PETROVIETNAM JOURNAL
Volume 6/2020, pp. 4 - 14
ISSN 2615-9902

APPLICATION OF DEEP LEARNING AND RANDOM FOREST ALGORITHMS IN A MACHINE LEARNING-BASED WELL LOG ANALYSIS FOR A SMALL DATA SET OF A SAND ZONE

Ruwantha Ratnayake[1], Pham Huy Giao[1,2]
[1]Asian Institute of Technology (AIT)
[2]Vietnam Petroleum Institute (VPI)
Email: hgiao@ait.asia/giaoph@vpi.pvn.vn

Summary

Artificial intelligence (AI) and machine learning (ML) have the potential to reshape the oil and gas exploration and production landscape. Once viewed as a promising novelty, AI and ML are not far away from becoming mainstream for all exploration and production companies. Earlier many researchers have worked on using intelligent analyses such as Artificial Neural Network (ANN), deep learning (DL), Fuzzy, Genetic Algorithm (GA) in well log interpretation, which are supposed to be effective for large data sets. Random forest (RF) algorithm so far has not been much applied for well log analysis. In this research, a code in Python language was developed for DL and RF analyses for well log interpretation. To highlight the advantages of the RF-based well log analysis we applied the new code for a small data set over a 50 m depth zone consisting of clay and sand zones.

Porosity, permeability and water saturation of the reservoir zone were predicted by the RF analysis, compared with those obtained by the DL analysis and validated with the core measurements. It was found that there is a significant improvement in the analysis running time and the accuracy of the RF-predicted well log answers compared to those results by DL analysis. It is therefore recommended that more applications of RF-based well log analysis be done for clastic reservoirs in Vietnam in the future.

Key words: Machine learning (ML), Python, random forest (RF), well log analysis, sand reservoir.

1. Introduction

RF is a classifier that evolves from decision trees. To classify a new instance, each decision tree provides a classification for input data; RF collects the classifications and chooses the most voted prediction as the result. The input of each tree is sampled data from the original dataset. In addition, a subset of features is randomly selected from the optional features to grow the tree at each node. Each tree is grown without pruning. Essentially, RF enables a large number of weak or weakly-correlated classifiers to form a strong classifier.

The RF algorithm is composed of different decision trees, each with the same nodes, but using different data that leads to different leaves. It merges the decisions of multiple decision trees in order to find an answer, which represents the average of all these decision trees.

Date of receipt: 22/11/2019. Date of review and editing: 22/11 - 24/12/2019.
Date of approval: 5/6/2020.

The RF algorithm is a supervised learning model, it uses labelled data to "learn" how to classify unlabelled data. The RF algorithm is used to solve both regression and classification problems, making it a diverse model that is widely used by engineers [1].

1.1. Earlier developments to RF

Ho proposed a method to overcome a fundamental limitation on the complexity of decision tree classifiers derived with traditional methods [2]. Such classifiers cannot grow to arbitrary complexity without sacrificing the generalisation accuracy on unseen data. The proposed method uses oblique decision trees which are convenient for optimising training set accuracy. The essence of the method is to build multiple trees in randomly selected subspaces of the feature space. The trees generalise their classification in complementary ways, and their combined classification can be monotonically improved.

Amit and Geman proposed a shape recognition approach based on the joint induction of shape features

and tree classifiers [3]. Because of virtually infinite number of features, they reached the conclusion that no classifier based on the full feature set could be evaluated as it was impossible to determine a priori whose features were informative. Due to the number and nature of features, standard decision tree construction based on a fixed length feature vector was not feasible. An alternative approach would be to entertain a small random of sample features at each node, constrain their complexity to increase with tree depth, and grow multiple trees. Terminal nodes contain estimates of the corresponding posterior distribution over shape classes. By sending the image down and aggregating the resulting distribution, the image can be classified.

In another paper by Ho [4], he proposed a method to solve the dilemma between overfitting and achieving maximum accuracy. This was done by constructing a decision-tree-based classifier that maintained the highest accuracy on training data and, at the same time, improved on generalisation accuracy as it grows in complexity. The classifier consisted of multiple trees constructed systematically by pseudo-randomly selecting subsets of components of the feature vector, that is, trees constructed in randomly chosen subspaces. When empirically tested against publicly available data sets, the subspace method proved its superiority when compared to single-tree classifiers and other forest construction methods. The next section introduces RF which is an ensemble method that combines existing techniques in order to construct a collection of decision trees with controlled variation.

1.2. RF algorithm

RF is an ensemble learning method used for classification and regression. Developed by Breiman [5], the method combines Breiman's bagging sampling approach [6] and the random selection of features, introduced independently by Ho [2, 4] and Amit and Geman. [3], in order to construct a collection of decision trees with controlled variation. Using bagging, each decision tree in the ensemble is constructed using a sample with replacement from the training data. Statistically, the sample is likely to have about 64% of instances appearing at least once in the sample. Instances in the sample are referred to as in-bag instances, and the remaining instances (about 36%) are referred to as out-of-bag instances. Each tree in the ensemble acts as a base classifier to determine the class label of an unlabelled instance. This is done via majority voting where each classifier casts one vote for its predicted class label, then the class label with the most votes is used to classify the instance.

Decision tree: Figure 1 shows a schematic decision tree that is a structure used in decision making process. This structure starts with a root node, which then branches to another decision node, repeating this process until a leaf is reached. A node asks a question in order to help classify the data. A branch represents the different possibilities that this node could lead to.

Some of the basic terminology related to decision trees are given below:

Root node: It represents entire population or sample and this further gets divided into two or more homogeneous sets.

Splitting: It is a process of dividing a node into two or more sub-nodes.

Decision node: When a sub-node splits into further sub-nodes, then it is called decision node.

Leaf node: Node which does not split is called leaf or terminal node.

Pruning: When a sub-node of a decision node is removed, this process is called pruning. It is the opposite process of splitting.

Branch/Sub-tree: A sub section of an entire tree is called branch or sub-tree.

Parent and Child node: A node, which is divided into sub-nodes is called parent node of sub-nodes whereas sub-nodes are the child of parent node.

Splitting decision trees: Breiman. [5] introduced additional randomness during the construction of

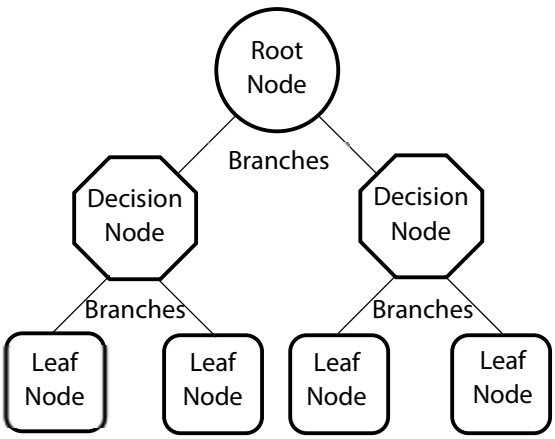


Figure 1. *Decision tree.*

decision trees using the classification and regression trees (CART) technique. Using this technique, the subset of features selected in each interior node is evaluated with the Gini index heuristics. The feature with the highest Gini index is chosen as the split feature in that node. Gini index has been introduced by Breiman et al. [7]. However, it has been first introduced by the Italian statistician Corrado Gini in 1912. The index is a function that is used to measure the impurity of data, i.e. the uncertainty of the data. In classification, this event would be the determination of the class label [8]. The general form of Gini index is shown below:

$$Gini = 1 - \sum_{i=1}^{c} (p_i)^2 \quad (1)$$

Where: Gini is the Gini index; p_i is the probability of an object being classified to a particular class; c is the number of unique labels.

Breiman [5] showed that the RF error rate depends on correlation and strength. Increasing the correlation between any two trees in the RF increases the forest error rate. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the RF error rate. Such findings seem to be consistent with a study made by Bernard et al. [9], which showed that the error rate statistically decreases by jointly maximising the strength and minimising the correlation.

1.3. Advantages of RF

Key advantages of RF are robustness to noise and overfitting [5, 10]. Overfitting generally occurs when a model is constructed in such a way that it fits the data more than it is warranted. A model which has been overfit will generally have poor predictive performance, as it does not generalise well. By generalisation we mean how well the model will make predictions for cases that are not in the training set. Hawkins pointed out that overfitting adds complexity to a model without any gain in performance or, even worse, leads to poorer performance [11]. A classifier that suffers from overfitting is likely to have a low error rate for the training instances (in-bag instances), and a higher error rate for the out-of-bag instances.

Other advantages of RF can be listed as follows:

- High versatility: Whether the task is regression or classification, RF is an applicable model for all the needs. It can handle binary features, categorical features, and numerical features. There is very little pre-processing that needs to be done. The data does not need to be rescaled or transformed.

- Parallelable: They are parallelisable, meaning that we can split the process to multiple machines to run. This results in faster computation time. Boosted models are sequential in contrast and would take longer to compute.

- Quick prediction/training speed: It is faster to train than decision trees because we are working only on a subset of features in this model, so we can easily work with hundreds of features. Prediction speed is significantly faster than training speed because we can save generated forests for future uses.

- Handles unbalanced data: RF methods for balancing error in class population unbalanced data sets. RF tries to minimise the overall error rate, so when we have an unbalanced data set, the larger class will get a low error rate while the smaller class will have a larger error rate.

- Low bias, moderate variance: Each decision tree has a high variance, but low bias. However, because we average all the trees in RF, we are averaging the variance as well so that we have a low bias and moderate variance model [1].

1.4. General applications of RF algorithm

There are several sectors where the RF can be applied as listed below:

Banking sector: The banking sector consists of most users. There are many loyal customers and also fraud customers. RF analysis can be used to determine whether the customer is a loyal or a fraud. A system uses a set of RF, which identifies the fraud transactions by a series of the pattern.

Medicines: Medicines needs a complex combination of specific chemicals. Thus, to identify the great combination in the medicines, RF can be used. With the help of machine learning algorithm, it has become easier to detect and predict the drug sensitivity of a medicine. Also, it helps to identify the patient's disease by analysing the patient's medical record.

Stock market: Machine learning also plays role in the stock market analysis. When it is needed to know the behaviour of the stock market, with the help of RF algorithm, the behaviour of the stock market can be analysed. Also, it can show the expected loss or profit which can be produced while purchasing a particular stock.

Applications of RF algorithm in oil & gas: In a research done by Chen, he successfully applied ML methods to predict well productivity and design hydraulic fracturing parameters in Montney and Duvernay Formations. He found out that ensemble models such as RF and ExBoost seem to outperform other types of ML methods (SVM, ANN) with a higher prediction accuracy [12].

1.5. Grid search method in ML

Grid search is the process of performing hyper parameter tuning in order to determine the optimum values for a given model. This is significant as the performance of the entire model is based on the hyper parameter values specified. 'GridSearchCV' in the sklearn library of Python is a method which calculates a score for different hyper parameter combinations based on accuracy (R^2 score), network building time and running time of the module. The combination which has the R^2 highest score is selected as the optimum combination. The R^2 score is calculated by Equations (2 - 4).

$$SST = \sum_{i=1}^{n} (y_i - \bar{y})^2 \quad (2)$$

$$SSR = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2 \quad (3)$$

$$R^2 = \frac{SSR}{SST} \quad (4)$$

Where: SST is the total variation in the data (sum of squared total), SSR is the sum of squares regression, yi is the y value for observation i, $\bar{y}$ is the mean of y values and $\hat{y}_i$ is the predicted values of y for observation I, and R^2 is the correlation coefficient.

1.6. Transfer functions

Transfer function is an algorithm process to transfer weighted sum to the hidden layers and the output layer. The transfer function is chosen to satisfy some specification of the problem that neural network is attempting to solve.

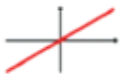
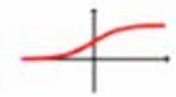
2. Methodology

The general workflow adopted in this study is explained in Figure 2.

2.1. Data collection and preparing the training input data

The published data set used in this study is taken from Darling [13] for a clastic reservoir located from 616 to 675 m deep. The well log data consist of gamma ray (GR), deep resistivity (LLD), sonic (DT), density (RHOB), and neutron porosity (NPHI). A part of the well data from 616 to 631 m was used as training data, while the part of well log data from 631 to 675 m was used for prediction. The target effective porosity (Φ_e) was calculated based on density (Φ_D) and neutron (Φ_N) porosity as shown in Equations (5) and (6) [14]:

Table 1. *Common transfer functions used in neural networks (ANN and DL analyses)*

Activation function	Equation	Derivative	1D graph
Linear	$f(z) = z$	$f'(z) = 1$	
Unit step (Heaviside function)	$f(z) = \begin{cases} 0, & z < 0 \\ 0.5, & z = 0 \\ 1, & z > 0 \end{cases}$	$f'(z) = 0$	
Sign (Signum)	$f(z) = \begin{cases} -1, & z < 0 \\ 0, & z = 0 \\ 1, & z > 0 \end{cases}$	$f'(z) = 0$	
Logistic (Sigmoid)	$f(z) = \frac{1}{1 + e^{-z}}$	$f'(z) = f(z)(1 - f(z))$	
Hyperbolic tangent (tanh)	$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$f'(z) = 1 - f(x)^2$	
ReLu	$f(z) = \begin{cases} 0, & z < 0 \\ z, & z > 0 \end{cases}$	$f'(z) = \begin{cases} 0, & z < 0 \\ 1, & z > 0 \end{cases}$	

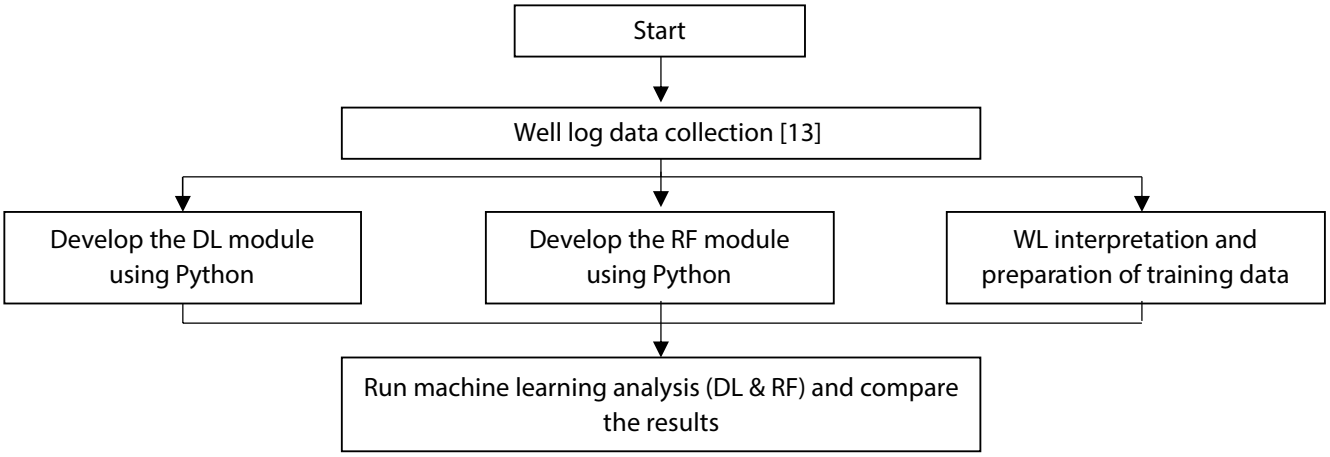


Figure 2. *Workflow of the study.*

Table 2. *Well log-calculated and core petrophysical parameters*

Depth (m)	Porosity		Permeability (mD)		Water saturation
	Core	Calculated	Core	Calculated	Calculated
620.116	0.02	0.03	0.01	0.11	1.0
622.097	0.02	0.022	0.02	0.05	1.0
624.078	0.1105	0.11	22	10.1	0.042
626.059	0.01	0.014	0.03	0.029	0.74
628.040	0.095	0.091	10.5	7.12	0.036
630.022	0.156	0.16	135.6	201.12	0.018
632.003	0.15	0.13	120	68.45	0.022
634.136	0.075	0.1	11	8.27	0.035
636.118	0.105	0.08	15.3	5.42	0.04
638.099	0.06	0.04	0.8	0.16	1.0
640.080	0.179	0.16	350	482.71	0.025
642.061	0.156	0.155	130	218.9	0.046

$$\Phi_D = \frac{\rho_m - \rho}{\rho_m - \rho_f} \quad (5)$$

$$\Phi_e = \frac{\Phi_N + \Phi_D}{2} \quad (6)$$

Where: ρ_m is the matrix density (g/cc) and is equal to 2.65 g/cc in this case for sandstone, ρ is bulk density (g/cc), ρ_f is fluid density (g/cc), Φ_D is density porosity, Φ_N is neutron porosity and Φ_e is effective porosity.

The water saturation of training data set was calculated using Simandoux (1963)'s method as shown in Equation (7). Simandoux equation was used because the zone of analysis includes shaly sand intervals.

$$S_w = \left(\frac{0.4 \times R_w \times a}{\Phi^m}\right) \times \sqrt{\left(\frac{V_{sh}}{R_{sh}}\right)^2 + \frac{5 \times \Phi^m}{a \times R_t \times R_w}} - \frac{V_{sh}}{R_{sh}} \quad (7)$$

Where: Φ is effective porosity, R_w is resistivity of water, V_{sh} is shale volume, R_t is formation resistivity, R_{sh} is resistivity of shale, a is an empirical constant and m is cementation exponent.

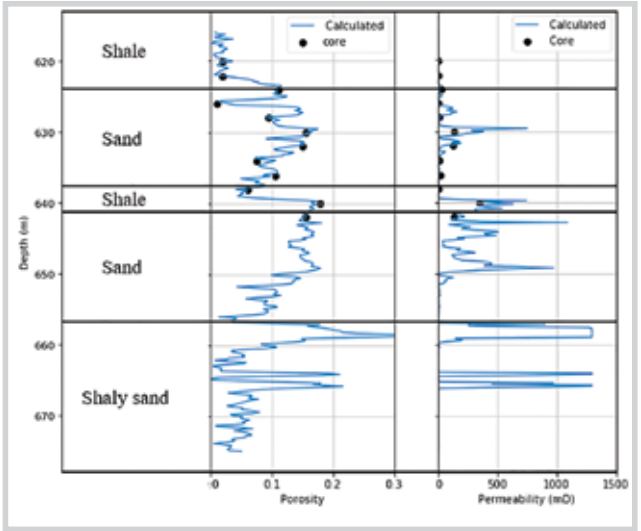


Figure 3. *Calculated petrophysical parameters vs core values.*

To determine permeability a poro-perm relationship was developed based on Equation (8) using the core data.

$$k = 10^{(k_a + k_b \times \Phi_e)} \quad (8)$$

Using core porosity and permeability values k_a and k_b were determined as -2 and 28.04 respectively. The core

measurements for this case study are represented in Table 2. The calculated petrophysical parameters as mentioned above are plotted versus the core values in Figure 3 that show a very good match.

2.2. Developing the DL module

DL could be usefully applied in well log analysis as indicated in a research by Giao and Sandunil [15]. Figure 4 shows the architecture of the DL network used in this study, which has three main layers, namely, input layer, hidden layer(s) and output layer. Input values of each hidden layer is multiplied by a certain weight and the summation is introduced to a transfer function assigned to each neuron. Training of an DL network is done using training examples. Grid search method which is an inbuilt function of sklearn library was used to find out the optimum hyper parameters for this data set. In this a total of 960 combinations were tested by varying the number of hidden layers from 1 to 50, neurons from 5 to 100, learning rate from 0.0001 to 0.1 and the transfer function being linear, unit step, sign, Sigmoid, tanh and Relu. Table 3 shows the best combination of hyper parameters which had the highest score that was used in DL code.

Table 3. *The best combination of hyper parameters*

Hyper parameter	Result
Gridsearch R^2 score	0.952
Number of hidden layers	50
Neurons per hidden layer	100
Learning rate	0.0001
Number of iterations	1000
Activation function	ReLu

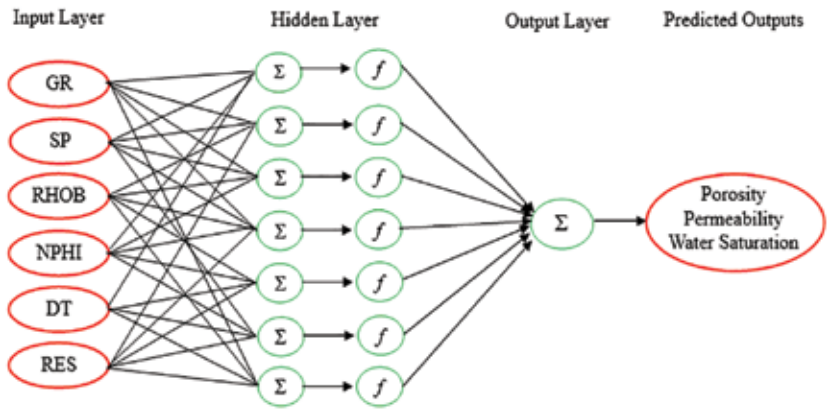


Figure 4. *Architecture of the DL network employed in this study.*

2.3. Developing the RF module

The RF module was developed with multiple number of decision trees and also coded in Python programming language as explained in Section 2.4. The flowchart of the Python code is shown in Figure 5. Normally in RF, the accuracy of the predicted results changes with the number of decision trees used. Therefore, in this case the trial and error method was used to find the optimum number of decision trees to get the most accurate results.

2.4. Coding and running the DL and RF modules in Python language

Python programming language was used in developing both the modules due to its ease to learn and the availability of vast amount of machine learning libraries. Number of standard libraries were used as shown in Table 4 in developing the codes.

An illustration of the Python codes for DL and RF analysis is shown in Tables 5 & 6. The Python package manager used in this study was Anaconda. Anaconda is a free and open-source distribution of the Python programming language for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. In order to create the code Jupyter Notebook was used [16], which is an open-source web application that allows to create and share documents that contain live code, equations, visualisations and narrative text.

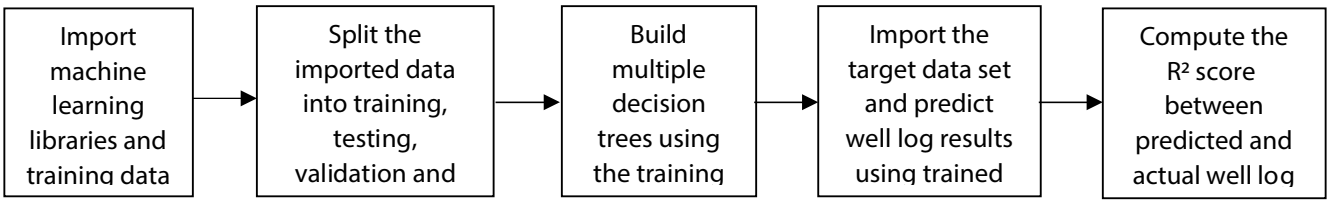


Figure 5. *Flowchart of the Python code created for RF module.*

Table 4. *Libraries used in the code*

Module	Library	Purpose
DL	Pandas	Import training and predicting data sets
	Matplotlib	Plotting the final interpretation plots
	Keras	Building the neural network with desired hyper parameters
	Sklearn	Splitting and scaling the training data, calculating the R^2 score
RF	Pandas	Import training and predicting data sets
	Matplotlib	Plotting the final interpretation plots
	Sklearn	Scale the training data and building random decision trees by using training data set

Table 5. *Structure of Python codes for DL module*

Task	Python code

Importing libraries	`import keras`
	`import sklearn`
	`import matplotlib`
	`import pandas as pd`

Building the neural network	`model = Sequential()`
	`number_of_hidden_layer=50`
	`number_of_neuron=100`
	
	`model.add(Dense(number_of_neuron,`
	`input_shape=(6,),`
	`activation='relu'))`
	
	`for i in range(number_of_hidden_layer):`
	`model.add(Dense(number_of_neuron,`
	`activation='relu'))`
	
	`model.add(Dense(1,))`

Training the network	`history = model.fit(X_train, y_train,`
	`epochs = 1000,`
	`validation_split = 0.1,`
	`verbose = 2)`

Testing and validating the network	

Running the module for predicting data set and saving the output	`y_test_pred = model.predict(X_test)`
	
	`result=pd.DataFrame(data=y_test_pred,`
	`index=df2.index,`
	`columns=['Porosity'])`

Table 6. *Structure of Python codes for RF module*

Task	Python code

Importing libraries	`import pandas as pd`
	`import skleran`
	`import matplotlib`

Splitting the data into training, testing and validation	`X_train, X_test,`
	`y_train, y_test = train_test_split(df1.iloc[:,0:6],`
	`df1.iloc[:,8:9],`
	`test_size=0.1,`
	`random_state=0)`

Creating decision trees	`from sklearn.ensemble import RandomForestRegressor`
	
	`regressor = RandomForestRegressor(n_estimators=6,`
	`random_state=0,`
	`verbose=2)`

Training phase	`regressor.fit(X_train, y_train)`
Testing, validating phase	`y_pred = regressor.predict(X_test)`

Running the module for predicting data set and saving the output	`y_test_pred = regressor.predict(X_test)`
	`df2['Pred_result'] = y_test_pred`
	`export_csv = df2.to_csv (r'Prediction_RF.csv')`

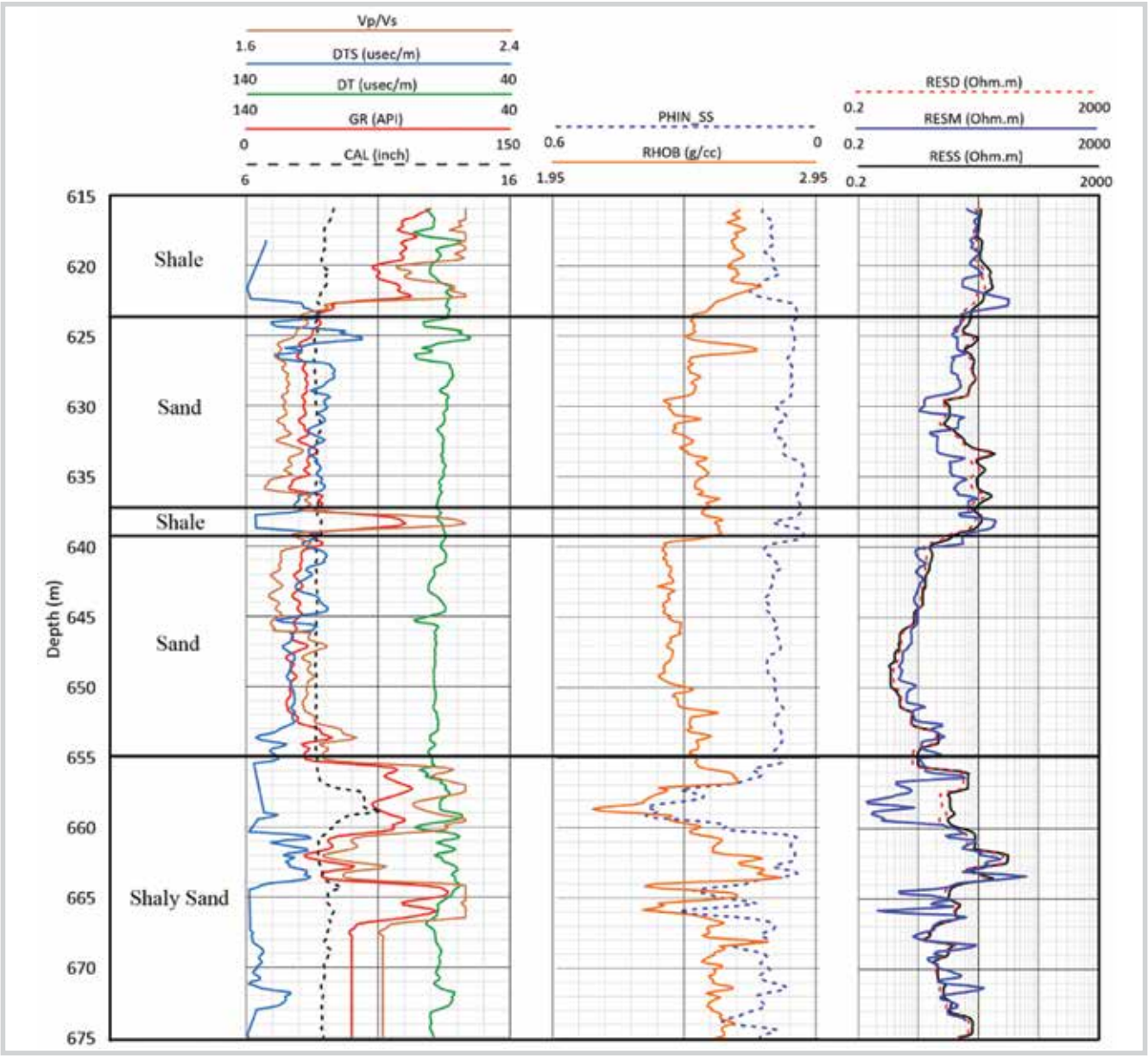


Figure 6. *Well log curves [13].*

Table 7. *Log responses and answers interpreted for the zoned layers at the study location [13]*

Zone	Lithology	Depth interval (m)	Log Responses				Log Answers				
			GR (API)	RHOB (g/cc)	PHIN	LLD (Ω.m)	Shale volume (V_{sh})	Density porosity (Φ_D)	Effective porosity (Φ_e)	Water Saturation (S_w)	Permeability (mD)
01	Shale	616-624	90	2.63	0.07	11	0.67	0.01	0.04	0.85	1.83
02	Sand	624-637	35	2.50	0.065	8	0.17	0.19	0.13	0.05	61.39
03	Shale	637-639	88	2.60	0.04	15	0.65	0.10	0.07	0.08	104.52
04	Sand	639-655	33	2.45	0.09	2	0.15	0.15	0.12	0.18	186.29
05	Shaly sand	655-668	74	2.55	0.12	15	0.52	0.04	0.08	0.65	215.5

3. Results and discussion

The collected log curves and data, taken from Darling [13], are shown in Figure 6 and Table 7.

Results from developed Python modules: Analyses for the public data set were done using both DL and RF modules developed, and the results are presented in Figures 7 and 8.

As shown in Table 8, the R^2 scores were calculated for each training, testing and predicting phase and compared

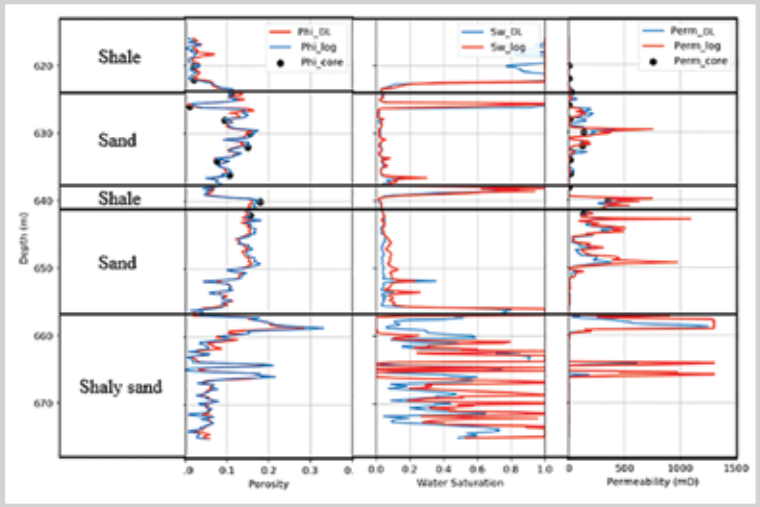


Figure 7. *Results for the DL module.*

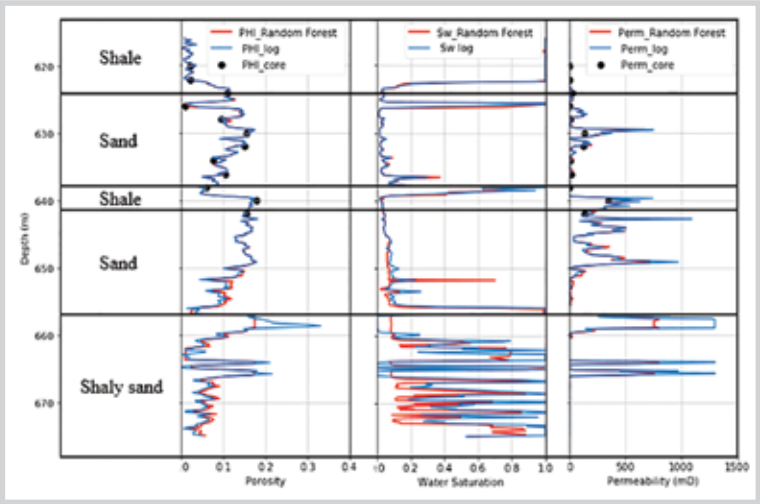


Figure 8. *Results from RF module.*

Table 8. *R^2 scores for DL and RF modules*

Well log answer	Data set	R^2 score		Running time (sec)	
		DL	RF	DL	RF
Porosity	Training	0.999	0.997	91	3
	Testing	0.989	0.984		
	Predicting	0.904	0.939		
Water Saturation	Training	0.954	0.989	98	5
	Testing	0.845	0.980		
	Predicting	0.754	0.894		
Permeability	Training	0.995	0.994	97	4
	Testing	0.932	0.975		
	Predicting	0.786	0.997		
Average		0.814	0.943	95.3	6

between two machine learning techniques that were used in this study, i.e., DL and RF.

It was also observed that the running time of the RF analysis is significantly lower compared to that taken by DL module to run as seen in Table 8 and Figure 9b.

4. Conclusions and recommendations

In this study, two machine learning-based analysis modules, i.e. DL and RF, were developed using Python programming language to perform well log analysis. These two modules were tested on a small size public data set of a clastic reservoir [9] and the accuracy of the results was compared. The following concluding remarks could be drawn:

- Based on a conventional well log interpretation on the study data set taken from Darling [13] a main sand reservoir zone from 639 to 655 m was identified with an average effective porosity (Φ_{D+N}) of 0.125, permeability (K) of 123.84 mD, and water saturation (S_w) of 0.18 that match well with the core measurements values, i.e. Φ_{core} = 0.13 and K_{core} = 149.24 mD.

- A number of DL analyses were conducted by varying the hyper parameters, i.e. number of hidden layer ranges from 1 to 50, number of neuron per hidden layer varies from 5 to 100, the learning rate varies from 0.0001 to 0.1, and the transfer function being linear, unit step, sign, Sigmoid, tanh and ReLu in both input and output layers. A total of 960 DL analyses have been run, out of which the best analysis was found to be the one having 50 hidden layers, 100 neurons per hidden layer, learning rate of 0.0001 and the ReLu transfer function that gave an average porosity of 0.124, permeability of 112.14 mD and water saturation of 0.14.

- Similarly, a number of RF analyses were run, varying the number of trees from 1 to 10, out of which the analysis with 6 trees was found the best RF analysis that gave an average porosity of 0.126, permeability of 122.15 mD and water saturation of 0.19.

- By comparing the results predicted by

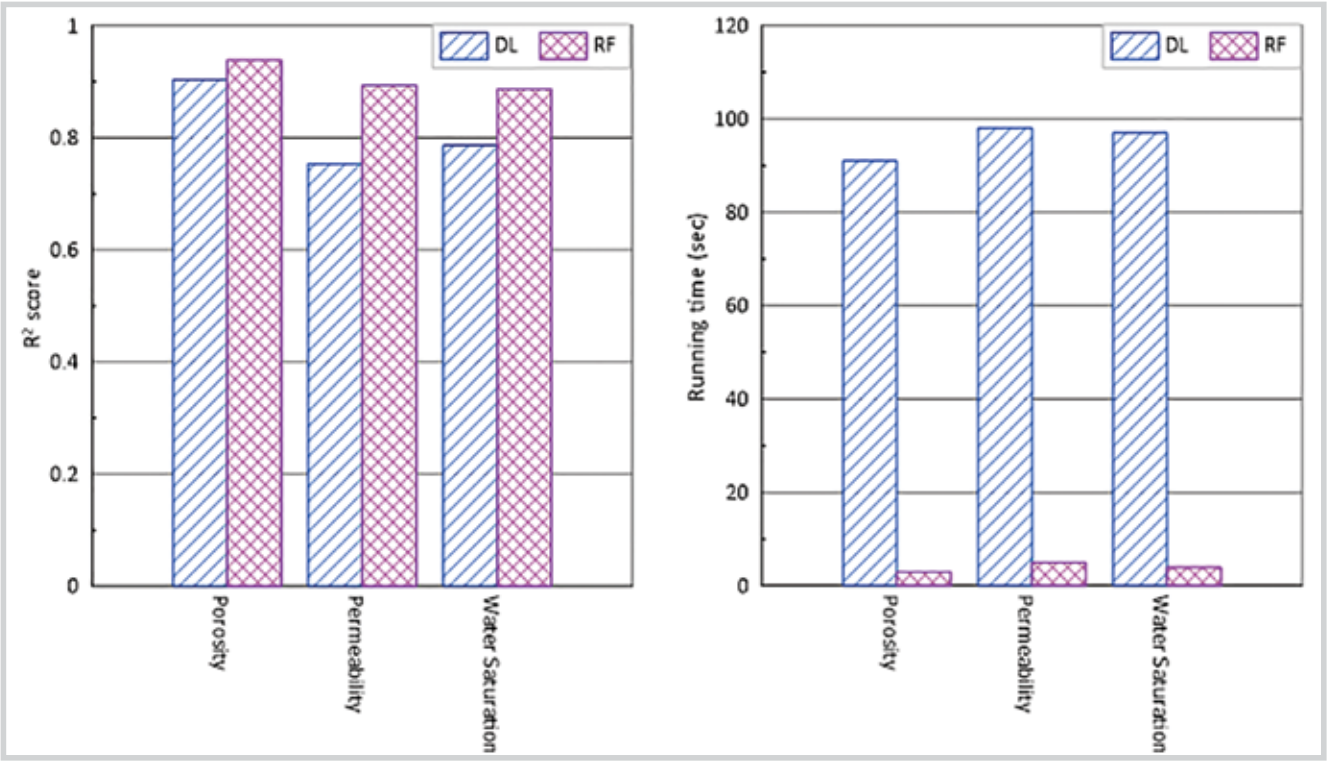


Figure 9. *Comparison of (a) R^2 score of the predicting phase from two modules, (b) running time of the modules.*

DL and RF analyses it was found that those by RF analyses are better than those predicted by DL analysis (Table 8 and Figure 9a), i.e. better R^2 and shorter running time. For example, the average running time is 6s for RF and 95.3s for DL, respectively.

- A notable advantage of RF analysis is that it could avoid the overfitting problem that is very common for an ANN or DL analysis. Overfitting can be detected when the R^2 value of the testing is significantly higher than that of predicting (Table 8).

- In term of code building, RF algorithm is easier to be developed in Python than ANN because the RF libraries are more diverse and a RF analysis requires less number of hyper parameters to be changed, i.e. only the number of the trees, while for an ANN or DL analysis more numbers of hyper parameters have to be tested, i.e. number of hidden layers, number of neurons per hidden layers, learning rate range, and type of activation or transfer function.

- Normally, machine learning-based analysis requires a big data set to be effective. However, in this study, the RF algorithm proved that it can be applied for a small data set, which would increase its applicability and can be recommended for more applications in well log analysis.

References

[1] Madision Schott, "Random Forest Algorithm for machine learning, Part 4 of a Series on Introductory Machine Learning Algorithms", 25/4/2019. [Online]. Available: http://medium.com/capital-one-tech/random-forest-algorithm-for-machine-learning-c4b2c8cc9feb.

[2] Tim Kam Ho, "Random decision forests", *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, 1995.

[3] Yali Amit and Donald Geman, "Shape quantization and recognition with randomized trees", *Neural Computation*, Vol. 9, No. 7, pp. 1545 - 1588, 1997.

[4] Tim Kam Ho, "The random subspace method for constructing decision forests", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 8, pp. 832 - 844, 1998.

[5] Leo Beriman, "Random Forests", *Machine Learning,* Vol. 45, pp. 5 - 32, 2001.

[6] Leo Breiman, "Bagging predictors", *Machine Learning*, Vol. 24, pp. 123 - 140, 1996.

[7] Leo Breiman, Jerome Friedman, R.A.Olshen, and Charles J.Stone, *Classification and regression trees*. Chapman & Hall/CRC, 1984.

[8] Mohamed Bader-El-Den and Mohamed Medhat Gaber, "GARF: Towards self-optimised random forests", *Proceedings of the 19th International Conference on Neural Information Processing*, pp. 506 - 515, 2012.

[9] Simon Bernard, Laurent Heutte, and Sébastien Adam, "A study of strength and correlation in random forests", *Proceedings of the 6th International Conference on Intelligent Computing*, pp. 186 - 191, 2010.

[10] Praveen Boinee, Alessandro De Angelis, and G.L.Foresti, Meta random forests, *International Journal of Computational Intelligence*, Vol. 2, No. 3, pp. 138 - 147, 2005.

[11] Douglas M.Hawkins, The problem of overfitting, *Journal of Chemical Information and Computer Sciences*, Vol. 44: pp. 1 - 12, 2004.

[12] Shengnan Chen, "Application of machine learning methods to predict well productivity in Montney and Duvernay", *Training course at SPE Canada Unconventional Resources Conference*, 17 March 2019.

[13] Toby Darling, *Well logging and formation evaluation*. Gulf Professional Publishing, 2005.

[14] Pham Huy Giao, *"Lecture notes of the CE71.70 course (Petrophysics)"*, Asian Institute of Technology, Bangkok, Thailand, 2018.

[15] Pham Huy Giao and Kushan Sandunil, *Applications of deep learning in predicting the fracture porosity*, Petrovietnam Journal, Vol. 10, pp. 14 - 22, 2017.

[16] Jupyter. [Online]. Available: https://jupyter.org.

[17] P.Simandoux, "Dielectric measurements in porous media and application to shaly formation", *Revue de L'Institut Français du Pétrole*, pp. 193 - 215, 1963.