**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, precisely speaking they are recorded as the format of rate versus month. 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 their different production time lengths. In our first step, we need to label those 200 samples. The typical production profile was shown in the figure 1. The production data in figure 1 is from the well with API number 4212132044, which is one of the 200 wells that we will be using during this project. The red dots showed the trend of declining rate of production. This well has a production time of 157 months until May 1, 2017. With the time increasing, the production rate will decrease below a threshold, which is called “abandon rate”. Normally, the life of one particular unconventional gas well would 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 looking at the cumulative production at the end of 360 months. After separating them into different type according to the P10, P50 and P90 values, we complete the labelling process.

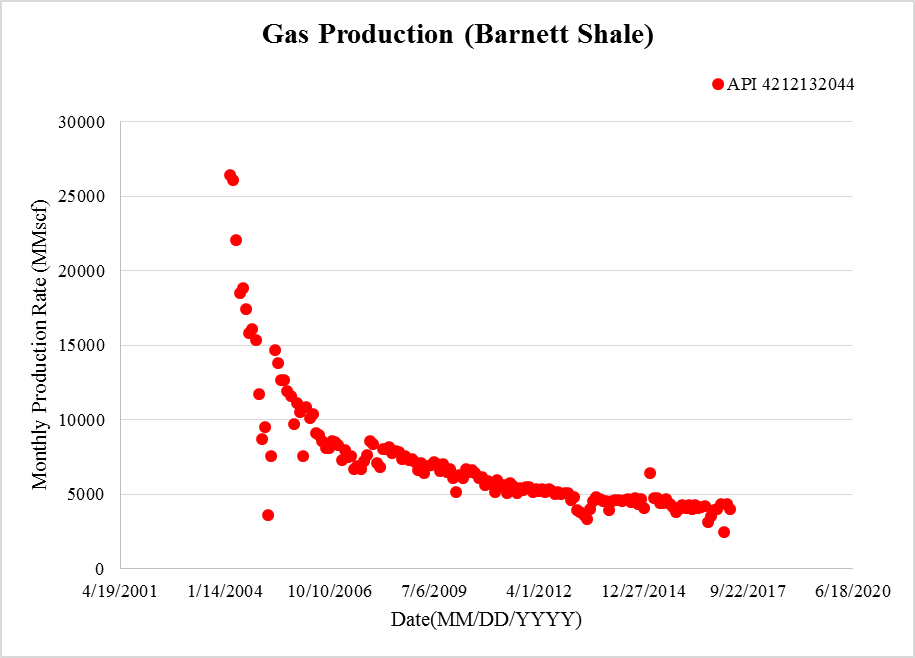


Figure 1 Typical Well Production Profile

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: Power’s Law, Stretched Exponential, and Dong’s method. The forecasted data will be used to compute the ultimate estimate recovery (EUR), which will be further used in the labelling stage.

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

( 1 )

( 2 )

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)

( 3 )

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)

( 4 )

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.

As shown in these two log-log plots in figure 2, the blue dot lines show the raw data plotted in log-log scale – rate versus material balance time. It is quite obvious that the slope of the decline trend line is still 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 when we try to use them to construct the type wells, which is specifically indicated by the red lines. In this way, I cannot use the available data and the 3 models mentioned above to implement the forecasting directly.

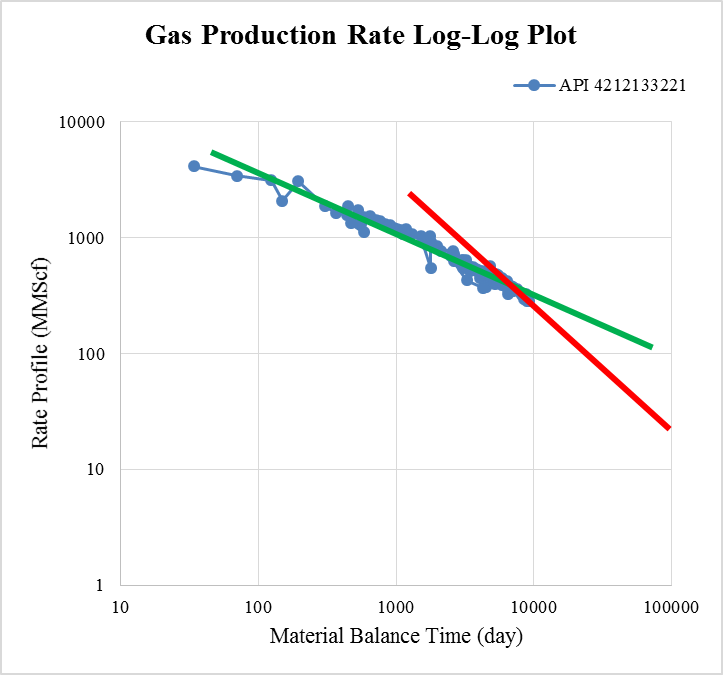
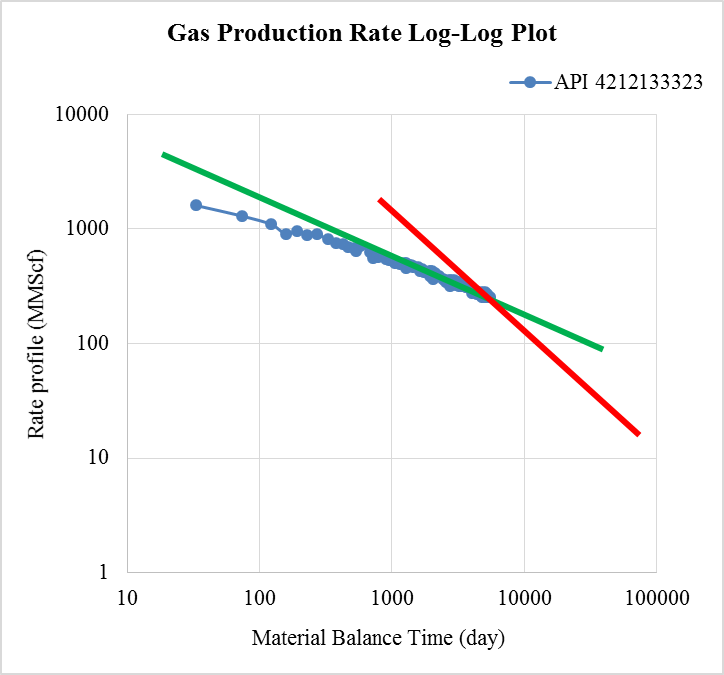
 

Figure 2 Wells Not Reaching Boundary Dominant Flow

To resolve this problem, we implement the forecasting by implement extrapolation first, which is performed 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 simply choose to specify the switch point by specifying the fixed decline slope (%/yr) at 6.5 as shown in figure 3.

With the parameters set as shown in figure 3, ValNav will automatically transfer into BDF mode after the decline rate (%/yr) of transient flow at 6.5% is identified. When ValNav is implementing the forecasting with the switch point specified, it will try the models above and the production histories will be extrapolated to 360 months. In this way, 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 achieve the forecasting. Figure 4 is our parameters setting in ValNav for BestFit. The BestFit automatically selects the most suitable data recently and the most suitable model to fit data first, and it will implement the forecasting based on the selected data and model. ValNav gives us fugure production data until it reach the abandon rate (i.e. 10 Mscf/d as we have done in this project), but we only need first 360 months data.

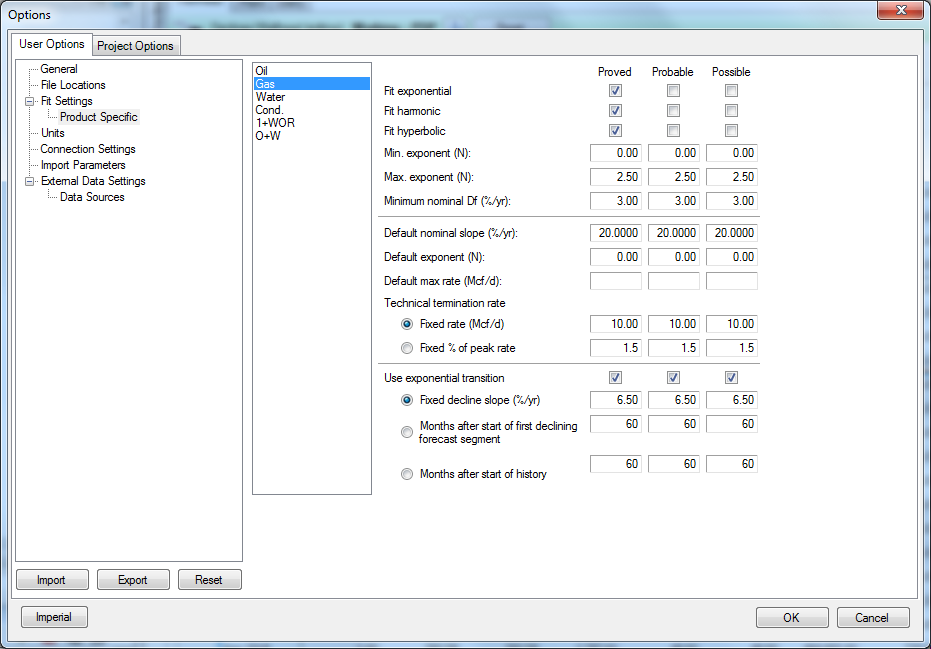


Figure 3 Specifying the Switch Point in ValNav

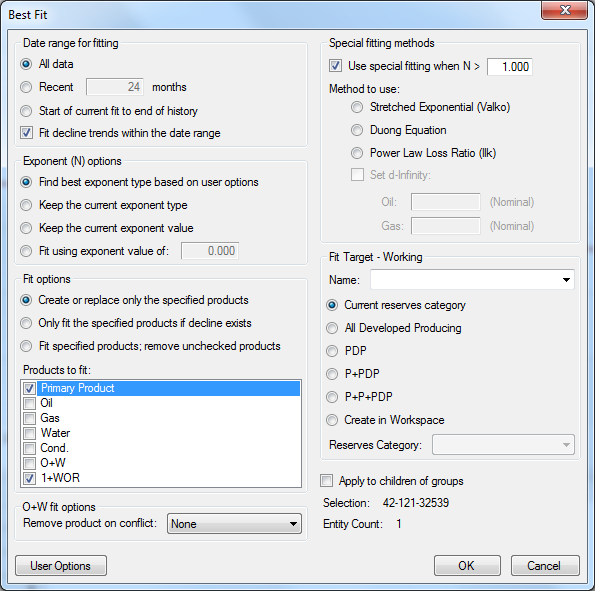


Figure Best Fit Parameter Setting in ValNav

After ValNav has completed the BestFit procedure, we can get a production profile for each well. The profile, as mentioned before, has a production life time 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, we sorted them in descending order accordingly. In this case, each EUR value for each well is corresponding to a “less than probability”. We plotted the EUR distribution in a log probability paper as shown in figure 5. The horizontal and vertical axis of figure 5 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.

Type well construction and evaluation

Given that EUR is predicted by ValNav as shown in figure 5, this step focuses on constructing type wells. With the insurance of BDF and the EUR value for each well, we can simply find the P90, P50, and P10 type wells from the plot in figure 5, which are 1538.053 MMscf, 2448.201 MMscf, and 3759.201 MMScf, respectively. The P90 value is the EUR value corresponding to the less than probability of 0.1 in figure 5, and P50 and P10 corresponds to 0.5 and 0.9 respectively. With the 3 values picked up, 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.

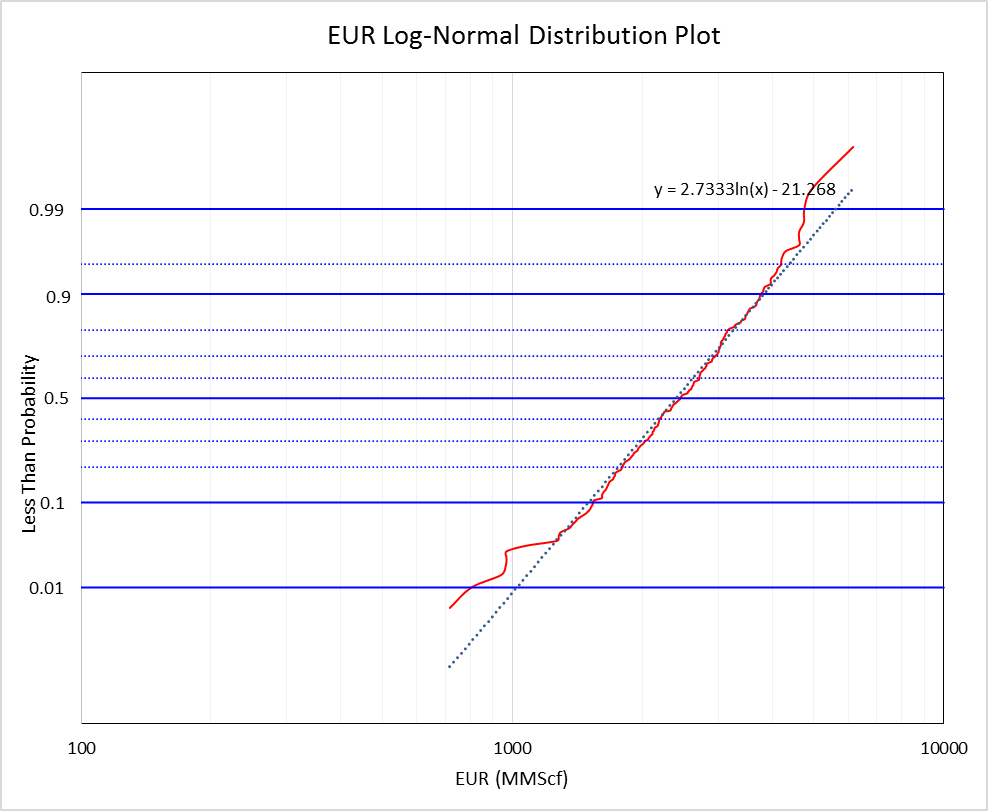


Figure EUR lognormal distribution

The minimal dispersion in this problem indicates a good probabilistic property of the type wells, which means that the type wells (e.g. 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 will be the classified with a given type such that they can be used to train 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, we 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)

One of our main projectiles is to classify the short production history data of a new well into one of the 4 types. The EUR value range is determined upon the determination of the type although we don’t have the available production data for this well’s whole life time.

Uniform Input Dimensionality

In the labelling section, we 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 sample to the machine learning algorithms has the same feature dimensions, we extrapolated each well’s raw data to 170 months. This procedure can be performed 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. Those terms will be more easily understood in machine learning fields.

4-fold Cross Validation Technique

Cross validation technique is commonly used to reduce the generalization error of machine learning algorithms. Before we implement the machine learning algorithms to classify the wells with only short production history into one of the 4 types as mentioned above, I separated the 200 samples into 4 groups. For every 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. We 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 and testing cases. In this way, the generalization error (i.e. the risk of overfitting) of the algorithm will be minimized by not relying on any specific group of data that might not be representative of the whole dataset.

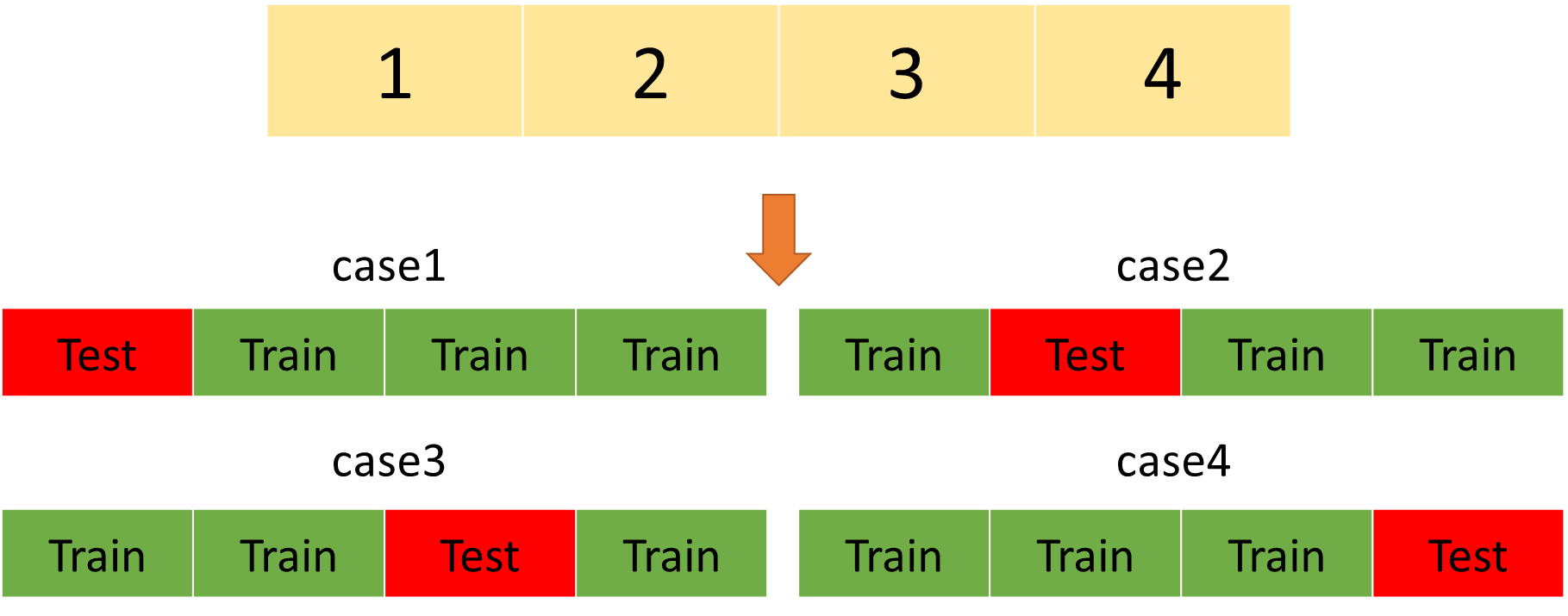


Figure Cross Validation Technique

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 when using MLP to solve our problems.

In this project, we simply choose the MLP as our training algorithm due to its simplicity to implement. The basic MLP architecture used in this project is constructed as shown in the figure 7.

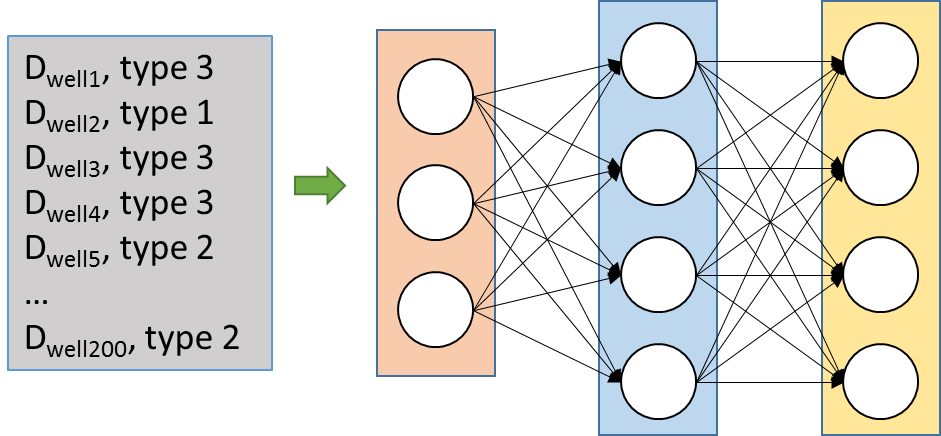


Figure Neural Networks Training Architecture

In figure 7, 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 are 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 8, each column represents a particular feature of all samples, and each row represents all features of a particular 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 equation (5). In this way, all the values will be converted into the value between [0, 1], and the risks of abnormal features are minimized.

( 5 )

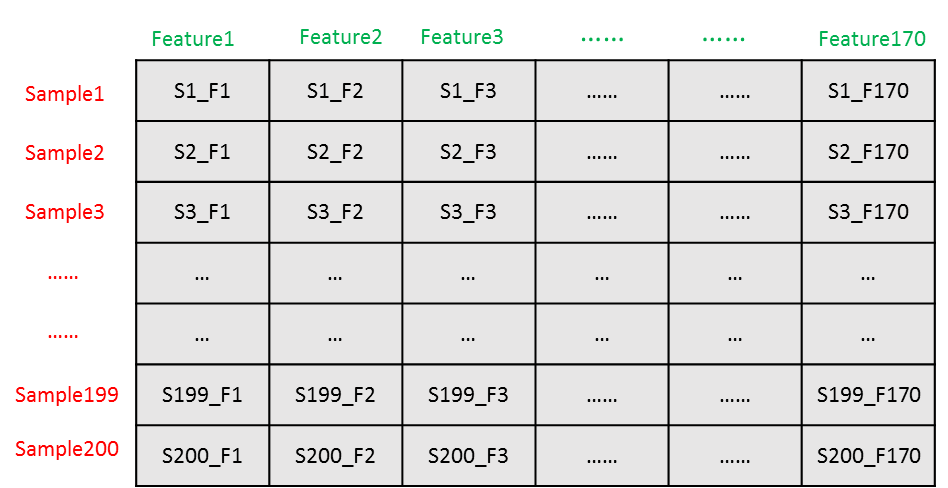


Figure Min-Max Normalization

The backpropagation algorithm is the core of the neural network algorithm. Algorithm 1 illustrates the back propagation algorithm. 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.

|  |
| --- |
| **Algorithm 1** Back Propagation |
| Initialize all weights to small random numbers.  Until satisfied, Do  For each training example, Do   1. Input the training example to the network and compute the network outputs 2. For each output unit 3. For each hidden unit 4. Update each network weight   where  Note: is the weight from to (i.e., ) |

The algorithm 1 is implemented using python package scikit-learn (MLPClassifier, scikit-learn 2017), the results are shown in figure 9.

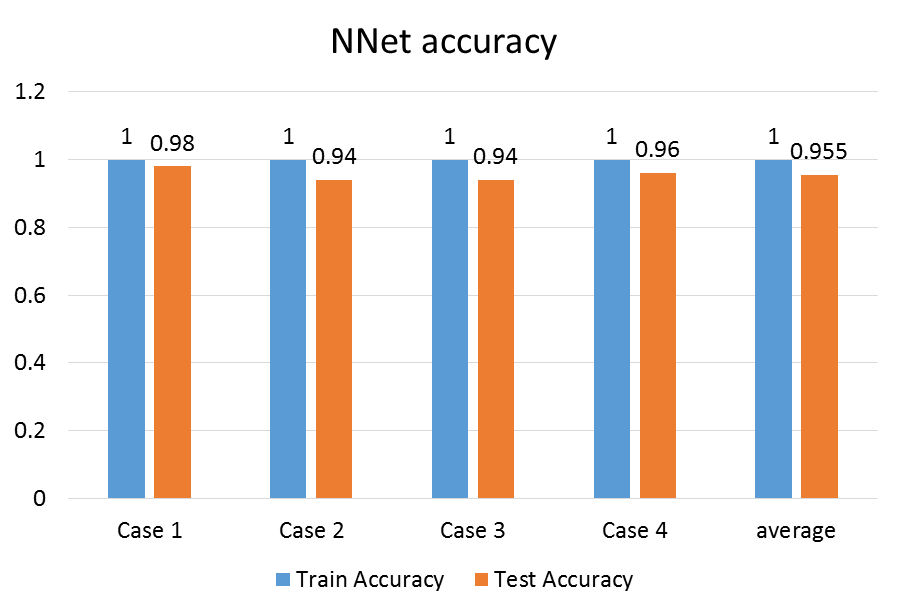


Figure Neural Network Algorithm Accuracy

In figure 9, 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 0.94 to 0.98. The averaged accuracy of 0.955 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 section. With the aid from NNet algorithm and the architecture parameters setting mentioned above, we have 95.5% confidence to correctly classify a well.

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 at the first in algorithm 1 to initialize all weights to small random numbers at first. Another possible option could be adding the momentum parameter as what we do when implementing 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 earlier, 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 10 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 10, the accuracy varies rapidly 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, 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 10 show that more hidden neurons improve the accuracy, which implies even 170 hidden neurons may not cause model overfitting in our problem specifically.

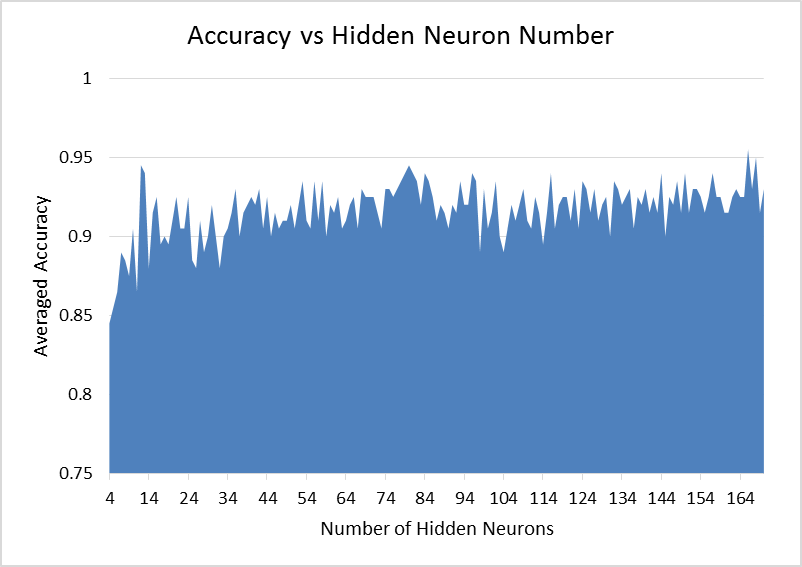


Figure Averaged Test Accuracy versus Number of Hidden Neurons

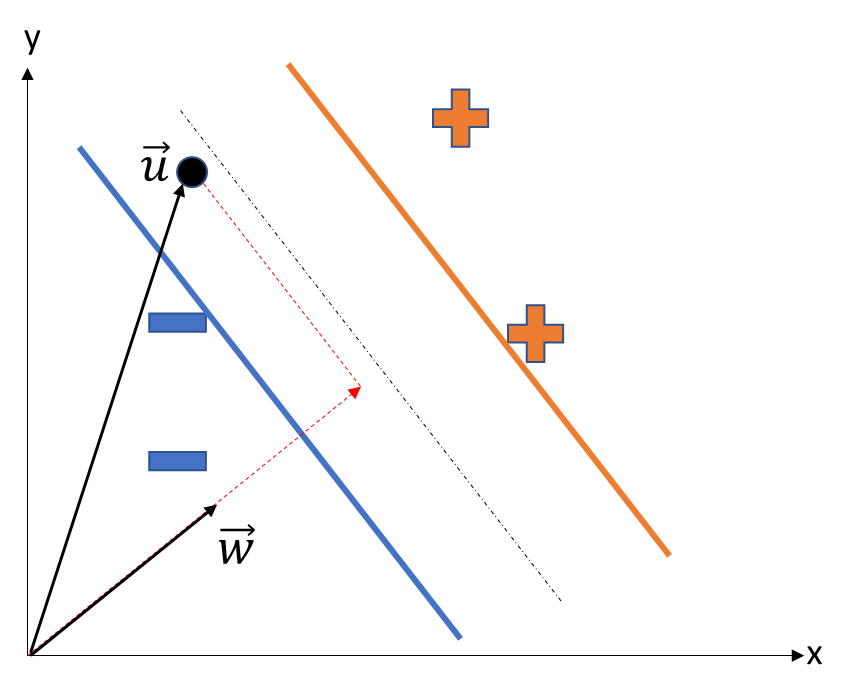
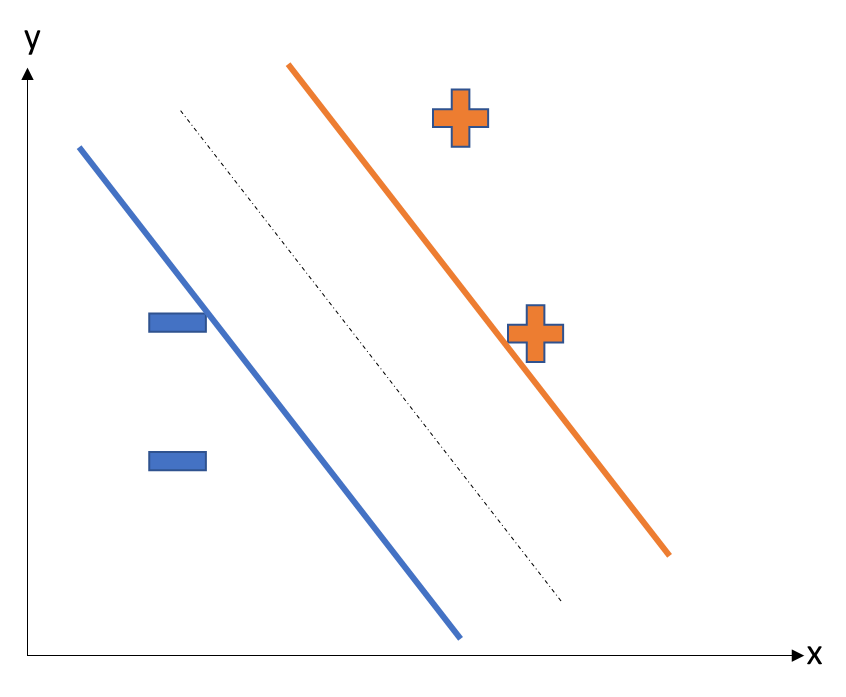
Since the weights are initialized with random small numbers, so each time we start a new training process, the number of hidden neurons that lead to the highest accuracy may vary. The result shown in figure 10 is only the determination that is exactly the weights initialization situation where the number of hidden neurons of 163 can achieve the best performance. Reader may see different result as they implement their own network architectures.

**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.

To simplify the explanations on SVM algorithm, the following will assume that we are dealing with binary classification problem. The possible label involved can only be either 1 or -1. Sometimes the label will be simply called a plus or minus. The multiple class classification can be implemented simply by generating multiple SVM classifiers.

Suppose we have several samples that are labelled with minus and plus signs which represent two classes. We separate the samples with a plane that is represented by the dotted black line in figure 11(a). The orientation of this plane can be different as long as the plane can succeed to separate the two kinds of samples into two different classes. The distance from the plane to blue line and to orange line is the same, and it is called margin, and the points that are exactly located on the blue lines and orange lines are called support vectors. This is also where the classifier name “Support Vector Machine” comes from. The objective of the algorithm is to find such support vectors to maximize the margin (e.g. to minimize the generalization error).



(a) (b)

Figure Support Vector Machine Principles (Hard Margin)

In figure 11(b), we add a vector perpendicular to the plane (dotted black line). The length of this vector is random. In the coordinate system as shown in figure 11(b), the dotted black line can also be represented by a vector . When we need to classify the coming new sample , we need to compute the length in the direction of , which is . If the length exceeds a certain constant , then this new sample will be classified as a plus sign, otherwise, it would be defined as minus. To formally define this, if the new point satisfies

( 6 )

, then the point would be labelled as a plus sign. Otherwise, the point will be a minus sign. Equation (6) is also called the decision rule. Now what we are going to do is to find a suitable vector and b value to maximize the margin. To achieve this, we define

( 7 )

for each sample between the blue line and orange line. From equation 7, we can derive the width of the margin as

( 8 )

Given equation 8, to find vector and b value that maximize the margin, we can simply define the objective function as

( 9 )

With the constraints from equation 7, we need to introduce the Lagrange multipliers to solve this convex programming problem. In this way, the objective function become

( 10 )

Equation 10 is also called Lagrange primal function, , and this becomes a dual problem. Set the partial derivative to be 0, we can easily get

( 11 )

( 12 )

By plugging equation 11 and equation 12 back to equation 10, we get the Lagrangian dual function

( 13 )

Subject to ,

Equation 13 manifests that the Lagrangian function solely depends on the pair of sample points when Lagrange multipliers are fixed. This gives us the general algorithm for a SVM classifier:

|  |
| --- |
| **Algorithm 2** Support Vector Machine |
| Repeat till convergence {   1. Select some pair and to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum) 2. Reoptimize with respect to and , while holding all the other ’s fixed.   } |

Using the algorithm 2, we can specify the suitable for each sample . Once the Lagrange multipliers are fixed, the plane with maximal margin are decided. Then we can use the equation 6 and equation 11 combined to decide the label of an unknown point

( 14 )

Above is the process of deciding a separating plane for separable sample set, which is also called hard margin. As for the non-separable sample set, we will create a soft margin that can tolerate misclassified samples. Figure 12 clearly states the differences between hard margin and the soft margin.

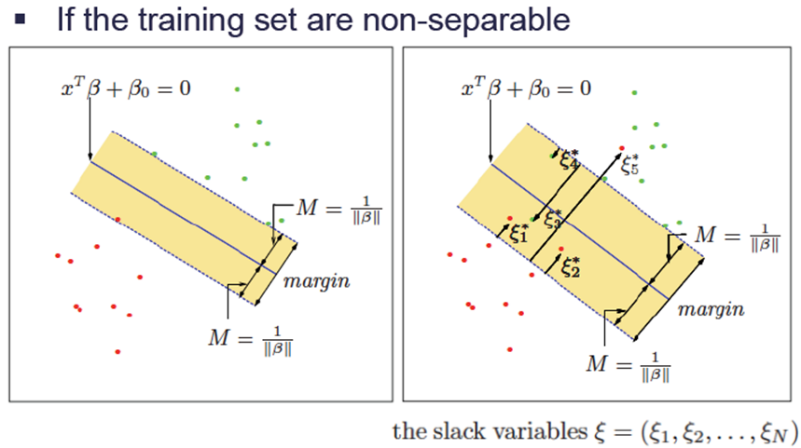


Figure Hard Margin and Soft Margin

In figure 12, the left side shows the hard margin schematically, while the right side indicates the situation of soft margin where the data sets are non-separable. Several samples are misclassified in soft margin (e.g. the green dots and red dots in the margin area in the graph on the right of figure 12).

The whole process of equation deduction for soft margin is similar. I briefly introduce it here. We add a penalty parameter C of error term to the Lagrange primal function, the Lagrange primal function becomes

( 15 )

Subject to ,

As we solve this quadratic problem, we can easily get the Lagrange dual function

( 16 )

Subject to ,

The KKT conditions for the soft margin are

We can see that the penalty term limits the range of , this is how the value affects the final accuracy. In this project, the data sets are non-separable. This can be inferred from figure 13. In figure 13, as we decrease the C value exponentially, the train accuracy and test accuracy both began to decrease when the C value reach at the interval (1E-5, 1E-7). This means if we keep reducing the penalty on the misclassified samples in the training process, we will expect a decrease of both training and test accuracies. It can be inferred that if C approaches 0, the accuracy will drop significantly in this problem.

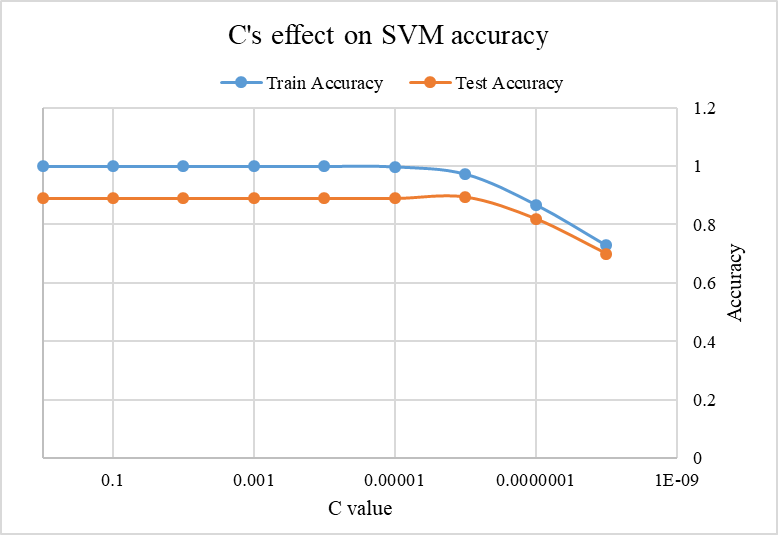


Figure The Effect of C value on the accuracy of Support vector Machine Algorithm

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 at 1.0. In either case (separable or non-separable case), we need to deal with a Lagrange dual function with a KKT condition, which serves as a key part of the mathematical theory for this algorithm.

In our specific problem, the algorithm was implemented using the scikit-learn package in python (SVC, scikit-learn 2017). 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.

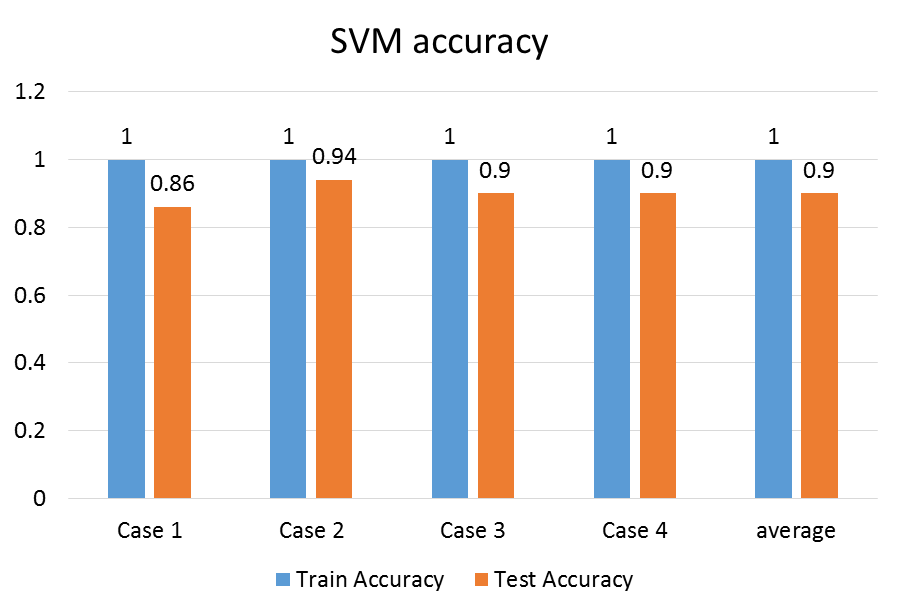


Figure Support Vector Machine Accuracy

Figure 14 shows both of the training and testing accuracies. There are 4 cases corresponding to the 4 fold cross validations. The accuracies for training set are 100%, and the test set accuracies are all above 0.86 with an average of 0.9. This implies a good minimization of generalization error. From figure 14, we can infer that given a coming well production data with only short production history, the average confidence of classifying this well into correct type is 90% when using support vector machine with parameters setting mentioned above. Once the type is determined, further EUR estimation is obvious and even financial decision in oil and gas companies can be implemented.

**RF**

RF is an ensemble approach that adopted divide-and-conquer idea to improve performance. The main principle behind RF is the fact 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 15 (Simafore 2017) shows the different types of trees at each level and their usages.

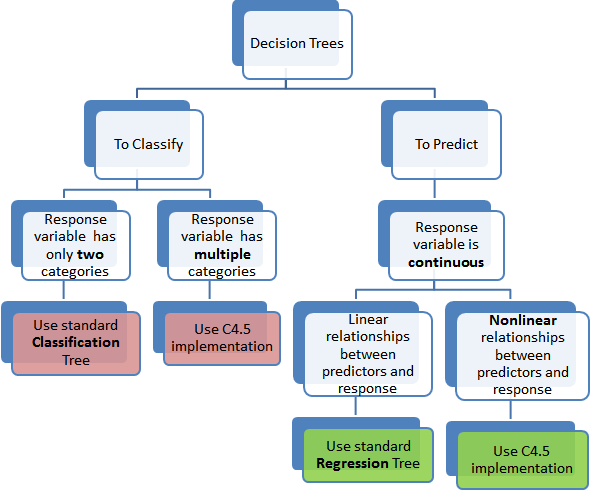


Figure 15 Classification and Regression Tree

In our classification problem, I compared two criterion to split the nodes: gini index and entropy gain. The training set accuracies for both criterions are all 100%. The difference is the test set accuracies. The results are shown in figure 16. In figure 16, we can see that the accuracy difference of each case varies. The accuracy of Gini index splitting criterion is always lower than that of entropy gain. In terms of the average performance, the accuracy of entropy gain is also higher. So in our project, we implement the RF algorithm based on entropy gain to split the nodes.

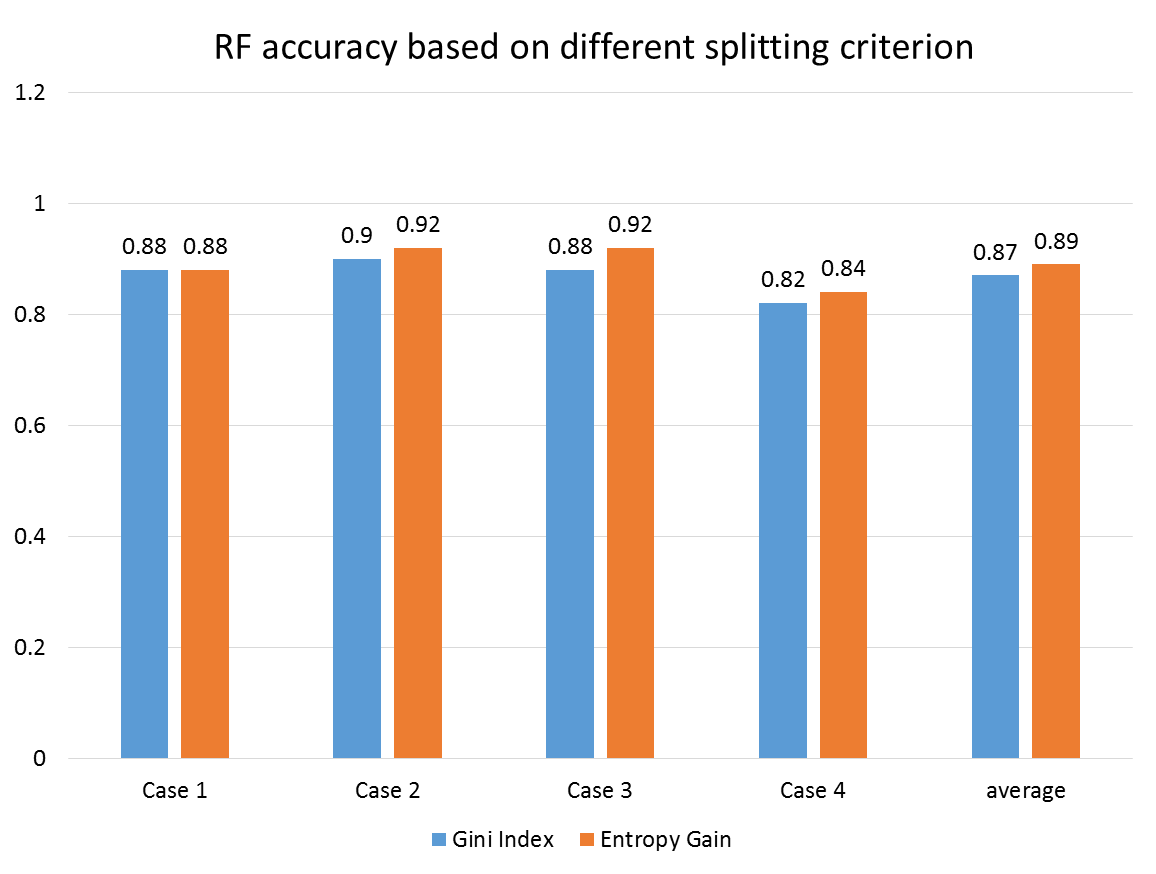


Figure 16 RF Accuracy: Gini index and Entropy Gain

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 are usually performed by choosing the threshold which minimize the impurity measure used as splitting criterion (Berzal et. al 2003). Figure 17 (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 as the criterion to split the tree nodes when constructing the decision tree.

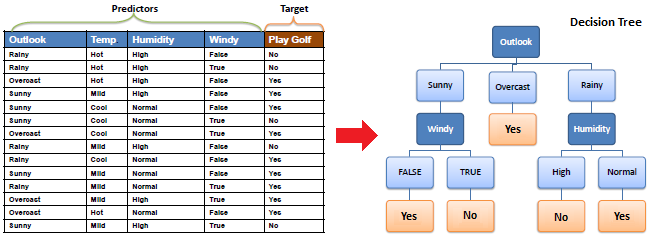


Figure 17 Example of Decision Tree

RF algorithm 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.

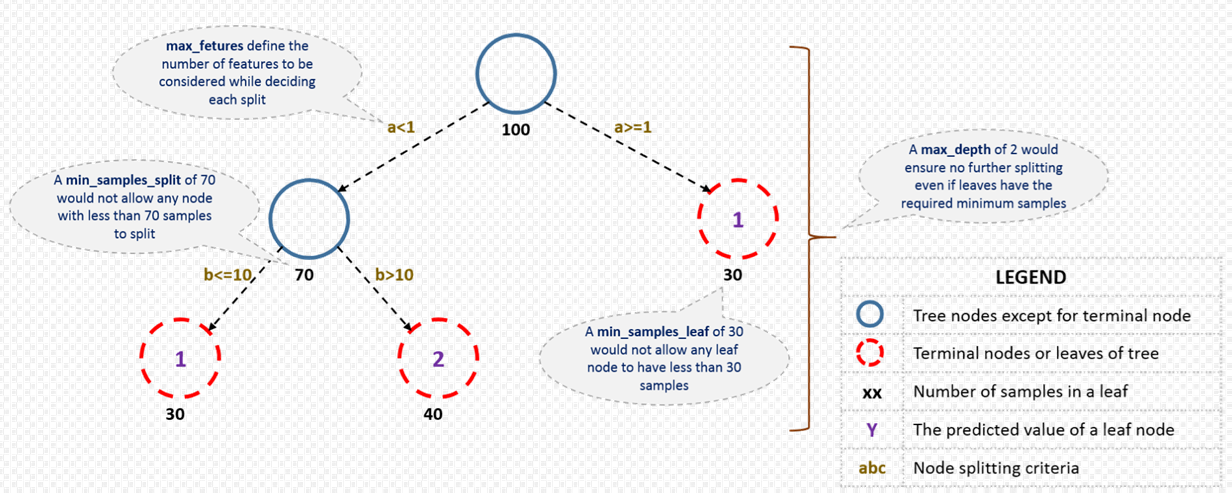


Figure 18 Structure Terms of Random Forest

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 entropy gain 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. Readers can refer to figure 18 (Analytics Vidhya 2018) and scikit-learn (RandomForestClassifier, scikit-learn 2017) for the definition of these parameters.

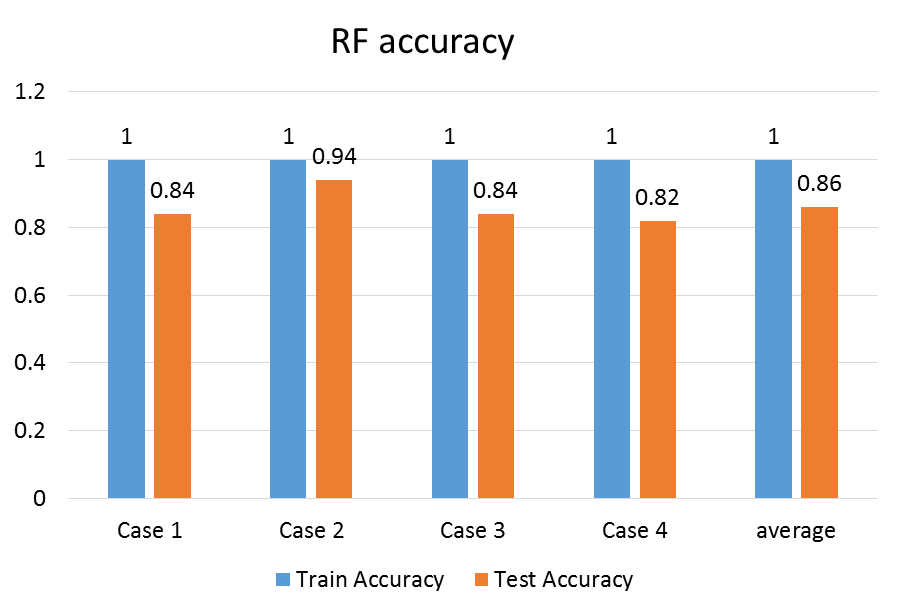


Figure 19 Machine learning algorithm application results

The classification results were shown in figure 19. From figure 19, we can see that the test set accuracies of RF on different cross validation cases vary, while the training accuracies can be 100% for all the cases. The average test accuracy is 0.86. Recall the results that we have got from our neural networks (NNet) and support vector machine (SVM) sections. The average test accuracy of NNet is 0.955, and the average test accuracy of SVM is 0.9. The accuracy of RF is lower than both. This shows the weakness of RF algorithm in solving our problem. As one well with short production history is classified into one of the type, we have only 86% percent that this classification was implemented correctly.

Summary

The Results and Discussion section shows the implementation details of this project. Initially, we preprocessed the data to get 200 wells’ production data from the same geologic area – Barnett Shale. We used BestFit in ValNav to implement the production forecasting for each well, and the cumulative production at the end of 360 months is the EUR of this well. As we plot the “Less Than Probability” versus sorted EUR on the log probability paper, we could see a lognormal distribution of the 200 EUR values. Since the EUR values for all 100 wells were known, the P90, P50 and P10 type wells were picked up, they are 1538.053 MMscf, 2448.201 MMscf, and 3759.201 MMScf, respectively.

We separated the 200 samples into 4 parts according to the EUR values of type well. And the labels for all wells are their corresponding types. After we have kept the input sample feature dimensions uniform, we then began to use the data to train classifiers with the 3 machine learning algorithms: neural networks (NNet), support vector machine (SVM), and random forest (RF). The cross validation technique was also employed to reduce the generalization error of the trained classifiers. The details of these 3 algorithms were also introduced.

In NNet, the min-max normalization technique is adopted to preprocess the input into neural networks. The average test accuracy of NNet is 0.955, which is the highest among the three. The number of neurons at hidden layer is determined to be 163 to achieve the highest accuracy. In the case of SVM, the non-separable property for our problem is identified through the tests using different C values. The averaged test accuracy of SVM is determined to be 0.9, while the average test accuracy of RF is 0.86, which is the lowest among the three. In RF, the results of two splitting criterion (Gini Index and Entropy Gain) are compared. To sum up, the NNet can achieve the highest accuracy, and it is the most suitable algorithm in terms of resolving the problem in this project.

**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>

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

Analytics Vidhya 2018 <https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/>

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