Results and Discussion

In this section, I will elaborate (1) details of how I implement this project, (2) what results did I find during the implementations, and (3) what conclusions could I get from the results.

Dataset Preprocessing

In the petroleum production field, production data are recorded as time series data, as the format of rate versus month, precisely speaking. The data used in this project were extracted from DrillingInfo (DrillingInfo 2017), which is an official website specifically focusing on providing nation-wide oil and gas production data. I picked up 200 gas wells from Barnett Shale reservoir that are active in production. Picking up production wells from the Barnett Shale ensures that those wells are from the same geologic area.

The 200 wells have different starting production time in month, but share the same ending production time in May 2017, which results in the different feature length for them. Our first step is to label those 200 samples. For each well in the field, there is a typical production profile as shown in the figure 1. The data in figure 1 are from one of the 200 wells we will be using during this project. The red dots showed the trend of declining rate of production. With the time increasing, the production rate will decrease below a threshold, which is called “abandon rate”. Normally speaking, the life of one particular unconventional gas well should last for over 360 months before it decreased to its abandon rate. This gives us a clue to label the 200 wells. We label the wells by separating them into different type according to the P10, P50 and P90 values.

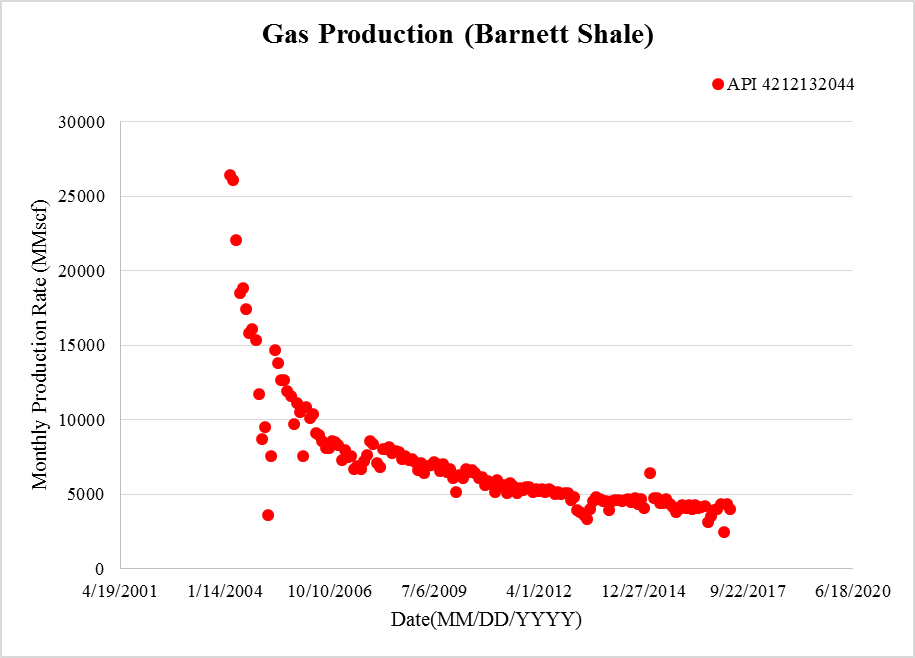


Figure Typical Well Production Profile

Forecasting

As we see from the figure 1, we have only limited data points (i.e. less than 170 months). For the sake of reliably labelling the available dataset, we used some mathematical models developed by researchers earlier in petroleum industry to implement the forecasting process. The forecasted data will be assumed as the ultimate estimate recovery (EUR), which will be further used in the labelling stage.

Power’s Law (Ilk, Rushing et al. 2008)

Where:

is the decline constant “intercept” at 1 time unit, 1/days

is the decline constant at infinite time, 1/days

is the time exponent, unitless

is the rate intercept bbl/day or Mcf/day

Stretched Exponential (Valko and Lee 2010)

Where:

is the production rate at any time

production rate at time = 0

is the characteristic time parameter

is the time exponent, unitless

Duong’s method (Duong, 2011)

Where:

is the cumulative production

is the production rate, vol/day

is the time, days

& are constants

Since we are dealing with type wells, we need to ensure that all the wells that we used to construct type wells have reached the boundary dominant flow (BDF). The reason is that before wells reach BDF, well production exhibit transient flow and it’s nearly impossible to forecast a well’s EUR given only transient flow data. The two examples of wells that do not exactly reach the boundary dominant flow are shown in figure 2. The two wells are selected from the 200 wells that we are going to use in this project.

In these two log-log plots, the blue lines show the raw data plotted in log-log scale. It is quite obvious that the slope of the decline trend line is 0.5 by the end of May 2017, which is indicated by the green line. This means that the two wells have not reach the BDF, which is specifically indicated by the red lines. In this way, I cannot use the data and the 3 models mentioned above to implement the forecasting directly.

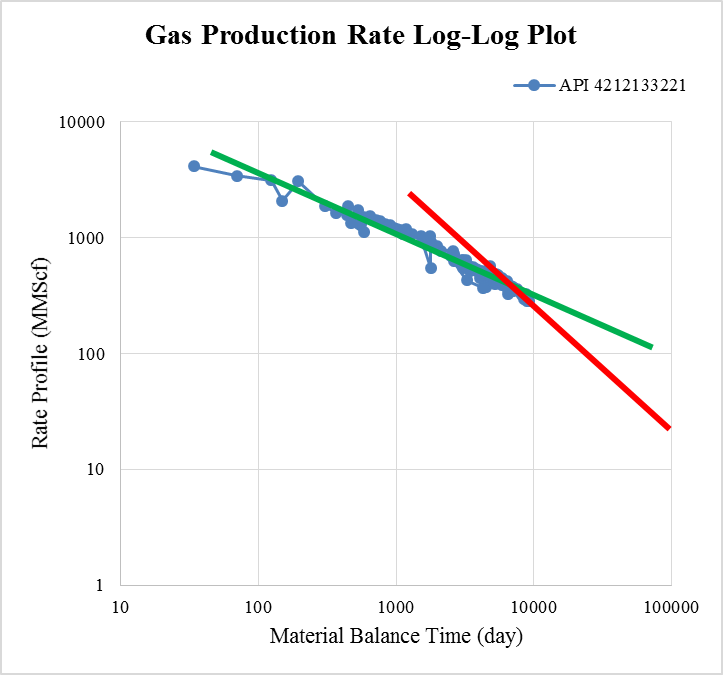
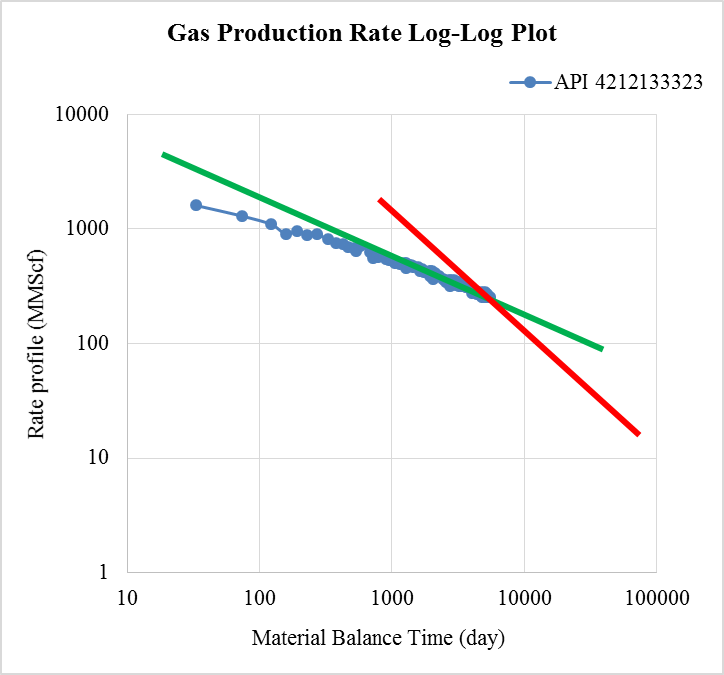
 

Figure Wells Not Reaching Boundary Dominant Flow

To resolve this problem, we implement the forecasting by implement extrapolation through specifying the switch point. In the software ValNav (ValNav 2017), we have three options to specify the switch point from linear flow to BDF: specifying the fixed decline slope (%/yr), specifying months after the start of first declining forecast segment, and specifying the months after start of history. Here we choose to specify the switch point by specifying the fixed decline slope (%/yr) at 6.5 as shown in figure 3.

After this setting, ValNav will automatically transfer into BDF mode after the decline rate (%/yr) of transient flow is identified. In this way, when it is implementing the forecasting, the production histories will be extended to 360 months, and thus we can make sure the 200 wells have reached the BDF by the end of the 360 months. This resolves the problem.

When using the software ValNav to predict the future production, we can easily choose the best fit from the three models for each well to do the forecasting. Figure 4 and figure 5 gave the GUI interface for the ValNav and BestFit, respectively. The BestFit automatically selects the most recent data and the most suitable model to fit data first and will implement the forecasting based on the selected data and model. The ValNav sometimes may give us data more than 360 months (E.g. when we only specify the abandon rate at 10 Mscf/d as we have done in this project), but we only use the 360 months data.

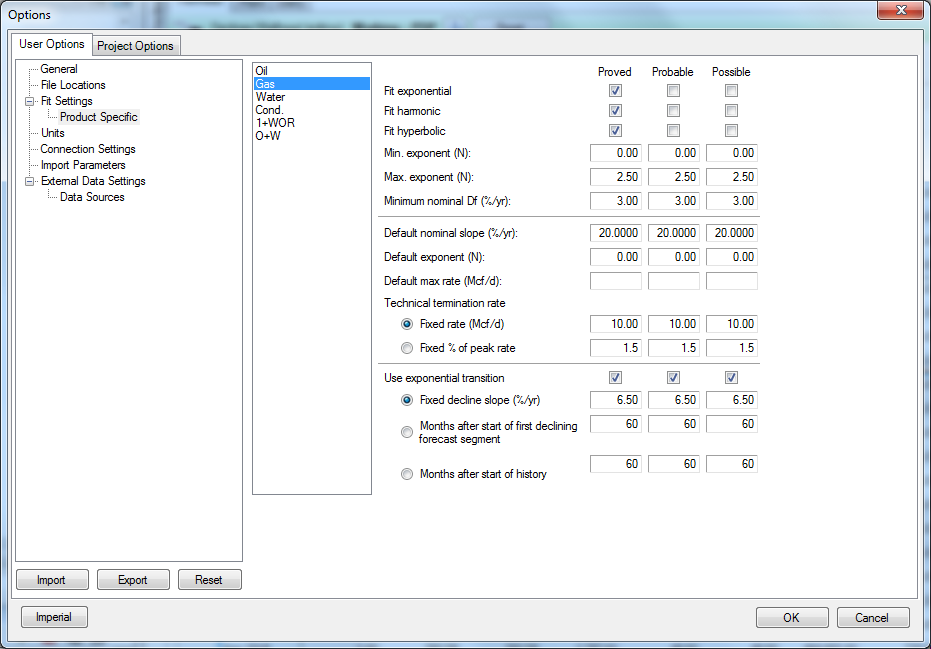


Figure Specifying the Switch Point in ValNav

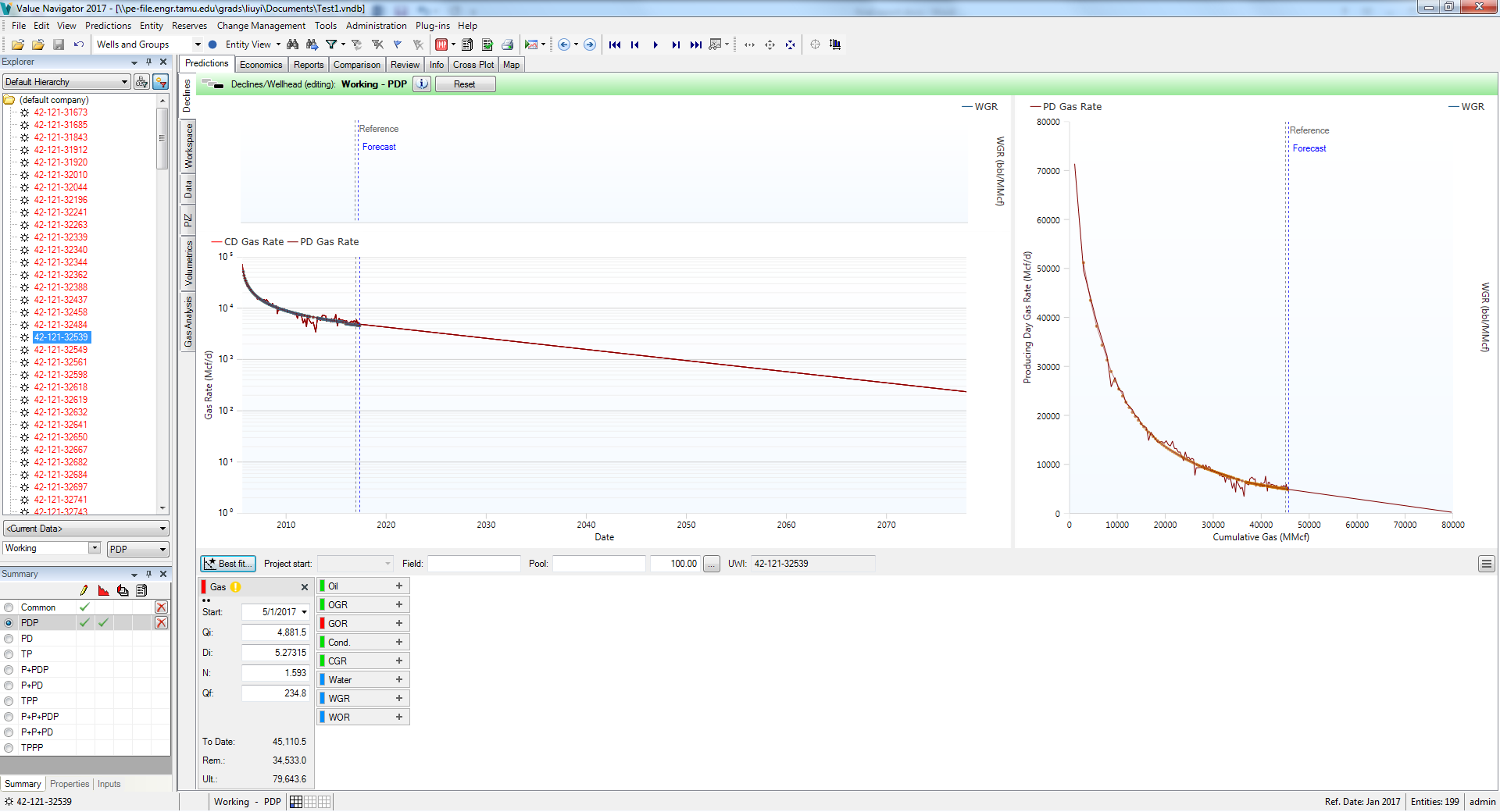


Figure ValNav Interface

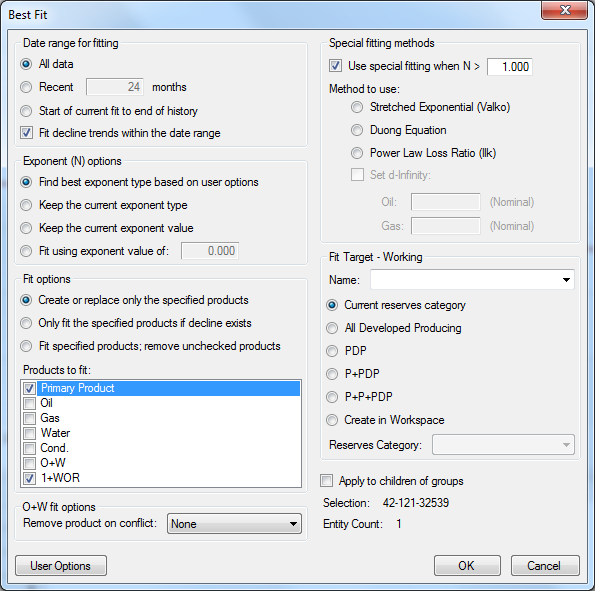


Figure Best Fit Interface in ValNav

After ValNav has completed the BestFit procedure, a typical production profile is shown in figure 4. The profile, as mentioned before, has a production life more than 360 months. However, we only need the first 360 months’ data. The EUR of each well is then thought to be the cumulative production at the end of 360 months.

LogNormal Distribution of EUR

With 200 EUR values given, I sort them in descending order accordingly. In this case, each EUR value for each well is corresponding to a “less than probability”. I plotted the EUR distribution in a log probability paper as shown in figure 6. The horizontal and vertical axis of figure 6 are EUR values, and “less than probability”, respectively, both in logarithmic scale. The majority of the EUR values are approximately located on a straight line, indicating a lognormal distribution of EUR values of wells from this geologic area.

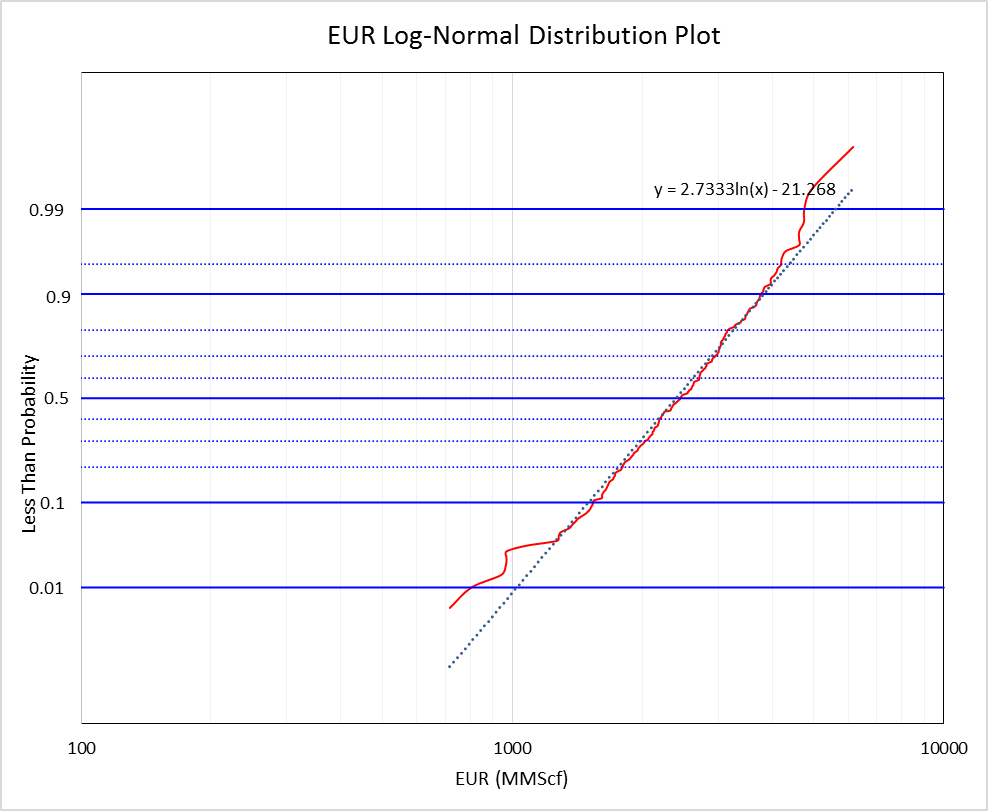


Figure EUR lognormal distribution

Type well construction and evaluation

Given that EUR is predicted by ValNav as shown in figure 6, this step focuses on constructing type wells. With the insurance of BDF and the EUR value for each well, I can simply find the P90, P50, and P10 type wells, which are 1538.053 MMscf, 2448.201 MMscf, and 3759.201, respectively. The P90 value is the EUR value corresponding to the less than probability of 0.1 in figure 6, and P50 and P10 corresponds to 0.5 and 0.9 respectively. From this, we can get the P10/P90 ratio at approximately 2.324, indicating a low distribution variance, which in turn manifest a low uncertainty in the distribution.

The minimal dispersion in this problem indicates a good probabilistic property of the type wells, which means that the type wells (P90 and P10) constructed can be a sound representative of wells in this geologic area. This is critical for us to reliably apply the type wells data generated in this step to our further usage.

Labeling

With the aid from ValNav, we have got the EUR and 360 months’ production data for each well. In our machine learning application process, each well’s 360 months’ production data will be the input of the machine learning algorithms, thus we need to label the 200 well samples using their EUR values in this section. The labels given to the well samples are called types. In this project, I separated the well samples into 4 types:

Type1 – Below P90 (EUR <= 1538.053 MMScf)

Type 2 – P50 ~ P90 (1538.053 MMScf <= EUR < 2448.201 MMScf)

Type 3 – P10 ~ P50 (2448.201 MMScf <= EUR < 3759.201 MMScf)

Type 4 – Above P10 (EUR >= 3759.201 MMScf)

Uniform Input Dimensionality

In the labelling section, I used each well’s cumulative production at the end of 360 months (i.e. EUR) as the criteria to label well samples with 4 different types mentioned above. In reality, when we classify the “relatively new” well samples with short history into one of the 4 types, we are dealing with those “relatively new” wells that don’t have a production history of 360 months. In this project, the well with the longest production history has a production history of 170 months.

To ensure that each input to the machine learning algorithms has the same dimension, I extrapolated each well’s raw data to 170 months. This procedure was completed simply by extracting the first 170 months’ production data from 360 months’ production data that were used in the labelling section. The 170 months’ production data will be the actual input when we train the classifiers using machine learning algorithms.

In the following clarification, the 200 well production data will be seen as the 200 samples, each has 170 features. This will be more easily understood in terms of machine learning fields.

4-fold Cross Validation Technique

Cross validation technique is commonly used to reduce the generalization error of machine learning algorithms. Before I implemented the machine learning algorithms to classify the wells with only short production history into one of the 4 types mentioned above, I separated the 200 samples into 4 groups. For each specific algorithm, each time I used the 3 of the 4 groups as the training set, and the only one group was left as the test set. I repeat this training process 4 times with different data group as the test set for each time. After that, the overall accuracy of this algorithm in our problem will be the average of the accuracies that we get from the 4 repeated training processes. In this way, the generalization error (i.e. the risk of overfitting) of the algorithm was minimized by not relying on any specific group of data that might be not representative of the whole dataset.

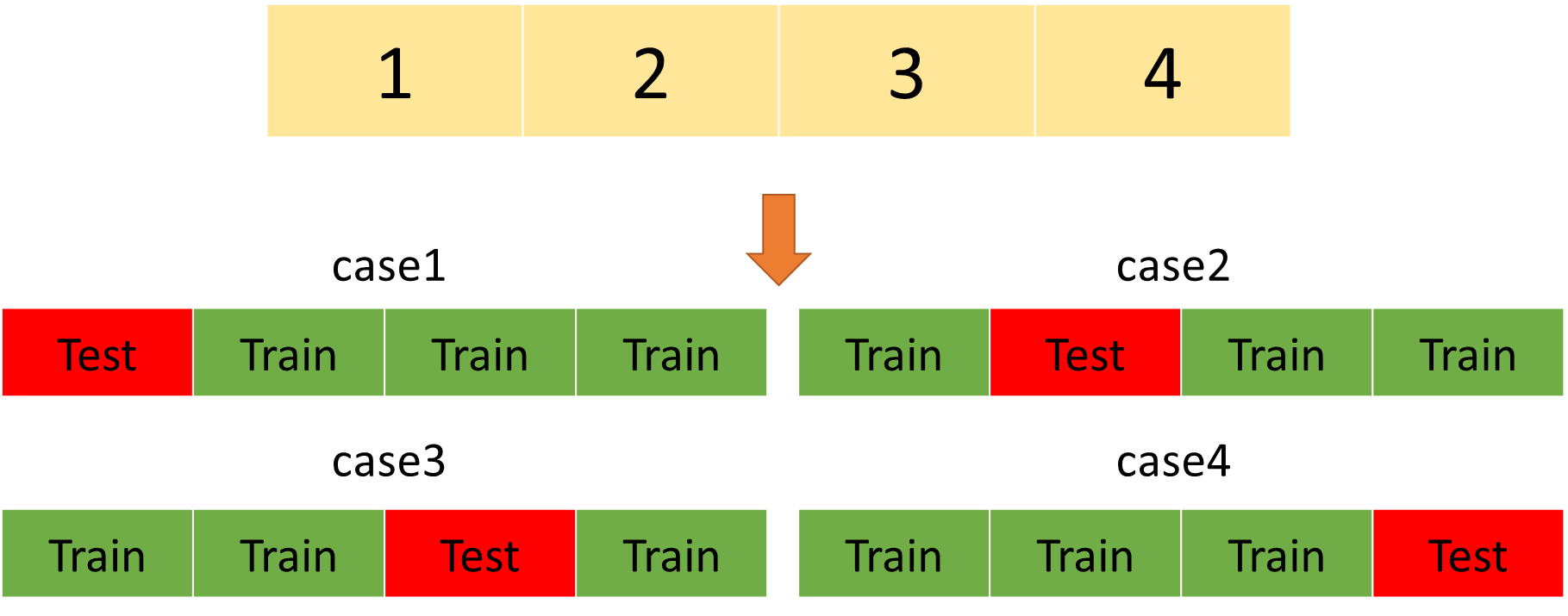


Figure Cross Validation Technique

Machine Learning Algorithm Applications

The machine learning algorithms I used include neural networks (NNet), support vector machine (SVM) and Random Forest (RF), the pros and cons of each algorithm are discussed individually. In this section, I will introduce in detail the principles of the 3 algorithms, how I implemented the algorithms, and the results and corresponding performance evaluations.

NNet

Neural network algorithm is becoming very popular in solving many regression and classification problems. There are some variants of neural networks: convolutional neural networks (CNN) which is commonly used in image recognition, recurrent neural networks (RNN) which is widely used in natural language processing (NLP), multi-layer perceptron (MLP), etc. MLP can deal with high-nonlinearity problems appropriately with suitable parameters setting. In addition, we don’t need to consider the dependence between each pair of variables, the number of parameters and hyper-parameters.

In this project, we simply chose the MLP as our training algorithm due to its simplicity to implement. The basic MLP architecture that in this project is constructed as shown in the figure 10.

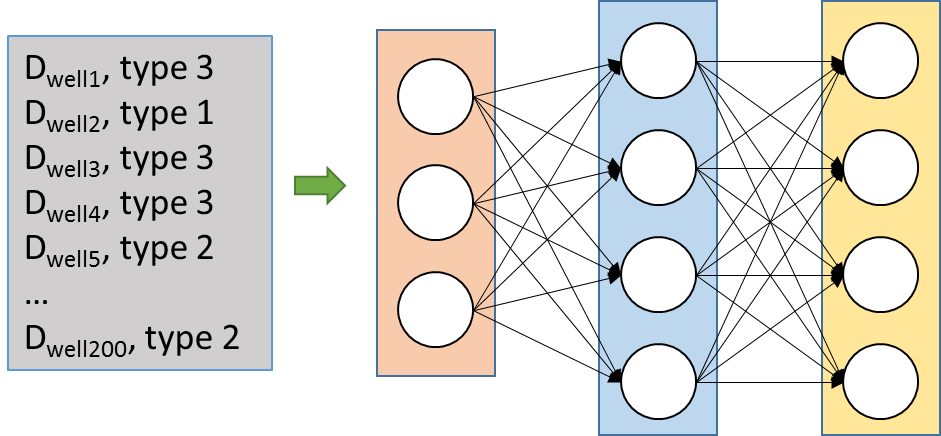


Figure Neural Networks Training Architecture

In figure 8, there is only 1 hidden layer in the architecture. The number of the input layer neurons, hidden layer neurons, and output layer neurons are 170, 163, and 4, respectively. The determination of hidden layer neurons numbers will be clarified later in detail. The classical logistic function was employed as forward activation function. In the backpropagation process, the weights are updated using lbgfs solver, which is one of the quasi-newton option. In addition, the learning rate and momentum parameters were set to be 0.1 and 0.5, respectively, to improve the overall performance of this architecture.

Before the data training, I normalized the 200 well samples. I used the min-max normalization for all the 200 samples to avoid abnormal feature values (e.g. some features are too low/high compared to others’). As shown in figure 9, each column represents a particular feature of samples, and each row represents a sample. In the min-max normalization, each column is selected, and the maximum and minimum value will be identified. The normalized value will be computed using the following equation. In this way, all the values will be converted into the value between [0, 1], and the risks of abnormal features are minimized.

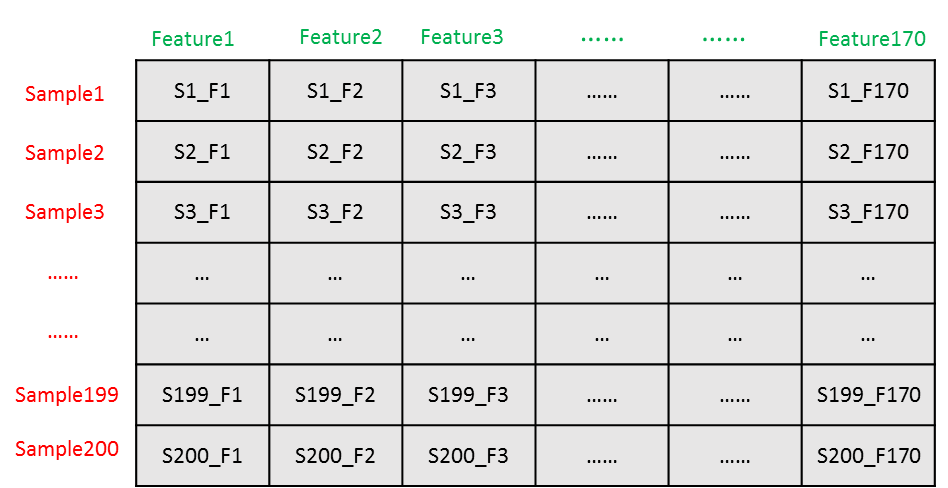


Figure Min-Max Normalization

The backpropagation algorithm is the core of the neural network algorithm. Figure 8 illustrates the back propagation algorithm schematically. It is based on a stochastic gradient descent algorithm.

Firstly, we initialize all weighs of all neuron connections to small random numbers. Those numbers may vary between (-1, 1). Secondly we enter into the repeated iteration. In each iteration, each of the 200 samples will be input into the algorithm to be fed forward. The feed forward process will produce an output, which is the type of each individual well. Either the output result of the feed forward process is or not the same with the type label that we have specifically assigned to the well, we will compute the error at the output layer. With the error from output layer, we can back propagate the error from output layer to input layer to update the weights accordingly until the stopping criteria is satisfied.

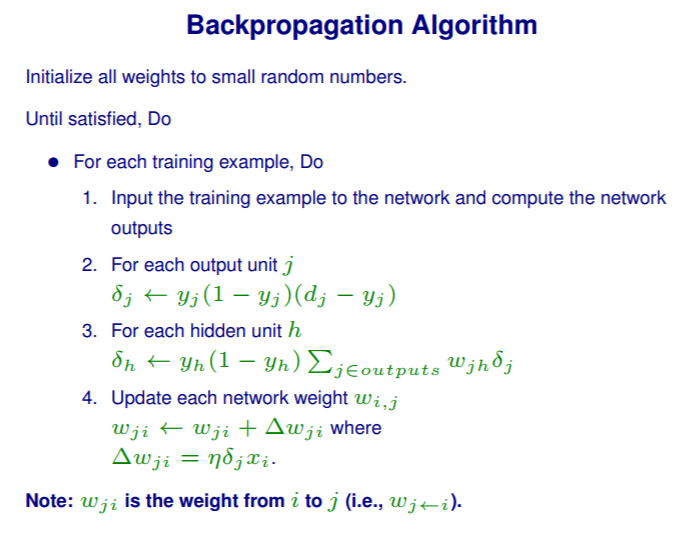


Figure Neural Network Backpropagation Algorithm

The algorithm shown in figure 10 is implemented using Python package scikit-learn (MLPClassifier, scikit-learn 2017), the results are shown in figure 11.

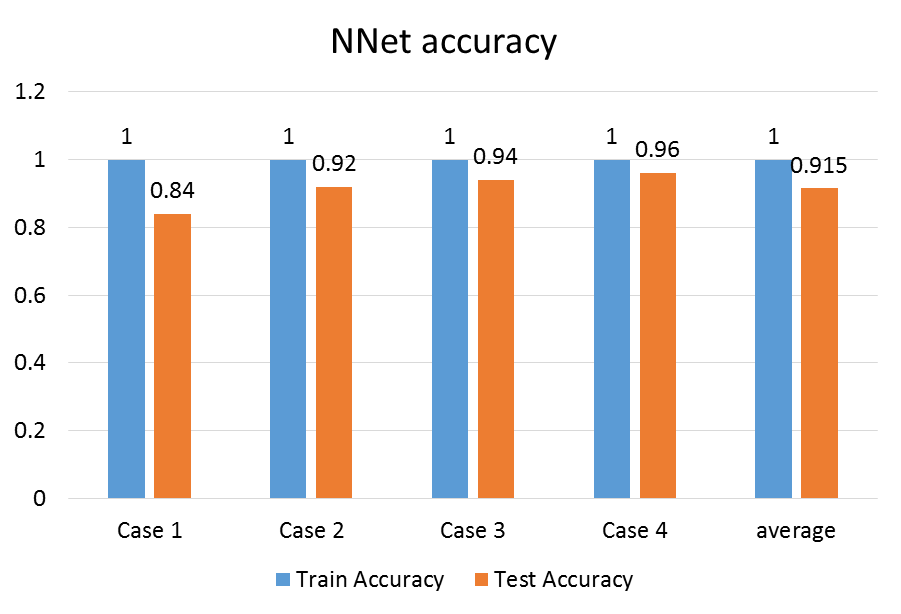


Figure Neural Network Algorithm Accuracy

In figure 11, we can see that MLP can achieve 100% training accuracy as we employ 4-fold cross validation techniques. The testing accuracy varies for different test cases, ranging from 0.84 to 0.96. The averaged accuracy of 0.915 was selected as the overall evaluation criteria for the MLP algorithm. The averaged accuracy is between minimal and maximum accuracy, we selected it to avoid the heavy dependence of algorithm’s performance on any individual test case. This is the advantage of cross validation as we have illustrated in the 4 fold cross validation process.

However, the main limitation is that MLP cannot guarantee the global minima solution, it may stick in a locally optimal solution and then stop updating the weights for neuron connections. A good option to mitigate this is to have random start position (e.g. randomly generate the weights for MLP each time) as what we have done in figure 10 to initialize all weights to small random numbers at first. Another possible option could be adding the momentum parameter as we implement the algorithm.

The performance of MLP can also be limited by the number of hidden layer neurons. Too many hidden neurons would cause overfitting problems, too few may lead to underfitting. This dilemma is resolved by choosing the most suitable value between the number of neurons of input layer and that of output neurons. As what I mentioned in the uniform input dimensionality section, each well’s production history was extrapolated to 170 months for the sake of uniform input dimensionality. So the input layer has 170 neurons, while the output layer has only 4 neurons. Thus the number of hidden layer neurons can be chosen from the interval [4, 170]. Figure 12 gives the relationship between the number of hidden neurons and test accuracy (i.e. the averaged 4 fold cross validation test accuracies).

As we can see clearly in figure 12, the accuracy varies greatly as the number of hidden neurons changes. The highest accuracy is reached at 0.955 when the number of hidden neurons comes to 163. Although the accuracy oscillates in figure 12, we can still see a general increasing trend as the hidden neurons increases. This might be due to the fact that the more the hidden neurons in MLP, the more non-linear relationships it can understand. As we add more hidden neurons into the neural network architecture, in theory, the model will become more complex, which will further decrease the accuracies of test accuracies. This is what we commonly called overfitting problem. However, the cross validation results in figure 12 show that more hidden neurons improve the accuracy, which implies even 170 hidden neurons may not cause model overfitting in our problem specifically.

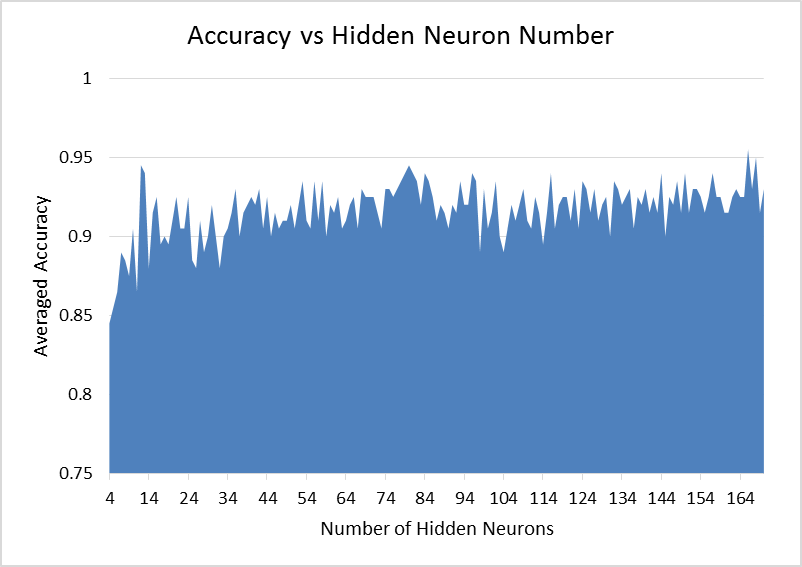


Figure Averaged Accuracy versus Number of Hidden Neurons

SVM

SVM is one of the most classical machine learning algorithm. Its primary objective is to find a plane that can separate the samples with largest margin, i.e. maximize the margin.

As we are dealing with the non-separable samples, we would expect some misclassified samples. Those misclassified samples need to be taken into considerations while the algorithm searching for the optimal hyperplane. In this case, we constructed the Lagrange primal function with a C coefficient. In either case (separable or non-separable case), we need to deal with a Lagrange dual function with a KKT condition, which serves as the mathematical theory for this algorithm.

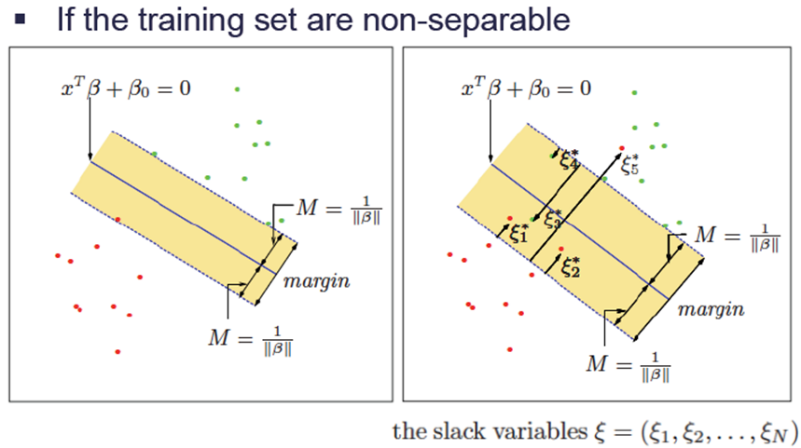


Figure Basic Principles of Support Vector Machines

In our specific problem, it is hard to find the optimal separating plane with hard margin that can perfectly separate all data samples into the group in which they should be, so we adopted the soft margin for the SVM algorithm to resolve our non-separable problem. This was indicated by the fact that the C value was set to be 1.0. An advantage of SVM is that it can use kernel method to transform the available data onto another space for better classification effect. However, in this problem, the simple “linear” kernel works best, which means that we do not need to transform the data onto other space. This may be due to the fact that the labels (i.e. types) of all wells in our problem were generated based on EUR values, and the EUR values, in turn, were calculated through linear combination of monthly production rate. Additionally, the stopping criteria (i.e. error tolerance) was set to be 1e-3.

RF

RF is an ensemble approach that adopted divide-and-conquer idea to improve performance. The main principle behind RF is that a group of “weak learners” can come together to form a “strong learner”. RF starts with a standard machine learning technique – classification and regression tree (CART), which, in ensemble terms, corresponds to the weak learner. Figure 14 (Simafore 2017) shows the different types of trees at each level and their usages. To solve our problem specifically, people usually use the C4.5 implementation when they are dealing with a classification problem and we have multiple classes.

The decision trees are recursively built following a top-down approach by repeated splits of the training dateset. When decision tree work with continuous numerical values, the binary splits usually performed by choosing the threshold which minimize the impurity measure used as splitting criterion (Berzal et. al 2003). Figure 15 (Saedsayad 2018) gives an example of leveraging the decision tree to predict whether or not playing golf. The core algorithm in this graph is ID3 which uses entropy and information gain to construct the decision tree.

In terms of the RF algorithm, it addressed the overfitting problem that may often arise in the decision tree algorithm. It works to generate a whole bunch of decision trees. Each decision tree was constructed based on a subset of training samples (randomly selected with replacement) and a subset of features (randomly selected without replacement). Those decision trees combined decide the final classification type through majority vote.

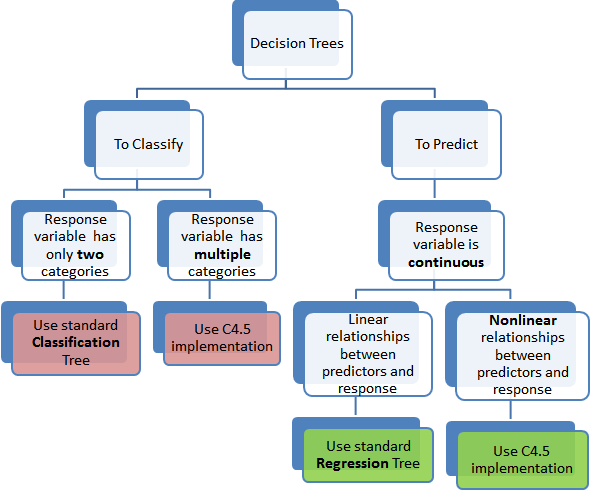


Figure 14 Classification and Regression Tree

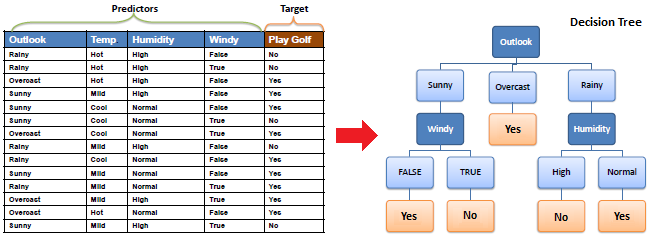


Figure Example of Decision Tree

In our classification problem, I compared the index as the criteria to split the nodes.

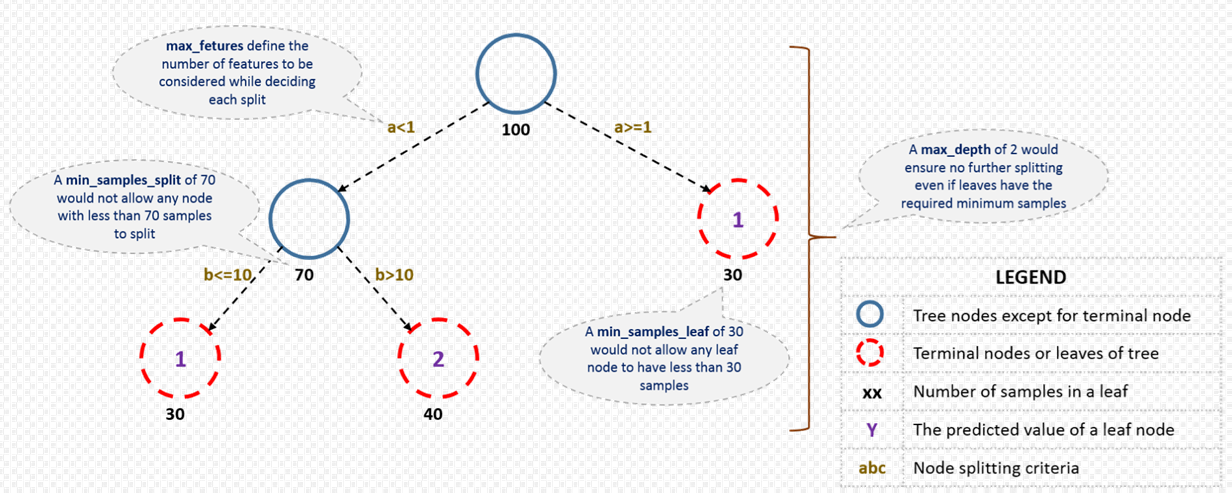


Figure Structure Terms of Random Forest

RF is an ensemble approach that adopted divide-and-conquer idea to improve performance. The main principle behind RF is that a group of “weak learners” can come together to form a “strong learner”. It can both be used in classification and regression problems (CART). RF starts with a standard machine learning technique called “decision tree” which, in ensemble terms, corresponds to the weak learner. In classification problem, we use the Gini index as the criteria to split the nodes. The number of decision trees, the maximum tree depth and maximum feature are set to be 100, 20 and 50, respectively.

The classification results were shown in figure 5.

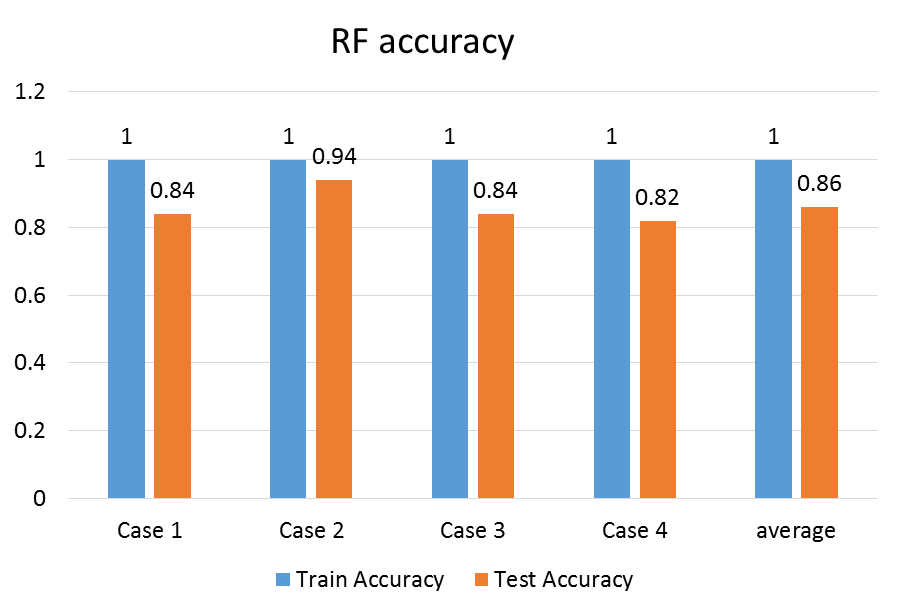


Figure Machine learning algorithm application results

Reference

DrillingInfo 2017 <https://info.drillinginfo.com>

D. IIK, J.A. Rushing, A.D. Perego, T.A. Blasingame, Exponential vs. Hyperbolic decline in tight gas sands: understanding the origin and implications for reserve estimate using Arp’s decline curves, Presented at the SPE Annual Technical Conference and Exhibition, 21-24 September (2008, January 1), 10.2118/116731-MS, Denver, Colorado, USA

Valko, P. P., & Lee, W. J. (2010, January 1). A Better Way To Forecast Production From Unconventional Gas Wells. Society of Petroleum Engineers. doi:10.2118/134231-MS

A.N. Duong, 2011, Rate-decline analysis for fracture-dominated shale reservoirs, SPE Reserves Evaluation Engineering, 14(03) (2011), pp.377-387, 10.2118/137748-PA

ValNav 2017 <http://www.energynavigator.com/value-navigator>

MLPClassifier, scikit-learn 2017 <http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html>

SimaFore 2017 <http://www.simafore.com/blog/bid/62482/2-main-differences-between-classification-and-regression-trees>

Berzal F., Cubero JC., Marín N., Sánchez D. (2003) Numerical Attributes in Decision Trees: A Hierarchical Approach. In: R. Berthold M., Lenz HJ., Bradley E., Kruse R., Borgelt C. (eds) Advances in Intelligent Data Analysis V. IDA 2003. Lecture Notes in Computer Science, vol 2810. Springer, Berlin, Heidelberg

Saedsayad 2018 <http://www.saedsayad.com/decision_tree.htm>