Deep Learning Approaches for Time Series Forecasting

By

Chindam Sai Dheeraj

Table of Contents

1.Introduction: 3

2.BackGround: 4

3. Approach: 5

3.1 Multiple time series analysis: 6

3.1.1 Deep AR Architecture: 6

3.1.2 N-Beats Architecture: 9

3.2. Multivariate time series Analysis: 11

3.2.1 LSTM: 12

3.2.2 ARIMAX: 13

3.3 The essence of Deep AR and N-Beats for multiple time series forecasting 14

3.4 Detailed breakdown of datasets and approaches implemented for analysis 15

3.4 Analysis and Evaluation 15

4. Implementation: 16

4.1 Multiple time series Analysis Implementations: 16

4.1.1 Deep AR and N-Beats Results and Analysis: 16

4.2 Multiple time series analysis recommendations: 19

4.3 Multivariate time series Implementations: 20

4.3.1 LSTM and ARIMAX Results and Analysis: 21

4.4 Multivariate time series recommendations: 22

4.3. Major drawbacks of deep learning based on experiments: 22

5.Conclusion: 23

6. Future work: 24

7. References: 24

Appendix: 26

Section 1 Project Software Details 26

Section 2: Sample prediction Screenshots 26

Section 3: Screenshots of outputs with respect to various datasets 27

1.Introduction:

In the real world, there are so many prediction problems that require a time component (usually a time series). Time series data mainly involves in data points recorded at certain regular or irregular time intervals. It can be used to either analyse past historical data or predict future outcomes over a certain period [8].

Time series forecasting techniques are widely used in many real-world use cases like passengers' prediction for airlines on a particular season, Inventory stock prediction concerning seasons for grocery stores, household energy estimation over a decade, etc. So, with the growing demand for time series forecasting in multiple industries and various sectors, there is a need for more diverse techniques for better prediction and analysing of challenging data. Handling time series data can be challenging at times, which may have noise and can be also lumpy.

Regular or traditional approaches in time series forecasting can be oversimplified and may involve a lot of pre-processing steps depending on the dataset. These steps can be complex, time-consuming and may require a lot of manual intervention [11].

Some of the limitations of forecasting using classical methods would be

- It would be highly challenging to use traditional methods for multiple time series forecasting problems (which involve creating a generalized model based on multiple different time series models) and growing attributes and dimensions for multivariate time series in real-world scenarios.
- To give account for some determinant factors like seasonality, trends, and other exploratory aspects might involve us to define fine tuning steps, heuristics, and manual interventions which may consume a lot of time [11].
- Classical time series models like ARIMA, AR, MA, and exponential smoothing try to learn from past observations and cannot learn complicated nonlinear relationships, it solely makes predictions based on the most recent history.

The limitations mentioned above can be some blockers for wide-scale utilisations for traditional methods in time series problems. In the last couple of decades, deep learning approaches have been successful in solving many real-world problems. Some of these approaches in natural language processing, audio classification, and image classification outperformed the traditional methods. Since time series forecasting is widely utilized for solving many prediction problems with challenging and increasing volumes in real-world data, there is a need to utilize deep learning approaches for time series forecasting. Many data scientists started to research various deep learning approaches for time series analysis and there are many research papers published on the same [8][11]. This area of deep learning is an active field of research.

Project Objectives:

- The main objective is to implement multiple modern time series models and recommend suitable approaches for use cases like multivariate and multiple time series forecasting based on merits, demerits, supporting implementations and logical analysis.
- For multiple time series forecasting, modern modelling approaches like Deep AR (autoregressive neural networks) and N-beats (Neural basis expansion) need to be developed and compared with effective scenarios and provide recommendations concerning the nature of the data sets.
- For Multivariate time series, LSTM models need to be compared with traditional methods like ARIMAX using datasets with varying characteristics to understand effective scenarios, merits, and demerits and recommend optimal approaches.

More details of implementations were covered in 3(Approach).

2.BackGround:

My previous experience with time series forecasting problems was by using traditional methods like vector autoregression, ARIMA, and SARIMA in which analysis of time series data was done with cause and effect by using tests like Granger causality tests and obtained good results concerning the prediction problem. Due to the limitations mentioned in Section 1 above, I came up with an idea to analyse deep learning approaches and understand their essence. My previous projects on NLP and machine learning projects helped me to understand the basic knowledge of neural networks. One of my previous projects was based on the research publication "effective use of word order for text categorization using CNN" by Rie Johnson [9] where I successfully implemented a deep learning framework of CNN for 1-dimensional word order sequences and obtained good results than the traditional text categorization techniques which did not consider word order. The concepts learned from my previous work can be utilized in this project because time series also involve sequences.

Usually in time series forecasting traditional methods solve most of the time series prediction problems but with increasing volumes, variety (which includes complicated non-linear relationships) and veracity of data, the manual intervention of data might not be beneficial. With increased features for prediction, there is a need to utilize and analyse deep learning approaches for time series forecasting. For scenarios like multiple time series forecasting it is not a good idea to build models separately for each time series, instead, a generalized model based on all-time series models can be developed using deep learning.

Deep learning concepts in this project will be implemented with the help of research publications and web resources like "Amazon Deep AR for multiple time series forecasting" [6] which describes the core architecture of Deep AR, "Deep Learning for time series forecasting" [8] which describes the basic workflow of neural networks for time series, and "time series forecasting with Deep Learning" [7] which gives in-depth concepts involved for deep learning approaches and N-Beats[13] which is original paper for the core architecture. Although deep learning reduces manual interventions, and a lot of pre-processing steps and handles large volumes of data to great extent still it is kind of black box for some prediction problems there can also be some disadvantages associated with it as well. For multiple scenarios, analysis was developed in this project on how and in what scenarios neural networks can be used for better predictions.

3. Approach:

In this project, several models like Amazon's Deep AR (probabilistic forecasting with autoregressive recurrent networks) which uses a stack of recurrent neural networks to produce point-in-time and probabilistic forecasts [6], N-Beats (Neural basis expansion analysis), LSTM (Long short-term memory networks) [7] were implemented for following time series analysis.

1. Multiple time series analysis.
2. Multivariate time series analysis.

3.1 Multiple time series analysis:

In this analysis, two approaches amazon's Deep AR and N-Beats were implemented on the following datasets.

1. Electricity Load Diagrams20112014 Data Set [3]

The dataset can be accessed here [3]. Each column in this dataset refers to unique time series data (electricity consumption of houses) of individual clients from 2011 to 2014. In other words, it contains 370 unique separate time series data for 370 households.

2. Stock Exchange Data [2]:

The dataset can be accessed here [2]. The data was all collected from Yahoo Finance, which had several decades of data available for most exchanges.

3.1.1 Deep AR Architecture:

Deep AR is a supervised learning algorithm for time series forecasting that uses RNNs to produce both probabilistic forecasts and points in time. probabilistic forecasts (like defining a confidence band for the predictions) [6]. Deep AR internally trains an auto-regressive recurrent neural network for producing a probabilistic forecast.

Deep AR can work seamlessly for multiple time series. For example, in retail, it may involve thousands of products, the traditional way is to develop separate time series models for each product but a deep AR model, is the first model that can work to create a single global model for multiple time series of all products.

Deep AR mainly involves utilising LSTM nodes and dense layers to approximate the parameters like mean (μ) and variance (σ) of the Gaussian likelihood function. Final predictions are obtained by maximising the Gaussian likelihood function for the all-time series.

3.1.1.1 Steps involved during the training process

- As shown in figure 3(a) for a given LSTM node at time step ‘t’ it takes input from covariates ‘x_(i,t)’ at the time ‘t’, the target variable value ‘z_(I,t-1)’ from the previous time step which is ‘t-1’ and output of the previous node ‘h_(i,t-1)’ at ‘t-1’.

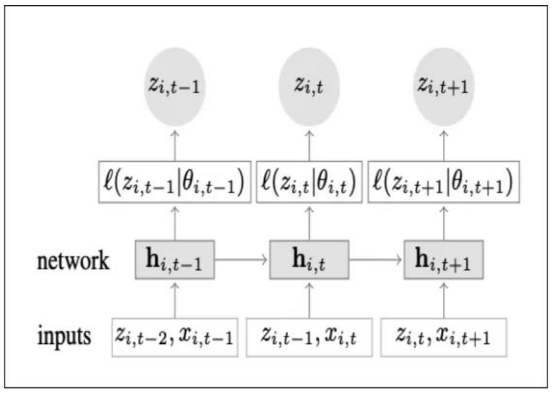


Figure 3(a) Deep AR Training Process Source [6]

- In the next step, the goal of the model is to calculate the best parameters (mean (μ) and variance (σ)) to develop the Gaussian likelihood function (as shown in Fig 3(b)). In statistics to find the best parameters, MLE formulas (maximum log-likelihood estimators) are used which are derived by differentiating likelihood function.

$$\ell_G(z|\mu,\sigma) = (2\pi\sigma^2)^{-\frac{1}{2}} \exp(-(z-\mu)^2/(2\sigma^2))$$

Figure 3(b) Source [14]

- In this approach instead of using differentiation to find these parameters, LSTM nodes and dense layers are shown in Figure 3 (c) will be used to derive optimal parameters (mean and variance) of the distribution.

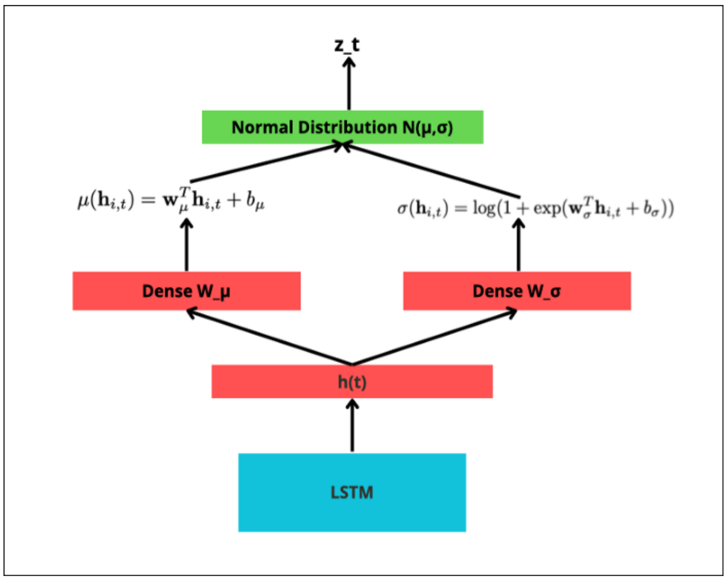

z_t
Normal Distribution N(μ,σ)
$\mu(\mathbf{h}_{i,t}) = \mathbf{w}_\mu^T \mathbf{h}_{i,t} + b_\mu$
$\sigma(\mathbf{h}_{i,t}) = \log(1 + \exp(\mathbf{w}_\sigma^T \mathbf{h}_{i,t} + b_\sigma))$
Dense W_μ
Dense W_σ
h(t)
LSTM

Figure 3 (c) Source [14]

- h(t) which was previously calculated from LSTM passes through two separate dense layers w_ μ and w_ σ dense layers for the calculation of mean and variance respectively.
- After calculating the parameter's mean(μ) and variance(σ), a Gaussian likelihood function will be developed. Then the predictions are sampled from this equation to be compared with actual values [14].
- Based on the error, weights will be updated through backpropagation for dense layers and the LSTM network. This process continues until it reaches the end of training data.

  Note: These Dense layer weights are shared across all time steps in the training process.

3.1.1.2 Steps involved during the inference process:

- The inference process is almost the same as the training process but there will not be any dense layers training to estimate parameters (μ, σ) to build the likelihood function. The trained dense layer weights during training process will be utilised.

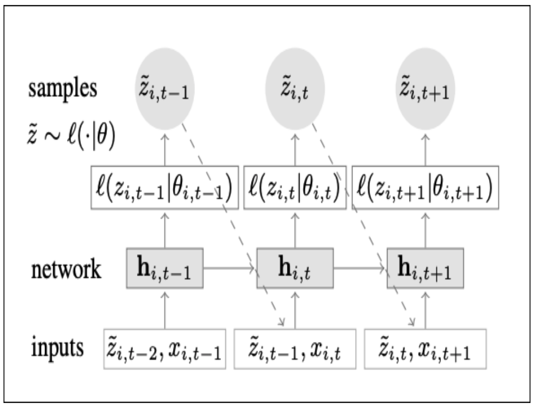

samples
$\tilde{z} \sim \ell(\cdot|\theta)$
$\tilde{z}_{i,t-1}$
$\tilde{z}_{i,t}$
$\tilde{z}_{i,t+1}$
$\ell(z_{i,t-1}|\theta_{i,t-1})$
$\ell(z_{i,t}|\theta_{i,t})$
$\ell(z_{i,t+1}|\theta_{i,t+1})$
network
$\mathbf{h}_{i,t-1}$
$\mathbf{h}_{i,t}$
$\mathbf{h}_{i,t+1}$
inputs
$\tilde{z}_{i,t-2}, x_{i,t-1}$
$\tilde{z}_{i,t-1}, x_{i,t}$
$\tilde{z}_{i,t}, x_{i,t+1}$

Figure 3(d) Source:[6]

- The model used the weights trained during the training process and utilise them to generate function and sample prediction values.
- The predicted value z^_ (i, t-1) is then passed to the next time step (since it is auto-regressive) as shown in Figure 3(d), this process continues until the end of the test data. The same weights of the dense layer will be used throughout the inference process.[14]

3.1.2 N-Beats Architecture:

This Deep learning approach was first introduced by Element AI (Montreal-based start-up) which beats the M4 competition Dataset's [15] winning solution accuracy and becomes the first pure deep learning model to do so. N-Beats tries to incorporate the mechanisms of statistical models using double residual stacks of fully connected layers [13].

The architecture mainly comprises 3 main sections

- Original Model (yellow colour): Collection of Stacks
- Stacks (Orange colour): Collection of blocks
- Blocks (Blue blocks): Processing Unit (forecasts and back casts)

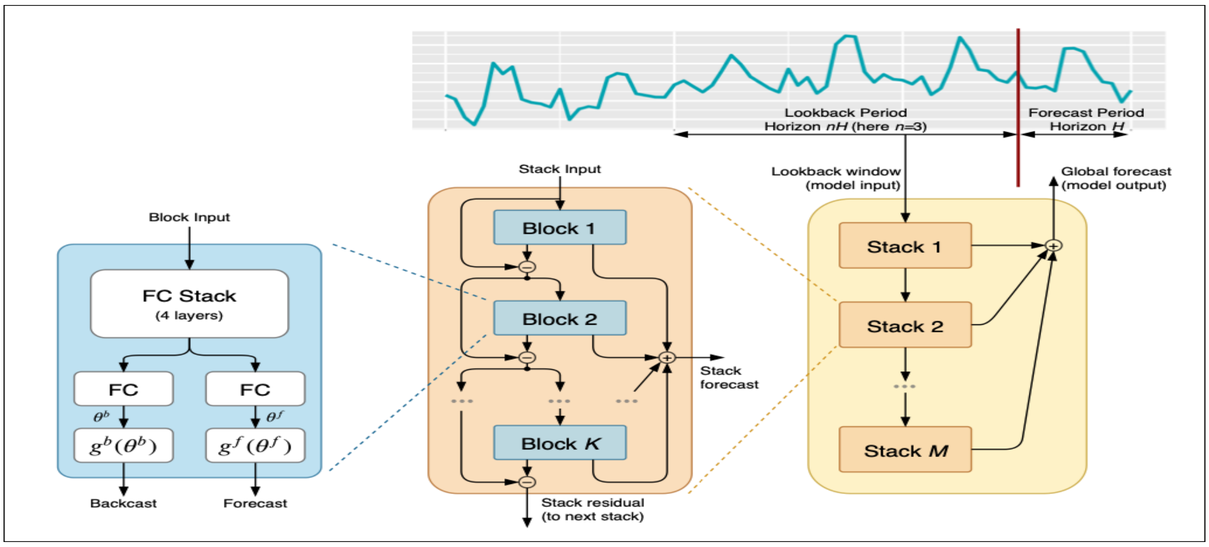


Figure 3(e) Source: [13]

In the original model (yellow colour box as shown in Figure 3(e)) the first stack receives a raw input time series of lookback period length (which is usually the multiple of prediction length for a given time series as shown in Figure 3 (e). Since each stack has a collection of blocks it passes through the first block inside the stack. The input inside the block passes through multiple layers of neural networks (as shown to left most blocks in Figure 3 (e)) which are directed to two different dense layers to estimate the Ө^b and Ө ^f expansion coefficients, By utilising basis layer transformations coefficients were projected to new space to generate back cast and forecast signals. The backcast signal tries to mimic the time series input and the forecast signal tries to predict the prediction length outcome based on input (as shown in figure 3(f)).

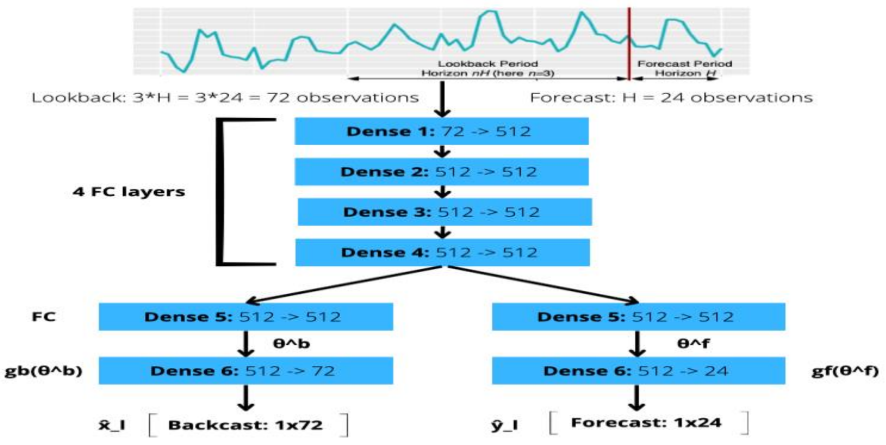


Figure 3(f) Source [16]

After processing the input, it generates 2 things back cast signals and forecast signals. The backcast signals will be subtracted from the actual block input to pass it to the next block inside the stack as shown in figure 3(g).

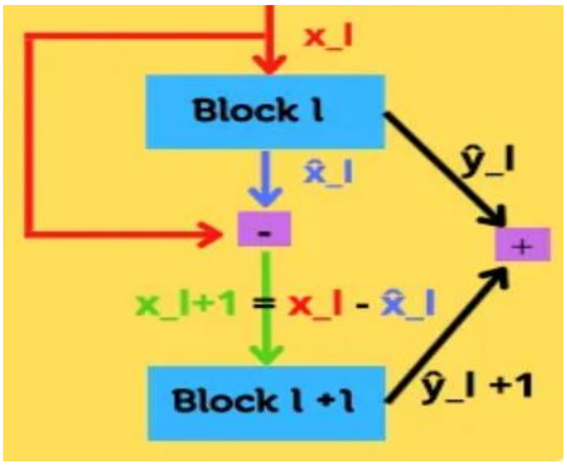


Figure 3(g) Source [16]

The forecasts generated from all the blocks will be concatenated to generate the final output of the stack. Similarly at the stack level as shown in the figure 3(e), it sends the residuals left in the last block of the previous stack as input to the next stack, this process continues for all stacks until the end of training data.

In this project, this architecture was extended to solve multiple time series problems by using General N-Beats architecture in which it has fixed 30 stacks each having one block. For multiple time series analysis, each time series passes through this architecture as sequences to generate forecasts by the same process mentioned above. In this architecture, the weights inside the block (blue boxes in fig 3(f)) are shared across all-time series but the parameters for block mechanisms like calculating residuals and summarising forecasts would be independent for each time series. Sharing of weights inside the block for all-time series helps in inter-transfer learning.

3.2. Multivariate time series Analysis:

This analysis mainly involves in utilising multiple covariates along with actual features in time series that behave as a catalyst to improve the overall model's performance. The goal was set to compare traditional methods like ARIMAX with LSTM networks to provide recommendations based on analysis and experiments. The overview of datasets that are utilised for experiments is as follows:

1. Stock Exchange Data [2]:

The dataset can be accessed here [2]. The data was all collected from Yahoo Finance, which had several decades of data available for most exchanges.

2. Appliances energy prediction Dataset with external and internal factors [4]

The dataset can be accessed here [4]. It contains temporal data on energy consumption for appliances with 26 external and internal factors. The house temperature and humidity conditions were monitored with a ZigBee wireless sensor network the energy data were logged every 10 minutes with m-bus energy meters.

Many real-time multivariate time series problems are solved by using LSTMs and ARIMAX models. So one of the goals of the project was set to implement, analyse and develop comparative analysis to provide optimal recommendations based on experiments and the nature of data.

3.2.1 LSTM:

LSTM is an enhanced version of RNN (recurrent neural networks) which was first introduced in 1997. RNNs cannot incorporate long-term memory, so RNNs are forgetful. LSTMs can retain long-term memory and utilize it to learn patterns in long sequences. In LSTM for a given time step it can process the long-term memory from remote states, short-term memory from the preceding cell, and current input data.

The longer durations help to solve the problem of vanishing gradient descent, this occurs when the gradient becomes too small to backpropagate and further improve the weights usually in these scenarios gradient calculations are less than 1[12]. Another problem would be vanishing gradients, which involve calculations greater than 1 this makes RNN very unstable and learns only from short-term memory but LSTM tries to keep the gradient steep enough that its searching process does not end in a dead-end.

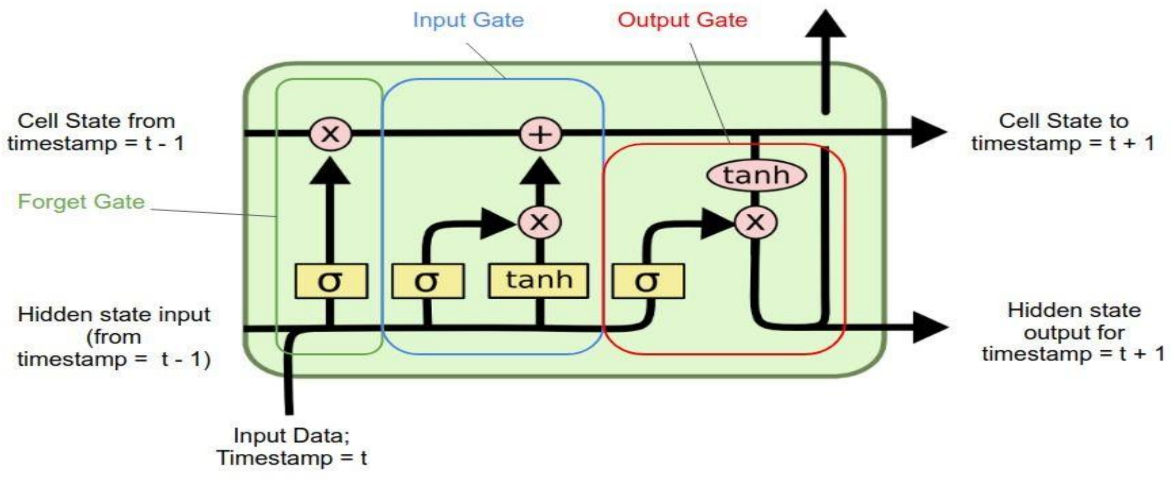

Input Gate
Output Gate
Cell State from timestamp = t - 1
Forget Gate
Cell State to timestamp = t + 1
tanh
σ
σ
tanh
σ
Hidden state input (from timestamp = t - 1)
Hidden state output for timestamp = t + 1
Input Data; Timestamp = t

Fig 3(h) Source https://www.turing.com/kb/comprehensive-guide-to-lstm-rnn

LSTM cells are said to be gated as shown in Fig 3(h), and information is selectively flown through the gates which can capture how much information is grasped and how much information is retained. Three main gates in LSTM are the input gate which handles the inflow of input data, forget gate which erases the unnecessary information in long-term memory and the output gate which passes information to the next layer. Eventually, concerning time, LSTM tries to learn which pieces of information are effective, forgets unnecessary information and utilizes the gates accordingly [12][8].

3.2.2 ARIMAX:

ARIMA (autoregressive integrated moving average model) is one of the most popular traditional time series models for linear and non-linear time-series predictions. Which mainly comprises 3 major mechanisms.

Auto regression (AR): This component tries to regress over its own ‘p’ previous values to make future predictions.

Integrated (I): one key aspect to consider before applying ARIMA is to make data stationary this is done by order of differencing(d) for time series. example first order differencing d=1.

Moving Average (MA): This component tries to regress over its past ‘q’ error(residuals) terms to predict future predictions.

These 3 components (p, d, q) are given to the model based on the nature of the data. In this project, ARIMA was extended to ARIMAX to perform multivariate time series analysis by incorporating exogenous features as multivariate features that would help to improve the predictions.

To understand the essence of these two methods and compare their respective performances, these models were implemented on stock exchange data and appliances energy consumption.

3.3 The essence of Deep AR and N-Beats for multiple time series forecasting.

While dealing with multiple time series problems depending on the nature of the domain there can hundreds of different time series that need to be modelled which is very challenging by traditional methods like ARIMA, SARIMA etc.

One possible approach using traditional methods is to run a loop for each time series and model the time series separately which does not provide any optimal solution because all pre-processing steps and analysis specific to individual time series characteristics are missed.

If the modelling was done independently for each time series with increased volumes of data it would take a lot of time to process the models. To overcome this modelling issue deep learning approaches like deep AR and N-beats can be utilised. So based on this idea for multiple time series datasets deep AR and N-Beats were implemented.

Both Deep AR and N-Beats have their own merits and demerits concerning the problems. Some of the key aspects for utilisation of these two approaches for time series problems are as follows in Table 3(a):

Deep-AR	N-Beats
It can provide probabilistic forecasts along with point-in-time forecasts	It can only provide point-in-time forecasting.
It might require more computational resources and takes more time to train.	It requires fewer computational resources and takes less time to train
This model takes extra care in handling	This model does not use any seasonal

seasonal patterns by using seasonal components.	components.
It only works for equally spaced time intervals	It works both equal and unevenly spaced time intervals.

Table 3(a)

3.4 Detailed breakdown of datasets and approaches implemented for analysis

S.no	Time series analysis	Dataset	Approaches Implemented
1	Multiple	Real-world electricity data of 370 households	Amazon's Deep AR , N-Beats
2	Multiple	Monthly Sales data of 810 different products	Amazon's Deep AR , N-Beats
3	Multivariate	Merged Dataset from appliance energy consumption and Belgium weather station	LSTM, ARIMAX
4	Multivariate	Stock Exchange Data	LSTM, ARIMAX

In this project, Comparative analysis along with respective implementations are done based on datasets mentioned in the above table. More Details of implementations are covered in Section 4.

3.4 Analysis and Evaluation

- Prediction accuracy would be calculated by using general evaluation calculations like mean absolute error, normalised root means square error and RMSE for actuals and outcomes.

- Implementation and comparative analysis of Amazon’s Deep AR and N-Beats for multiple time series would be presented in this project. (Covered in Implementation).
- Reasoning and analysis would be developed on why, how, and when deep learning approaches can be effective.

Project Requirements: Intel I5 processor or more, Notebook: Google collab pro Subscription, GPU: K800, P100, T4, RAM: 16-20 GB, Microsoft Power BI, Microsoft excel.

4. Implementation:

4.1 Multiple time series Analysis Implementations:

Implementations were done on the following use cases:

1. Estimation of electricity load consumption for 369 Households (electricity data)
2. Monthly sales prediction for 810 products. (Product sales data)

4.1.1 Deep AR and N-Beats Results and Analysis:

Model parameters and data statistics are described in below table 4(a).

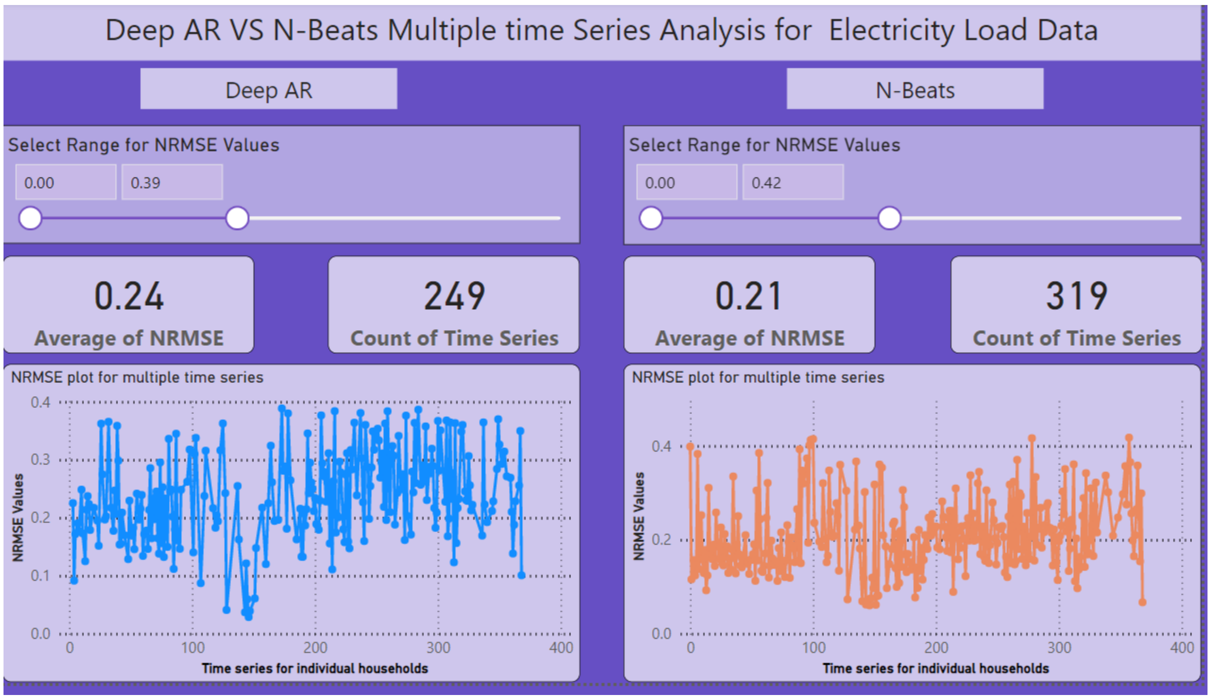
Sno	**Model parameters**	**Deep AR**	**N-Beats**
1	Context length	672	960
2	Prediction length	96	96
3	epochs	50	50
4	Learning Rate	0.01	1e-3
5	Training size	138911	138911
7	Testing Size	1345	1345
10	Training Time	4 hours	1 hour

Table 4(a) Dataset statistics and parameters for electricity data

Experiments on these datasets were performed after data pre-processing steps like normalisation and data preparation steps like creating template structures of gluon TS Implementations.

Deep AR vs N-Beats on electricity Data:

N-beats outperformed Deep AR with 319-time series predictions less than 0.4 (NRMSE score) whereas on other hand Deep AR got 249-time series with less than 0.4 NRMSE score as shown in image 4(a). This is because electricity data involves non-linear trends and sudden fluctuations due to multiple external factors which needs a highly sophisticated non-linear modelling architecture.


Deep AR VS N-Beats Multiple time Series Analysis for Electricity Load Data
Deep AR
N-Beats
Select Range for NRMSE Values
0.00
0.39
Select Range for NRMSE Values
0.00
0.42
0.24
Average of NRMSE
249
Count of Time Series
0.21
Average of NRMSE
319
Count of Time Series
NRMSE plot for multiple time series
NRMSE Values
Time series for individual households
NRMSE plot for multiple time series
NRMSE Values
Time series for individual households

Power BI dashboards for electricity load data analysis Image 4 (a)

Although Deep AR is sophisticated and much more advanced than N-Beats, Deep AR's performance may reduce on these types of data sets due to its inclination towards autoregressive nature to some extent that

involves predicting outcomes based on the lagged values with optional co-variates which makes it less effective to observe non-linear trends whereas n-beats on other hand have a better strategy in handling these scenarios by using double residual stacks.

Another important factor is noise involved in electricity data usually N-beats have better capability to understand noisy data because of their architecture (as shown in figure 3 (e)) while processing the data, as it uses residual-based architecture which can handle noise. In Deep, AR noise was handled implicitly by utilising a probabilistic model which makes it less effective in comparison with N-Beats.

S.no	Approach	Electricity Data (# of time series)		Weekly Sales (# of time series)	
		NRMSE<0.4	NRMSE>0.4	NRMSE<0.5	NRMSE>0.5
1	Deep-AR	249	120	322	488
2	N-Beats	319	66	249	561

Tabular representation of implementation results Table 4 (b)

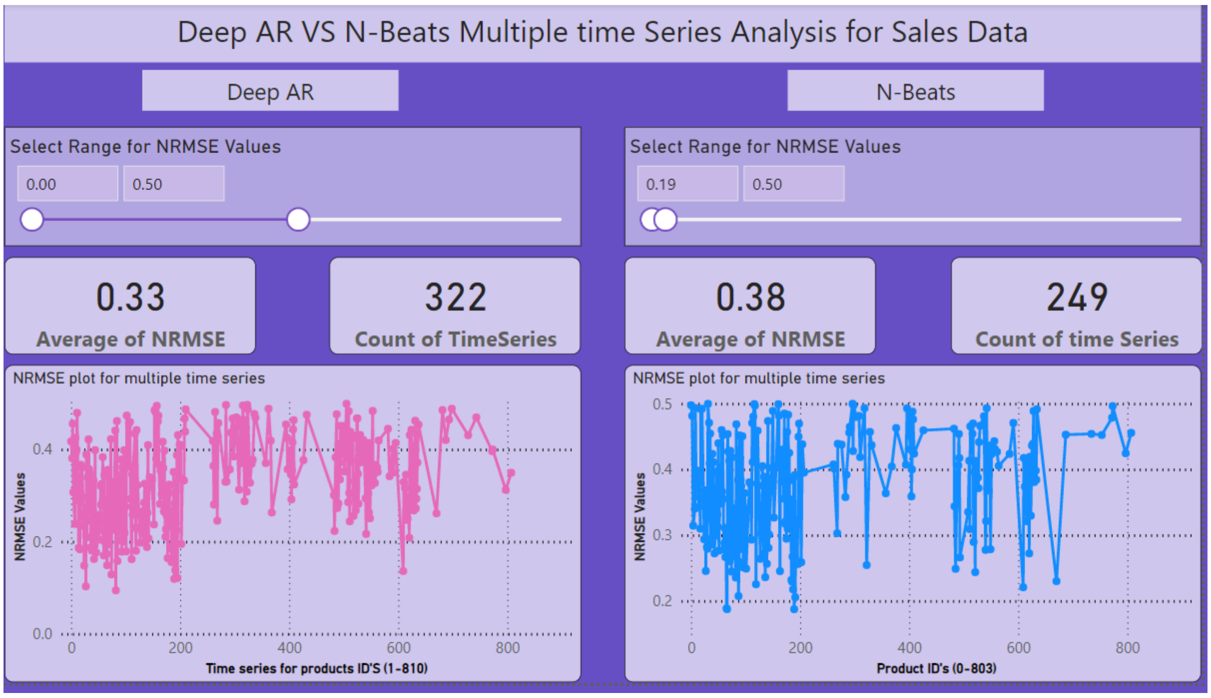
Deep AR Vs N-Beats on Sales Data:

Sno	**Model parameters**	**Deep AR**	**N-Beats**
1	Context length	18	7
2	Prediction length	7	7
3	epochs	30	30
4	Learning Rate	0.001	0.01

Model parameters for sales data Analysis Table 4 (c)

Both models Deep AR and N-Beats obtained poor results on sales data with approximately 40 per cent of the total time series with a descent RMSE scores as shown in Table 4(b). Sales data of 810 products have a wide variety of characteristics in which it does not have regular and repeating patterns for all the products this makes it very challenging for the model to go through internal learnings with individual time series patterns for N-Beats.

This dataset has 800-time series with 52 weeks of data which might be not enough for modelling time series with complicated patterns and it might also overfit the training data for a few product time series. This experiment gives evidence that Deep AR and N-Beats model performances can be reduced with an increase in the number of time series and fewer data values. Some products might also have very high fluctuation with non-repetitive seasonal patterns, for example, in real-world events like inflation, recession sales of some luxury products may fluctuate unevenly throughout unequal intervals of time. As shown in the figure out of both models deep AR performed significantly better. Some of the reasons for these results are there can be a greater number of products that have linear patterns and strong seasonality for a greater number of products, Deep AR explicitly can take special care for seasonality due to which its performance was slightly better than N-Beats.


Deep AR VS N-Beats Multiple time Series Analysis for Sales Data
Deep AR
N-Beats
Select Range for NRMSE Values
0.00
0.50
Select Range for NRMSE Values
0.19
0.50
0.33
Average of NRMSE
322
Count of TimeSeries
0.38
Average of NRMSE
249
Count of time Series
NRMSE plot for multiple time series
NRMSE Values
Time series for products ID'S (1-810)
NRMSE plot for multiple time series
NRMSE Values
Product ID's (0-803)

Power BI Dashboards for sales data analysis

4.2 Multiple time series analysis recommendations:

Both Deep AR and N-Beats are powerful modelling approaches for multiple time series but their usage needs to be based on the specifics of the data. To understand which modelling approach needs to be followed will be based on their respective strengths and aligning it with data characteristics. Based on the experiments strengths of Deep AR includes handling strong temporal dependencies since it is auto-regressive, extra care for handling seasonality involved in data, and a greater number of parameters while training dense layers may help to learn complicated patterns.

For N-beats has strong capabilities of handling non-linear patterns and noise in the data, faster model training and is less expensive computationally since weights inside blocks are shared across all other time series during training, N-Beats has the capability of inter-transfer learning.

Although both approaches have their own merits and demerits when dealing with multiple time series forecasting problems it's very uncertain to understand the nature of all-time series patterns and understand the characteristics of data so it is ideal to implement multiple techniques and compare the results for further improvements. Both approaches have a rich set of hyperparameters available by tuning models with proper documentation for every experiment, higher performance can be obtained. With the increase in the number of individual time series and less data for both approaches modelling gets complicated and may also lead to poor performances respectively as evident from experiments on sales data analysis in 4.2.

4.3 Multivariate time series Implementations:

Implementations were done on the following use cases:

1. Appliances electricity consumption (Appliances data)
2. Closing Stock prediction for NYE stocks (Stock exchange data)

Datasets Statistics:

Sno	**Data Parameters**	**Appliances Data**	**Stock exchange Data**
1	Dataset Size	140256	6746
2	Time granularity	15 mins	3 Days
3	Number of features	39	5
4	Data pre processing	Partial	Partial
5	Target Feature	Appliance consumption	Closing price of NYC Stock
6	Time duration of Data	5 months	57 years

Table 4(d) Dataset statistics for multivariate analysis

Model statistics of LSTM and ARIMAX models:

S.no	LSTM Parameters			ARIMAX Parameters		
	Parameters	Appliances Data	Stock Exchange Data	Parameters	Appliances Data	Stock Exchange Data
1	epochs	35	50	AR (p)	2	2

2	Optimiser	Adam	Adam	I(d)	0	1
3	# LSTM nodes	25	30-45	MA(q)	2	1
4	Layers	3	3	Train size	18748	6070
5	Train Seq-Size	18,947	6057	Test size	987	676
6	Test Seq-Size	788	674	# features utilised	6	4
7	# features	12	5	Training Time	12	8 mins
8	Training Time	25 mins	13 mins			
9	Sequence Length	40	15	-		

Table 4 (e) Model Statistics for multivariate Analysis

4.3.1 LSTM and ARIMAX Results and Analysis:

Sno.	Approach	Appliances Consumption		NYC Stock Prediction	
		RMSE	Mean Absolute error	RMSE	Mean Absolute error
1	ARIMAX	109	56	34	20
2	LSTM	80	37	310	202

Table 4 (f) LSTM and ARIMAX results

ARIMAX vs LSTM on appliances consumption data:

As shown in Table 4 (f) LSTM model performed better than ARIMAX with RMSE Score of 80 and ARIMAX has a score of 109. This is due to the nature of appliance consumption being very nonlinear with a wide variety of covariates like temperatures, humidity etc. Even though ARIMA can handle non-linearity in data to some extent but it cannot handle complicated non-linear patterns from multiple features. Given the size of the dataset as shown in Table 4 (d) , ARIMA cannot work well with large volumes of data with complicated nonlinear patterns as compared to Neural networks due to which the LSTM model got better results in this use case.

ARIMAX vs LSTM on stock exchange data:

As shown in the results table 4 (f) ARIMAX model performed better than LSTM model with an RMSE Score of 34 whereas for LSTM RMSE value is 310. The Dataset utilised has simpler patterns which can be modelled easily with general methods such as trend, seasonality etc this helped ARIMAX to outperform LSTMs, on contrast in these scenarios LSTM's can make things complicated with its architecture of neural networks and will lead to lower performance. Usually, ARIMA models perform better on consistent patterns and on datasets having strong stationary characteristics whereas LSTMs may face some challenges with short-term fluctuations and performance might be reduced.

4.4 Multivariate time series recommendations:

Based on the above experiments and approaches implemented in this project, modelling approaches that need to be utilised for a time series problem can be understood by analysing the strengths of approaches.

Based on LSTM implementation on electricity data and stock exchange data. For a given time series, the LSTM approach can be considered if it involves in following:

- Complicated non-linear data.
- complicated data interrelationships, temporal dependencies,
- consideration for long-term dependencies
- Increased number of features.

For a given time series, the ARIMAX approach can be considered if it involves in following:

- Data has simplistic interpretable patterns with stationarity,
- Smaller datasets and smaller forecast ranges,
- Interpretability is high priority than performance,
- Less l features for multivariate analysis.

Considering above strengths of LSTM's and ARIMAX modelling approaches need to be chosen accordingly.

4.3. Major drawbacks of deep learning based on experiments:

- Deep learning approaches have many advantages like solving and understanding complicated non-linear patterns, although these models are more sophisticated there would always be some sort of uncertainty while relying on the performance of these models and understanding the results because they are not interpretable.

- These approaches may require large volumes of data with high computational requirements which eventually increases the time to train the models (as evident from sales analysis in 4.2)
- Optimising hyperparameters can become a challenging task because most of the models are not interpretable which makes the selection of parameter combinations difficult in case of complicated datasets. (Based on implementations on electricity data).

5.Conclusion:

The main goal of the project was to implement and compare deep learning approaches to provide recommendations for multiple and multivariate time series analysis. This project implemented its goals by developing multiple deep learning approaches for multiple time series analysis, by implementing Deep AR and N-Beats for a couple of real-world use cases like monthly sales prediction and electricity load forecasting. The comparative analysis of these 2 techniques helped to generate recommendations based on data characteristics and implementations. Further, for multivariate time series forecasting, the LSTM model was compared with traditional methods like ARIMAX using datasets with varying characteristics. A thorough analysis of these 2 techniques based on performance and data characteristics for each approach was developed, the project provided optimal recommendations for multivariate time series forecasting for these two approaches as well. Overall, this project not only successfully implemented modern time series models but also provided practical recommendations for their effective application in different scenarios.

The findings, analysis and recommendations developed based on multiple implementations in this project can help to better understand and give proper direction on selecting optimal deep learning approaches (specified in this project) while dealing with time series problems.

6. Future work:

In this project, N-Beats generic architecture was implemented to understand multiple time series forecasting but with N-Beats interpretable model that uses a limited number of stacks with special restrictions during the training process it helps to interpret which parts of time series data are responsible for predictions. This interpretability feature helps to understand the modelling process and results more clearly. Further Temporal Fusion Transformers which works like deep AR can also be implemented and analysed to come up with a comparison report to provide recommendations on experiments, data and domain characteristics.

7. References:

[1] Store Item Demand Forecasting Challenge. [Dataset]. Available: https://www.kaggle.com/competitions/demand-forecasting-kernels-only/data. [Accessed: February 02, 2023]

[2] Stock Exchange. Data Yahoo Finance. [Dataset]. Available: https://www.kaggle.com/datasets/mattiuzc/stock-exchange-data?select=indexProcessed.csv. [Accessed: February 02, 2023]

[3] Electricity Load Diagrams. [Dataset]. Available: http://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014 [Accessed: February 02, 2023]

[4] Appliances energy prediction [Dataset]. Available: https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction [Accessed: February 02, 2023]

[5] Hourly energy consumption [Dataset]. Available: https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption [Accessed: February 02, 2023]

[6] D. Salinas, T. Flunkert, and J. Gasthaus, "DeepAR: probabilistic forecasting with autoregressive recurrent networks," arXiv preprint arXiv:1704.04110, Apr. 2017. Available: https://arxiv.org/abs/1704.04110. [Accessed: February 02, 2023].

[7] B. Lim and S. Zohren, "Time-series forecasting with deep learning: a survey," Royal Society Open Science, vol. 8, no. 2, p. 202092, Feb. 2021. Available: https://royalsocietypublishing.org/doi/full/10.1098/rsta.2020.0209. [Accessed: February 02, 2023].

[8] J. Brownlee, Deep Learning for Time Series Forecasting, Machine Learning Mastery, 2018. Available: https://books.google.ca/books?id=o5qnDwAAQBAJ&dq=deep+learning+for+time+series+forecasting&lr=&source=gbs_navlinks_s. [Accessed: February 02, 2023].

[9] R. Johnson and T. Zhang, "Effective use of word order for text categorization using CNN," arXiv preprint arXiv:1412.1058, Dec. 2014. Available: https://arxiv.org/abs/1412.1058. [Accessed: February 02, 2023].

[10] The Best Deep Learning Models for Time Series Forecasting. Nikos Kafritsas. Available: https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0. [Accessed: February 02, 2023]

[11] Deep Learning for Time Series Forecasting: Is It Worth It? Lina Faik. Available: https://medium.com/data-from-the-trenches/deep-learning-for-time-series-forecasting-is-it-worth-it-bfc95a1c9de4. [Accessed: February 02, 2023].

[12Understanding Recurrent Neural Network (RNN) and Long Short Term Memory(LSTM) Vijay Choubey. Available: https://medium.com/analytics-vidhya/undestanding-recurrent-neural-network-rnn-and-long-short-term-memory-lstm-30bc1221e80d [Accessed: February 02, 2023].

[13] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," International Conference on Learning Representations, Apr. 2020. Available: https://arxiv.org/pdf/1905.10437.pdf. [Accessed: February 02, 2023].

[14] Deep AR: Mastering Time-Series Forecasting with Deep Learning. Nikos Kafritsas. Available: https://towardsdatascience.com/the-best-deep-learning-models-for-time-series-forecasting-690767bc63f0. [Accessed: February 02, 2023]

[15] M4 Forecasting Competition Dataset [Dataset]. Available: https://www.kaggle.com/datasets/yogesh94/m4-forecasting-competition-dataset. [Accessed: February 02, 2023]

[16] N-BEATS: Time-Series Forecasting with Neural Basis Expansion. . Nikos Kafritsas . Available: https://towardsdatascience.com/n-beats-time-series-forecasting-with-neural-basis-expansion-af09ea39f538. [Accessed: February 02, 2023]

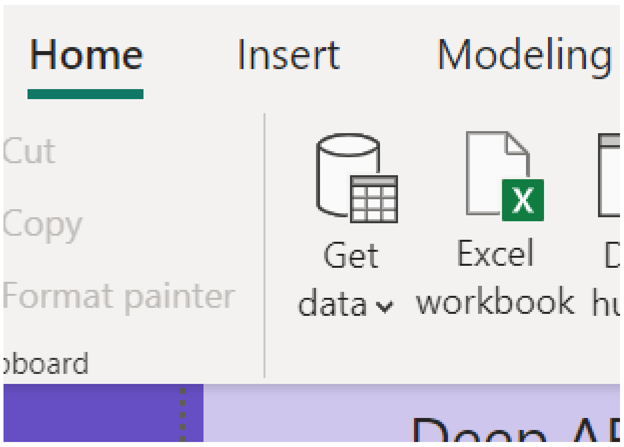
Appendix:

Section 1 Project Software Details.

- Microsoft Power BI Desktop (BI desktop version was provided with free of cost by Microsoft)

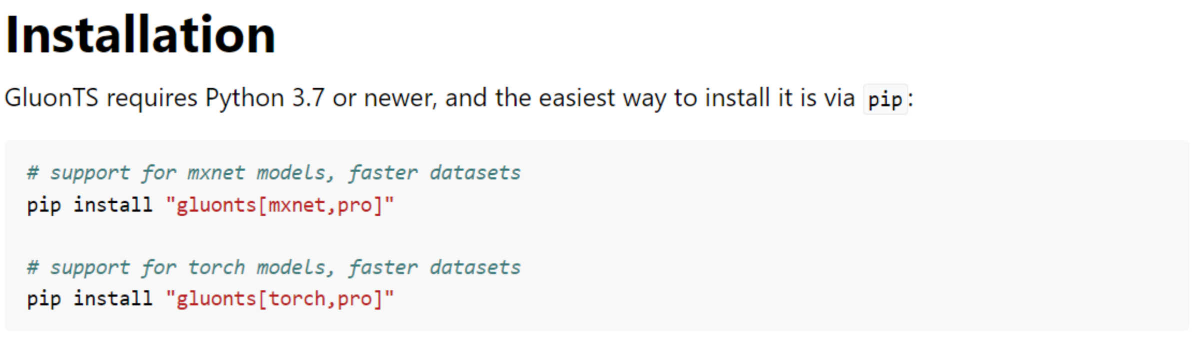
  Source: https://powerbi.microsoft.com/en-ca/

  Steps to import Data : In power BI-> get Data-> load



- All multiple time series models are implemented using Gluon TS Packages and dependencies.

  Source: https://ts.gluon.ai/stable/



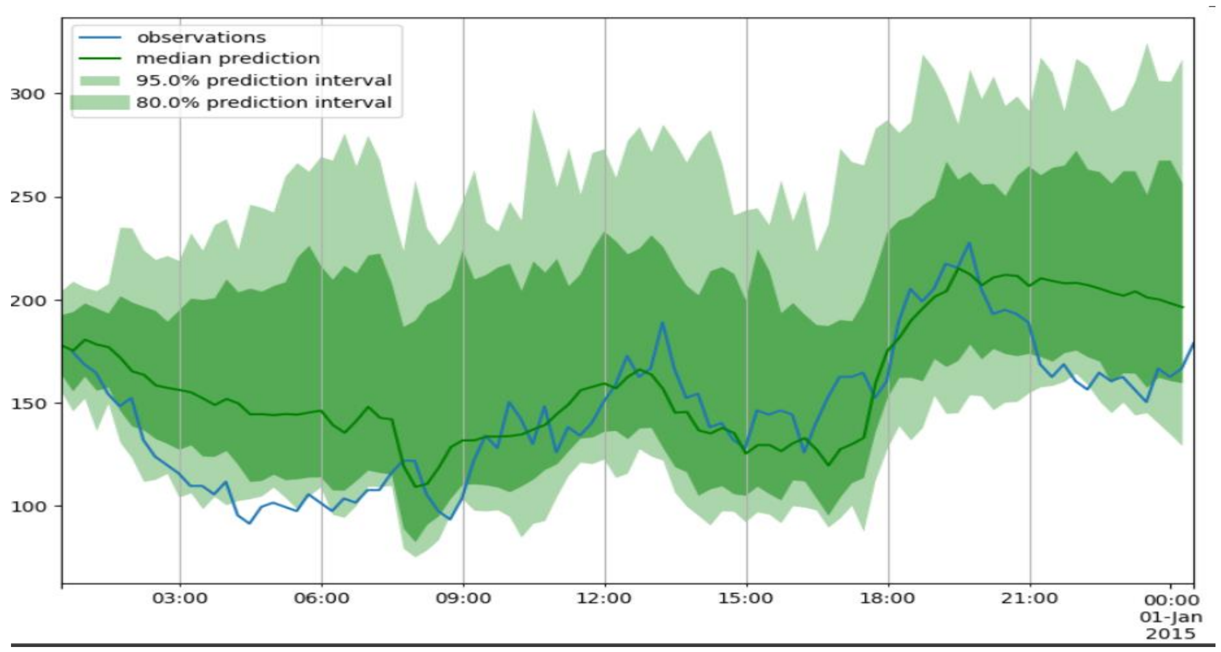
Installation

GluonTS requires Python 3.7 or newer, and the easiest way to install it is via `pip`:

```
# support for mxnet models, faster datasets
pip install "gluonts[mxnet,pro]"

# support for torch models, faster datasets
pip install "gluonts[torch,pro]"
```

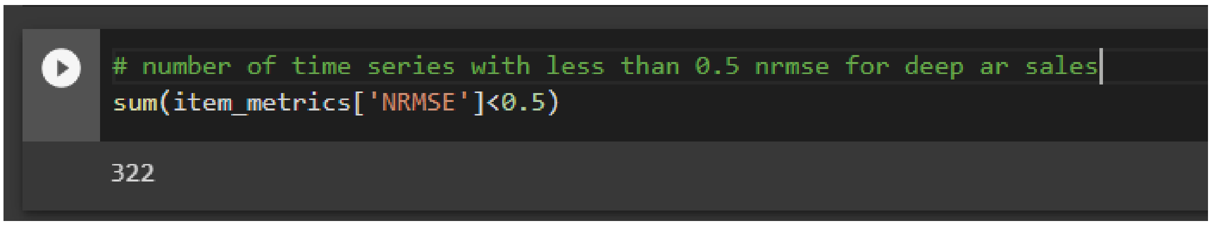
Section 2: Sample prediction Screenshots

Sample probabilistic output of electricity time series predictions with 80-90 % prediction intervals.



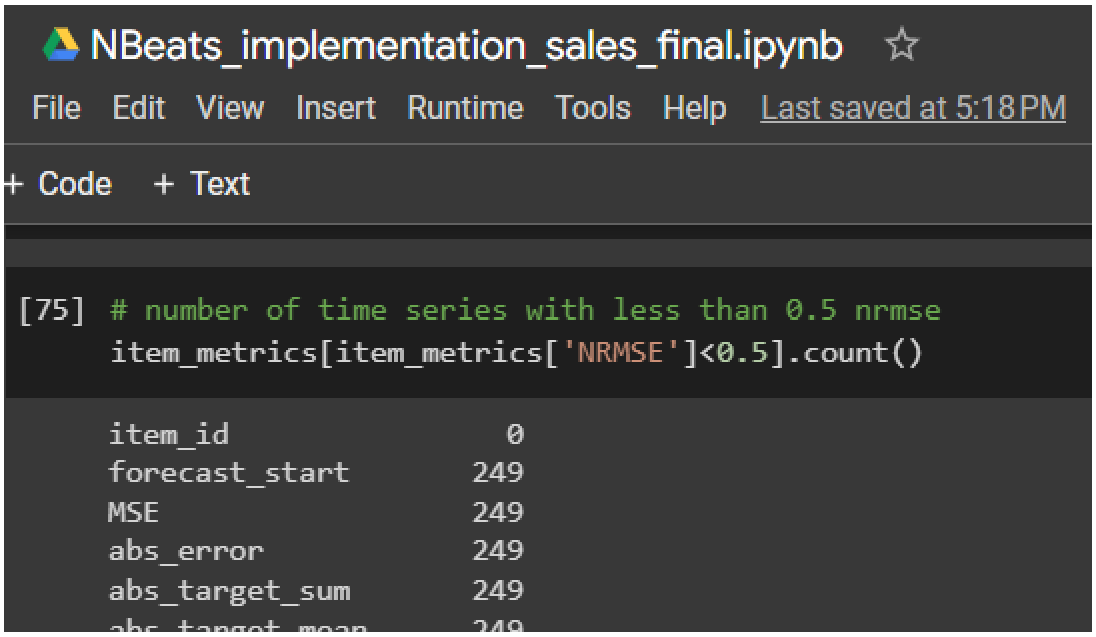
Section 3: Screenshots of outputs with respect to various datasets

1. Sales Data Analysis results (Multiple time series Analysis)



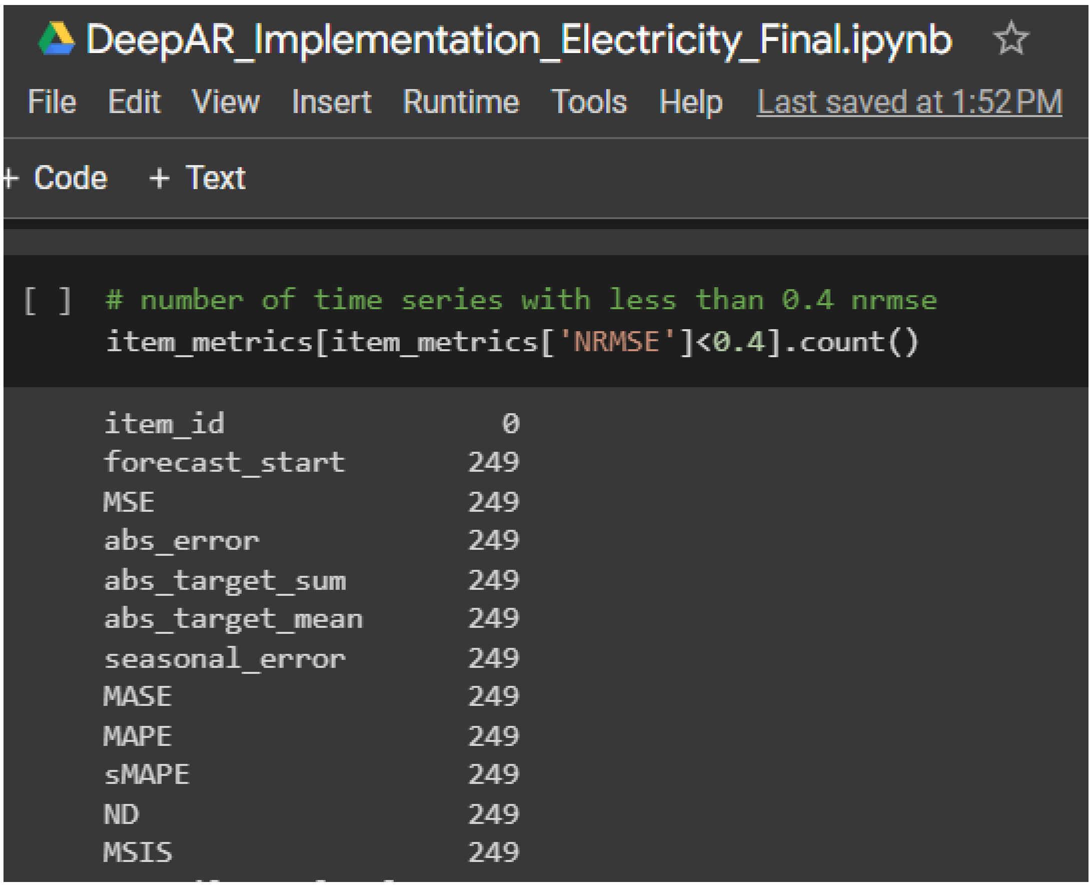
```
# number of time series with less than 0.5 nrmse for deep ar sales
sum(item_metrics['NRMSE']<0.5)
```

```
322
```

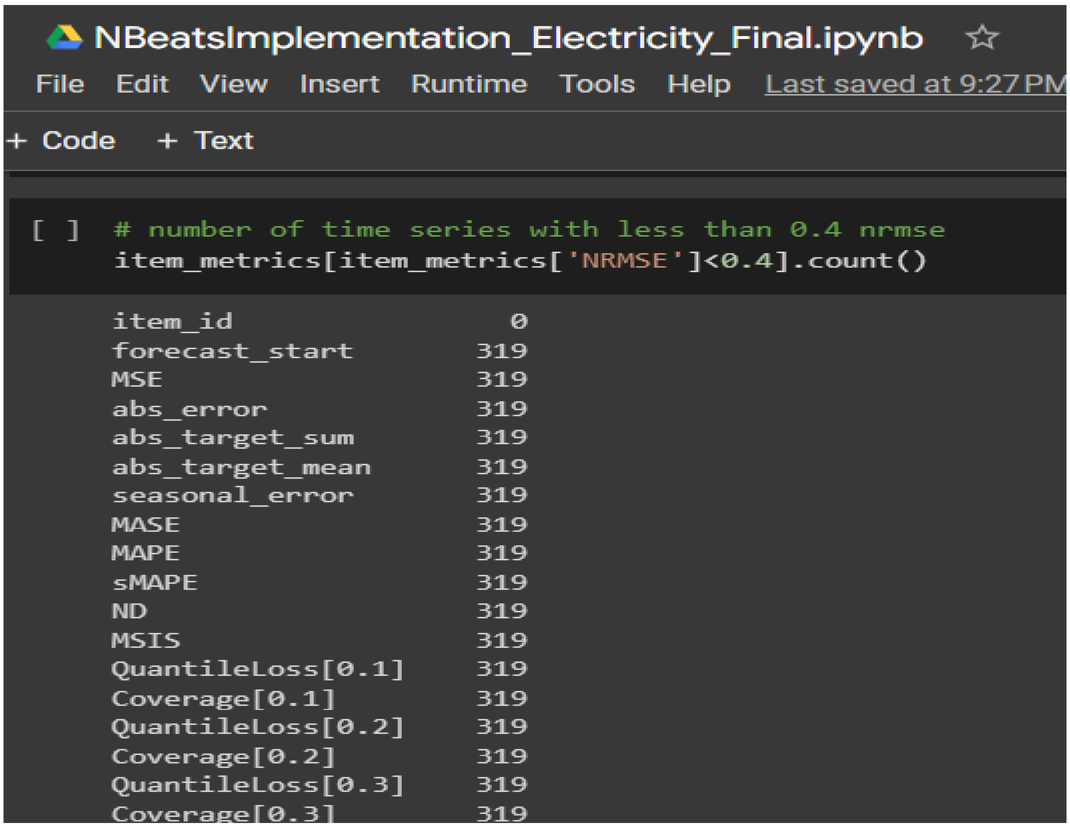

NBeats_implementation_sales_final.ipynb
File Edit View Insert Runtime Tools Help Last saved at 5:18 PM
+ Code + Text

```
[75] # number of time series with less than 0.5 nrmse
     item_metrics[item_metrics['NRMSE']<0.5].count()

     item_id              0
     forecast_start     249
     MSE                249
     abs_error          249
     abs_target_sum     249
```

2. Electricity Load dataset analysis (Multiple time series analysis)


DeepAR_Implementation_Electricity_Final.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:52 PM
+ Code + Text

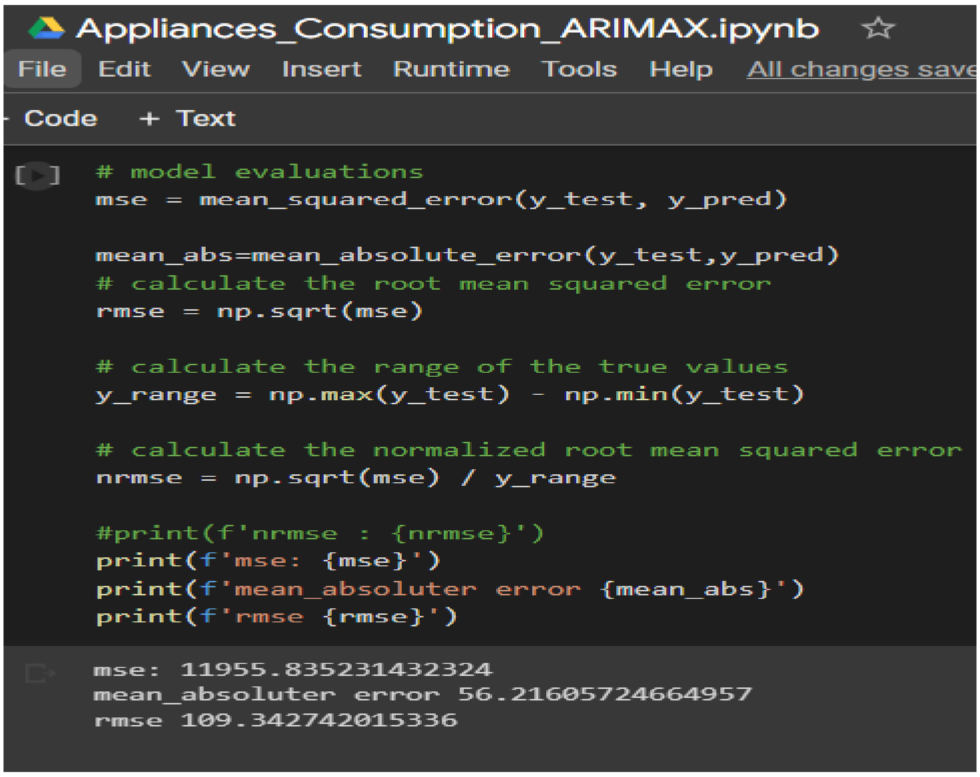
```
[ ] # number of time series with less than 0.4 nrmse
    item_metrics[item_metrics['NRMSE']<0.4].count()

    item_id              0
    forecast_start     249
    MSE                249
    abs_error          249
    abs_target_sum     249
    abs_target_mean    249
    seasonal_error     249
    MASE               249
    MAPE               249
    sMAPE              249
    ND                 249
    MSIS               249
```


NBeatsImplementation_Electricity_Final.ipynb
File Edit View Insert Runtime Tools Help Last saved at 9:27 PM
+ Code + Text

```
# number of time series with less than 0.4 nrmse
item_metrics[item_metrics['NRMSE']<0.4].count()
```

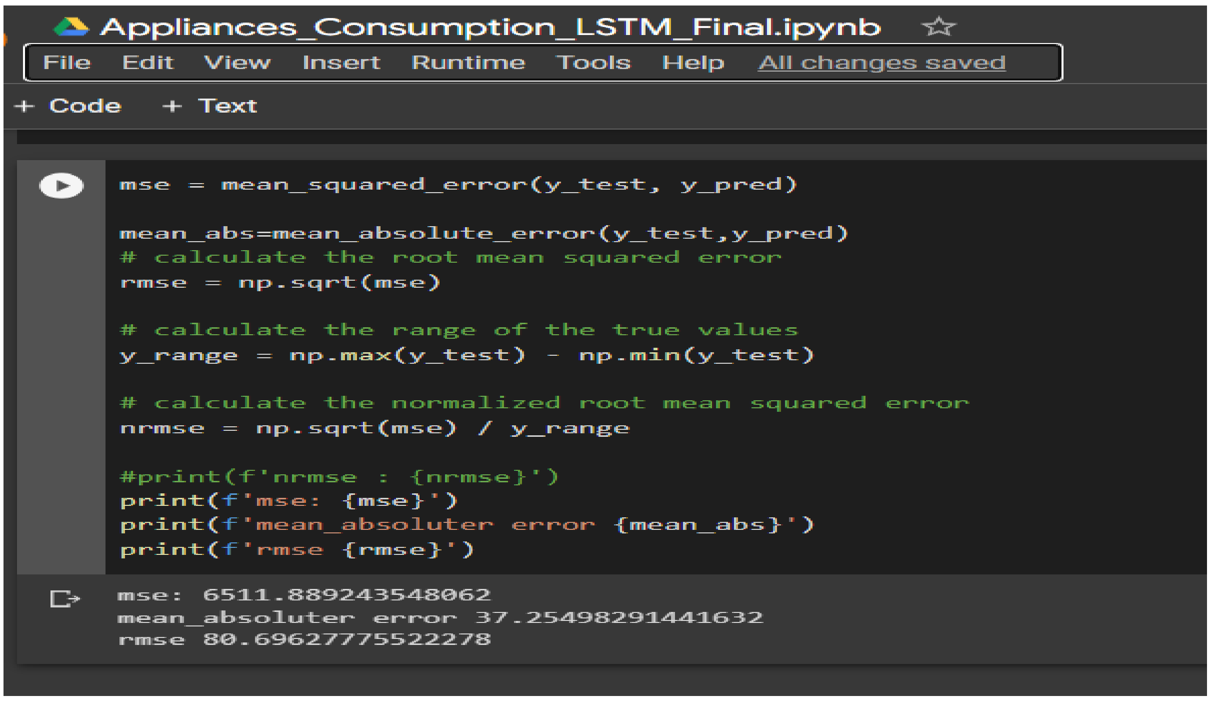
```
item_id                    0
forecast_start           319
MSE                      319
abs_error                319
abs_target_sum           319
abs_target_mean          319
seasonal_error           319
MASE                     319
MAPE                     319
sMAPE                    319
ND                       319
MSIS                     319
QuantileLoss[0.1]        319
Coverage[0.1]            319
QuantileLoss[0.2]        319
Coverage[0.2]            319
QuantileLoss[0.3]        319
Coverage[0.3]            319
```

3. Appliance Consumption Data Analysis Screenshots (Multivariate Analysis)


Appliances_Consumption_ARIMAX.ipynb
File Edit View Insert Runtime Tools Help All changes save
Code + Text

```
# model evaluations
mse = mean_squared_error(y_test, y_pred)

mean_abs=mean_absolute_error(y_test,y_pred)
# calculate the root mean squared error
rmse = np.sqrt(mse)

# calculate the range of the true values
y_range = np.max(y_test) - np.min(y_test)

# calculate the normalized root mean squared error
nrmse = np.sqrt(mse) / y_range

#print(f'nrmse : {nrmse}')
print(f'mse: {mse}')
print(f'mean_absoluter error {mean_abs}')
print(f'rmse {rmse}')
```

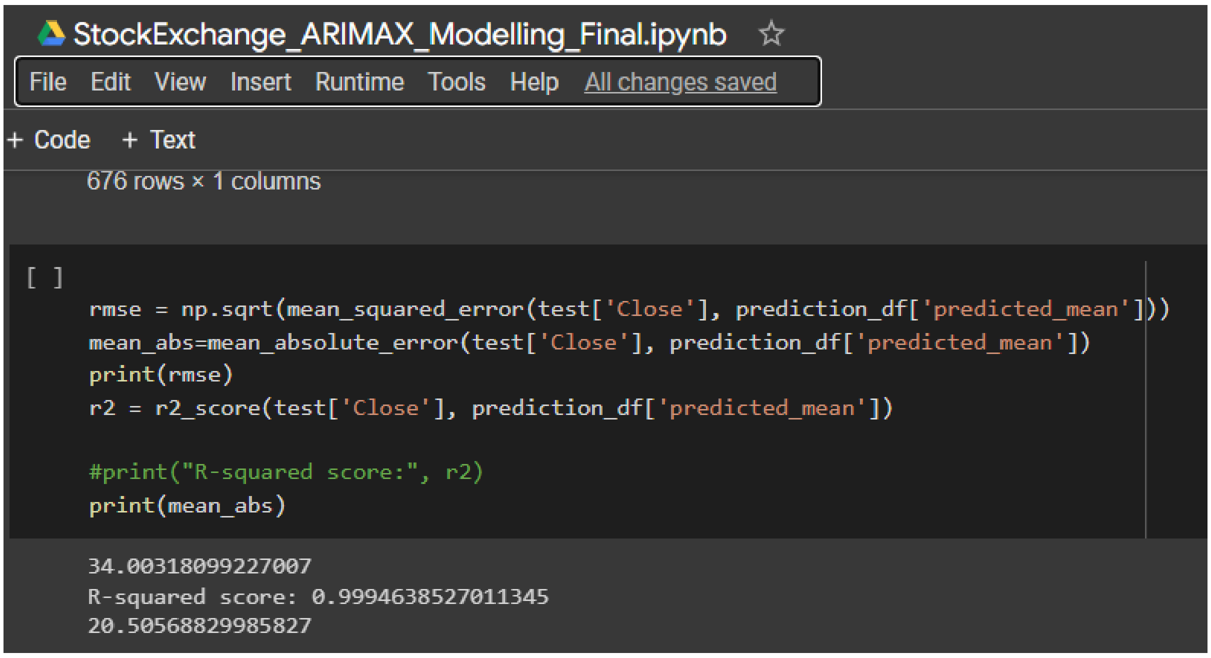
```
mse: 11955.835231432324
mean_absoluter error 56.21605724664957
rmse 109.342742015336
```


Appliances_Consumption_LSTM_Final.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
mse = mean_squared_error(y_test, y_pred)

mean_abs=mean_absolute_error(y_test,y_pred)
# calculate the root mean squared error
rmse = np.sqrt(mse)

# calculate the range of the true values
y_range = np.max(y_test) - np.min(y_test)

# calculate the normalized root mean squared error
nrmse = np.sqrt(mse) / y_range

#print(f'nrmse : {nrmse}')
print(f'mse: {mse}')
print(f'mean_absoluter error {mean_abs}')
print(f'rmse {rmse}')
```

```
mse: 6511.889243548062
mean_absoluter error 37.25498291441632
rmse 80.69627775522278
```

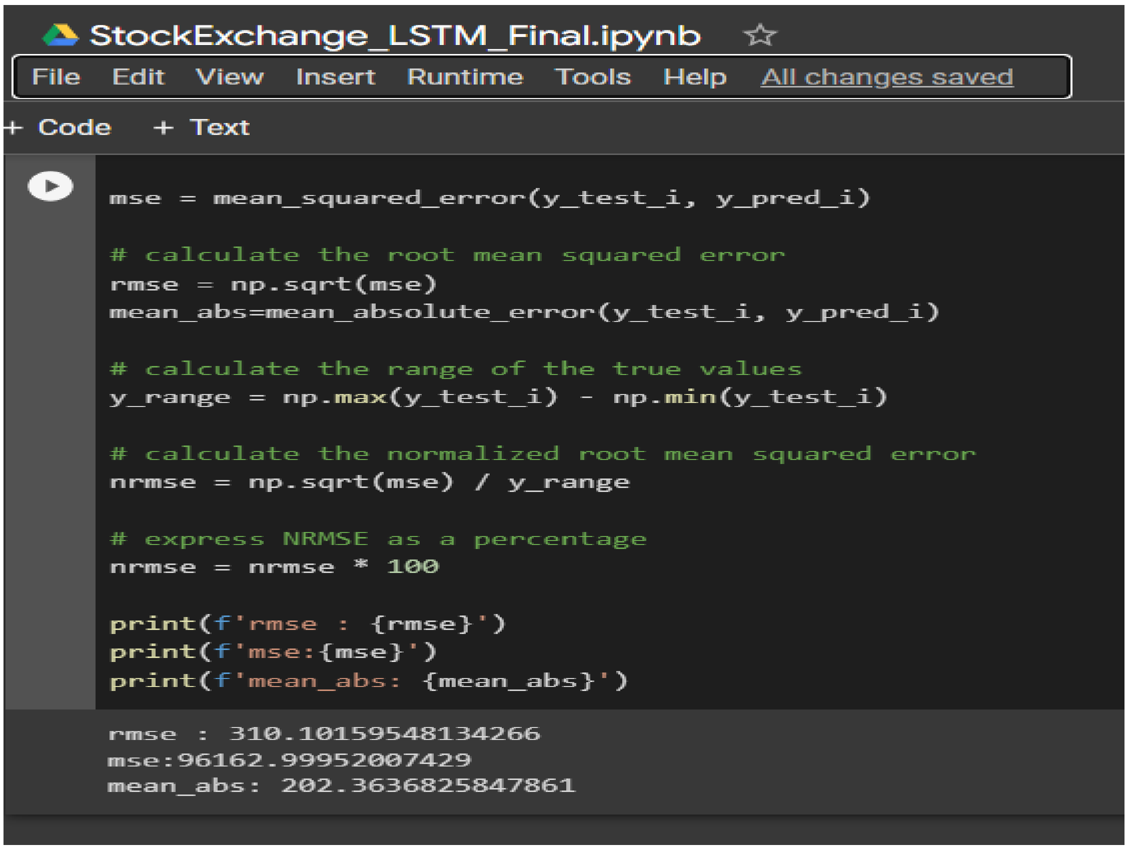
4. Stock exchange Data Analysis Screenshots (Multivariate Analysis)


StockExchange_ARIMAX_Modelling_Final.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

676 rows × 1 columns

```
[ ]
rmse = np.sqrt(mean_squared_error(test['Close'], prediction_df['predicted_mean']))
mean_abs=mean_absolute_error(test['Close'], prediction_df['predicted_mean'])
print(rmse)
r2 = r2_score(test['Close'], prediction_df['predicted_mean'])

#print("R-squared score:", r2)
print(mean_abs)
```

```
34.00318099227007
R-squared score: 0.9994638527011345
20.50568829985827
```


StockExchange_LSTM_Final.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```python
mse = mean_squared_error(y_test_i, y_pred_i)

# calculate the root mean squared error
rmse = np.sqrt(mse)
mean_abs=mean_absolute_error(y_test_i, y_pred_i)

# calculate the range of the true values
y_range = np.max(y_test_i) - np.min(y_test_i)

# calculate the normalized root mean squared error
nrmse = np.sqrt(mse) / y_range

# express NRMSE as a percentage
nrmse = nrmse * 100

print(f'rmse : {rmse}')
print(f'mse:{mse}')
print(f'mean_abs: {mean_abs}')
```

```
rmse : 310.10159548134266
mse:96162.99952007429
mean_abs: 202.3636825847861
```