Automated Well Modelling -BHP (Bottom Hole Pressure) Forecasting

A Project Report Submitted
for the Open Ended Lab/Project
(Course No.: ID 3801)

by

Om Prakash Tiwari (132001027)

under the guidance of

Dr. Sahely Bhadra

Chapter 1

Introduction

Time series forecasting is currently a buzz in today's world. Numerous industries use forecasting in a variety of ways. Weather forecasting, climate forecasting, economic forecasting, healthcare forecasting, engineering forecasting, finance forecasting, retail forecasting, business forecasting, environmental studies forecasting, social studies forecasting, and many more practical applications are among them.

We frequently employ Arima models, which are built on auto regression and the moving average, in traditional time series analysis. For smaller datasets, it performs far better, while for larger datasets, the error significantly rises. Only univariate time series models can be used with the Arima model. We can utilise the Sarimax for multivariate models, which can be applied to time series variables with seasonal trends.

The above model is not that affective for large multivariate time series forecasting so we need some other type of forecasting models which could map the relation between different input variables and correctly predicts/forecasts the time dependent variable.

There are a variety of time series models like Arima, reformer, lstma, prophet, informer and many more but most of them fail to forecast for a longer sequence (more than 48 days).

Let's take a short introduction to Arima Model.

Arima:

This approach involved considering a value Y(time-dependent variable) at time point t and adding/subtracting based on the Y values at previous time points (e.g., t-1, t-2, etc.), and also adding/subtracting error terms from previous time points.

$$Y(t) = c+\phi 1ydt-1+\phi pydt-p+...+\theta 1et-1+\theta qet-q+et$$

ARIMA models are typically expressed like "ARIMA (p, d, q)", with the three terms p, d, and q defined as follows:

- In order to improve predictions based on local periods of growth/decline in our data, we must add/subtract p number of previous ("lagged") Y values from Y in the model. This encapsulates ARIMA's "autoregressive" characteristics.
- The number d indicates how many times the data must be "differenced" in order to create a stationary signal (i.e., a signal that has a constant mean over time). This perfectly encapsulates ARIMA's "integrated" character. If d=0, the model is already "stationary," meaning that our data tends to stay the same over the long term. In this instance, you are technically doing ARMA alone rather than AR-I-MA. If p is 1, then the data is increasing or decreasing linearly. If p is 2, the data is increasing or decreasing exponentially.
- The error term's q represents the number of preceding or lagging error values that are added to or subtracted from Y. This encapsulates ARIMA's "moving average" component.

Note: we will see informer model in detail below (This serves as the basis for our training and forecasting)

Chapter 2

Challenge Statement and Research Plan

Challenge Statement

The primary scope of this project is to forecast the bottom hole pressure (BHP) and production rates for six months. Input is production data for the well (multiple features). Output is Bottom Hole Pressure (BHP). (Because the pressure changes in the reservoir are caused by production withdrawal). There is a network of 5 production and 2 injection wells (Inject water volumes to various production wells) is also dependent on each other.

Current scenario

- Current approaches are Incapable of Predicting well production with minimal uncertainty.
- Long term prediction with high accuracy is challenging itself.
- Data has both time dependency and graph-based dependency. Which can be really hard to model.
- Time consuming

Chapter 3

Materials and Methods

Data Source:

The data utilised in this report is from a specific corporation, and by using it, we can monitor the operation of these wells and can predict the data needed to determine the well's remaining life in the future.

Bottom-hole data and well production data can be used to predict the future BHP(bottom hole pressure) that will certainly help in –

- Assessing the well's life
- Visualizing the well's IPR (inflow performance relationship) changes with time
- Will be an input for the production optimization of the well
- Will help in assessing when the artificial lift is needed in the well

Approach:

We aim to utilize the combined work by (Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, Wancai Zhang) in Informer forecasting model: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting.

Informer Model Details and Implementation:

Informer is an upgraded version of the Transformer model that maintains a larger prediction capacity while being more efficient in terms of architecture, memory usage, and computation. There are several severe issues with Transformer that prevent it from being directly applicable to LSTF (Long sequence Time series forecasting), including quadratic time complexity, high memory usage, and inherent limitation of the encoder-decoder architecture. Transformer models have outperformed RNN models in terms of capturing long-range dependency. They are actually used for the NLP tasks and yield impressive results but on the cost of dozens of GPUs and expensive deploying cost. When used to solve LSTF problems, the self-attention mechanism and Transformer architecture's efficiency become the bottleneck.

Deep learning techniques are implemented, mainly develop an encoder-decoder prediction paradigm by using RNN and their variants. Hosted on google colab and can be accelerated using GPU. Implemented and trained using Pytorch framework along with torch library.

- **Architecture** - encoder-decoder is used with RNN and their variants. (see fig.- 1)

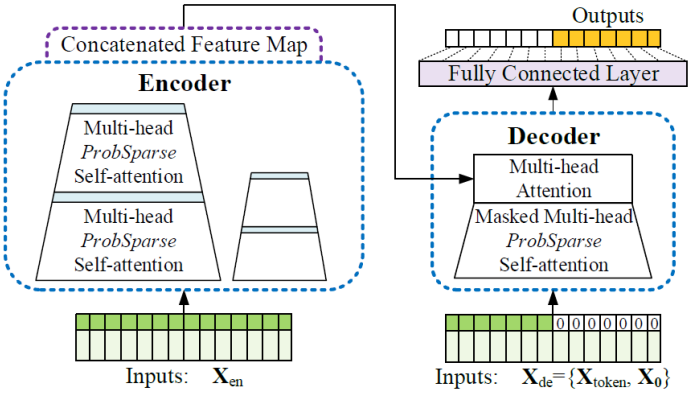

Fig - 1

Left: The encoder receives enormous inputs of extended sequences (green series). The proposed Prob-Sparse self-attention is used in place of canonical self-attention. The self-attention distilling process, which uses a blue trapezoid to extract dominating attention and significantly shrink the network size. The replicas' robustness is increased by layer stacking.

Right: The decoder takes in long sequences as inputs, nullifies the target elements,

analyses the feature map's weighted attention composition, and then generates instant predictions of the output elements (orange series).

Self-attention mechanism – (enables the inputs to interact with one another ("self") and determine which inputs deserve more attention ("attention"))

The canonical self-attention is defined based on the tuple inputs, i.e, query, key and value, which performs the scaled dot-product as

A (Q, K, V) = Softmax$\left(\frac{QK^T}{\frac{1}{d^{\frac{1}{2}}}}\right)$V, where Q $\in R^{L_Q \_ d}$, K $\in R^{L_K \_ d}$, V $\in R^{L_V \_ d}$ and d is the input dimension.

Prob-Sparse Self-attention – (computes a small number of similarity scores from a sequence rather than all potential pairs, which minimizes computation time and the memory needs of the attention mechanism)

Based on the proposed measurement, we have the Prob-Sparse self-attention by allowing each key to only attend to the u dominant queries:

A (Q, K, V) = Softmax$(\frac{\bar{Q}K^T}{\frac{1}{d^{\frac{1}{2}}}})$V $\bar{Q}$ - a sparse matrix of the same size of q and it only contains the Top-u queries under the sparsity measurement M(q, K).

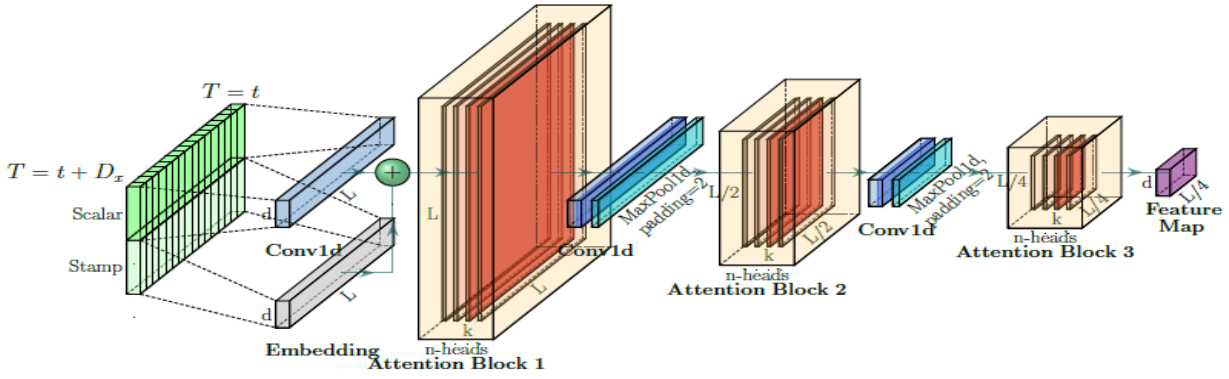

Fig - 2

Encoder: Allowing for Processing Longer Sequential Inputs under the Memory Usage Limitation. (see fig.- 2)

The encoder is designed to extract the robust long-range dependency of the long sequential inputs. After the input representation, the t-th sequence input Xt has been shaped into a matrix Xt(encoder) ∈ 2 R(Lx) x d(model).

$$X(t)\ j+1 = \text{MaxPool}\ (\text{ELU}(\ \text{Conv1d}([X_{t\,j}]AB)\ )\ .$$

It contains the Multi-head ProbSparse self attention and the essential operations, where Conv1d(.) performs an 1-D convolutional filters (kernel width=3) on time dimension with the ELU(.) activation function to detect patterns in the time series.

Decoder: Generating Long Sequential Outputs Through One Forward Procedure. it is composed of a stack of two identical multihead attention layers. However, the generative

inference is employed to alleviate the speed plunge in long prediction. We feed the decoder with the following vectors as Xt(decoder) = Concat(X(t) token,X(t)0) ∈ R(L(token)+Ly) x d(model) . where X(t) ∈ RL(token) x d(model) is the start token, Xt(0)∈ R(Ly) x d(model) is a placeholder for the target sequence (set scalar as 0). It prevents each position from attending to coming positions, which avoids auto-regressive. A fully connected layer acquires the final output, and its outsize d(y) depends on whether we are performing a univariate forecasting or a multivariate one.

Generative Inference (efficiently applied on NLP task)
Let's take predicting 168 (7*24 hours) points as an example (7-day temperature prediction in the experiment section), we will take the known 5 days before the target sequence as "start-token", and feed the generative-style inference decoder with X(decoder) = {X5d,X0}. The X0 contains target sequence's time stamp, i.e., the context at the target week. Then our proposed decoder predicts outputs by one forward procedure rather than the time consuming "dynamic decoding" in the conventional encoder-decoder architecture.

Training

a) Source of data : For training, the data is collected from a certain company. The data for production well and injection well is given. Size of data vary from well to well with a minimum of 2 year data on a daily basis.
b) Data pre processing : Data preprocessing and cleaning part is done mostly using pandas along with numpy python library. The rows consisting of garbage values gets deleted and all the numerical data is converted into float64 so that there won't be any error while training the model. Matplotlib and seaborn library is used to visualize the data and to get deeper insights of the same.
c) Training/Test/Validation data: The data is divided into 3 sets with 0.7% of training data, 0.2% of test data and the remaining validation data. see fig.- 3

	DATEPRD	NPD_WELL_BORE_NAME	X1	Y	X3	X4	X5	X6	x7	X8	X9	X10	X11	X12	X13
0	07-Apr-14	15/9-F-1 C	0.0	0.000	0.000	0.000	0.0	0.00000	0.000	0.00	0.000	0.0	0	0.0	NaN
1	08-Apr-14	15/9-F-1 C	0.0	NaN	NaN	NaN	0.0	1.00306	0.000	0.00	0.000	0.0	0	0.0	NaN
2	09-Apr-14	15/9-F-1 C	0.0	NaN	NaN	NaN	0.0	0.97901	0.000	0.00	0.000	0.0	0	0.0	NaN
3	10-Apr-14	15/9-F-1 C	0.0	NaN	NaN	NaN	0.0	0.54576	0.000	0.00	0.000	0.0	0	0.0	NaN
4	11-Apr-14	15/9-F-1 C	0.0	310.376	96.876	277.278	0.0	1.21599	33.098	10.48	33.072	0.0	0	0.0	NaN

Fig – 3 (Individual production well data with their features)

Note – Date represented in column ("DATEPRD") and all the well data corresponding to each well for a certain date combined in a single row. Which consists of 746 rows and 29 columns, including the NaN and garbage values. (see fig. – 4)

	DATEPRD	NPD_WELL_BORE_NAME_x	X1_x	Y_x	X3_x	X4_x	X5_x	X6_x	x7_x	X8_x	...
0	2014-04-07	15/9-F-1 C	0.0	0.000	0.000	0.000	0.0	0.00000	0.000	0.000	...
1	2014-04-08	15/9-F-1 C	0.0	NaN	NaN	NaN	0.0	1.00306	0.000	0.000	...
2	2014-04-09	15/9-F-1 C	0.0	NaN	NaN	NaN	0.0	0.97901	0.000	0.000	...
3	2014-04-10	15/9-F-1 C	0.0	NaN	NaN	NaN	0.0	0.54576	0.000	0.000	...
4	2014-04-11	15/9-F-1 C	0.0	310.376	96.876	277.278	0.0	1.21599	33.098	10.480	...
...	...	...	...	...	...	...	...	...	...	...	...
741	2016-04-17	15/9-F-1 C	0.0	270.268	101.198	263.770	NaN	0.00000	6.498	9.802	...
742	2016-04-18	15/9-F-1 C	0.0	276.869	101.231	263.155	NaN	0.00000	13.714	11.742	...
743	2016-04-19	15/9-F-1 C	0.0	282.105	101.178	273.530	NaN	0.26519	8.576	10.119	...
744	2016-04-20	15/9-F-1 C	0.0	285.096	101.131	266.816	NaN	0.00000	18.280	9.872	...
745	2016-04-21	15/9-F-1 C	0.0	0.000	0.000	0.000	0.0	0.00000	0.000	0.000	...

746 rows × 29 columns

Fig – 4 (Consists of both production and injection well along with their features)

Some Input Parameters for informer model –

```
args = dotdict()              # standard Python dict. that provides access to its keys as attributes
args.model = 'informer'   # using informer model to train and predict
args.data = 'Custom'       # custom data
args.features = 'M'         # M : multivariate predict multivariate
```

```
args.freq = ‘d’              # frequency for time features – (daily basis)
args.seq_len = 5             # input sequence length of Informer encoder
args.label_len = 5           # start token length of Informer decoder
args.pred_len = 20           # prediction sequence length
args.batch_size = 32         # Batch size of model
args.loss = 'mse’            # loss is mean squared error
```

Training set = 488
Test set = 55
Validation set = 127

Note - In order to make up for the absence of data values in our final training dataset, I reduced the values for sequence, label and prediction length. (final dataset consists of around 700 rows/datapoints).

Chapter 4

RESULTS

- **Future prediction for 20 days -**

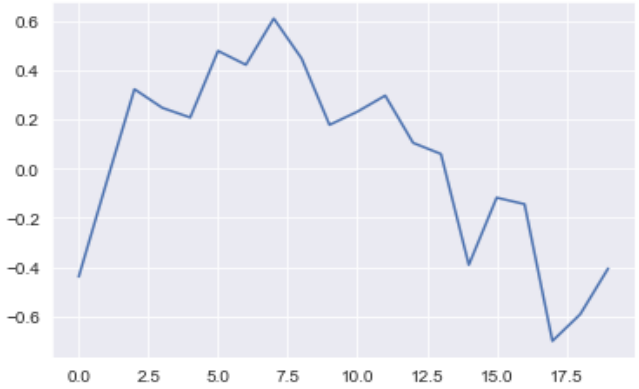

Fig - 5

- **Comparing Truth and Prediction for Test data-set (120 days) -**
  Test shape – (120,20,61)

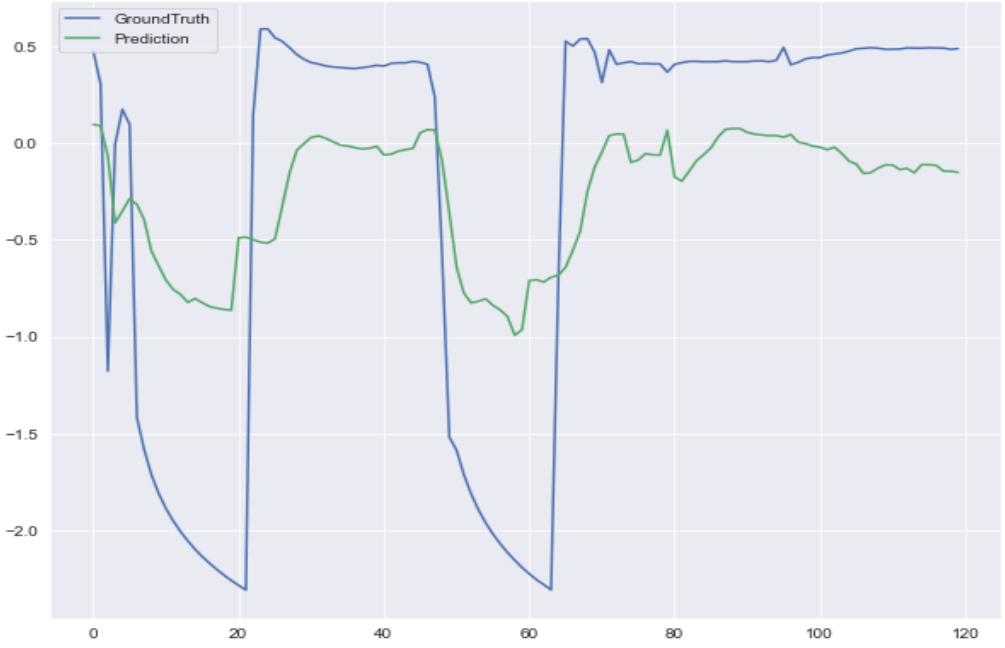

Fig -6

Average training loss – 3.948
Validation loss – 1.65
Test loss – 3.97

Chapter 5

Conclusion and Discussion

We have successfully demonstrated the working of Informer model on a custom dataset and got the results for the same.

We've come thus far, but the real objective is to forecast BHP for six months. Because the final merged data is so small (only roughly 700 rows/datapoints), which is used for the informer model's forecasting and training phases. Here, the data gets limited if we need to forecast for more than 40/50 days because there are only 400–500 training sets, 15–30 validation sets (very less), and 80–100 test sets (we can change this but reducing the training set leads to underfitting of the model). Because of this, the mse (mean squared error) score rises up to 4.

Looking at figure-6 we can say that the model doesn't fit/generalized well due to the lack of data, the same informer model is working very well for the individual well forecasting where we have more than 3000 rows/datapoints.

I can say that the informer model is effective for forecasting long sequence time series but the lack of input data can become a limitation. The more sophisticated informer model can more easily extract long historical features and can model the interaction between the production and the injection well.

References

[1]. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting (research article)

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, Wancai Zhang.

[2]. Github Repository – Data, exp_informer, models, scripts, utils

[3]. Informer model – Google Colab

[4].https://online.stat.psu.edu/stat462/node/188/#:~:text=More%20generally%2C%20a%20lag%20k,the%20observations%20at%20previous%20times.