2023

Hydraulic Fracturing Treatment Optimization Using Machine Learning

Abdullah Johar
aj0055@mix.wvu.edu

Recommended Citation

Johar, Abdullah, "Hydraulic Fracturing Treatment Optimization Using Machine Learning" (2023). *Graduate Theses, Dissertations, and Problem Reports*. 11697.
https://researchrepository.wvu.edu/etd/11697

Hydraulic Fracturing Treatment Optimization Using Machine Learning

Abdullah Johar

Thesis submitted

to the Benjamin M. Statler College of Engineering and Mineral Resources

at West Virginia University

In partial fulfillment of the requirements for the degree of

Master of Science

In

Petroleum and Natural Gas Engineering

Ebrahim Fathi, Ph.D., Chair

Samuel Ameri, Ph.D.

Kashy Aminian, Ph.D.

Department of Petroleum and Natural Gas Engineering

Morgantown, West Virginia

2023

Keywords: Marcellus Shale, Boggess, Machine Learning, Hydraulic Fracture Treatment, Optimization, Artificial Intelligence

1- ABSTRACT

Hydraulic Fracturing Treatment Optimization Using Machine Learning

Abdullah Johar

Friction reducer (FR) is a chemical additive that is used in hydraulic fracturing operations to reduce friction between the fracturing fluid and the walls of the wellbore in order to overcome the tubular drag while pumping at a high flow rate in stimulation treatments. In recent years, the oil and gas industry tend to use a high viscosity Friction reducer (HVFR) in fracturing fluids for operational and economic reasons. Slick water fracturing fluids have low concentrations of friction reducers. The concentration of the FR used in shale gas reservoirs varies between 0.5 and 2.0 gallons per thousand gallons (gpt). However, the optimum FR concentration required for each stage is not extensively studied. Most of the oil and gas companies are relying on previous practices and they generally use more FR concentrations than required simply to make sure they can achieve the designed injection rate and avoid screening out. This resulted in excess FR concentration used in stimulation practices that will result in major economic loss. Excess FR means the amount of FR that is beyond required, which is only a waste of money.

The primary goal of this proposal is to fully comprehend and quantify the performance of the Hydraulic Fracturing Friction Reducer in a wide range of concentrations used in the completion of eight horizontal Marcellus Shale wells, as well as to optimize the amount of friction reducer required to complete the job and reduce the cost associated with it. Several machine learning approaches will be employed, and data from over 400 hydraulic fracturing stages will be evaluated to optimize the amount of friction reducer. Finally, an economic analysis will determine the project's net present value (NPV) and accumulated savings of FR cost per stage.

Data collected from Boggess pad from the MSEEL project including Boggess 1H, 3H, 5H, 9H, and 17H will be used. The collected data was the one second reading for the completion process for each stage from MSEEL project website.

ACKOWLEDGEMENTS:

And Say: "My Lord, increase me in knowledge." Surat Taha.

I would like to begin by expressing my thanks to God for allowing me to pursue this research topic and successfully complete my thesis. I would like to thank my research advisor and professor Dr. Ebrahim Fathi for giving me the chance to work with him and for providing the support during my research period. I would like to extend my appreciation to my committee members, Prof. Samuel Ameri and Dr. Kashy Aminian, they have been so helpful to me and I did not feel that I am away from my home country thanks to all committee members, whenever I needed any help they were always there for me. Many thanks to my family members and my friends who supported me through everything I have faced in school and life.

TABLE OF CONTENTS

Contents

List of Figure

List of Tables

2-Introduction:

Hydraulic fracturing, which can also be called fracking, indicates a process that is used in the oil and gas industry. It is mainly used to extract hydrocarbons from extremely tight underground formations. In the process of hydraulic fracturing, a high-pressure fluid is pumped into the formations to create fractures. This allows hydrocarbon fluid to move to the wellbore and becomes extractable.

An issue that can be raised when using this process is that the fracturing fluid that is used in this method will cause significant friction with the casing and formation, which leads to loss in pressure and fracturing energy and reduced efficiency of the stimulation. This causes productivity to decrease. To fix this issue, a set of friction reducers can be added to the fracking fluid, to decrease or friction pressure loss, increasing efficiency and further optimizing the process.

In the framework of PNGE (Petroleum and Natural Gas Engineering), friction reducer optimization is a procedure that chooses and optimizes friction reducers in hydraulic fracturing operations to further decrease friction and, at the same time, increase the efficiency of the well. This might require the testing of a variety of friction reducers while also trying to find what's the best concentration of the friction reducers within the fluid. It might also require the optimization of the flow rate to be pumped. Pressure is also required to be tested to find out the best pressure required to maximize the effectiveness of the fluid while also reducing the risk of damaging the formation.

In general, the optimization of friction reducers is a mandatory aspect of hydraulic fracturing operations in PNGE and can help maximize the retrieval of hydrocarbons from underground formations. Having said so, this project is mainly focused on the optimal use of friction reducers to accomplish the job and decrease the cost of it by using Artificial intelligence (AI) and Machine Learning techniques (ML).

Artificial Intelligence (AI) and Machine Learning (ML) are being used more often in the oil and gas industry to maximize the efficiency of many processes. This also includes hydraulic fracturing and the use of friction reducers.

In the setting of hydraulic fracturing, AI and ML can help us optimize the use of friction reducers by pinpointing the best concentrations and combinations of friction reducers from previous or current data that was used in previous wells. This provides information to reduce the costs of the overall process, maximize the efficiency and potential of the hydraulic fracturing process, and reduce environmental hazards.

AI and ML are very useful, and they help optimize friction reducers in the oil and gas industry. They also help maximize the efficiency and effectiveness of hydraulic fracturing operations while minimizing the costs and environmental risks. Furthermore, these technologies will help us immensely as they continue to improve in the future.

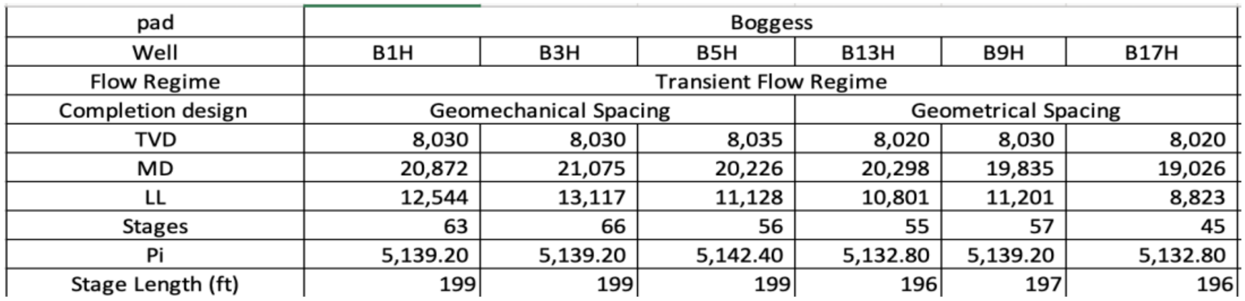
In this project, AI and ML will be applied to the Marcellus shale, i.e., Boggess pad, that contain real one seconds time data to optimize the friction reducers and chemical requirements in the well and reduce the cost spent on the well.

The Marcellus Shale Energy and Environment Laboratory (MSEEL) project is a project that is conducted by the U.S. Department of Energy's National Energy Technology Laboratory (NETL) associated with several other organizations, including West Virginia University, Ohio State University, and Northeast Natural Energy. The project launched in 2015 and studied the environmental risks of natural gas development in the Marcellus Shale formation, which covers several other states in the Northeastern United States.

The MSEEL project performed on two pads including MIP wells and BOGGESS wells. This project will focus on BOGGESS wells to perform the optimization. The wells are known as follows:

- Boggess 1H
- Boggess 3H
- Boggess 5H
- Boggess 9H
- Boggess 13H
- Boggess 17H

The following Table 1 shows the summary of drilling and completion characteristics of the 6 wells in Boggess pad along with the flow regimes identified using decline curve analysis and type of completion design.



pad	Boggess					
Well	B1H	B3H	B5H	B13H	B9H	B17H
Flow Regime	Transient Flow Regime					
Completion design	Geomechanical Spacing			Geometrical Spacing		
TVD	8,030	8,030	8,035	8,020	8,030	8,020
MD	20,872	21,075	20,226	20,298	19,835	19,026
LL	12,544	13,117	11,128	10,801	11,201	8,823
Stages	63	66	56	55	57	45
Pi	5,139.20	5,139.20	5,142.40	5,132.80	5,139.20	5,132.80
Stage Length (ft)	199	199	199	196	197	196

Table 1 Boggess well analysis summary

Well Name	Number of stages
Boggess-1H	63
Boggess-3H	66
Boggess-5H	56
Boggess-9H	57
Boggess-13H	55
Boggess-17H	45

Table 2 Boggess wells number of stages

The following figures1,2,3,4,5, and 6 show the gas rate and cumulative gas production from Boggess wells started in November 2019 till June 2021 that was used for this study.

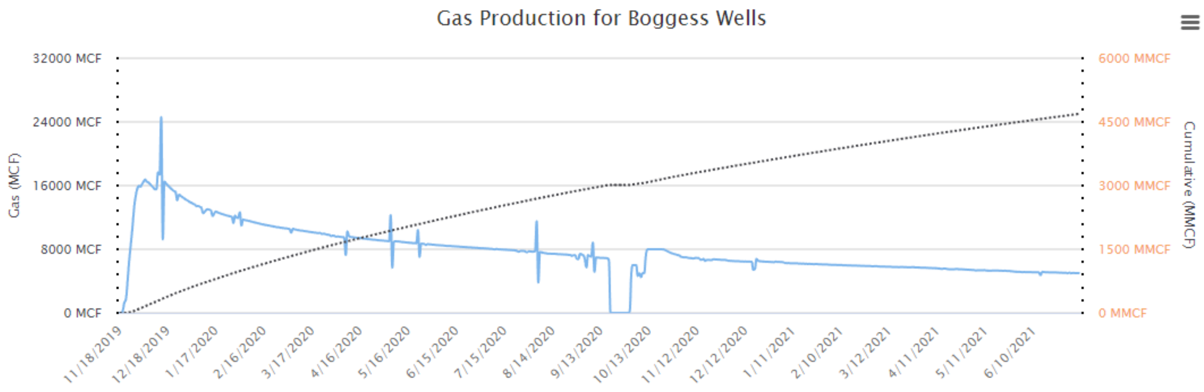


Figure 1 Boggess - 1H production

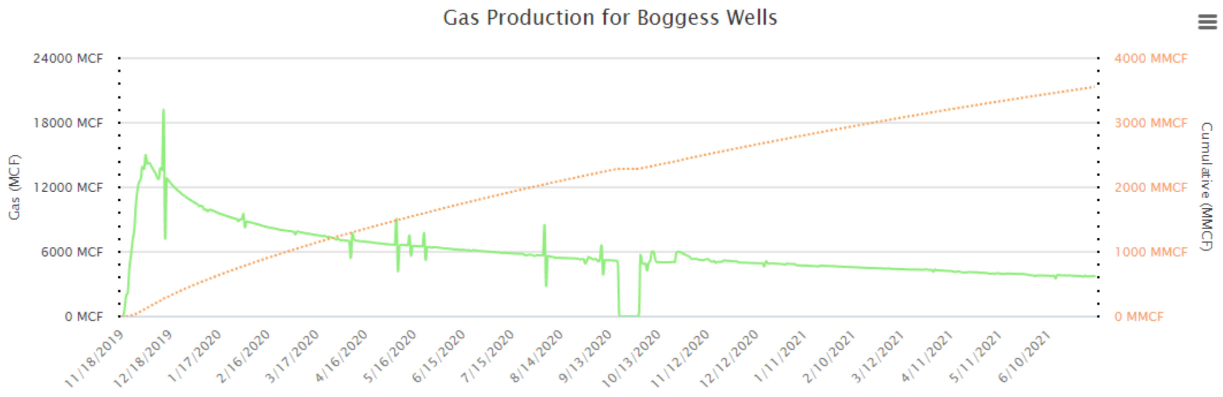


Figure 1 Boggess - 3H production

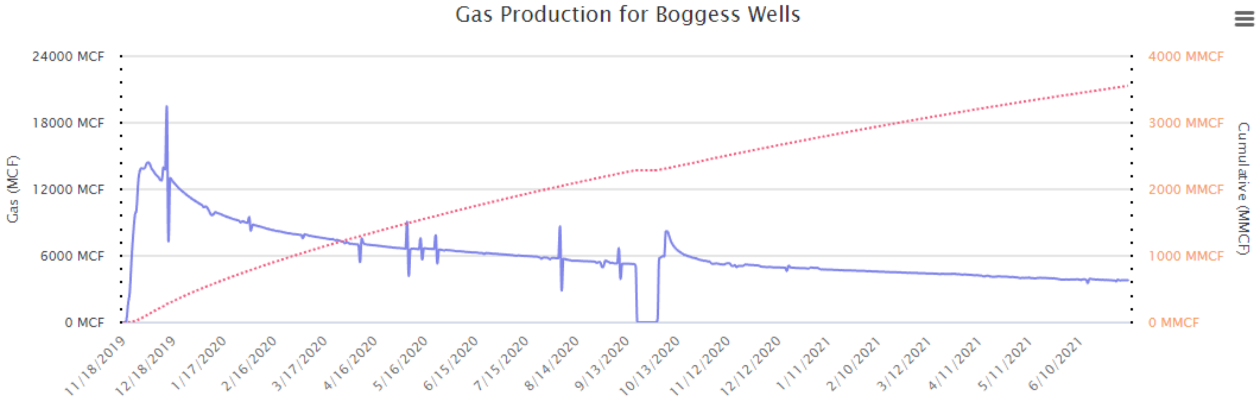


Figure 3 Boggess - 5H production

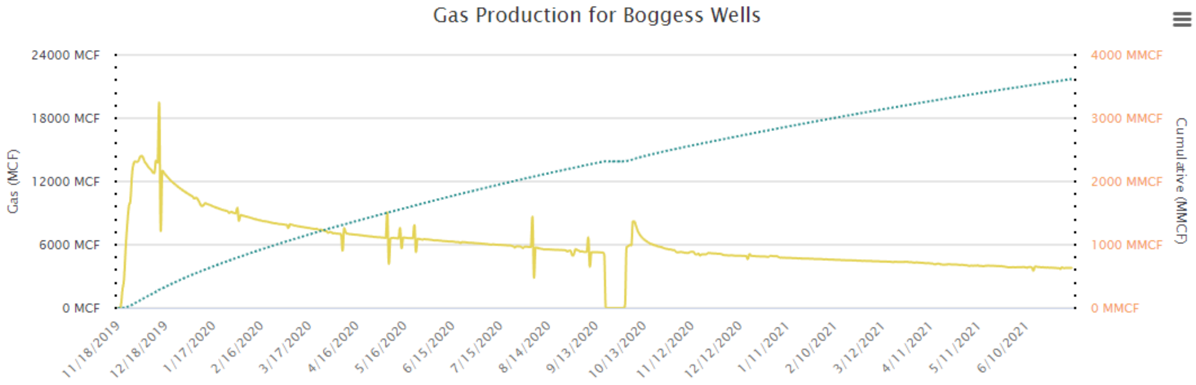


Figure 4 Boggess - 9H production

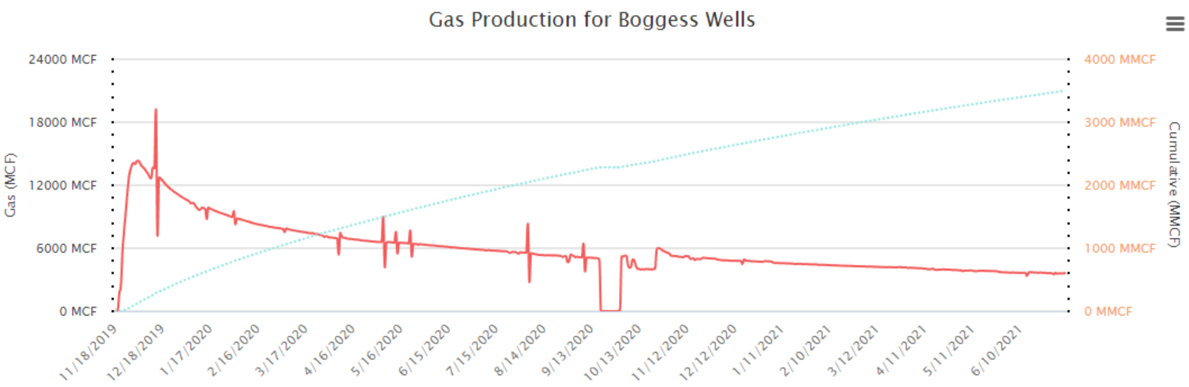


Figure 5 Boggess - 13H production

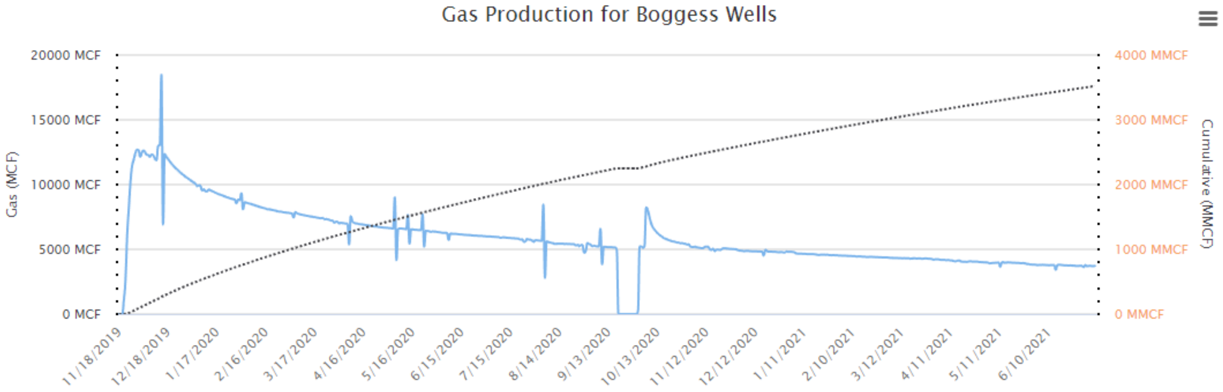


Figure 6 Boggess - 13H production

The following list of stages are the problematic stages figure### that the MSEEL project has mentioned in their quarterly report of 2021 Q1, these stages will also be applied in our model in this paper.

The stages are the following:

1H-14, 1H-20,

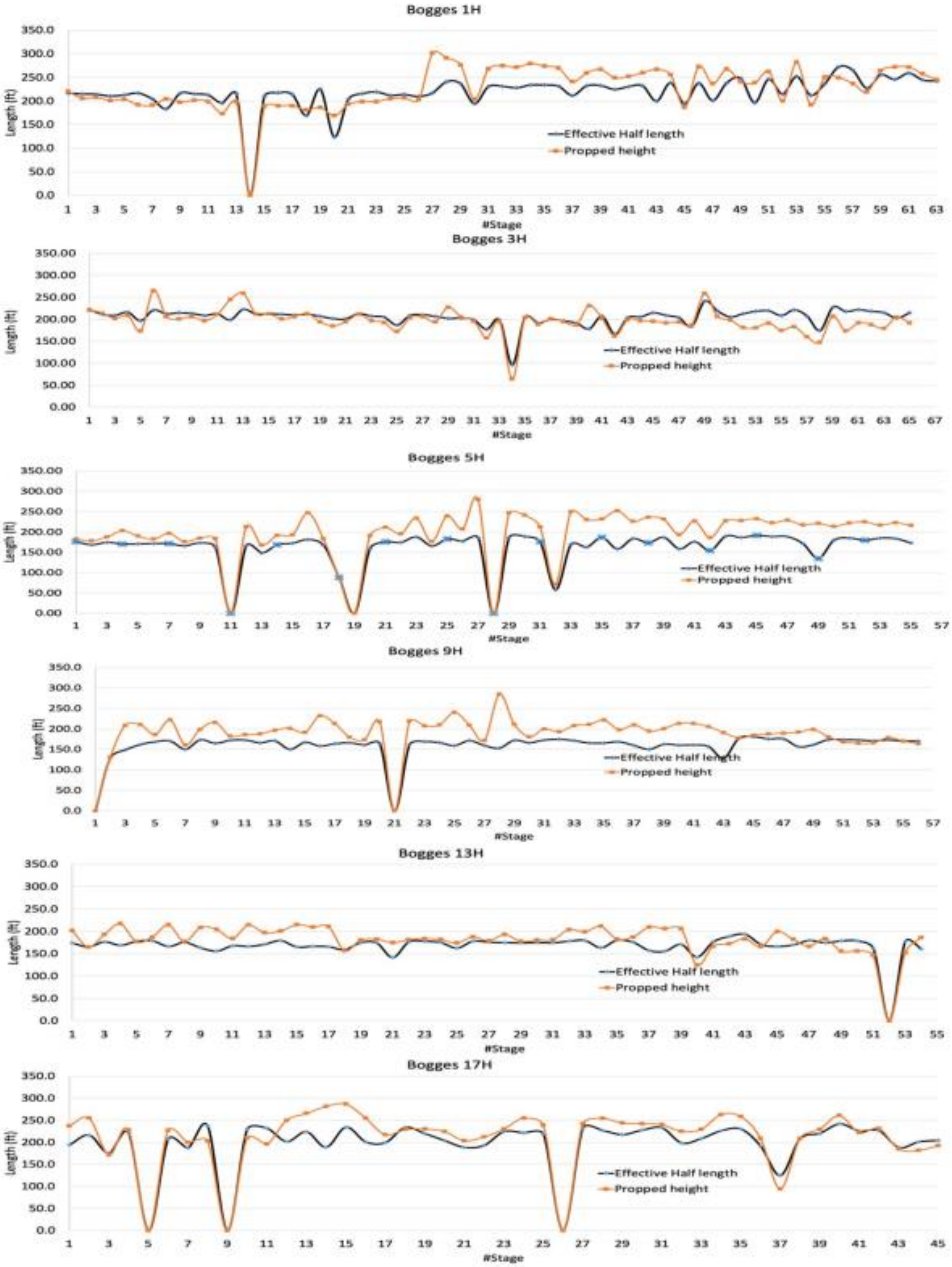
3H-34,

5H-11, 5H-19, 5H-28, 5H-32,

9H-21,

13H-52,

17H-5, 17H-9, 17H-26


Bogges 1H

Bogges 3H

Bogges 5H

Bogges 9H

Bogges 13H

Bogges 17H

Figure 7 Quarterly report problematic stages.

3-Project Overview:

To complete the project, there are multiple stages and steps to go through. First, data needs to be collected (i.e., Data Collection) and then preprocessed (i.e., Data Pre-processing and Visualization). Then model development comes into play to find the best ML model that can history match and predict the pressure behavior while hydraulic fracturing. The ML model then will be used to predict the optimum friction reducer and chemicals that need to be injected.

The workflow to complete the project is as follows:

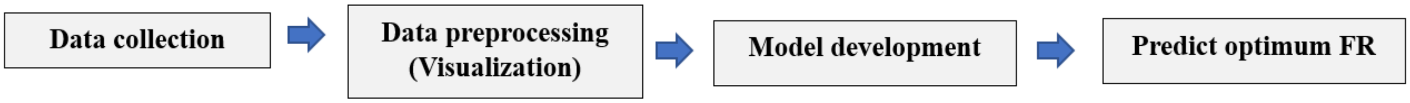


Figure 8 Project workflow

3.1 – Data collection:

In AI and ML, gathering data indicates the process of collecting a vast amount of data which is then used to teach machine training models or to enhance the performance of recent models. Furthermore, Data gathering is an essential part of AI and ML because they depend on vast amounts of data to learn and predict more accurately while making correct decisions.

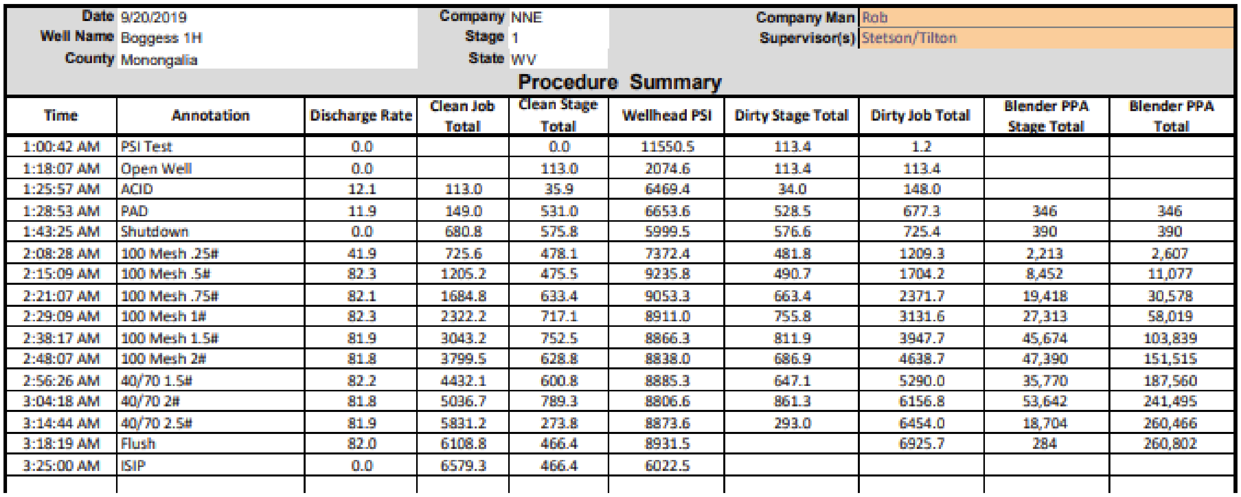
For this project, the data has been collected from the MSEEL project website (i.e., www.mseel.org). Data associated with 6 wells in Boggess pad is collected from different formats. As an example, Boggess 1H stimulation data includes a one-second real-time data that is on a CSV file, with a total of 63 files each for one stage. The file includes all the information on a second basis regarding pressure, rate and any additives including chemicals, sands, and. FR.

3.2 – Data Preprocessing:

Data preprocessing is a crucial part in data analysis and machine learning algorithms. The collected data is prepared and then converted to be used as an input for the learning algorithms.

3.2.1- Data Visualization: The initial step to cleanse the data is to look at it and manifest it. This occurs by plotting the data to visualize and see any unnecessary inclinations in the database, which could occur due to a human or instrument error.

For example, the one second data recorded for the Boggess 1H at stage 1 is plotted in Figure 10. The data is following the sand schedule presented in figure 9 that includes the pressure test, Acid injection to clean up the perforations, injecting Pad to create the fracture following with 100 mesh and 40/70 mesh and finally flushand shut-in the injection.



Date	9/20/2019	Company	NNE	Company Man	Rob
Well Name	Boggess 1H	Stage	1	Supervisor(s)	Stetson/Tilton
County	Monongalia	State	WV		

Procedure Summary

Time	Annotation	Discharge Rate	Clean Job Total	Clean Stage Total	Wellhead PSI	Dirty Stage Total	Dirty Job Total	Blender PPA Stage Total	Blender PPA Total
1:00:42 AM	PSI Test	0.0		0.0	11550.5	113.4	1.2		
1:18:07 AM	Open Well	0.0		113.0	2074.6	113.4	113.4		
1:25:57 AM	ACID	12.1	113.0	35.9	6469.4	34.0	148.0		
1:28:53 AM	PAD	11.9	149.0	531.0	6653.6	528.5	677.3	346	346
1:43:25 AM	Shutdown	0.0	680.8	575.8	5999.5	576.6	725.4	390	390
2:08:28 AM	100 Mesh .25#	41.9	725.6	478.1	7372.4	481.8	1209.3	2,213	2,607
2:15:09 AM	100 Mesh .5#	82.3	1205.2	475.5	9235.8	490.7	1704.2	8,452	11,077
2:21:07 AM	100 Mesh .75#	82.1	1684.8	633.4	9053.3	663.4	2371.7	19,418	30,578
2:29:09 AM	100 Mesh 1#	82.3	2322.2	717.1	8911.0	755.8	3131.6	27,313	58,019
2:38:17 AM	100 Mesh 1.5#	81.9	3043.2	752.5	8866.3	811.9	3947.7	45,674	103,839
2:48:07 AM	100 Mesh 2#	81.8	3799.5	628.8	8838.0	686.9	4638.7	47,390	151,515
2:56:26 AM	40/70 1.5#	82.2	4432.1	600.8	8885.3	647.1	5290.0	35,770	187,560
3:04:18 AM	40/70 2#	81.8	5036.7	789.3	8806.6	861.3	6156.8	53,642	241,495
3:14:44 AM	40/70 2.5#	81.9	5831.2	273.8	8873.6	293.0	6454.0	18,704	260,466
3:18:19 AM	Flush	82.0	6108.8	466.4	8931.5		6925.7	284	260,802
3:25:00 AM	ISIP	0.0	6579.3	466.4	6022.5				

Figure 9 sand schedule, Boggess 1H-1.

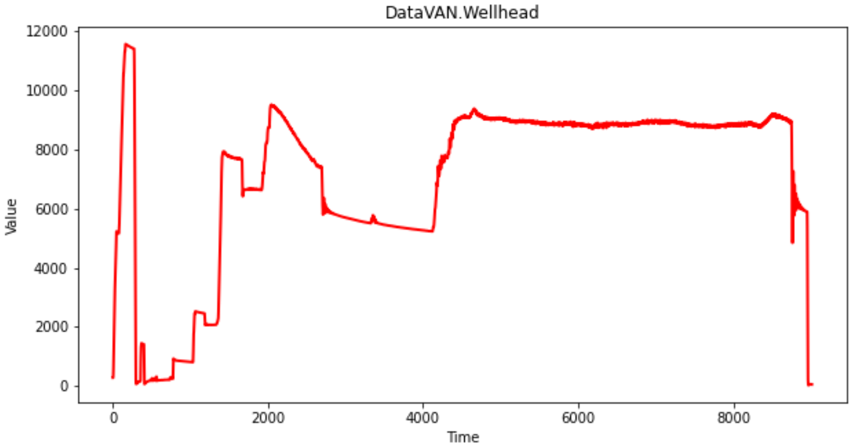


Figure 10 wellhead pressure generated plot, Boggess 1H-1.

3.2.2- Data imputation: It is a crucial step to be done to prepare the data for Machine Learning (ML). Data imputation is known as an array of procedures that occur to replace missing data in the data set with an approximation value gained from other available data points. Having missing data is very common due to recording or measurement failure or presence of outliers in the data. The whole data set is investigated for missing data and if the missing data is less than 20% the data has been imputed using K-Nearest Neighbor Imputation (KNN).

3.2.3- Dropping NA: It is crucial to cleanse the data from any NA values or filling the NA values. This can also happen due to measurement or recording failure or human error.

3.2.4- Drop duplicates from data frame*:* When the data frame is created, any clone or duplicate data points due to measurement or recording failure or human error will be eliminated to get the best results.

3.2.5- Outlier Detection: Outlier detection algorithms are mainly used to find outliers within the data set. Outliers are deviations that appear in a data set that are unlike the usual data points, or just simply being different from the rest of the data points. K-means clustering is an example of it. Outlier detection deviates from visualization. Outlier detection happens by automatic detection, while visualization is comprehension by vision. After detecting the outliers or anomalies in the data to make sure that is due to measurement or recording failure or human error and not a a extreme or limiting case studies or older design the outlier will be investigated to make sure physical plausibility of data before any decision on keeping or eliminating the data been made.

After understanding ML techniques, finding the optimum FR and model development will be discussed in the methodology section of this work. One of the more important aspects and outcomes of this project is to look out and detect Screen outs. Screen out defines as the occurrence that happens while drilling or hydraulic fracturing where the pressure from drilling or the fracturing fluid does not carry enough strength to overcome the formation pressure which leads to formation solids breaching into the wellbore. This outcome is unwanted because it can cause damage, as well as reducing efficiency and productivity. It might also cause malfunctions and damage to the equipment which will require expensive repairs. Reduced productivity can happen due to incomplete frack job due to screen out resulting in decrease in the flow rate of hydrocarbons. Screen out can be seen as formation damage that is defined as formation pores being clogged causing severe damage and reduced porosity. The erosion and corrosion of the wellbore casing can also happen resulting in wellbore damages by breakages and leaks.

Screen out damages can be severe and hefty, both on the equipment, wellbore, and the economy. Depending on the severity of the damage, the screen out cost can add up to $150000 per screen out. These include many variables, an example of that is repairing the equipment, redrilling, and loss of production. The equipment repairs can be costly as well. Another example of this is wellbore cleaning or what is known as wellbore cleanouts, that is expensive too. The damage of the screen out caused on the formation might need remedial action such as acidizing or hydraulic fracturing, which can be extremely expensive.

Screen out prevention needs to take place with utmost efficiency. Preventing screen outs can happen by knowing why and how they occur and planning in case something happens. Consistent monitoring and maintenance need to take place for the drilling to prevent any risks or hazards.

4-Literature review:

4.1 Friction Reducers:

Hydraulic fracturing is crucial to the petroleum industry to maximize oil and gas recovery from unconventional reservoirs. A notable reduction in the efficiency of a well stimulation can be caused by the friction between the fracturing fluid and the wellbore. Chemical additives called Friction reducers are applied in hydraulic fracturing operations to decrease friction and enhance fluid flow. Many studies have been conducted to explore the impacts of using friction reducers and to analyze their effects on well productivity in the petroleum industry.

A study that investigated the impacts of using friction reducers in hydraulic fracturing operations was carried out by Fathi et al. (2015). It was discovered that the application of friction reducers showed a substantial reduction in pumping pressure and overall, better well productivity. An additional study by Zolfaghari et al. (2016) had examined different types of friction reducers and their effects on hydraulic fracturing operations. This study had noted that a significant reduction of friction can be observed when friction reducers are used, and that the properties of the formation are essential when deciding the desired concentration of friction reducer to be used.

A study conducted by Gao Et Al. (2017) observed that a reduction in pressure caused by the use of friction reducers during hydraulic fracturing had led to an overall improvement in well productivity. Another study done by Wang et al. (2018) looks into the impact of using friction reducers on the hydraulic fracturing of shale formations. This study noted that a significant pressure drop is caused by the use of friction reducers in such operations which in turn had led to an increase in productiveness of the well and an improvement in fracture conductivity.

Friction reducers' effect on the environment was explored in a study by Peng et al. (2016). The study looked into the possible negative effects of using friction reducers in hydraulic fracturing operations on the environment. It was concluded in the study that certain specific friction reducers are indeed damaging to the environment, and that the application of environmentally safe friction reducers is more recommended to avoid undesirable results.

Friction reducers are essential for reducing friction and enhancing the flow of fluid in hydraulic fracturing, both of which contribute to increased well productivity. According to multiple studies, we can observe that friction reducers have been shown to significantly reduce pumping pressure, increase fracture conductivity, and well productiveness. The properties of formation determine the best concentration of friction reducer to be used. Additionally, the environmental impact of friction has been investigated, and it was recommended to use friction reducers that are environmentally safe. In the petroleum industry, friction reducers are a crucial part of hydraulic fracturing operations. When used and applied properly, they can improve well performance and lead to increases in profitability. Details of flow loop measurement to quantify the quality of FR can be obtained in Hydraulic fracturing in unconventional reservoirs: theories, operations, and economic analysis published by belyadi et al. 2016 and 2019.

4.2 Artificial intelligence and machine learning:

Due to their capacity to offer insights and optimize operations, artificial intelligence (AI) and machine learning (ML) have grown in popularity in the petroleum and natural gas engineering (PNGE) sector and much research has been conducted to analyze how these tools can be used to improve results and optimize operations in the PNGE field.

Research conducted by Wang et al. (2019) looked into the application of ML methods to PNGE well performance prediction. According to the research, ML models were highly accurate at predicting performance, which helped with decision-making in PNGE operations. In a different study by Wang et al. (2021), the drilling operations in PNGE were optimized by using AI and ML techniques. The study found that using AI and ML techniques significantly decreased the time and cost associated with drilling operations. Lashari et al 2019 use neural network to develop ML model for accurate and real-time drilling performance monitoring and optimization.

The application of AI and ML techniques to reservoir modeling was examined in a study by Liu et al. (2020). The study discovered that the application of AI and ML techniques improved the ability to predict the rates of production and produced reservoir models that were more precise and accurate. Similar to this, a study done by Zhao et al. (2020) investigated the use of AI and ML methods in hydraulic fracturing design. The study concluded that the use of AI and ML has resulted in overall better well performance and more accurate predictions of hydraulic fracture geometry.

Studies were conducted around the use of AI and ML to optimize maintenance operations. One such study was conducted by Jianf et al. (2019) which employed AI and ML methods to improve the drilling equipment maintenance schedule. According to the study, using AI and ML methods can notably decrease maintenance costs and downtime. The new concept of automated machine learning is applied in oil and gas industry by Fathi et al 2022 for High-quality fracture network mapping using high frequency logging while drilling (LWD) data. The same group published multiple studies on application of automated machine learning for near wellbore fracture mapping, and evaluation of Marcellus shale gas production (Fathi et al 2022b, Carr et al 2022, Pham et al 2021).

Due to their ability to improve processes overall, AI and ML techniques are being integrated more in the PNGE field. According to the studies, the use of AI and ML can result in very noticeable improvements in well performance, and significantly more accurate predictions of production rates and hydraulic fracture geometry. Additionally, drilling operations and maintenance schedules can be optimized with the help of AI and ML techniques, leading to substantial savings of cost and time. The use of these technologies in PNGE is a quickly expanding field of study, and these technologies are looking to be essential for enhancing the overall effectiveness and profitability of PNGE operations. The machine learning techniques are generally divided in the followings:

4.2.1 Supervised Learning:

Supervised learning is a method of machine learning (ML) in which the algorithm learns from a labeled data set in order to make predictions. The algorithm used the knowledge it has gained after training on a collection of data with known inputs and outputs to make predictions about brand new data. In PNGE, supervised learning can be applied to predictions of well performance, production rates, and detection of potential drilling hazards.

4.2.2 Unsupervised Learning:

Unsupervised learning is a method of machine learning (ML) in which the algorithm learns from an unlabeled data set to find patterns and relationships in the data. Data structures and patterns that might not be apparent to humans can instead be found by this algorithm. In PNGE, stratigraphic features and reservoir traits can be determined by grouping well data using machine learning. Such as Principal Component Analysis (PCA): an unsupervised ML algorithm that reduces the dimensionality of the given data to bring up the most important features for the model to perform better. By Zhang et al. (2017).

4.2.3 Reinforcement Learning:

Reinforcement learning is a form of machine learning (ML) where an algorithm learns through trial and error to optimize a given task. The algorithm gains feedback through its many tries on the task and uses the mentioned feedback to optimize its performance more and more over time. Reinforcement learning can be used in PNGE for improving production and drilling operation performance. By Sutton et al. (2018).

4.2.4 Deep Learning:

Deep learning is a method of using artificial intelligence (AI) which uses artificial neural networks to model complex relationships in given data. Multiple layers make up this process which is able to extract many features from the data. Deep learning can be applied in PNGE in reservoir characterization, improving hydraulic fracturing operations, and finding better spots for production. By Goodfellow et al (2016).

4.2.5 Convolutional Neural Networks (CNNs):

CNNs are a deep learning algorithm which is used for image recognition and categorization. They are made up of multiple layers which use characteristics they extract from the data to categorize the data. CNNs are able to be used in PNGE to recognize lithologic features in seismic data and well logs.

4.2.6 Decision Trees:

Decision trees are an ML technique in which the algorithm learns a sequence of if-then statements to make predictions. Based on the values of the input, the algorithm divides the data collection into smaller groups, resulting in a structure of data that resembles a tree shape. Decision trees can be used in PNGE to predict well performance and spot possible drilling risks.

4.2.7 Random Forests:

Random forests are a machine learning method that employs multiple decision trees to enhance and improve the accuracy of predictions. This algorithm combines the predictions of many decision trees in order to arrive at a final, more accurate prediction. As shown below.

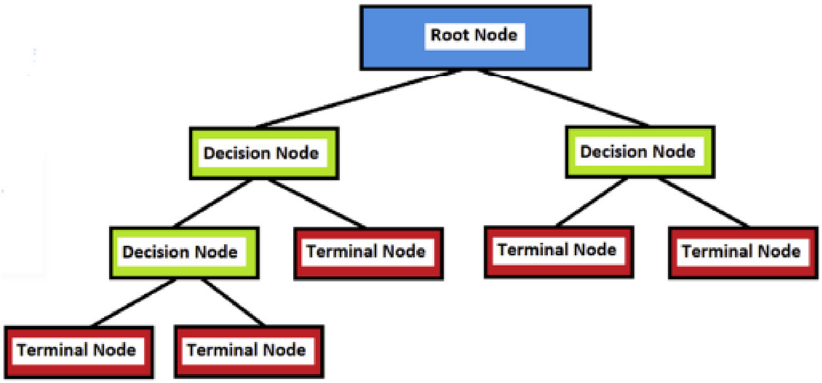


Figure 11 Decision trees

And the following figure can simplify what random forest means. make decisions randomly.

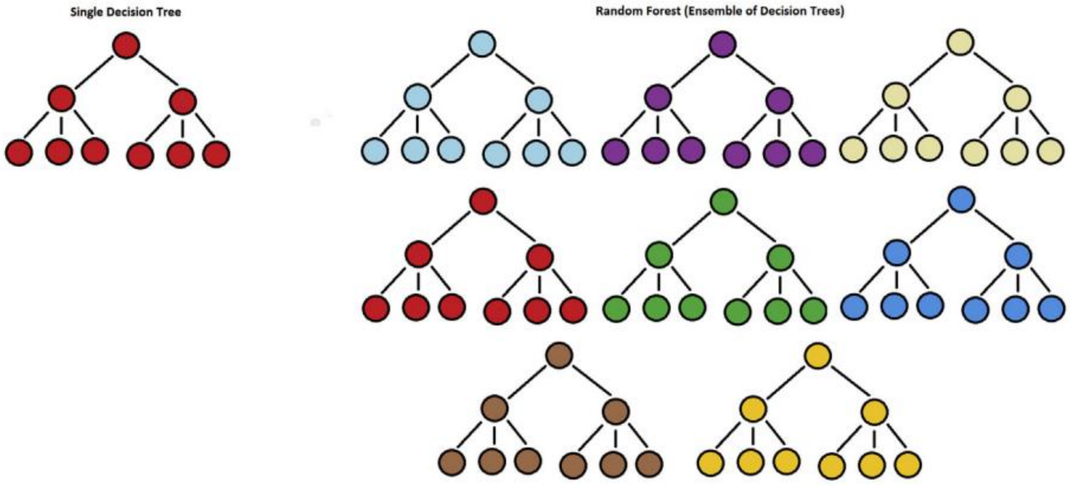


Figure 12 Decision trees vs random forest

4.2.8- Extra trees:

is another form of supervised ML algorithm similar to random forest used for classification and regression problems. The main difference between extra trees and the random forest is that random forest chooses the optimum split when splitting nodes.

4.2.9 Support Vector Machines (SVMs):

SVMs are a form of machine learning used for regression analysis and classification. Based on the given input, this algorithm separates the input data set into two groups and establishes a boundary between them. SVMs can be applied in the PNGE field to improve predictions and identify potential risks as well. Its benefit is maximizing the margin between support vectors through a separating hyperplane. Where the hyperplane is drawn in a black solid line to separate the data into red and blue lines, the closest blue or the red to the separator line is referred to as a support vector, and therefore it is called SVM.

It can be seen that a variety of AI and ML techniques exist which can be applied in the PNGE field. Each technique has unique advantages and uses. Some of the AI and ML techniques used in PNGE include supervised learning, unsupervised learning, reinforcement learning, deep learning, CNNs, decision trees, random forests, and SVMs. These methods can be used to predict well performance, optimize drilling operations, enhance and improve reservoir modelling, locate output sweet spots, and improve hydraulic fracturing operations among many other things.

4.3 Applications of AI and ML Methods in PNGE:

Numerous research has shown how successful AI and ML techniques are for PNGE. A supervised learning algorithm was used by Wang et al. (2019) to accurately predict well productivity. In different research done by Wang et al. (2021), drilling procedures were optimized using reinforcement learning, which resulted in a sizable decrease in drilling time and expense. As shown in the studies conducted by Wu et al. (2019) and Lashgari et al. (2020), deep learning methods have been used to improve reservoir characterization and finding production sweet spots.

Because they can offer insightful data and streamline processes, AI and ML techniques are being used more and more in the field of PNGE. As discussed, these techniques have been shown to provide very desirable results and studies have proven that AI and ML methods can improve drilling operations, minimize costs, and optimize reservoir modelling. This makes these methods of AI and ML an area of research that is being heavily focused on due to the promising results they have shown so far.

4.4 Difference between regression and classification:

In artificial intelligence and machine learning, there are two kinds of techniques called regression and classification. These two machine learning methods are used to evaluate and predict data. Despite the fact that both methods are used to evaluate data, each has different goals and approaches. The following will be a summary of the differences between regression and classification in terms of AI and ML:

4.4.1 Regression:

Regression is a type of machine learning technique that utilizes input variables to predict a continuous value. It is used to analyze the interaction between two or more variables and ascertain the impact of one variable on the other. Regression is a statistical technique in which an algorithm learns to predict a numerical value, such as temperature, weight, price, etc. based on the variables of the input.

There are two primary types of regression algorithms: linear regression and non-linear regression. Linear regression models assume that there is a linear relationship between the input and out variables. On the other hand, non-linear regression models can capture more complicated relationships between the variables due not presuming a linear relationship between the input and output.

4.4.2 Classification:

Classification is a machine learning method that is utilized to predict a categorical value based on the input variables. This method can be used to analyze data into distinct groups or categories. The classification algorithm learns, from the given input data, which groups or categories to assign input variables to.

There are two primary types of classification: binary classification and multi-class classification. Binary classification is where the algorithm classifies the available data into two groups. While multi-class classification is an algorithm capable of classifying data into more than only two groups.

Regression is used to predict a continuous value, whereas classification is used to predict a categorical value. This is the main distinction between the two methods. While classification algorithms examine the relationship between input variables and distinct groups or categories, regression algorithms analyze the relationship between input variables and output variables.

To summarize, regression and classification are two important techniques used in AI and ML. Despite the fact that both methods are indeed used to analyze a set of data, their methods and objectives are different. A continuous value is predicted using regression, whereas a categorical value is predicted using classification. When choosing the best machine learning method for a job, it is essential to understand the differences between regression and classification.

Knowing the differences and distinctions between the two types, we observe that the regression technique is the most suitable method out of the two for our project in order to achieve exact and precise values for each individual case. In addition, a machine learning (ML) algorithm is a set of mathematical procedures which are used to solve a problem by analyzing sets of data. A machine learning algorithm’s objective is to discover patterns or connections in the data, which can be then applied on new and unknown data to make predictions and decisions. Machine learning algorithms are frequently used for tasks like classification, regression, anomaly detection, and clustering. It is possible to supervise these algorithms, in which case they are trained on data with already known outcomes. It is also possible to use these algorithms unsupervised where they are trained on unlabeled data with unknown outcomes. The particular issue being addressed, and the characteristics of the data will determine which algorithm is most suitable to use. There are various kinds of machine learning methods, from linear regression to complex deep learning networks, and each has advantages and disadvantages.

There are several popular machine learning algorithms that are commonly used for regression tasks, depending on the specific requirements of the problem at hand. Some of the most popular regression algorithms include:

- Linear Regression
- Polynomial Regression
- Decision Tree Regression
- Random Forest Regression
- Support Vector Regression (SVR)
- Gradient Boosting Regression
- Neural Network Regression
- XGBoost Regression

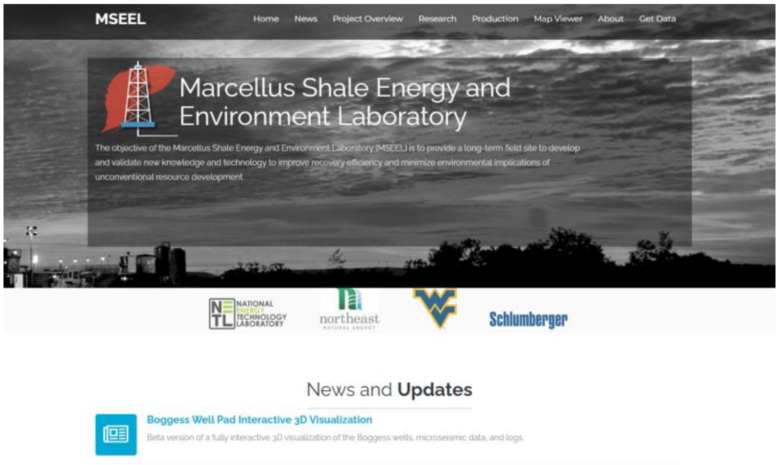
5-Methodology

Methodology workflow: we are going to investigate which ML model works best to predict the pressure from all other the features we have. And test them on all stages including the problematic one. And then will come up with a conclusion of the best model. After deciding, which is the best model, we will see if this best model can work on other stages with only being trained in a single first stage, if it won't work then more data needs to be added to make the prediction.

5.1- Data collection and preprocessing:

As mentioned before, the data for this project was collected from the MSEEL website (MSEEL.ORG) figure13 for BOGGESS wells. After collecting the data, the first step is to look into the data, understand it, and visualize it to make sure that the data looks all right and has no errors so we can run the model without any issues.

The CSV files were obtained from the "Get Data" button → Boggess → then select the one-second completion data of the specified well. As shown in figure 14.



Marcellus Shale Energy and Environment Laboratory

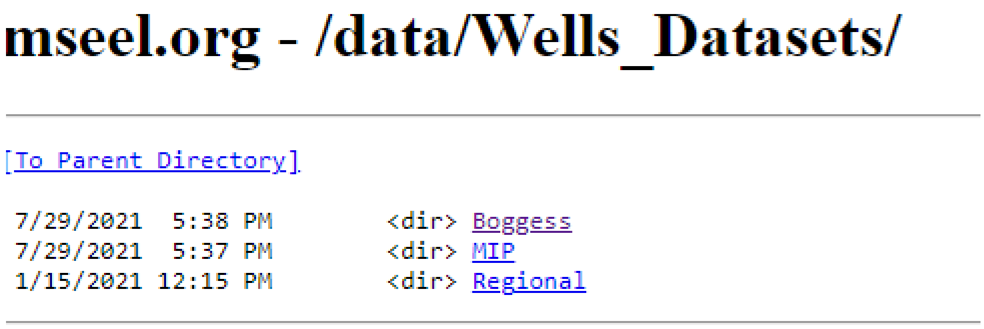
The objective of the Marcellus Shale Energy and Environment Laboratory (MSEEL) is to provide a long-term field site to develop and validate new knowledge and technology to improve recovery efficiency and minimize environmental implications of unconventional resource development.

News and **Updates**

Boggess Well Pad Interactive 3D Visualization

Beta version of a fully interactive 3D visualization of the Boggess wells, microseismic data, and logs.

Figure 13 MSEEL Project website.



mseel.org - /data/Wells_Datasets/

[To Parent Directory]

```
 7/29/2021  5:38 PM        <dir> Boggess
 7/29/2021  5:37 PM        <dir> MIP
 1/15/2021 12:15 PM        <dir> Regional
```

Figure 14 MSEEL website dataset's location.

5.1.1- Visualizing the data:

Visualizing the data is the first step before applying any preprocessing. The idea is to look into the data and confirm the physical plausibility of the data. For example, if there is any missing data or any shut-ins to be aware of. This step is really helpful because it makes the engineer get familiar with the data and understand how it behaves. The main issue faced during visualization of Boggess wells was the error in the time stamp assigned to the data that needed to be fixed and correlated to the one-second data.

To visualize the data, the code is developed in Python open-source package to read the CSV files assigned to each stage of stimulation in each well, then, to drop any NA values, and finally, create plots for all columns with respect to time. Table 2 shows all the data columns obtained from the MSEEL website CSV files.

#	Columns	Meaning / Unit
1	Time Stamp	one second data
2	Annotations	
3	Blender 113.DISCHARGE	The pumping Rate Barrel per mins
4	Formula Server 2.Clean Job Total	slick water (no sand) (Barrels)
5	Formula Server 2.Clean Stage Total	slick water per annotations (Barrels)
6	DataVAN.Wellhead	Wellhead Pressure (psia)
7	Local Formulas.Blender PPA S.T.	Stage total of the blender (pounds of proppant added)
8	Local Formulas.Blender PPA Total	The total of the blender (pounds of proppant added)
9	Formula Server 2.FR GPT	Friction reducer gallons (per 1,000 gallons)
10	Formula Server 2.BIO GPT	biocides (gallons per 1,000 gallons)
11	Formula Server 2.SCALE GPT	reduces the amount of rust in the casing (gallons per 1,000 gallons)
12	Formula Server 2.Backup FR GPT	FR backup (gallons per 1,000 gallons)
13	Formula Server 2.Blender PPA	blender added (pounds per proppant added)
14	Formula Server 2.BIO	biocides (pounds)
15	Formula Server 2.Scale	pounds
16	Formula Server 2.FR	FR (pounds)

Table 3 CSV given data columns.

5.2-Machine learning model development:

There are lots of techniques under supervised machine learning. Supervised machine learning means that the machine is going to use labeled datasets to train the model to predict an output as a map or values. The following list is going to be the technique that goes under supervised machine learning:

5.2.1 Multilinear regression: there are lots of types of regressions, such as linear regression, which uses the data to draw a linear line across a graph where it shows the relationship between two variables. This technique is one of the simple and most used methods for regression in ML algorithms that uses linear line equation as follows:

$$y = mx + b \qquad \text{equation (1)}$$

5.2.2 Random Forest feature selection. After performing a test train split to the data, feature selection will be made by using a random forest, which basically combines many decision trees into a single model to know which feature has more effect, It is known as a forest of decision trees. Which decision tree is a method used in regression; from its name, you can get a picture of what it does. A decision tree basically splits the data into subtrees and then splits the subtree into another subtree, as is shown in the figure below.

5.2.3 Extreme gradient boosting: also known as XGBoost, it uses L1 and L2 regularization, which benefits the model in generalization and overfitting reduction. Which also creates a series of decision trees and combines them to make more accurate predictions that will help the model to run and predict better. In our case we will use a minmax scaler to scale our dataset. And there are sets of parameters that need to be adjusted. For our model here are the parameters and their input that made our model to be better.

- nthread [1]
- objective [squared error]
- learning rate [0.7]
- max depth [1:11]
- min child weight [0.5]
- subsample [0.6]
- colsample bytree [0.1,0.5]
- n estimators [500]

5.2.4 Neural Network:

it is a type of machine learning model that is used to predict the reservoir properties. It is composed of interconnected nodes, the same neurons that are in the human brain. It learns from the data and recognizes the patterns to make predictions. It can predict the pressure data for optimization purposes. For our model here are the parameters and their input that made our model to be better.:

First, we used the min max scaler to scale our data, after that we used the flowing combination:

- hidden layer size [500,100,100,100]
- max iteration [1000]
- random state [20]
- solver [adam]
- activation [relu]

5.3- Screen-out detection using positive and negative slope:

Hydraulic fracture procedures are complicated and demanding operations which require multiple contributors and can cause very high economical costs if a Screen-out happens. Screen-outs lead to increasingly higher costs of time and money for all contributors due to delays in the operation. Advanced warning of a Screen-out allows for better decision-making in terms of whether an operation should be ceased or continued. Therefore, being able to prevent a Screen-out via advanced warnings is very helpful to keep losses at a minimum for all the contributing members as well as optimizing the performance of the hydraulic fracture procedure.

A Screen-out is bound to occur when the surface pressure slope deviates from the inverse slope. Being warned of this occurrence prematurely is a significant advantage as it enables the displacement process to start sooner and evades the possibility of leaving extra proppant in the wellbore leading to more losses. On the other hand, if advanced warnings using the inverse slope method show that the surface pressure slope remains consistent with the inverse slope, then the treatment can be continued.

The inverse slope, or the positive surface pressure slope, can be controlled by adjusting the proppant in the wellbore area and its amount. The inverse slope can be compared to the negative pressure slope to predict the occurrences of Screen-outs. If both of the slopes have an equal magnitude, then a Screen-out is extremely unlikely to happen. Conversely, if it is observed that the inverse slope and surface pressure slope are in deviation, which is a sure sign that a Screen-out will occur, a displacement process of the proppant in the wellbore area can be initiated early to avoid the Screen-out and its related complications.

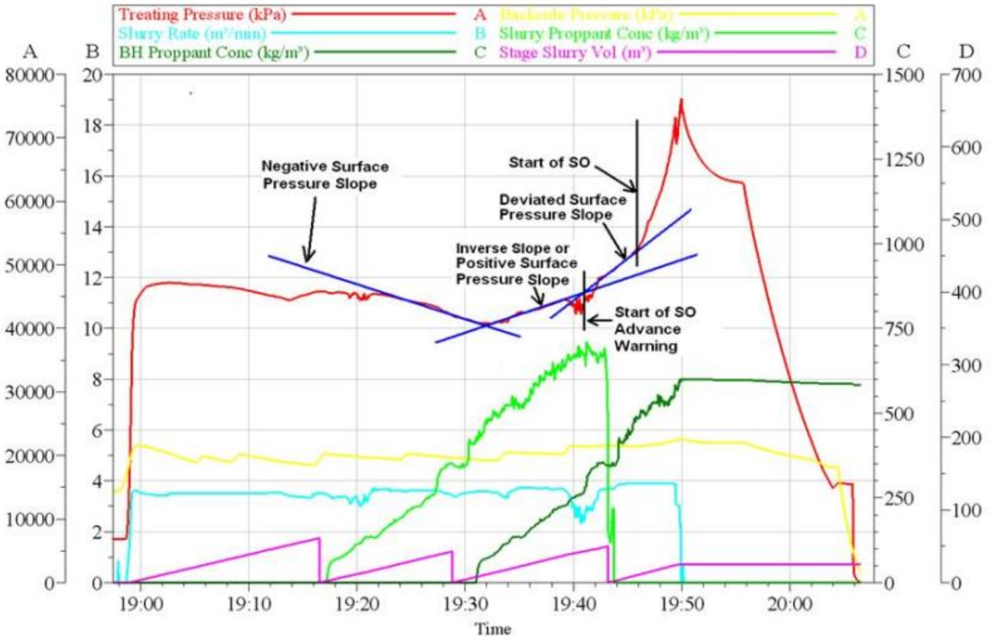


Figure 15 surface pressure behavior of a screened-out hydraulic fracture treatment.

5.4- Economic Analysis:

Excessive usage of friction reducers in frac jobs has been observed, resulting in millions of dollars in cost to the well's economics. Basically, the amount that could be optimized can save millions of dollars in total for all the wells combined. So, in this project, economic analysis will be performed by relying on the FR cost, FR daily pump fee, stages water volume, FR concentration, Undetected troublesome FR concentration, and the total number of stages in the well. As shown in the following equation, the FR cost per stage is proportional to the water volume of stages, friction reducer concentration, and the daily pump fee of FR. Basically, when we multiply the Vw with the FR concentration and we add them to the FR daily pump fee per stage, we will find the FR cost per stage as shown below.

$$\frac{FR\ Cost}{Stage} = (V_w * .042 * FR_{conc}) + \frac{FR_{Daily\ Pump\ Fee}}{Stages_{Day}} \qquad (Equation\ 2)$$

$$\text{which, } V_w = stages\ water\ volume \qquad (Equation3\ )$$

6-Results and Discussion:

6.1- Visualizing the data results:

After collecting the data from the MSEEL website, visualizing was performed to investigate the physical plausibility of the data collected. Figure 5 shows the BOGGESS 1H stage 1slurry injection rate in BBL/minute. As you can see, the rate was building around 4000 seconds after starting the process and then got stabilized until little after 8000 seconds and then stopped, which indicated that from 4000 seconds to 8000 seconds, the fracture propagates to the formation.

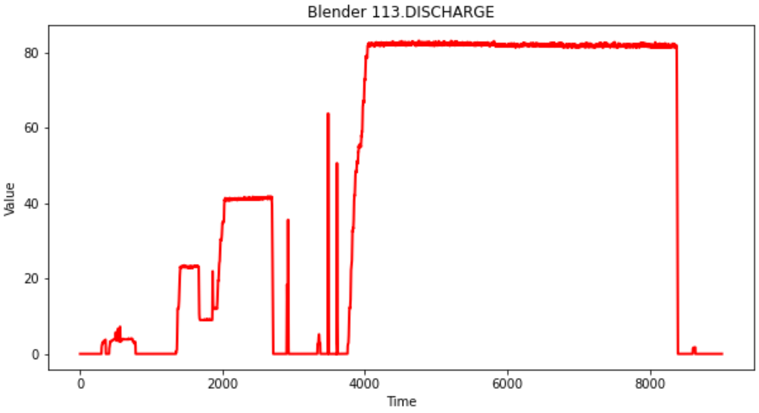


Figure 16 Blender discharge generated plot,1H-1.

In the following figure 17, we can see the Friction reducer when where it started injecting in the well. Which is at 4000 seconds and have been maintaing manually at a value of 7.5 in order for the pressure to ba maintaned

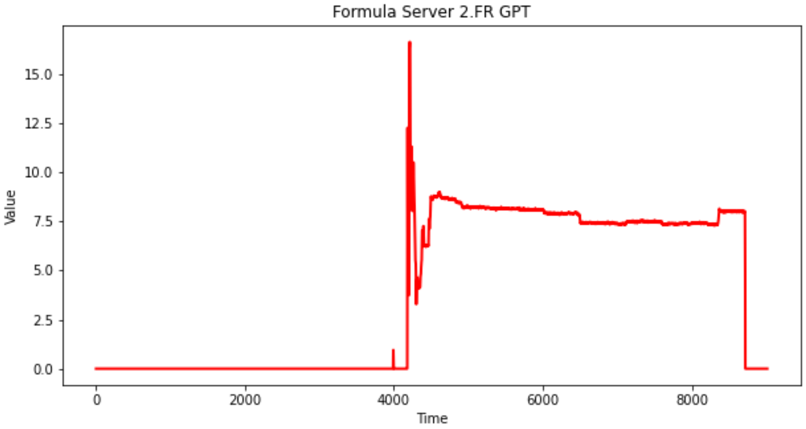


Figure 17 Friction reducer generated plot,1H-1.

Figure 6 shows the pressure behavior associated with Boggess 1H stage 1. That shows the pressure builds up to close to 12000 psi before formation breaks. During the same 4000 second to 8000 seconds the pressure is stabilized while different size and concentration of proppant is injected to the well. This happened with the help of dumping friction reducer into the wellbore while fracing.

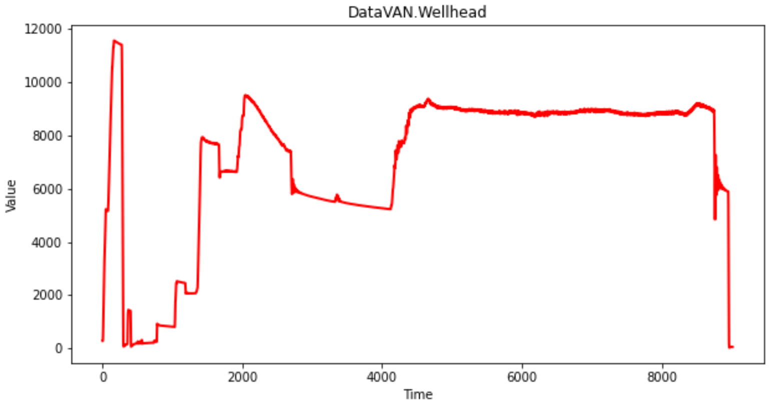


Figure 18 Friction reducer generated plot,1H-1.

6.2- Data engineering:

Before we start the model development, we need to do the finalized data engineering model, which will include deleting any NA values, dropping all the unnecessary annotations, normalizing the dataset, and performing the KNN imputation. The min-max scalar using for normalization as follows:

6.2.2- Minmax scaler equation.

The min-max scaler as shown below will transform the original feature of data to fall within the specified range, typically between 0-1. And the equation uses the minimum value of the data set and the maximum value of the dataset that you want to scale them to a specific range as shown in the equation below.

$$Min\ Max\ scaler = \frac{x - xmin}{xmax - xmin},$$

$$where\ x\ is\ the\ original\ feature\ value\ that\ need\ scaling \quad (Eqiation\ 4)$$

6.3 Machine learning model development results: Each parameter has its own distribution depending on the nature of the parameters, most of the ML techniques are based on multi Gaisen distribution, which requires the parameters to have a normal Gaisen distribution. To achieve that the quantile transform has been used to transform the distributions to the standard distributions.

The following figure 19 shows the total parameters and their transformed distribution using the quantile transform.

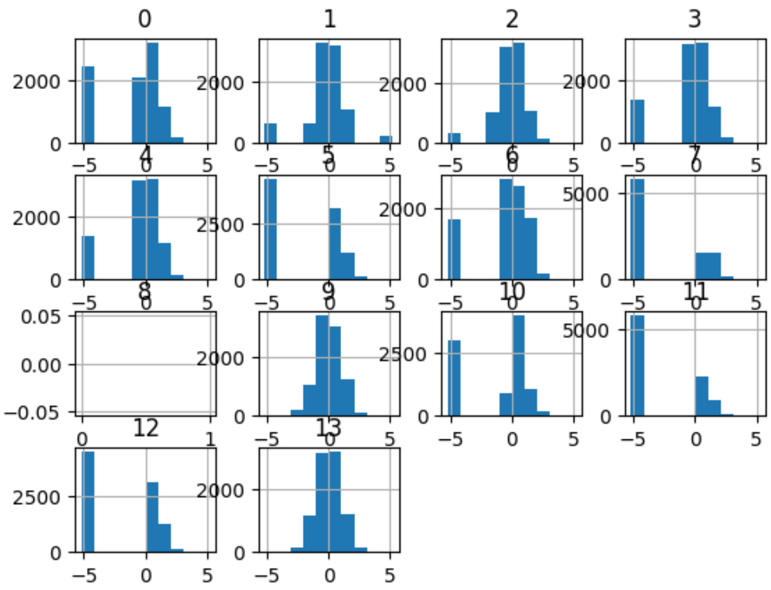


Figure 19 Transform distribution using quantile transform.

As shown in figure 19, from top left here the order shown below:

1- Blender 113.DISCHARGE
2- Formula Server 2.Clean Job Total
3- Formula Server 2.Clean Stage Total
4- Local Formulas.Blender PPA S.T.
5- Local Formulas.Blender PPA Total
6- Formula Server 2.FR GPT
7- Formula Server 2.BIO GPT
8- Formula Server 2.SCALE GPT
9- Formula Server 2.Backup FR GPT
10- Formula Server 2.Blender PPA
11- Formula Server 2.BIO
12- Formula Server 2.Scale
13- Formula Server 2.FR

For the sake of this project, let’s discuss and reveal what are the categories that got eliminated by our engineering thinking and decision making. First, annotations were deleted. Secondly, anything that includes total results will be deleted because we want our work to be done stage by stage. So, the following got deleted.

- 'Formula Server 2.Clean Job Total'
- 'Local Formulas.Blender PPA Total'
- 'Local Formulas.Blender PPA S.T.'
- 'Formula Server 2.Clean Stage Total'

And the following got deleted also because there will be no use for them in our model.

- 'Formula Server 2.FR GPT’
- 'Formula Server 2.BIO'
- 'Formula Server 2.Scale'
- 'Formula Server 2.Backup FR GPT'

For the sake of the project, we kept the most important features to predict the pressure, and as shown below.

- Blender 113.DISCHARGE
- Formula Server 2.BIO GPT
- Formula Server 2.SCALE GPT
- Formula Server 2.Blender PPA
- Formula Server 2.FR

6.3.2- Random Forest feature selection:

Feature importance indicates the measurement of how much each feature can affect the predictive capabilities of the model. It is used to know the required features that will help researchers to prioritize data collection. This way of measuring allows researchers to understand the factors that would allow them to customize their approaches way better, causing them to achieve optimum results.

In machine learning, the test train split is a process that facilitates the assessment of how well a predictive model performs. It involves dividing a dataset into two parts which are the training set, and the testing set.

The training set is utilized to inform the model on how to make predictions by studying patterns and similarities between the input features and the wanted variables. Meanwhile, the testing set is used to determine how accurate the model's predictions are by using the algorithm to a set of data and compares them against actual values predictions.

This facilitates the determination of how well and accurately the model can generally predict new unknown data. By leaving a part of the data aside as the testing set, we can confirm the model's performance against data it's not familiar with. This process provides us with the ability to determine if the model is working as intended and if it can be relied upon to give us accurate predictions.

As a first trial the training size of 80 % and the testing size of 20% of the Boggess 1H stage1 is used for the modeling. What we are trying to do is to make our model see a partial amount of the real data and see how accurately it can predict the rest of the hidden data. In this way we will know if the trained model can predict the whole next stage of the same well or not. If so, then it would be great and can be used for the next stages of the same well, if it will not be able to we predict the pressure we will increase the training set. Figure 20 shows an example of pressure behavior of Boggess 1H stage 1 used for training (i.e., left side of the red line) and testing (right side of the red line).

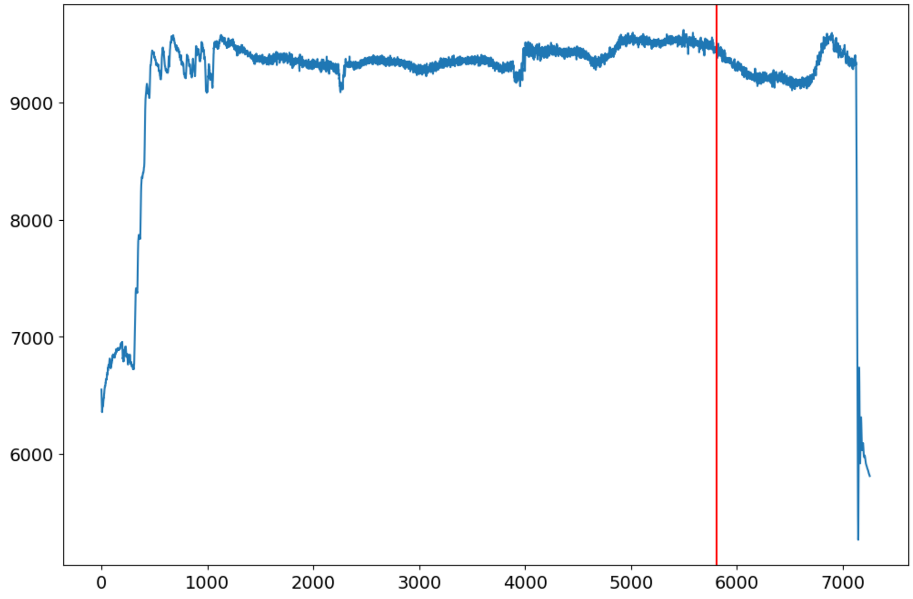


Figure20 trained and hidden data split code output.

After splitting the dataset, random forest feature importance was performed to see and understand which components have more effect in our model.

And the results of the random forest feature importance are as follows in figure 21 for BOGGESS 1H stage 1. As it is shown, the blender discharge is the highest feature then goes after that the blender proppant PPA. And the last two features are the Friction reducer and then the Biocide (GPT).

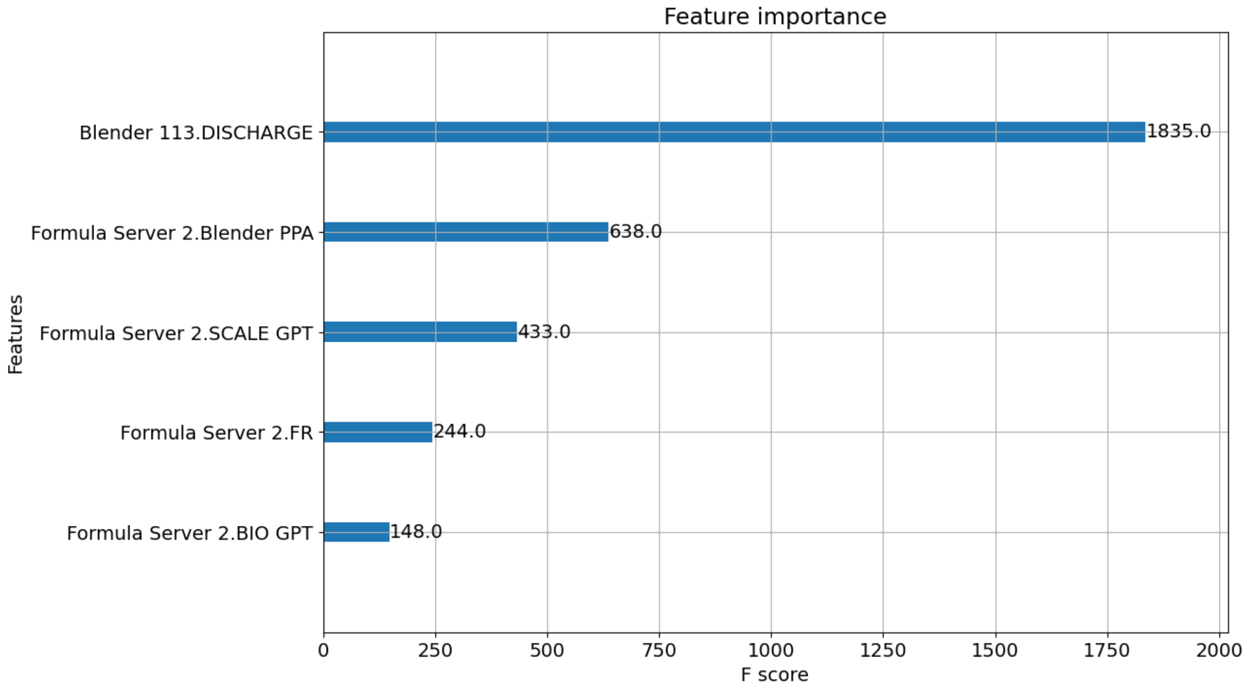


Figure 21 feature importance of random forest

6.3.3- Principal Component Analysis (PCA):

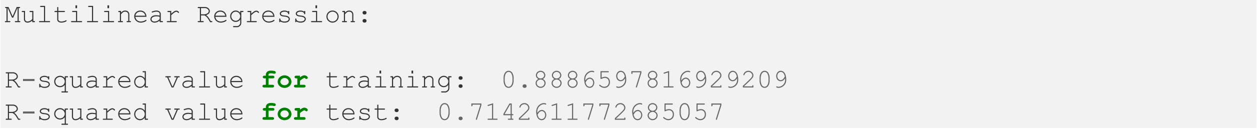
Principal component analysis is a common step of ML development where the technique will help in reducing the dimensionality of the problem by shrinking the parameter space domain but preserving the maximum information possible regarding the objective function in this case pressure. We have performed the DCA analysis, and the results show that 5 parameters are enough to carry %92 of the information. Followings are the scores of 5 parameters. The details of the process and the code used to perform the DCA is presented in the appendix.

In the followings the performance of different ML techniques in predicting the wellhead pressure during hydraulic fracturing using injection rates, proppant, FR, and chemical additives are presented and the best models are selected for more detailed studies of the pressure behavior and screen out analysis.

PHASE 1: Choose best models:

6.3.4 Multilinear regression:

The multilinear regression is applied to predict the well head pressure behavior of the Boggess 1H stage2. Figure 22 shows the quality of the prediction. As it shows, poor prediction is obtained using this technique. The accuracy of the Multilinear model is quantified with R2 test score R2 training score where in both cases the value was 0.88 and 0.71 which is an okay prediction quality. The following figure 22 details the output of the model and the code used to generate these results and are presented in appendix 1.



```
Multilinear Regression:

R-squared value for training:  0.8886597816929209
R-squared value for test:  0.7142611772685057
```

Figure 22 Multi regression output

Figure 23 also shows the injection rate, friction reducer, sand concentration, and the delta pressure between the actual and the predicted pressure using Multilinear regression model.

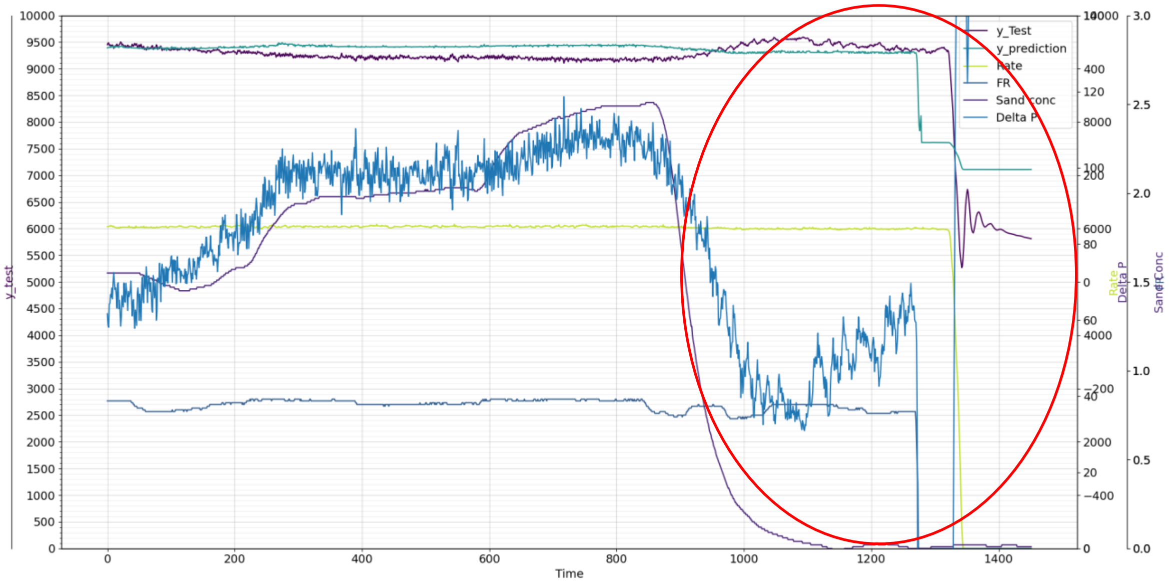
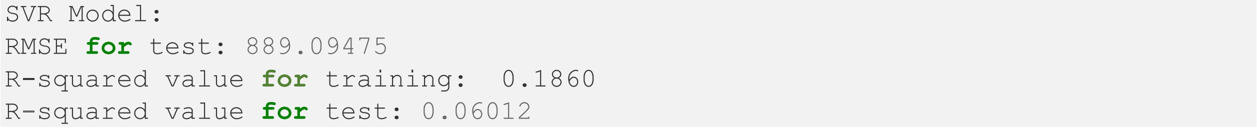


Figure 23 Multi regression Model prediction vs actual pressure plot code output.

As shown in figure 23, the predicted pressure (in green) (y_prediction) was generated and has been shown with the actual pressure data (y_test). Moreover, as it also shows, the delta pressure between the prediction and the actual got generated to see the difference between them to know how accurate the prediction is. The predicted pressure line is in the same pattern as the actual pressure, but it is a little bit off in the middle and way off at the end, where the drop pressure occurs. But it can tell us that the model had an idea of what the actual dataset looks like.

6.3.5 Support vector machine (SVM):

The next ML model used is the SVM. The workflow goes first into creating support vector machine model, after that the model will be trained and make the predictions for the tested data, finally, it will evaluate the prediction accuracy by giving us the RMSE and the R-squared value. The hyperparameters of this model describe the error and gamma if the Gaussion RBF kernel is selected and controls the curvature weight of the decision boundary. Figure 24 shows the quality of wellhead pressure predictions that shows poor quality of SVM for this problem with both Test R^2 of 0.06012. The following figure 24 shows the RMSE and R^2 of training and test sets respectively.



```
SVR Model:
RMSE for test: 889.09475
R-squared value for training:  0.1860
R-squared value for test: 0.06012
```

Figure 24 (SVM) SVR output scores

Figure 25 also shows the injection rate, friction reducer, sand concentration, and the delta pressure between the actual and the predicted pressure using Support Vector machine (SVM) model, which is very poor.

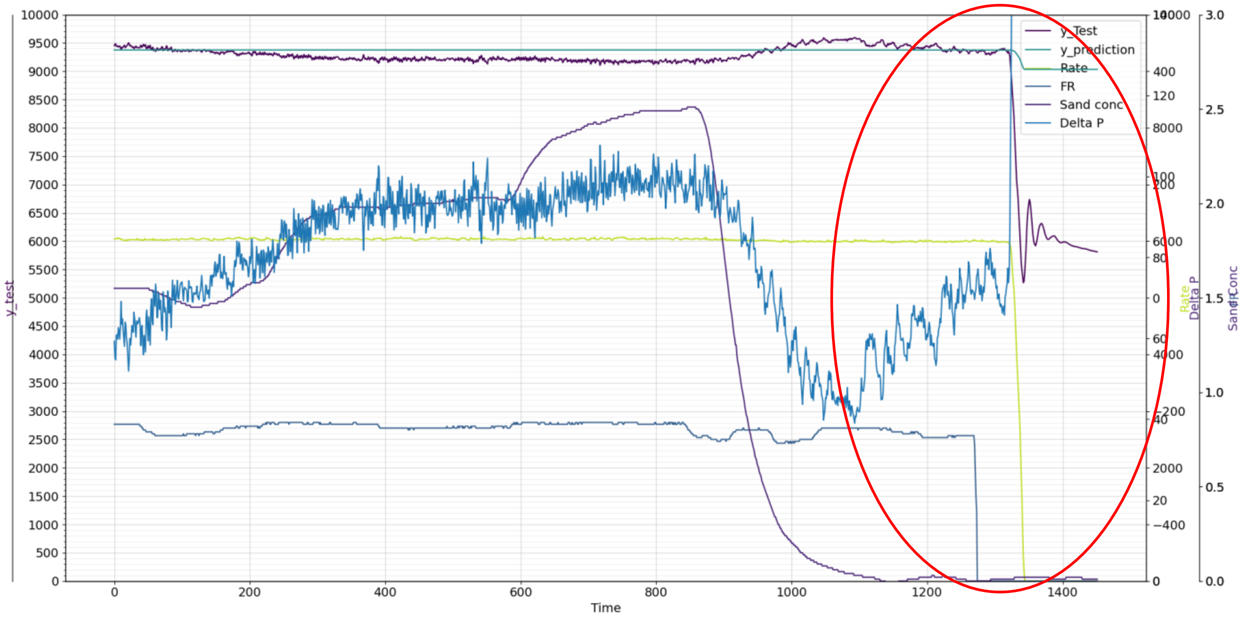
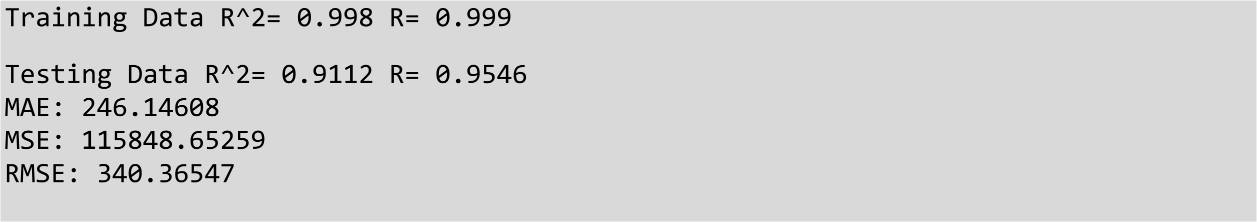


Figure 25 SVM Model prediction vs actual pressure plot code output.

As shown in figure 25, the predicted pressure (in green) (y_prediction) was generated and has been shown with the actual pressure data (y_test). Moreover, as it also shows, the delta pressure between the prediction and the actual got generated to see the difference between them to know how accurate the prediction is. The predicted pressure line is not in the same pattern as the actual pressure, and it is way off, and it did not track where the drop pressure occurs.

6.3.6- Extreme gradient boosting (XGBoost):

The XGBoost is the next ML model tested for this problem. In addition to setting the features and objective of the model the hyperparameters of the XGBoost model need to be set. For this purpose, the random search technique is used to optimize the XGBoost, hyperparameters including nthreads, objectives, learning rates, max depth, min child weight, subsample, colsample by tree, and number on estimator. After that the accuracy of the training data, testing data will be obtained. Adding on that the error analysis results will also be shown, where the MAE, MSE, and RMSE values will be generated as an output. As shown in figure 26. High accuracy is obtained between y_test that is the well head pressure used for history matching and y_prediction that is the prediction of wellhead pressure by XGBoost model. Both the training and testing data sets show high quality quantified with R^2 values above 90%. Figure below shows the four metrics used to quantify the accuracy of the XGBoost.



```
Training Data R^2= 0.998 R= 0.999

Testing Data R^2= 0.9112 R= 0.9546
MAE: 246.14608
MSE: 115848.65259
RMSE: 340.36547
```

Figure 26 XGBOOST output scores.

Figure 27 also shows the injection rate, friction reducer, sand concentration, and the delta pressure between the actual and the predicted pressure using XGBoost model.

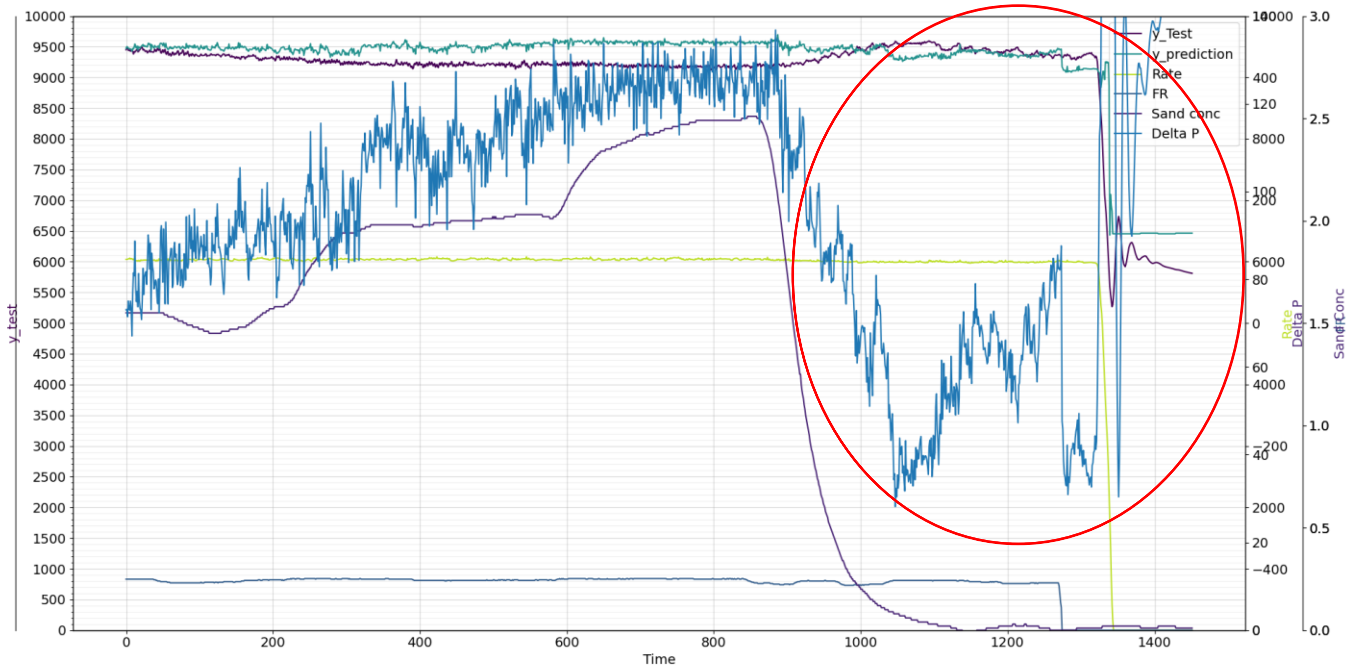


Figure 27 XGBOOST Model prediction vs actual pressure plot code output.

As shown in figure 27, the predicted pressure (in green) (y_prediction) was generated and has been shown with the actual pressure data (y_test). Moreover, as it also shows, the delta pressure between the prediction and the actual got generated to see the difference between them to know how accurate the prediction is.

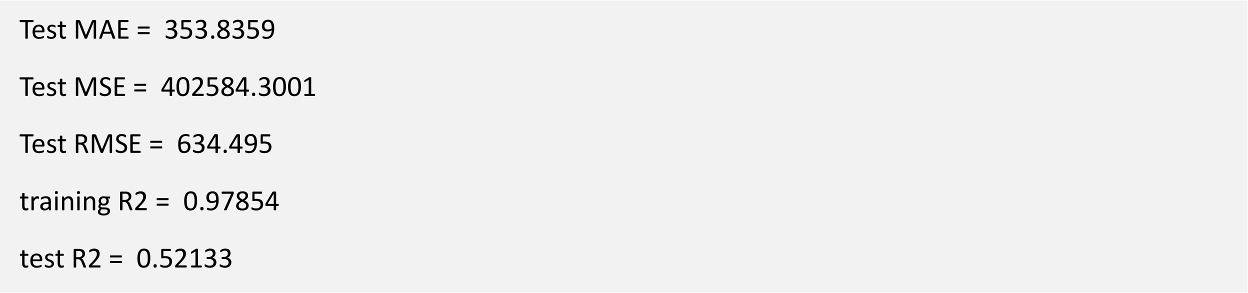
The predicted pressure line is in the same pattern as the actual pressure. Also, it has the same drop in pressure at 1350 seconds, which can tell us that the model had an idea of what the actual dataset looks like even though we did not show the model the actual data.

As shown in the results, the training data got a score of 0.998 which is so good, and the testing also got a great score which is 0.9112. and for the MAE, MSE, and RMSE the results are considering good because we used the minmax scaler to fix the data.

6.3.7- Neural Network:

For neural networks, we used the MLP regressor, and the minmax scaler. After training the neural network by splitting the data set into trained and blind datasets. Tunning then comes into place where we inserted 1000 for max iteration, 20 for random state, ‘adam’ for solver, and ‘relue’ for activation.

After finishing up the setup for the neural network, we ran the model on our 1H2 dataset, and the results came out as follows:



```
Test MAE =  353.8359

Test MSE =  402584.3001

Test RMSE =  634.495

training R2 =  0.97854

test R2 =  0.52133
```

Figure 28 Neural Network output scores.

The next generated plot shows the result of the prediction and how accurate the prediction is. It is not that bad, but whenever we compare it with the XGBoost method, we will find that the XGBoost got better results and prediction. Even though the training has a score of 97%, but as you can see the testing has a score of 52% in Neural Network, while in the XGBoost the testing score was 91%

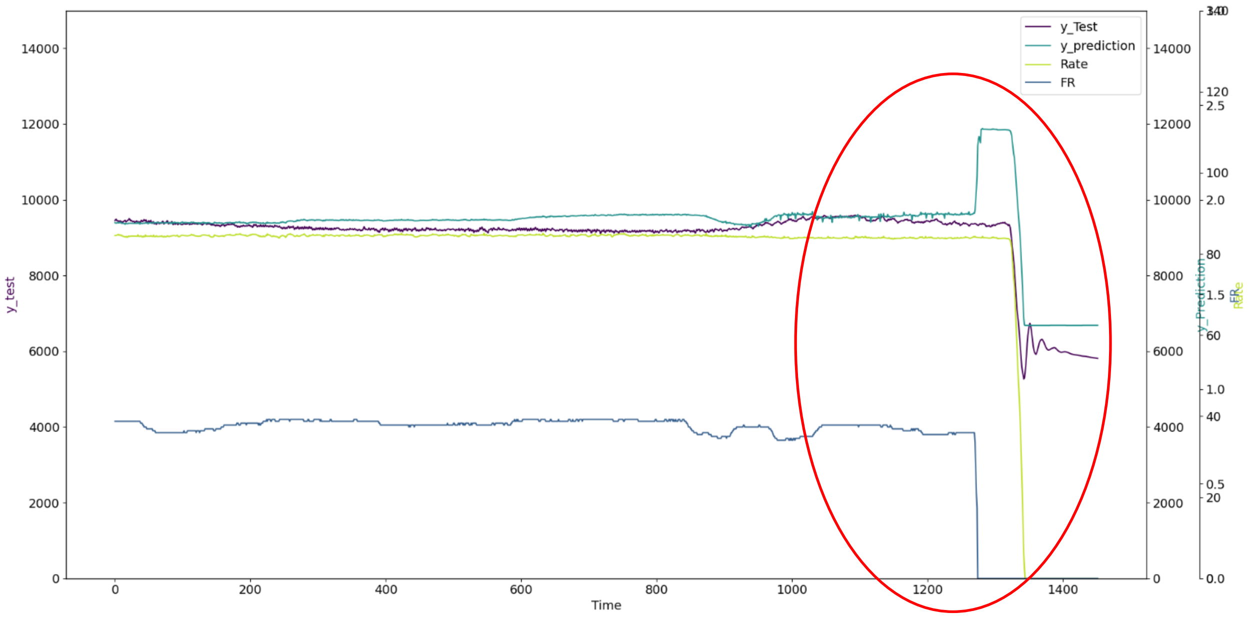


Figure 28 Neural Network Model prediction vs actual pressure plot code output.

Phase 1 outcome summary:

The following table shows the best model after running them on several wells with different stages. And the table shows that the best model created is the XGBoost after comparing the scores with the other models.

Phase 1	Training Score	Testing Score
XGBoost	99.80%	95.46%
Neural Network	97.85%	52.13%
SVR	18.60%	6.01%
Multi Regression	88.86%	71.42%

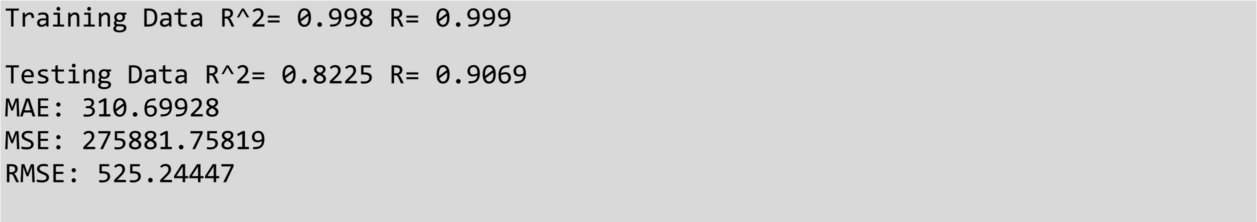
Table 4 Phase 1 summary scores.

PHASE 2: Test the best model abilities:

In phase 2, we will test the ability of our best model (XGBoost). And to do that we will test the model on different stages without training the stages and use the new stage data as a blind test.

We will do two tests, the first one is a good normal stage. The second test will be on a problematic stage and see the accuracy we will get.

The following result will be on Boggess 1H stage 3 after using the whole stage data as a blind test. And as shown the results are good for using a whole blind new data set. Where the training R^2 is 0.998 and the testing is equal 0.8225.



```
Training Data R^2= 0.998 R= 0.999

Testing Data R^2= 0.8225 R= 0.9069
MAE: 310.69928
MSE: 275881.75819
RMSE: 525.24447
```

Figure 29 1H-3 run scores.

The following is the predicted pressure generated plot for Boggess 1H stage 3. The prediction is good, and it is catching the flow, which supports our outcome that the XGBoost model is good.

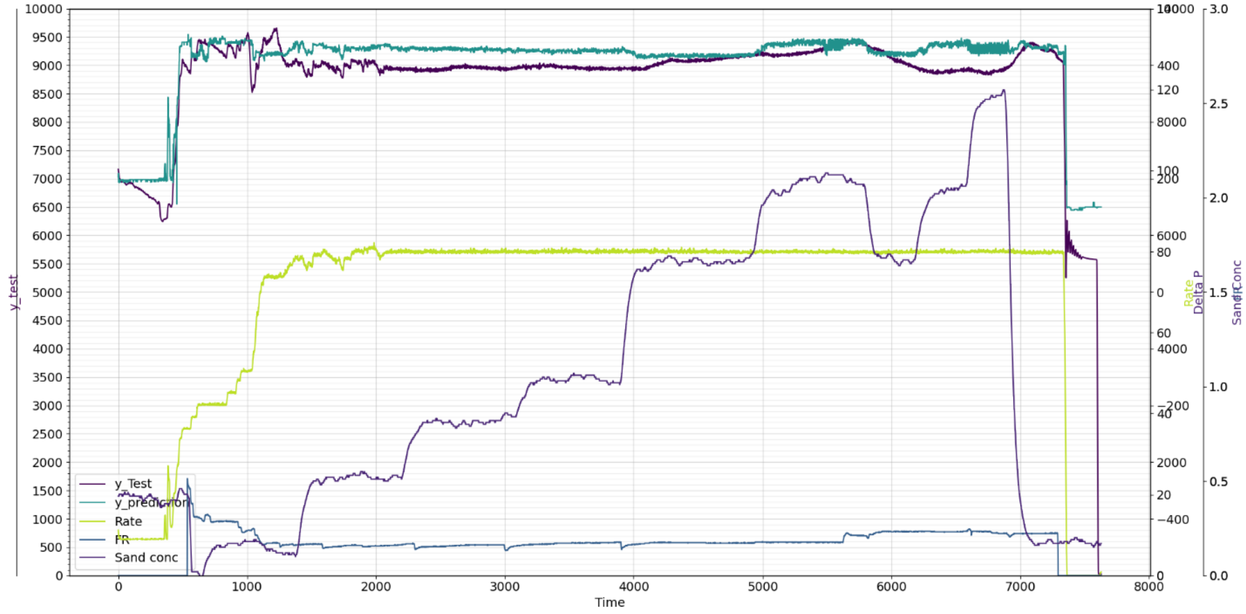
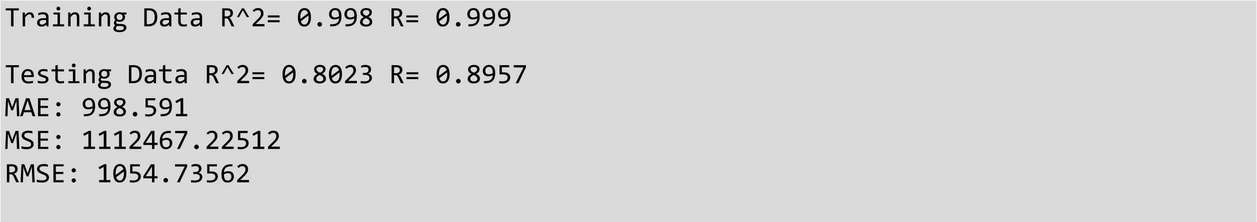


Figure 30 XGBOOST 1H-3 prediction vs actual pressure plot code run.

Now, we will use the same model on a problematic stage adding on that to a whole different well which will make it harder on the model and let us see the outcome. We will use the model on Boggess 3H stage 34, while using a trained model on Boggess 1H stage 2.

The following results show how good the created XGBoost model is, the testing score came out R^2=0.8023.



```
Training Data R^2= 0.998 R= 0.999

Testing Data R^2= 0.8023 R= 0.8957
MAE: 998.591
MSE: 1112467.22512
RMSE: 1054.73562
```

Figure 31 3H-34 run scores.

The following is the predicted pressure generated plot for Boggess 3 H stage 34. The prediction is very good for applying the model into a whole different well, and it is catching the flow, which supports our outcome that the XGBoost model is good.

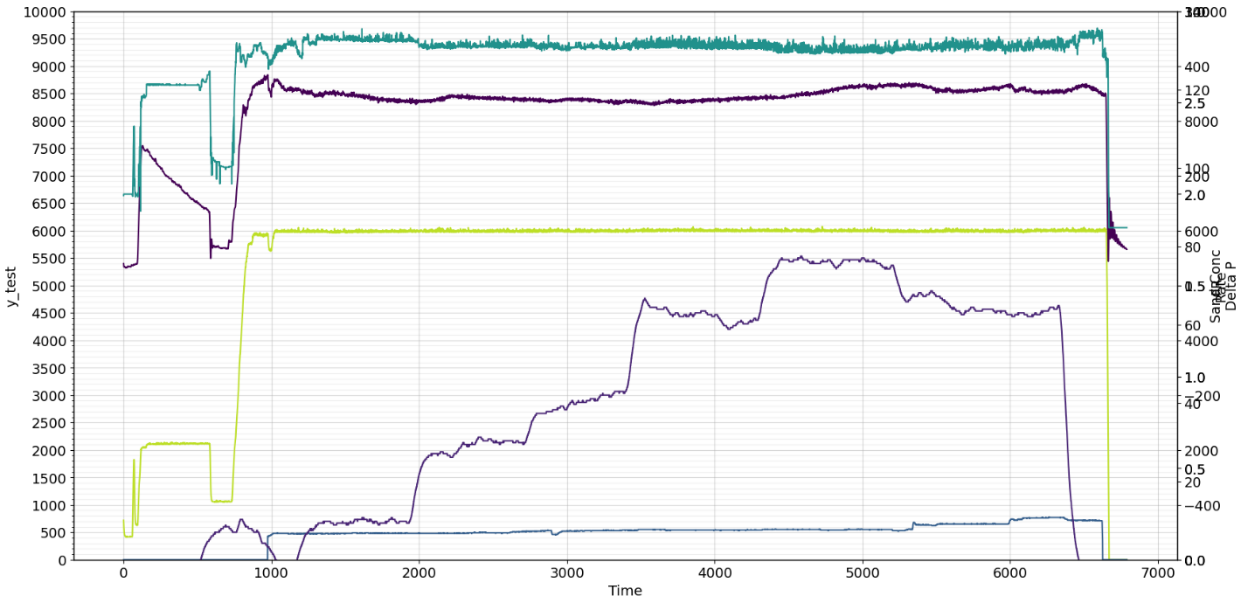


Figure 32 XGBOOST 3H-34 prediction vs actual pressure plot code run.

PHASE 3: FR reduction analysis:

On the following well, Boggess 9H stage 21, the friction reducer got a high weight as shown in the following feature importance figure. The following generated figures will show the impact of the FR whenever we reduce it.

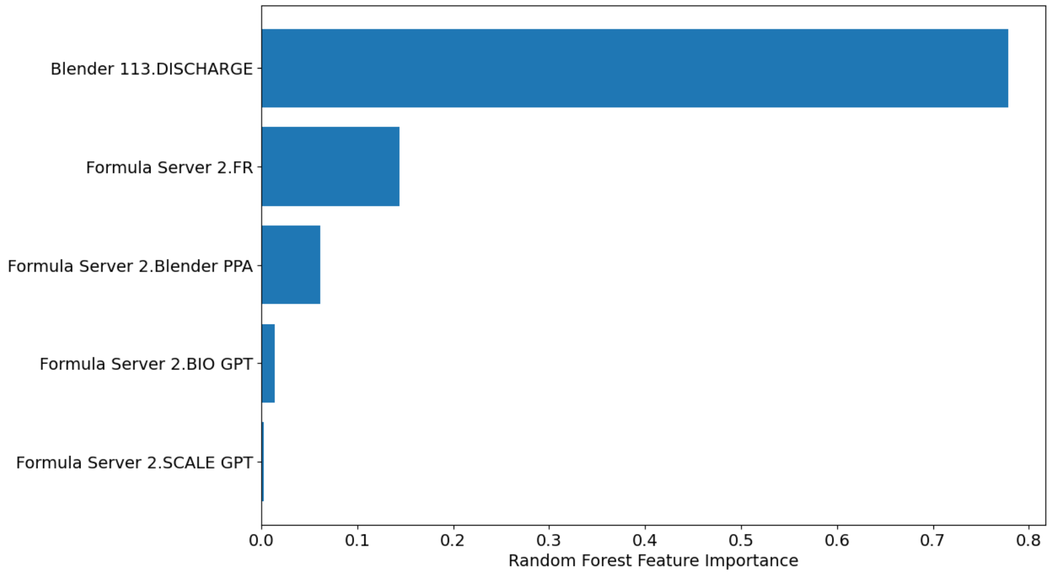


Figure 33 9H-21 feature importance.

For the 100% FR the following figure shows the delta pressure and how does the original delta pressure look like.

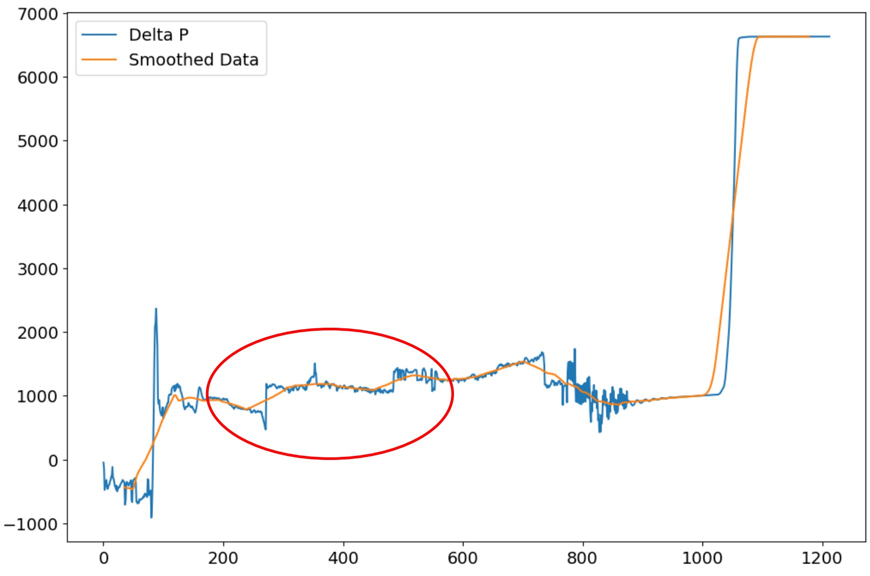


Figure 34 100% FR with no reduction.

The following figures are the 70% FR and the 50% FR, and it shows a little bit of changes in delta pressure. Which indicates that whenever we reduce friction, the pressure will be affected.

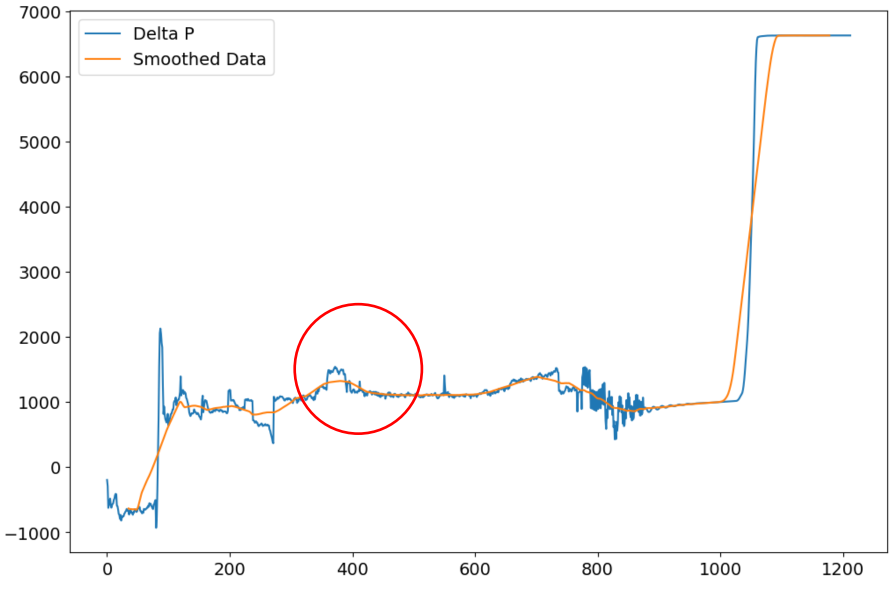


Figure 35 70% FR delta pressure.

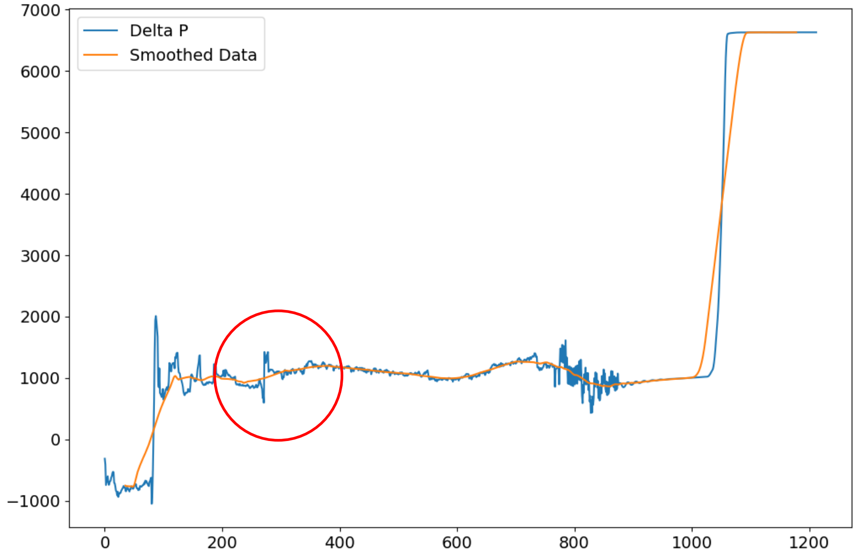


Figure 36 50% FR delta pressure.

The other option we have is that the feature importance will not include the FR to be one of the most important factors, in this a new combination of features is required to define the objective function. The combination of treating pressure*FR/treating rate can be used for this purpose to capture the impact of FR.

PHASE 4: Screen-Out:

In previous cases we did not have screen out stages to train the model as shown in figure 37 Boggess 1H stage 2, but the way it works is to look in the derivatives of the rate for the first, second, and third derivative, and see when the derivative start to go up, that is an indication start of a screen out. But in our case in figure 37 on the right is the pressure and on left is the slope of it, which there is not a noticeable pattern to detect screen-out.

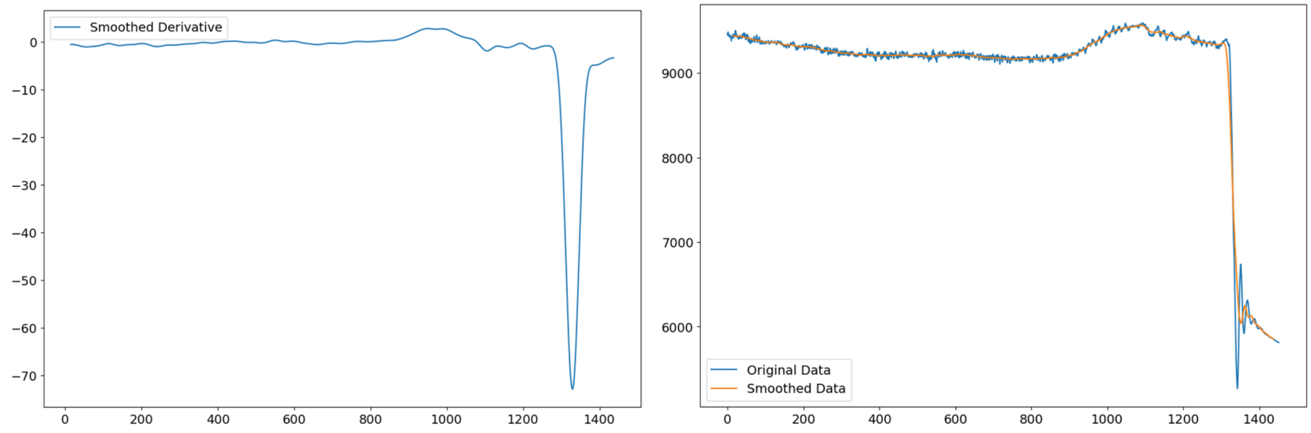


Figure 37 Boggess 1H-2 original data with smoothed derivative.

The following figure 38 is Boggess 5H stage 28, the idea was also to look if there will be any screen-outs, so we used the pressure plot and made it smooth and applied the derivative on it.

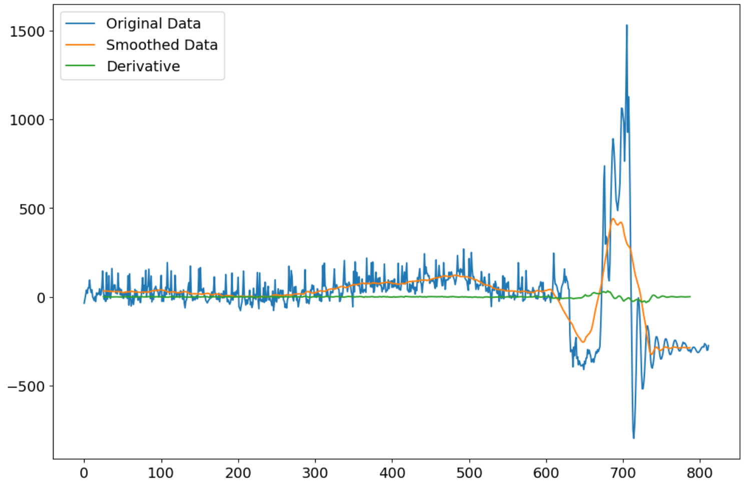


Figure 38 Boggess 5H-28 generated pressure plot with the slope.

The following figure shows the slope pattern, and there is no indication of a screen out also. There is not any second positive slope as is shown. Which indicates there is no screen-out.

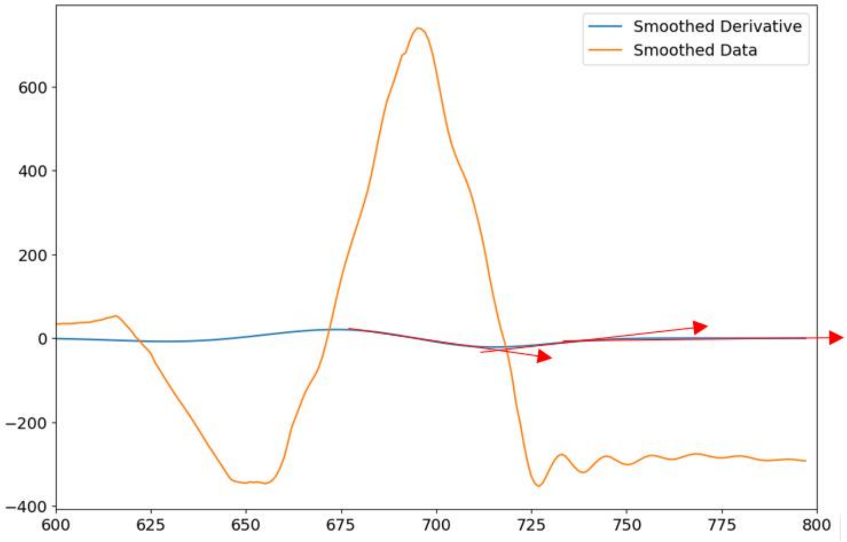


Figure 39 Boggess 5H-28 generated pressure plot with the slope [interval 600-900].

But if there was a screen out which in my case there is none, we will use the pressure slope behavior, as shown in figure 15. Where the negative slope is equal to the magnitude of the opposite positive slope, and it follows it with second positive slope.

6.4 Economic Analysis:

The success of horizontal well hydraulic fracturing stimulation, which can give maximal reservoir contact, is crucial in achieving economically competitive oil and gas production rates from unconventional shale resources. This can be accomplished by injecting 400,000 to 600,000 BBLs of fracturing fluid in a single well. However, there are technical and environmental problems associated with this technique that need to be resolved. The challenge in meeting designed pumping rates is overcoming tube friction pressure, theoretically lowering hydraulic horsepower demand by 80%. (Virk, 1975). This can be accomplished by including friction reducers in the fracturing fluid; however, excessive usage of friction reducers in frac jobs has been observed, resulting in millions of dollars cost to the well's economics. Basically, the amount that could be optimized can save millions of dollars in total for all the wells combined. Followings is an example in Marcellus shale:

Estimates are based on averages.

FR Cost	$11.04/gal
FR Daily Pump Fee	$1,370/day
Stages Water Volume	7,600 bbl
FR Concentration	0.83 gpt
Undetected troublesome FR Concentration	1.25 gpt
Total number of stages	11,922 stages

Table 5 Marcellus shale given values.

$$\frac{FR\ Cost}{Stage} = (V_w * .042 * FR_{conc}) + \frac{FR_{Daily\ Pump\ Fee}}{Stages_{Day}} \qquad (Equation\ 3)$$

$$\text{which, } V_w = stages\ water\ volume \qquad (Equation\ 4)$$

- Undetected troublesome stages use amounts of FR as high as 1.25 gpt.
- looking at FR conc. used in wells without any screen out issue we found they use 1.25 gpt which is 0.42 gpt more than requires 0.83 gpt FRconc as it is shown below:

$$1.25 - 0.83 = 0.42 \text{ gpt extra FR used}$$

- Based on the equation above: $\frac{\text{FR Cost}}{\text{Stage}} = (7{,}600 * 0.042 * 0.42) + 1{,}370 = \$1{,}500$
- With an average down time of 8 hours, screen out cost ≈ $12,500

1/5 of the stages treated harder and = 0.42 excess FR/stage was used, and the excess FR concentration is defined as the additional FR concentration beyond required 0.83 gpt injected into the well.

$$\frac{40}{5} = 8\ stages \rightarrow 8 * \$1{,}500 = \$12{,}000\ wasted$$

Now assume we had 1 screen out, that’s another $12,500

Total wasted $= \$12{,}000 + \$12{,}500 = \$24{,}500\ total\ wased$

Annual cost of extra Fr used is obtained by studying 11,922 stages that fractured in 2018, estimates are based on 1/5 of stages treating hard (1.25 gpt FR concentration injected instead of 0.83 gpt of FR) and 1 screen out per 1000 stages

Annual extra FR Cost:

$$= \frac{11{,}922}{5} * \$1{,}500 + \frac{1}{1000} * 11{,}922 * \$12{,}500\ (screen\ out\ cost) = \$\ \mathbf{3,740,000}$$

Because not being able to optimize the friction reducers, money will be lost due to:

- Lower surface treating rate
- High pumping pressure
- Excessive FR usage
- Screen Outs

7-Conclusion

After performing the machine learning techniques on the data sets that have been obtained from The MSEEL project, the model came out with good results. This project had Four phases outcome. In phase 1, the model was created and the XGBoost has been selected to be the best model in comparison with the other models, adding on that the model predicted the pressure with good testing score. In phase 2, the XGBoost model was tested to see if it can have a good pressure prediction on other fully blind datasets either on the same well or a whole different well. And the results came out positive as mentioned above in the results section of this thesis. In phase 3, the friction reducers analysis was applied and finally, in phase 4 the aim was to detect the screen-out using the derivative method. The method weas applied on our datasets, but our wells came out without any screen-outs.

In conclusion, all goals of this project have been reached, and an economic analysis has also been performed in our data. For future work I would recommend applying more methods on the same data and trying different techniques to predict the pressure and do more search on friction reducer sensitivity analysis.

8-References

1- Ali, M., & Islam, R. (2019). Petroleum reservoir characterization using decision tree and random forest algorithms. Journal of Petroleum Exploration and Production Technology, 9(4), 3271-3284.

2- Bagherpour, R., Kharrat, R., & Brevick, J. (2018). Random forest as a predictive model for geomechanical applications. Journal of Natural Gas Science and Engineering, 49, 237-249.

3- Belyadi, H., Fathi, E., Belyadi, Fatemeh. (2016). *Hydraulic Fracturing in Unconventional Reservoirs: Theories, Operations, and Economic Analysis* (1st ed.). Gulf Professional Publishing.

4- Belyadi, H., Fathi, E., Belyadi, Fatemeh. (2019). Hydraulic Fracturing in Unconventional Reservoirs: Theories, Operations, and Economic Analysis (2st ed.). Gulf Professional Publishing.

5- E Fathi, TR Carr, MF Adenan, B Panetta, A Kumar, BJ Carney. (2022)High-quality fracture network mapping using high frequency logging while drilling (LWD) data: MSEEL case study. Machine Learning with Applications, 100421, https://doi.org/10.1016/j.mlwa.2022.100421

6- Ebrahim Fathi, Timothy Carr, Mohammad Faiq Adenan, Brian Panetta, Abhash Kumar, BJ Carney. (2022 b). Near Wellbore Natural Fracture Mapping Using a New Hybrid Automated Machine Learning Workflow (AMLW): A Real Field Application in the Marcellus Shale Energy and Environmental Laboratory. AAPG, CCUS 2022 conference 29–31 March 2022 Houston, Texas University of Houston

7- Fathi, E., Keshavarz, A., and Zoveidavianpoor, M. (2015). Experimental investigation of friction reducers in hydraulic fracturing fluids. Journal of Natural Gas Science and Engineering, 22, 183-189.

8- Gao, S., Fan, Y., and Chen, J. (2017). Experimental study on the effect of friction reducer on hydraulic fracturing. Petroleum Exploration and Development, 44(6), 1086-1091.

9- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

10- Jiang, H., Zhang, Y., and Cao, W. (2019). A machine learning-based approach for optimizing drilling equipment maintenance scheduling. Journal of Petroleum Science and Engineering, 181, 1062-1070.

11- Liu, Z., Chen, Y., Li, C., and Li, D. (2020). An artificial intelligence approach to reservoir modeling in petroleum engineering. Energy Exploration & Exploitation, 38(5), 1415-1434.

12- Peng, P., Li, Q., Li, J., and Liu, Y. (2016). Potential environmental impacts of friction reducers used in hydraulic fracturing. Journal of Environmental Management, 181, 44-51.

13- Shan e Zehra Lashari, Ali Takbiri-Borujeni, Ebrahim Fathi, Ting Sun, Reza Rahmani, Mehdi KhazaeliDrilling Performance Monitoring and Optimization Using Artificial Intelligence Algorithms", Journal of Petroleum Exploration and Production Technology, 24 April 2019, 9 (4), 2747-2756

14- Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press.

15- Timothy Carr, Ebrahim Fathi, Rob Bohn, Mohammad Faiq Adenan, Liwei Li, Brian Panetta, BJ Carney, Natalie Mitchell. (2022 c). Importance of Preexisting Fractures to Completion and Production Efficiencies in the Marcellus Shale Energy and Environmental Lab MSEEL. SPE Hydraulic Fracturing Technology Conference and Exhibition

16- Vuong Van Pham, Ebrahim Fathi, Fatemeh Belyadi. New hybrid approach for developing automated machine learning workflows: a real case application in evaluation of marcellus shale gas production. Fuels 2 (3), 286-303

17- Wang, Y., Fang, B., and Dai, C. (2019). Machine learning for predicting well performance in petroleum engineering. Journal of Petroleum Science and Engineering, 174, 736-745.

18- Wang, Y., Xing, Y., Zhang, Y., and Guo, W. (2021). A case study on the application of artificial intelligence to drilling operations in petroleum engineering. Journal of Natural Gas Science and Engineering, 95, 103868.

19- Wang, Y., Yu, B., Wang, X., Zhang, J., and Xu, X. (2018). Experimental investigation of the effect of friction reducer on shale hydraulic fracturing. Journal of Petroleum Science and Engineering, 166, 306-316.

20- Xie, S., Zhang, X., Wang, S., & Song, X. (2020). A convolutional neural network for facies classification from well logs. Journal of Natural Gas Science and Engineering, 83, 103486.

21- Zang, Y., Li, X., Zhou, L., & Liu, G. (2019). Machine learning approach for rock strength prediction based on well logs data. Journal of Natural Gas Science and Engineering, 65, 260-269.

22- Zhang, J., & Li, X. (2017). Unsupervised machine learning in reservoir characterization: A review. Journal of Natural Gas Science and Engineering, 38, 359-377.

23- Zhao, X., Gao, Y., Hu, Q., and Wu, J. (2020). Application of artificial intelligence in hydraulic fracturing design in petroleum engineering. Journal of Petroleum Science and Engineering, 195, 107475.

24- Zolfaghari, A., Javadi, M. S., and Ahmadi, M. A. (2016). Experimental study of friction reducers for hydraulic fracturing in shale formations. Journal of Petroleum Science and Engineering, 146, 370-378.

9-Appendix

Code Model:

```
import numpy as np
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.inspection import permutation_importance
from matplotlib import pyplot as plt
from sklearn import preprocessing
import xgboost as xgb
from xgboost.sklearn import XGBRegressor
from sklearn.model_selection import GridSearchCV
from sklearn import metrics
from sklearn.datasets import make_regression
import pickle
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline
from sklearn.neural_network import MLPRegressor
```

Data engineering:

[input]:

```
#path:
data_lists = r'Datasets/Boggess_1H/1h2.xlsx'

df=pd.read_excel(data_lists, sheet_name='JobData') # change this
df.head()

print(len(df))
#Drop till acid, and drop after injection:
idx1 = np.where(df["Annotations"] == 'ACID')
#idx2 = np.where(df["Annotations"] == 'Flush')
result = df.truncate(before=idx1[0][0])
#after=idx2[0][0])


DropAnnotations=result.drop(['Annotations','Time Stamp','Formula Server
2.Clean Job Total','Formula Server 2.FR GPT',
'Local Formulas.Blender PPA S.T.','Local Formulas.Blender PPA Total','Formula
Server 2.BIO',
'Formula Server 2.Scale','Formula Server 2.Clean Stage Total','Formula Server
2.Backup FR GPT'],axis=1)

print(len(DropAnnotations))
print(DropAnnotations.columns)


#arrange
arranged_df = DropAnnotations.loc[:,['Blender 113.DISCHARGE',
       'Formula Server 2.BIO GPT', 'Formula Server 2.SCALE GPT',
       'Formula Server 2.Blender PPA', 'Formula Server 2.FR',
'DataVAN.Wellhead']]
arranged_df.head()

#Drop NA values
check_nan = arranged_df.isnull().values.any()
print(check_nan)


#Normalization:
dataset = arranged_df


print(dataset.head())
```

[Output]:

```
7877
7257
Index(['Blender 113.DISCHARGE', 'DataVAN.Wellhead', 'Formula Server 2.BIO
GPT',
       'Formula Server 2.SCALE GPT', 'Formula Server 2.Blender PPA',
       'Formula Server 2.FR'],
      dtype='object')
False
     Blender 113.DISCHARGE  Formula Server 2.BIO GPT  \
620                   3.75                       0.0
621                   3.53                       0.0
622                   3.37                       0.0
623                   3.27                       0.0
624                   3.19                       0.0

     Formula Server 2.SCALE GPT  Formula Server 2.Blender PPA  \
620                         0.0                           0.0
621                         0.0                           0.0
622                         0.0                           0.0
623                         0.0                           0.0
624                         0.0                           0.0

     Formula Server 2.FR  DataVAN.Wellhead
620                  0.0           6549.37
621                  0.0           6522.19
622                  0.0           6481.71
623                  0.0           6441.95
624                  0.0           6403.50
```

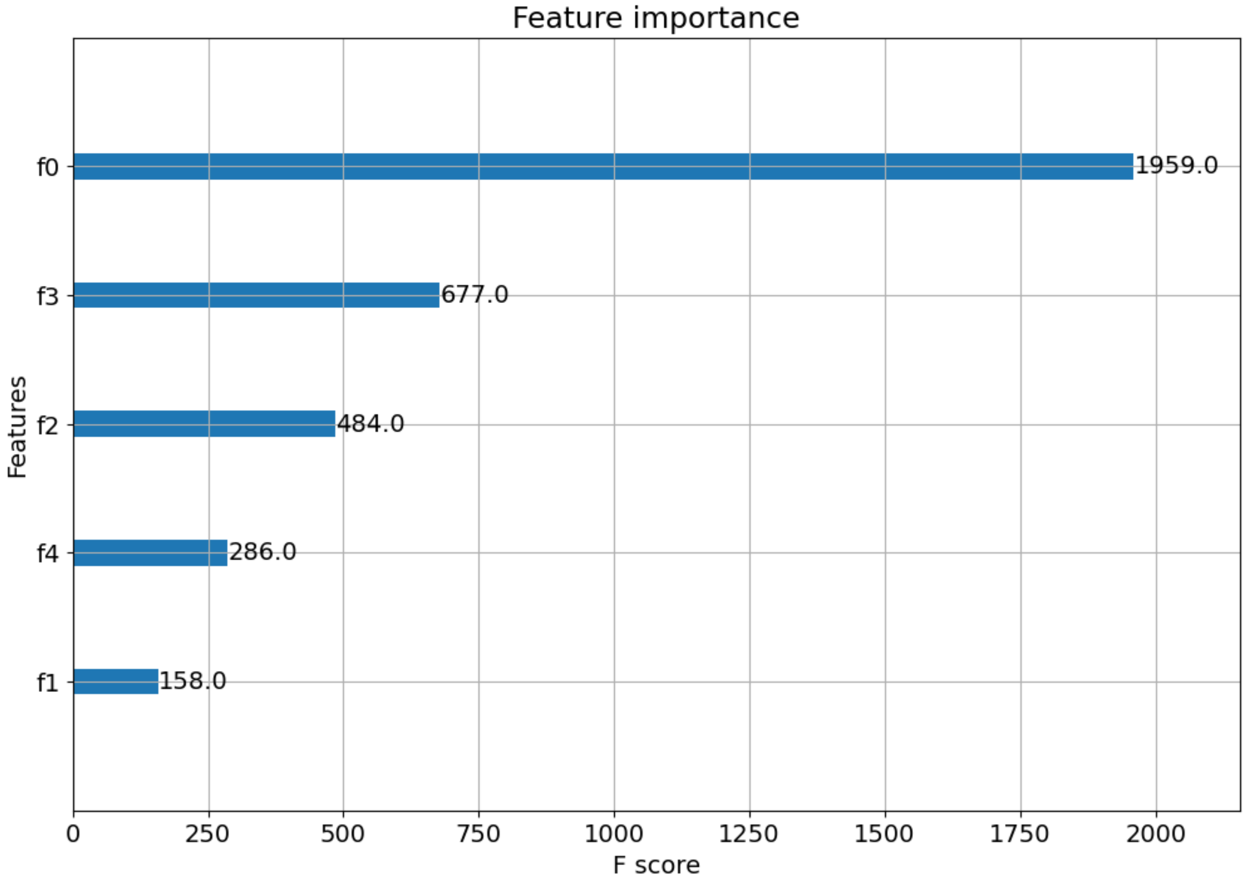
Train test split → min max scaler → feature importance

[Input]:

```
train_size=0.80

X = dataset.iloc[:,:-1]
y = dataset.iloc[:,-1]

# In the first step we will split the data in training and remaining dataset
X_train, X_test, y_train, y_test = train_test_split(X,y,
train_size=train_size,random_state=3,shuffle=False)


print(X_train.shape), print(y_train.shape)
#print(X_valid.shape), print(y_valid.shape)
print(X_test.shape), print(y_test.shape)

#min max scaler for random forest:

X_scaler = MinMaxScaler()
X_scaler.fit(X_train)

X_train_normalized = X_scaler.transform(X_train)

print('Feature Minimums: ', X_train_normalized.min(axis=0))
print('Feature Maximums: ', X_train_normalized.max(axis=0))

y_train.shape

y_scaler = MinMaxScaler()
y_train_normalized = y_scaler.fit_transform(y_train.values.reshape(-1,1))

y_train_normalized.shape


print('Feature Minimums: ', y_train_normalized.min(axis=0))
print('Feature Maximums: ', y_train_normalized.max(axis=0))

#feature importance random tress:

features = dataset.columns[:-1]
plt.rcParams.update({'figure.figsize': (12.0, 8.0)})
plt.rcParams.update({'font.size': 14})

rf = RandomForestRegressor(n_estimators=100)
rf.fit(X_train_normalized, y_train_normalized)

sorted_idx = rf.feature_importances_.argsort()
plt.barh(dataset.columns[:-1][sorted_idx],
rf.feature_importances_[sorted_idx])
plt.xlabel("Random Forest Feature Importance")
```

[output]:

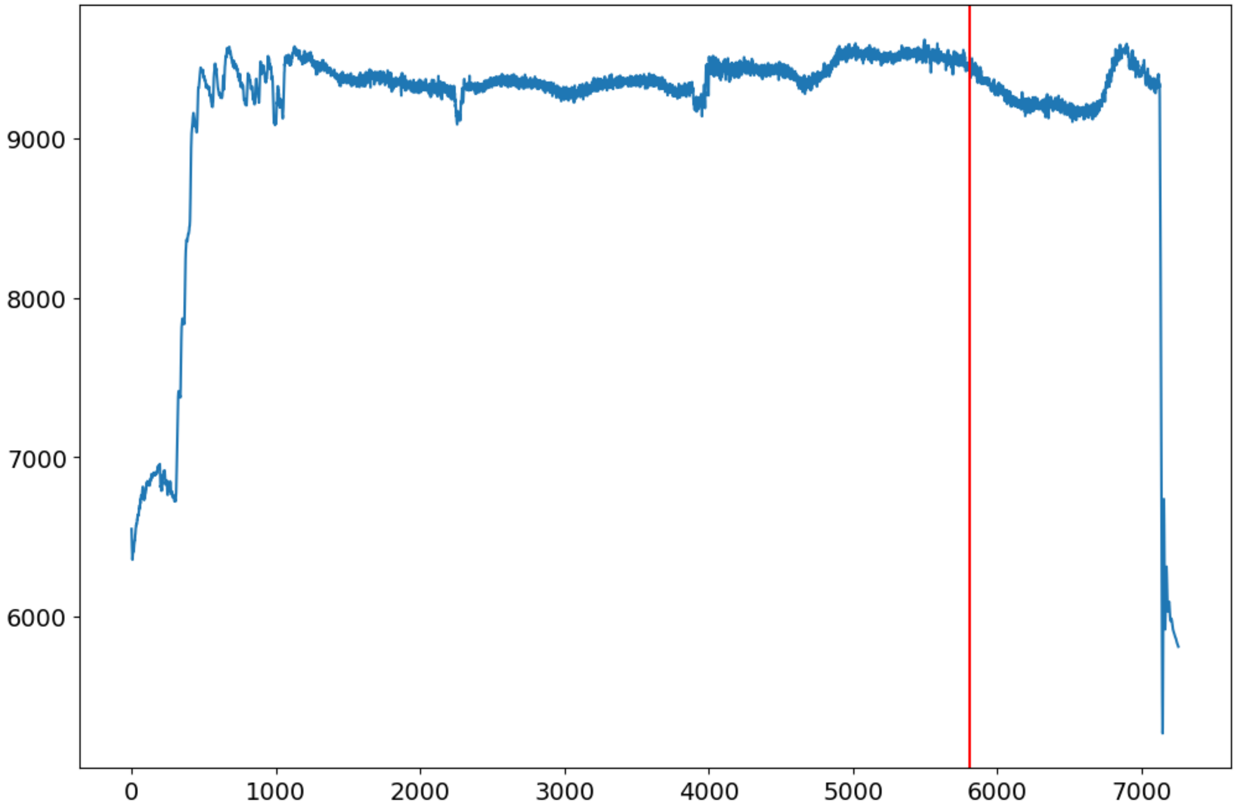
```
(5805, 5)
(5805,)
(1452, 5)
(1452,)
Feature Minimums:  [0. 0. 0. 0. 0.]
Feature Maximums:  [1. 1. 1. 1. 1.]
Feature Minimums:  [0.]
Feature Maximums:  [1.]
```



[Input]:

```
plt.plot(range(0, len(X)),y)

train_coords = train_size * len(X)

# x coordinates for the lines
xcoords = [train_coords]
# colors for the lines
colors = ['r']

for xc,c in zip(xcoords,colors):
    plt.axvline(x=xc, label='line at x = {}'.format(xc), c=c)
```

[Output]:



XGBOOST MODEL:

[Input]:

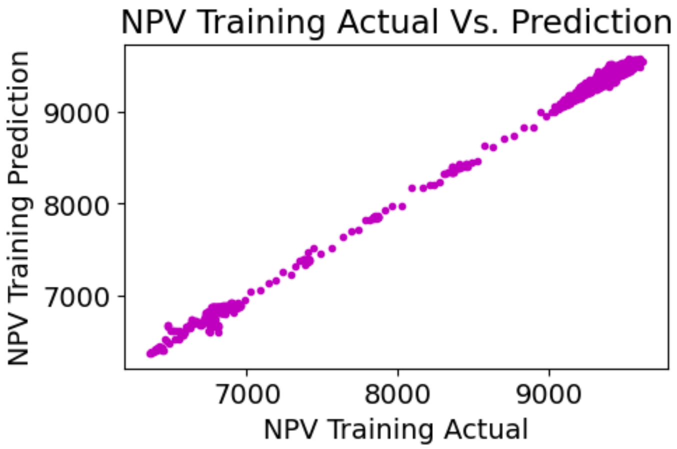
```
pipe_XGB = Pipeline([('scaler', MinMaxScaler()), ('XGB', XGBRegressor())])
# pipe_XGB.fit(X_train, y_train)
# pipe_XGB.score(X_test, y_test)

parameters = {'XGB__nthread':[1], #when use hyperthread, xgboost may become
slower
              'XGB__objective':['reg:squarederror'],
              'XGB__learning_rate': [0.7], #so called `eta` value
              'XGB__max_depth': range(1, 11, 2),
              'XGB__min_child_weight': [0.5],
              'XGB__subsample': [0.6],
              'XGB__colsample_bytree': [0.1, 0.5],
              'XGB__n_estimators': [500]}

xgb_grid = GridSearchCV(pipe_XGB,
                        parameters,
                        cv = 2,
                        n_jobs = 5,
                        verbose=True)

xgb_grid.fit(X_train, y_train)

print(xgb_grid.best_score_)
print(xgb_grid.best_params_)

y_pred_train=xgb_grid.predict(X_train)
y_pred_test=xgb_grid.predict(X_test)

corr_train=np.corrcoef(y_train, y_pred_train) [0,1]
print('Training Data R^2=',round(corr_train**2,4),'R=',
round(corr_train,4))

corr_train=np.corrcoef(y_test, y_pred_test) [0,1]
print('Testing Data R^2=',round(corr_train**2,4),'R=',
round(corr_train,4))

plt.figure(figsize=(5,3))
plt.plot(y_train, y_pred_train, 'm.')
plt.xlabel('NPV Training Actual')
plt.ylabel('NPV Training Prediction')
plt.title('NPV Training Actual Vs. Prediction')

plt.figure(figsize=(5,3))
plt.plot(y_test, y_pred_test, 'm.')
plt.xlabel('NPV Testing Actual')
plt.ylabel('NPV Testing Prediction')
plt.title('NPV Testing Actual Vs. Prediction')

#Error Analysis:
print('MAE:', round(metrics.mean_absolute_error(y_test,
y_pred_test),5))
print('MSE:', round(metrics.mean_squared_error(y_test, y_pred_test),5))
print('RMSE:', round(np.sqrt(metrics.mean_squared_error(y_test,
y_pred_test)),5))


#----------------------------------------------------------------------------
--------------------
import matplotlib.pyplot as plt

x = range(len(y_test))
y = y_test

fig, host = plt.subplots(figsize=(20,10))

par1 = host.twinx()
par2 = host.twinx()
par3 = host.twinx()
par4 = host.twinx()
par5 = host.twinx()

major_ticks = np.arange(0, 10001, 500)
minor_ticks = np.arange(0, 10001, 100)

host.set_ylim(0, 10000 )


par1.set_ylim(0, 10000)
par2.set_ylim(0, 140)
par3.set_ylim(0, 3)
par4.set_ylim(0, 3)
par5.set_ylim(-500, 500)

#host.set_xticks(major_ticks)
#host.set_xticks(minor_ticks, minor=True)
host.set_yticks(major_ticks)
host.set_yticks(minor_ticks, minor=True)

# And a corresponding grid
host.grid(which='both')

# Or if you want different settings for the grids:
```

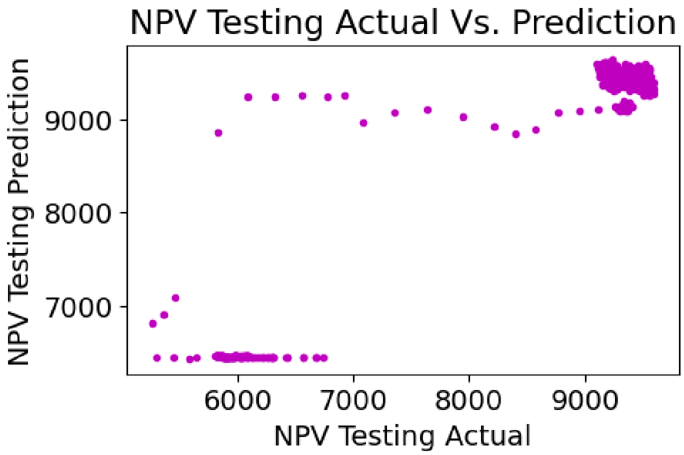
```
host.grid(which='minor', alpha=0.2)
host.grid(which='major', alpha=0.5)


# host.yaxis.grid(True, color ="black")
# host.xaxis.grid(True, color ="black")

host.set_xlabel("Time")
host.set_ylabel("y_test")
#par1.set_ylabel("y_Prediction")
par2.set_ylabel("Rate")
par3.set_ylabel("FR")
par4.set_ylabel("Sand Conc")
par5.set_ylabel("Delta P")

color1 = plt.cm.viridis(0)
color2 = plt.cm.viridis(0.5)
color3 = plt.cm.viridis(0.9)
color4 = plt.cm.viridis(.3)
color5 = plt.cm.viridis(.1)
color6 = plt.cm.viridis(.1)

fr_reduced = X_test['Formula Server 2.FR']*0.3
copyX_test = X_test.copy()

copyX_test['Formula Server 2.FR'] = fr_reduced

copyy_pred_test=xgb_grid.predict(copyX_test)

p1, = host.plot(x, y_test, color=color1, label="y_Test")
p2, = par1.plot(x, copyy_pred_test, color=color2, label="y_prediction")
p3, = par2.plot(x, X_test['Blender 113.DISCHARGE'], color=color3,
label="Rate")
p4, = par3.plot(x, fr_reduced, color=color4, label="FR")
p5, = par4.plot(x, X_test['Formula Server 2.Blender PPA'], color=color5,
label="Sand conc")
p6, = par5.plot(x,copyy_pred_test - y_test,label="Delta P")


lns = [p1, p2,p3,p4,p5, p6]

host.legend(handles=lns, loc='best')

par2.spines['left'].set_position(('outward', 60))
par3.spines['right'].set_position(('outward', 60))
par4.spines['right'].set_position(('outward', 60))
par5.spines['left'].set_position(('outward', 60))


host.yaxis.label.set_color(p1.get_color())
par1.yaxis.label.set_color(p2.get_color())
par2.yaxis.label.set_color(p3.get_color())
```

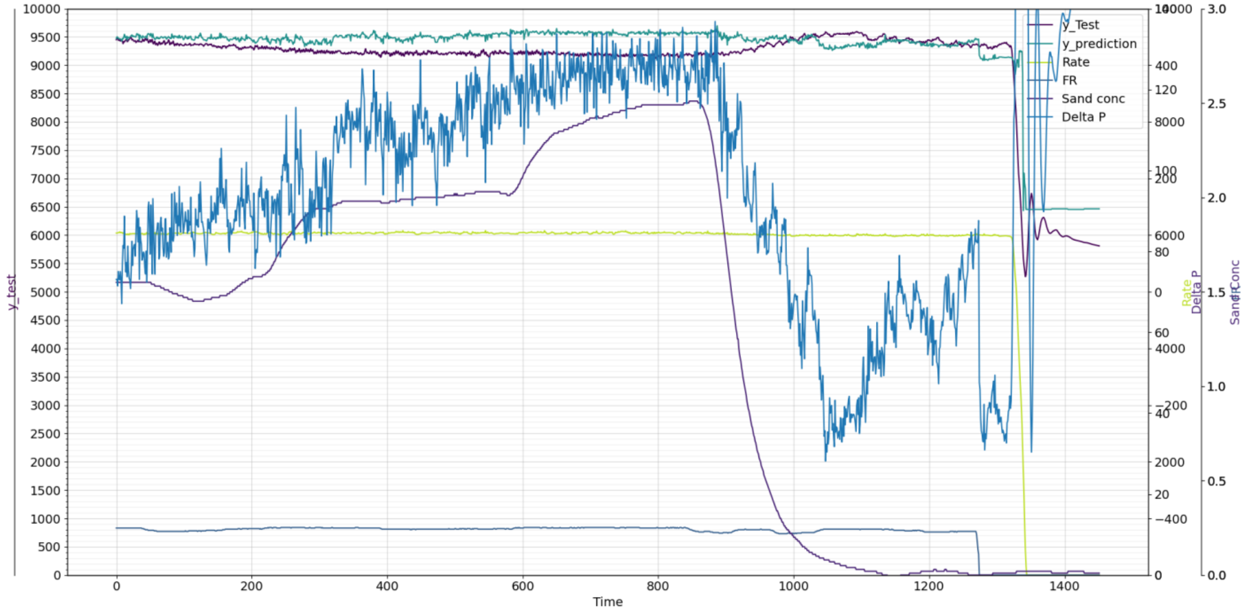
```
par3.yaxis.label.set_color(p4.get_color())
par4.yaxis.label.set_color(p5.get_color())
par5.yaxis.label.set_color(p5.get_color())
fig.tight_layout()
```

[output]:

```
Fitting 2 folds for each of 10 candidates, totalling 20 fits
-0.015597501644310063
{'XGB__colsample_bytree': 0.5, 'XGB__learning_rate': 0.7, 'XGB__max_depth':
3, 'XGB__min_child_weight': 0.5, 'XGB__n_estimators': 500, 'XGB__nthread': 1,
'XGB__objective': 'reg:squarederror', 'XGB__subsample': 0.6}
Training Data R^2= 0.998 R= 0.999
Testing Data R^2= 0.9112 R= 0.9546
MAE: 246.14608
MSE: 115848.65259
RMSE: 340.36547
```




NPV Testing Actual Vs. Prediction
NPV Testing Prediction
7000
8000
9000
6000
7000
8000
9000
NPV Testing Actual


y_test
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
5500
6000
6500
7000
7500
8000
8500
9000
9500
10000
0
200
400
600
800
1000
1200
1400
Time
y_Test
y_prediction
Rate
FR
Sand conc
Delta P
Rate
Delta P
Sand Conc
FR
0.0
0.5
1.0
1.5
2.0
2.5
3.0

Neural Network:

[Input]

```
pipe_MLP = Pipeline([('scaler', MinMaxScaler()), ('MLP',
MLPRegressor(hidden_layer_sizes=(50,),early_stopping=True,
                         validation_fraction=0.1, tol=1e-8,
n_iter_no_change=50,
                         random_state=1, max_iter=1000, verbose =True))])

# Check number of samples in train and test sets
print('Number of All Samples =', X.shape[0])
print('Number of Train Samples =', X_train.shape[0])
print('Number of Test Samples =', X_test.shape[0])


# Train the neural network model using the normalized input and target data
pipe_MLP.fit(X_train,y_train)


# Save the Neural Network Model
pickle.dump(pipe_MLP, open('Saved_Model_TEST1.pkl','wb'))

# Save the Input Scaler Object
pickle.dump(X_scaler, open('X_Min_Max_TEST1.pkl','wb'))

# Save the Target Scaler Object
pickle.dump(y_scaler, open('y_Min_Max_TEST1.pkl','wb'))
```

```
pipe_MLP = Pipeline([('scaler', MinMaxScaler()), ('MLP',
MLPRegressor(hidden_layer_sizes=(500, 100, 100, 100), max_iter =1000,
random_state = 20, verbose = False,
                        solver = 'adam', early_stopping = True,
validation_fraction = 0.1, activation = 'relu', tol = 1e-8, n_iter_no_change
= 50))])

pipe_MLP.fit(X_train, y_train)

predicted_train= pipe_MLP.predict(X_train)


# X_test_normalized = X_scaler.transform(X_test)

predicted_test= pipe_MLP.predict(X_test)

# predicted_test =
y_scaler.inverse_transform(prediction_test_normalized.reshape(-1,1))
```

```
MAE_test = metrics.mean_absolute_error(y_test,predicted_test)
MSE_test = metrics.mean_squared_error(y_test,predicted_test)
RMSE_test = MSE_test ** 0.5

print('Test MAE = ',MAE_test)
print('Test MSE = ',MSE_test)
print('Test RMSE = ',RMSE_test)

print('training R2 = ', metrics.r2_score(y_train, predicted_train))
print('test R2 = ', metrics.r2_score(y_test, predicted_test))

x = range(len(y_test))
y = y_test
```

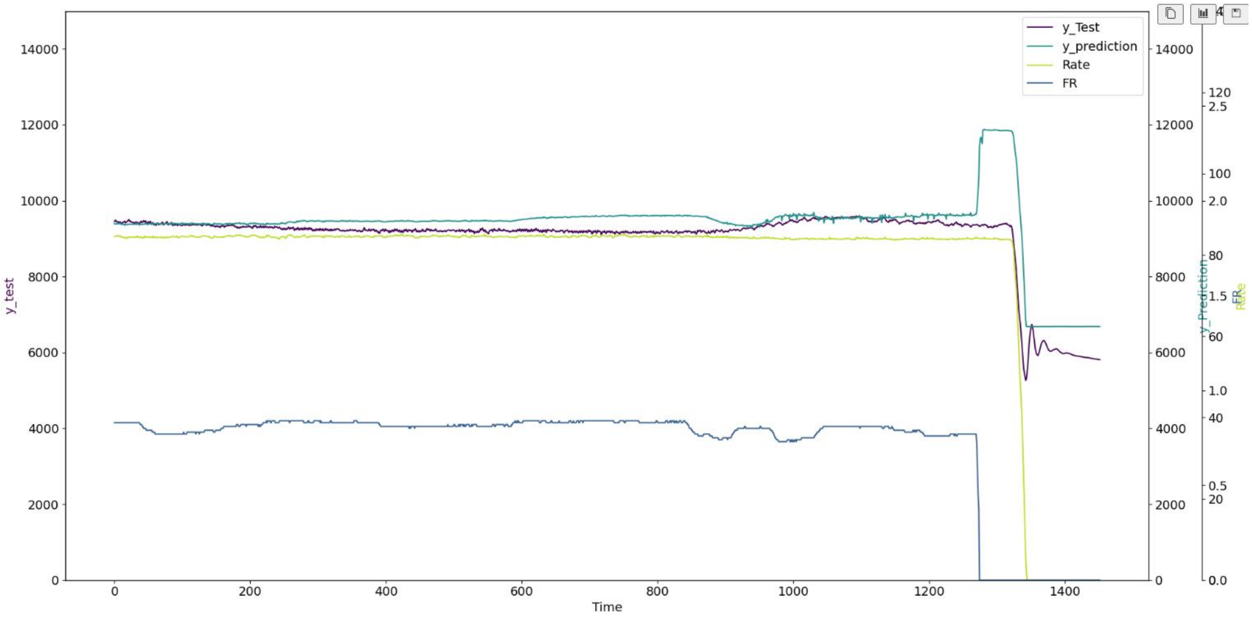
[Output]:

```
Test MAE =  353.8359

Test MSE =  402584.3001

Test RMSE =  634.495

training R2 =  0.97854

test R2 =  0.52133
```

[Input]:

```
import matplotlib.pyplot as plt

fig, host = plt.subplots(figsize=(20,10))

par1 = host.twinx()
par2 = host.twinx()
par3 = host.twinx()

host.set_ylim(0, 15000)

par1.set_ylim(0, 15000)
par2.set_ylim(0, 140)
par3.set_ylim(0, 3)


host.set_xlabel("Time")
host.set_ylabel("y_test")
par1.set_ylabel("y_Prediction")
par2.set_ylabel("Rate")
par3.set_ylabel("FR")

color1 = plt.cm.viridis(0)
color2 = plt.cm.viridis(0.5)
color3 = plt.cm.viridis(0.9)
color4 = plt.cm.viridis(.3)


p1, = host.plot(x, y_test, color=color1, label="y_Test")
p2, = par1.plot(x, predicted_test, color=color2, label="y_prediction")
p3, = par2.plot(x, X_test['Blender 113.DISCHARGE'], color=color3,
label="Rate")
p4, = par3.plot(x, X_test['Formula Server 2.FR'], color=color4, label="FR")


lns = [p1, p2,p3,p4]

host.legend(handles=lns, loc='best')

par2.spines['right'].set_position(('outward', 60))
par3.spines['right'].set_position(('outward', 60))


host.yaxis.label.set_color(p1.get_color())
par1.yaxis.label.set_color(p2.get_color())
par2.yaxis.label.set_color(p3.get_color())
par3.yaxis.label.set_color(p4.get_color())


fig.tight_layout()
```

[Output]:


y_Test
y_prediction
Rate
FR
y_test
14000
12000
10000
8000
6000
4000
2000
0
Time
0
200
400
600
800
1000
1200
1400
y_Prediction
FR
Rate
120
2.5
100
2.0
80
1.5
60
1.0
40
0.5
20
0.0

SVM:

[Input]

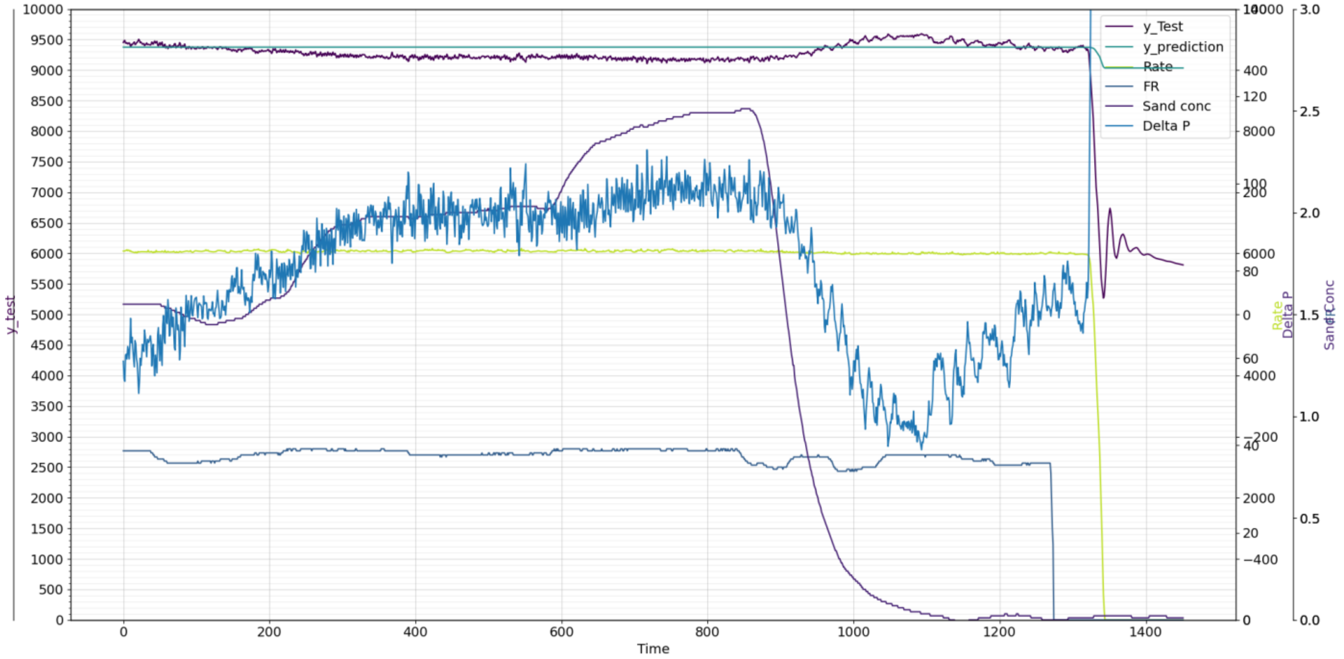
```
import numpy as np
from sklearn.svm import SVR
from sklearn.metrics import r2_score
from sklearn.metrics import r2_score, mean_squared_error

# assuming you have training data X_train and labels y_train

# create SVR object
reg = SVR(kernel='rbf')

# train the model
reg.fit(X_train, y_train)

# assuming you have test data X_test
# make predictions
y_pred_train = reg.predict(X_train)
y_pred_test = reg.predict(X_test)

# calculate R-squared value for training and test
r2_train = r2_score(y_train, y_pred_train)
r2_test = r2_score(y_test, y_pred_test)


rmse_test = np.sqrt(mean_squared_error(y_test, y_pred_test))

print("R-squared value for test: ", r2_test)
print("RMSE for test: ", rmse_test)

print("R-squared value for training: ", r2_train)
print("R-squared value for test: ", r2_test)
#-----------------------------------------------------------------------------
------

import matplotlib.pyplot as plt

x = range(len(y_test))
y = y_test

fig, host = plt.subplots(figsize=(20,10))

par1 = host.twinx()
par2 = host.twinx()
par3 = host.twinx()
par4 = host.twinx()
par5 = host.twinx()

major_ticks = np.arange(0, 10001, 500)
minor_ticks = np.arange(0, 10001, 100)
```

```
host.set_ylim(0, 10000 )


par1.set_ylim(0, 10000)
par2.set_ylim(0, 140)
par3.set_ylim(0, 3)
par4.set_ylim(0, 3)
par5.set_ylim(-500, 500)

#host.set_xticks(major_ticks)
#host.set_xticks(minor_ticks, minor=True)
host.set_yticks(major_ticks)
host.set_yticks(minor_ticks, minor=True)

# And a corresponding grid
host.grid(which='both')

# Or if you want different settings for the grids:
host.grid(which='minor', alpha=0.2)
host.grid(which='major', alpha=0.5)


# host.yaxis.grid(True, color ="black")
# host.xaxis.grid(True, color ="black")

host.set_xlabel("Time")
host.set_ylabel("y_test")
#par1.set_ylabel("y_Prediction")
par2.set_ylabel("Rate")
par3.set_ylabel("FR")
par4.set_ylabel("Sand Conc")
par5.set_ylabel("Delta P")

color1 = plt.cm.viridis(0)
color2 = plt.cm.viridis(0.5)
color3 = plt.cm.viridis(0.9)
color4 = plt.cm.viridis(.3)
color5 = plt.cm.viridis(.1)
color6 = plt.cm.viridis(.1)

fr_reduced = X_test['Formula Server 2.FR']
copyX_test = X_test.copy()

copyX_test['Formula Server 2.FR'] = fr_reduced

#copyy_pred_test=xgb_grid.predict(copyX_test)

p1, = host.plot(x, y_test, color=color1, label="y_Test")
p2, = par1.plot(x, y_pred_test, color=color2, label="y_prediction")
p3, = par2.plot(x, X_test['Blender 113.DISCHARGE'], color=color3,
label="Rate")
```

```
p4, = par3.plot(x, fr_reduced, color=color4, label="FR")
p5, = par4.plot(x, X_test['Formula Server 2.Blender PPA'], color=color5,
label="Sand conc")
p6, = par5.plot(x,y_pred_test - y_test,label="Delta P")


lns = [p1, p2,p3,p4,p5, p6]

host.legend(handles=lns, loc='best')

par2.spines['left'].set_position(('outward', 60))
par3.spines['right'].set_position(('outward', 60))
par4.spines['right'].set_position(('outward', 60))
par5.spines['left'].set_position(('outward', 60))


host.yaxis.label.set_color(p1.get_color())
par1.yaxis.label.set_color(p2.get_color())
par2.yaxis.label.set_color(p3.get_color())
par3.yaxis.label.set_color(p4.get_color())
par4.yaxis.label.set_color(p5.get_color())
par5.yaxis.label.set_color(p5.get_color())
fig.tight_layout()
```

[Output]

```
R-squared value for test:  0.060121732984577125
RMSE for test:  889.0947560478603
R-squared value for training:  0.18605536102028886
R-squared value for test:  0.060121732984577125
```



Multiregression:

[Input]

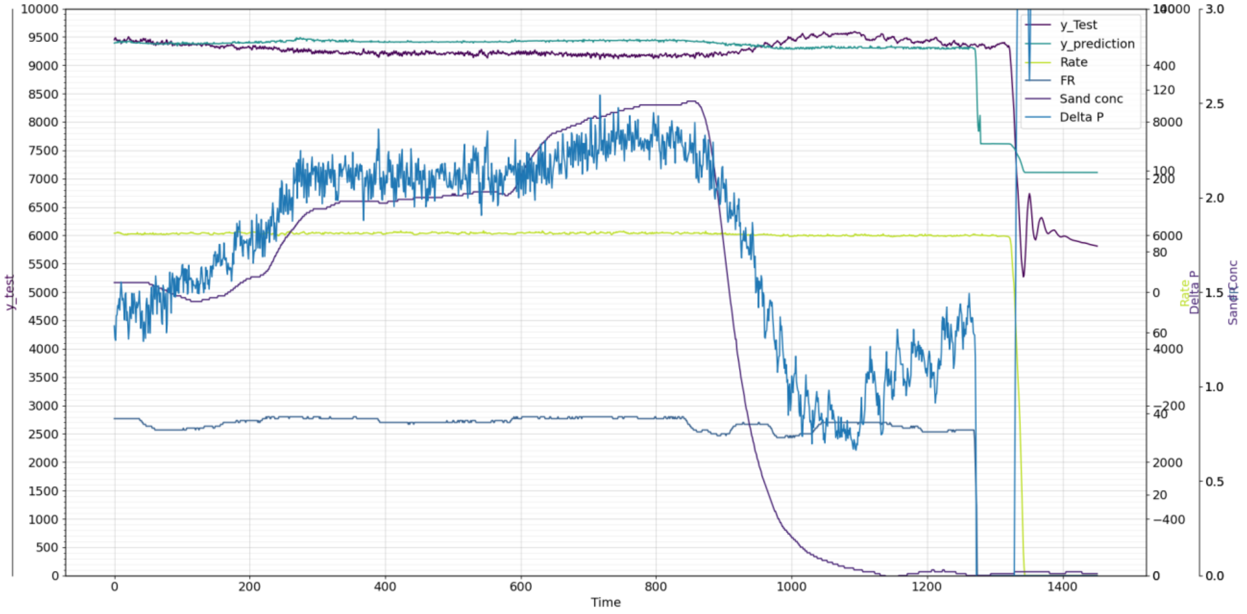
```
##Multilinear regression:

import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# assuming you have training data X_train and labels y_train

# create multilinear regression object
reg = LinearRegression()

# train the model
reg.fit(X_train, y_train)

# assuming you have test data X_test
# make predictions
y_pred_train = reg.predict(X_train)
y_pred_test = reg.predict(X_test)

# calculate R-squared value for training and test
r2_train = r2_score(y_train, y_pred_train)
r2_test = r2_score(y_test, y_pred_test)

print("R-squared value for training: ", r2_train)
print("R-squared value for test: ", r2_test)

#R^2 = 0.539 (could be better) closer to 1.00

#-------------------------------------------------------------------
import matplotlib.pyplot as plt

x = range(len(y_test))
y = y_test

fig, host = plt.subplots(figsize=(20,10))

par1 = host.twinx()
par2 = host.twinx()
par3 = host.twinx()
par4 = host.twinx()
par5 = host.twinx()

major_ticks = np.arange(0, 10001, 500)
minor_ticks = np.arange(0, 10001, 100)

host.set_ylim(0, 10000 )
```

```
par1.set_ylim(0, 10000)
par2.set_ylim(0, 140)
par3.set_ylim(0, 3)
par4.set_ylim(0, 3)
par5.set_ylim(-500, 500)

#host.set_xticks(major_ticks)
#host.set_xticks(minor_ticks, minor=True)
host.set_yticks(major_ticks)
host.set_yticks(minor_ticks, minor=True)

# And a corresponding grid
host.grid(which='both')

# Or if you want different settings for the grids:
host.grid(which='minor', alpha=0.2)
host.grid(which='major', alpha=0.5)


# host.yaxis.grid(True, color ="black")
# host.xaxis.grid(True, color ="black")

host.set_xlabel("Time")
host.set_ylabel("y_test")
#par1.set_ylabel("y_Prediction")
par2.set_ylabel("Rate")
par3.set_ylabel("FR")
par4.set_ylabel("Sand Conc")
par5.set_ylabel("Delta P")

color1 = plt.cm.viridis(0)
color2 = plt.cm.viridis(0.5)
color3 = plt.cm.viridis(0.9)
color4 = plt.cm.viridis(.3)
color5 = plt.cm.viridis(.1)
color6 = plt.cm.viridis(.1)

fr_reduced = X_test['Formula Server 2.FR']
copyX_test = X_test.copy()

copyX_test['Formula Server 2.FR'] = fr_reduced

#copyy_pred_test=xgb_grid.predict(copyX_test)

p1, = host.plot(x, y_test, color=color1, label="y_Test")
p2, = par1.plot(x, y_pred_test, color=color2, label="y_prediction")
p3, = par2.plot(x, X_test['Blender 113.DISCHARGE'], color=color3,
label="Rate")
p4, = par3.plot(x, fr_reduced, color=color4, label="FR")
p5, = par4.plot(x, X_test['Formula Server 2.Blender PPA'], color=color5,
label="Sand conc")
p6, = par5.plot(x,y_pred_test - y_test,label="Delta P")
```

```
lns = [p1, p2,p3,p4,p5, p6]

host.legend(handles=lns, loc='best')

par2.spines['left'].set_position(('outward', 60))
par3.spines['right'].set_position(('outward', 60))
par4.spines['right'].set_position(('outward', 60))
par5.spines['left'].set_position(('outward', 60))


host.yaxis.label.set_color(p1.get_color())
par1.yaxis.label.set_color(p2.get_color())
par2.yaxis.label.set_color(p3.get_color())
par3.yaxis.label.set_color(p4.get_color())
par4.yaxis.label.set_color(p5.get_color())
par5.yaxis.label.set_color(p5.get_color())
fig.tight_layout()
```

[Output]

```
R-squared value for training:  0.8886597816929209
R-squared value for test:  0.7142611772685057
```


y_test
Time
y_Test
y_prediction
Rate
FR
Sand conc
Delta P
Rate
Delta P
FR
Sand Conc