PROFESSIONAL TECHNICAL PAPER
ONLINE PRESENTATION
24 - 25 OCTOBER 2020
IATMI

A Digital Oilfield Comprehensive Study: Automated Intelligent Production Network Optimization

Aulia Ahmad Naufal[*1], Sabrina Metra[2]

[1]Production Engineer, Schlumberger
[2]Production Engineer, Schlumberger
* Email: anaufal@slb.com

Abstract. Production optimization on a network level has been proven to be an effective method to maximize production potential of a field with low capital. But as it stands, it is a heavy process to start along with its several challenges such as data quality issues, tedious plus repetitive work processes to deploy and re-use a complete network model. Leveraging technologies from PIPESIM flow assurance simulator, python API toolkit, open-source machine learning packages in python, and a commercial visualization dashboard, this paper proposed a series of workflows to simplify model deployment and set up an automatic advisory system to provide insight as a mean to justify an engineer's day to-day engineering decision.

A total of three steps was prepared to achieve field-level automated optimization system. First, is the creation of digital twin of well and network model. To eliminate potential data errors, reduce time consumed, and to merge various part of the model into one, a scalable python script was made. Second, an automated calibration workflow is created as performance issues also arises for individual branch calibration matching. Hence a combination of technologies was utilized to automate daily data acquisition and model update from production database and run a supervised machine learning model to continuously calibrate the network model. The last one is creating the customizable optimization workflow based of field KPIs, which results are derived from daily optimization run. The results are available in a personalized network surveillance dashboard accessible for engineers to create rapid decisions.

From the first and second steps, time consumed was reduced from 30 minutes/well to 10 minutes/well in bulk well modelling workflow and from 2 hours to 10 minutes for the network model merge with the assumption of 100 wells in one network. It would also greatly increase data integrity and consistency issues as it eliminates wearisome input process. On the last step, the model was successfully updated with the latest production data and the well IPRs' Liquid PI, reservoir pressure, and holdup factor are predicted from ML with more than 90% accuracy. As result delivery, the surveillance dashboard will be populated daily with the network production data, flowing parameters, and operation recommendations. It is estimated more than 90% time is saved from manual individual runs to digital comprehensive optimization.

Keyword: Digital Oilfield, Production Optimization, Surveillance Workflow, Artificial Intelligence, Production Automation

1 Introduction

1.1 Background

Production optimization is a propitious work despite the severity of the process. Furthermore, to get a detailed understanding of the production system thermohydraulic, facility design, and the amount of changes one could make to have a major effect on production target, the modeling is required to be done in a network level to properly account for the interdependency of wells and surface equipment and determine the system deliverability as a whole by optimizing operational parameters (Hammadi, et al., 2020).

In an optimization case, aside from having the maximum oil production rate as large as possible, the need to ensure that engineer uses his/her resources (e.g. Gas in gas lift injections, electrical power in ESP-equipped wells) does exist. In response to this requirement; a multiphase flow simulator for wells and pipelines (PIPESIM) which includes an accurate numerical representation of an asset's potential performance was used to present the automated optimization workflows in this paper.

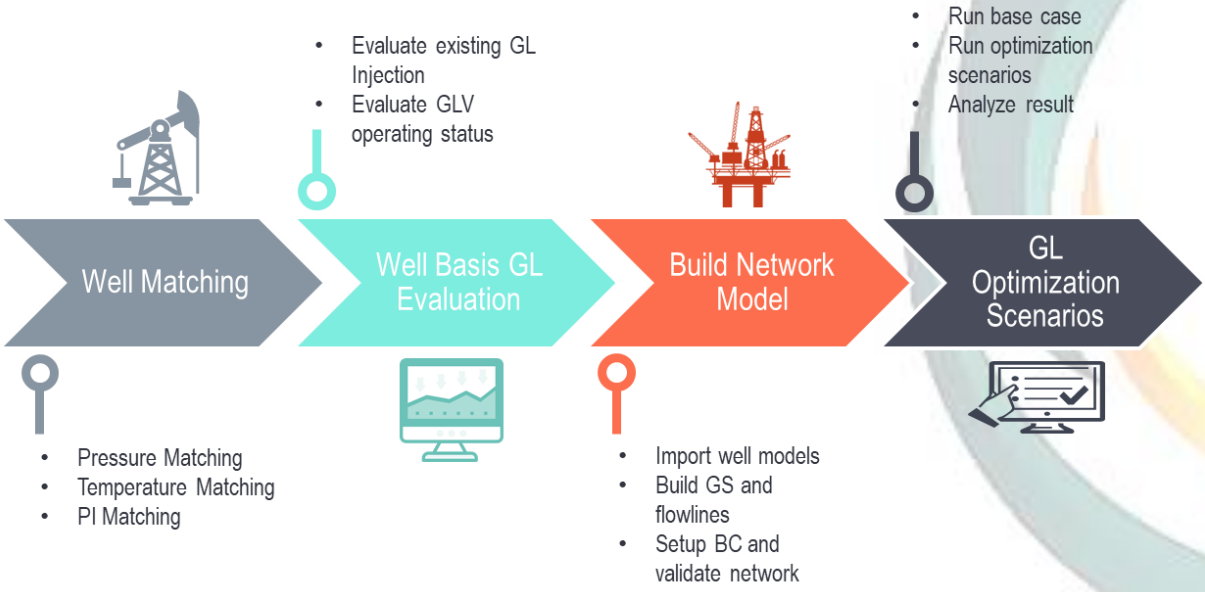


Figure 1 - Main steps of deploying a network optimization model

Summarized from Hammadi, et al. (2020) as shown in **Figure 1**, there are four steps that are needed to be taken in order to have a valid network model which can accurately describe the production system fluid flow behavior and suitable to support in day to day decision making for operations and optimizations purposes:

1. **Well Model Building and Matching**: starting with data gathering and review, this is a step where individual well models were built and calibrated to current conditions by data matching.

With the help of a Well Test and a Flowing Gradient Survey (FGS) data, a pressure-matching process can be undertaken to obtain the most acceptable flow correlation and correction factors (Friction and Holdup Factors). Should one have temperature points or profile available, one can match the values to get a sensible U value (heat transfer coefficient). Lastly, if there are any uncertainties present in the inflow part of our system, we can adjust our IPR parameters to match the operating condition (in this case, Liquid PI as the study used Well PI as its IPR model).

2. **Well Basis GL Evaluation**: in addition to wells calibration, two operations (Gas lift response evaluation and Gas lift diagnostics) have been created for each of the wells to show how the production rate and possible gas injection depth vary as a function of injection rate and the sensitivity parameter, and to get insight on how the gas lift system setup (gas lift valves) operate (is there any multi-pointing case?) in the well.
3. **Network Modelling**: the network construction starts by importing all matched well models, then add network flowlines and equipment along with each component's setting (if flowlines' pressure and temperature data are available, data matching can be conducted at this stage) and finally update the network model with the most updated well test data to ensure the model is calibrated and suitable for the main goal which is the Production Optimization.
4. **Gas Lift Optimization Scenarios**: to perform the optimization with maximizing oil production rate as the objective function, the following steps are required:
    a. Base Case creation: necessary step to establish a base line (current operational conditions).
    b. Global and Local constraints definition: to supply constraints into the optimization scenario, well constraints, branches and nodes in the network were specified.
    c. Flow performance curves generation: included in the PIPESIM optimization run, gas lift performance curves are generated for each well which will be the basis for optimizing the wells on a network scale.

Based on the elaborated topics above, in order to build a robust optimization workflow, heavy and time-consuming processes from well modeling up to running the optimization itself are essential to be done, especially when the network size is scaled up to large number of wells (>100 wells). To add, when a production optimization run has attained a result, the best way to amplify the value that one can get from the workflow is to not stop there, the workflow should be continuous and recommend operating parameters on the basis of the everchanging wells and network operating conditions (e.g. separator pressure, Gas-Oil-Ratio (GOR), Water-Cut (WC), gas lift injection rate, well productivity (liquid PI), reservoir pressure).

However, to reach the last evergreen solution, the detailed workflow's deployment has some underlying challenges:

1. **Ensuring data integrity and consistency throughout stages of the workflow**: A lot of manual inputs need to be done from earlier well model building stage to daily update of field data to calibrate the model. Data input error or inconsistences even in a value's decimal length can be rolled-up and have significant impact that is hard to trace in a bigger scale.
2. **Time-consuming and repetitive actions**: To build the full-scale optimization manually, take an example a to build a well model, an engineer will have to deal with the simulator's different user

interfaces to input the data (15-30 minutes) and do the exact same process for amount of wells times. Not to mention the network daily field data update to both the wells' and network's components.

3. **Resources constraints**: A team of production and reservoir engineers do have time and resource limitation as there are other high-valued works to be done.

1.2 Objective

The objectives of this automated intelligent production network optimization workflows creation are:

1. To minimizes risk for data quality issues from human errors
2. To speed-up pre and post deployment of a continuous production network optimization workflow hence saving time and resources
3. To extend the solutions to complete a digital oilfield workflow:
    a. Add a production surveillance dashboard
    b. Extract publicly open temperature data from a weather service API
    c. Add a Machine Learning (ML) Auto-calibration workflow

1.3 Methodology

The author has created four comprehensive workflows that automates some parts of the production network optimization workflow in order to complete the objectives listed above. Using some technologies such as multiphase flow simulator, the simulator's API, Microsoft Excel, data visualization tools, relational database, and Python open-source packages (pandas, numpy, matplotlib, seaborn, pyodbc, and scikit-learn) which are discussed in the next section.

For this case study purposes, the multiphase flow simulator used is PIPESIM along with its Python toolkit API for its simplicity and robust libraries availability for integration. For relational database, SQL Server is chosen, and for visualization tools, Power BI is picked. For the last two the options are not limited to the selected within this case study and can be adjusted of what's already been implemented on the field.

The workflows that were built to achieve automated production network optimization:

1.3.1 Workflow 1 - Bulk Well Modeling

In order to simplify step number one in the production network optimization workflow, the workflow that provide engineers to put inputs for well model (e.g. well general, casing/ liner/ tubing, artificial lifts, downhole equipment, completions, and fluids properties) into an Excel sheet to avoid potential data errors and time consumed in building large number of well models using Graphical User Interface (GUI). After the data has been inputted to the Excel sheet, a python script embedded to the Excel file that reads all the input according to the variable it will be put into for all the wells will be run and individual well models inside a designated well model will be created accordingly within minutes of work.

Figure 2 - Bulk Well Modeling Workflow

1.3.2 Workflow 2 – Automatic Network Merge

In a big network optimization model construction, the subsurface team handles the well models' creation and matching while the surface team responsible for the network part. This workflow is established to enable both departments works to integrate in a large scale (many chunks of production network and large number of wells) into one network model so that all matched results from previous processes can be brought over. Where a simple copy paste might have failed, a python script is created to ensure that the mechanism is working

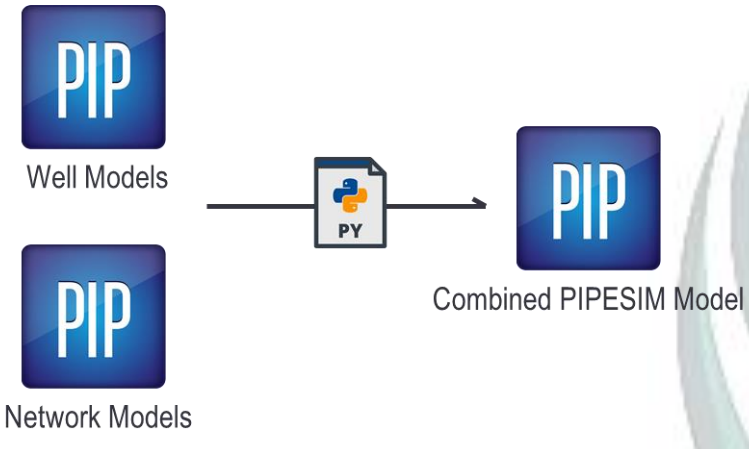


Figure 3 - Automatic Network Merge Workflow

1.3.3 Workflow 3 – Daily Data Automatic Feeding

As elaborated before, to gain more value from the production network optimization workflow and prepared Network Model, the deployment of the workflow should be continuous, and it should recommend new optimization scenario or give insights about current operating condition as the field operating condition changes. Engineers have data continuously stream into their production database, thus a system that will be able to extract the latest data which have different frequencies (e.g. well test data is sporadic, while field equipment data is daily) into the prepared network model should exists to ensure the validity of the model. The goal is to have a digital twin of the real production system in the field.

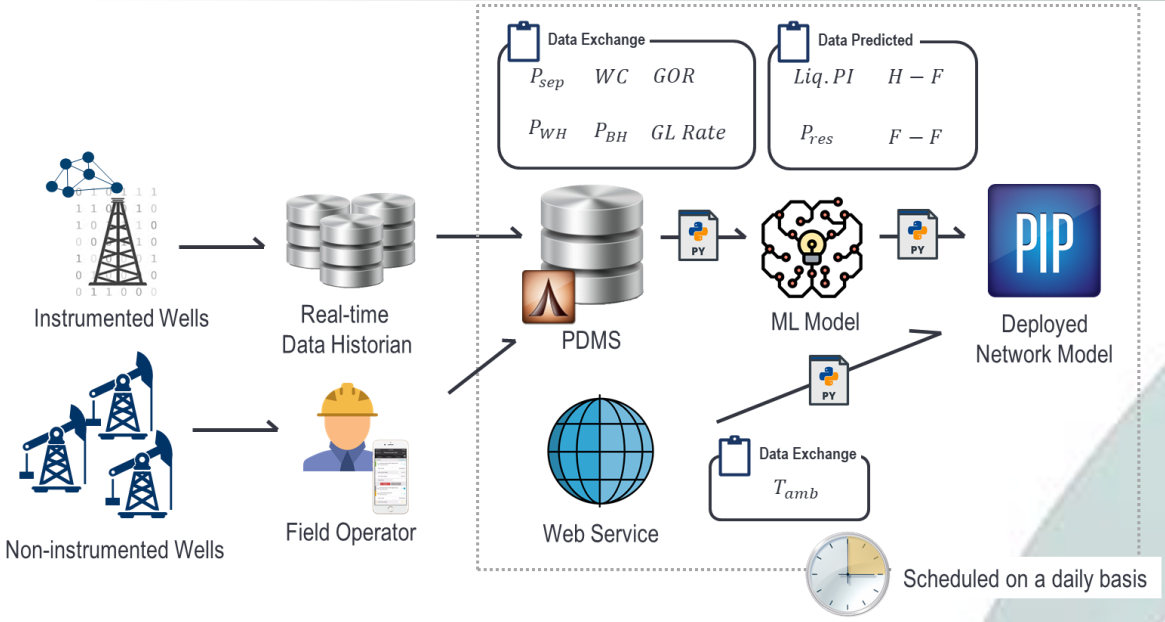


Figure 4 - Daily Data Automatic Feeding

This automatic update may seem simple but in a large network model which consists of hundreds of wells, done daily and manually, the task can be a time-consuming and tedious process which leads to high-risk of poor data quality. Thus, a python script has been prepared to obtain data from the Production Database Management System (PDMS) and the web service, to push and finally calibrate the model with the latest:

1. Well test data (e.g. water cut, gas oil ratio)
2. Field equipment data (e.g. gas lift injection rates, separator pressures)
3. Ambient temperature data from a weather service

1.3.3.1 Machine Learning Auto-calibration Workflow

As a part of workflow number three, the ML Auto-calibration workflow exist to predict some unknown parameters that will change as the production lifecycle of a well goes-on such as the well's completion's Productivity Index (PI) and reservoir pressure (**Figure 5**). Through training and tuning, supervised machine learning models to predict the earlier parameters, as well as the well's flow correlation's holdup-factor and friction-factor were made. The two latter constants (**Figure 6)** can be used to give weight to frictional and hydrostatic pressure-drop calculations of the well and hence tune the calculation. The list of assumptions to this workflow is as follow:

1. Wellbore models were assumed to rely on the datum depth of reservoir pressure. This would ensure that the flowing bottom-hole pressure and the reservoir pressure were referring to the same depth; hence there wouldn't be mismatch due to hydrostatic head
2. Every flow correlation set in each well is the most suitable to the well's flowing condition (Flow correlation ML prediction model was not created in this study)

3. The well models were configured to use the Well PI as their IPR Model. For those wells that cross the bubble point pressure region the option to use Vogel equation was applied

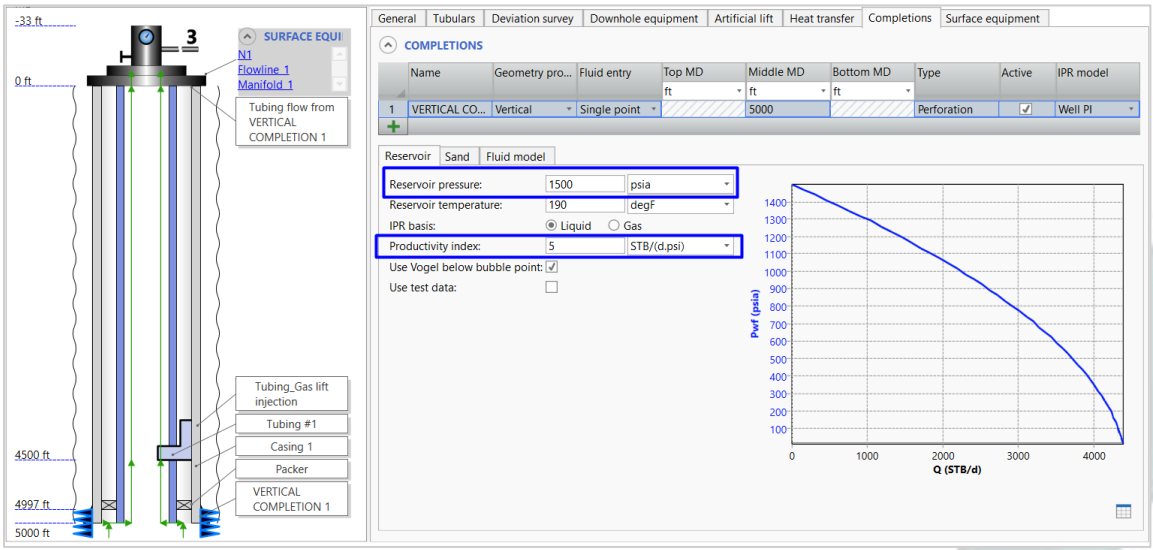


Figure 5 - Physical Well Model - Reservoir Properties and Parameters to be predicted

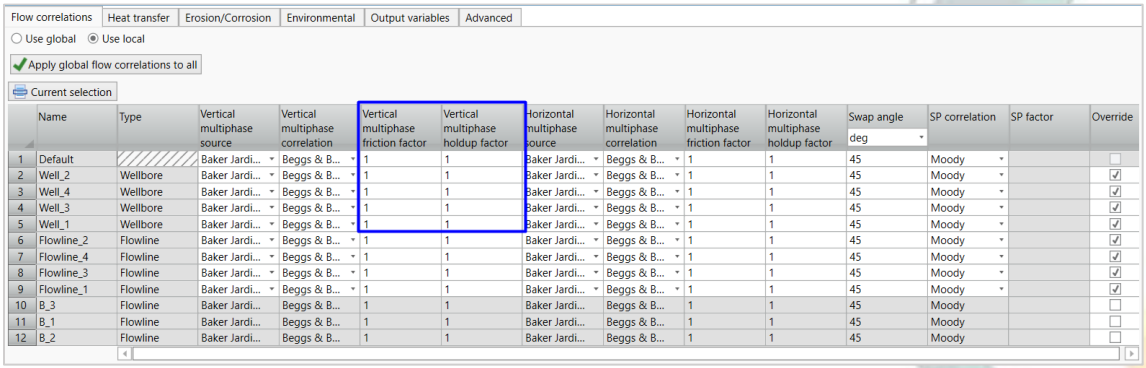


Figure 6 - Simulation Settings and Parameters to be predicted

The following processes to get the ready to-be-deployed model were used (**Figure 7**):

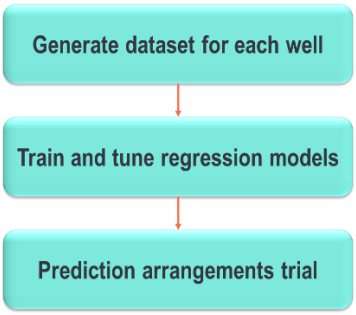


Figure 7 - ML Auto-calibration Workflow

- **Step 1 -** Generate dataset for each well: With the physical model and chosen flow correlation in place, nodal simulations were run to get the combinations of variations these values for each well
  - Wellhead Pressure
  - Gas Lift Rate
  - Water Cut
  - Reservoir Pressure
  - Holdup-Factor
  - Friction-Factor

  The parameter chosen to be sensitized is subject to the well, reservoir, and fluid type (e.g. a gas well that has no connate water or aquifer shouldn't mind sensitizing the water cut parameter) And the values used for each parameter were based on the current value of each parameter. The maximum and minimum values for each parameter should be enough to cover the possibilities of the parameter value to change in the field.

- **Step 2** – Train and tune four regression models to get the best model to predict the four parameters: In this process, there are several important steps to note,
  - Data preparation to be supplied to the machine learning model. The default features to be supplied to the machine learning model are the ones extracted from the latest well test data and field equipment data.
  - Selection the best supervised ML algorithm to be used. In this case, these following algorithms were tested:
    - K-Nearest Neighbour (KNN)
    - Support Vector Regression (SVR)
    - Random Forest Regression (RFR)
    - Extreme Gradient Boosting (XGBoost)
    - Linear Regression
    - Elastic Net Regression
  - Tuning of the selected algorithm's hyperparameter using Randomized Search CV.
  - Cross-validation of training data using Randomized Search CV.
  - Feature engineering (feature reduction) process to further improve the model performance by reducing parameter which consists of these following steps:
    - Feature importance using mean score decrease method.
    - Feature-feature and feature-target correlation using normalized pearson plot.
    - Feature scaling should the algorithm is sensitive to unscaled features (e.g. KNN, SVR)
- **Step 3 –** Try for prediction arrangements. The predicted parameters (Liquid PI, Reservoir Pressure, Friction-factor, and Holdup-factor) could be correlated to one another. It would be possible if one parameter is best to be predicted after the prediction of the other

At last, along with the latest field data extracted from the database and weather service, the ML model's prediction results will be populated to the prepared PIPESIM model daily.

1.3.4 Workflow 4 – Smart Advisory System

This workflow serves to provide solution to the dynamic operation condition. Should workflow number three has already been deployed, one will have calibrated and ready to-be-used network model every day. Thus, to get an in-depth network analysis and operation recommendation with the deployed model

pertaining to the current condition, an automatic daily advisory system can be built by running both PIPESIM's Network Simulation and Network Optimization tasks.

From Network Simulation task, PIPESIM will output Pressure and Temperature profiles along the network, as well as other flowing parameters such as Maximum Erosional Velocity Ratio (EVR) and Maximum Liquid Loading Velocity Ratio (LLVR). These important derived flowing parameters are used to monitor erosion and liquid loading risks, and it can only be obtained by running such simulation thus it gives added value to the surveillance framework. On the other hand, the Network Optimizer gives out the recommendation on how to get maximum oil or maximum gas by varying set control variables (e.g. gas lift rates) within set constraints. Hence, for example, one will get the recommended gas lift rates or ESP frequency to get the maximum oil rate from its oil production network based on stated constraint (such as surface facility capacity limitation).

A Python script powered by PIPESIM Python Toolkit API acts as the integrator of this workflow. It runs the above tasks daily (triggered from the Task Scheduler) and send the results to the company's PDMS which from there, a prepared PowerBI dashboard will pull and gather the data along with the latest well test, field equipment and back-allocation data to ease the engineer to have an all-in-one canvas as the basis of daily engineering justifications

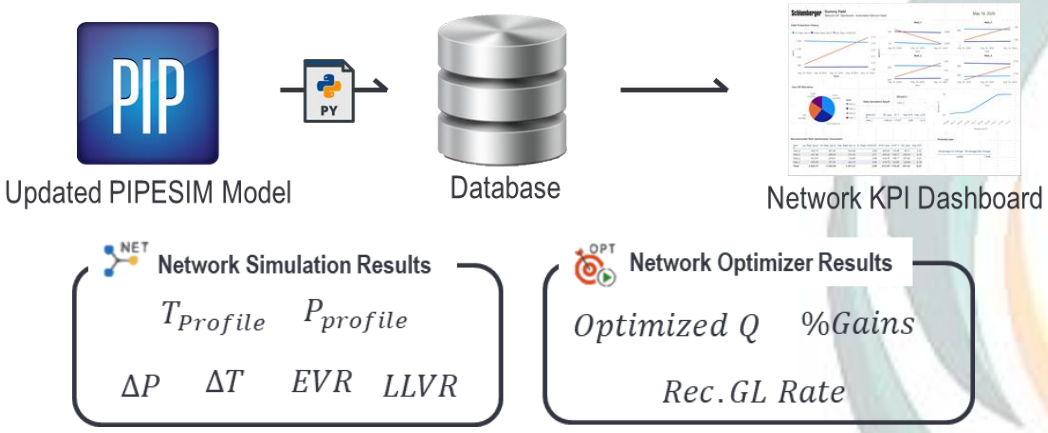


Figure 8 - Smart Advisory System Workflow

2 Basic Theory

2.1 Multiphase Steady-state Flow Simulator

PIPESIM is used to simulate multiphase flow simulation throughout the production system from the reservoir through to the surface facilities to enable comprehensive production (and injection) system analysis. Steady state means that the mass flow rate is conserved throughout the system, no accumulation of mass in any component of the system. This simulator is used starting from design to the optimization period of oil, gas and water production and injection system. It is most often used by reservoir, production, and facilities engineer to model well performance, design artificial lift systems, model pipeline networks and facilities, analyze field development plans, and optimize production.

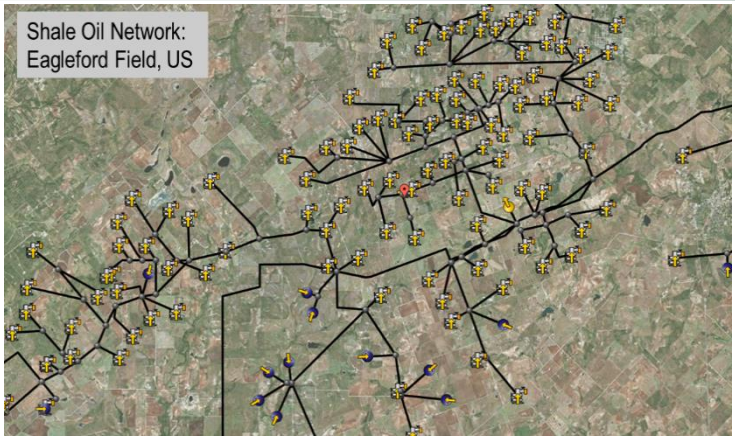


Figure 9 - Example of a Network Model with a GIS Map

2.1.1 Network Simulation

Network modeling is needed should one needs to take account the interaction of the different components (such as compressors, separators, and wells) producing into common gathering systems. As the wellhead pressure and, by extension, the deliverability of any well is influenced by the back pressure imposed by the production system. It also allows the engineers to determine the effects of changes such as adding new wells, adding compression, looping flowlines, and changing separation system.

To be able to analyze the whole components in a network, the network should be solved based on the boundary conditions (Pressure/ Flow rate/ PQ curve) set on each component. The network will be converged when the pressure balance and mass balance at each node are within the specified tolerance. Then it will output parameters along the flow path (profile) and at each node in the network. In the Network Simulation input pane (**Figure 10**), one can also override the phase ratios (e.g. WC, GOR) set to each well.

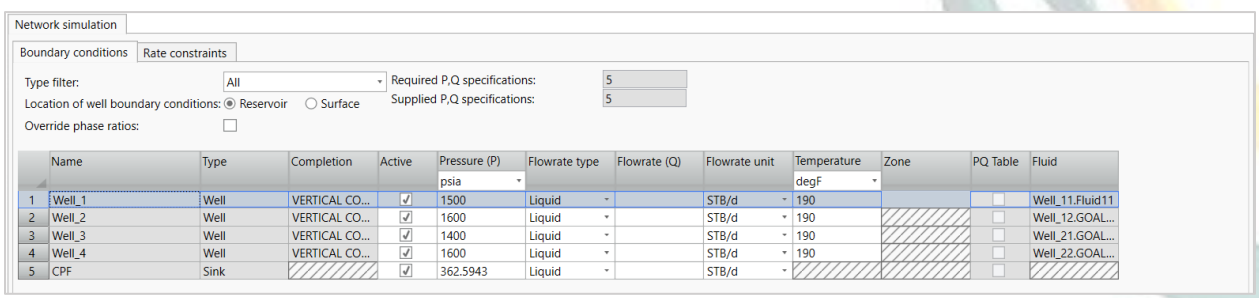


Figure 10 - PIPESIM Network Simulation input

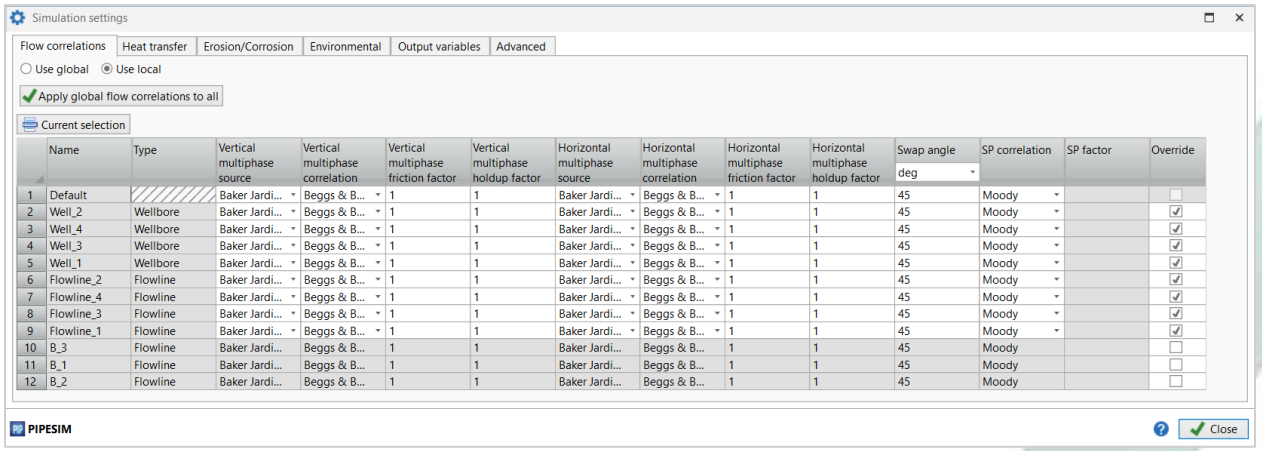
2.1.1.1 Simulation Settings

Before running the Network Simulation, it is a common practice to set the Network tolerance inside the simulation settings. Another important input to be set in the simulation settings is to choose flow correlations (**Figure 11**). Flow correlations to be chosen in PIPESIM are divided into three categories:

- Vertical multiphase correlation

- Horizontal multiphase correlation
- Single phase correlation

Each of these correlations can be set for global (all wells and flowlines will use the same correlations), or locally like the one showed in **Figure 11**.



	Name	Type	Vertical multiphase source	Vertical multiphase correlation	Vertical multiphase friction factor	Vertical multiphase holdup factor	Horizontal multiphase source	Horizontal multiphase correlation	Horizontal multiphase friction factor	Horizontal multiphase holdup factor	Swap angle	SP correlation	SP factor	Override
											deg			
1	Default		Baker Jardi...	Beggs & B...	1	1	Baker Jardi...	Beggs & B...	1	1	45	Moody		☐
2	Well_2	Wellbore	Baker Jardi...	Beggs & B...	1	1	Baker Jardi...	Beggs & B...	1	1	45	Moody		☑
3	Well_4	Wellbore	Baker Jardi...	Beggs & B...	1	1	Baker Jardi...	Beggs & B...	1	1	45	Moody		☑
4	Well_3	Wellbore	Baker Jardi...	Beggs & B...	1	1	Baker Jardi...	Beggs & B...	1	1	45	Moody		☑
5	Well_1	Wellbore	Baker Jardi...	Beggs & B...	1	1	Baker Jardi...	Beggs & B...	1	1	45	Moody		☑
6	Flowline_2	Flowline	Baker Jardi...	Beggs & B...	1	1	Baker Jardi...	Beggs & B...	1	1	45	Moody		☑
7	Flowline_4	Flowline	Baker Jardi...	Beggs & B...	1	1	Baker Jardi...	Beggs & B...	1	1	45	Moody		☑
8	Flowline_3	Flowline	Baker Jardi...	Beggs & B...	1	1	Baker Jardi...	Beggs & B...	1	1	45	Moody		☑
9	Flowline_1	Flowline	Baker Jardi...	Beggs & B...	1	1	Baker Jardi...	Beggs & B...	1	1	45	Moody		☑
10	B_3	Flowline	Baker Jardi...	Beggs & B...	1	1	Baker Jardi...	Beggs & B...	1	1	45	Moody		☐
11	B_1	Flowline	Baker Jardi...	Beggs & B...	1	1	Baker Jardi...	Beggs & B...	1	1	45	Moody		☐
12	B_2	Flowline	Baker Jardi...	Beggs & B...	1	1	Baker Jardi...	Beggs & B...	1	1	45	Moody		☐

Figure 10 - Flow Correlation Selection

To help match the flow correlation calculation to the field data, engineers put some calibration factors to the friction factors and holdup factors:

- Friction factors: adjust the frictional pressure drop linearly. For example, if the friction factor is set to 0.5, the friction element of pressure drops computed by the correlation is halved. (Schlumberger, 2019)
- Holdup factors: adjust the liquid holdup from the value predicted by the correlation with a non-linear relationship:

$$H_L = f_H H_{Lc} + (1 - f_H) H_{Lc}^2 \quad \ldots (1)$$

In the equation above, H_{Lc} is the value predicted by the correlation and f_H is the holdup factor.

2.1.2 Network Optimizer

Network Optimizer adjust the control variables set (gas lift rates/ ESP/ PCP speeds/ choke bean size) in order to minimize/ maximize the value of a tailored objective function (e.g. maximize oil produced/ minimized water produced). Applied constraints, local and/or global, will be honored during the optimization. For local constraints, there were many options to limit the maximum liquid rate, water rate, and gas rate in each well or in one group, as well as the amount of the injected gas into the wells. Table 1 shows the list of local and global constraints that can be defined in the model. All input to the optimizer is combined in one single task

Table 1 - PIPESIM Network Optimizer Local and Global Constraints

	Min/Max Q liquid	Min/Max Q total gas	Max Q water	Max outlet temp	Max drawdown	Min Pb margin	Min/Max Q gas lift	Max CHP	Min/Max power	Min/Max freq.	Min/Max bean size	Max GOR	Max EVR	Max velocity	Max CO2	Max H2S
Gas Lifted	👍	👍	👍	👍	👍	👍	👍	👍								
ESP Lifted	👍	👍	👍	👍	👍	👍			👍	👍						
PCP	👍	👍	👍	👍	👍	👍			👍	👍						
Choke Contr.	👍	👍	👍	👍	👍	👍					👍					
Flow Lines	Max 👍	👍	👍									👍	👍	👍	👍	👍
Sinks	Max 👍	👍	👍									👍			👍	👍
Global Constr.	Max 👍	👍	👍									👍			👍	👍

The task works by running well performance curves at first, before finding the optimum value to satisfy the objective function. The optimization algorithm is developed by Schlumberger Doll Research (Cambridge, MA) and it is based on Basic Open-source Mixed Integer (BONMIN) framework which applied on Mixed Integer Non-Linear Programs (MINLP) solvers.

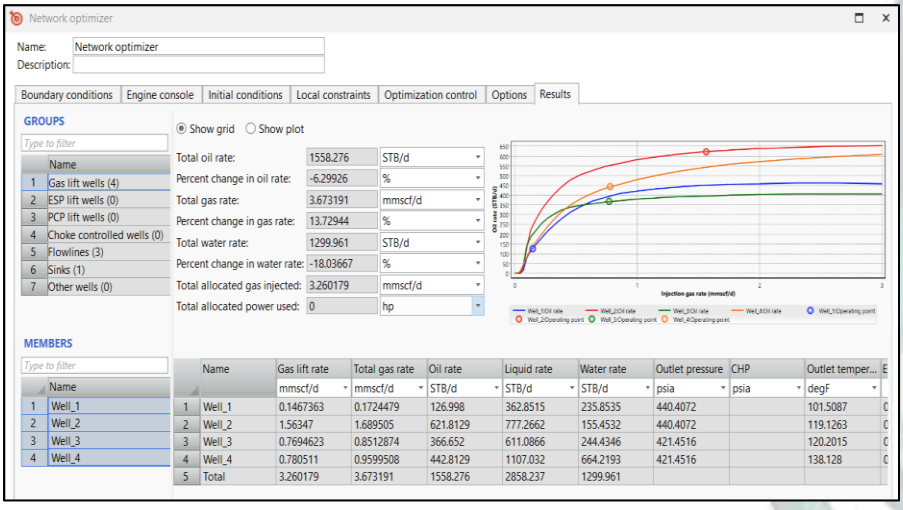


Figure 11 - Example of PIPESIM Network Optimizer Result

2.1.3 PIPESIM Python Toolkit

PIPESIM Python Toolkit is a Software Development Kit (SDK) for working with PIPESIM using Python programming language, in the form of Python module named Sixgill which designed to perform actions with a similar user perspective to the PIPESIM GUI. It runs independently from the PIPESIM GUI by creating a session (started when it opens a PIPESIM model from Python program). A session can be considered as an input/ output (I/ O) communications channel with a PIPESIM model. User can query the model for its contents and parameters; add, delete and modify the model components; and perform tasks on the model (e.g. Nodal Analysis, Network Optimizer, etc.).

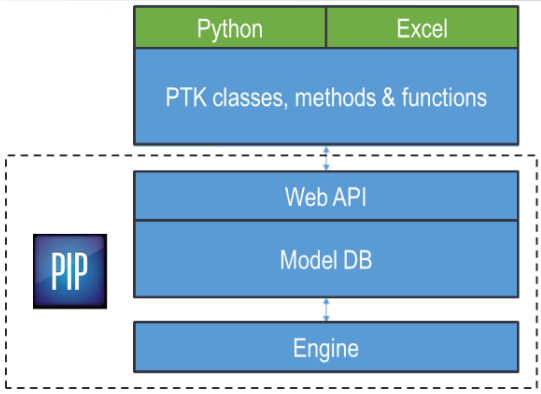


Figure 12 - PIPESIM Python Toolkit Infrastructure *(Schlumberger, 2019)*

Other than the Sixgill package, the installation of PIPESIM Python Toolkit also comes with PyXLL and xlwings packages which help users to interface PIPESIM from Microsoft Excel as PyXLL brings its PIPESIM Python Toolkit Excel plugin to user's Excel application. Another common package that is included is pandas, which is useful for working with data sets and arrays. As the tool uses Python as its main driver, users can also integrate its PIPESIM model with other open-source libraries to extend and boost the values of their workflows. In this study for example, other packages used to enhance the workflows are pyodbc to connect with databases, scikit-learn for machine learning workflows, and jcopml to ease the creation of machine learning models.

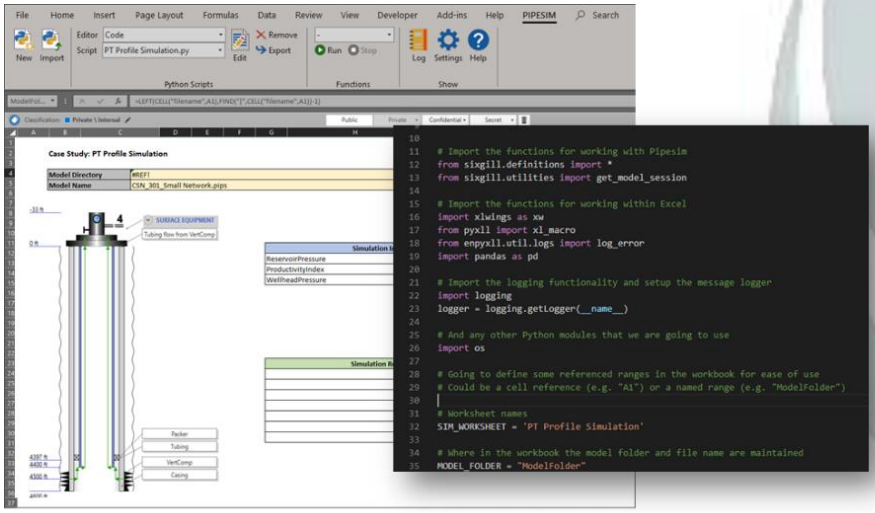


Figure 13 - Interfacing PIPESIM from Excel with PIPESIM Python Toolkit

2.2 Supervised Machine Learning

In this study, the method that the author used is called Supervised Machine Learning. From James, Witten, Hastie, & Tibshirani, (2013), there are a set of inputs X that are readily available, and the Machine Learning model will try to predict the Y by using:

$$\hat{Y} = \hat{f}(X) \quad \ldots (2)$$

Where $\hat{f}$ represents our estimate for the real f, and $\hat{Y}$ represents the resulting prediction for Y. $\hat{f}$ is often treated like a *black box*, in the sense that one is not typically concerned with the exact form of $\hat{f}$, if it yields accurate predictions for Y. When an algorithm is 'fitted' to a training data, it constructs the $\hat{f}$ by reducing the error function written as:

$$\begin{aligned} E\left(Y - \hat{Y}\right) &= E\left|f(X) + \epsilon - \hat{f}(X)\right|^2 \\ &= \left|f(X) - \hat{f}(X)\right|^2 + Var(\epsilon) \end{aligned} \quad \ldots (3)$$

Where $E\left(Y - \hat{Y}\right)$ represents the average, or *expected value*, of the squared difference between the predicted and actual value of Y, and $Var(\epsilon)$ represents the *variance* associated with the error term ϵ which becomes the irreducible part of the term.

2.2.1 Data Preparation

In order to be able to fit the data to get the best value of $\hat{Y}$ that as close as possible to Y, a set of training data (X and Y) should be provided to train the model and at the same time, a set of test data (X and Y) ready to evaluate the and score the model should also be available. Both sets need some preparation, for example the ML package that we are using cannot directly read string values from a categorical column, hence all categorical column should be encoded, the most common method is to use *one-hot encoding*. Thus, to make the process seamless, *pipelines* were used. After dataset splitting, numerical column will be collected by numerical pipeline to undergo some numerical preprocessing (e.g. data impute, scaling, transformation, polynomial features generation) and at the same time, categorical columns will be gathered by the categorical pipeline to apply the preprocessing for categorical values (e.g. data impute, encoding).

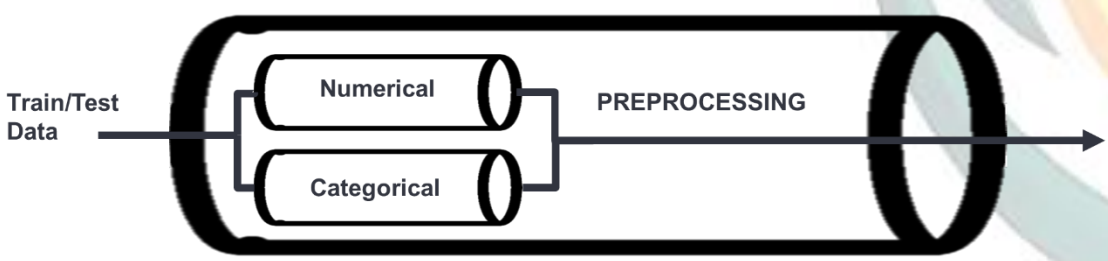


Figure 14 - Preprocessing Pipeline Schematic

2.2.2 Regression Algorithm

In this study, six regression algorithms mentioned in the **methodology** section were tested. There was *parametric* algorithm (reduces the problem of estimating f down to one of estimating a set of parameters) such as Linear Regression and Elastic Net Regression, and *non-parametric* method like K-Nearest Neighbors, Support Vector Regression, Random Forest Regression and Extreme Gradient Boosting

(XGBoost). All algorithm will be trained and solved to reduce the error function, which in this case, is the R^2:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \quad \ldots (4)$$

which will be negative should our prediction is worse than just taking the average value. In this study, only two algorithms were finally selected to be use, KNN and RFR, hence only those two algorithms will be discussed in this paper.

2.2.2.1 K-Nearest Neighbor (KNN)

KNN algorithm takes the value of the nearest N neighbors to be considered for the predicted value. Besides the number of the neighbors, KNN also has the option to put distance-based weighting to these nearest N neighbors. Finally, the algorithm allows user to select the distance calculation method (e.g. Euclidean, Manhattan, and Chebyshev) to be used should one opt in for the distance-based weighting.

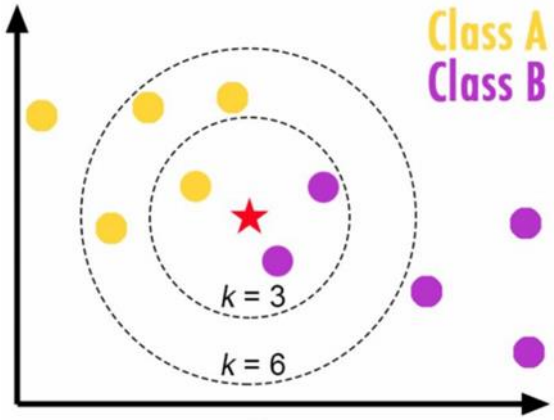


Figure 15 - Illustration of KNN Algorithm *(James, Witten, Hastie, & Tibshirani, 2013)*

2.2.2.2 Random Forest Regressor (RFR)

RFR model was created by applying bootstrap and aggregation to decision tree method. Decision tree is a ML algorithm which model built upon a lot of decisions towards its feature. In **Figure 16**, a decision tree with 2-level depths which is applied to a univariate data. When one fit a decision tree to a training data, the algorithm will determine where the decisions will be put in order to produce the lowest error.

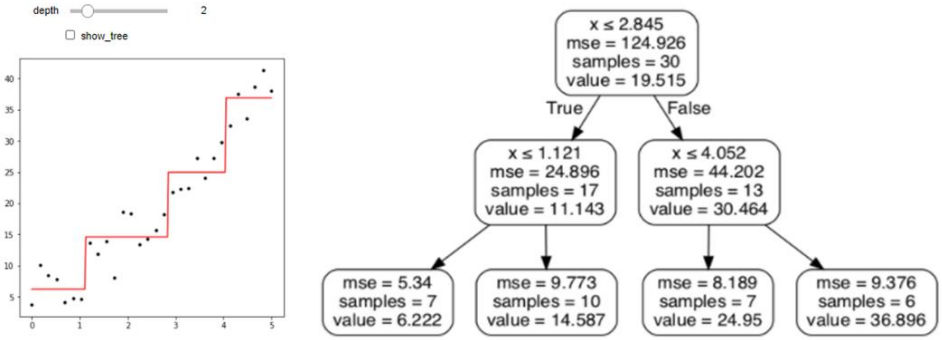


Figure 16 – Illustration of Decision Tree Algorithm *(Putra, LuWiji, 2019)*

In a hyperparameter tuning case however, when one tunes the tree depth and make it unconstrained, it can reach a level where the number of samples in each of the decision tree becomes only one sample. This case ill has a very high accuracy but overfitting tendency. Then the idea of bootstrap and bagging comes to play. From 100 rows of available in the training data set, several number of trees will be created with 60% of those 100 rows of data by random sampling with replacement (bootstrapping). Next, after each of the decision tree is fitted to their respective data, the result will be aggregated to form a random forest model. Thus, this shows that random forest is the further development of decision tree which overcomes the over-fitting problem by performing bootstrap and aggregation, or commonly known as bagging.

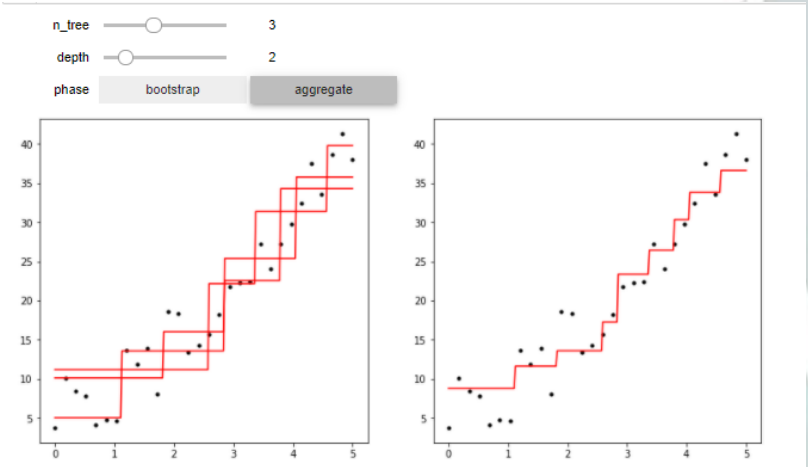


Figure 17 – Illustration of Random Forest Algorithm *(Putra, LuWiji, 2019)*

2.2.3 Hyperparameter Tuning and Cross Validation

All the machine learning algorithms have some hyperparameter to be tuned to obtain the best parameter to be used for our data. For example, in KNN, the number of N nearest neighbor, whether to use the distance-based weighting or not, and the distance calculation to be used needs to be determined. Meanwhile when one train and validate some training data, there could be some luck factor pertaining to the automatically selected validation set. Hence one can use K-Fold Cross Validation method which conducts the training and validating process K-number of times by taking the validation set in turns from the training data available. To accommodate all these steps, Randomized Search CV (RandomSearchCV) was used. It tries several hyperparameter combinations within specified ranges of each hyperparameter,

and for each combination run, it conducts a K-Fold Cross Validation K-number of times to get the representative validation score for that certain combination. The best combination then selected from the one that has the best validation score. An illustration of Randomized Search CV with an SVR algorithm can be seen in **Figure 18**.

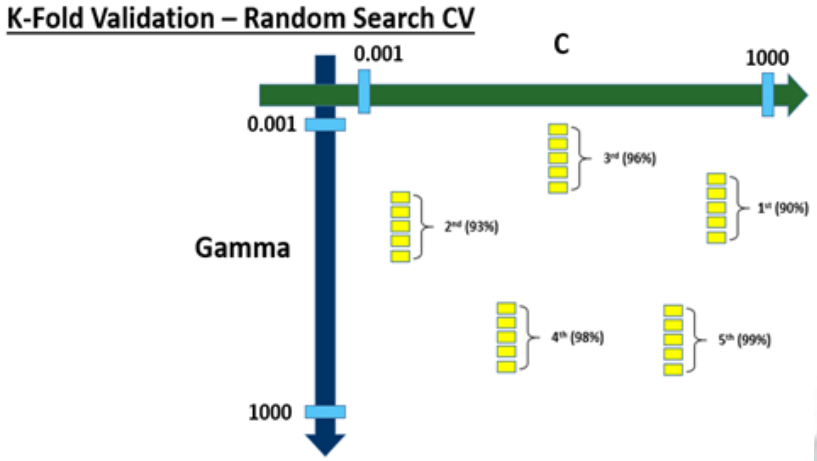


Figure 18 – Illustration of Randomized Search CV *(Irfan, Vidrianto, Rahmawati, & Naufal, 2020)*

2.2.4 Feature Engineering

Feature engineering is a step where one can do to improve his/her model by eliminating/ generating the number of features fed to the model.

2.2.4.1 Feature Importance

Feature importance is a method to filter the features so that only the ones that have large influence on the model that will be used for training the model. One of the tools to show feature importance used in this study is the mean score decrease plot. This plot shows the scoring reduction should a feature's column gets shuffled. The more the score reduction, the more important the feature. For example, in **Figure 19**, only 4 features have influence on the model's prediction result. Hence, should one be not looking to another analysis, he/she can retrain the model using these four features.

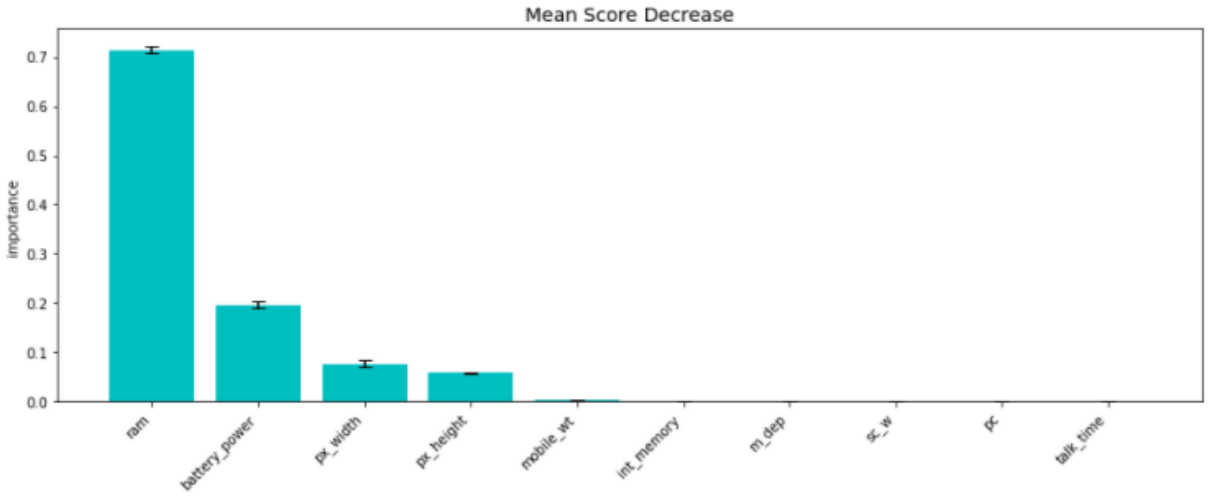


Figure 19 - Mean Score Decrease Plot *(Putra, J.COp ML, 2018)*

2.2.4.2 Correlation Matrix

This study also uses a normalized pearson plot as a tool to see feature-feature and feature-target linear relationships. The higher the correlation a feature has in the feature-target matrix, the more influence the feature is to the result. On the other hand, should two features have a high feature-feature relationship, one feature can be dropped. This analysis should be combined with the feature importance analysis to select features to be fed.

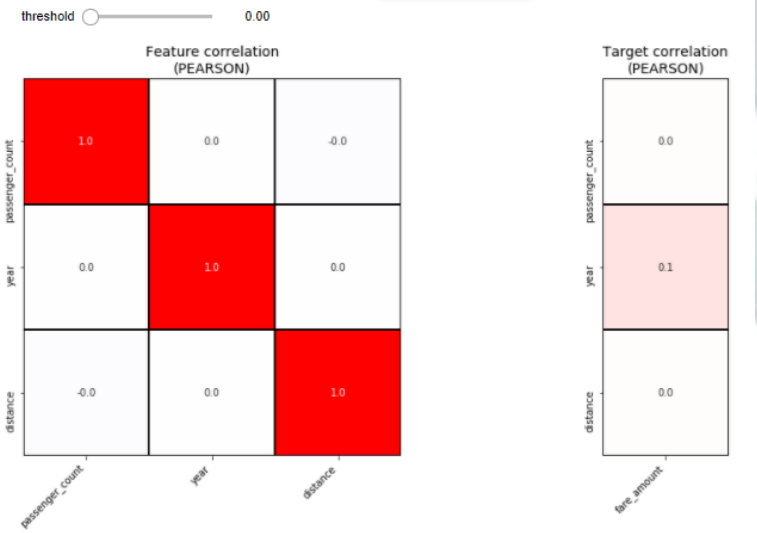


Figure 20 - Correlation Matrix Plot *(Putra, J.COp ML, 2018)*

2.2.4.3 Feature Scaling and Transformation

In some cases, the numerical features that fed to the model needs to be scaled, below are the advantages of feature scaling in some algorithms:

- Distance will be more balance (for algorithms that honour distance such as KNN/ SVR)
- Help machine learning solver, gradient descent, to get better solution due to scaled loss plane
- In linear regression algorithms, the variable coefficients would be more balance

There are three types of feature scaling methods:

1. Standard Scaler: the data is positioned to have its centroid (e.g. mean) to zero and scaled the data to have a standard deviation of 1.

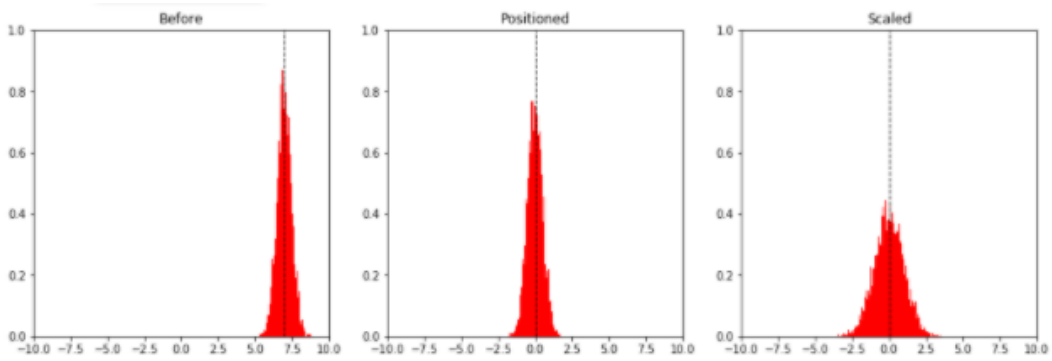


Figure 21 - Illustration of Standard Scaler *(Putra, Supervised Learning by J.COp, 2018)*

The standard scaler has a weakness as it can only handle normal distributed data, not skewed. Hence one can transform the features first using *Box-Cox/ Yeo-Johnson* transformation before scaling using standard scaler.

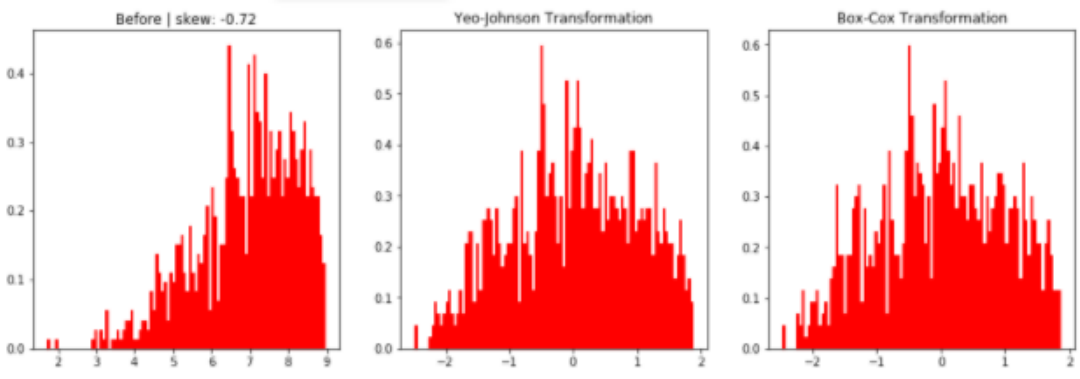


Figure 22 - Illustration of Data Transformation *(Putra, Supervised Learning by J.COp, 2018)*

2. MinMax Scaler: This method shifts the minimum to zero and stretch the data to a scale that is desired. The weakness of this scaler is that if the data has outliers, then all the data will get compressed.

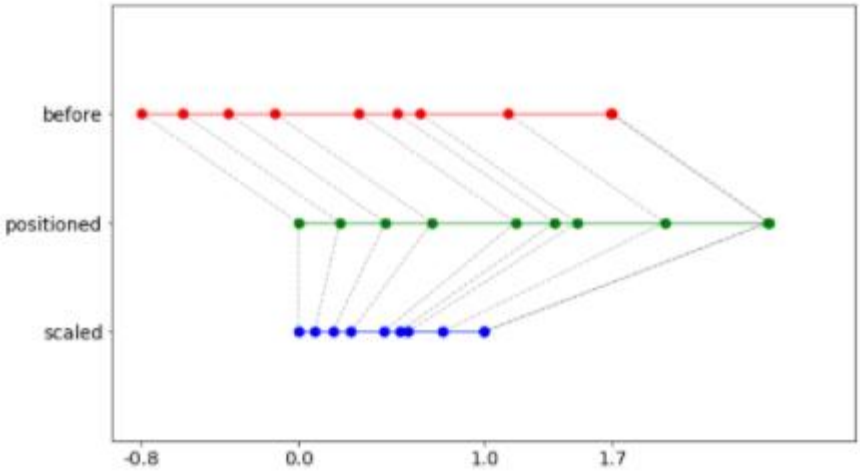


Figure 23 - Illustration of MinMax Scaler *(Putra, Supervised Learning by J.COp, 2018)*

3. Robust Scaler: It is the same as MinMax, but instead of scaling with the minimum and maximum value, it scales with its quartiles thus avoid the compression due to outliers weakness.

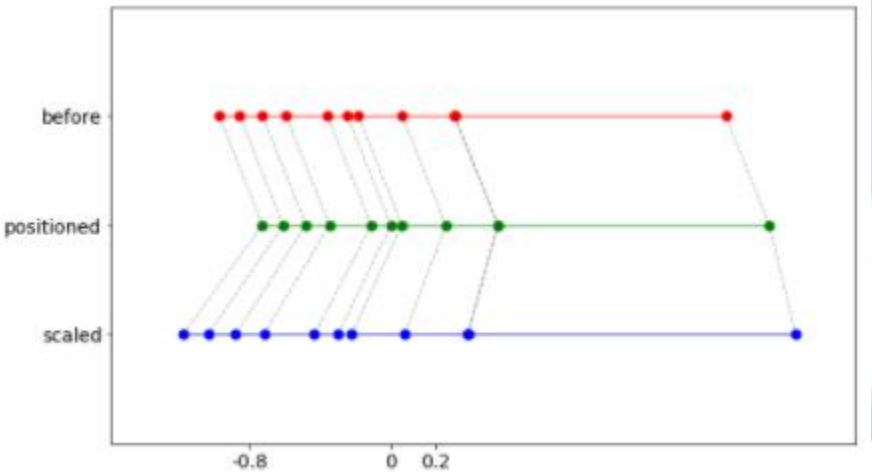


Figure 24 - Illustration of Robust Scaler *(Putra, Supervised Learning by J.COp, 2018)*

3 Case Study

3.1 Bulk Well Modeling

To implement the first workflow and measure the time taken to run the script and construct the well model inside a simulation file, 30 well data has been put into the created Excel format (Figure 25). The script then run and the time taken is recorded. It takes approximately 90" to complete the workflow with 30 wells which yields 3" for each well to be created inside a Network Model. Aside from the workflow time to be reduced, this workflow also lowers the risk of data input error and thus ensure data integrity and consistency because there is no data input to the GUI at all. The workflow is in an Excel format so that it is reproducible with another data and scalable to larger number of wells.

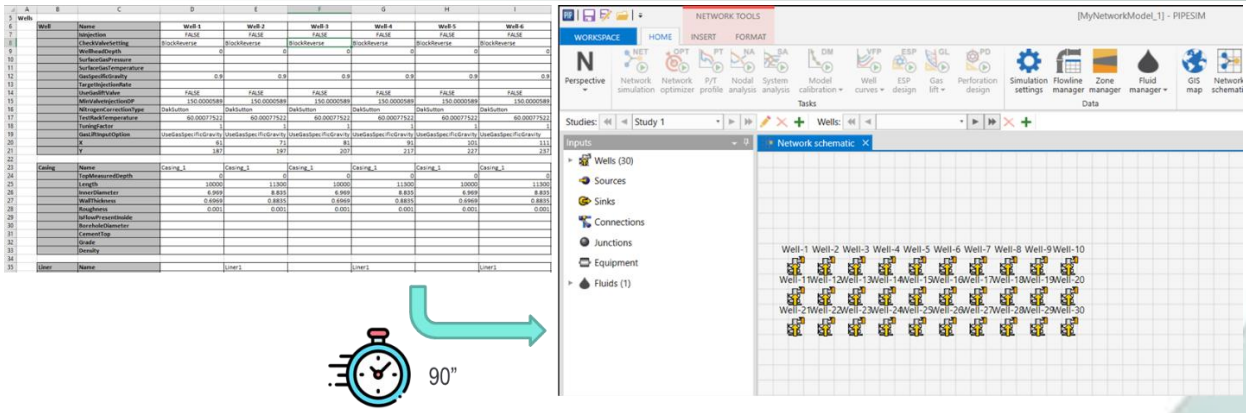


Figure 25 - Bulk Well Modeling Demo

3.2 Automatic Network Merge

To test the tool built for the workflow, the author prepared three different simulation models which contains different components. From **Figure 26**, 2 wells from Wells_GS_1 model supposed to replace sources attached to junction N1 and N2 from Network_GS_1 model. Then, the combined network should connect to another part of the full network system from Network_GS_2 file. In the real-world situation, it is possible to encounter this kind of challenge as the well models were calibrated by the subsurface team and the network model was done by the surface team. With the prepared Python script, different chunks of the model can be carried over into one single simulation model bringing the simulation settings and fluid model along. This ensure data integrity and consistency, as user doesn't have to do manual rebuilding of the network as well as fasten the process. The workflow is also scalable to many more simulation model, either it is single branch model or network model, to be combined at once

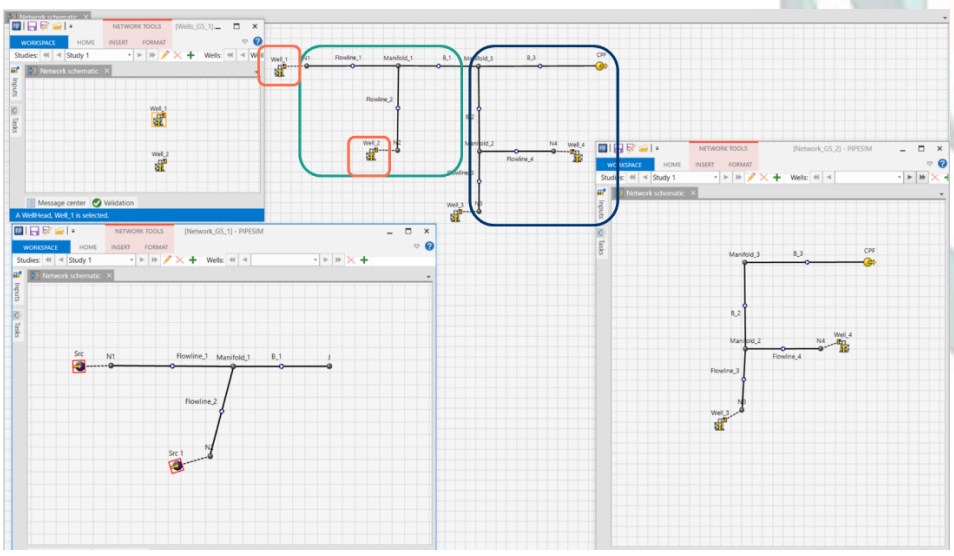


Figure 26 - Automatic Network Merge Demo

3.3 Daily Data Automatic Feeding and Smart Advisory System

For this workflow testing purposes, the author used a dummy field which has four gas-lifted wells (Well_1, Well_2, Well_3 and Well_4) with one sink at a Crude Processing Facility (CPF). The wells are oil wells with GOR ranges from 200 SCF/STB - 400 SCF/STB and WC vary around 20% – 65%. All wells have one Gas Lift mandrel, perforated at one interval below the End of Tubing (EOT) and completed with a production tubing.

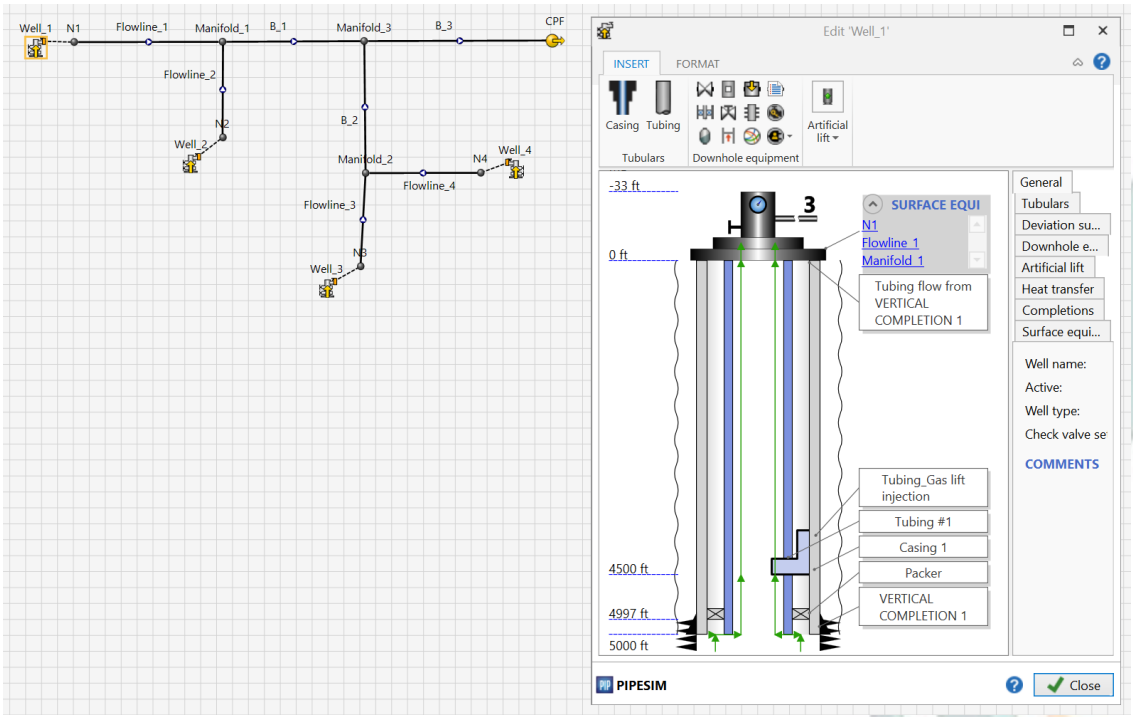


Figure 27 - Dummy PIPESIM Network Model

As the workflow has been discussed, the prepared workflow will take the latest well test and field equipment data prepared plus an ambient temperature data retrieved from a public weather service and store them in an on-premise SQL Server database (**Figure 28**). The workflow also runs series of parameters prediction (Liquid PI, Reservoir Pressure, and Holdup Factor) using trained ML model which creation discussed in the next sub-section in this paper. Then all data will be fed into the Dummy Network Model to have a ready-to-use calibrated simulation model.

Query 1 | Last_well_test_query | +

Activate premium features

Language SQL | Connection SLB-2S93JR2\SQLDEV2017.DUMMY_FIELD | Use DUMMY FIELD

```
SELECT * FROM dbo.WELL_TEST$ WHERE DATE IN (SELECT MAX(DATE) FROM dbo.WELL_TEST$ GROUP BY WELL)
```

Results | λ | SQL | IL | Tree | Format | Export | Activate Autocompletion

Result Set (4 items)

DATE	WELL	LIQ_RATE	OIL_RATE	WATER_RATE	GAS_RATE	ST_INJ_RATE	PROD_GAS_RATE	GOR	WATER_CUT	OUTLET_PRESSURE	BOTTOMHOLE_PRESSURE
5/19/2020 12:00:00 AM	Well_1	143.7392	47.43392	96.30524	0.4545342	0.445	0.009534225	201.000149260276	66.9999833030934	410.384001972244	1471.252
5/19/2020 12:00:00 AM	Well_2	678.3972	529.1498	149.2474	1.079772	0.975	0.1047717	198.000074837031	22.0000023585003	417.56381019345	1260.8
5/19/2020 12:00:00 AM	Well_3	526.9736	313.5493	213.4243	0.818197	0.75	0.06819701	217.500118801094	40.4999984818974	405.306140412058	1048.68666666667
5/19/2020 12:00:00 AM	Well_4	716.3416	293.7001	422.6416	0.7863053	0.67	0.1163053	396.000205651956	59.0000078174994	403.267885769682	1528.366
		2065.4516	1183.83312	881.61854	3.1388085	2.840	0.298808235	1012.500548550357	188.4999919609905	1636.521838347434	5309.10466666667

Figure 28 - Last Well Test Data

The goal for the workflow is, as mentioned, to automate the run of Network Simulation and Network Optimizer task. For the Network Optimizer, the objective function is set to Maximize Oil Rate, with Gas lift rate as a control variable, and Maximum injection gas rate is limited at 3 MMscf/D (**Figure 29**). These two tasks run daily, and the results are stored to prepared table inside the same SQL Server Database, which can be visualized through a PowerBI dashboard.

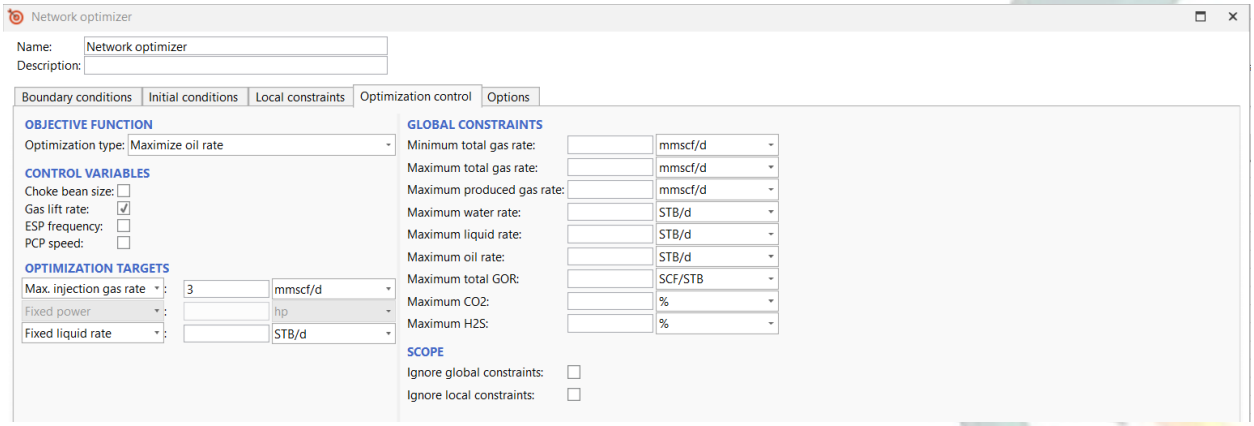


Figure 29 - Illustration of Network Optimization's Optimization Control

Figure 30 is a production surveillance dashboard made to complete the study. This dashboard will be refreshed daily as the automatic feeding and simulation tasks also triggered to run every day. From this dashboard, one can get insight on how the field perform from the production profiles along with the current gas lift injection allocations monitoring. The network simulation result can also be fond in the middle-right of the page, where one can see the branch's flowing parameter (pressure drop along the branch, temperature change, max EVR and max LLVR) as well as the temperature profile along the branch. In the bottom of the page is the operation recommendation coming from previous network optimizer run. The table recommends the gas lift injection to be increased in Well_1 (from 0.445 MMscf/D to 0.56 MMscf/D) and Well_4 (from 0.67 MMscf/D to 1.00 MMscf/D) then to be decreased in Well_2 (from 0.975 MMscf/D to 0.88 MMscf/D) and Well_3 (from 0.75 MMscf/D to 0.56 MMscf/D). With the recommended parameters, the simulation predicts the oil rate to increase to 1,349.24 STB/d (10.92% change).

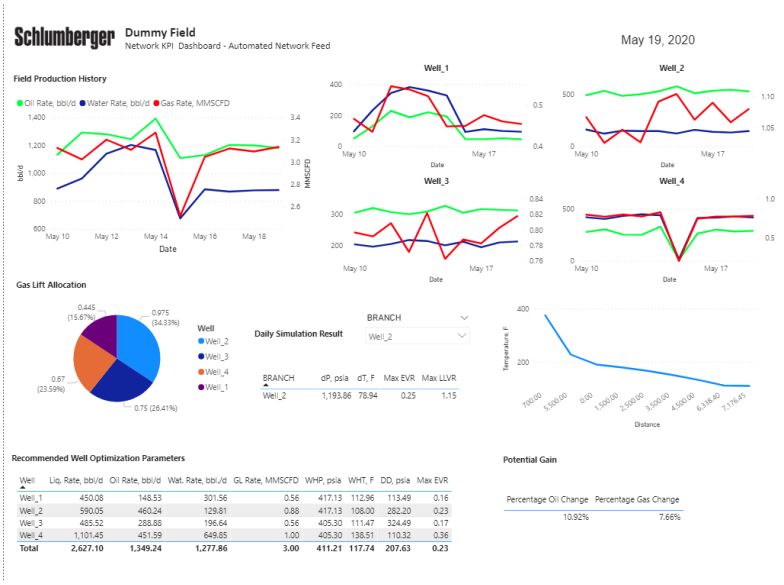


Figure 30 - Smart Advisory System PowerBI Dashboard

By having these 2 workflows in place, engineers can get insight that helps to monitor field and wells performance based on flow parameters that one could not get without running the simulation. It also acts as an advisory system to increase production and prevent or reduce downtime in a production network. Furthermore, due to the automatic nature of this workflow, time is saved, data integrity and consistency are preserved due to the elimination of well test and field data manual input. Lastly, new insights are automatically delivered to the production surveillance dashboard every day without any fingertips.

3.3.1 Machine Learning Model Construction

Machine learning models to predict the four parameters were trained and deployed. After many steps taken which discussed in the next sub-section, 3 ML models were prepared. ML model to predict Liquid PI used RFR with some feature engineering (Test score: 0.9985), while ML model to predict Holdup Factor used KNN with some feature engineering (Test score: 0.9993). Reservoir Pressure's ML model built using RFR, some feature engineering and Liquid PI (Test score: 0.9999) (it is best to predict Reservoir Pressure after Liquid PI prediction). Lastly, the author decided to drop the friction-factor prediction due to low test scores. The analysis on this is available in the next sub-sections.

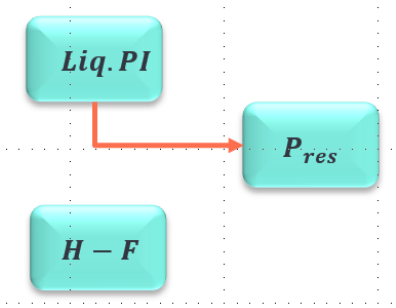


Figure 31 - ML Prediction Workflow

3.3.1.1 Dataset Generation

Nodal simulations are run with these combinations in each well (**Table 2**) which yielded 28125 rows for every well. The simulation was run with a prepared PIPESIM Python Toolkit script, and a filter was also scripted to filter-out un-converged simulation results. In total, 80003 rows of data were generated with a distribution for each well listed in **Figure 32**.

```
In [18]: 1 df['WELL'].value_counts()

Out[18]: Well_3    24378
         Well_2    21967
         Well_1    18220
         Well_4    15468
         Name: WELL, dtype: int64
```

Figure 32 - Dataset Distribution by Well

Table 2 - Dataset Generation Sensitivities

Well	WHP	Gas lift rate	Liquid PI	HF	FF	Pres	WC	Total Row
Well_1	350	0.25	4	0.8	0.8	1100	50	28125
	400	0.45	4.5	0.9	0.9	1200	65	
	450	0.65	5	1	1	1300	80	
	500		5.5	1.1	1.1	1400		
	550		6	1.2	1.2	1500		
Well_2	350	0.75	1	0.8	0.8	1200	5	28125
	400	0.95	1.5	0.9	0.9	1300	20	
	450	1.15	2	1	1	1400	35	
	500		2.5	1.1	1.1	1500		
	550		3	1.2	1.2	1600		
Well_3	325	0.55	0.5	0.8	0.8	1000	25	28125
	375	0.75	1	0.9	0.9	1100	40	
	425	0.95	1.5	1	1	1200	55	
	475		2	1.1	1.1	1300		
	525		2.5	1.2	1.2	1400		
Well_4	325	0.45	9	0.8	0.8	1200	45	28125
	375	0.65	9.5	0.9	0.9	1300	60	
	425	0.85	10	1	1	1400	75	
	475		10.5	1.1	1.1	1500		
	525		11	1.2	1.2	1600		

3.3.1.2 Liquid PI

The first step in predicting the parameters is to run AutoML from Putra, J.COp ML (2018) to obtain the best algorithm to be used to predict the parameter as well as to get the baseline model. In **Figure 32**, the best algorithm to predict Liquid PI is RFR with 0.9974 test score.

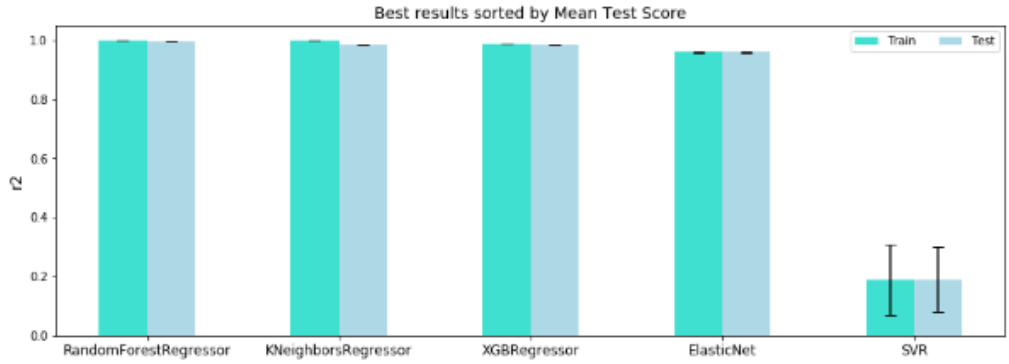


Figure 33 - Liquid PI AutoML

Then after running RFR with a more comprehensive Randomized Search, the result is better with an improved 0.9975 test score. With the model on this stage, analysis plots to do some feature engineering was drawn. From the mean score decrease plot in **Figure 33,** and normalized pearson correlation plot in **Figure 34**, the best model from feature engineering was obtained with a test score of 0.9985 and 5 number of features (Well. Liquid Rate, Water-cut, Gas Lift Injection Rate, Bottom-hole Pressure)

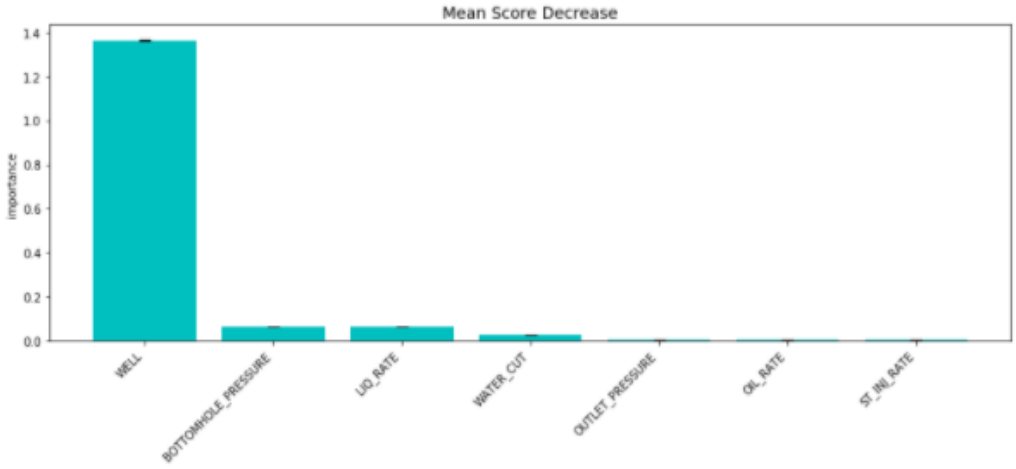


Figure 34 - Liquid PI Mean Score Decrease

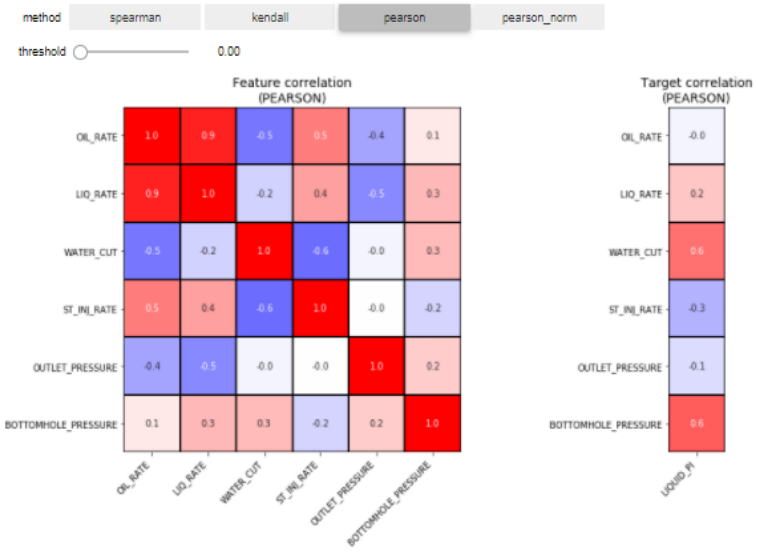


Figure 35 - Liquid PI Normalized Pearson Correlation Plot

Among all parameters, Liquid PI has the second-highest individual test score after Holdup Factor. Due to there might be a slight correlation between the two as seen from **Figure 35**, the author tried to train and test the model with the addition of Holdup Factor. All results are summarized in **Table 3**, with the best result is the one after the features are reduced and not adding the Holdup Factor feature.

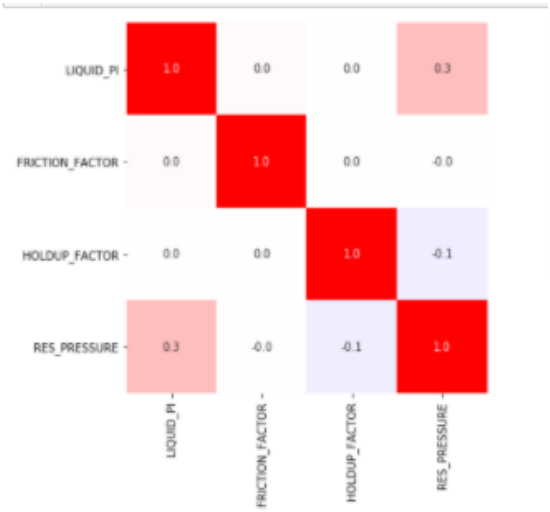


Figure 36 - Predicted Parameters Normalized Pearson Correlation Plot

Table 3 - Liquid PI ML Result Summary

Number	Order	Features	Algo	Preprocessor	Optimizer	train_R2	cv_R2	test_R2	Model Info
1	1st	Well, Qo, Ql, WC, Qinj, WHP, BHP	AutoML - RandomForestRegressor			0.999561052	0.996927017	0.997441842	max_depth: 55 max_features: 0.8184656610700978 min_samples_leaf: 1 n_estimators: 107
2	1st	Well, Qo, Ql, WC, Qinj, WHP, BHP	RandomForestRegressor		RandomSearchCV	0.999661256	0.996948785	0.99747341	max_depth: 48 max_features: 0.8219772826786358 min_samples_leaf: 1 n_estimators: 188
3	1st	Well, Ql, WC, Qinj, BHP	RandomForestRegressor		RandomSearchCV	0.999602884	0.997994891	0.998462766	max_depth: 63 max_features: 0.9446974381141753 min_samples_leaf: 2 n_estimators: 163
4	2nd (HF)	Well, Qo, Ql, WC, Qinj, WHP, BHP, HF	RandomForestRegressor		RandomSearchCV	0.999658641	0.996911166	0.997442147	max_depth: 48 max_features: 0.8219772826786358 min_samples_leaf: 1 n_estimators: 188
5	2nd (HF)	Well, Ql, WC, Qinj, BHP,HF	RandomForestRegressor		RandomSearchCV	0.999580277	0.997861787	0.998362411	max_depth: 63 max_features: 0.9446974381141753 min_samples_leaf: 2 n_estimators: 163

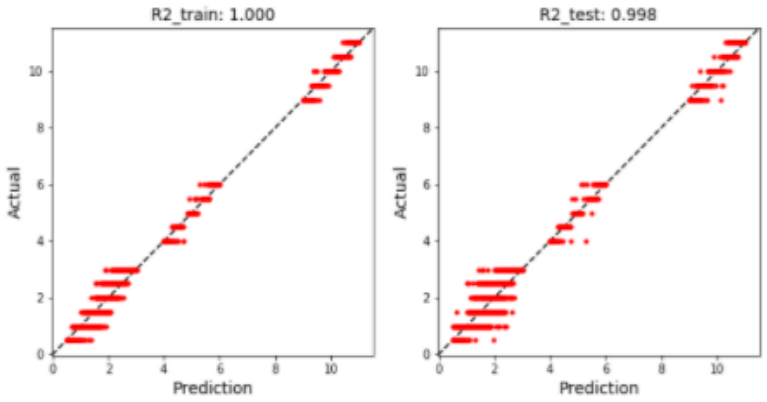


Figure 37 - Liquid PI Actual vs Prediction Plot

3.3.1.3 Holdup Factor

The same steps were applied to predict Holdup Factor. For the individual prediction, Holdup Factor has the best test score among all predicted parameters with 0.9993. The algorithm selected is KNN, as for the feature engineering, both feature reduction and feature scaling were tried because KNN is sensitive to scaling. However, as can be seen from **Table 4**, feature scaling in this case didn't improve the model, but feature reduction did. Hence the feature selected was reduced to Liquid Rate, Oil Rate, Gas Lift Injection Rate, Wellhead Pressure, and Bottom-hole Pressure to obtain the best model.

Table 4 - Holdup Factor ML Result Summary

Number	Order	Features	Algo	Preprocessor	Optimizer	train_R2	cv_R2	test_R2	Model Info
1	1st	Well, Qo, Ql, WC, Qinj, WHP, BHP	AutoML - KNeighborsRegressor			1	0.983	0.989	n_neighbors: 35 p: 1.2285500217972998 weights: distance
2	1st	Well, Qo, Ql, WC, Qinj, WHP, BHP	KNeighborsRegressor		RandomSearchCV	1	0.997	0.999	n_neighbors: 2 p: 1.3225507642386005 weights: distance
3	1st	Well, Qo, Ql, WC, Qinj, WHP, BHP	KNeighborsRegressor	num_pipe(scaling='standard', transform='yeo-johnson')	RandomSearchCV	1	0.997	0.999	n_neighbors: 2 p: 1.3225507642386005 weights: distance
4	1st	Well, Qo, Ql, WC, Qinj, WHP, BHP	KNeighborsRegressor	num_pipe(scaling='minmax')	RandomSearchCV	1	0.997	0.999	n_neighbors: 2 p: 1.3225507642386005 weights: distance
5	1st	Well, Qo, Ql, WC, Qinj, WHP, BHP	KNeighborsRegressor	num_pipe(scaling='robust')	RandomSearchCV	1	0.997	0.999	n_neighbors: 2 p: 1.3225507642386005 weights: distance
6	1st	Ql, Qo, Qinj, WHP, BHP	KNeighborsRegressor		RandomSearchCV	1	0.998	0.9993	n_neighbors: 2 p: 1.3225507642386005 weights: distance
8	1st	Ql, Qinj, WHP, BHP	KNeighborsRegressor		RandomSearchCV	1	0.947	0.95	n_neighbors: 9 p: 1.0159662522202142 weights: distance

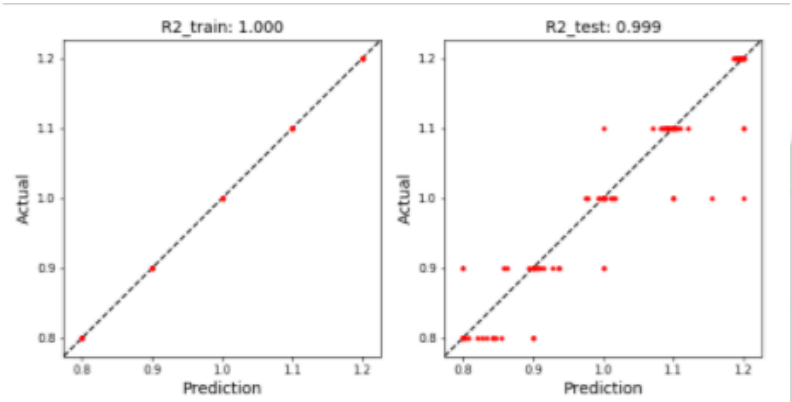


Figure 38 - Holdup Factor Actual vs Prediction Plot

3.3.1.4 Reservoir Pressure

To predict reservoir pressure, RFR is the algorithm selected. After feature engineering without the other predicted parameters as features, the best test score obtained is 0.9892 with only Well, Liquid Rate, and Bottom-hole Pressure as features. By that test score, the model is the 3rd highest among all predicted parameters. And taking **Figure 36** to consideration, the model was once again trained with the addition of Liquid PI/ Holdup Factor/ Liquid PI and Holdup Factor as additional features. The best result has 0.9999 test score with Liquid PI as an additional feature to the earlier three reduced features.

Table 5 - Reservoir Pressure ML Result Summary

Number	Order	Features	Algo	Preprocessor	Optimizer	train_R2	cv_R2	test_R2	Model Info
1	1st	Well, Qo, Ql, WC, Qinj, WHP, BHP	AutoML - RandomForestRegressor			0.99715	0.98	0.9824	max_depth: 55 max_features: 0.8184656610700978 min_samples_leaf: 1 n_estimators: 107
2	1st	Well, Qo, Ql, WC, Qinj, WHP, BHP	RandomForestRegressor		RandomSearchCV	0.99769	0.98	0.9828	max_depth: 48 max_features: 0.8219772826786358 min_samples_leaf: 1 n_estimators: 188
3	1st	Well, Ql, BHP, Qinj	RandomForestRegressor		RandomSearchCV	0.99689	0.985	0.9883	max_depth: 63 max_features: 0.9446974381141753 min_samples_leaf: 2 n_estimators: 163
4	1st	Well, Ql, BHP	RandomForestRegressor		RandomSearchCV	0.99843	0.987	0.9892	max_depth: 48 max_features: 0.8219772826786358 min_samples_leaf: 1 n_estimators: 188
5	2nd (HF)	Well, Qo, Ql, WC, Qinj, WHP, BHP, HF	RandomForestRegressor		RandomSearchCV	0.99765	0.98	0.9827	max_depth: 48 max_features: 0.8219772826786358 min_samples_leaf: 1 n_estimators: 188
6	2nd (HF)	Well, Ql, BHP, HF	RandomForestRegressor		RandomSearchCV	0.99677	0.985	0.9878	max_depth: 63 max_features: 0.9446974381141753 min_samples_leaf: 2 n_estimators: 163
7	2nd (LP)	Well, Ql, BHP, LP	RandomForestRegressor		RandomSearchCV	0.99999	1	0.9999	max_depth: 48 max_features: 0.8219772826786358 min_samples_leaf: 1 n_estimators: 188
8	3rd (LP +	Well, Ql, BHP, HF, LP	RandomForestRegressor		RandomSearchCV	0.99999	1	0.9999	max_depth: 48 max_features: 0.8219772826786358 min_samples_leaf: 1 n_estimators: 188

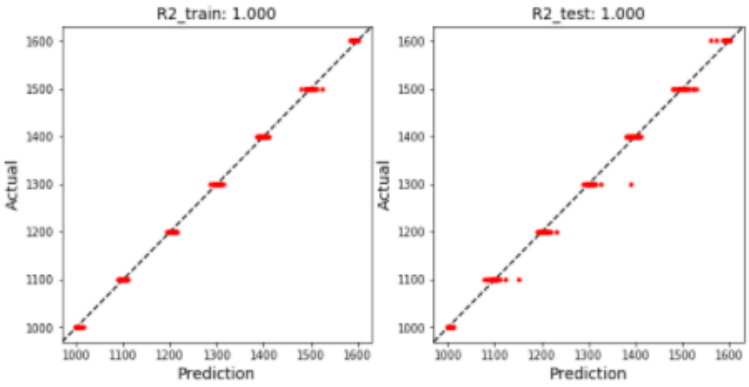


Figure 39 - Reservoir Pressure Actual vs Prediction Plot

3.3.1.5 Friction Factor

As we can see from **Table 6**, the test scores for Friction Factor prediction have no value above 0.3. The trials that have been conducted are AutoML, feature engineering (reducing features decreased the test score in this case), and all possible re-arrangements of the prediction (added Liquid PI/ Holdup Factor/ Reservoir Pressure/ Liquid PI and Holdup Factor/ Liquid PI and Reservoir Pressure/ Holdup Factor and Reservoir Pressure/ All three as features) although from **Figure 36** Friction Factor doesn't correlate with other predicted parameters.

Table 6 - Friction Factor ML Result Summary

Number	Order	Features	Algo	Preprocessor	Optimizer	train_R2	cv_R2	test_R2	Model Info
1	1st	Well, Qo, Ql, WC, Qinj, WHP, BHP	AutoML - RandomForestRegressor			0.84331	0.166	0.2531	max_depth: 55 max_features: 0.8184656610700978 min_samples_leaf: 1 n_estimators: 107
2	1st	Well, Qo, Ql, WC, Qinj, WHP, BHP	RandomForestRegressor		RandomSearchCV	0.83834	0.166	0.2531	max_depth: 55 max_features: 0.8184656610700978 min_samples_leaf: 1 n_estimators: 107
3	1st	Qo, Ql, WHP, BHP	RandomForestRegressor		RandomSearchCV	0.82587	0.148	0.2279	max_depth: 55 max_features: 0.8184656610700978 min_samples_leaf: 1 n_estimators: 107
4	2nd (HF)	Well, Qo, Ql, WC, Qinj, WHP, BHP, HF	RandomForestRegressor		RandomSearchCV	0.85158	0.17	0.2628	max_depth: 55 max_features: 0.8184656610700978 min_samples_leaf: 1 n_estimators: 107
5	2nd (LP)	Well, Qo, Ql, WC, Qinj, WHP, BHP, LP	RandomForestRegressor		RandomSearchCV	0.84905	0.17	0.2631	max_depth: 55 max_features: 0.8184656610700978 min_samples_leaf: 1 n_estimators: 107
6	2nd (RP)	Well, Qo, Ql, WC, Qinj, WHP, BHP, RP	RandomForestRegressor		RandomSearchCV	0.84353	0.167	0.2586	max_depth: 55 max_features: 0.8184656610700978 min_samples_leaf: 1 n_estimators: 107
7	3rd (HF +	Well, Qo, Ql, WC, Qinj, WHP, BHP, HF, LP	RandomForestRegressor		RandomSearchCV	0.85423	0.169	0.2723	max_depth: 55 max_features: 0.8184656610700978 min_samples_leaf: 1 n_estimators: 107
8	3rd (HF +	Well, Qo, Ql, WC, Qinj, WHP, BHP, HF, RP	RandomForestRegressor		RandomSearchCV	0.85523	0.169	0.2652	max_depth: 55 max_features: 0.8184656610700978 min_samples_leaf: 1 n_estimators: 107
9	3rd (LP +	Well, Qo, Ql, WC, Qinj, WHP, BHP, LP, RP	RandomForestRegressor		RandomSearchCV	0.85003	0.168	0.2651	max_depth: 55 max_features: 0.8184656610700978 min_samples_leaf: 1 n_estimators: 107
10	4th (HF +	Well, Qo, Ql, WC, Qinj, WHP, BHP, HF, LP,	RandomForestRegressor		RandomSearchCV	0.85923	0.17	0.2728	max_depth: 55 max_features: 0.8184656610700978 min_samples_leaf: 1 n_estimators: 107

To analyze the cause of this, a nodal analysis of a well in the network model is run. The result was as expected, the total frictional pressure drops of the well which directly affected by the Friction Factor is very low (6.6 psi) compared to the elevational pressure drops (1025.424 psi) which is tuned by the Holdup Factor. Thus, for all the wells in this dummy network, the effect of Friction Factor variation is almost negligible which makes the Machine Learning cannot recognize a pattern from changing Friction Factor. Hence for this study, Friction Factor prediction is excluded.

Select columns...

	Operating point	ST Liq. at NA	P at NA	TOT INJ Gas	Tot. Elev. DP	Tot. Frict. DP
		STB/d	psia	mmscf/d	psi	psi
1	Flowrate=456....	456.0464	1431.666	0.445	1025.424	6.625658

Figure 40 - Well_1 Nodal Analysis Result

4 Conclusion

Therefore, to answer all the objectives mentioned earlier, this study can be concluded:

1. The four automated workflows minimize risk for data quality issues from human errors as it replaces many steps of manual data input
2. All the workflows cut down the time consumed. As from the first and second steps, time consumed was reduced from 15 minutes/well to 3 seconds/well and from 30 minutes to 2 minutes for the network model merge with the assumption of using the dummy model. For the last 2 workflows, data gathering, manual input, running the simulation and bring the result to visualization would take 30 minutes each day, with the automated workflows, those processes are reduced to 2 minutes for each run without any human effort.
3. In the Auto-Calibration workflow, the feature importance in building machine learning workflow allow us to see that a field has its own feature importance, and it will be varied depend on the field's condition. For this case study, the critical value to be calibrated is Holdup Factor, Liquid PI and reservoir pressure. For other fields, especially gas fields, the friction factor may be an important feature to build the machine learning model for calibration.
4. With the capability and extensibility of Python and PIPESIM Python Toolkit API, the solutions are extended to:
    a. Daily updated production surveillance dashboard which insight consists of flow parameters and operation recommendation to maximize oil rate
    b. Extraction of Ambient Temperature data from a weather service API
    c. Auto-calibration Machine Learning workflow with >95% accuracy

5 Recommendation

To enhance the workflows, these several points are recommended:

1. Make the Automatic Network Merge script to be an import plugin within PIPESIM
2. Add field-level features to the Auto-calibration ML model input
3. For the Auto-calibration ML to be adopt to other field settings, the building steps needs to be re-done as different field has different characteristic and feature importance
4. Explore other ML result enhancement methods:
    a. Polynomial features generation
    b. Binning for numerical features
    c. Use deep learning algorithms
5. Augment the integration with reservoir simulation model that allows network coupling to obtain a digital twin from reservoir to production system.

6 Nomenclature

API	Application Programming Interface
BONMIN	Basic Open-Source Mixed Integer
CV	Cross-Validation
EOT	End of Tubing

ESP	Electrical Submersible Pump
EVR	Erosional Velocity Ratio
FGS	Flowing Gradient Survey
GOR	Gas-Oil-Ratio
GUI	Graphical User Interface
I/ O	Input/ Output
IPR	Inflow Performance Relationship
KNN	K-Nearest Neighbor
KPI	Key Performance Indicator
LLVR	Liquid Loading Velocity Ratio
MINLP	Mixed Integer Non-Linear Programs
ML	Machine Learning
MMscf/D	Million Standard Cubic Feet per Day
PCP	Progressing Cavity Pump
PDMS	Production Database Management System
PI	Productivity Index
psi	pounds per square inch
RFR	Random Forest Regression
SDK	Software Development Kit
SQL	Standard Query Language
STB/D	Standard Barrel per Day
SVR	Support Vector Regression
WC	Water-Cut
XGBoost	Extreme Gradient Boosting

7 References

[1] Brown, K. E. (1984). *The Technology of Artificial Lift Methods.* Tulsa, Oklahoma: Penwell Books.

[2] Guo, B., Lyons, W. C., & Ghalambor, A. (2006). *Petroleum Production Engineering A Computer Assisted Approach*. Lousiana: Guld Professional Publishing.

[3] Hammadi, B., Agoudjil, K., Fahem, A., Tadjine, F., Benabdellah, H., Makhloufi, A., . . . Younsi, T. (2020). Unlocking the Production Potential of Brown Fields through Gas Lift Optimization GLO. *International Petroleum Technology Conference.* Dhahran, Saudi Arabia.

[4] Irfan, M., Vidrianto, M., Rahmawati, S. D., & Naufal, A. A. (2020). A Breakthrough Approach for Predicting ESP Wells Virtual Flow Rate by Using Supervised Machine Learning Method. *IATMI Symposium.*

[5] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R.* New York: Springer Science+Business Media.

[6] Putra, W. D. (2018, July). *J.COp ML*. Retrieved April 2020, from PyPI: https://pypi.org/project/jcopml/

[7] Putra, W. D. (2018, June). *Supervised Learning by J.COp*. Retrieved April 2020, from github: https://github.com/WiraDKP/supervised_learning

[8] Putra, W. D. (2019, March). *LuWiji*. Retrieved April 2020, from PyPI: https://pypi.org/project/luwiji/

[9] Schlumberger. (2019). PIPESIM Online Help Version 2019.3.

[10] Schlumberger. (2019). PIPESIM Python Toolkit 2019.2 Documentation.