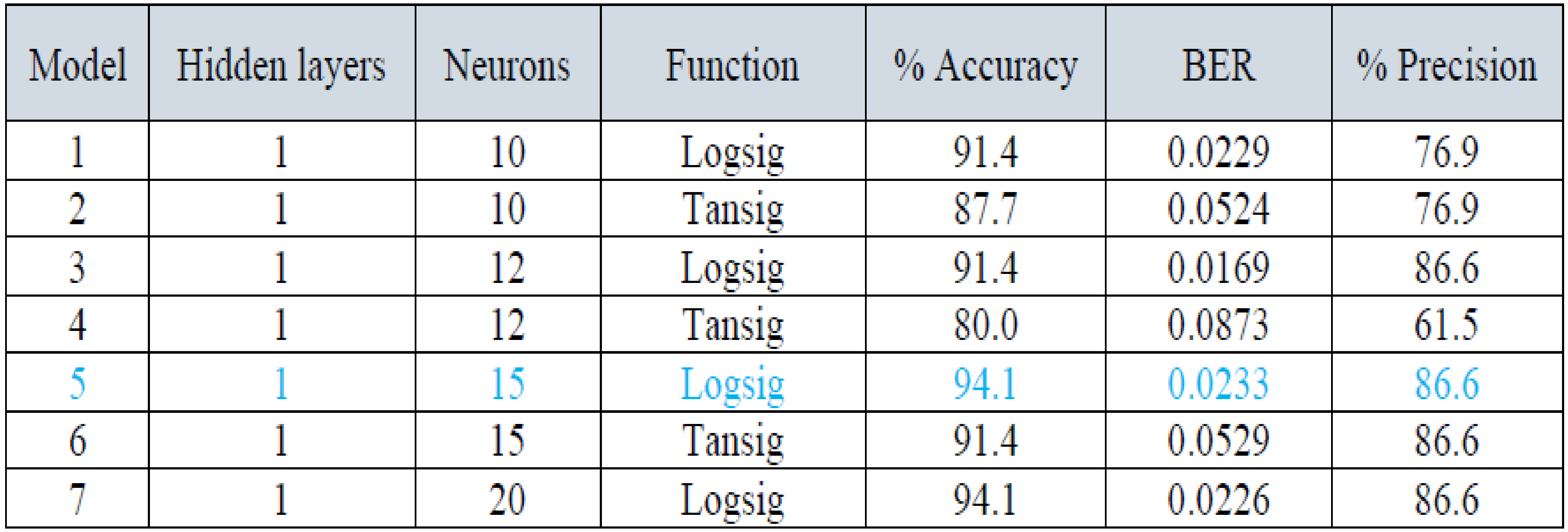
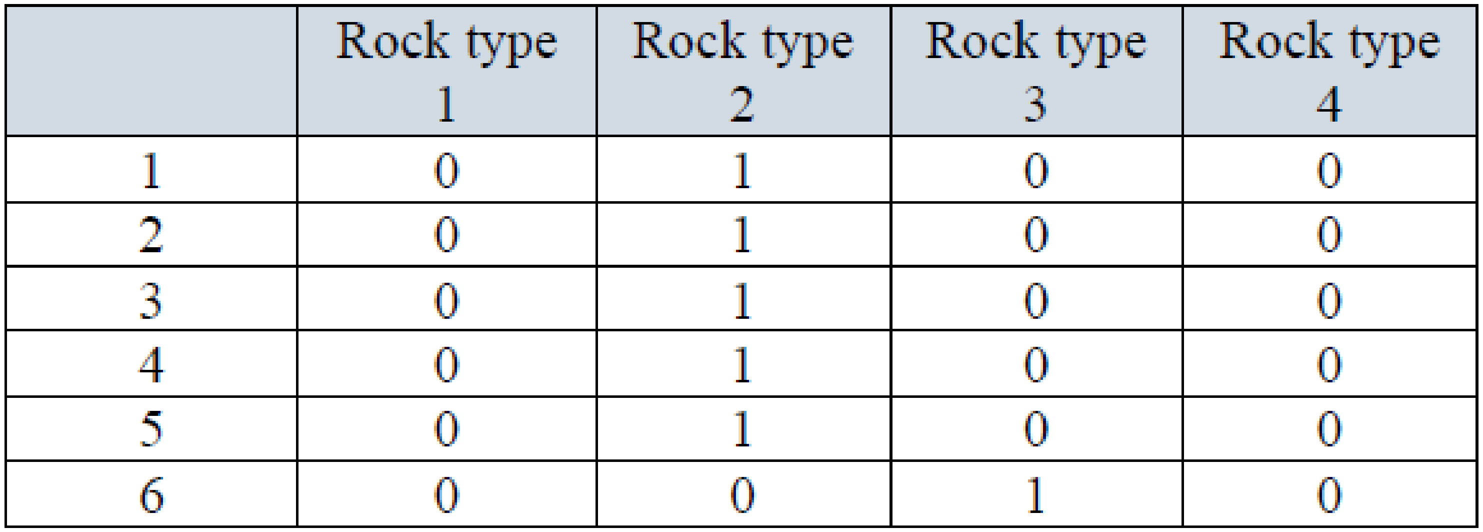


Figure 13. Pressure profile and the rock type classification in example 2.



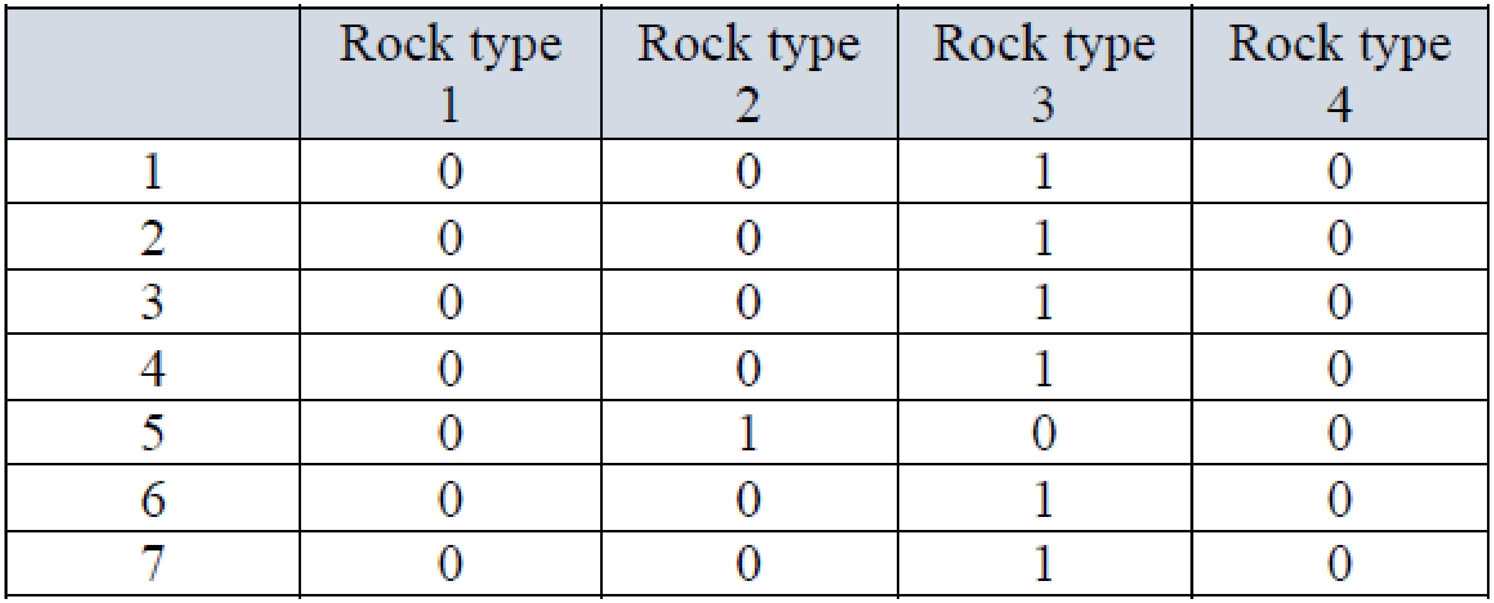
| Model | Hidden layers | Neurons | Function | % Accuracy | BER | % Precision |
|---|---|---|---|---|---|---|
| 1 | 1 | 10 | Logsig | 91.4 | 0.0229 | 76.9 |
| 2 | 1 | 10 | Tansig | 87.7 | 0.0524 | 76.9 |
| 3 | 1 | 12 | Logsig | 91.4 | 0.0169 | 86.6 |
| 4 | 1 | 12 | Tansig | 80.0 | 0.0873 | 61.5 |
| 5 | 1 | 15 | Logsig | 94.1 | 0.0233 | 86.6 |
| 6 | 1 | 15 | Tansig | 91.4 | 0.0529 | 86.6 |
| 7 | 1 | 20 | Logsig | 94.1 | 0.0226 | 86.6 |

Table 1. Comparison results for different ANN architectures.



| | Rock type 1 | Rock type 2 | Rock type 3 | Rock type 4 |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 0 |
| 6 | 0 | 0 | 1 | 0 |

Table 2. Rock type classification in well 1.



| | Rock type 1 | Rock type 2 | Rock type 3 | Rock type 4 |
|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 1 | 0 |
| 5 | 0 | 1 | 0 | 0 |
| 6 | 0 | 0 | 1 | 0 |
| 7 | 0 | 0 | 1 | 0 |

Table 3. Rock type classification in well 2.