



# UG Project

## Load Forecasting using Machine Learning and Statistical Techniques

Under the supervision of Prof. Sobhita Meher

BY :

Eshaan Agarwal 20085027 (EEE Part-4 B.Tech)

Aadish Jain 20085001 (EEE Part-4 B.Tech)

Tanvi 20085102 (EEE Part-4 B.Tech)

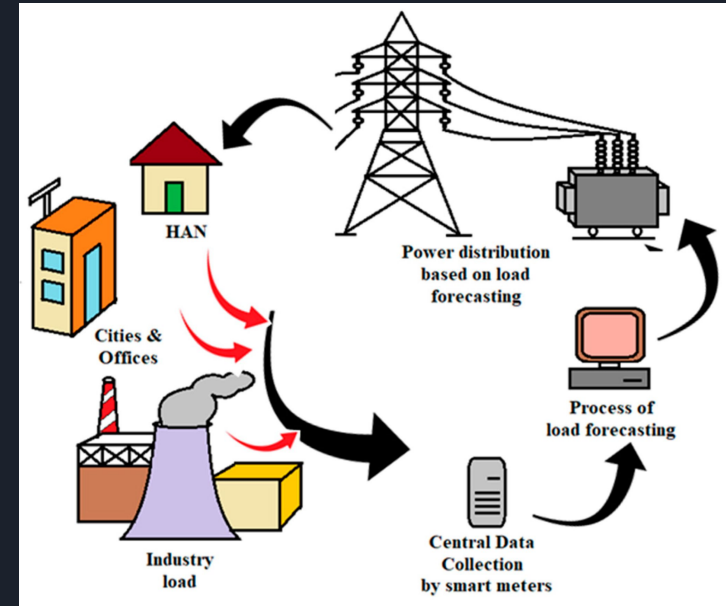


# Table of Content

- Introduction to problem: What is Load Forecasting?
- Purpose of Load forecasting
- Types of Load forecasting
- Factors affecting Load forecasting
- Dataset
  - Preprocessing and Scrapping
  - Analysis: Trends and Seasonality
- Load curve sample
- Algorithms implemented: Benefits, Drawbacks and Comparison
- Results
- Conclusion
- Acknowledgement and References

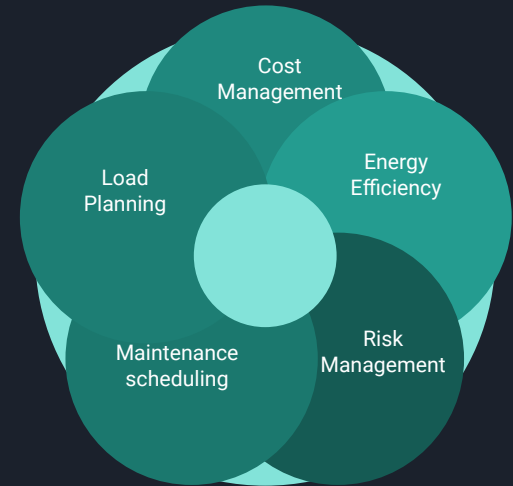
# Load Forecasting

- Process of predicting the future demand or consumption of electricity or energy in a particular area.
- Use of statistical and mathematical models to analyze historical consumption data and identify trends, patterns, and factors that affect energy consumption in the future.
- Essential tool for energy companies, utilities, and grid operators :
  - plan and optimize their generation and transmission resources,
  - meet the future energy demands of their customers.
- Timely implementations of such decisions lead to improvement of network reliability and to reduced power failures.

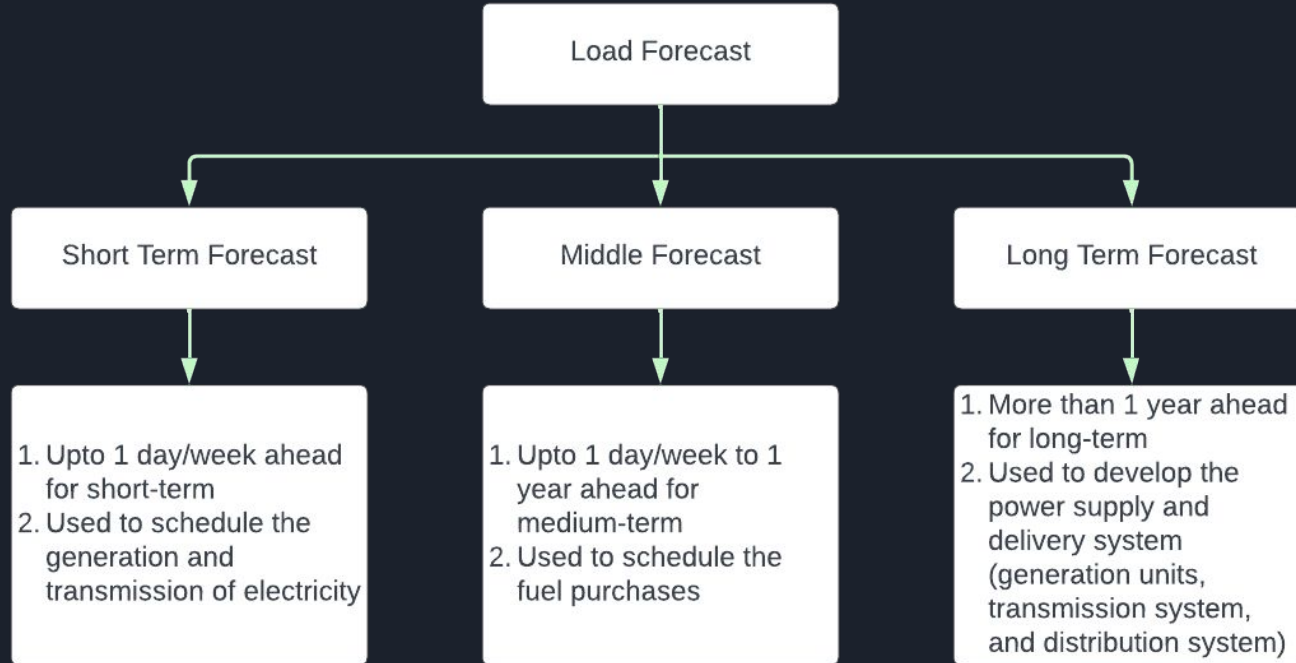


# Purpose of Load forecasting

- **Planning:** Plan for future energy demand and ensure that there is enough capacity to meet that demand. Accurate load forecasting helps utilities and energy providers determine the optimal amount of resources to invest in power generation and transmission infrastructure.
- **Cost management:** Helps manage costs by reducing the need for expensive peak load generators or purchasing power from external sources during periods of high demand.
- **Energy efficiency:** Helps in identifying periods of low demand when energy usage can be reduced, helping to promote energy efficiency and reduce costs.
- **Maintenance scheduling:** Help utilities and energy providers schedule maintenance and repairs to power generation and transmission infrastructure during periods of low demand, minimizing the impact on energy supply.
- **Risk management:** Energy companies manage risks associated with energy price volatility, capacity shortages, and unexpected events such as extreme weather conditions.

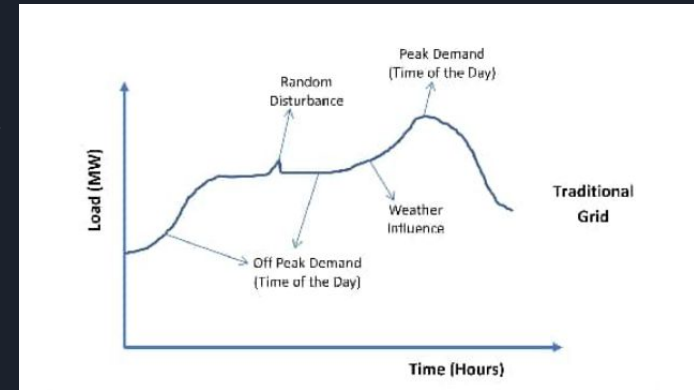


# Types of forecasting



# Factors affecting Load forecasting

- **Weather conditions:** Extreme temperatures, such as heatwaves and cold snaps, can significantly impact energy demand for heating and cooling.
- **Time of day and day of the week:** Energy demand varies depending on the time of day and day of the week, with peak demand typically occurring during weekday afternoons and evenings.
- **Seasonal variations:** Seasonal variations in energy demand, such as higher demand for heating during the winter months, can impact load forecasting.
- **Policy and regulatory changes:** Changes in energy policies and regulations, such as the introduction of renewable energy incentives or carbon taxes impact energy demand and load forecasting.

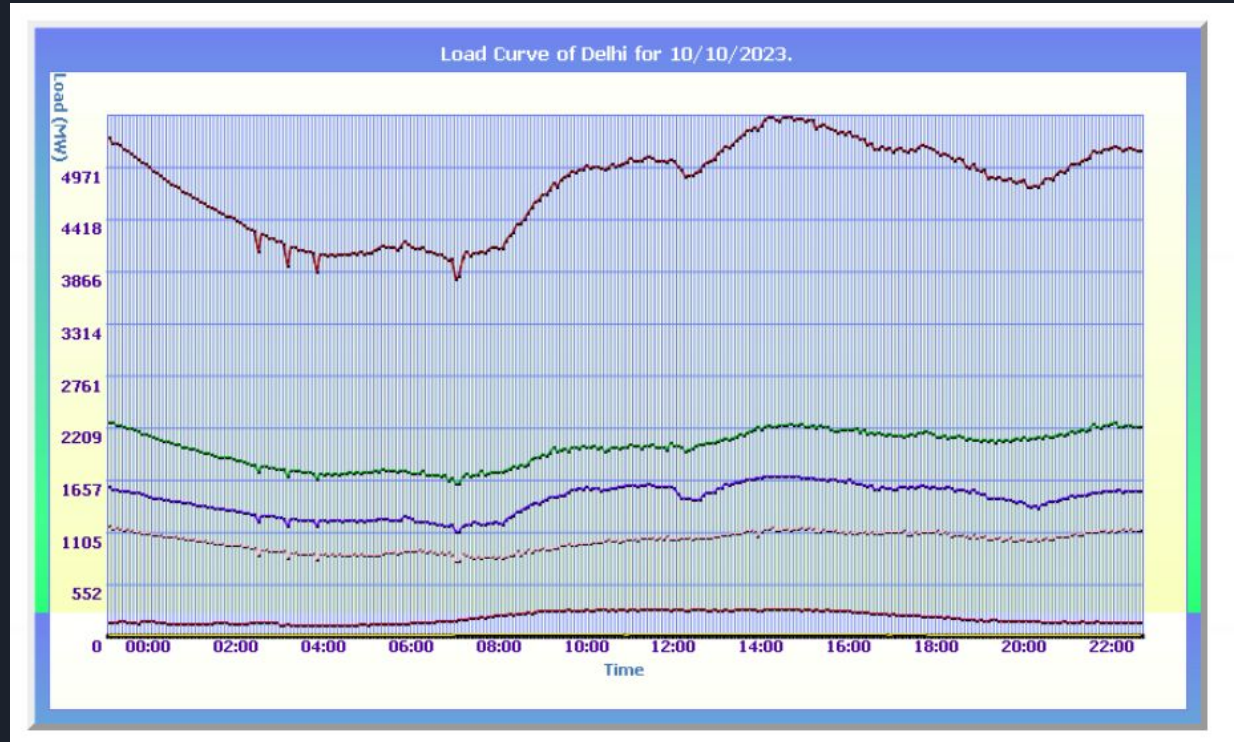




# Dataset

- Dataset - Time series data of [Delhi Electricity Board's Data](#) which provides demand of the state in load values (MW) in time step of 5 minutes using Data Scraping Script
  - Can be used to scrape data for real-time prediction
  - 288 values (  $24 \times 60 / 5$  ) of time-series data for each day
  - Scraped Data for 5 months using Beautiful Soup from Oct 2022 to Feb 2023
  - Total data points - approx 43200 data points ( $288 \times 5 \times 30$ )
- Humidity and Temperature data in time step of 5 minutes for corresponding period scraped from a weather site for NewDelhi (INEWDE9)
- Features Used -
  - Time of the day.
  - Day of the week.
  - Temperature at the corresponding time.
  - Humidity at the corresponding time.

# Load Curve of Delhi at 10/10/2023





# Dataset Preprocessing and Scrapping

	datetime	load	humidity	temperature
0	01/10/2022 00:00	4581.220	46	24
1	01/10/2022 00:05	4553.49	49	22
2	01/10/2022 00:10	4533.33	44	24
3	01/10/2022 00:15	4507.54	60	27
4	01/10/2022 00:20	4489.97	43	20
...	...	...	...	...
35026	30/03/2023 23:20	2353.0	42	23
35027	30/03/2023 23:25	2342.45	45	30
35028	30/03/2023 23:30	2347.89	60	30
35029	30/03/2023 23:35	2333.57	65	23
35030	30/03/2023 23:40	2308.27	53	28



This data is being extracted from SLDC website of Delhi and Wunderground by an automated script.

The mentioned website updates the data for every 5 minutes. Thus we have data in the time steps of 5 minutes.

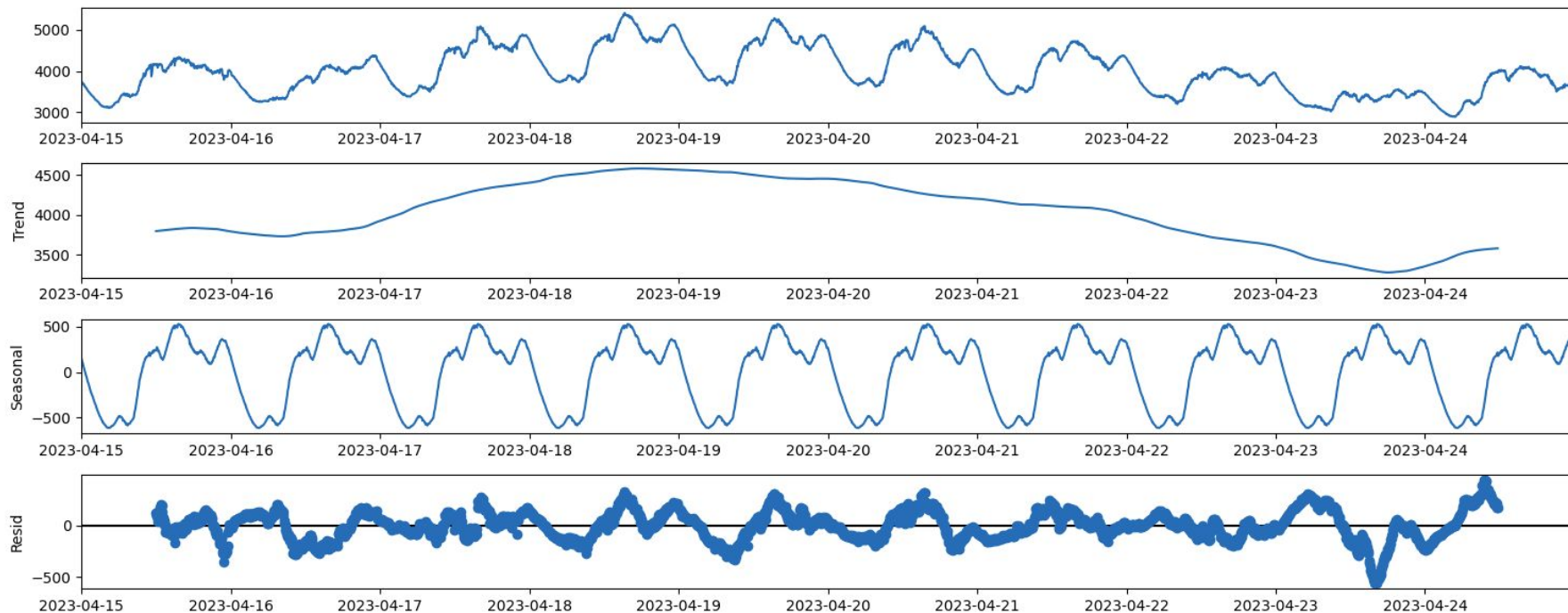
Problem - However, there are certain days when the SLDC website was not functional due to maintenance or any other factors, which implies the data collected might have some missing values.

Solution used - Used standard techniques for imputing like forward fill and back fill to make data more uniform

```
data_new = data.asfreq(freq='5T', method='bfill')
```

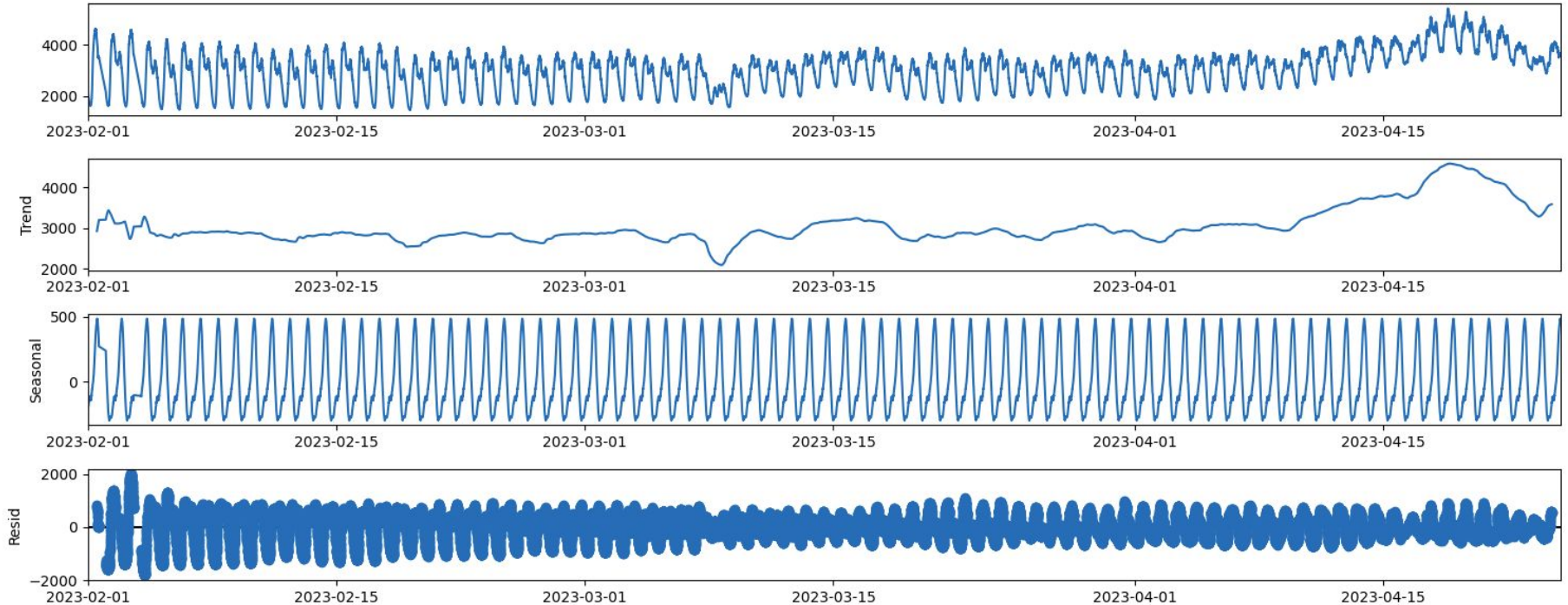
# Dataset Analysis (Trends and Seasonality)

Trends and Seasonality in Load data from 15-04-23 to 24-04-23



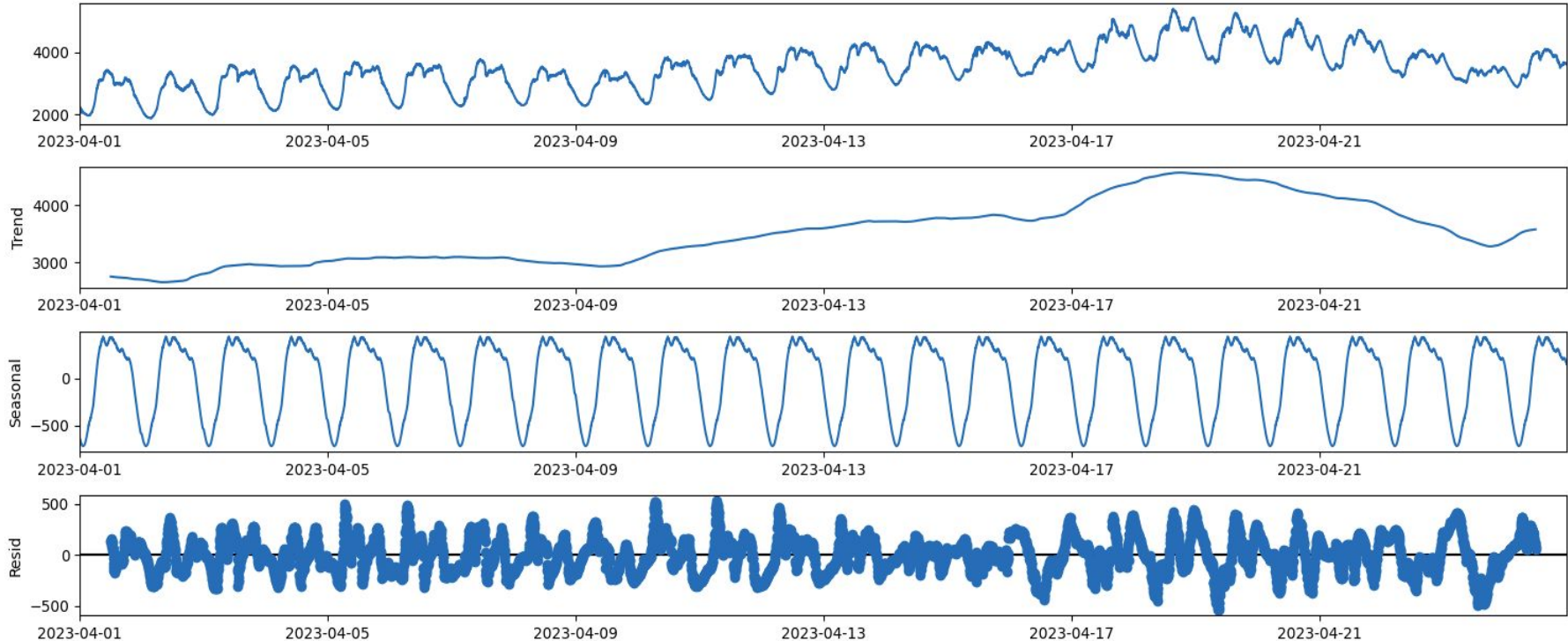
# Dataset Analysis (Trends and Seasonality)

Trends and Seasonality in Load data from 01-02-23 to 24-04-23



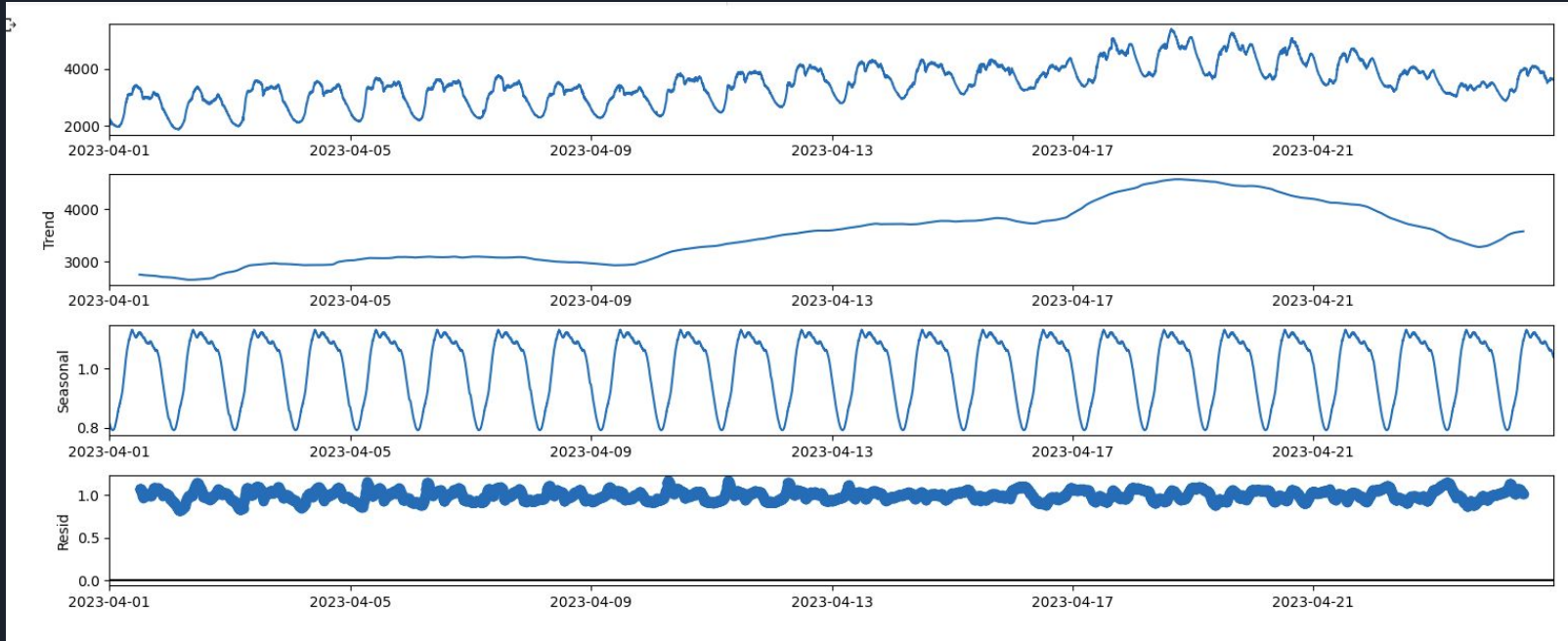
# Dataset Analysis (Trends and Seasonality)

Weekly Trends and Seasonality in Load data from 01-04-23 to 24-04-23



# Dataset Analysis (Trends and Seasonality)

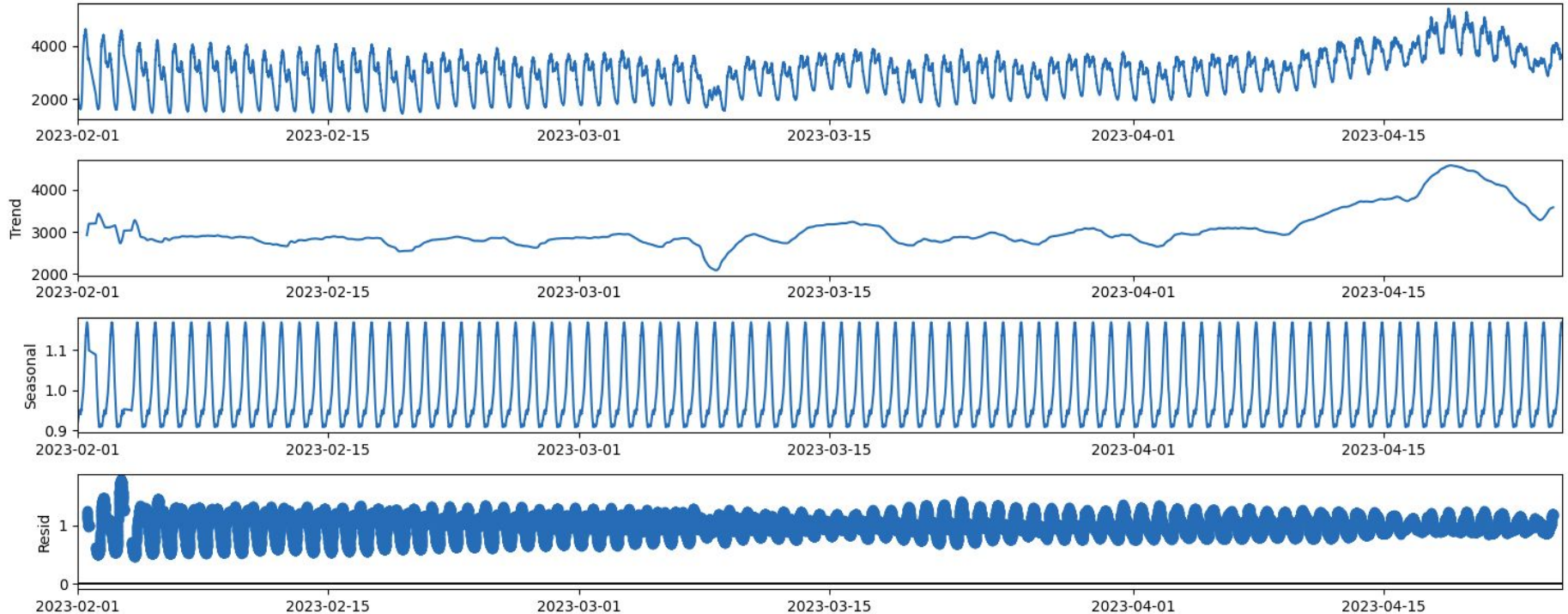
Multiplicative Trends and Seasonality in Load data from 01-04-23 to 24-04-23





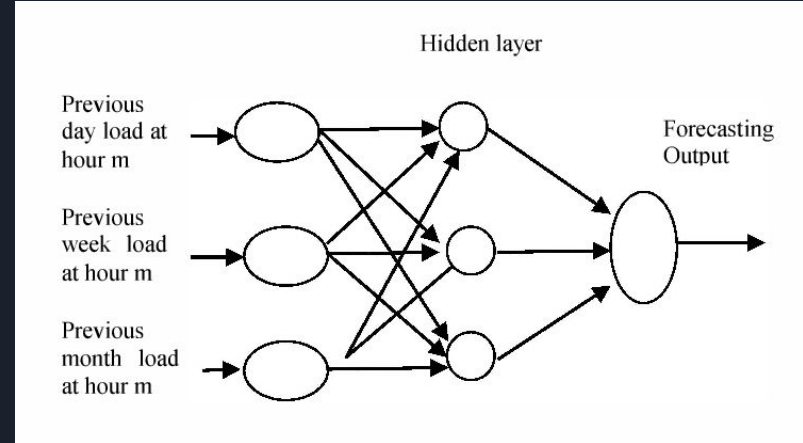
# Dataset Analysis (Trends and Seasonality)

Multiplicative Trends and Seasonality in Load data from 01-02-23 to 24-04-23



# Algorithms Implemented for Load forecasting

1. Simple Moving Average (SMA) - [link](#)
2. Weighted Moving Average (WMA) - [link](#)
3. Simple Exponential Smoothing (SES) - [link](#)
4. AutoRegressive Integrated Moving Average (ARIMA) - [link](#)
5. Seasonal AutoRegressive Integrated Moving Average (SARIMA) - [link](#)
6. XGBoost Algorithm - [link](#)
7. Simple Feedforward Neural Network (FFNN) - [link](#)
8. Recurrent Neural Networks - [link](#)
9. Long Short-Term Memory - [link](#)
10. Gated Recurrent Units - [link](#)
11. Convolutional Neural Networks - [link](#)





# Simple Moving Average (SMA)

- Calculating the average of the load data for a specific time period and then using this value to forecast the load for the next time period.
- The formula for a simple moving average is

$$F_t = \frac{\sum_{i=1}^n A_{t-i}}{n}$$

where  $F(t)$  = Forecast for the coming period,  $n$  = Lookup sliding window, and  $A(t-1)$ ,  $A(t-2)$ ,  $A(t-3)$  and so on are the actual occurrences in the past period, two periods ago, three periods ago and so on respectively.

- There is a closer following of the trend in a shorter time span but it produces more oscillation. On other hand, a longer time span gives a smoother response but lags the trend.

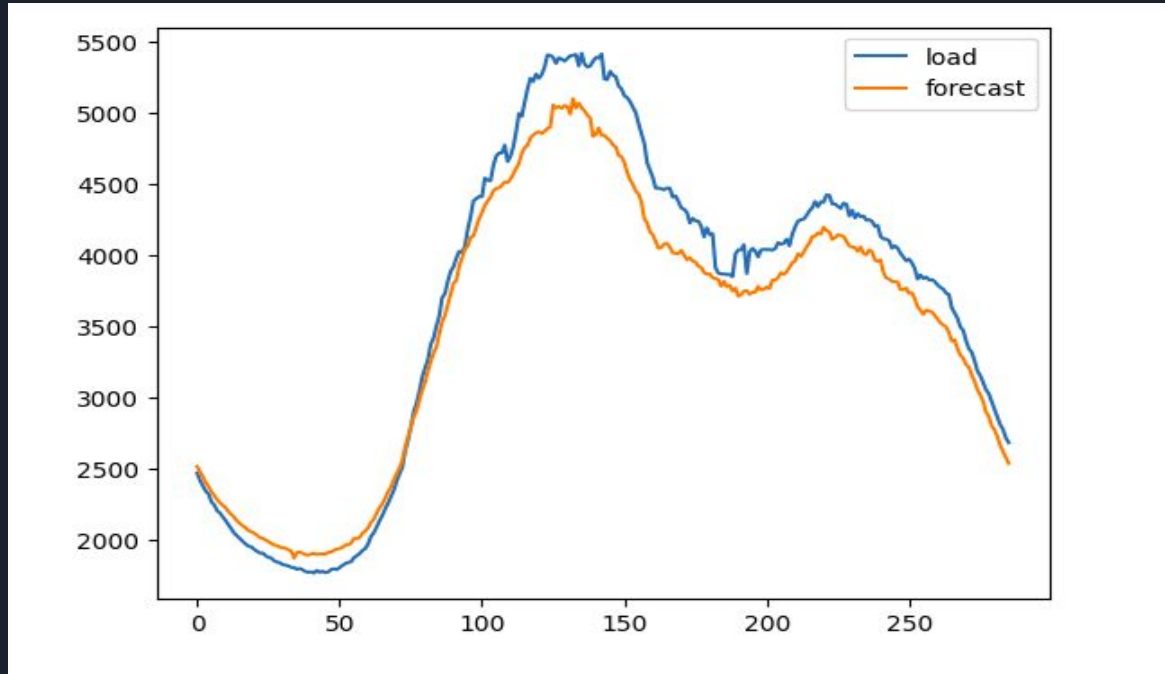




# Drawbacks of Simple Moving Average (SMA)

- **Limited accuracy:**
  - Good for short-term forecasts, but accuracy decreases for longer-term forecasts.
  - Method assumes that the load pattern is stable over the forecast horizon and does not account for other factors that may affect the load.
- **Ignores trends and seasonality:**
  - Assumes that the load demand is constant over time and does not consider trends or seasonality in the data which Lead to inaccurate forecasts.
- **Slow to respond to changes:** Based on a fixed window of historical data, which means that it can be slow to respond to sudden changes in load demand.
- This can be a problem in situations where load demand changes rapidly, such as during **extreme weather events.**

# Results



Actual Load vs forecasted load for SMA at 14-04-23



# Weighted Moving Average (WMA)

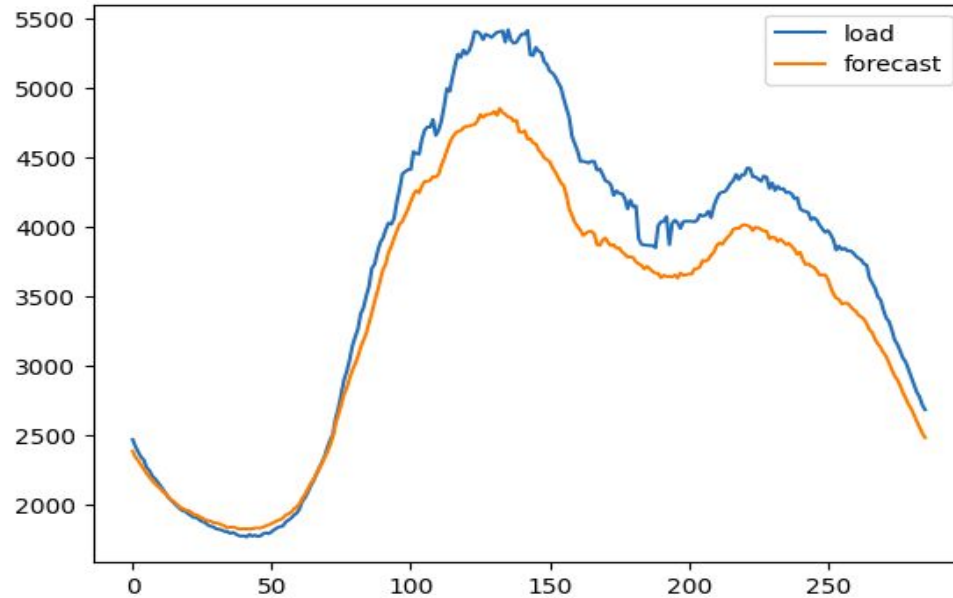
- The simple moving average gives equal weight to each component of the moving average database, a weighted moving average allows any weights to be placed on each element, providing, of course, that the sum of all weights equals 1.
- The formula for the weighted moving average is

$$F_t = \sum_{i=1}^n w_i A_{t-i}$$

$$\sum_{i=1}^n w_i = 1$$

Where  $F(t)$  = Forecast for the coming period,  $n$  = the total number of periods in the forecast,  $w(i)$  = the weight to be given to the actual occurrence for the period  $t-i$ ,  $A(i)$  = the actual occurrence for the period  $t-i$ .

# Results



Actual Load vs forecasted load for WMA at 14-04-23



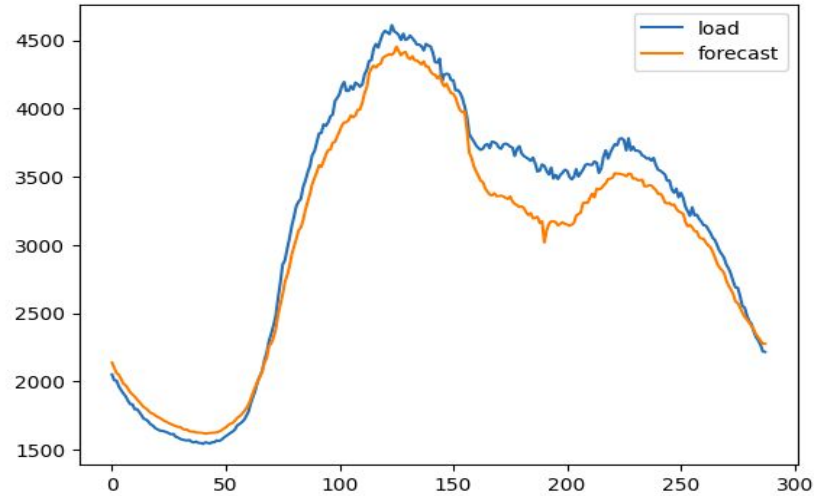
# Simple Exponential Smoothing (SES)

- The Simple Exponential Smoothing (SES) method is a widely used statistical technique for load forecasting. It is based on the assumption that the future load will be a function of the past load values and an exponentially weighted average of the past forecast errors.
- The simple exponential smoothing forecast is given by

$$F_t = \alpha A_{t-1} + (1 - \alpha)F_{t-1}$$

where  $F(t)$  is the exponentially smoothed forecast for  $i$ th interval,  $\alpha$  is the desired response rate, or smoothing constant.

# Results



Actual Load vs forecasted load for SES at 14-04-23

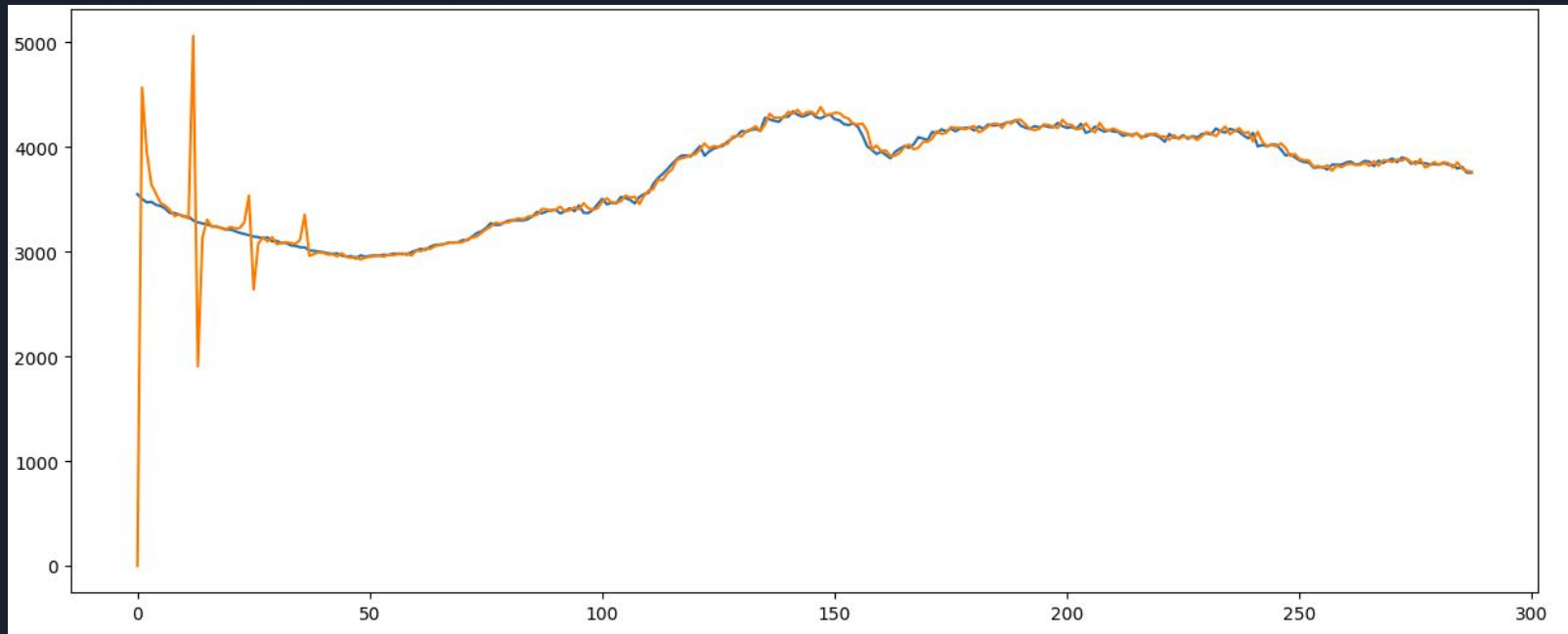


# AutoRegressive Integrated Moving Average (ARIMA)

- ARIMA models are widely used because they can capture the patterns and trends in historical load data and use them to make predictions about future load.
- It can capture both short-term and long-term patterns in the data and adjust to changes in those patterns over time and trends in the data, as well as are able to handle non-stationary data.
- The ARIMA model is a combination of three components:
  - **Autoregression (AR)** - this component models the linear relationship between the previous values of the series and the current value.
  - **Moving Average (MA)** - this component models the linear relationship between the errors (the difference between the actual values and the predicted values) of the series and the lagged values of the errors.
  - **Integration (I)** - this component models the non-stationary behavior of the series by taking the differences between the values of the series at different time points.

# Results

## Actual Load vs Forecasted Load using ARIMA





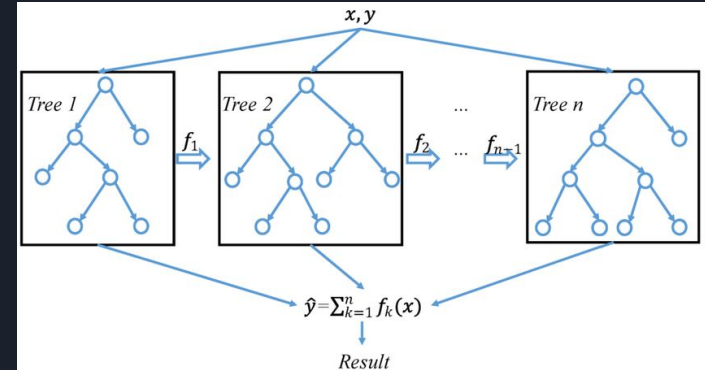


# Comparing SMA, WMA, SES and ARIMA

1. **Accuracy:** ARIMA is generally considered to be the most accurate method for load forecasting. SES can also be reasonably accurate for short-term forecasts, while SMA and WMA are generally less accurate.
2. **Flexibility:** ARIMA and SES are more flexible than SMA and WMA. They can adapt to changes in load demand patterns over time and account for trends, seasonality, and other factors that can affect load demand. SMA and WMA, on the other hand, are based on a fixed window of historical data and cannot adapt to changes in demand patterns.
3. **Data requirements:** SMA and WMA only require historical load demand data, while SES and ARIMA may require additional data such as weather data, economic data, or other external factors that can influence load demand.
4. **Computation:** SMA and WMA are relatively simple and computationally efficient methods. SES is slightly more complex than SMA and WMA, while ARIMA is the most computationally intensive method.

# XGBoost Gradient Boosting Algorithm

- Gradient boosting is an ensemble learning method that combines the predictions of multiple weak learners (typically decision trees) to create a more accurate and robust model.
- The key idea behind gradient boosting is to sequentially fit new base learners to the residuals (or errors) of the previous learners, with the aim of reducing the residuals at each step.
- One of the main features of XGBoost is its optimized algorithm for finding the best splits in the decision trees. Instead of exhaustively searching all possible splits, XGBoost uses a technique called "approximate tree learning" to efficiently find the optimal splits, which greatly speeds up the training process.





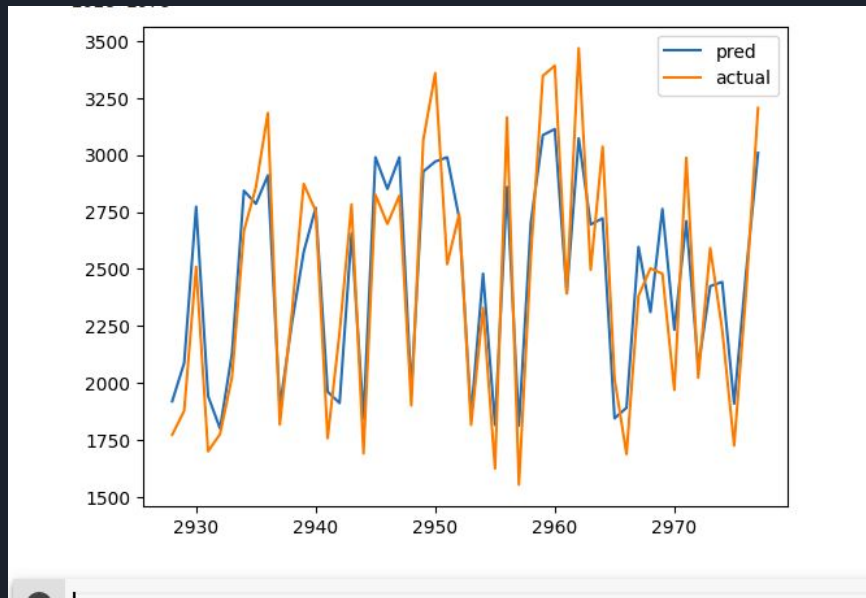
# Benefits of XGBoost Gradient Boosting Algorithm

- **Accuracy:** Known for its improved accuracy compared to traditional methods.
- **Efficiency:** Optimized for speed and efficiency, making it well-suited for large-scale load forecasting tasks. It can handle large datasets and non linear relationships.
- **Flexibility:** XGBoost is flexible and can handle various types of data, including numerical, categorical, and ordinal data. This makes it well-suited for load forecasting tasks that involve different types of features.
- **Regularization:** Handles missing data values effectively by assigning a default direction for them when splitting a tree node. This means that the missing values are treated as either left or right children, depending on which direction gives the best split.

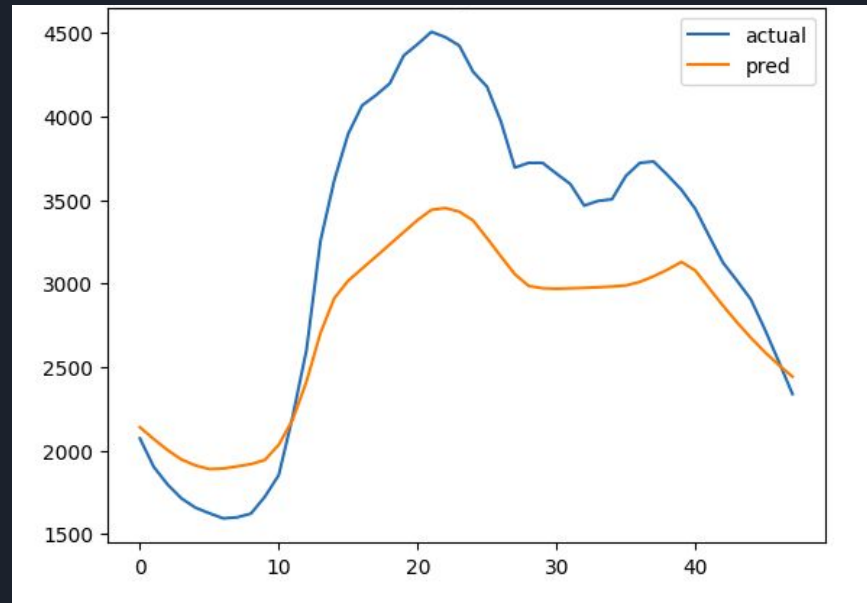
```
model = XGBRegressor(n_estimators=1000, max_depth=7, eta=0.1, subsample=0.7, colsample_bytree=0.8)
```

# Results

XGBoost



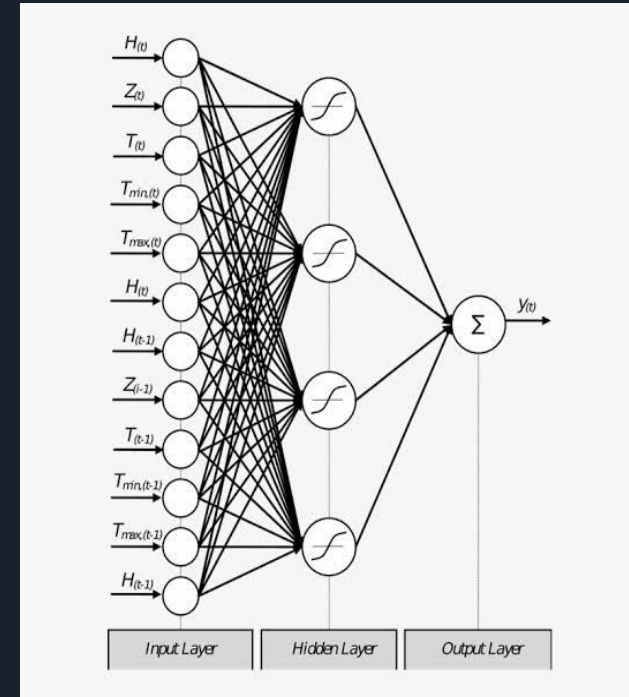
Actual Load vs Predicted Load across a period of 50 days



Actual Load vs Predicted Load on 15-03-23

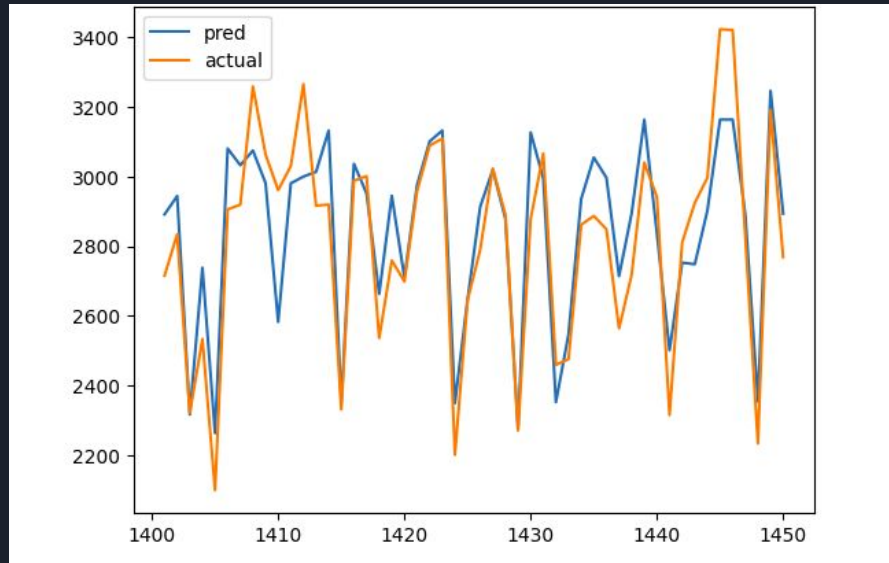
# Simple Feedforward Neural Network (FFNN)

- Type of artificial neural network.
- Called "**feedforward**" because the data flows in one direction through the network, from input to output, without any feedback connections or loops.
- Neurons in each layer are fully connected to the neurons in the next layer, and non linear activation function introduces non-linearity into the network and **allows it to learn complex patterns in the data**.
- FFNNs are particularly well-suited for load forecasting because they can learn complex non-linear relationships between the load demand and various input factors.
- Suitable for Short term, mid-term and long term forecasting based on the complexity and volume of data fed to the network.

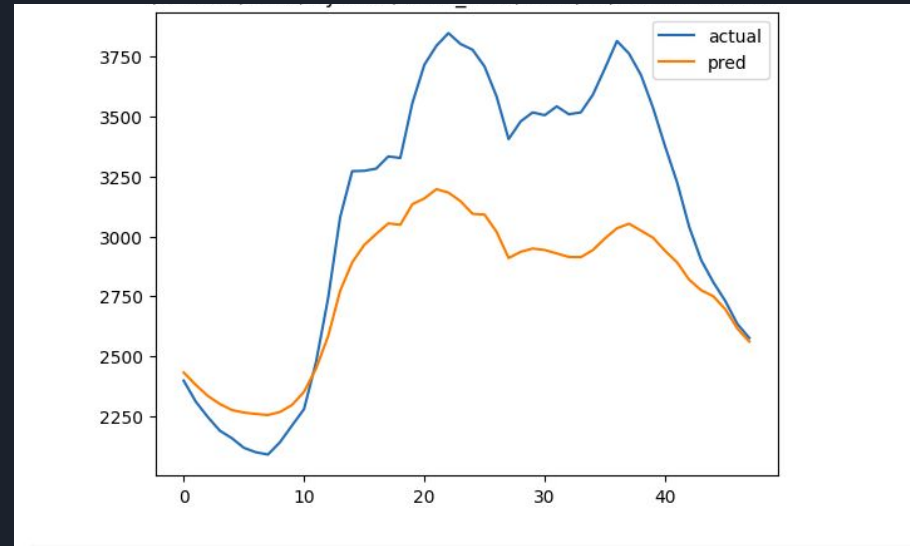


# Results

## Feed Forward Neural Network



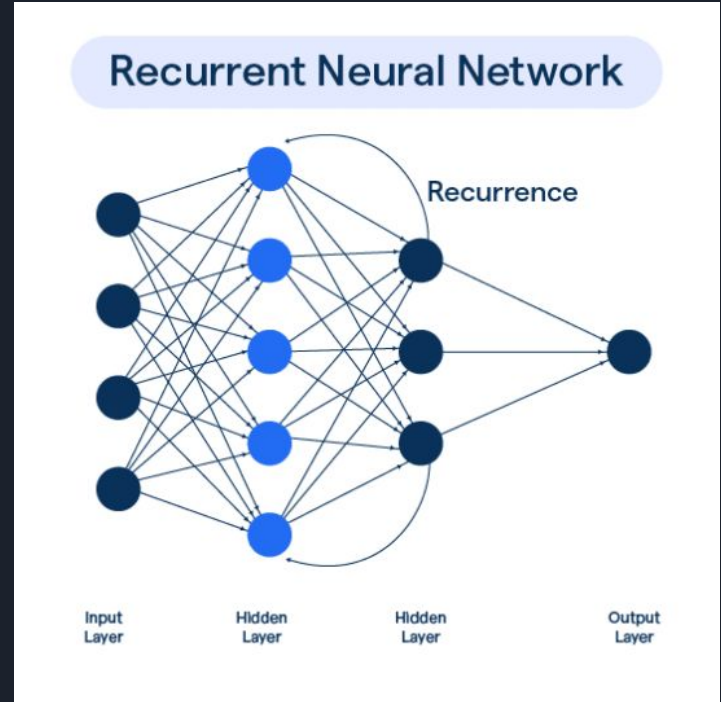
Actual Load vs Predicted Load across a period of 50 days



Actual Load vs Predicted Load on 13-03-23

# Recurrent Neural Networks (RNN)

- RNNs are designed to work with sequential data, making them well-suited for tasks where the order of input data is crucial, such as time series forecasting.
- RNNs excel at capturing temporal dependencies in sequential data, allowing them to learn patterns and trends in short-term load fluctuations.
- The key feature of RNNs is the presence of recurrent connections that enable the network to maintain a hidden state, preserving information from previous time steps.
- RNNs are effective in recognizing and predicting short-term patterns in load data. They can capture daily and weekly fluctuations, responding well to variations in load demand.





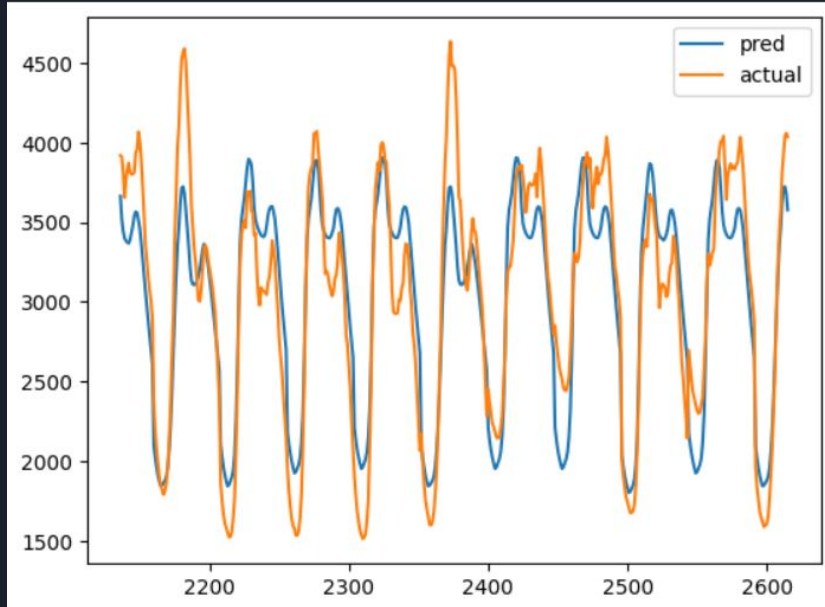
# Merits and Drawbacks of RNN

Merits	Drawbacks
<ul style="list-style-type: none"><li>• Effective at capturing temporal dependencies.</li><li>• Naturally handles sequential data in load forecasting.</li><li>• Retains memory for considering historical context.</li><li>• Adaptable to varying input/output lengths and real-time changes.</li><li>• Efficiently learns patterns and dependencies across time.</li><li>• Handles multivariate input data for comprehensive forecasting.</li><li>• Learns from historical load patterns for accurate predictions.</li></ul>	<ul style="list-style-type: none"><li>• Faces vanishing/exploding gradient problem.</li><li>• Limited memory for capturing very long-range dependencies.</li><li>• Training complexity and computational intensity.</li><li>• Difficulty in capturing complex long-term patterns.</li><li>• Sensitivity to hyperparameter tuning.</li><li>• Challenges in modeling irregular patterns or sudden changes.</li><li>• Limited parallelization during training.</li></ul>

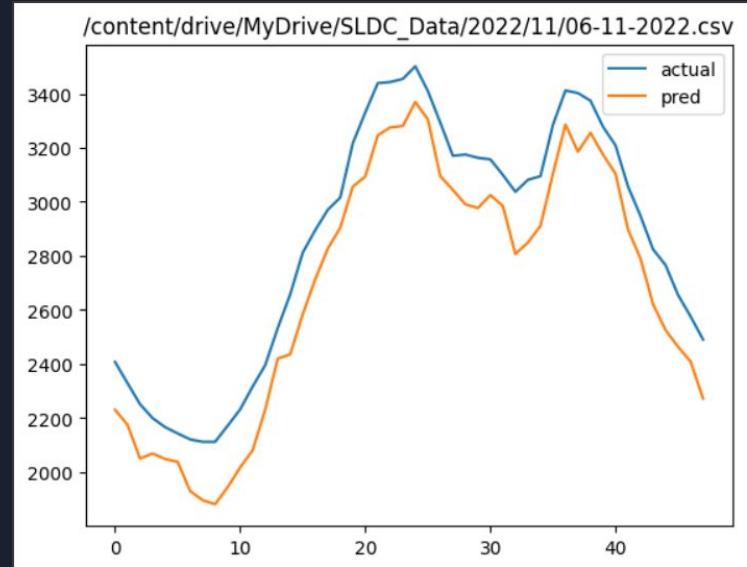


# Results

## Recurrent Neural Networks



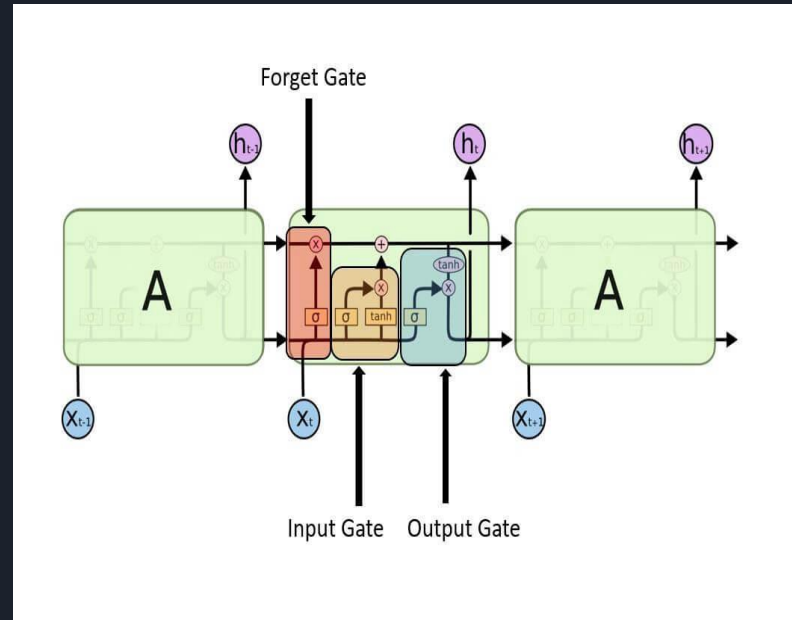
Actual Load vs Predicted Load across a period of 50 days



Actual Load vs Predicted Load on 06-11-22

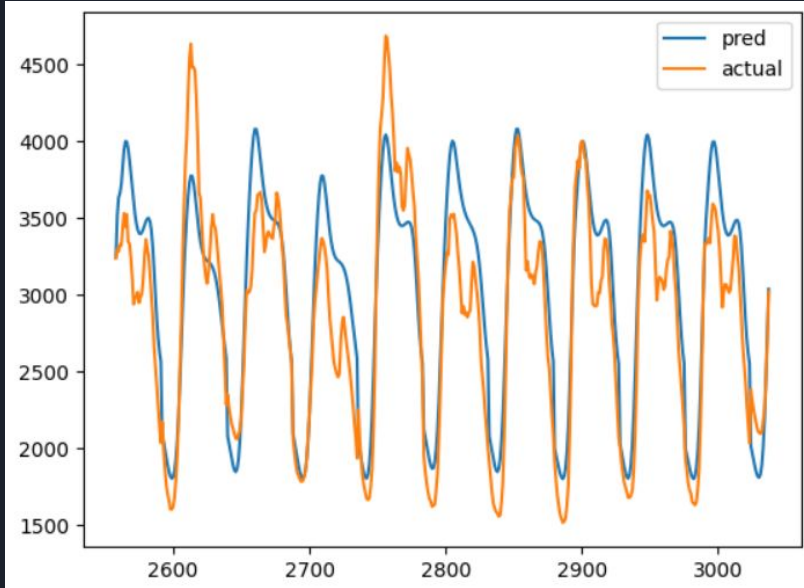
# Long Short-Term Memory (LSTM)

- LSTMs, a type of RNNs are designed to address the vanishing gradient problem. They excel at handling sequential data, making them ideal for time series forecasting like short-term load forecasting.
- LSTMs introduce memory cells and gating mechanisms (input, forget, output gates) for selective information retention, crucial for capturing both short and long-term dependencies.
- Input gate regulates new information flow, forget gate controls retention of past information, and output gate determines information for the next layer or final prediction. These gates enable precise control over information flow.
- LSTMs have a more complex architecture compared to basic RNNs, incorporating memory cells and gating mechanisms. While this complexity addresses the vanishing gradient problem, it also requires careful tuning of hyperparameters for optimal performance.

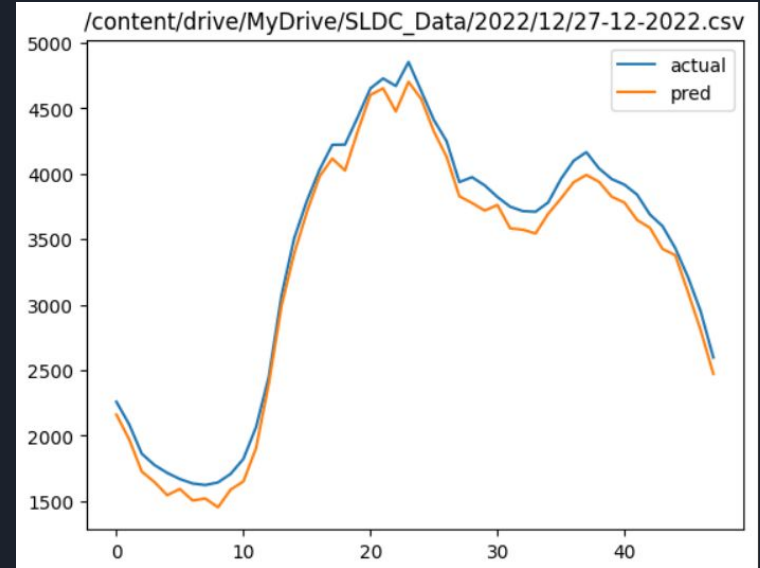


# Results

## Long Short-Term Memory



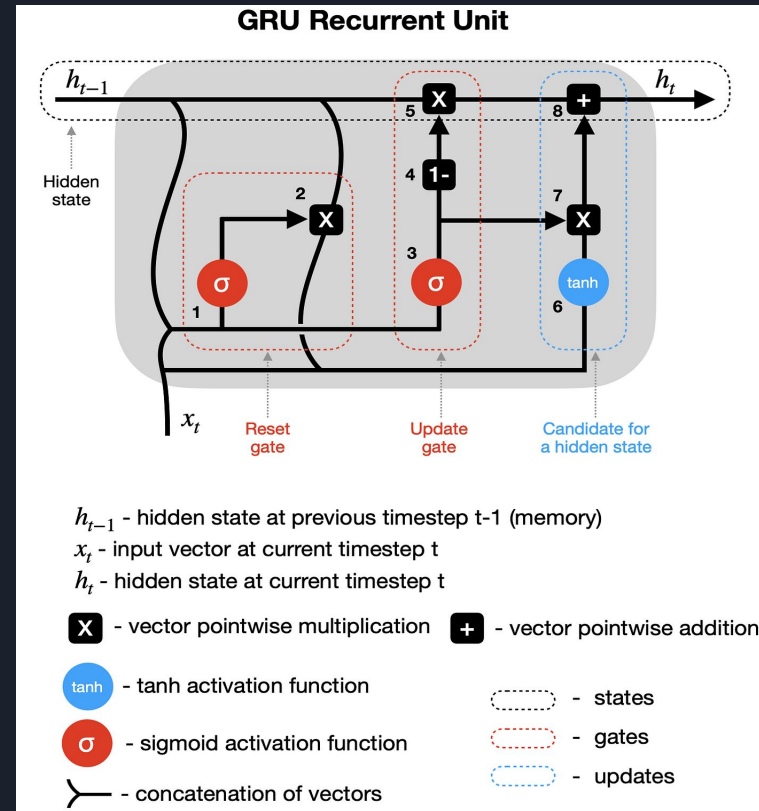
Actual Load vs Predicted Load across a period of 40 days



Actual Load vs Predicted Load on 27-12-22

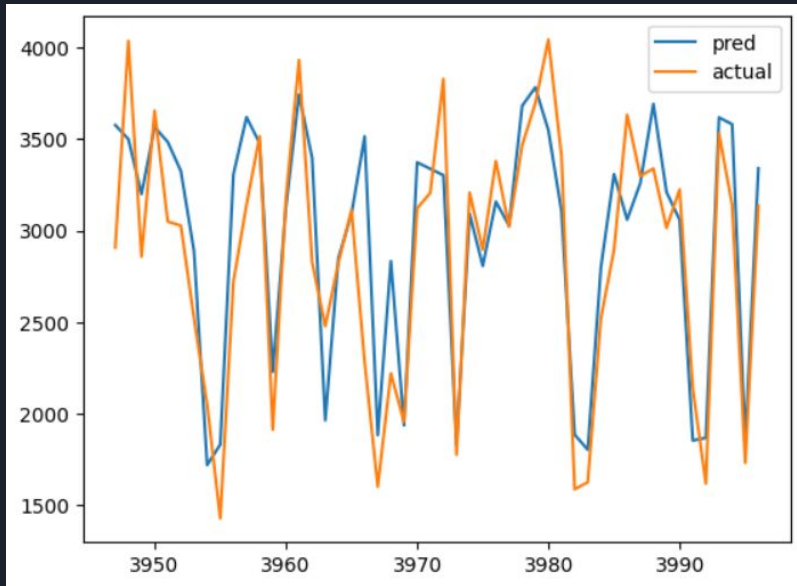
# Gated Recurrent Units (GRU)

- GRUs excel in handling sequential data efficiently, making them well-suited for Short-Term Load Forecasting (STLF).
- Two gating mechanisms in GRUs (reset and update gates) allow for adaptive information flow. The network can selectively reset its hidden state and update the stored information, facilitating the modeling of changing load conditions.
- GRUs are particularly effective in capturing short-term dependencies, making them suitable for STLF tasks focused on immediate future predictions.

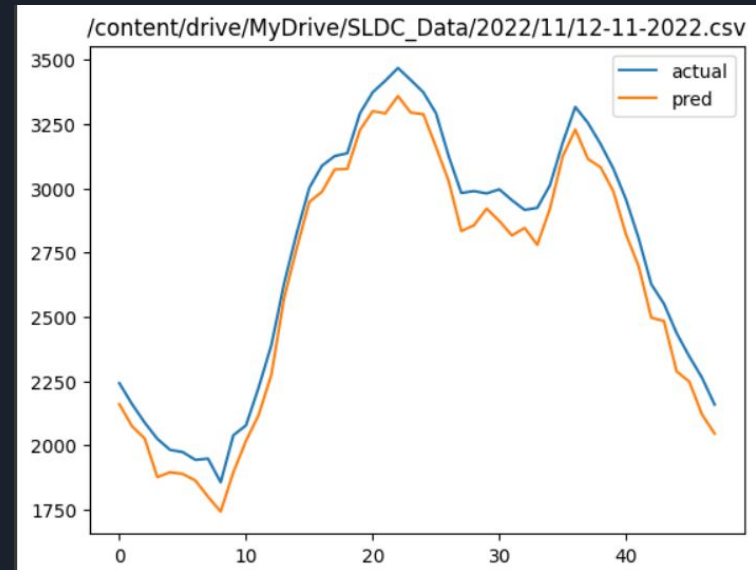


# Results

## Gated Recurrent Units



Actual Load vs Predicted Load across a period of 40 days



Actual Load vs Predicted Load on 12-11-22

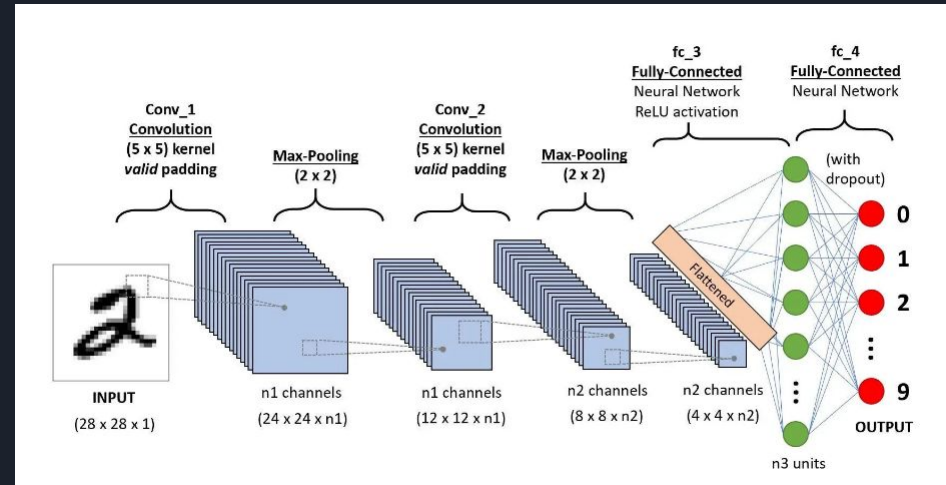


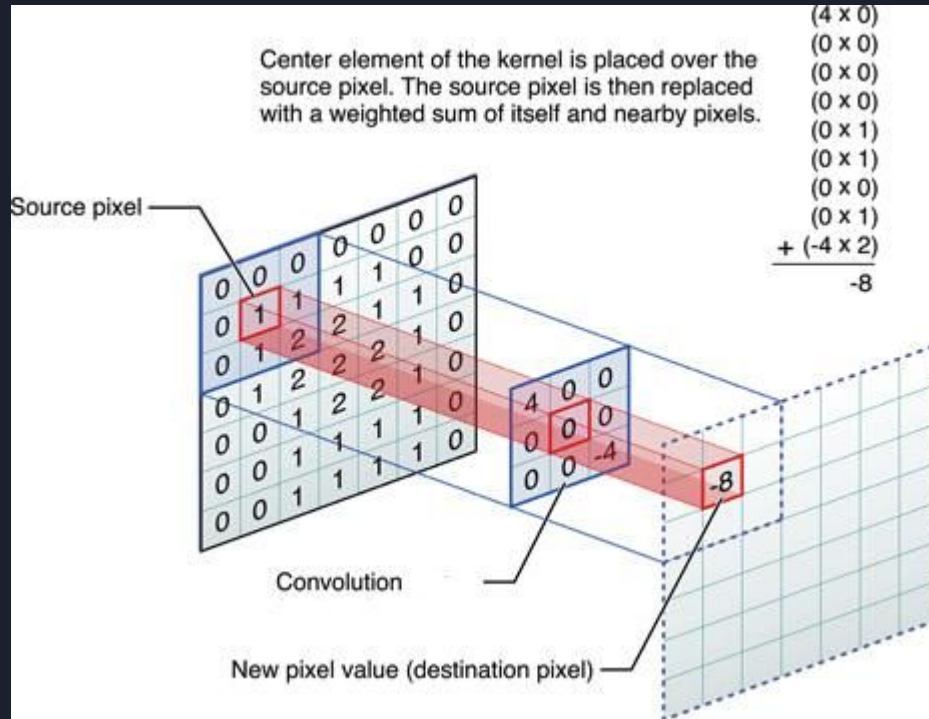
# LSTM vs GRU

Property	LSTM	GRU
Gating Mechanisms	Three gating mechanisms (input, forget, output gates) control information flow.	Two gating mechanisms (reset and update gates) streamline information processing.
Memory Cells	Utilizes a dedicated memory cell for storing and retrieving information.	Single hidden state combines functions of cell state and hidden state in LSTMs.
Complexity and Speed	Can be slower due to higher complexity.	Often faster to train due to simplicity.
Performance and Adaptability	Effective at capturing both short and long-term dependencies due to comprehensive understanding of sequences.	Excels in capturing short and moderate-term dependencies, suitable for quick adaptations to changing patterns.
Resource Efficiency	More resource-intensive.	Computationally more efficient, suitable for applications with limited resources.

# Convolutional Neural Networks (CNN)


- CNNs adapt to temporal sequences, capturing short-term load patterns through local feature extraction.
- Learns hierarchical features at different scales, enabling the identification of short-term dependencies in the load data.
- The convolutional layer captures local temporal patterns, while the pooling layer efficiently retains salient features, allowing a CNN to discern crucial short-term dependencies in load data.
- Adaptable to multivariate STLF, accommodating multiple input features for a comprehensive understanding of short-term load patterns.





# Convolution Operation





5	1	3	4
8	2	9	2
1	3	0	1
2	2	2	0

8	9
3	2

2 by 2 filter with stride = 2

# Max Pooling Operation

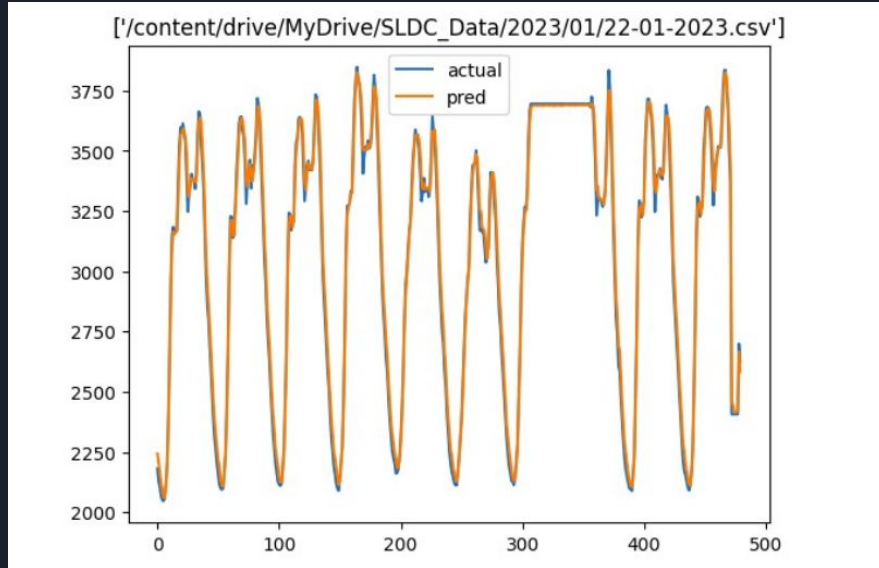


# Merits and Drawbacks of CNN

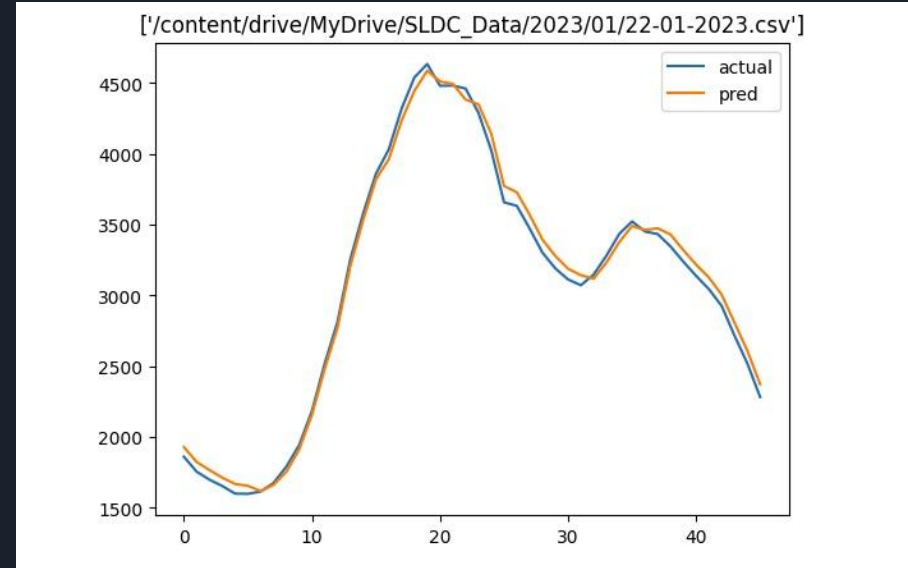
Merits	Drawbacks
<ul style="list-style-type: none"><li>• CNNs identify and extract important local features in the data, enhancing their ability to discern critical patterns.</li><li>• The architecture of CNNs is well-suited for capturing and modeling short-term dependencies, crucial in short-term load forecasting.</li><li>• CNNs are tailored to adapt and utilize temporal convolutions, enabling them to effectively process and learn temporal patterns in sequential data.</li><li>• CNNs can seamlessly handle multivariate input data.</li></ul>	<ul style="list-style-type: none"><li>• CNNs may have constraints in modeling long-range dependencies compared to dedicated sequential models like RNNs.</li><li>• The performance of CNNs in STLF can be influenced by the sensitivity of their outcomes to the choices made in hyperparameter tuning.</li><li>• The architecture of CNNs, especially with multiple layers, can introduce computational complexity and resource-intensive training.</li><li>• CNNs typically necessitate fixed-size inputs, potentially requiring preprocessing steps for handling variable-length sequences in STLF.</li></ul>

# Results

## Convolutional Neural Networks



Actual Load vs Predicted Load across a period of 50 days



Actual Load vs Predicted Load on 22-01-23



# Implementation Details

## Data Preprocessing:

For RNN, LSTM, GRU and CNN models :

- Data was made stationary by detrending using one lag differencing and was re-scaled to  $[-1, 1]$  scale.
- This helps in faster convergence and prevents features with relatively large magnitudes like previous year load demand to carry a larger weight during training.
- The model was trained on last 100 days of data, with each training data vector containing load of a specific time for last 5 days and the label was the load of the 6th day at the same corresponding time.

For ARIMA :

- Use last 5 months data at a frequency of 30 minutes instead of 5 minutes due to computational overload.
- Best hyper-parameters were searched using grid search method and used for model training.
- As the data is seasonal with seasonality and their of varying nature of trends in data given the season of year the optimized values of  $p, d, q$  were obtained accordingly.

# Model Architectures

## RNN

Model: "sequential"

Layer (type)	Output Shape	Param #
simple_rnn (SimpleRNN)	(None, 128)	16640
dense (Dense)	(None, 1)	129

=====  
Total params: 16769 (65.50 KB)  
Trainable params: 16769 (65.50 KB)  
Non-trainable params: 0 (0.00 Byte)

## GRU

Model: "sequential"

Layer (type)	Output Shape	Param #
gru (GRU)	(None, 4, 50)	7950
dropout (Dropout)	(None, 4, 50)	0
gru_1 (GRU)	(None, 4, 50)	15300
gru_2 (GRU)	(None, 4, 50)	15300
gru_3 (GRU)	(None, 50)	15300
dense (Dense)	(None, 1)	51

=====  
Total params: 53901 (210.55 KB)  
Trainable params: 53901 (210.55 KB)  
Non-trainable params: 0 (0.00 Byte)

# Model Architectures

## LSTM

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
lstm (LSTM)	(None, 4, 50)	10400
lstm_1 (LSTM)	(None, 50)	20200
dense (Dense)	(None, 25)	1275
dense_1 (Dense)	(None, 1)	26

=====

Total params: 31901 (124.61 KB)  
Trainable params: 31901 (124.61 KB)  
Non-trainable params: 0 (0.00 Byte)

## CNN

Model: "sequential\_2"

Layer (type)	Output Shape	Param #
=====		
conv1d_1 (Conv1D)	(None, 2, 64)	576
max_pooling1d_1 (MaxPooling1D)	(None, 1, 64)	0
flatten_1 (Flatten)	(None, 64)	0
dense_2 (Dense)	(None, 256)	16640
dense_3 (Dense)	(None, 1)	257

=====

Total params: 17473 (68.25 KB)  
Trainable params: 17473 (68.25 KB)  
Non-trainable params: 0 (0.00 Byte)



# Results of various algorithms implemented

Algorithm	RMSE	MAPE
SMA	195.66	5.21
WMA	101.61	3.16
SES	167.88	4.3
SARIMA	83.78	2.71
XGboost	206.77	5.82
FFNN	188.67	4.85



# Results of various algorithms implemented

Algorithm	RMSE	MAPE
RNN	105.66	3.1
GRU	93.61	2.76
LSTM	78.88	2.6
CNN	57.78	2.48





# Conclusion

- Load Forecasting is an important application of machine learning in electrical engineering which is significantly gaining importance due to increase complexity in transmission and load networks.
- Implemented and conducted literature survey of various state of art techniques of load forecasting
- Statistical techniques like SMA, WMA, ARIMA provide good performance in short term load forecasting but fail to map complex temporal patterns in load time series analysis.
- Deep Learning Technique like FFNN, LSTM provide a ray of hope in understanding these complex patterns and also account other factors like weather, humidity into forecasting of load.
- Various state of art techniques of time series forecasting like CNN, RNN have been in implemented in load forecasting which can provide great results in big volume of data.
- We aim to prototype these algorithms on real-time dataset of Delhi Load Dispatch Centre to benchmark and understand practical importance in load forecasting by incorporating other factors like weather.



# Acknowledgement

- Under guidance:

Ms. Sobhita Meher, Assistant Professor, Dept of Electrical Engineering, IIT BHU, Varanasi

- References:

- Dataset - <https://www.delhisldc.org/>
- <https://arxiv.org/pdf/1906.04818.pdf>
- <https://paperswithcode.com/task/load-forecasting>
- <https://arxiv.org/pdf/2007.04517.pdf>
- <https://arxiv.org/pdf/2208.00728v1.pdf>
- <https://arxiv.org/abs/1906.04818>
- <https://caciitg.com/resources/tsa/>
- <https://paperswithcode.com/paper/es-drnn-a-hybrid-exponential-smoothing-and>
- <https://paperswithcode.com/paper/optimal-adaptive-prediction-intervals-for>
- <https://machine-learning.paperspace.com/wiki/convolutional-neural-network-cnn>
- <https://medium.com/analytics-vidhya/from-zero-to-hero-with-convolutional-neural-networks-d30a5e63a664>
- <https://medium.com/@prateekgaurav/nlp-zero-to-hero-part-2-vanilla-rnn-lstm-gru-bi-directional-lstm-77fd60fc0b44>



Thank You!!