

Load Forecasting Using Deep Learning

Project Supervisor:
Dr. Rajib Kumar Mandal
Associate Professor & HoD
Department of Electrical Engineering
National Institute of Technology, Patna

April 30, 2024



Table of Contents

- 1 Introduction
- 2 Time Series
- 3 Stationary Time Series
- 4 Stages in Time Series Forecasting
- 5 ARIMA Model
- 6 GRU Model
- 7 Comparison of both Models
- 8 References

Introduction to Project

- Welcome to our exploration into the dynamic world of time series analysis, where we delve into the predictive power of *GRU (Gated Recurrent Unit)* and *ARIMA (AutoRegressive Integrated Moving Average)* models.
- Time series data, characterized by its sequential nature and dependence on past observations, poses unique challenges and opportunities for prediction.
- **GRU**, a variant of **LSTM** excels in capturing long-term dependencies, while ARIMA is renowned for its effectiveness in forecasting.
- *Our project aims to unravel insights, compare performances, and navigate the complexities of time series forecasting using these models.*

Join us as we embark on this analytical journey!

Objectives of Project

① Primary Objectives

- To develop and compare the performance of **GRU** and **ARIMA** models for time series forecasting.

① Secondary Objectives

- Explore the dataset
- Preprocess the data
- Implement LSTM Model
- Implement ARIMA Model
- Evaluate Model Performance
- Perform Hyperparameter Tuning
- Interpret Results

Input, Output and File Structure of Project

- We've used a public dataset available on Kaggle.
- This Dataset contains only 2 columns, once column is date and the other column relates to the consumption in Mega Watts. This is on a Monthly Basis from 1985 till 2018.
- We've split the dataset into 4:1 train-test split ratio.
- We forecasted the consumption on the test set.
- We've made 2 Files separately for ARIMA and LSTM models.
- We thought the Final Output to be ensemble of both the Models but things came out that the error is best minimized only by using GRU alone.

What is Time Series? And why it is used?

- Time series data is data that is recorded over consistent intervals of time. **[7522190]**

But why do we even bother about it? What's so Special?

- Time series helps turn raw data into insights companies can use to improve performance and track historical outcomes.

How is it different from general ML Regression Problems?

Components of Time Series

- Time series decomposition is a technique used to break down a time series into its constituent components.
- Decomposing a time series helps in understanding its underlying patterns and extracting useful information for analysis and forecasting.
- **Components of Time Series:**
 - Trend: The increasing or decreasing value in the series.
 - Level: The average value in the series
 - Seasonality: The repeating short-term(within a year) pattern in the series.
 - Cyclicity: The pattern existing over a long-time(5-6 years) in the series. (Long-Term Seasonality).
 - Noise: The random variation in the series.
- Types of Time Series Decomposition:
 - Additive Decomposition
 - Multiplicative Decomposition

Concept of Stationarity

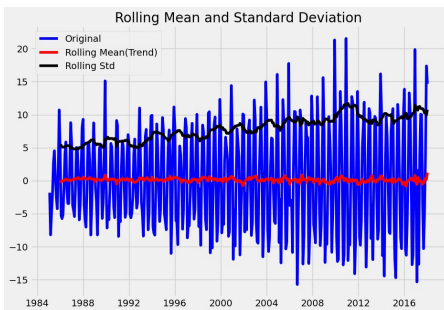
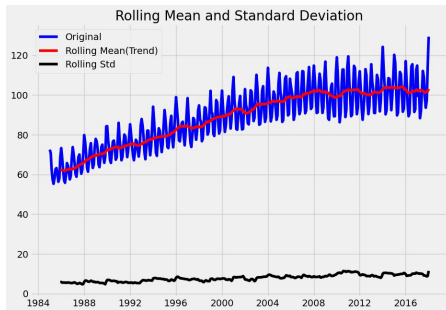
- A Stationary Time Series is one whose statistical properties, such as mean, variance, and autocorrelation, remain constant over time. In simpler terms, it does not exhibit any trend, seasonal effects, or other systematic patterns.
- Characteristics:
 - Constant Mean over Time.
 - Constant Variance over Time.
 - No component of Seasonality.

Why we need it?

Test to Check Stationarity

- One of the most popular test to use is **ADF (Augmented Dickey-Fuller) Test**. It can be used to determine the presence of unit root in the series, and hence help us understand if the series is stationary or not.
- The null and alternate hypothesis of this test is:
 - Null Hypothesis: The series has a unit root (value of $\alpha = 1$)
 - Alternate Hypothesis: The series has no unit root.
- If we succeed to reject null hypothesis, then we can say the series stationary.
- Mainly this is decided by the ADF Stat and critical values in conjunction with accepted p-values.
- If ($p\text{-value} < 0.05$) & (ADF Statistics (more -ve, more favourable) surpasses the 1, 5 and 10% critical values), then the series is considered to be stationary.

Raw Data vs Stationarized Data

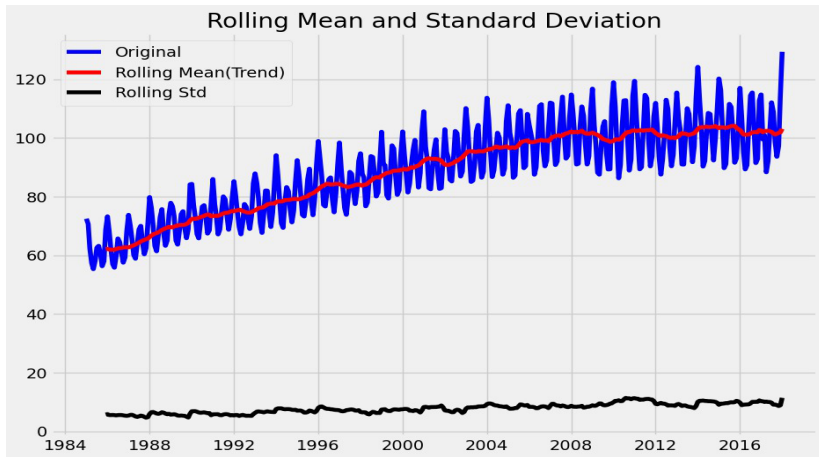


- The first graph is raw with varying statistical parameters. It's not possible to use models with these Series.
- The second graph is stationarized and then Models can be used upon.

Stages in Time Series Forecasting

- Visualising Time Series
- Stationarizing Time Series
- Model Selection
- Optimal Parameters Selection for our Models
- Training the Models
- Predictions & Evaluation of Model

Visualising the Time Series



- Only by visual inspection, we can clearly see that Our Time Series is non-stationary with an Upward Trend and Seasonal Components.

Stationarizing Time Series

Strategies for Stationarizing Time Series Data

- Apply differencing: Subtract consecutive observations to remove trends and make the series stationary.
- Transformations: Use mathematical transformations like logarithmic or square root to stabilize variance and reduce non-stationarity.
- Seasonal adjustment: Remove seasonal patterns using methods like seasonal decomposition to make the series stationary.
- Detrending: Fit a regression line to the data and subtract it to eliminate trend effects.
- Integration: Apply differencing multiple times to achieve stationarity, especially for non-stationary series with a unit root, as in ARIMA models.

We've used First Order Differencing for our Time Series.

- **SARIMA Family of Models**

- AR Model
- MA Model
- ARMA Model
- **ARIMA Model(Autoregressive Integrated Moving Average Model)**
- SARIMA Model(Seasonal ARIMA)

- **Neural Network Family of Models**

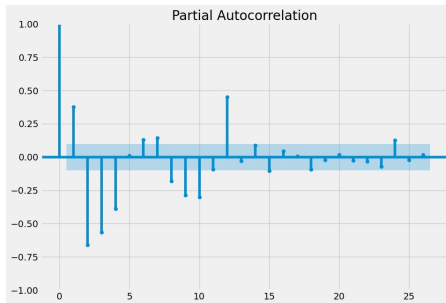
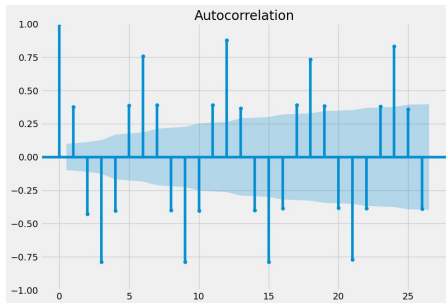
- RNN(Recurrent Neural Networks)
- LSTM(Long Short-Term Memory)
- Bi-directional LSTM
- CNN-LSTM
- **GRU**

- **We've used ARIMA and GRU Models for our Project.**

Optimal Parameters Selection for proposed ARIMA Model

- There are three parameters in ARIMA model i.e p, d and q .
- In the context of ARIMA models, p, d and q represent the orders of the autoregressive (AR), differencing (I), and moving average (MA) components respectively.
- **In this Project, optimal values of parameter p, d and q are 2, 0 and 4 respectively.**

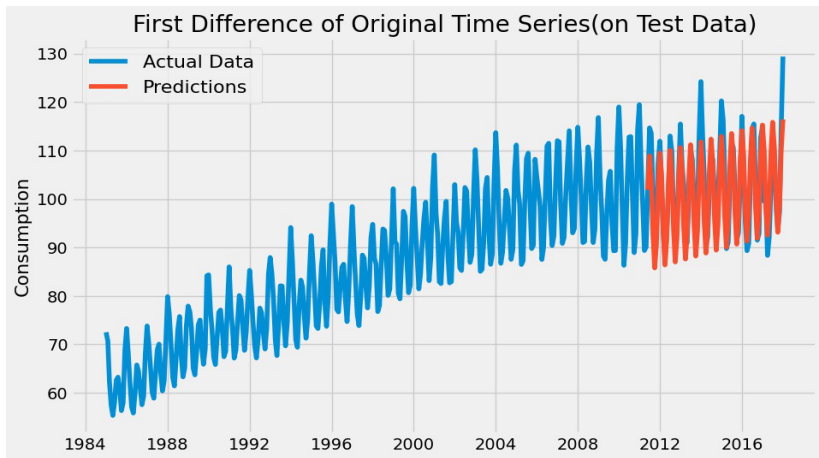
Training ARIMA Model



- The ACF plot displays the autocorrelation of a time series with its own lagged values.
 - A significant correlation at lag k indicates that the series could be predictable up to that lag.
 - The number of significant lags in the ACF plot helps in determining the **q parameter**.

- The PACF plot shows the correlation between the time series observations at different lags, while removing the effect of intermediate lags. It helps in identifying the direct effect of one lag on another.
 - Identify the lag value where the PACF plot first crosses the upper confidence interval (significant partial correlation).
 - Choose p as the lag value just before this first significant partial correlation. It helps in getting optimal value of **p parameter**.
- Alternatively, these optimal parameter values can also be obtained from the `autoarima` function from `pmdarima` library in Python.
- **It gives us the best parameters by considering the AIC (Akaike Information Criterion). Lower the AIC value, more better the Model is.** Lower AIC via higher log likelihood or less parameters of the Model. We've used this option.
- **The final optimal values obtained for p and q are 2 and 4 respectively.**

Predictions from ARIMA



- In this graph, the red one shows the predictions on the test dataset & we compared the original series with it,

Evaluation using Performance Metrics

Common error metrics used in load forecasting using ARIMA model time series include:

- Mean Absolute Error (MAE): This metric measures the average absolute errors between the forecasted values and the actual values.
- Mean Absolute Percentage Error (MAPE): MAPE calculates the percentage difference between the forecasted values and the actual values, providing a relative measure of accuracy.
- Mean Squared Error (MSE): MSE measures the average of the squared differences between the forecasted values and the actual values, giving more weight to larger errors.
- Root Mean Squared Error (RMSE): RMSE is the square root of the MSE, providing a measure of the standard deviation of the errors.
- Mean Absolute Scaled Error (MASE): MASE compares the forecasted values to a naive forecast (e.g., using the previous period's actual value) to assess the relative performance of the model

Comparison of Model's Result with Latest Papers and Publications

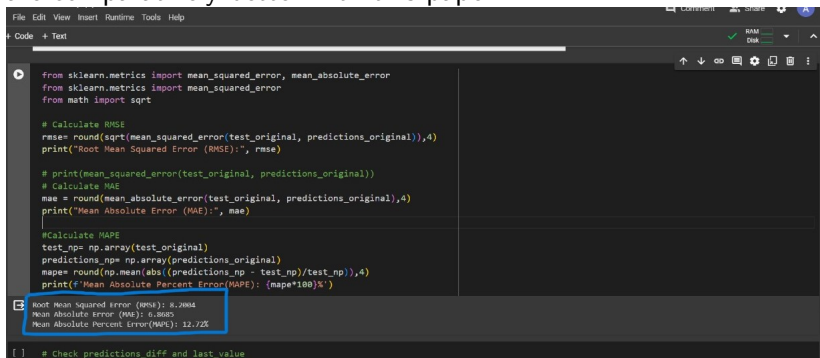
CSSE 2022, October 21-23, 2022, Guilin, China

Xuan Zhou and Xing Wu

Table 1: Model experiment results indicators

MODEL	RMSE	MAE	MAPE%
ARIMA	13.267	10.358	20.230
MLP	6.771	5.198	14.988
LSTM	6.632	5.139	14.785
Bi-LSTM	6.597	5.042	14.815
CNN-LSTM	6.308	4.801	12.910
LSTM-Attention	6.181	4.725	12.846
LSTMDA	6.123	4.692	12.463

- This is the Result we obtained from **Xuan Zhou's Paper on Load Prediction**.
- In our initial set of experimentations, we found out the values which are comparatively better with this paper.



```
File Edit View Insert Runtime Tools Help
+ Code + Text

from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.metrics import mean_squared_error
from math import sqrt

# Calculate RMSE
rmse= round(sqrt(mean_squared_error(test_original, predictions_original)),4)
print("Root Mean Squared Error (RMSE):", rmse)

# print(mean_squared_error(test_original, predictions_original))
# Calculate MAE
mae = round(mean_absolute_error(test_original, predictions_original),4)
print("Mean Absolute Error (MAE):", mae)

#Calculate MAPE
test_np= np.array(test_original)
predictions_np= np.array(predictions_original)
mape= round(np.mean(abs((predictions_np - test_np)/test_np)),4)
print(f'Mean Absolute Percent Error(MAPE): {mape*100}%')

Root Mean Squared Error (RMSE): 8.2004
Mean Absolute Error (MAE): 6.8685
Mean Absolute Percent Error(MAPE): 12.72%

[ ] # Check predictions_diff and last_value
```

- We found out the following values:
 - **RMSE: 8.2004**
 - **MAE: 6.8685**
 - **MAPE: 12.72%**

GRU at a Glance

- GRU stands for Gated Recurrent Unit, one of the popular variants of LSTM.
- It simple to train, it has less no of gates as compared to LSTM.
- We found that results were more accurate in case of GRU so we proceed with it.
- We've used the GRU Model in the Deep Learning Paradigm for the predictions.
- This involved a rigorous scripting in the TensorFlow Framework, developed by Google, one of the popular frameworks out today.
- The error comes out to be least among the popular sequence models tried by us.

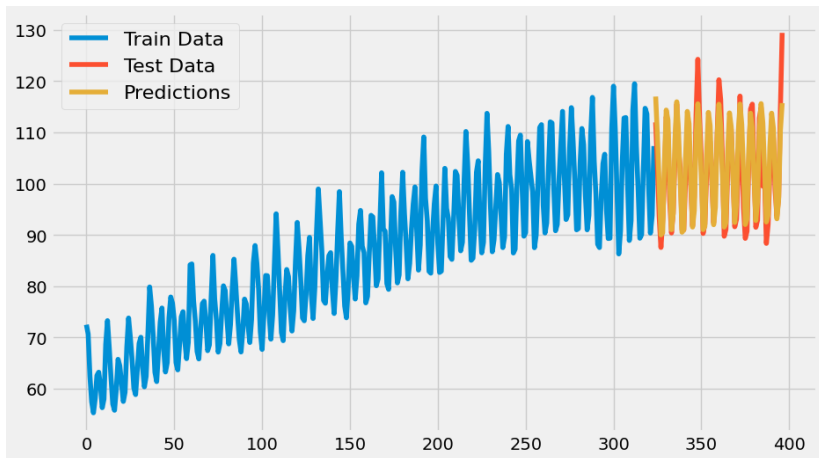
Hyperparameter Tuning

- We've have taken **two** layers of GRU each haing **20 neurons** along with a Dense Layer.
- The total number of trainable parameters came out to be **2351**.
- With the help of Time Series Generation module to generate input sequences. The length is kept at 12.
- We've kept the number of epochs as 100 and the patience parameter of Early Stopping to be 10.

Training GRU Model

- We've used Adam Optimizer with default values of learning rate.
- We've also tried learning rate scheduler, but the results were disappointing.
- We've used **Early Stopping** Technique here to avoid Overfitting.
- We've also used callbacks for employing Early Stopping and Saving Best Model.

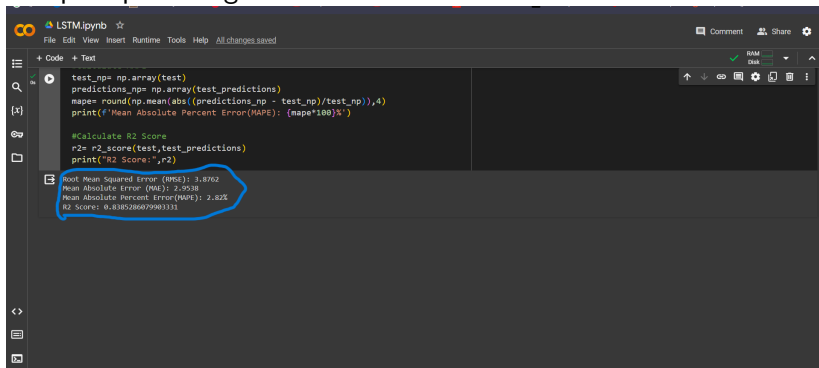
Predictions from GRU



- In this graph, the orange one shows the predictions on the test dataset & we compared the original series with it,

Evaluation Results

- In our initial set of experimentations, we found out the values which are quite promising.



```
LSTM.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

test_np= np.array(test)
predictions_np= np.array(test_predictions)
mape= round(np.mean(abs((predictions_np - test_np)/test_np)),4)
print(f'Mean Absolute Percent Error(MAPE): {mape*100}%')

#Calculate R2 Score
r2= r2_score(test,test_predictions)
print("R2 Score:",r2)

Root Mean Squared Error (RMSE): 3.8762
Mean Absolute Error (MAE): 2.9538
Mean Absolute Percent Error(MAPE): 2.82%
R2 Score: 0.8385286879903331
```

- We found out the following values:
 - **RMSE: 3.8762**
 - **MAE: 2.9538**
 - **MAPE: 2.82%**

Comparison of both Models

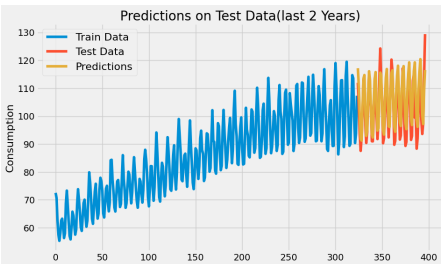
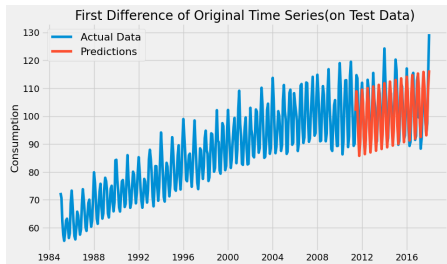
Errors with GRU model

- Root Mean Squared Error (RMSE): 5.998
- Mean Absolute Error (MAE): 4.877
- Mean Absolute Percent Error(MAPE): 4.8

Errors with ARIMA model

- Root Mean Squared Error (RMSE): 4.7735
- Mean Absolute Error (MAE): 3.8698
- Mean Absolute Percent Error(MAPE): 10.13

Predictions of both models



- 1st prediction is of ARIMA Model
- 2nd prediction is of GRU Model

Links to Colab Notebooks:

- [ARIMA](#)
- [GRU](#)

Limitations and Future Work Directions

- We're really limited by the high frequency variations in the data.
- Also, we would like to have a large dataset to work up as it will give more broader perspective.
- For more better predictions, we're planning to employ **Wavelet Transform** in our project to handle those high frequency variations.
- Also usage of ML models like XGBoost will reduce error as compared to ARIMA & GRU models considerably.
- As the last step forward, we plan to make the Final Output as the Ensemble of some of the Models.

References

- 1 **Xuan Zhou & Xing Wu** s' Paper: Load prediction model based on LSTM and attention mechanism
- 2 Time series analysis and possible applications
- 3 Analysis and forecasting of Time-Series data using S-ARIMA, CNN and LSTM
- 4 Time Series Analysis of Electricity Consumption Forecasting Using ARIMA Model
- 5 Predict stock prices with ARIMA and LSTM
- 6 Understanding LSTM- A tutorial into Long Short-Term Memory Recurrent Neural Networks
- 7 Puja P. Pathak's Blog on Time Series Analysis
- 8 Nagesh Chauhan's Notebook on Kaggle

That's all from Our Side!



Thank You
For Your Attention

- Group members:
 - 2102011: Ayush Raj
 - 2102024: Aditya Manna
 - 2102046: Ratan Deep
 - 2102060: Kumar Vaibhav