Cooper Smidt and Chuck Agu

Dr. Tyagi

Pete 4241

12/1/2018

Final Report

Our project began with choosing a topic. We chose to do LSTM modeling on production data because we understand the value of the ability to accurately predict well production. After choosing this topic, we dove in to exploratory data analysis on the QRI dataset, specifically the Euler field in the Athena formation. At the onset, we agreed that wells with more data would be better to model, so we used python to help us select a well. After analyzing which wells had the most data, we decided to select well 21.

For our first experiment, we applied our LSTM algorithm to oil production data curated from Well 21 in the Euler field. We used an LSTM algorithm from machinelearningmastery.com for our model, and used the root mean square error score (RMSE) to give us a quantitative idea on how effective our model was in training our production data. We ran the model without cleaning any of the data and observed an RMSE score of 816. We thought that we could decrease the error score by removing all of the non-productive times; predicting when exactly well would be shut-in was not part of the scope of the project. After we deleted 73 months of shut-in times, the model achieved an error score of 231.5. This significant improvement confirmed our theory on getting rid of shut-in times to decrease error.

After testing Well 21, we hypothesized that the less shut-in times removed, the higher the performance of the algorithm. We thought this would be true because shutting in the well changes parameters such as the average pressure of the reservoir and the state of the flow (transient, pseudo-steadystate), and changing these parameters will change the amount of production. We used python again to help us select a well with a relatively large amount of datapoints and a small proportion of shut-in times to producing times. Well 2 matched our criteria with 228 producing months and 15 shut-in months; Well 21 had about 14% of its life shut-in, while well 2 only had 6% of its life shut-in. After running the algorithm with the same tuning parameters, we recorded an error score of 224, which was 7 less than the error score of Well 21. This was not the significant improvement we were looking for, but it gave us some insight as to how much of a factor shut-in times play in model performance. An interesting project going forward would be using PCA to determine the impact shut-in time has on model’s performance.

The next phase of our project consisted of tuning the algorithm my changing the number of epochs and neurons. The different starting conditions built into the algorithm yield slightly different results each time the algorithm is run, so we decided to loop through the algorithm 30 times at each given condition and recorded the mean RMSE score and the variance of the RMSE score. We initially thought that more neurons and epochs would decrease the error score, but we did not observe this direct relationship. For Well 21, the lowest error score of 231.28 was achieved with 3 epochs and 8 neurons, while the highest error score of 263.7 was achieved with 15 epochs and 15 neurons. For Well 2, more epochs and neurons generally reduced the error score; the lowest error score of 219 was achieved with 15 epochs and 15 neurons, while the highest error score of 228.44 was achieved with 2 epochs and 2 neurons.

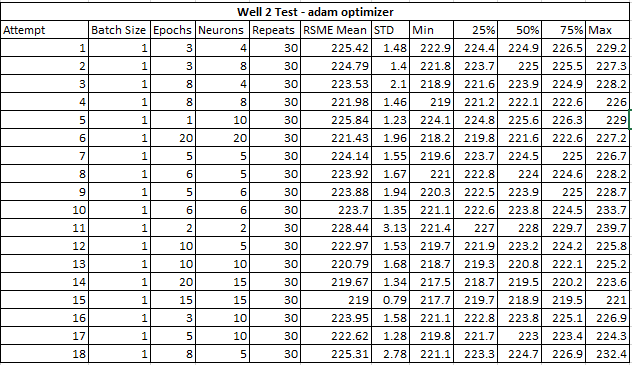
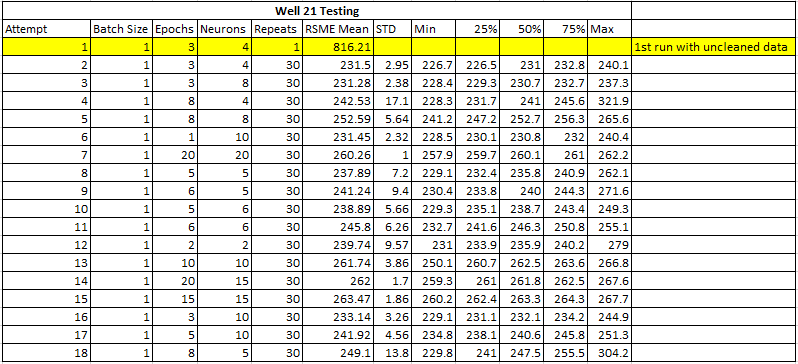
The final topic we investigated before our presentation was comparing the performance of the adam optimizer and the stochastic gradient decent (SGD) optimizer using data from Well 2. We started this project using the adam optimizer for both Well 21 and Well 2; we generated data using the SGD optimizer and the same configurations of epochs and neurons as we used previously. We noticed the SGD optimizer consistently generated higher error scores, and we decided to statistically confirm a difference between the performance using a t-test. From the scipy.stats module, we utilized the ttest\_ind function to determine a difference between the means of the groups. The null hypothesis was that the means of the RMSE scores for the adam optimizer and the SGD optimizer were the same. With a P value of .005, we rejected the null hypothesis, which indicates that there is a difference in the performance of the optimizer. When using the adam optimizer, the model achieved lower RMSE scores and executed the code much quicker.

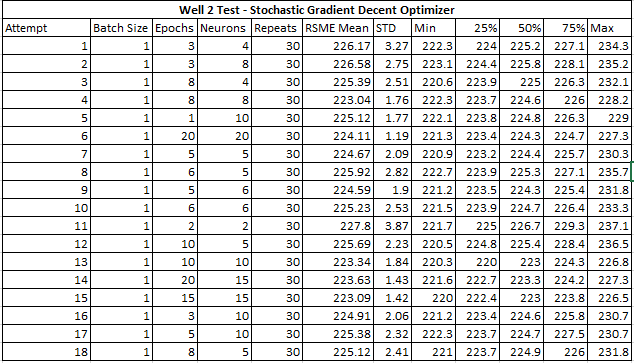
After completing our experiments, we continued reading up LSTM modeling and realized we made a mistake: we did not compare the RMSE score of the training set with the RMSE score of the test set. It is vital to compare these scores to determine if the model is properly fit. Also, the number of epochs we chose were not sufficiently high. Our post-presentation work focuses on remedying this mistake.

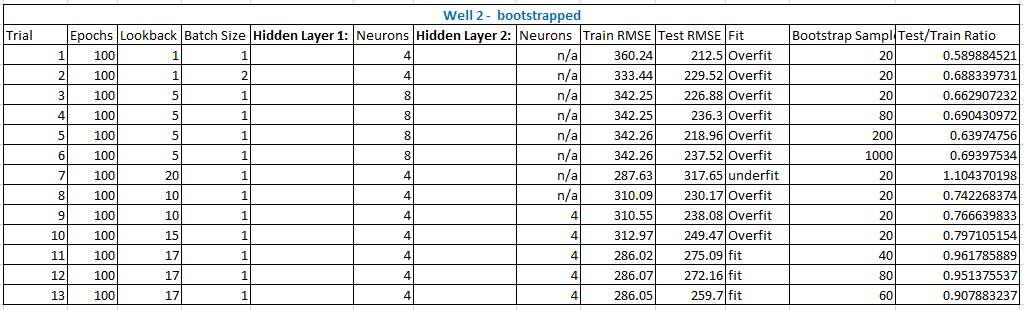
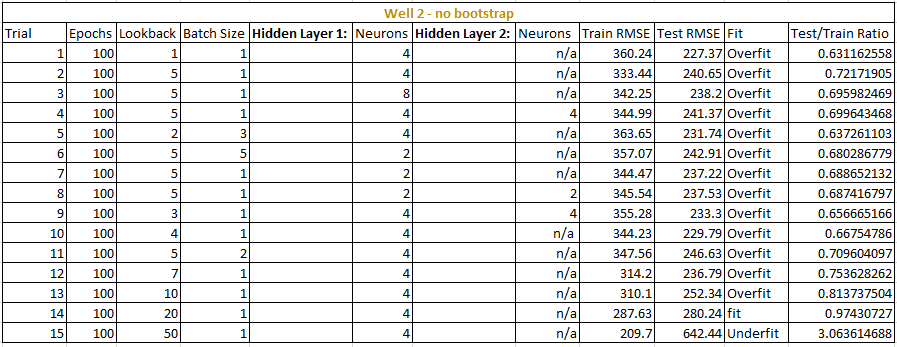
In order to print the RMSE score for the training set, we switched algorithms, using the LSTM algorithm posted on the Moodle page. We chose to tune the model for Well 2; we accepted the fitness of the model if the ration of test RMSE to train RMSE was greater than .9. We toggled with the number of lookbacks, batch sizes, hidden layers, and number of neurons in the hidden layers for the first 9 trials. The best test/train ratio was achieved at trial number 2, with 5 lookbacks, a batch size of 1, 1 hidden layer, and 4 neurons. We tuned this configuration from trials 10 to 15, and at trial 14, we found a configuration that demonstrated a good fit. The lookback feature turned out to be the most important parameter to tune during this process.

After finding a good fit for the model, we tried employing the bootstrapping method to increase the test size. For the first 3 trials, we toggled with the lookback, batch size, and number of neurons, and increases the test size by 20 bootstrapped samples . Comparing the results to the non-boostrapped trials, we observed worse performance in the model. In trials 4-6, we toggled with the number of bootstrapped samples, and saw no significant change in performance. Starting at trial 7, we used the best fit configuration for the non-bootstrapped experiment and began tuning parameters with 20 bootstrapped samples. The first results displayed an underfit model; the training score performed better than the test score. At trial 8, we tuned the lookback down to 10, and returned to overfitting the model. To fix this overfitting, we decided to add another hidden layer, and at trial 11, we recorded a fit model. The error score at trial 11 was less than the error score in the best non-bootstrapped model, which tells us that the bootstrapping was effective. We then sought out the optimal number of bootstrapped samples that had the lowest error score while still having a test to train error ratio greater than .9, and at trial 13, we observed our best version of the model with a bootstrapped sample size of 60 and a RMSE score of 259.7.

This project served as a great first step in exploring the world of machine learning. The first time the code *finally* ran correctly was one of the greatest technological feats of our lives. Seeing how easy it was to use machine learning and to generate results made the entire enterprise less daunting. The immense amount of online readings and resources has made the opportunity to become a data scientist a feasible goal. Thank you for guiding us on our academic journey into machine learning; we are excited to continue our education on this subject for our senior design project.

**Results from tuning model and comparing optimizers before presentation**



**Results from fitting the model on Well 2**