**ANL488 FINAL PROJECT REPORT**

**Using Artificial Neural Network to predict Oil and Gas**

**Recovery Factor**



**Submitted by**

**Ho Kai Toong**

**SCHOOL OF BUSINESS Singapore University of Social Sciences**

**Presented to Singapore University of Social Sciences in partial fulfillment of the requirements for the**

**Degree of Bachelor of Science in Business Analytics**

**2023**

Abstract

Predicting oil & gas (O&G) recovery factor (RF) has always been challenging due to the complex nature of oil reservoirs. An accurate prediction allows O&G businesses to optimize their oil production and revenue. Therefore, an efficient predictive technique is vital to tackle the complexity of predicting the RF.

This project has adopted the use of artificial neural network (ANN) to predict the RF on two given datasets which include Tertiary Oil Recovery System (TORIS) and the Gulf of Mexico (GOM) datasets. The evaluation methods used in this study were mean absolute error (MAE) and coefficient of determinations (R²). This project utilized IBM SPSS Modeler and Python to construct a series of ANN models which include a single ANN model, ANN with bagging, ANN with boosting, and ANN with K-Fold.

This study has shown that using Python to construct the ANN models has helped to achieve optimized results. The single ANN and K-Fold ANN were the best performing models for TORIS and GOM respectively. For TORIS, a R² of 0.34 and MAE of 0.089 were obtained on the testing set whereas GOM produced better results of R² of 0.62 and MAE of 0.077 on the testing set. Despite the moderate performance of the models, it was observed that approximately 67% of the predictions in TORIS and 74% in GOM can still fall within a margin of error of ±10%.

The project also compared the optimized results of ANN and simpler machine learning (ML) models to evaluate which is the recommended model to predict the RF on the TORIS and GOM datasets. The results have indicated that less complex ML models, such as decision trees, have outperformed ANN. Therefore, for predicting the RF on these datasets, the use of simpler ML models is more suitable than employing ANN.

**Table of Contents**

[Abstract i](#_Toc150010067)

[Introduction 1](#_Toc150010068)

[Business Problem 1](#_Toc150010069)

[Application of ANN 2](#_Toc150010070)

[Project Objective 2](#_Toc150010071)

[Literature Review 3](#_Toc150010072)

[Data Understanding and Preparation 6](#_Toc150010073)

[TORIS 6](#_Toc150010074)

[GOM 9](#_Toc150010075)

[Modelling 11](#_Toc150010076)

[Test Design 11](#_Toc150010077)

[IBM SPSS Modeler Model Construction 12](#_Toc150010078)

[Single ANN 12](#_Toc150010079)

[ANN with Bagging, Boosting, and K-Fold 15](#_Toc150010080)

[Python Model Construction 17](#_Toc150010081)

[Single ANN 17](#_Toc150010082)

[ANN with Bagging 22](#_Toc150010083)

[ANN with Boosting 23](#_Toc150010084)

[ANN with K-Fold 24](#_Toc150010085)

[Model Assessment 25](#_Toc150010086)

[Evaluation 27](#_Toc150010087)

[Optimized ANN Model Results 27](#_Toc150010088)

[Final Evaluation 29](#_Toc150010089)

[Discussion & Recommendation 30](#_Toc150010090)

[Conclusion 32](#_Toc150010091)

[References 33](#_Toc150010092)

# Introduction

## Business Problem

The prediction of RF is a critical process in O&G sector. RF is typically measured by the proportion of O&G available for extraction over the total volume of hydrocarbons in a reservoir. It is one of the most significant parameters for O&G development as it directly reflects how much of the hydrocarbons can be economically recovered in a reservoir. In other words, the higher the RF, the higher the efficiency of oil production from the reservoir (Afari et al., 2015).

An accurate estimation of RF can also help O&G businesses to discover factors that lead to a higher rate of RF which can generate a higher return on investments. Therefore, by accurately predicting the RF, O&G companies can effectively strategize their resources to optimize oil production and revenue (Sachin & Karmakar, 2020).

However, due to the composite nature of oil reservoirs, predicting O&G RF can be challenging. The heterogeneity of oil reservoirs, reservoir pressure, and rock parameters such as porosity and permeability tend to create unpredictable trends on the O&G recovery (Ahmed et al., 2019). In the early days, the O&G industry encountered more difficulties in achieving accurate predictions of RF due to the lack of production data. As a result, operators in the O&G sector had to rely on their field experiences and various analogue studies to predict the RF (Kumar et al., 2022). Thus, an efficient predictive technique is essential to tackle the complexity of estimating O&G RF.

## Application of ANN

ANN is suitable in predicting O&G RF. It is known to discover complex, nonlinear, and multi-dimensional functional relationships between the inputs and output. Given the intricate nature of oil reservoirs, ANN is immensely appropriate for the prediction of O&G RF. Furthermore, predicting O&G RF is typically a regression task which is applicable to ANN as it can predict a continuous output (Chan et al., 2022).

Studies have also shown an increasing usage of AI to predict O&G RF (Li et al., 2020). Additionally, it is observed that predicting O&G RF with ANN has produced good predictive results which will be discussed in the literature review section.

## Project Objective

This project aims to assess the ability of ANN to predict the RF on the given datasets and this will be based on its performance on the testing set. There are two phases to this project. During the initial phase, a series of ANN models will be constructed using a data mining software known as IBM SPSS Modeler. In the second phase, the ANN models will be constructed using Python on Jupyter Notebook which is the main focus of this project. Therefore, the project will also study the difference in performance between the two platforms. As this project is also a complementary work to Kumar et al. (2022), the final objective for this project is to compare the optimized ANN results with their ML results and determine which model is better suited for predicting RF using the same set of data.

# Literature Review

A study adopted the use of predictive analytics to predict the ultimate recovery factor (URF) of oil reservoirs in Gulf of Mexico. The dataset consisted of 4,000 oil reservoirs with 82 attributes for each reservoir (Gowtham & Wu, 2017).

The predictive techniques adopted in the study consisted of multiple linear regression, robust linear regression, least absolute shrinkage and selection operator (LASSO), K-nearest neighbors, decision tree, random forest, ANN, and ensemble model. Root mean square error (RMSE) and MAE were used as evaluation methods on the testing set to assess the performance of the models.

Based on the modelling results, excluding the ensemble model, ANN emerged to be the champion model with the lowest RMSE and MAE of 8.5% and 6.0% respectively. Multiple linear regression was ranked as the second best model with RMSE and MAE of 9.0% and 7.0% respectively. And robust linear regression was the third best model with RMSE and MAE of 9.3% and 7.2% respectively. Decision Tree was the worst performing model with RMSE and MAE of 13.2% and 10.6% respectively.

It was observed that the champion model contained lesser errors compared to the other predictive models. The construction of the ANN model was optimized at 1 hidden layer and 3 nodes in it. The results also showed that 80% of the predictions on the testing set contained an error of no more than 8%.

A further study adopted the use of artificial intelligence (AI) to predict the oil recovery factor (ORF) for Water Drive Sandy Reservoirs. The dataset involved 130 water drive sandstone reservoirs and an additional 38 reservoirs of data were collected to test the predictive performance of the models (Ahmed et al., 2019).

Four AI models including ANN, radial basis neuron networks (RBFNN), adaptive neuro-fuzzy inference system with subtractive clustering (ANFIS-SC), and support vector machines (SVM) were used to predict the ORF for this study. The evaluation measures were R² and absolute average percentage error (AAPE).

Among the four AI models, the ANN model had the best performance on the testing data with the highest R² of 0.94 and lowest AAPE of 7.92%. ANFIS-SC was the second best performing model with R² of 0.91 and AAPE of 8.53%, followed by RBFNN with R² of 0.88 and AAPE of 8.78%, and then SVM with R² of 0.90 and AAPE of 10.44%.

The construction of the best performing ANN model contained one hidden layers with 5 neurons in it. It was also observed that the trainlm (Levenberg–Marquardt) function was adopted during the training process and tan-sigmoid was used as the activation function in the output layer. Subsequently, the trained model was applied to establish the empirical correlation and predict the ORF and hence achieving the best R² and AAPE results.

Another study utilized AI to predict the movable ORF of a layered reservoir as a result of waterflooding. A three-dimensional simulation model was built and contained 10 layers and a total of 64,000 cells (Shams et al., 2020).

Three AI models were used to perform the prediction of ORF which included ANN, Non-linear Regression (NLR), and Adaptive neuro-fuzzy inference system (ANFIS). The mean absolute percentage error (MAPE) and R² were used to evaluate and compare the predictive models’ performance.

Based on the models’ performance comparison, the ANN model was identified as the best model in performance which achieved MAPE of 7.08% and R² 0.997 on the testing set. The second best model was ANFIS which obtained MAPE and R² of 9.47% and 0.996 respectively. The worst performing model was NLR with MAPE and R² of 34.23% and 0.950 respectively.

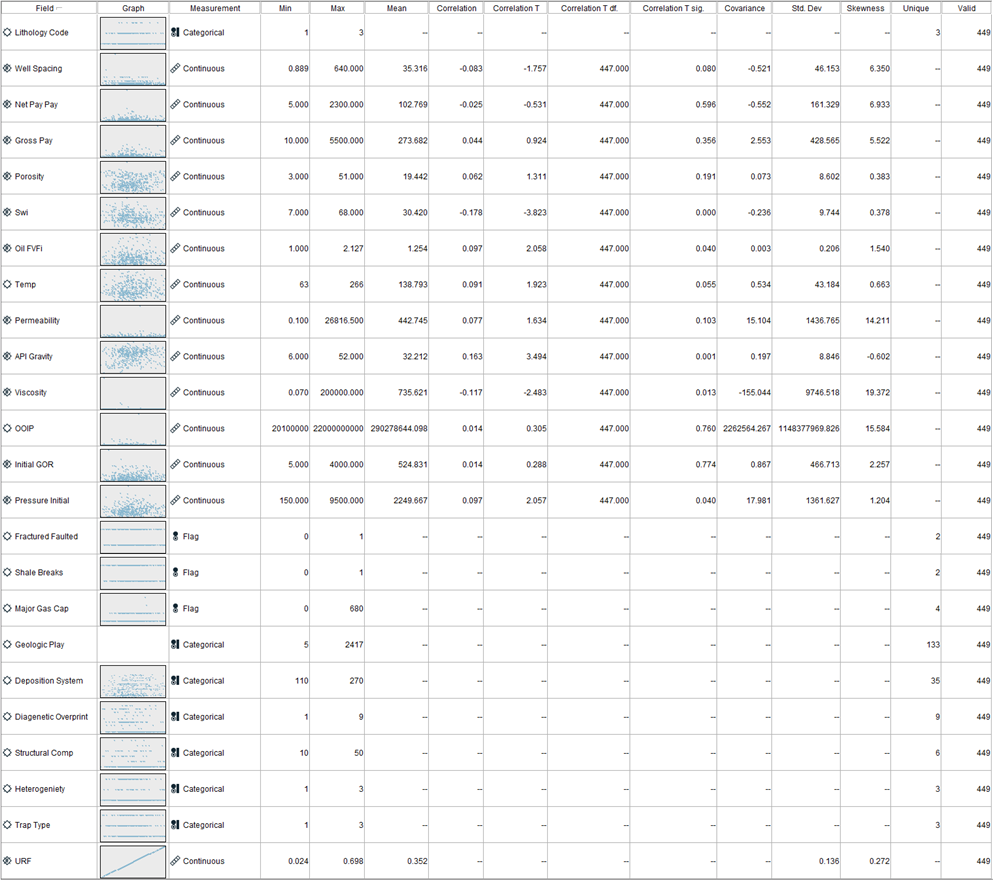
The construction of the ANN model was optimized at a hidden layer with 14 neurons. Similar to the previous study, the Levenberg–Marquardt function was used to train the model and tan-sigmoid was applied as the activation function in the output layer.

In addition, two further studies had adopted the use of ANN to predict O&G RF and produced decent predictive results (Afari et al., 2015; Surajudeen et al., 2022). Also, another study concluded that ANN is one of the most popular techniques to predict the O&G R&F (Sachin & Karmakar, 2020). Therefore, this project aims to leverage the capabilities of ANN to predict the O&G RF.

# Data Understanding and Preparation

The two datasets used in this study include the TORIS dataset (U.S Department of Energy, 1995) which contains 449 records with 24 variables and the GOM dataset (Bureau of Ocean Energy Management, 2019) which consists of 4,512 records with 15 variables.

## TORIS

****

**Figure 1. Statistics Table for Variables (TORIS)**

Referring to figure 1, there are 10 categorical and 13 continuous inputs used to predict a continuous output “URF”. It was observed that the continuous variables contained values in different ranges. For instance, “Temp” ranges from 63 to 266 and “Viscosity” ranges from 0.07 to 200,000. To prevent dominant variables from affecting the predictive model results, normalization was performed on the continuous inputs to ensure that they are in a similar range (Lee, 2021). Furthermore, label and one-hot encoding were performed on the categorical variables.

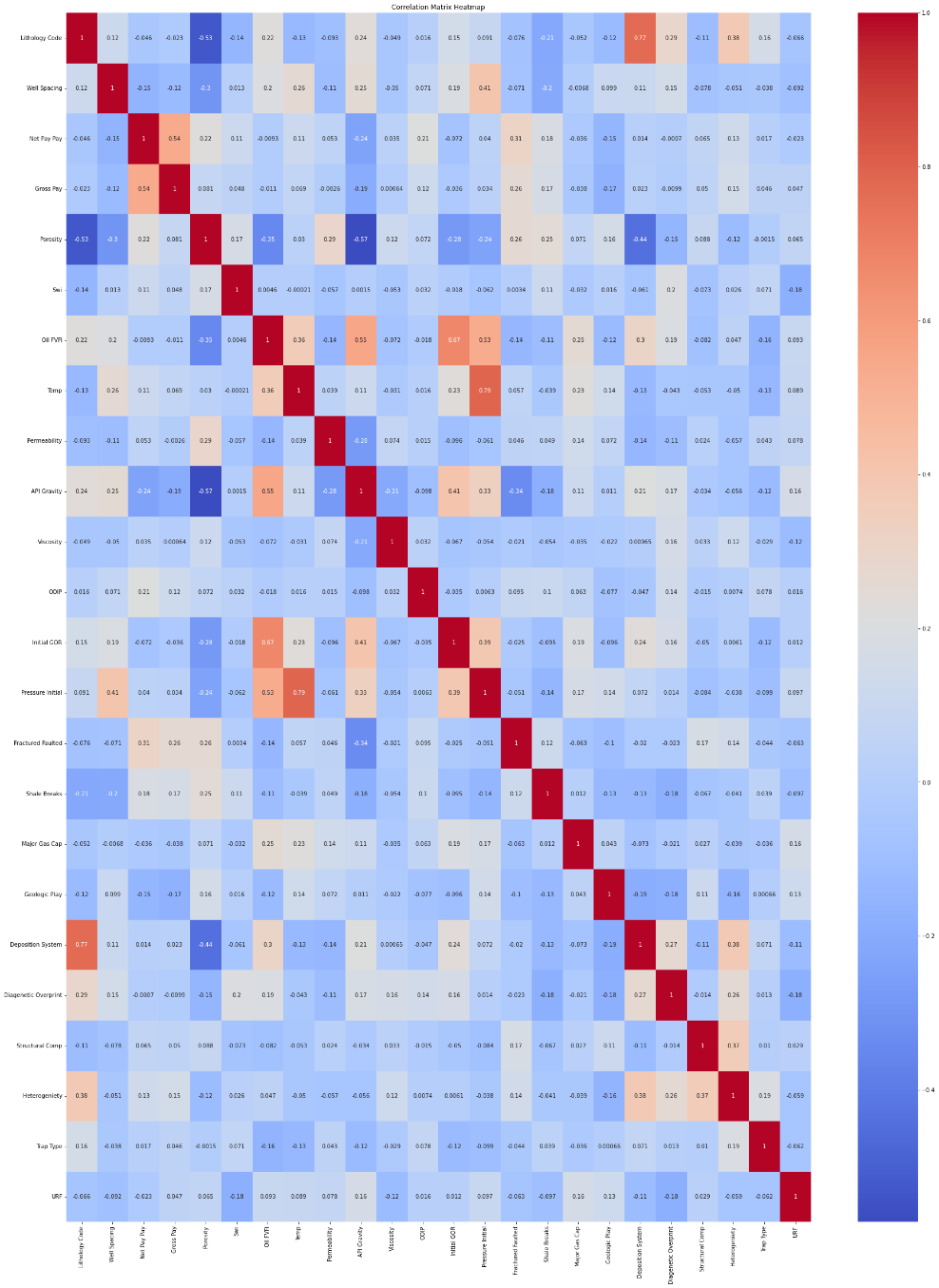
The TORIS dataset contained no missing values but it was observed that “Major Gas Cap” contained some unusual values.

**A graph with blue dots

Description automatically generated**

**Figure 2. URF vs Major Gas Cap Plot (TORIS)**

Based on figure 2, there are two odd values of 600 and 680 in the “Major Gas Cap” variable. Since it is supposed to be a flag variable where 0 = No Major Gas Cap and 1 = Has Major Gas Cap, the entire record where “Major Gas Cap” = 600 and 680 were removed instead of replacing them with assumed values.

****

**Figure 3. Correlation Matrix for Variables (TORIS)**

Multicollinearity refers to two or more variables that are highly correlated to one another. As ANN will be adopted in this study, a check for multicollinearity was also performed to ensure that the predictive performance is not disrupted (Chan et al., 2022). The Pearson Correlation was used where 0 indicates that the inputs are entirely not correlated and 1 implies that they are fully correlated. For this study, above 0.7 suggests a high correlation between the inputs.

Based on figure 3, “Lithology Code” and “Deposition System” contain a correlation of 0.77 and “Pressure Initial” and “Temp” contain a correlation of 0.79. Despite both pairs being highly correlated, all inputs will be used during the primary construction of the ANN model and one variable from each pair may be removed subsequently to improve the ANN results.

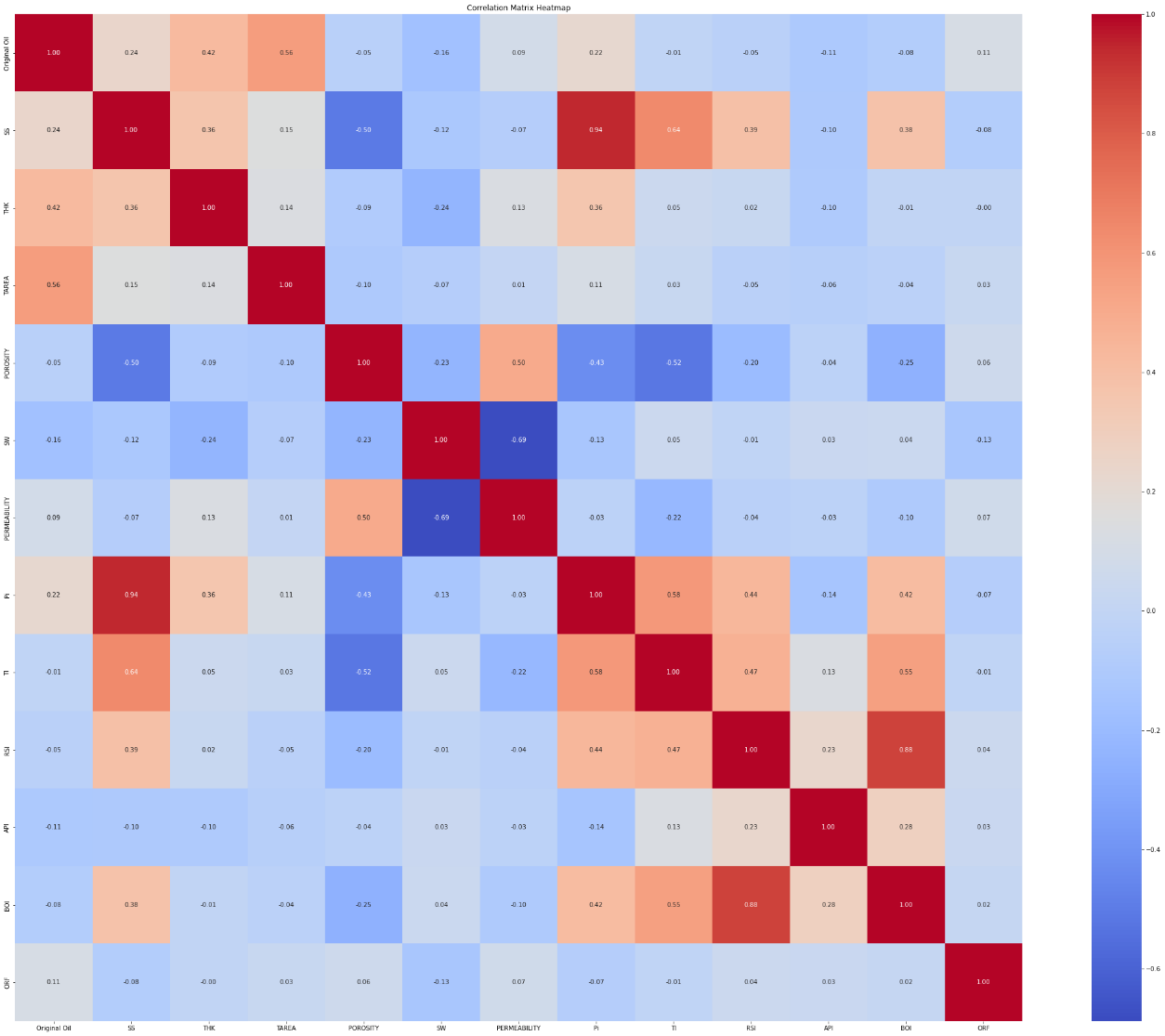
## GOM

A screenshot of a computer

Description automatically generated

**Figure 4. Statistics Table for Variables (GOM)**

For the GOM dataset, there are 2 categorical and 12 continuous inputs used to predict a continuous output “ORF” as shown in figure 4. There was no missing value noticed in this dataset. However, there was an extreme outlier of 2.32 in the target variable “ORF” and the whole record was removed since ORF should only range from 0 to 1. Similarly, normalization was performed on the continuous variables whereas label and one-hot encoding were conducted on the categorical inputs.



**Figure 5. Correlation Matrix for Variables (GOM)**

A correlation matrix was also performed on GOM. According to figure 5, two pairs are highly correlated with “Pi” and “SS” containing a correlation of 0.94 and “BOI” and “RSI” holding a correlation of 0.88. Similar to TORIS, all inputs will be used initially and hyperparameters will be tuned afterwards to improve the performance of the ANN models.

# Modelling

## Test Design

As predicting RF is a regression task, MAE and R² will be used as evaluation methods. MAE measures the average absolute difference between the predicted and actual values. Thus, the lower the MAE, the closer the prediction values are to the actual values. R² is known as the goodness of fit of the model which means the higher the R² , the better the model is able to fit the data (Chan et al., 2022).

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | Partition | | |
| Training Set | Validation Set | Testing Set |
| TORIS | 70 | 10 | 20 |
| GOM | 80 | 10 | 10 |

**Table 1. Partition Size for the Training, Validation, and Testing Sets for TORIS and GOM**

Table 1 displays the various partition size for the training, validation, and testing sets for TORIS and GOM. As TORIS is a relatively small dataset with only about 400 records, 20% of the data was used to represent the testing set. As for GOM, with a records of around 4,000 records, a smaller percentage of 10% was sufficient to represent the testing set.

A diagram of a diagram

Description automatically generatedA diagram of a data set

Description automatically generated with medium confidence

**Figure 6. Single Model, Bagging, Boosting (Left), and K-Fold Cross-Validation (Right) Illustration (Chan et al., 2022).**

Figure 6 displays an illustration of the four ANN models that will be constructed for this project. The first model is a single ANN model which is the simplest approach where a single ANN model is trained (Chan et al., 2022).

The second model is ANN with bagging also known as Bootstrap Aggregation which is an ensemble technique to improve the stability of the model. The initial stage of bagging involves bootstrapping where multiple samples of the original training set are created randomly which is suitable for a small dataset like TORIS. For a regression task, the final prediction will be the average predictions of all the base models.

The third model is ANN with boosting which is also an ensemble model that concentrates on improving the accuracy of the model. It trains multiple weak learners sequentially and the final prediction is calculated by combining all the weighed predictions of the weak learners.

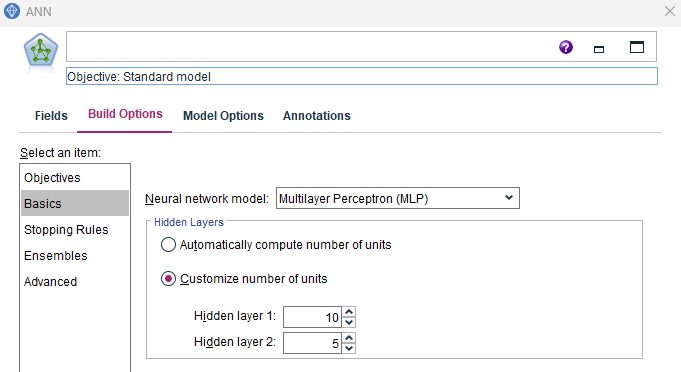
The last model is ANN with K-Fold cross-validation which maximizes data utilization. During training, the data is split into K number of folds where K-1 folds are used for training and the remaining one fold is used for validation. This maximizes all the data points as every fold takes turns to be trained and tested unlike the standard cross-validation where only a certain percentage of the dataset is used as the training or testing set. The final predictive results will be based on the average of the K iterations’ results.

## IBM SPSS Modeler Model Construction

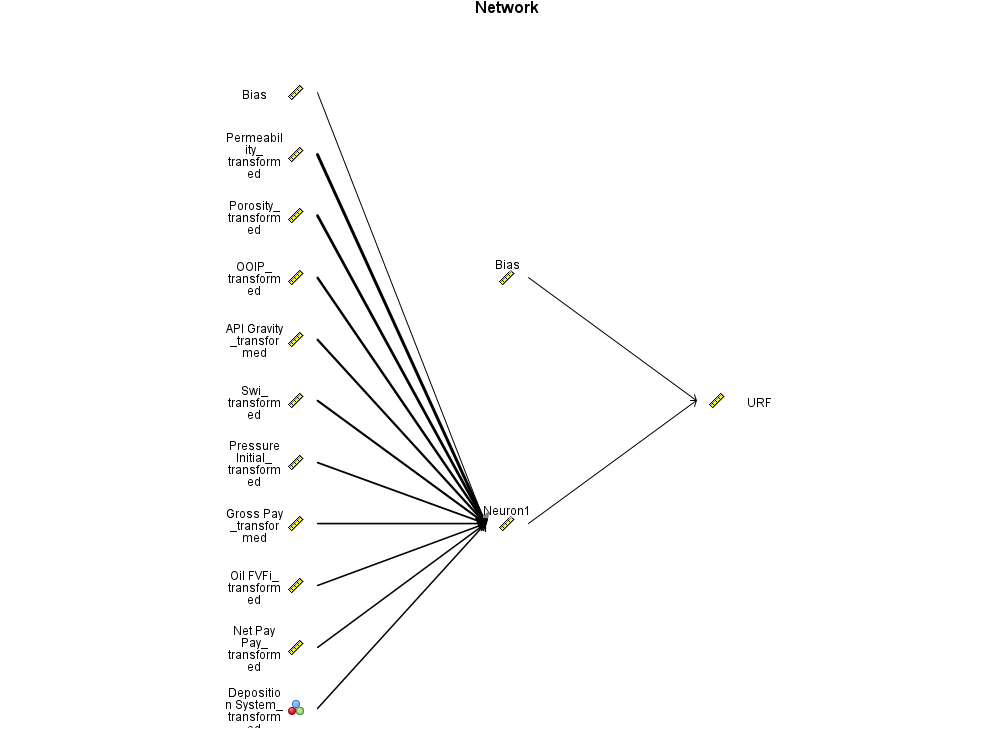
### Single ANN

The basic hyperparameters to tune an ANN model on IBM SPSS Modeler are typically the number of hidden layers and neurons. IBM SPSS Modeler also allows one to either customize or let the system automatically compute the number of hidden layers and neurons as shown in figure 7 below.

A screenshot of a computer

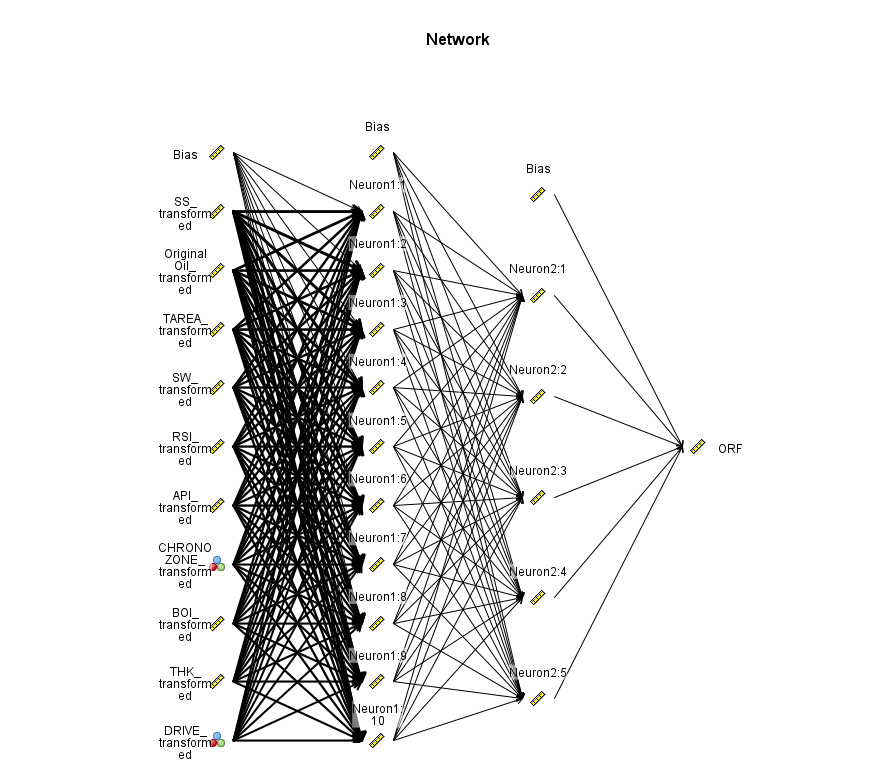
Description automatically generated  **Figure 7. Basic Parameter Settings for TORIS (Left) and GOM (Right)**

After several experiments, it was observed that the settings to achieve the optimized results for TORIS and GOM were to automate and customize the number of hidden layers and neurons respectively as shown in figure 7.



**Figure 8. Single ANN Model (TORIS)**

According to figure 8, TORIS is optimized at 1 hidden layer and 1 neuron by automatic computation. IBM SPSS Modeler uses only the top 10 most important inputs to train the model where “Permeability” is the most important factor located at the top of the network, followed by “Porosity”, “OOIP”, “API Gravity”, “Swi”, “Pressure Initial”, “Gross Pay”, “Oil FVFI”, “Net Pay Pay”, and lastly “Deposition System”.



**Figure 9. Single ANN Model (GOM)**

As shown in figure 9, GOM is optimized at 10 and 5 neurons in the first and second hidden layers respectively by manual selection in the settings. The top 10 variables include “SS’, “Original Oil”, “TAREA”, “SW”, “RSI”, “API”, “CHRONOZONE”, “BOI”, “THK”, and “DRIVE”.

With these, TORIS has obtained a training R² of 0.21 and training MAE of 0.094 where GOM has achieved a training R² of 0.19 and training MAE of 0.120. The same settings for TORIS and GOM were used to construct the ANN models with boosting, bagging, and K-Fold.

### ANN with Bagging, Boosting, and K-Fold

A white rectangle with black text

Description automatically generated

**Figure 10. Ensemble Model Settings**

A screenshot of a graph

Description automatically generated A screenshot of a graph

Description automatically generated

**Figure 11. ANN Model with Bagging for TORIS (Left) and GOM (Right)**

For ensemble models, the number of components models can be determined in the settings as shown in figure 10. For simplicity, 50 and 100 component models were constructed for TORIS and GOM respectively during the bagging process. Figure 11 displays the top 10 best performing individual base models with their overall training accuracy rates, the method (ANN) used, the number of variables (Predictors) used, number of synapses (Model Size) used, and number of data (Records) used for bagging. As for Boosting, 40 and 50 ANN base models were constructed for TORIS and GOM respectively.

A diagram of a diagram

Description automatically generated with medium confidence

**Figure 12. ANN with 5-Fold for TORIS and GOM**

As constructing K-Fold cross-validation on IBM SPSS Modeler involves heavy effort and time, only 5-Fold cross-validation was constructed for TORIS and GOM as shown in figure 12.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | TORIS | | GOM | |
| Training R² | Training MAE | Training R² | Training MAE |
| Single ANN | 0.21 | 0.094 | 0.19 | 0.120 |
| Bagged ANN | 0.81 | 0.050 | 0.44 | 0.108 |
| Boosted ANN | 0.96 | 0.013 | 0.36 | 0.106 |
| K-Fold ANN | 0.26 | 0.091 | 0.16 | 0.134 |

**Table 2. Training results on IBM SPSS Modeler for TORIS and GOM**

As observed from table 2, implementing ensemble techniques and K-Fold cross-validation generally helped to improve the training performance of a single ANN model.

For TORIS, it was seen that the bagged ANN, boosted ANN, and K-Fold ANN contained higher training R² and lower training MAE than the single ANN model which suggests that they can help to achieve better results during training. For GOM, only K-Fold ANN did not help to improve the single ANN model results. A different value of K-Fold may be able to improve its performance. However, as shown in figure 12, constructing a 5-Fold cross-validation already involves heavy manual work which can be time-consuming. Thus, experimenting with different number of K-Fold will be a tedious process. Furthermore, constructing ANN models through IBM SPSS Modeler is not the key focus of this project. Thus, 5-fold was used.

## Python Model Construction

### Single ANN

On Python, there are several more hyperparameters to tune an ANN model. Apart from the number of hidden layers and neurons, the following hyperparameters can also be adjusted:

1. Epoch (Number of training cycles)
2. Batch Size (Number of training samples utilized for one forward and backward pass during each epoch)
3. Learning Rate (The rate to control the ANN in adjusting its weights during training)
4. Optimizer (Algorithms to minimize the loss during training)
5. Activation Function (Functions to model the complex relationships during training)

For TORIS, a manual grid search approach was adopted to discover the optimized hyperparameters. Nested loops were used to process a specific set of hyperparameters grid to run all possible combinations. Several manual tuning experiments were conducted to narrow the grid selection as including more options in the hyperparameters grid would lead to more combinations and computational time to generate the results.

|  |  |
| --- | --- |
| Hyperparameters to Tune | Hyperparameters Grid |
| Inputs | All |
| Epochs | 9, 12, 15 |
| Batch Sizes | 5, 10, 15 |
| Activation functions in Hidden Layers | Relu, Tanh |
| Optimizers | Adam, Nadam, RMSprop |
| Learning Rates | 0.001, 0.002, 0.003 |
| Number of Hidden layers | 1, 2, 3 |
| Number of Neurons in Hidden Layer 1 | 3, 10 |
| Number of Neurons in Hidden Layer 2 | 2, 5 |
| Number of Neurons in Hidden Layer 3 | 2, 3 |

**Table 3. Finalized Hyperparameters Grid (TORIS)**

Referring to table 3, the final hyperparameters grid was determined and run which took a computational time of 44 minutes 33 seconds and generated 858 valid combinations. While it was computationally expensive to perform a grid search, it was useful to identify potential hyperparameters with the optimized results as shown in table 4 below.

|  |  |  |  |
| --- | --- | --- | --- |
| Hyperparameters | 1 Hidden-Layer ANN | 2 Hidden-Layer ANN | 3 Hidden-Layer ANN |
| Number of Neurons | 3 | 10, 5 | 10, 5, 3 |
| Learning Rate | 0.001 | 0.002 | 0.003 |
| Epoch | 9 | 9 | 12 |
| Batch Size | 5 | 15 | 5 |
| Activation Function | Relu | Tanh | Relu |
| Optimizer | Adam | Adam | Nadam |
| Training R² | 0.26 | 0.48 | 0.52 |
| Training MAE | 0.087 | 0.068 | 0.064 |

**Table 4. Optimized Hyperparameters for ANN with Hidden Layers of 1, 2, 3 (TORIS)**

Among the 858 combinations, the best-performing ANN models with 1, 2, and 3 hidden layers were identified in Table 4. It was observed that increasing the number of hidden layers and neurons led to higher training R² and lower training MAE, indicating that more hidden layers and neurons can improve results. It was also noticed that as the number of hidden layers and neurons increase, a higher learning rate is required for optimization.

Although it was observed that using more hidden layers and neurons can improve the training performance, it might not be the optimized solution. Figure 13 below illustrates the impact on the learning curves when a different number of hidden layer, neuron, and learning rate was adopted on the ANN model.

A graph on a white background

Description automatically generated A graph on a white background

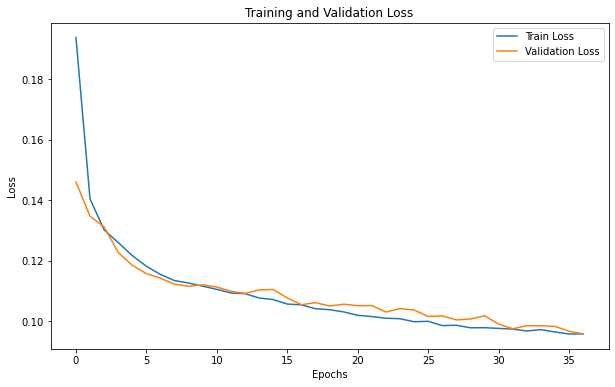
Description automatically generated

A graph on a white background

Description automatically generated

**Figure 13. Learning Curves of ANN Models (TORIS)**

According to figure 13, it was observed that as the number of hidden layers and neurons increase, there reached a point where the validation loss began to rise, while the training loss continued to decrease. This is a clear sign of overfitting. Additionally, the validation loss fluctuated even more when a higher learning rate was used which implies more instability of the model during the training process. The figure highlighted that the one-hidden layer ANN exhibited the best fit, featuring the smoothest and most stable learning curves compared to the other models. Therefore, the one-hidden layer ANN emerges as the optimal model for TORIS.



**Figure 14. Learning Curves of ANN Model (GOM)**

As GOM is about 10 times the size of TORIS, performing a grid search on GOM will be extremely time-consuming. Hence, different hyperparameters were experimented manually to identify the optimized hyperparameters for GOM. Figure 14 portrays a decent fit of the learning curves with the optimized hyperparameters used.

|  |  |  |
| --- | --- | --- |
| Hyperparameters | TORIS | GOM |
| Inputs Excluded | - | CHRONOZONE, BOI, Pi |
| Z-Score Threshold | - | 3 |
| Number of Hidden Layer | 1 | 2 |
| Number of Neurons | 3 | 7, 5 |
| Learning Rate | 0.001 | 0.001 |
| Epoch | 9 | 37 |
| Batch Size | 5 | 5 |
| Activation Function | Relu | Relu |
| Optimizer | Adam | Adam |

**Table 5. Optimized Hyperparameters for TORIS and GOM**

Referring to table 5 for TORIS, all inputs were used to obtain the optimized results. For GOM, “BOI” and “Pi” were removed as they have high correlations with “RSI” and “SS” respectively and they were also the less important variables. Also, it was observed that excluding “CHRONOZONE” produced better results which may be due to the variable containing many categorical values which affected the predictive results. Furthermore, the Z-score method was used on GOM to exclude certain outliers and achieve optimization in results (Mahmood, 2022).

As GOM is a larger dataset than TORIS, it requires more hidden layers, neurons, and epochs to capture the complexity and patterns among the variables. Thus, the optimized hyperparameters for both datasets are determined in table 5 which will be used to construct the remaining models.

### ANN with Bagging

|  |  |  |  |
| --- | --- | --- | --- |
| TORIS | Number of Bootstrap Samples | | |
| 50 | 100 | 200 |
| Computational Time (Seconds) | 114 | 218 | 405 |
| Training R² | 0.40 | 0.42 | 0.44 |
| Training MAE | 0.081 | 0.081 | 0.080 |

**Table 6. Number of Bootstrap Samples (TORIS)**

|  |  |  |  |
| --- | --- | --- | --- |
| GOM | Number of Bootstrap Samples | | |
| 100 | 200 | 500 |
| Computational Time (Seconds) | 2,971 | 5,892 | 17,314 |
| Training R² | 0.45 | 0.46 | 0.46 |
| Training MAE | 0.095 | 0.094 | 0.094 |

**Table 7. Number of Bootstrap Samples (GOM)**

For bagging, three different bootstrap samples were trained which include 50, 100, 200 for TORIS and 100, 200, 500 for GOM as shown in tables 6 and 7 respectively. The number of bootstrap samples typically means the number of component models used for bagging. Since GOM has more records than TORIS, more bootstrap samples was used to maximize the full potential of bagging. It was observed that increasing the number of bootstrap samples took up more computational time and had no significant impact on the training results for both datasets. Therefore, it is not worth using a higher amount of bootstrap samples. Hence, bootstrap samples of 50 and 100 were used for TORIS and GOM respectively.

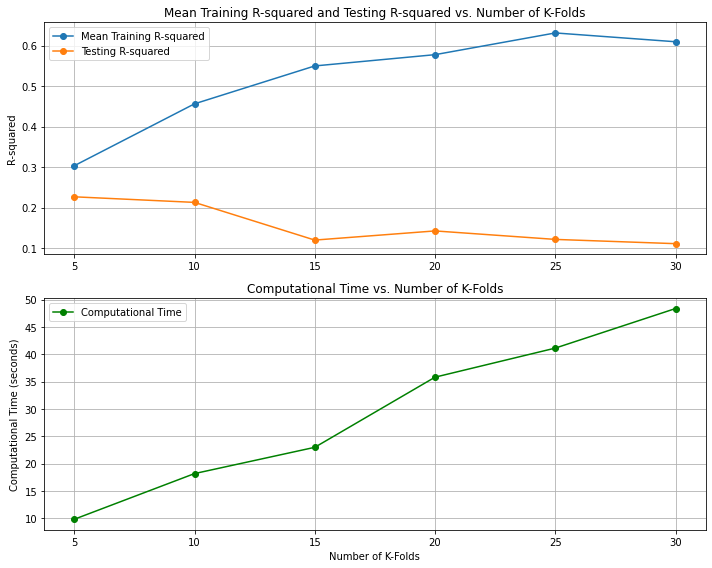
### ANN with Boosting

|  |  |  |
| --- | --- | --- |
| Hyperparameters | TORIS | GOM |
| Computational Time (Seconds) | 9 | 58 |
| Max Depth | 5 | 10 |
| Number of Base Estimators | 50 | 200 |
| Learning Rate | 0.01 | 0.05 |

**Table 8. AdaBoostRegressor Hyperparameters for TORIS and GOM**

During the initial stage of boosting, the ANN model served as a base model and was trained to make the predictions. Subsequently, adaptive boosting was applied on the predictions to enhance the performance of the ANN model. The adaptive boosting model used was decision trees which is a common choice in Python. The hyperparameters for the boosting model are specified on table 8. Due to differences in data size, GOM required a larger number of computational resources, including tree max depth, the number of base estimators, and learning rate, as indicated in the table, compared to TORIS.

### ANN with K-Fold

 A graph of two people

Description automatically generated with medium confidence

**Figure 15. K-Fold Cross-Validation Plot for TORIS (Left) and GOM (Right)**

To discover the optimal K-Fold value, various K-Fold values were tested. Specifically, for TORIS, 5-Fold, 10-Fold, 15-Fold, 20-Fold, 25-Fold, and 30-Fold were explored, as displayed in the left plot of Figure 15. An observation revealed that as the number of K-Fold increases, the mean training R² continues to rise, but the testing R² begins to decline. This was a sign of overfitting where the model performed well on the training set but not on the testing set. Furthermore, using more K-Fold required longer computational time. Thus, 5-Fold was the optimal model for TORIS.

Smaller K-Fold values were experimented with for GOM as longer computational time is needed due to its size. 5-Fold, 6-Fold, 7-Fold, 8-Fold, 9-Fold, and 10-Fold were tested as shown in the right plot of figure 15. It was observed that the model generally performed better on the testing set than the training set which showed no sign of overfitting. As the performances of 9-Fold and 10-Fold were comparable, 9-Fold is the optimal choice for GOM as it took lesser computational time.

## Model Assessment

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | TORIS | | | | GOM | | | |
| Training R² | Testing R² | Training MAE | Testing MAE | Training R² | Testing R² | Training MAE | Testing MAE |
| Single ANN | 0.21 | 0.09 | 0.094 | 0.102 | 0.19 | 0.16 | 0.120 | 0.125 |
| **Bagged ANN** | 0.81 | 0.07 | 0.050 | 0.101 | 0.44 | **0.46** | 0.108 | **0.111** |
| Boosted ANN | 0.96 | 0.04 | 0.013 | 0.101 | 0.36 | 0.36 | 0.106 | 0.108 |
| **K-Fold ANN** | 0.26 | **0.11** | 0.091 | **0.102** | 0.16 | 0.15 | 0.134 | 0.134 |

**Table 9. Optimized ANN Results on IBM SPSS Modeler**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | TORIS | | | | GOM | | | |
| Training R² | Testing R² | Training MAE | Testing MAE | Training R² | Testing R² | Training MAE | Testing MAE |
| **Single ANN** | 0.26 | **0.34** | 0.087 | **0.089** | 0.44 | 0.52 | 0.095 | 0.088 |
| Bagged ANN | 0.40 | 0.20 | 0.081 | 0.099 | 0.45 | 0.51 | 0.095 | 0.091 |
| Boosted ANN | 0.36 | 0.32 | 0.082 | 0.089 | 0.54 | 0.48 | 0.090 | 0.093 |
| **K-Fold ANN** | 0.30 | 0.23 | 0.086 | 0.100 | 0.47 | **0.62** | 0.091 | **0.077** |

**Table 10. Optimized ANN Results on Python**

After training all the models, predictions were made on the testing set, and their performance was evaluated on unseen data. Based on the results in table 9 for IBM SPSS Modeler, K-Fold ANN was the best performing model with the highest R² of 0.11 and a MAE of 0.102 on the testing set compared to the other models for TORIS. It was also observed that there were signs of overfitting, especially for bagging and boosting, where they performed very well on the training set but extremely poorly on the testing set. For GOM, there was no sign of overfitting and the best performing model was bagged ANN with the highest testing R² of 0.46 and a testing MAE of 0.111.

According to the results in Table 10 for Python, the optimized model for TORIS was the single ANN, achieving the highest testing R² of 0.34 and the lowest testing MAE of 0.089. In the case of GOM, the best-performing model was K-Fold ANN, which yielded the highest R² of 0.62 and the lowest MAE of 0.077 on the testing set. Therefore, the optimized ANN models for TORIS and GOM were the single ANN and K-Fold ANN respectively.

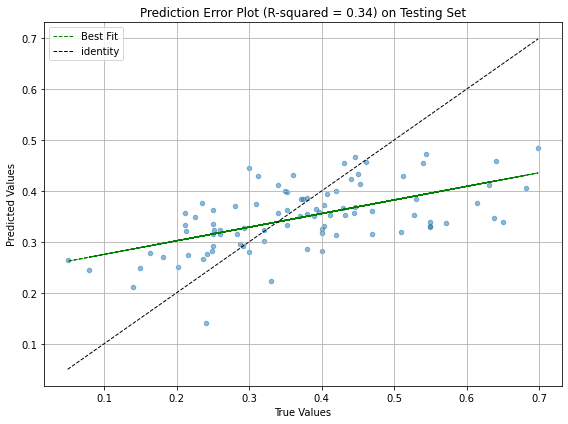
A common trend observed was that bagging, boosting, and K-Fold generally improved the training performance of the ANN model for both datasets in IBM SPSS Modeler and Python. However, it was noticed that TORIS did overfit when evaluated on the testing set, whereas GOM displayed more stable performance between the training and testing sets. This difference can likely be attributed to the dataset size, with TORIS containing only around 400 records, which is considerably smaller compared to GOM, which has roughly 4,000 records.

Another trend observed was that the models constructed on Python have outperformed the models constructed on IBM SPSS Modeler with higher R² and lower MAE on the testing set. A direct comparison of the single ANN model between IBM SPSS Modeler and Python demonstrates a significant increase in testing R², from 0.09 to 0.34 for TORIS and from 0.16 to 0.52 for GOM. This improvement is likely due to the additional hyperparameters available for tuning in Python, which enables the models to achieve better results.

# Evaluation

## Optimized ANN Model Results

A graph with blue dots

Description automatically generated 

**Figure 16. Residual vs Predicted Plot and Prediction Error Plot (TORIS- Single ANN)**

Two data visualizations were created each for TORIS and GOM based on the optimized ANN models’ results. For the residual vs predicted plot, the ideal scenario involves all data points clustering closely around the black horizontal line, signifying that the model can make predictions with a margin of error close to 0%. However, Figure 16 reveals that the data points are scattered, with only a few aligning with the black horizontal line. This is likely because the ANN model struggles to effectively capture the underlying data patterns, possibly due to some noisy data. It was also estimated that 67% of the predictions are within a margin of error of ±10%.

As for the prediction error plot, ideally the data points should circulate along the black reference line or the gap between the green best fit line and the black reference line should be as narrow as possible. Referring to figure 16, there is still a significant gap between the lines and with a 0.34 testing R² indicates that the ANN model can only explain 34% of the variability of the TORIS dataset. Therefore, the ANN model exhibits poor predictive performance on TORIS.

**A screen shot of a graph

Description automatically generated A graph with a line and dotted line

Description automatically generated**

**Figure 17. Residual vs Predicted Plot and Prediction Error Plot (GOM – K-Fold ANN)**

According to the residual vs predicted plot in figure 17, there are more data points gathered along the black horizontal line which suggests that GOM has a more accurate prediction compared to TORIS. Approximately 74% of the predictions fall within a margin of error of ±10%. The model appears to struggle when predicting ORF values between 0.1 and 0.3 compared to the range of 0.4 to 0.6. One potential reason for this is the relatively smaller number of ORF values available for training in the range of 0.1 to 0.3 compared to the range of 0.4 to 0.6.

According to the prediction error plot, the gap between the best fit line and reference line is much narrower for GOM compared to TORIS which implies that GOM has a better predictive performance than TORIS. With 0.62 R² on the testing set suggests that the ANN model can explain 62% of the variability of the GOM dataset. Thus, the ANN model contains a moderate predictive performance on GOM.

## Final Evaluation

|  |  |  |
| --- | --- | --- |
| Dataset | R² on Testing Set | |
| ANN | Kumar et al. (2022) |
| TORIS | 0.34 | 0.81 |
| GOM | 0.62 | 0.88 |

**Table 11. Optimized ANN vs ML Models in Kumar et al. (2022)**

The final evaluation is to evaluate the performance on the testing set between the optimized ANN and the ML models used in Kumar et al. (2022). For TORIS, it was observed that the optimized ANN model with a testing R² of 0.34 was lower than 0.81 which was the average result of four models consisting of decision trees with Bagged K Neighbours (KNN), Blended 5-Fold, Random Forest (RFR), and Bagged Category Boosting (CatBoost). Similarly, the optimized ANN model used on GOM contained a lower performance of 0.62 R² on the testing set compared to 0.88 which was the average result of four models containing Decision Trees with Catboost, Bagged Catboost, Stacked 10-Fold, and RFR. Therefore, the ML models used in Kumar et al. (2022) produced a better performance than ANN in predicting RF.

Based on table 11, it was observed that both models performed better on GOM than TORIS. Furthermore, there was a drastic difference on the testing R² between TORIS and GOM when ANN was used for the prediction. In contrast, the testing R² results were more comparable when the decision tree models in Kumar et al. (2022) were utilized. One potential reason is that ANN is extremely sensitive to the size of the dataset. As TORIS is a relatively small dataset, it may not be sufficient for the ANN model to capture the complexity of the data during training compared to GOM. On the other hand, when the data is limited, decision trees are less sensitive than ANN because of their robustness to noisy data (Chan et al., 2022). Hence, deep learning networks like ANN may not be suitable to predict the RF on these datasets whereas simpler models like decision trees are more appropriate.

# Discussion & Recommendation

Building the ANN models using IBM SPSS Modeler and Python each has its own advantages and disadvantages. IBM SPSS Modeler provides a friendlier user experience than Python. Constructing the ANN models on IBM SPSS Modeler simply involved connecting the nodes together without coding them. Moreover, it was easier and faster to create data visualizations on IBM SPSS Modeler than Python. On Python, the ANN models, figure title, plot size, plot colour, and font size had to be manually coded to construct them. However, IBM SPSS Modeler also has its drawbacks. For instance, it was unknown what activation functions were used on the ANN models which serves as a black-box approach. It was also difficult to track and document the computational time to run the models. In addition, constructing a model with specific needs can be time-consuming. For example, constructing a 10-fold or 15-fold model will be extremely tedious on IBM SPSS Modeler.

On the other hand, Python offers several ML libraries to fulfil any specific requirement. For instance, TensorFlow and Keras are popular modules to build deep learning models such as ANN. Constructing a K-Fold ANN model was easier on Python than IBM SPSS Modeler as it only involved importing the K-Fold module and different K-Fold values can be explored by simply changing the value in the codes and letting it run automatically. However, one disadvantage is that running the ANN models on Python generally took up more computational time compared to IBM SPSS Modeler.

In terms of model performance, Python did help to improve the ANN results significantly. The reason being is that the tuning of hyperparameters were limited on IBM SPSS Modeler. The main hyperparameters to tune an ANN model was simply just the number of hidden layers, neurons and training cycles. Furthermore, only a maximum number of 2 hidden layers was allowed which means constructing an ANN model with three hidden layers and above is not possible on IBM SPSS Modeler. Conversely, Python allows other more hyperparameters to tune such as the number of hidden layers and neurons, epoch, batch size, learning rate, activation function, and optimizer which contributed to a better predictive performance.

Interestingly, the results of the ANN models were not as good as what the literature reviews claim to be. The results showed a contrasting view where the optimized ANN model obtained a poor to moderate performance on the datasets. However, it is essential to note that the performance of ANN is subjected to various factors such as the variables, dataset size, platform, and model architecture used to train the model. In other words, when a different dataset, platform, and model architecture is used, it can produce different results.

ANN is also prone to overfitting especially when the dataset is small or when the model is complex. An example of TORIS, when the model became more complicated, the learning curves started to show signs of overfitting and instability. ANN typically needs a huge amount of training data to generalize well. With limited amount of data, an ANN model may memorize the patterns too well during training instead of learning the underlying patterns which can lead to overfitting. Thus, one disadvantage of ANN is that it is extremely sensitive to the size of the dataset and model complexity (Chan et al., 2022).

For future work, it may be worthwhile to explore using support vector machines (SVM) to predict O&G RF. The reason being is that SVM is less sensitive to the size of the dataset and noisy data which may generate better results compared to ANN.

# Conclusion

In conclusion, this study has demonstrated the capabilities of using ANN to predict O&G RF. While the ANN results may not achieve the highest level of accuracy, the majority of the RF can still be predicted by ANN within an error margin of ±10%. Furthermore, this study has highlighted the distinct experiences of adopting IBM SPSS Modeler and Python to construct the ANN models and how Python has enhanced the performance of the models. For the TORIS and GOM datasets, it can be concluded that adopting simpler models, such as decision trees, is more appropriate for predicting RF compared to using ANN.

(Word Count: 5,986)

# References

Afari, S. A., Kwame, S., Wilberforce, N. A., & Anthony, M. (2015). Assessment of petroleum reservoir recovery factor using complexity scoring and artificial neural network. *International Journal of Scientific Engineering and Research, 3*(7), 124-129.

Ahmed, A. A., Salaheldin, E., Chen, W., & Abdulazeez, A. (2019). Estimation of oil recovery factor for water drive sandy reservoirs through applications of artificial intelligence. *Energies*.

Bureau of Ocean Energy Management. (2019). *Atlas of Gulf of Mexico gas and oil sands data*. U.S Department of the Interior. https://www.data.boem.gov/Main/GandG.aspx

Chan, S. P., Tan, S. C., Wang, D., & Ren, J. (2022). ANL307 Predictive modelling (study

guide). *Singapore: Singapore University of Social Sciences*.

Gowtham, T., & Wu, X. (2017). Using data analytics on dimensionless numbers to predict the ultimate recovery factors for different drive mechanisms of Gulf of Mexico oil fields.

Kumar, M., Swaminathan, K., Rusli, A., & Thomas-Hy, A. (2022). Applying data analytics & machine learning methods for recovery factor prediction and uncertainty modelling.

Lee, K. H. C. (2021). ANL303 Fundamentals of data mining (study guide). *Singapore: Singapore University of Social Sciences*.

Li, H., Cao, N., Tian, H., & Cheng, S. (2020). Applications of artificial intelligence in oil and gas development. *Archives of Computational Methods in Engineering, 28*, 937-949

Mahmood, S. (2022, June 10). *Outlier detection (part 1)*. Medium. https://towardsdatascience.com/outlier-detection-part1-821d714524c#:~:text=For%20example%2C%20a%20z%2Dscore,similar%20to%20standard%20deviation%20method.

Sachin, C., & Karmakar, G. P. (2020). Artificial intelligence techniques and their application in oil and gas industry. *Artificial Intelligence Review*.

Shams, K., Sidqi, A. A., Hasan, Y. A., & Rahul, G. (2020). A novel empirical correlation for waterflooding performance prediction in stratified reservoirs using artificial intelligence. *Neural Computing and Applications.*

Surajudeen, S., Hassan, S., Afza, S., Raji, I. O., & Yarima, M. H. (2022). Prediction of electromagnetic properties using artificial neural networks for oil recovery factors. *Colloid Journal, 85*(1), 151-165

U.S Department of Energy. (1995). *Tertiary oil recovery information system.* U.S Department of Energy. https://edx.netl.doe.gov/dataset/2006-oil-and-gas-industry-software