NOTE: This paper was presented at SEG19 International Exposition, Sept. 15-20, 2019, in San Antonio.

Predicting Gas Production Using Machine Learning Methods: A Case Study

Tania Mukherjee (Stonebridge Consulting), Thomas Burgett* (Jonah Energy), Telha Ghanchi* (Stonebridge Consulting), Colleen Donegan (Jonah Energy), Taylor Ward (Jonah Energy).*

Summary

Predicting well performance and production prior to drilling has always been a challenge in the oil and gas industry. The need of developing robust methods of prediction helps it to be financially more viable and technically superior. With the emergence of big data, the ever-increasing computational power, and the growing library of platforms supporting a multitude of machine learning methods, data and analytics departments can play a crucial role in the development of fact-based methodologies and predictive models of future performance with high levels of confidence. In this paper we lay a series of supervised regression methods that proved highly predictive when compared to existing methods of estimating undrilled well production. We conducted an extensive machine learning modeling exercise using data from an active Jonah Energy (Jonah) gas field in Sublette County, Wyoming. Our objective was to predict well performance and annual-measured gas production in the first year of production. We considered several multidimensional and complex data (geological, drilling and production data) as direct and indirect factors that control production. We used a fact-based methodology for feature selection that focused on the use of measurement data rather than estimations or interpretation. Finally, we tested and compared various machine learning algorithms—including Linear Regression, Principal Component Analysis, Neural Network Regression, Boosted Decision Tree, and Binned Decision Tree—to find the optimum prediction of the current gas field. Ultimately, we judged the best results resulted by using Binned Decision Tree where the Coefficient of Determination was 0.63 and squared error was 0.36. Further Ensemble Decision forest was also tested to rule out any overfitting. Given the complexity of geology and physical limitation present dataset, the current production prediction seems prudent.

Introduction

The target gas field produces primarily from upper Cretaceous tight sands and siltstones. The gross productive interval is several thousand feet thick and consists of discontinuous interbedded classics (sandstones, siltstones, and mudstones). The main production happens from an Upper Lance formation and Lower Mesaverde Formation (Cluff & Cluff., 2004). The Lance formation is fluvial in nature and has some packages of thick amalgamated sands that are heterogeneous but highly connected over long distances. These sands were deposited by a combination of medium-sized meandering channels and braided streams. The Mesaverde channels are comprised more commonly of single-channel fill complexes that are shallower and wider than those in the Lance and tend to be highly discontinuous. There is some amalgamation, but to a lesser extent than in the Lance. Mesaverde shales have fossils and structures that exhibit some brackish floodplain influence.

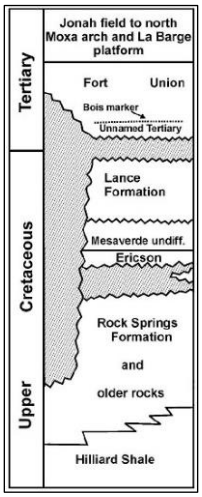

Figure 1: Stratigraphy of the Jonah Field Area; Modified after Hanson et al (2004)

The field is fault-bound and believed to benefit from a mix of structural and stratigraphic trapping mechanisms. The source of gas in Lance sands is largely believed to be primarily from the Hilliard (or Baxter) formation, several thousand feet deeper in section. (Figure 1, Modified Stratigraphy).

For our exercise, Jonah identified approximately 1,630 wells and 75 features to test and train machine learning models. Those well data were recorded between the year of 1993-2018 and scattered throughout the 28,000-acre gas field. Among the 75 features, some have a direct relationship with production such as OGIP and formation thickness, while other factors such as total slurry or stage count have more complicated effects on production. Data cleansing and conditioning prove to be the most critical and time-consuming part in a machine learning exercise. To do analytical use cases, the data needs to be totally numeric. The following flow chart shows a generic machine learning flow for any machine learning exercise.

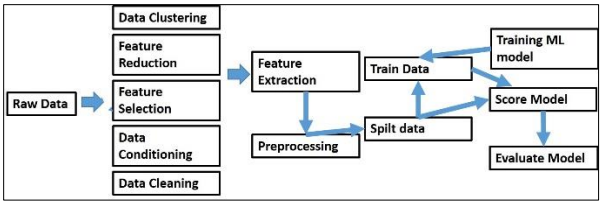

Figure 2: Generic Machine Learning Flow.

For our exercise, data cleansing involved a lot of standardization and filtering of repeated, incomplete, or incorrect records. Data cleansing occurred in two passes. The first pass focused on eliminating data that was deemed not useful. The second pass was to remove data that has no effect on production such as vender information, tool details etc. Dimensionality reduction methods were tested with the

help of principal component analysis. A methodical approach was taken on parameter/feature selection. Multiple passes were made both with and without certain values to check their effectiveness on production prediction. In our exercise, we selected 25 parameters to predict initial production. Once the data cleansing was done, we tested five different machine learning algorithms to optimize the prediction error. In the next section, we will discuss the different algorithms used in our work.

Machine Learning Algorithms Used

For the all our machine learning experiments, we used Azure Machine Learning Studio. Azure Machine Learning Studio is a cloud-based predictive analytics service that allows an agile "create and deploy" flow for generating predictive models as analytics solutions. A step-up approach was taken in this case, i.e., start with the simplest algorithm, see the results, and slowly add more layers of complexity as needed.

Linear Regression. The first supervised machine learning method tested was Linear Regression. Linear Regression by principle performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for discovering the relationships between variables and forecasting. Different regression models differ from each other based on the kind of relationship between their dependent and independent variables. A Linear Regression model assumes that prediction errors follow the Gaussian distribution. This is usually fine except when the prediction target spreads out over several orders of magnitude (Zheng & Casari, 2018). Figure 3 depicts the steps of Linear Regression machine learning. For our test, a Linear Regression model gave a Relative Squared Error 0.93 and Coefficient of Determination of 0.07.

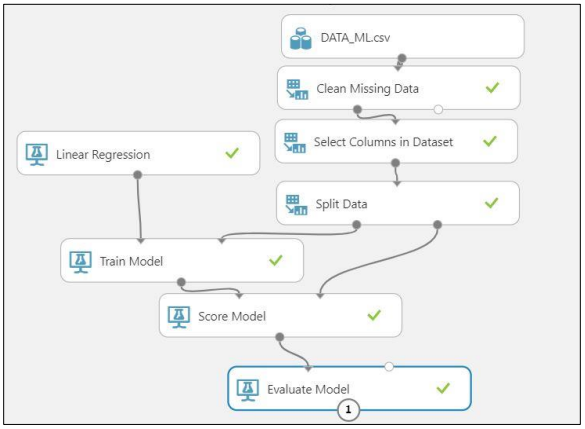

Figure 3: Flow chart showing Linear Regression model building steps.

Principal Component Analysis (PCA) and Neural Network (NN) Regression. When using PCA for dimensionality reduction, the primary question to be addressed is how many principal components (k) to use. Like all hyperparameters, this number can be tuned iteratively based on the quality of the resulting model. However, there are also heuristics that do not involve expensive computational methods (O'Reilly, 2018). After dimensionality reduction, a set of the neural networks was tried with various nodes. There are many parameters in neural networks controlling final model performance. The most crucial parameters are the number of hidden layers, learning rate and the number of iterations.

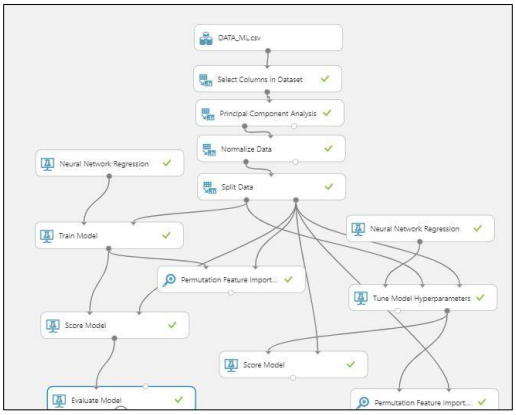

Figure 4: Flow chart showing steps of NN without PCA model building steps.

Based on the dataset, one can define the decision boundary as a set of decision lines, and a number of decision boundaries should separate the number of classes as needed. The number of the decision lines represents the number of hidden neurons in the first hidden layer. A new hidden layer should be added to connect the lines created by the previous layer. Thus, the number of hidden neurons in each new hidden layer equals the number of connections to be made (Gad, 2018). If the learning rate is low, then training is more reliable, but optimization will take significant time because steps towards the minimum of the loss function are tiny. There is no optimal number that generalizes across all data-sets of a fixed size, but there are certain metrics to evaluate the quality of fit relative to that set. In this case, using NN without PCA improved the squared error prediction to 0.33. The number of hidden nodes is 1000, the learning rate of .01, and the numbers of iteration used are also 1000 (Figure 4).

Decision Tree, Boosted and Binned. The next algorithm tested was Decision Tree. While one of the simplest algorithms, Decision Tree has proved to be very useful across a multitude of modeling exercises. This regression method is also a supervised learning method and therefore requires a labeled dataset. The label column must contain numerical values. Decision Tree builds classification or

regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while simultaneously an associated "master" Decision Tree is incrementally developed. Boosting means that each tree is dependent on prior trees. The algorithm learns by fitting the residual of the trees that preceded it. Thus, boosting in a Decision Tree ensemble tends to improve accuracy with some small risk of less coverage. In this case, we used a slow learning rate (0.5) and a comparatively simpler structure (number of leaves is 8 and minimum number of samples per leaf node is 8 and the total number of the tree constructed 500), which gave a squared error of 0.37. Further binning the inputs into four groups further reduced the squared error to 0.36**.**

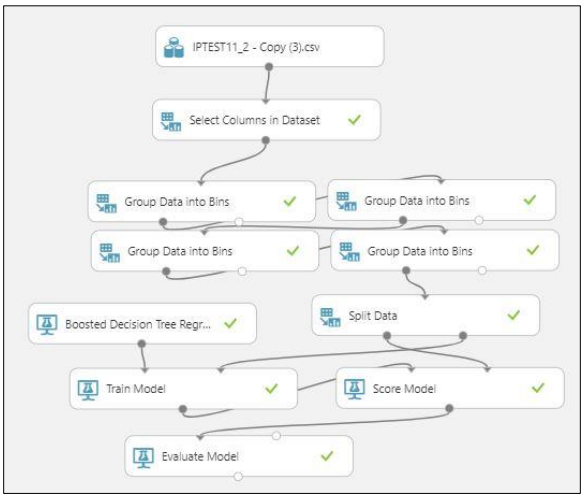

Figure 6: Flow chart showing steps of Binned Decision Tree model building steps.

Comparison and selection of Algorithm

From our experiment, it is evident that Neural Network Regression and Decision Tree yields similar results. The squared error of the initial tests was 0.95 and over the course of 3-4 weeks and many iterations we were able to achieve an error of 0.36. However, there are few caveats. Anything that involves a matrix—e.g., linear regression, logistic regression, NN regression, and so on, is affected by the scale of the input, whereas Tree-based models do not need scaling or normalization. If a model is sensitive to the scale of input features, feature scaling could also be helpful. As the name suggests, feature scaling changes the scale of the feature; it has also been referred to as normalization. Feature scaling is usually done at the individual feature level. Decision Trees, gradient boosted machines, and random forests are not sensitive to scaling, unless the scale of the input grows over time. This can be the case if the feature is an accumulated count, where it will grow outside of the range that the tree was trained on. If that is the case, it might be necessary to rescale the model periodically. In table 1 the results are compared.

	Relative Squared Error	**Coefficient of Determination**
Linear Regression Linear Regression	0.93	0.07
N. Network Regression	0.337059	0.662941
Decision Tree	0.367724	0.632276

Table 1: Prediction Value comparison between NN and Decision Tree.

In this experiment, NN regression methods are not only time consuming but also the "scored model output" is scaled. If one wants to test and compare the effectivity of the prediction model by blind testing or in future wants to deploy it in production, the input needs to be scaled and normalized same way. It is doable but would be time consuming to create and cumbersome to maintain. On the other hand, input in Decision Trees does not need any scaling and the results can be compared readily, and the runtime is at least 8-times faster than that of the NN regression methods. Table 2 summaries the comparison between the two methods.

NN Regression	**Decision Tree**
Slower (both for training and classification)	Faster than NN
Output is not intuitive and usually requires de-scaling	Output can be consumed raw by end users
Need scaling and normalization	No scaling and/ normalization needed
Can model more arbitrary functions (nonlinear interactions, etc.) and therefore might be more accurate, provided there is enough training data. However, it can be prone to over-fitting as well	Need to prune the tree to avoid over-fitting.
No hidden layers	Hidden layers

Table 2: Comparison of Neural Network Regression and Decision Tree methods.

Results and Further Discussion

Upon reviewing all models, their usefulness, drawbacks, and ease of use, the decision was made to use Decision Tree. The error prediction reduced from .95 to .3 which is a significant improvement. Further feature engineering and ensemble training methods are expected to reduce observed error even more. However, because the goal of any machine learning problem is to find a single model that will best predict the observed outcome, here it was no different. A single machine learning model must be finalized for future deployment that can predict the initial production with an acceptable error range. To make sure, the selected model is the best-suited one and best/most accurate predictor, and an "ensemble methods testing" is the best practice. It can take a myriad of models into account and average those models to produce one final model. Thus, Decision Forest method was tried to confirm that the selected Decision Tree is the most suitable model for the provided dataset. However, it is important to note that Decision Trees are not the only form of ensemble methods, just the most popular and relevant in data science today and because they work best with limited data with physical limitations.

Acknowledgments

The authors would like to express our thanks to Jonah Energy for allowing us to use their data for our experiments, for their valuable contributions and feedback throughout the process, and their kind permission to publish our findings. We would also like to thank Stonebridge Consulting for their support for and commitment to the project and for their expertise in finalizing this paper.