Capstone Project 1 Final Report

1. Introduction: Synthetic Sonic Curves Generation

Well logs are interpreted/processed to estimate the in-situ petrophysical and geomechanical properties, which is essential for subsurface characterization. Various types of logs exist, and each provides distinct information about subsurface properties. Certain well logs, like gamma ray (GR), resistivity, density, and neutron logs, are considered as “easy-to-acquire” conventional well logs that are run in most of the wells. Other well logs, like nuclear magnetic resonance, dielectric dispersion, elemental spectroscopy, and sometimes sonic logs, are only run in limited number of wells.
Sonic travel-time logs contain critical geomechanical information for subsurface characterization around the wellbore. Often, sonic logs are required to complete the well-seismic tie workflow or geomechanical properties prediction. When sonic logs are absent in a well or an interval, a common practice is to synthesize them based on its neighboring wells that have sonic logs. This is referred to as sonic log synthesis or pseudo sonic log generation.

Compressional travel-time (DTC) and shear travel-time (DTS) logs are not acquired in all the wells drilled in a field due to financial or operational constraints. Under such circumstances, machine learning techniques can be used to predict DTC and DTS logs to improve subsurface characterization. The goal of the “SPWLA’s 1st Petrophysical Data-Driven Analytics Contest” is to develop data-driven models by processing “easy-to-acquire” conventional logs from Well #1, and use the data-driven models to generate synthetic compressional and shear travel-time logs (DTC and DTS, respectively) in Well #2. A robust data-driven model for the desired sonic-log synthesis will result in low prediction errors, which can be quantified in terms of Root Mean Squared Error by comparing the synthesized and the original DTC and DTS logs.
You are provided with two datasets: train.csv and test.csv. You need to build a generalizable data-driven models using train dataset. Following that, you will deploy the newly developed data-driven models on test dataset to predict DTS and DTC logs. The data-driven model should use feature sets derived from the following 7 logs: Caliper, Neutron, Gamma Ray, Deep Resistivity, Medium Resistivity, Photo-electric factor and density. The data-driven model should synthesize two target logs: DTC and DTS logs.

The predicted values should be in the same format as sample_submission.csv, and submit together with your notebook for evaluation.

2. Description of the dataset, how you obtained, cleaned, and wrangled it

All the values equals to -999 are marked as missing values.

- CAL - Caliper, unit in Inch,

- CNC - Neutron, unit in dec

- GR - Gamma Ray, unit in API

- HRD - Deep Resisitivity, unit in Ohm per meter,

- HRM - Medium Resistivity, unit in Ohm per meter,

- PE - Photo-electric Factor, unit in Barn,

- ZDEN - Density, unit in Gram per cubit meter,

- DTC - Compressional Travel-time, unit in nanosecond per foot,

- DTS - Shear Travel-time, unit in nanosecond per foot,

The test data has all features that you used in the train dataset, except the two sonic curves DTC and DTS.

For evaluation: We will be evaluated by the metirc. RMSE. DTC and DTS are in the same weight during the evaluation. Understanding and optimizing your predictions for this evaluation metric is paramount for this competition.

3. Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an open-ended process where we calculate statistics and make figures to find trends, missing values, outliers, anomalies, patterns, or relationships within the data. First, a few conventional EDA approaches were performed to understand the dataset.

	CAL	CNC	GR	HRD	HRM	PE	ZDEN	DTC	DTS
count	20525.000000	20525.000000	20525.000000	20525.000000	20525.000000	20525.000000	20525.000000	20525.000000	20525.000000
mean	8.426679	0.274416	49.889253	2.598719	5.835466	3.833792	2.410734	88.312221	182.051067
std	1.845912	3.062495	54.811017	3.465665	422.449589	4.375818	0.181713	23.542419	84.670122
min	5.930400	0.014500	1.038900	0.123600	0.134100	-0.023200	0.680600	49.970500	80.580400
25%	6.629100	0.120300	16.036800	0.810000	0.797300	0.049800	2.236100	70.423100	127.148800
50%	8.578100	0.187700	37.498000	1.814900	1.829300	3.287800	2.466500	79.695400	142.678500
75%	8.671900	0.329000	61.140700	3.337400	3.463300	7.061300	2.563700	102.482800	192.757800
max	21.064200	365.885000	1470.253400	206.718200	60467.761700	28.106400	3.259700	155.980300	487.438400

Figure 1: Description of the dataset, about its statistic values.

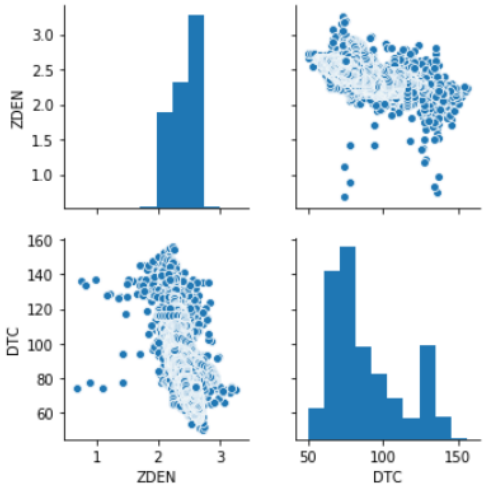


Figure 2: Pair plot comparisons of the density log and DTC log.

4. Filling in Missing values

The missing values from different logs were investigated, and it seems that two targets in training files contain large amount of missing values, about 25%. The other 7 logs contain less than 2.5% amount of missing values. From the initial plot of different logs, it can be seen that missing values in other logs look like huge spikes, but DTC and DTS contain large chunks of continuous missing values.

	Missing Values
CAL	510
CNC	735
GR	254
HRD	385
HRM	385
PE	579
ZDEN	681
DTS	4865
DTC	4054

Figure 3: Value counts of missing values in logs.

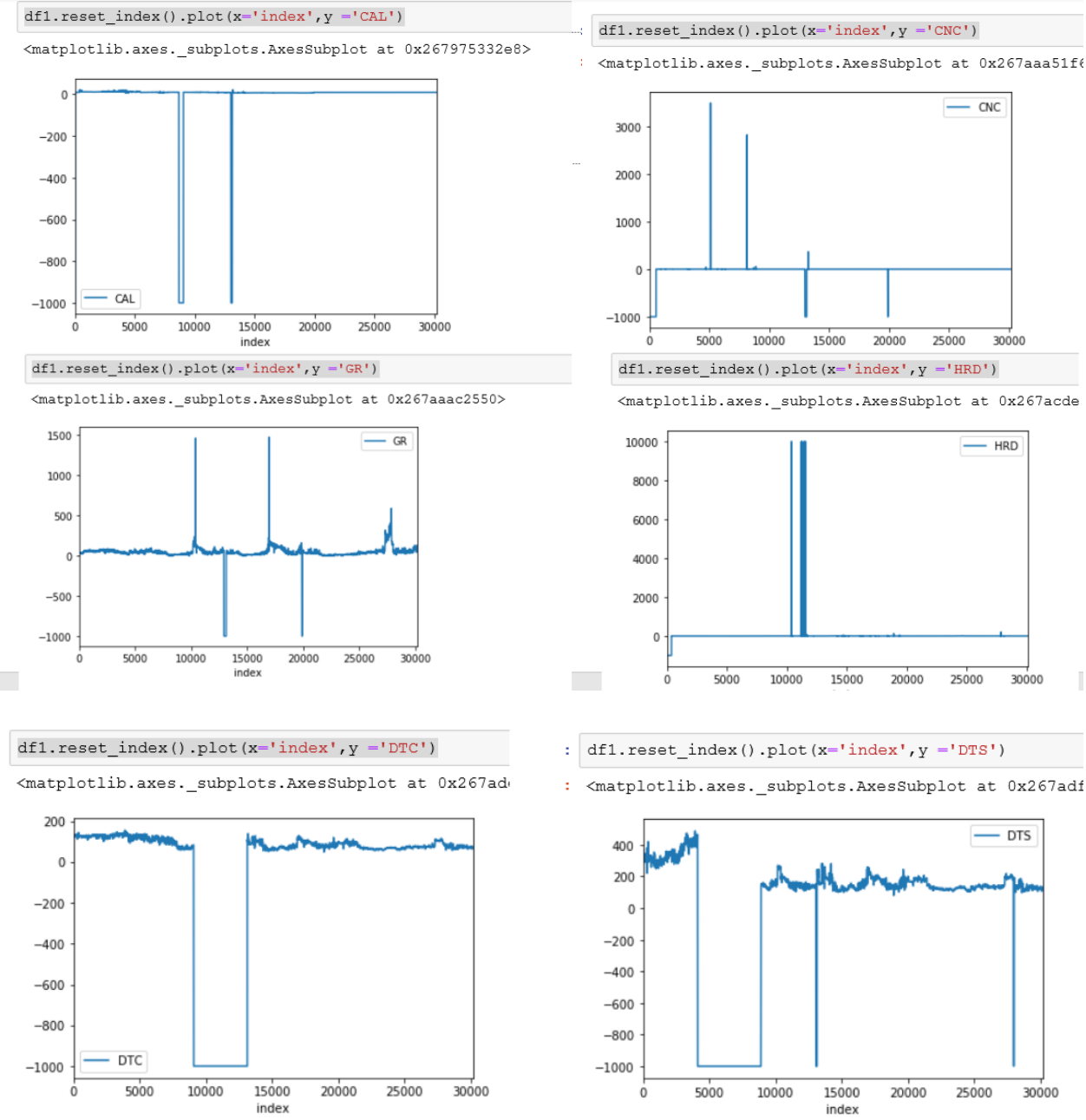


Figure 4: Plot of different logs.

5. First fill missing values by using Peudo-labeling method

First, ML modeling based approach is used to fill in missing values. This approach uses target without missing values to develop a ML model, such as XGBoost. Then this model is used to fill in missing values of the target.

However the filled in missing values do not improve the validation accuracy. There is no big difference after filled DTC missing values. From the RMSE values measured on validation data, the result is similar to that with missing values (2.8).

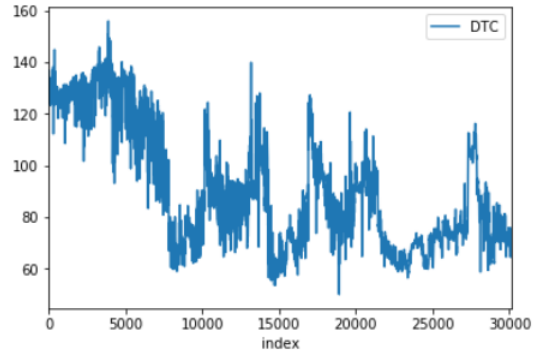


Figure 5: Figure of DTC log with filled in missing values.

Second, Psudolabel method is used to fill in missing values.

DTC RMSE is 40% compared with DTS for Tree based model:

DTC		DTS	
Predicted from features only	2.73	Predicted from features only	7.17

The RMSE for both DTC and DTS is: 5.44

This RMSE difference exist in both XGBoost and Random Forest. The accuracy between these two methods are negligible. Although Yu's original notebook showed RMSE error around 3.7. But that is on training data only. His RMSE error for test data is similar to XGBoost.

Apply PseudoLabel to fill in missing values:

Applied sampling rate test: only add a various percentages of filled in DTC values, and tested CV score, to decide how much sampling rate of missing data can be added into training data to improve final prediction.

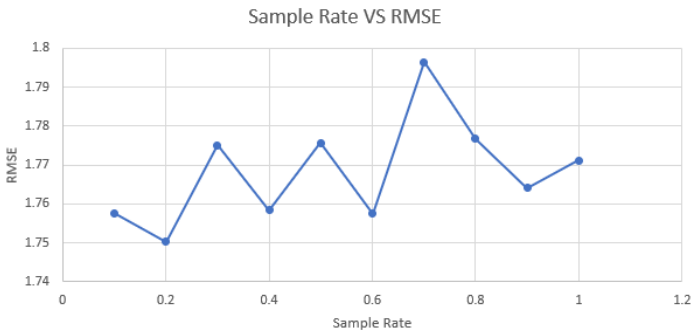


Figure 6: The accuracy against varying percentage of psudolabeled data.

I used 0.9 for sampling rate to ensure more psudolabeled data for training. If the sample rate is too small, there is no point for doing this.

DTC RMSE without missing values fill in		DTC RMSE with missing values fill in	
Predicted from features only	2.73	Predicted from features only	2.66

DTS RMSE without missing values fill in		DTS RMSE with missing values fill in	
Predicted from features only	7.17	Predicted from features only	7.66

Figure 7: Comparison of Pseudo labeled data to fill in missing data.

From the test, it can be seen that Pseudo label method achieved 2.5% improvement in DTC, no improvement of DTS.

In Depth Analysis

After initial data exploration, an in-depth Machine Learning algorithm testing was performed.

1. Separated Different Zoning by Petrophysics Chracteristic

Petrophysics-based zonation for both training and test set based on GR, ZDEN, CNC. Compared to test set, I found the training set showed similar spatial patterns, to take advantage of the spatial correlation. I performed inter-well correlation. Per-zone estimation has been proved to be a good preprocessing method for well-log interpretation (Pan et al., in press; Pan et al., 2019).

For this dataset, there are 5 zones in the training dataset, as shown in the following figures with different colors representing different zones:

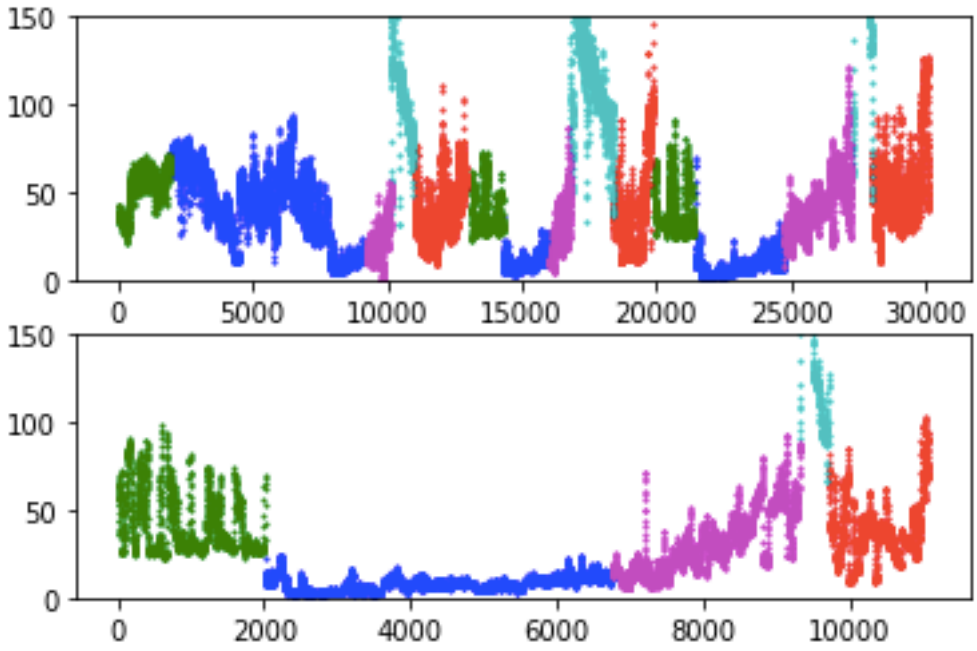


Figure 1: Top figure shows training data rezoning. Bottom figure shows test data rezoning distribution.

Then the cross plots of different well logs is shown below. as expected, different zones show different correlations and spans of data.

2. Preprocess data to reduce impact of miscorrelation between index and depth

Preprocessing: 10000 data points in the test well should roughly correspond to 5000 ft, which is too long for a well, thus I think it could be resampled. The artifacts in the cross plots (continuous dots) also indicate the existence of resampling. To alieviate the aliasing problem caused by resampling, and try to restore the true correlation, I calculated gradients to differentiate real data from interpolated data, and use medidan filter to capture the general trends of different well logs. I also perform logrithmic transformation to resistivity logs to avoid large weight on resisitivity logs.

Figure 2: Cross plot of different logs shows clear different patterns of different zones.

```
df1['HRD_log']=np.log(df1.HRD.values)
df2['HRD_log']=np.log(df2.HRD.values)
df1['HRM_log']=np.log(df1.HRM.values)
df2['HRM_log']=np.log(df2.HRM.values)

# Calculate the gradients of input logs
# Interpolated well logs should have similar gradients
for i in df1.keys():
    df1[i+'grad']=np.gradient(df1[i].values)
for i in df2.keys():
    df2[i+'grad']=np.gradient(df2[i].values)
#Observe the patterns of logs
```

Figure 3: This code snippet shows how to perform gradient calculation on different well logs.

3. Model Building with Deep Learning Approach

I Perform sonic logs estimation for each zone. Caliper log is not used because there is no strong correlation between caliper log and sonic logs, sonic logs is sensitive to the density and porosity of the formation, thus the gradients of logs that measures the density and porosity of the formation are used as input features.

There are two more processing steps were adopted before using the ML model: (1)Min and Max scalar was applied to the data for normalization. (2) Median filter was applied to reduce the amount of transient noises existing in data.

After preprocessing a simple fully connected ANN network, with very few nodes, was applied. The first layer has 24 nodes, second layer 8 nodes, and then connected to a dense layer before dropout.

```
RMSE of test data (#1 DTC): 4.77
RMSE of test data (#2 DTS): 17.28
Overall RMSE = 12.68
```

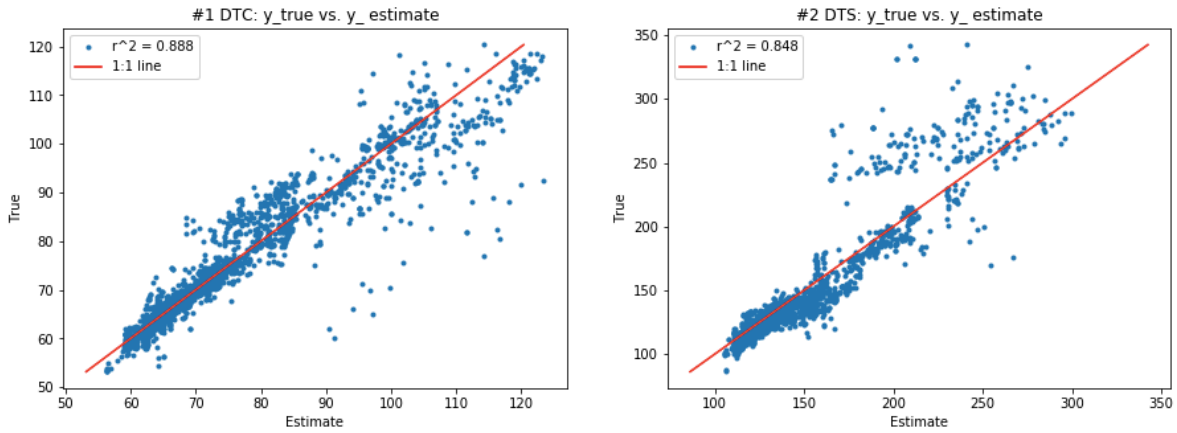


Figure 4: Comparison of predicted data and test data shows good agreement.

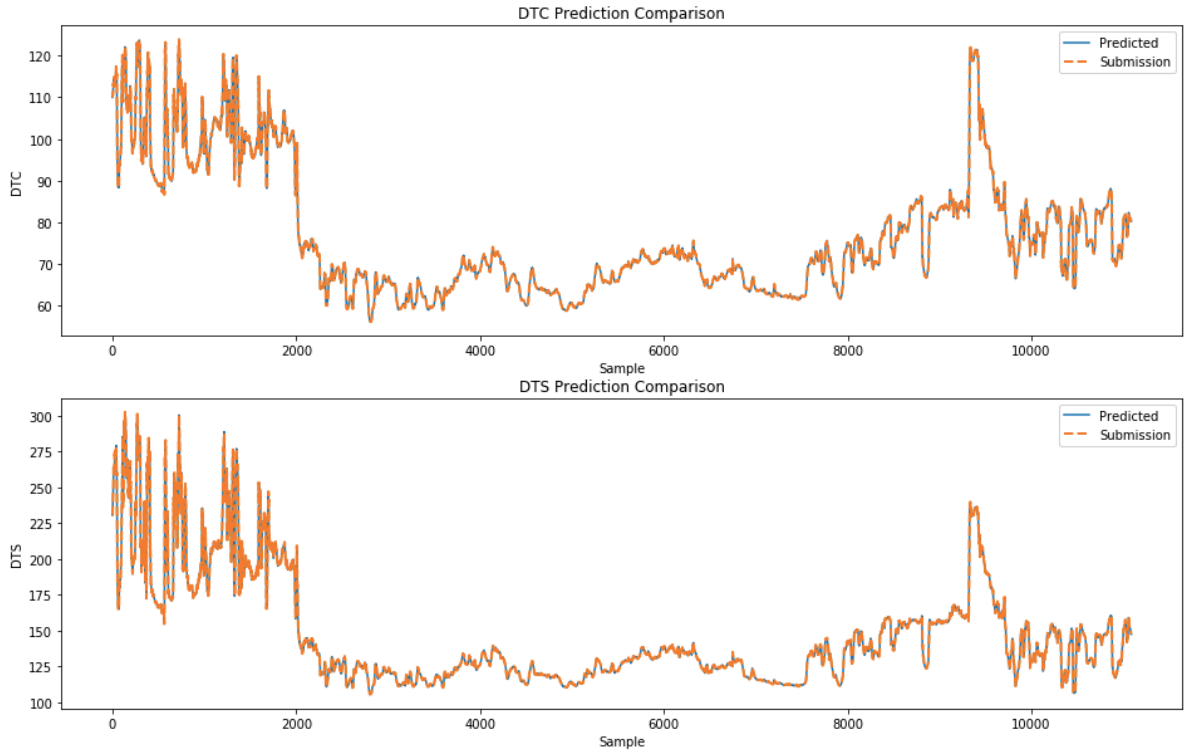


Figure 5: Prediction results plot.

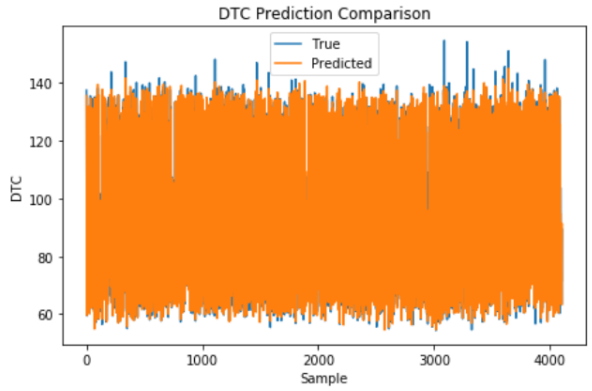
4. Comparison of Different ML Model Performance

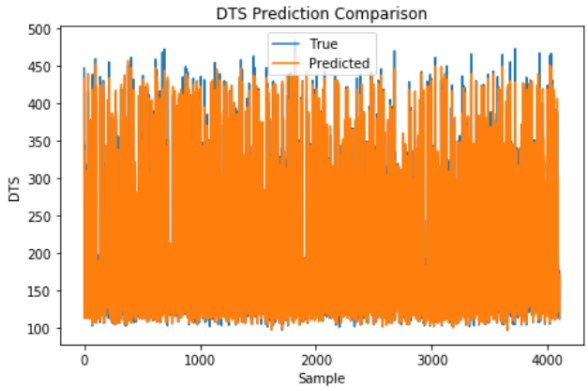
Besides the simple ANN network, other types of ML algorithms are also tested for their performances. First, XGBoost was also tested in this project, and the following table shows hyperparameter tuned with XGBoost for performance improvement.

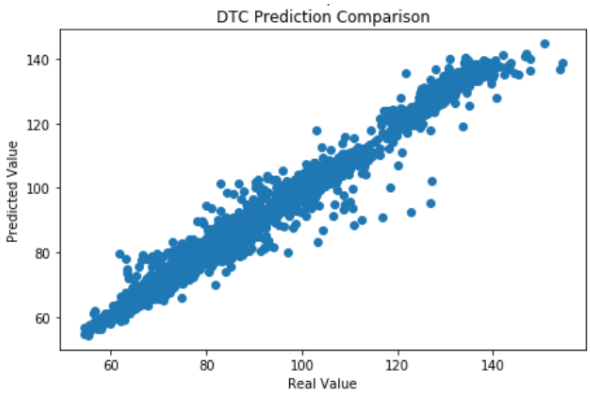
	score	max_depth	min_child_weight	learning_rate	n_estimators	colsample_bytree	subsample	gamma
7	0.989576	17	2	0.05	180	0.9	0.9	0.6
6	0.989536	17	2	0.05	160	0.9	0.9	0.6
5	0.989448	17	2	0.05	140	0.9	0.9	0.6
4	0.989368	17	2	0.05	130	0.9	0.9	0.6
3	0.989231	17	2	0.05	120	0.9	0.9	0.6
2	0.988953	17	2	0.05	110	0.9	0.9	0.6
1	0.988422	17	2	0.05	100	0.9	0.9	0.6
0	0.984736	17	2	0.05	80	0.9	0.9	0.6

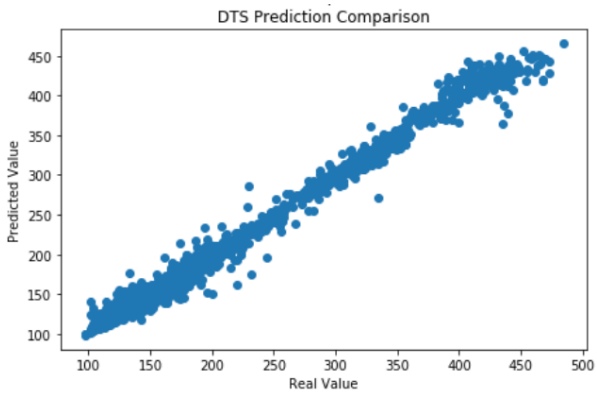
After tuning these parameters, the XGBoost algorithm was applied for prediction. The accuracy of the predicted and target values are 0.99.

```
R2: 0.9901
Root Mean Square Error is: 5.43997
```









LSTM algorithm alone:

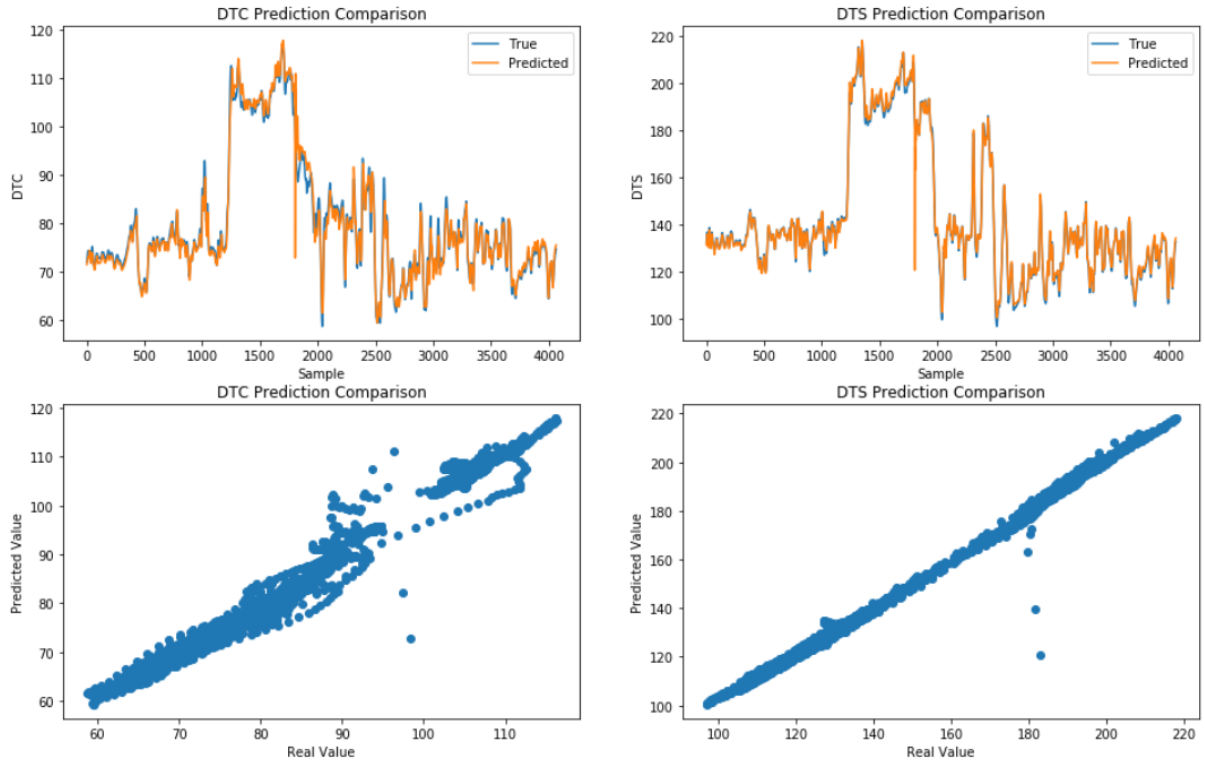
Then LSTM is applied, no change on Min-Max scaler and other data preprocessing steps. I used re-shape to prepare data for Multi-variate LSTM. Then two layer LSTM, with each layer 256 neurons, history window size 40, epoch 100, Relu activation, MSE loss. The initial result is promising, RMSE=2.84006, R2=0.9679.

CNN + LSTM algorithm:

To improve the deep learning algorithm efficiency, a two layer CNN network is added in front of the LSTM network. After adding the CNN network, both the efficiency and accuracy are improved. The final R2 value from CNN+LSTM is 0.988. This is comparable to the best XGBoost algorithm.

```
Model: "model_5"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_6 (InputLayer)         (None, 40, 9)             0
_________________________________________________________________________________________________
conv1d_9 (Conv1D)            (None, 40, 128)           5888        input_6[0][0]
_________________________________________________________________________________________________
conv1d_10 (Conv1D)           (None, 40, 128)           82048       conv1d_9[0][0]
_________________________________________________________________________________________________
max_pooling1d_5 (MaxPooling1D)  (None, 14, 128)        0           conv1d_10[0][0]
_________________________________________________________________________________________________
conv1d_11 (Conv1D)           (None, 14, 256)           164096      max_pooling1d_5[0][0]
_________________________________________________________________________________________________
conv1d_12 (Conv1D)           (None, 14, 256)           327936      conv1d_11[0][0]
_________________________________________________________________________________________________
max_pooling1d_6 (MaxPooling1D)  (None, 5, 256)         0           conv1d_12[0][0]
_________________________________________________________________________________________________
lstm_9 (LSTM)                (None, 5, 256)            525312      max_pooling1d_6[0][0]
_________________________________________________________________________________________________
lstm_10 (LSTM)               (None, 5, 256)            525312      lstm_9[0][0]
_________________________________________________________________________________________________
time_distributed_5 (TimeDistrib (None, 5, 128)         32896       lstm_10[0][0]
_________________________________________________________________________________________________
dropout_5 (Dropout)          (None, 5, 128)            0           time_distributed_5[0][0]
_________________________________________________________________________________________________
flatten_9 (Flatten)          (None, 640)               0           dropout_5[0][0]
_________________________________________________________________________________________________
flatten_10 (Flatten)         (None, 640)               0           dropout_5[0][0]
_________________________________________________________________________________________________
dense_14 (Dense)             (None, 1)                 641         flatten_9[0][0]
_________________________________________________________________________________________________
dense_15 (Dense)             (None, 1)                 641         flatten_10[0][0]
=================================================================================================
Total params: 1,664,770
Trainable params: 1,664,770
Non-trainable params: 0
_________________________________________________________________________________________________
None
Train on 16380 samples, validate on 4065 samples
```

```
R2: 0.9882
Root Mean Square Error is: 1.76934
```



5. Summary and Future Direction

In this project, a well log data science model is developed. Different algorithm, XGBoost, LSTM and CNN+LSTM are used for prediction. XGBoost has the best accuracy with R2 = 0.99. Deep neural network, CNN +LSTM is also impressive, with R2= 0.98.

In the future, more data from different wells will be tested. The developed will be tested for its accuracy against more data collected.