Time Series Analysis with Non-Linear Models for Energy Consumption Prediction

A Project Work Submitted for the course

Machine Learning
(CSL7620)

INDIAN INSTITUTE OF TECHNOLOGY, JODHPUR
Department of Computer Science
Session 2024-26

Submitted by:

Shashank Mishra (M24MAC011)

Md Moin Munir Ahmed (M24MAC004)

Vimal Kumar Verma (M24MAC015)

ABSTRACT

This shall aid in the development of precise models in the field of load forecasting using ML techniques and deep learning. In other words, it shall evolve its models capable of making electricity load demand predictions in a given area by using historical data of seasonal patterns. This will include not only some of the classic ML algorithms like linear regression, decision tree regression, random forest regression, and XGB Regressor but also deep learning models based on LSTM networks. The project also includes an analysis of the performance of the models on different metrics such as the R2 score, Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). The dataset for the project has been gathered from open-source and will be pre-processed before training as well as testing the models. The project further discusses and analyses how different feature engineering and hyper parameters tuning on model performance. The outcome of the project will be a strong, accurate load forecasting model using the powerful techniques of ML and deep learning that will help power grid operators in better resources management while ensuring stable and efficient supply of power to customers.

Table of Contents

Chapter-1
Load Demand Forecasting

1. Introduction -

This is very critical for utility companies, since it allows decisions on purchasing power, switching loads, and planning for any new infrastructure. The quality of the load forecast is used by energy suppliers, ISOs, financial institutions, and all the players in power generation, transmission, and distribution. There are, however, three types of forecasts: short-term (1 hour to 1 week), medium-term (1 week to 1 year), and long-term (over 1 year). Every forecast has its utility functions, such as the importance of short-term forecasts for the everyday running of operations and preventing overloads, while long-term forecasts are the ones that can guide capital spending in deregulated markets.

forecasting methods vary from simple statistical methods to AI algorithms- regression, neural networks, fuzzy logic, and more - while for medium and long-term forecast, there are common end-use and econometric approaches while, in most cases, short-term methods use time series, similar day approaches, and machine learning methods. Besides load predictions, it is also useful with weather normalization because historical weather data forms its basis, which usually lasts from 25 to 30 years. This makes load forecasting accuracy all the more essential in the recent past, especially in deregulated power markets, where price volatility and changes in demand increase manifold.

Forecasting Period and its Applications

Time Horizon	**Functional requirement**
Long Term	
5 to 20 years	Prospective planning of the power system
1 to 5 years	Power system planning, system strengthening, tariff planning, sale/purchase of generation capacity
Medium Term	
One year	Annual plans for maintenance scheduling of generating units and transmission lines, scheduling of fuel supplies, financial planning, bilateral agreements for capacity trading.
Quarterly	Outage planning, hydro generation schedules, planning of scheduled load shedding, study of generation and transmission constraints
Monthly	Evolving operating plans, hydro generation schedules, outage planning
Short Term	
Weekly	Unit commitment, operational planning, resource planning, outage clearance, fuel allocations, bilateral agreements for energy trading.
Daily	Scheduling of generation, trading of energy, dispatch of generation, EMS functions.

Four minutes to few hours	Security assessment, economic dispatch, other MIS functions.

Table 1.1

2. <u>Importance of Load Forecasting-</u>

The electrical load forecast predicts the future electricity demand of a region. In turn, this is very valuable for utility companies, grid operators, and power system planners. Therefore, error-free electrical load forecasting will eventually contribute to the stability, reliability, and cost efficiency of the power systems involved.

Load forecasting helps utilities optimize the resource management so that they can better plan for generation, transmission, and distribution; avoid inefficient investment in infrastructure; and keep infrastructure healthy. Second, it supports the stability of the grid because operators would now be able to balance supply to the demand, help them avoid having a blackout, and the like.

Load forecasting also permits seamless integration of renewables. The ability to predict different kinds of demand fluctuations enables grid operators to better manage the intermittent sources, such as wind and solar, which leads to the eventual decrease in fossil fuels and emissions. Lastly, the proper load forecasting decreases electricity costs for the consumers because this would enable utilities to implement demand-side management programs such as time-of-use pricing to shift the usage to off-peak hours more efficiently.

3. <u>Important Factors for Forecasting</u>

Major influences that are considered in short-term load forecasting include time, weather, and customer segments. Some of the influences connected to time include season, day of the week, and hour whereby weekends, holidays, and days before weekends show different load patterns. Weather, particularly temperature and humidity, plays a significant role, and most of the indices that are known to account for seasonal effects include the Temperature-Humidity Index (THI) and Wind Chill Index (WCI).

Most electric utilities serve customers of different types such as residential, commercial, and industrial. The electric usage pattern is different for customers that belong to different classes but is somewhat alike for customers within each class. Therefore, most utilities distinguish load behaviour on a class-by-class basis

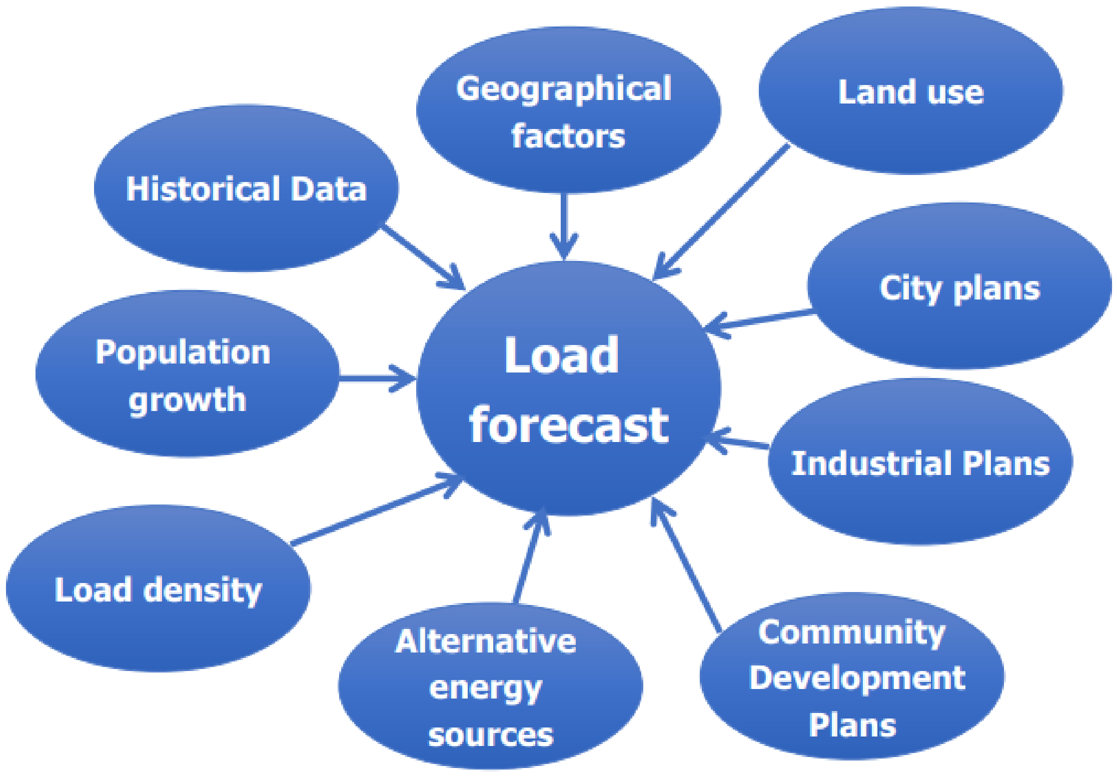


Fig 1.1 Factors influencing load forecasting

4. <u>Challenges in Load Forecasting-</u>

4.1 Forecasting is a function of the factors like the weather. However, in some instances the weather turns out to be unreliable, and hence the forecasting may end up otherwise if the actual weather is different from the predicted one. Furthermore, the regions are different in terms of weather conditions, and that definitely will impact on electricity demand. Particularly it will have a negative effect on revenues if the utility generates more to cover an anticipated high demand and finds that the consumption is much lower than what was generated either using expensive methods such as fossil fuel generators, etc. [3]

4.2 The majority of the experienced utility forecasters use manual methods that rely on comprehensive understanding of a wide range of contributing factors based on particular data or events that are likely to occur. Manual forecasting cannot be sustainable since the number of forecast is growing and becoming complex day by day. Thus, utilities ought to find technologies that can deliver precise results and shun problems that are most likely to occur in case the experienced forecasters retire or leave employment.

4.3 Usage behavior differs between consumers using different types of meters and especially between the smart and traditional meters as well as different tariffs. Utility must understand this and prepare separate forecast model for each of the metering systems and then add them up for the final forecast value otherwise they would receive inaccurate forecasting.

4.4 It is very hard to get accurate data for the consumption behavior as it changes depending on variables like price and the corresponding demand based on such a price change.
4.5 It is also very challenging to carry out the load forecasting task because of the complex loads which may vary depending on the seasons and the total consumption for two similar seasons may vary.
4.6 At times, it is challenging to fit numerous complex factors that affect the demand for electricity into the models of forecasting. Additionally, sometimes, achieving an accurate demand forecast using certain parameters such as change in temperature, humidity, among others affecting consumption may be difficult.
4.7 If the utility does not comprehend nor determine an appropriate margin of error when using short-term load forecasting, they will likely incur losses. [4]

Chapter-2

Technologies Used

1. **Python**

   Originally released in 1991, Python is an easy language to use, readable, and ready for doing all web development, data analysis, AI, scientific computing, automation, and more. It has many libraries including NumPy, Pandas, SciPy, Matplotlib, TensorFlow, and Scikit-Learn that let developers get things done efficiently. Python is an interpreted language, so it executes code quickly without compilation, which makes it great for prototyping and fast development. Additionally, cross-platform compatibility makes it run on Windows, Linux, and macOS.

2. **Google Colab**

   Google Colab, or Google Colaboratory, is a free cloud environment to write, run, and share Jupyter notebooks in a web browser. It allows free usage of GPUs and TPUs; thus, many researchers and scientists develop deep learning-based models and other computationally expensive applications on Colab. Libraries such as NumPy, Pandas, and TensorFlow are installed natively in Colab. Colab is integrated with Google Drive, making it easy to store, access, and even collaborate on projects from anywhere one has internet access.

3. **Models Used**

   3.1. Linear Regression-

   Linear regression is a statistical technique for modelling the relationship between the dependent variable (y) and one or more independent variables (x), which is one of the most widely used methods in predictive analytics on continuous outcomes. Examples of continuous outcomes include sales, salaries, or product prices. The linear relationship is established by the intercept to denote the expected value of y if x is zero. The slope shows how much y changes for a one-unit increase in x. Since it assumes this linear relationship, the model is very popular in such fields like finance or economics and, even some areas of machine learning, where it makes its predictions purely based on variable relationships. The mathematical representation of Fig 2.1 is given as:

   $$y = b_0 + b_1 x + \epsilon$$

   Here,

   - y is the dependent variable
   - x is independent variable
   - b_0 is y intercept
   - b_1 is the slope and
   - ϵ is the statistical error

   Consider the below image in Fig 2.1

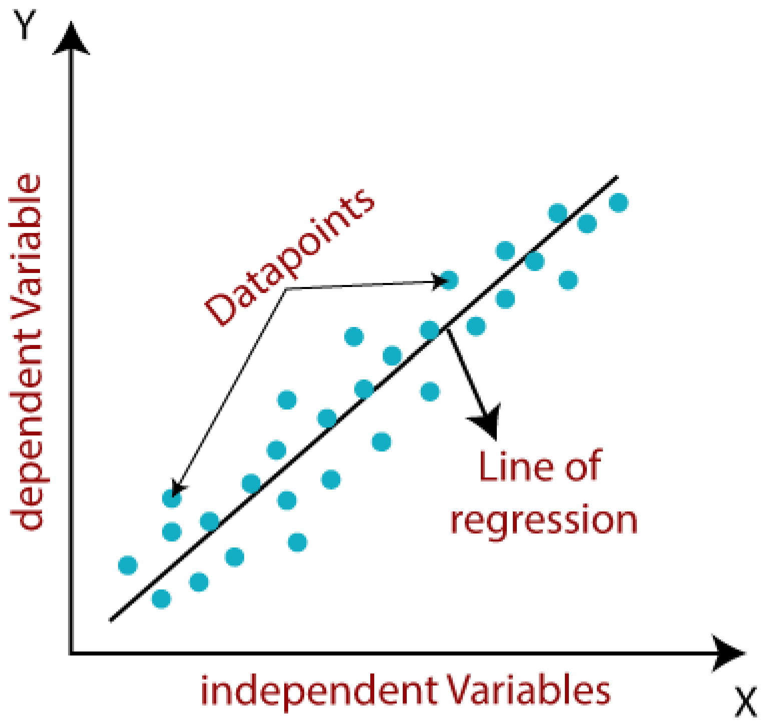


Fig 2.1: Linear Regression Simulation

The best-fit line is determined by minimizing the sum of the squared differences between the observed values of the dependent variable and the predicted values based on the independent variable. This method is known as the least squares method.
The mathematical model for a multiple linear regression with two or more independent variables can be expressed as:

$$y = \beta_0 + \beta_1 x_1 + \ \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$$

Where y is the dependent variable, x_1, x_2,…, x_n are the independent variables, β_0, β_1, β_2, …, β_n are the intercept and slope coefficients, respectively, and ε is the error term.

In multiple linear regression, the goal is to find the best-fit plane that describes the relationship between the variables. The plane is determined by minimizing the sum of the squared differences between the observed values of the dependent variable and the predicted values based on the independent variables.

<u>Advantages of Linear Regression</u>

1. Simple Linear Relationship: Good performance of linear regression on simple linearly separable data; easily depicts the relationship between variables.
2. Simple: Convenient to implement, interpret and train on in an effective manner.
3. Mitigation of Overfitting: Linear regression can overfit; however, the problem can be mitigated using dimensionality reduction, regularization, L1, L2, and cross-validation.

Disadvantages of Linear Regression

1. Linearity Assumption: It assumes linearity between the independent and dependent variables, which under normal circumstances rarely occurs in reality data due to a nonlinear relationship.
2. Sensitivity to overfitting and noise: If it is observed that the number of observations is less than the number of features, linear regression becomes sensitive to overfitting because it is likely to consider noise to be a part of the model.
3. Outlier Sensitivity: The model is highly sensitive to outliers, so one should ideally be very careful about detecting outliers and correcting them before doing linear regression.

3.2. Decision Tree Regression

Decision tree regression is a form of regression technique that uses the decision tree to model relationships between a dependent variable and one, two, or more independent variables, with an allowance for non-linear relationships and interactions. Unlike standard linear regression, it does not assume that linearity exists between variables. A decision tree is a hierarchical model with nodes and branches; nodes represent features and branches denote decision splits based on those features. Internal nodes conduct tests on specific variables, and each of the leaf nodes corresponds to the result of that prediction. This is one of the most popular techniques used in regression as well as classification tasks because of interpretability and versatility in supervised learning.

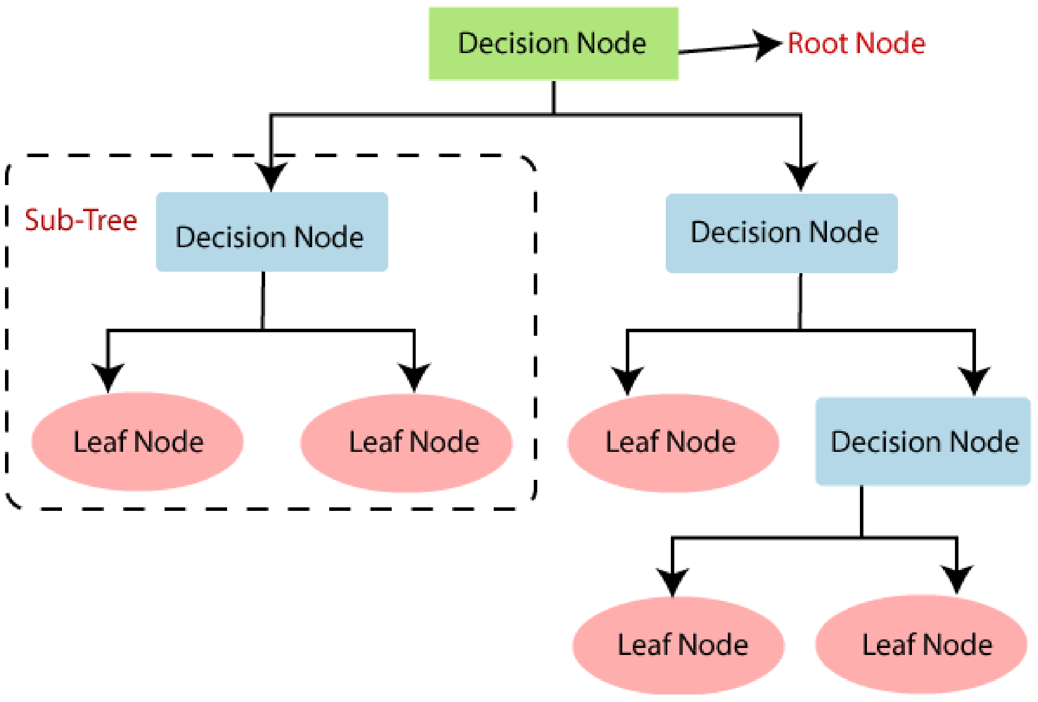


Fig 2.2: Decision Tree Algorithm

A decision tree is a three-way model with the structure of a tree. Among all, it comprises three major nodes: the whole dataset itself as the root node; splitting further from that makes for interior nodes because of the features' indication along with branches being

the decision rules; while the leaf nodes indicate final outcomes. This structure comes in very handy in solving decision-making problems.
A prediction is made by traversing the tree, following True/False decisions from each node to a terminal node because the final prediction is based on the average value of the dependent variable within that leaf. Repeated iterations improve a tree's predictions for each data point.

<u>Working of Decision Tree :</u>
So, the algorithm works in a decision tree when predicting class: starting from the root node, match its attribute value with record's attribute and proceed along the matching branch at each node up to the leaf node where the final classification is actually outputted.
The procedure for constructing a decision tree can be summarized as follows:

1. Start with the root node, which contains all data set.
2. Determine the best attribute in the given data set using Attribute Selection Measure (ASM).
3. Partition the data set into distinct subsets based on the possible values of the most discriminatory attribute.
4. Make a decision tree node with this attribute. Systematically extend new branches from each subset produced in Step 3, repeat the process until further classification is impossible, and label each terminal node as a leaf node.

<u>Advantages;</u>

1. This concept is thus understandable because it conveys the process people apply in practical choices.
2. It is highly effective for solving decision-oriented problems.
3. It encourages thinking about all possible outcomes of any given problem.
4. It requires less data cleaning compared to many other algorithms. [6]

<u>Disadvantages:</u>

1. The decision tree is too layered, thereby complex.
2 It may have overfitting problem that could be optimized using the Random Forest algorithm.
3. Additional class labels can increase the computational complexity of the decision tree.

3.3. Random Forest Regression

Random Forest is an ensemble algorithm for both classification and regression used for supervised machine learning. More than one decision tree is combined with their training on different subsets of the same dataset with an aim to improve the model's accuracy and robustness rather than solely depending on a single tree. All predictions generated from all the trees are summed up, while in the case of classification, majority voting, and in regression, a form of averaging, are used to generate the output. For a larger number of trees, the accuracy will also get better as well as reduce overfitting.

The below diagram explains the working of the Random Forest algorithm:

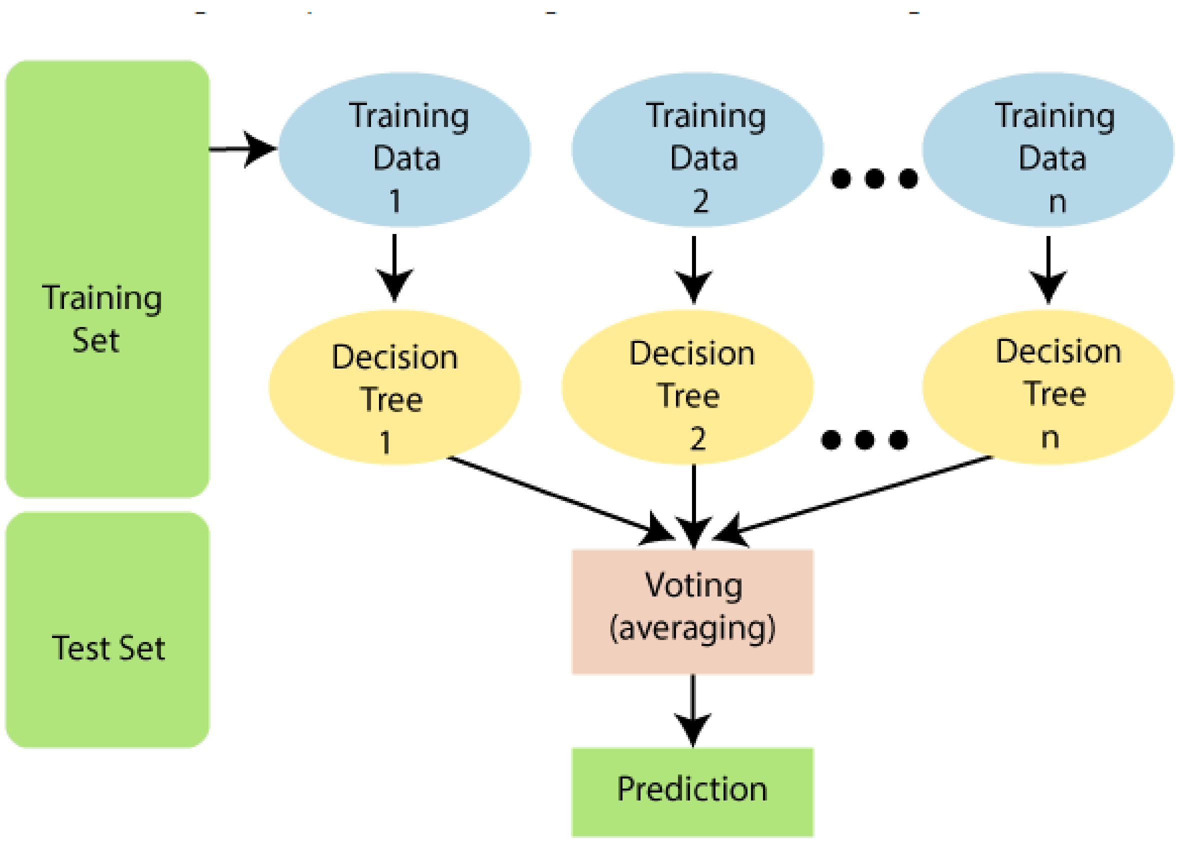


Fig 2.3: Random Forest Algorithm

Features

1. This algorithm is more precise than the decision tree algorithm.
2. It handles missing data really well.
3. It doesn't need hyperparameter tuning to be capable of making accurate predictions.
4. This reduces the case of overfitting commonly observed in the decision trees.
5. At each node, the tree selects a random subset of features so that, at every split, there will be diversity in the forest.

Working of Random Forest Method:

Random Forest operates in two phases. First, it builds the forest by combining multiple decision trees created in the previous step. The forecast then is obtained from the trees built in the first phase.

The working process can therefore be described to include these steps:

1. Pick some random K data points from the training set.
2. Construct decision trees from the chosen data points (subgroups).
3. Select the number N of decision trees to build.
4. Repeat steps 1 and 2 until N trees are constructed.

Advantages

1. Random Forest can handle problems of both classification and regression.
2. This version is able to handle a high-dimensional large dataset.
3. It improves the model's accuracy and eliminates the problem of overfitting.

Disadvantages

1. Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

3.4. XGB Regression

XGB Regression stands for Extreme Gradient Boosting Regression. Extreme Gradient Boosting (XGBoost) is an open-source library that provides an efficient and effective implementation of the gradient boosting algorithm.

Gradient boosting is an ensemble machine learning algorithm with the performance of classification and regression tasks. Unlike bagging, where multiple independent trees are built in parallel, gradient boosting builds trees sequentially such that each new tree corrects errors from the previous one, which is known as boosting. The hyperparameters are max_depth, n_estimators, and learning_rate, which determines how much the contributions of all trees summed up contribute to the overall result, which typically lies in the 0 to 1 range.

The trees are added one at a time in gradient boosting, with each learning from the previously added trees in order to minimize the errors they make in their predictions. It uses a differentiable loss function and gradient descent. As more trees are added, the loss gradient is minimized, and that's why this algorithm is called "gradient boosting."

Features

1. **Execution and Speed:** XGBoost, implemented in C++, is optimized for fast execution and is often at least comparable in this regard to other ensemble methods.
2. **Parallelization:** The core XGBoost algorithm is parallelizable and uses multiple cores, GPUs, and even distributed networks to be proficient in processing big data.
3. **High Performance:** XGBoost outperforms all other algorithms on a large proportion of the AI benchmark datasets.
4. **Adjustable tuning parameters:** This also allows a user to choose how to tune the best set of parameters while offering much functionality regarding the tuning of parameters in support of regularization, cross-validation, and customized objectives.

Functioning

1. With the goal to minimize the loss function with each step, it should improve predictions.

2. A weak learner is used to make incrementally incremented predictions.
3. Decision Trees find the splitting points using metrics like Gini Impurity for better loss function minimization.
4. An additive model combines weak models and learns through successive approximations of the loss function.
5. Add trees sequentially without modifying any previously existing trees. Gradient descent is often exploited to tune hyperparameters and update weights appropriately.

Advantages:
1. High accuracy: XGBoost is known for its high accuracy and is often used in data science competitions.
2. Speed: XGBoost is designed to be fast and scalable, making it suitable for large datasets.
3. Feature selection: XGBoost has built-in feature selection capabilities, which can help to identify the most important features in your data.
4. Regularization: XGBoost includes regularization techniques such as L1 and L2 regularization, which can help to prevent overfitting.

Disadvantages:
1. Complexity: XGBoost is a complex algorithm with many hyperparameters to tune. This can make it difficult to get the best performance from the algorithm.
2. Black-box model: XGBoost is a black-box model, meaning that it can be difficult to understand how it is making its predictions.
3. Memory usage: XGBoost can be memory-intensive, particularly when working with large datasets or many features.
4. Interpretability: Because XGBoost is a black-box model, it can be difficult to interpret the results and understand how the algorithm arrived at its predictions.

3.5. LSTM Deep Learning Model

Long short-term memory (LSTM) is a type of recurrent neural network architecture in deep learning, designed to store information over long periods of time. Unlike the standard RNNs, LSTMs have "memory cells" which remember data across time steps and three gates - input, forget, and output which manage decisions about what information to allow through. LSTMs are applied widely to speech recognition, language modeling, and machine translation tasks as well as broader sequence learning tasks: activity recognition and music transcription.

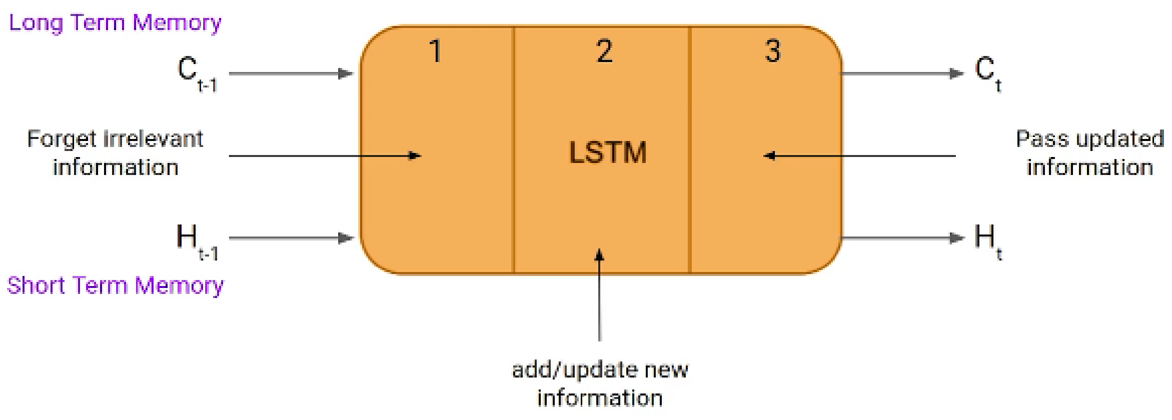


Fig 2.4: LSTM Network Architecture

The architecture of the LSTM model is depicted as in Fig 2.4, comprising three components: the Forget gate determines whether to forget or not to forget information that has already come in, the Input gate learns the new information based on what's coming in during that time step, and finally, the Output gate controls and passes updated information to the next timestamp. An LSTM unit, just like a layer in a feedforward neural network, consists of two together, with each neuron having a hidden state and current state.

Forget Gate

So, we should decide inside a cell of the LSTM neural network whether we need to remember something from the previous time step or forget it. The equation for the forget gate is as follows:

$$f_t = \sigma(X_t * U_f + H_{t-1} * W_f)$$

Let’s try to understand the equations, here

- X_t: Input to the current time stamp
- U_f: Weight associated with input
- H_{t-1}: Hidden state of the previous timestamp
- W_f: Weight matrix associated with the hidden state

Then, a sigmoid function is applied. It will make f_t a number between 0 and 1. This f_t is later multiplied with the cell state of the previous timestamp, as shown below.

$$C_{t-1} * f_t = \begin{cases} 0, & if\ f_t = 0\ (forget\ everything) \\ C_{t-1}, & if\ f_t = 1\ (forget\ nothing) \end{cases}$$

Input gate

The input gate is used to quantify the importance of the new information carried by the input. Here is equation of the input gate.

$$i_t = \sigma(X_t * U_i + H_{t-1} * W_i)$$

Here,

- X_t: Input to the current time stamp
- U_i: Weight matrix of input
- H_{t-1}: A hidden state of the previous timestamp
- W_i: Weight matrix of input associated with hidden state

Again we have applied the sigmoid function over it. As a result, the value of I at timestamp t will be between 0 and 1.

<u>New information</u>

The equation of new information is given by:

$$N_t = \tanh(X_t * U_c + H_{t-1} * W_c)$$

It is a function of the hidden state at the previous timestamp t-1, and input at timestamp t. It employs the activation function tanh in such a way as to ensure that the value of the new information Nt always falls in the range -1 to +1. Therefore, if Nt is negative, this new information is subtracted from the cell state; if positive, it is added. However, Nt is not added to the cell state directly, and the new equation is as follows:

$$C_t = f_t * C_{t-1} + i_t * N_t$$

Here, C_{t-1} is the cell state at the current timestamp, and the others are the values we have calculated previously.

Output gate

Here is the equation of the Output gate, which is pretty similar to the two previous gates.

$$O_t = \sigma(X_t * U_O + H_{t-1} * W_O)$$

This sigmoid function will also ensure a value between 0 and 1. So now, in order to compute the current hidden state, we are going to use Ot and tanh of the updated cell state. As depicted below.

$$H_t = O_t * \tanh(C_t)$$

Amazingly enough, the hidden state depends on long term memory (Ct) and the current output. Suppose you want to use the output of the current timestamp; just apply the SoftMax activation on the hidden state Ht.

$$Output = Softmax(H_t)$$

Below in Fig 2.5 is given a more detailed diagram of the LSTM network.

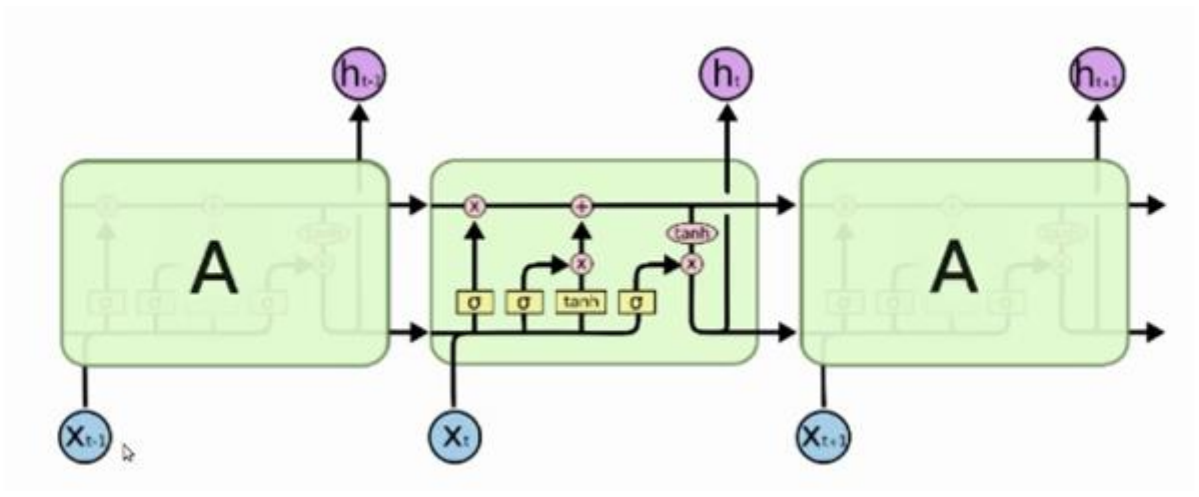


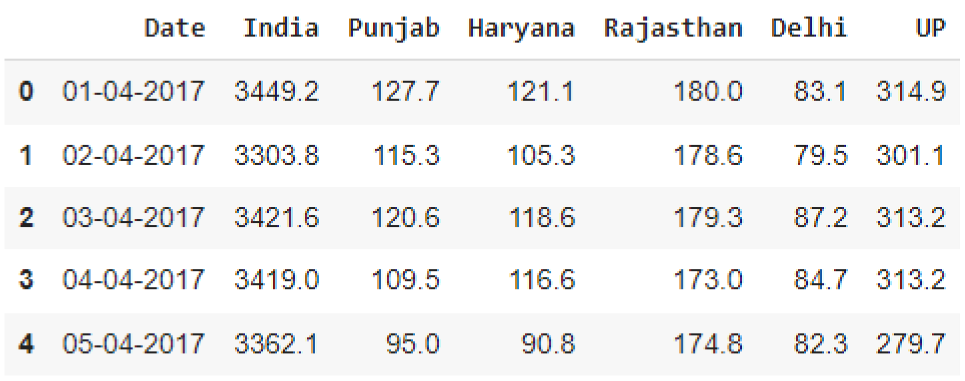
Fig 2.5: Detailed diagram LSTM Network

Chapter-3

Data Pre-processing

1. **Training and Testing Data**

The dataset contains daily Energy Consumption (in MUs) for All Over India and five states of Northern Region (Punjab, Haryana, Rajasthan, Delhi, Uttar Pradesh) form year 2017-22. All the data was taken from Grid Controller of India's website and is available in public domain.



	Date	India	Punjab	Haryana	Rajasthan	Delhi	UP
0	01-04-2017	3449.2	127.7	121.1	180.0	83.1	314.9
1	02-04-2017	3303.8	115.3	105.3	178.6	79.5	301.1
2	03-04-2017	3421.6	120.6	118.6	179.3	87.2	313.2
3	04-04-2017	3419.0	109.5	116.6	173.0	84.7	313.2
4	05-04-2017	3362.1	95.0	90.8	174.8	82.3	279.7

Fig 4.1: Daily Load Consumption Data (in MUs)

Data used in training was of split 0.8 while testing data is 0.2 times the size of original data. Feature Engineering was used to make models more reliable and following concepts were introduced during data pre-processing.

2. **Feature Engineering**

Feature engineering is used to improve the reliability of the model. The date column is broken down into year, month, and day-of-week columns. Autocorrelation and partial correlation functions are used to incorporate lag features.

Lag measures the time span after the occurrence of a particular event before it impacts another variable. This helps capture relationships in variables over time, which may lead to more accurate predictions and decisions.
Autocorrelation quantifies the relationship of a variable with its past values, generally including a relationship between historical observations and current measurements.
Partial correlation assessing the relationship of two variables that may depend upon control variables. Example: Ice cream sales given temperature influences of past days.
Seasonal decomposition separates a time series into its seasonal, trend, and residual components to better understand underlying patterns and isolate the seasonal aspect for further analysis.

The process of seasonal decomposition involves fitting a model to the time series that separates it into its three components:

1. **Seasonal component:** This means the pattern may repeat in the data at certain points in time, perhaps at day, week and yearly periodicities. Often modeled using seasonal moving averages or SARIMA models.
2. **Trend component:** This is a long-term and sometimes cyclical behavior in- and out-of-data, usually modelled by linear regression or exponential smoothing.
3. **Residual component:** It is that portion of the system noise or variation that the seasonal or trend components could not explain. Often, it is regarded as white noise.

Once the seasonal, trend, and residual components have been separated, each component can be analysed separately to gain insights into the patterns and relationships in the data. Seasonal decomposition can also be used to remove the seasonal component from the data, making it easier to analyse the remaining trend and residual components. The period of 365 days was used in seasonal decomposition

3. Data Pre-processing

3.1. <u>All India Data set:</u>

The All India dataset contains the daily data set for total electricity use of all the states and union territories of India for five years, 2017 to 2022. Seasonal decomposition was done using the seasonal_decompose function. Results were shown by using libraries of Matplotlib and Seaborn, as shown in Figure 3.2. Electricity demand reflects the social standards, economic growth, geographical diversity, and demographic factors of a country. Electricity use can be forecasted even though there are several factors involved, such as weather conditions, location, sunrise, sunset, and a change of seasons.

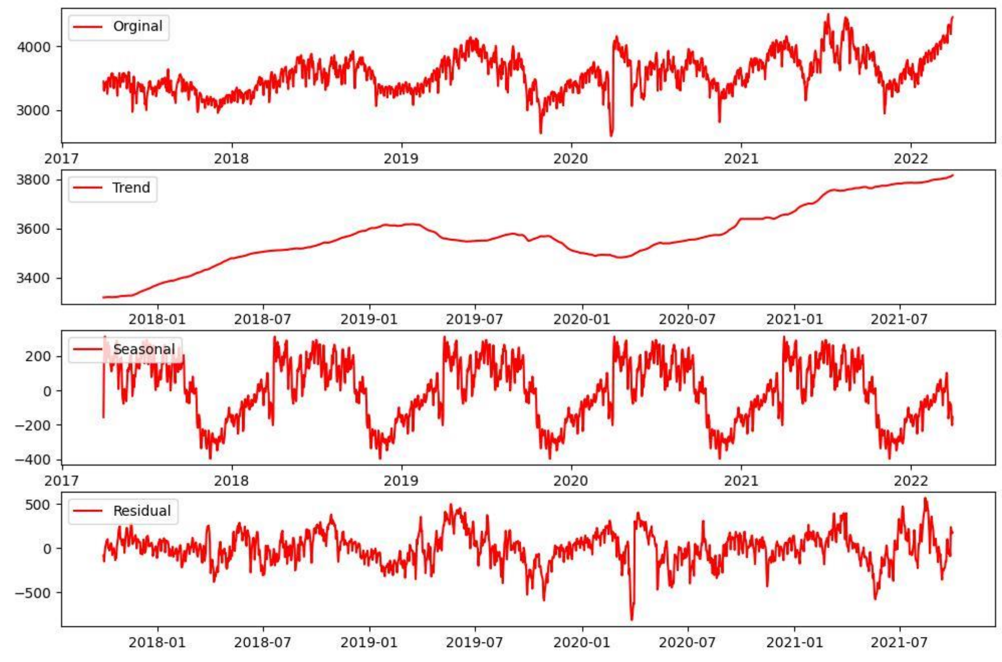


Fig 3.2: Seasonal Decomposition of All India Data Set

The time series data indicates changes. Electricity is used least often in December and most often during the summer months. This economy growth of India would reflect further in the consumption of electricity as the increasing trend in the data depicts. Figures 3.3 and 3.4 are autocorrelation and partial correlation plots. These plots describe how the load is correlated at various lags of time and what impact other variables have caused.

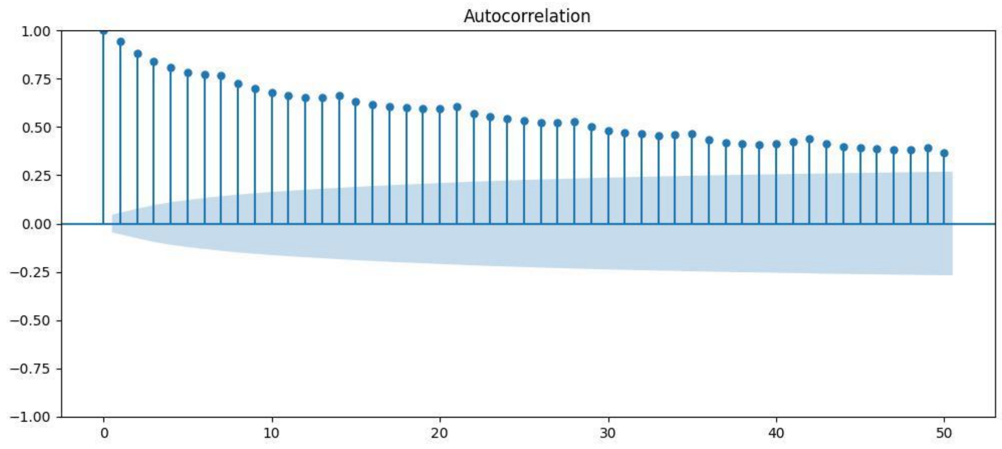


Fig 3.3: Auto correlation Plot of All India Data set

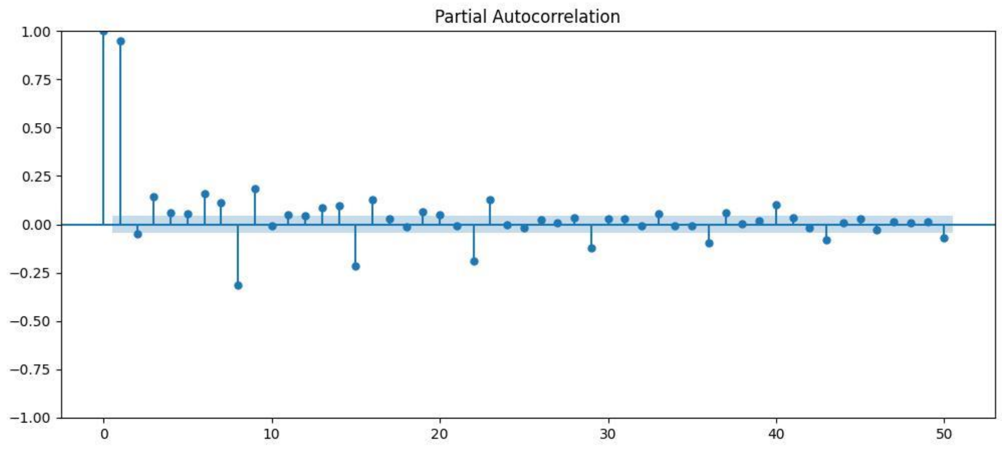


Fig 3.4: Partial correlation Plot of All India Data set

The day 1 and day 8 are showing strong correlation with the data. Hence, the adding two lag columns with 1 and 8 days shift respectively will make the model more reliable.

3.2. Punjab Data set:

The Punjab data set contains the daily total consumption by the state of Punjab. The data is for the period of five years from 2017 to 2022. The data set was loaded using pandas and seasonality test was performed using the seasonal decompose function. The results are displayed in Fig 3.5.

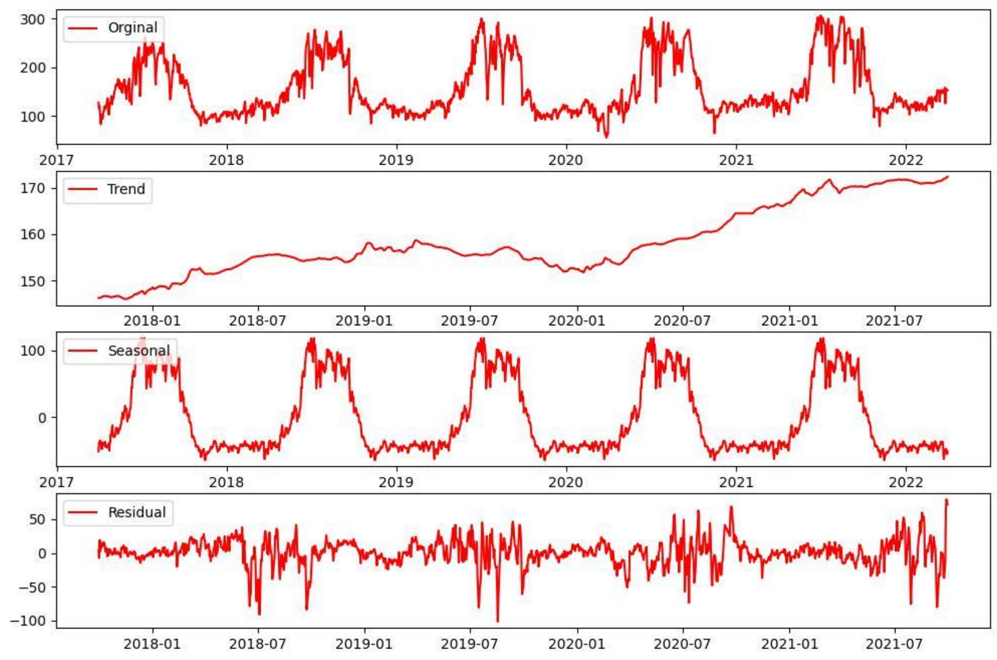


Fig 3.5: Seasonal Decomposition of Punjab Data Set

Punjab is an agricultural state, and hence, the load is consistently high during the summer months. Like the all-India dataset, the load is at its minimum in December and at the peak during the summer. Autocorrelation and partial correlation are plotted in Figures 3.6 and 3.7.

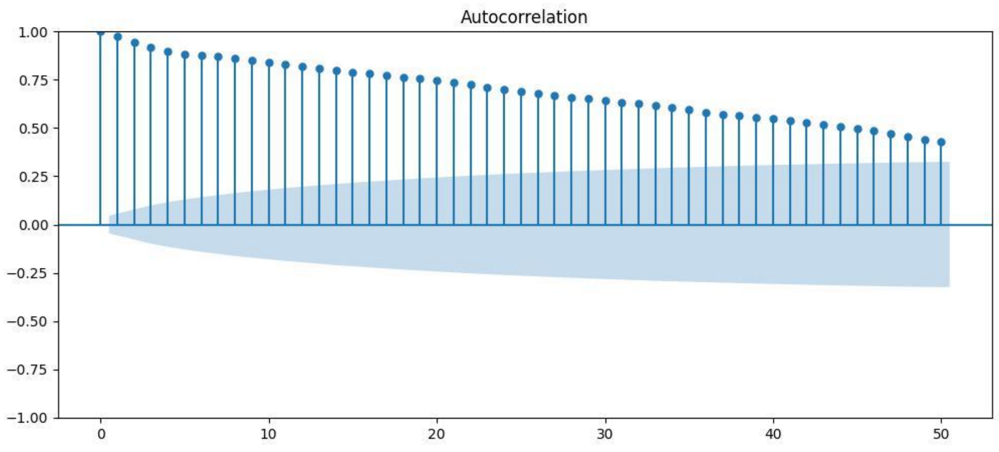


Fig 3.6: Auto correlation Plot of Punjab Data set

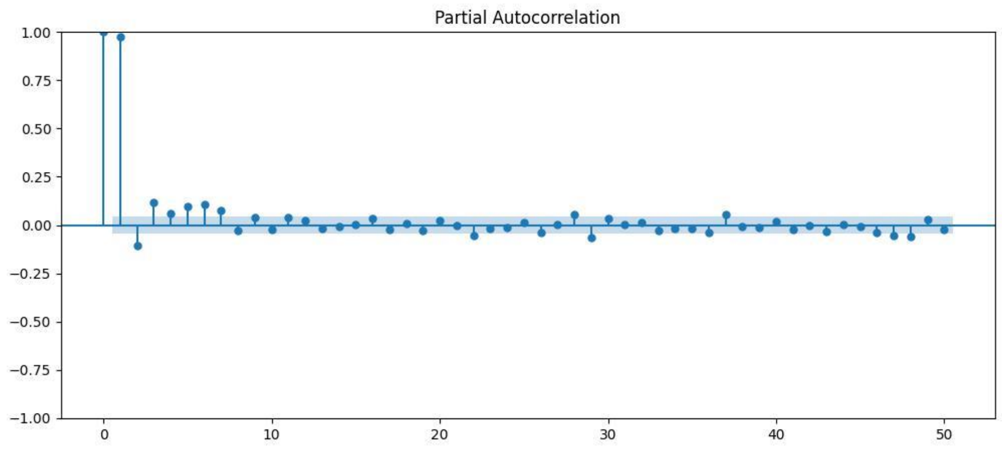


Fig 3.7: Partial correlation Plot of Punjab Data set

Only day 1 is showing strong correlation with the data. Hence, the adding one lag columns with 1 day shift will make the model more reliable.

3.3. Haryana Data set:

The Haryana data set contains the daily total consumption by the state of Haryana. The data is for the period of five years from 2017 to 2022. The data set was loaded using pandas and seasonality test was performed using the seasonal decompose function. The results are displayed in Fig 3.8.

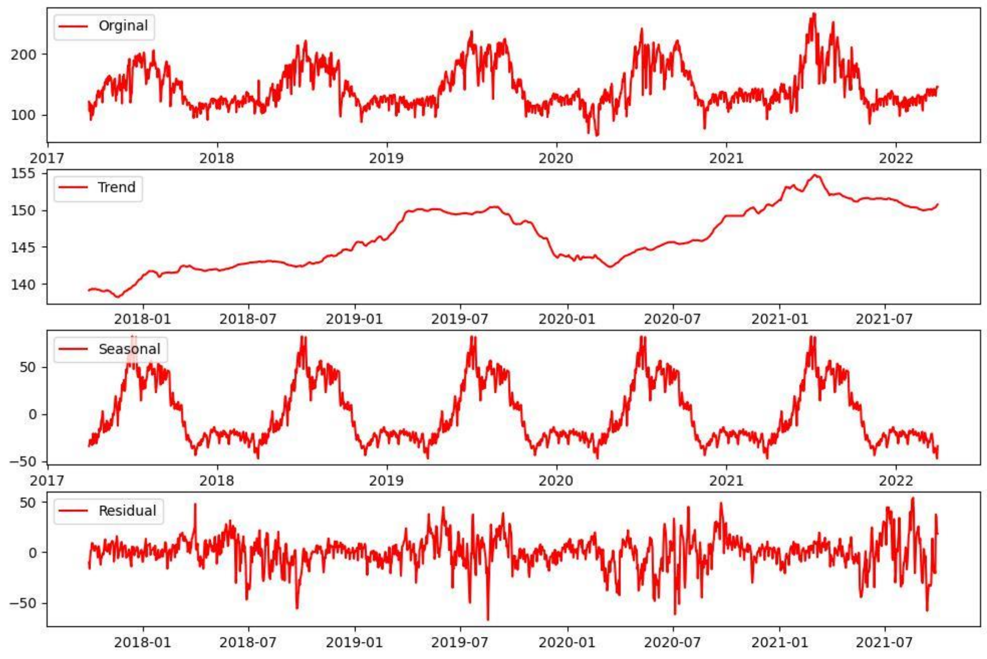


Fig 3.8: Seasonal Decomposition of Haryana Data Set

As Haryana is also an agricultural state like Punjab with heavy dependency on agriculture one could see consistent high demand during the summer months similar to Punjab. The data is seasonal with load at its all year low during the month of December and recording all year high around the summer season. The data follows the similar trend as all India data set. Auto and partial correlation are plotted in Fig 3.9 and Fig 3.10.

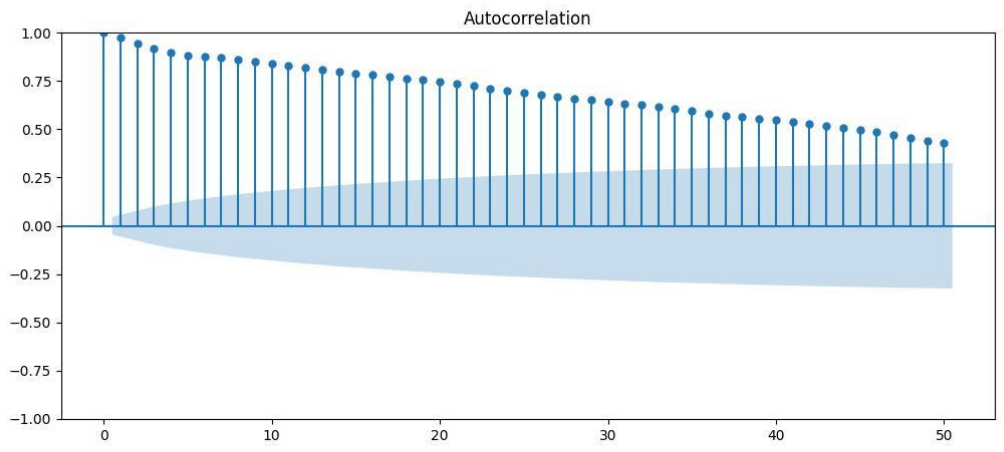


Fig 3.9: Auto correlation Plot of Haryana Data set

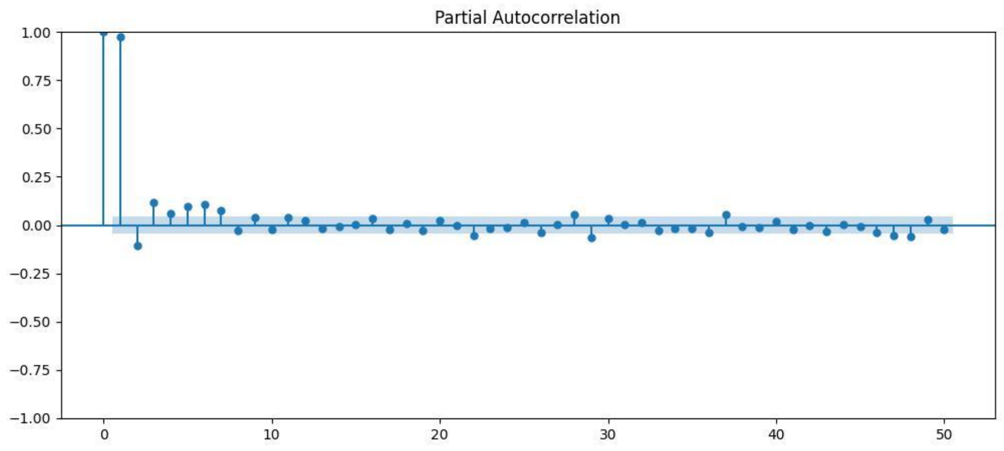


Fig 3.10: Partial correlation Plot of Haryana Data set

Only day 1 is showing strong correlation with the data. Hence, the adding one lag columns with 1 day shift will make the model more reliable.

3.4. Rajasthan Data set:

The Rajasthan data set consists of the daily total consumption by state of Rajasthan. The data pertains to five years from 2017 up to 2022. Data set was loaded using pandas and seasonality test was performed by making use of the seasonal decompose function. The result is depicted in Fig 3.11.

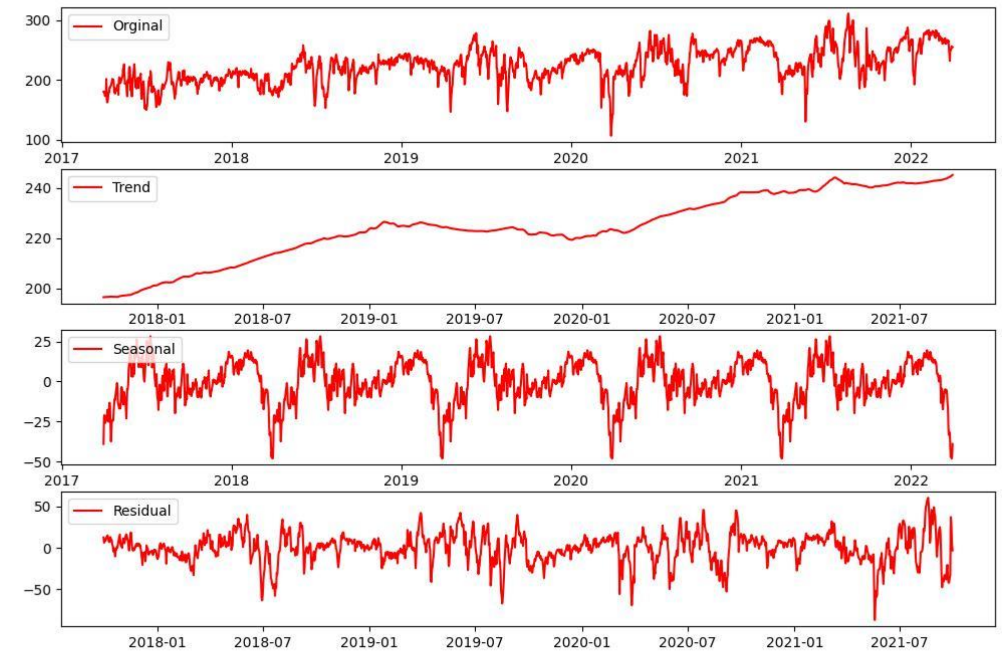


Fig 3.11: Seasonal Decomposition of Rajasthan Data Set

Since the climate in Rajasthan is normally hot and dry, its peak demand for electricity occurs practically throughout the year barring winters when the demand is relatively low. The seasonality is obvious from the data as well; the load is lowest in December and highest during the summer months. Figures 3.12 and 3.13 present autocorrelation and partial correlation plots.

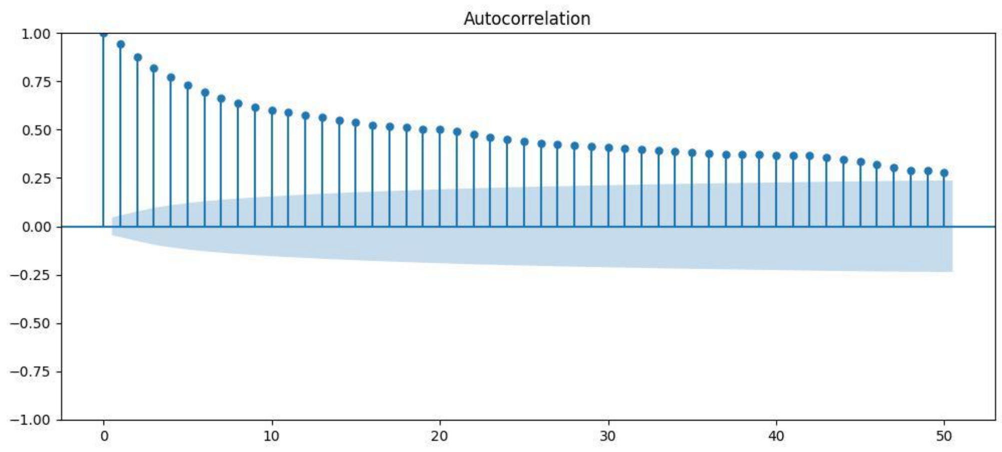


Fig 3.12: Auto correlation Plot of Rajasthan Data set

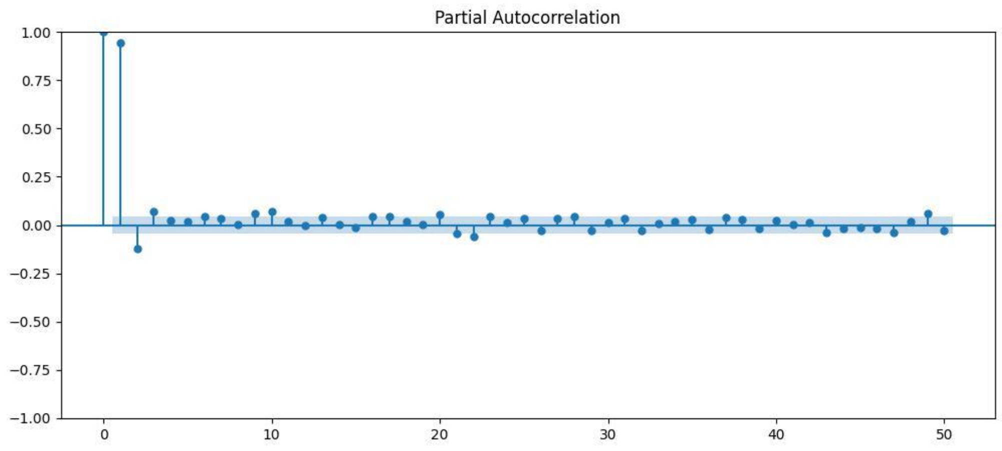


Fig 3.13: Partial correlation Plot of Rajasthan Data set

The day 1 is showing strong correlation with the rest of data. Hence, the one lag column with 1 day shift respectively will make the model more reliable.

3.5. Delhi Data set:

Delhi has data available for the five years period from 2017 to 2022 that contains the daily total consumption by union territory of Delhi. The data set was loaded using pandas and seasonality test was performed using the seasonal decompose function. The results are presented in Fig 3.14.

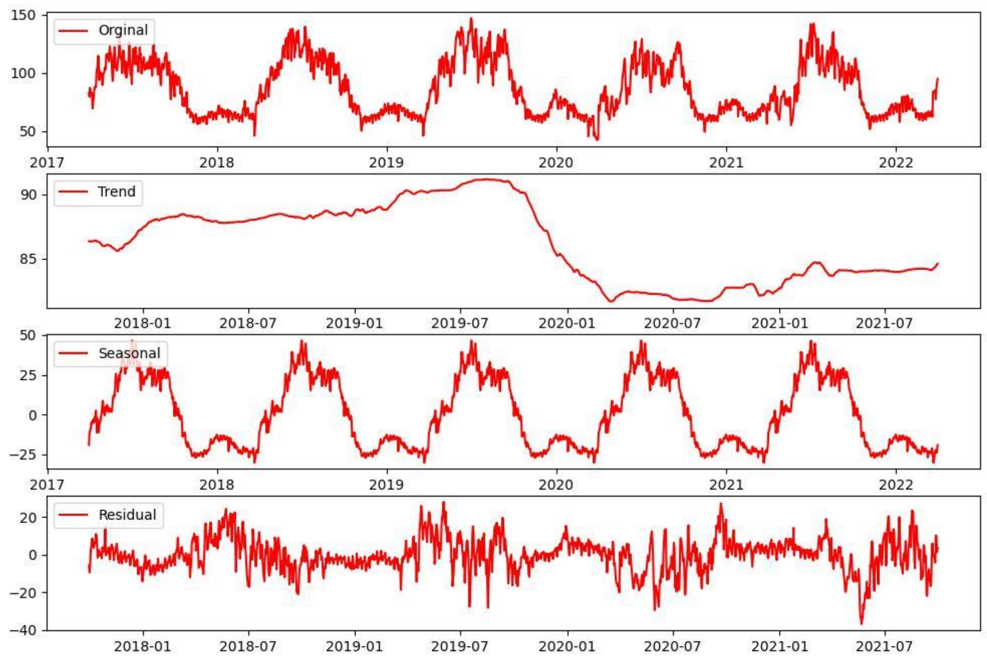


Fig 3.14: Seasonal Decomposition of Delhi Data Set

Figure 3.14 Load consumption comparison between Delhi and Haryana The load consumption is similar in seasonality with Haryana, with minimum load in December, and maximum load in the summer months. As an industrial urban city, Delhi has experienced a steep decline in electricity demand during the COVID-19 lockdown, which is reflected in the trend. The autocorrelation and partial correlation plot is provided in Figures 3.15 and 3.16.

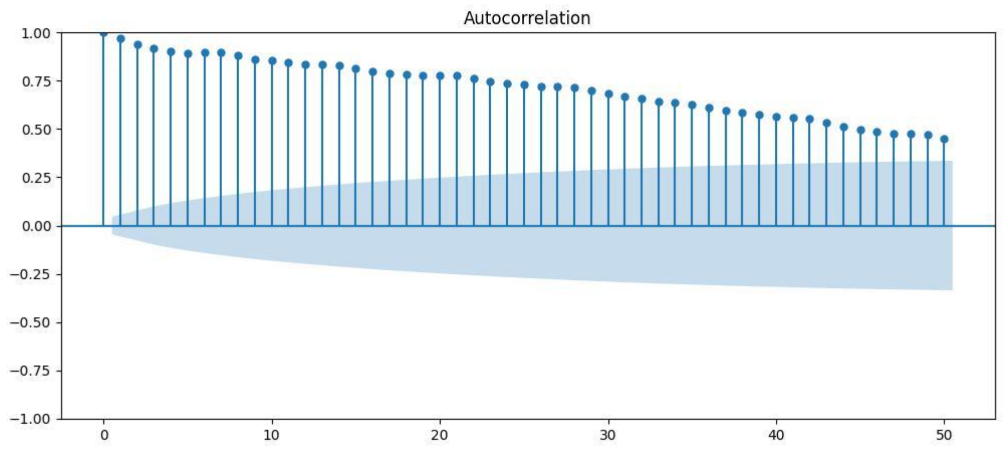


Fig 3.15: Auto correlation Plot of Delhi Data set

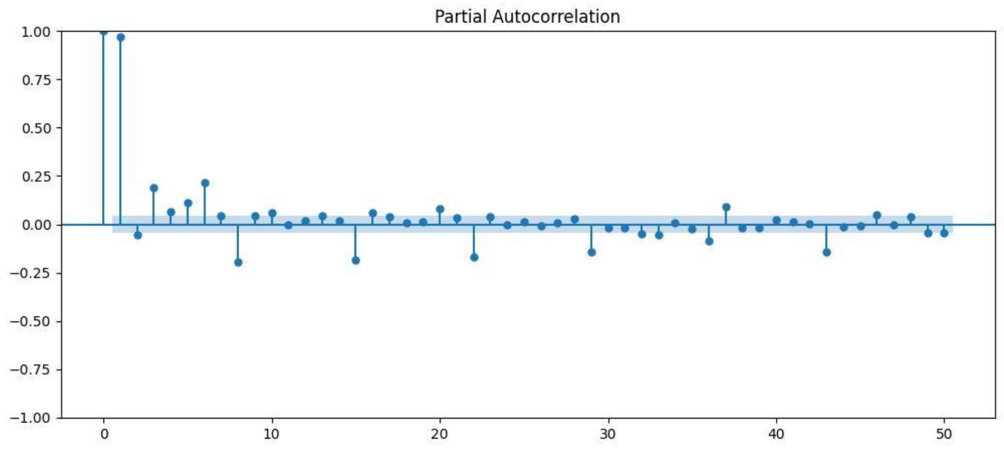


Fig 3.16: Partial correlation Plot of Delhi Data set

The day 1 and day 6 are showing strong correlation with the data. Hence, the adding two lag columns with 1 and 6 days shift respectively will make the model more reliable.

3.1. <u>UP Data set:</u>

The UP dataset contains the total daily consumption data for the state of Uttar Pradesh, from the years 2017 to 2022. It was loaded in by using pandas and further seasonality test has been performed by using the seasonal decompose function. The result obtained is shown in Figure 3.17.

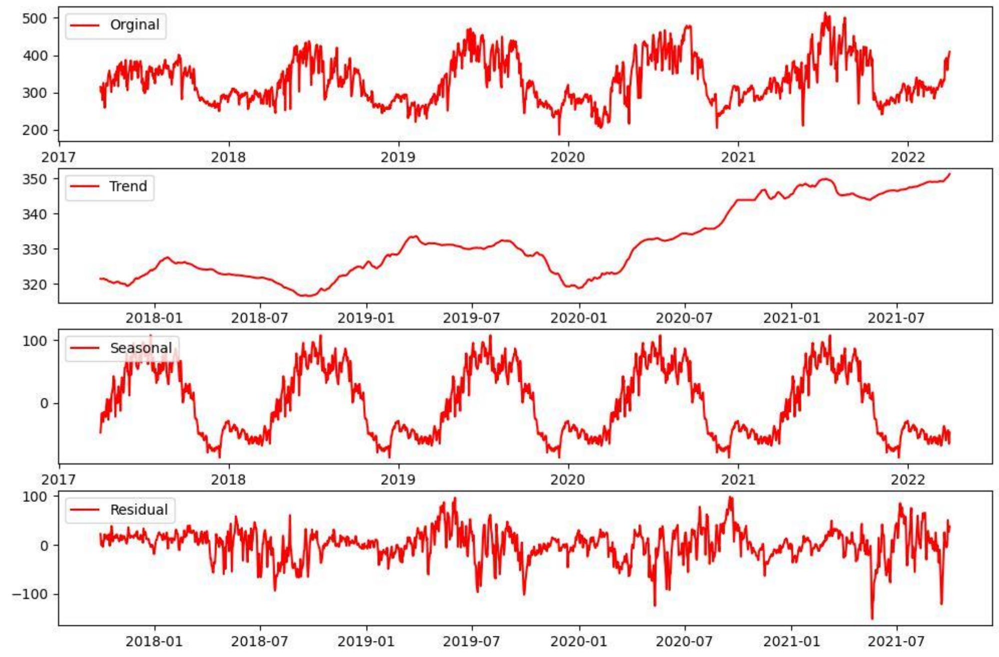


Fig 3.17: Seasonal Decomposition of UP Data Set

One can notice easily the seasonality and trend of the data. The data is seasonal with load at its all year low during the month of December and recording all year high around the summer season. As UP GDP grows, the demand of electricity will also keep increasing and the trend reflects that. Auto correlation and partial correlation are plotted in Fig 3.18 and Fig 3.19.

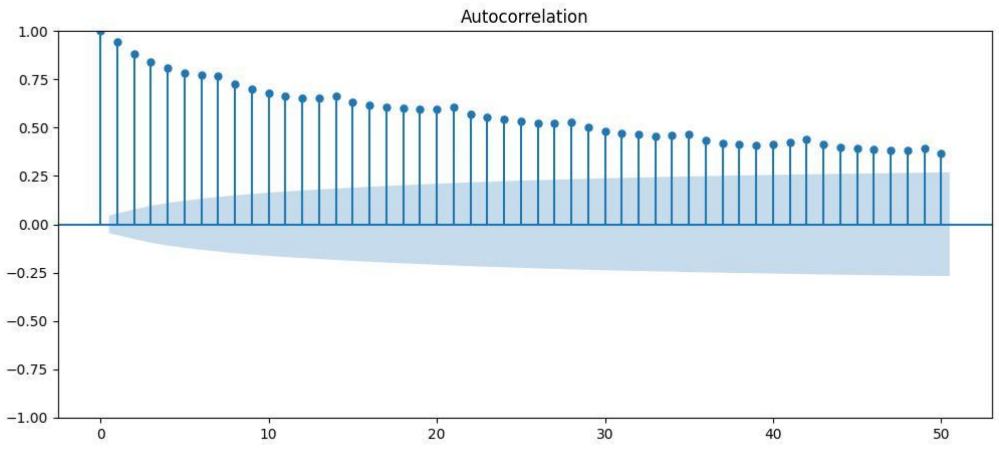


Fig 3.3: Auto correlation Plot of UP Data set

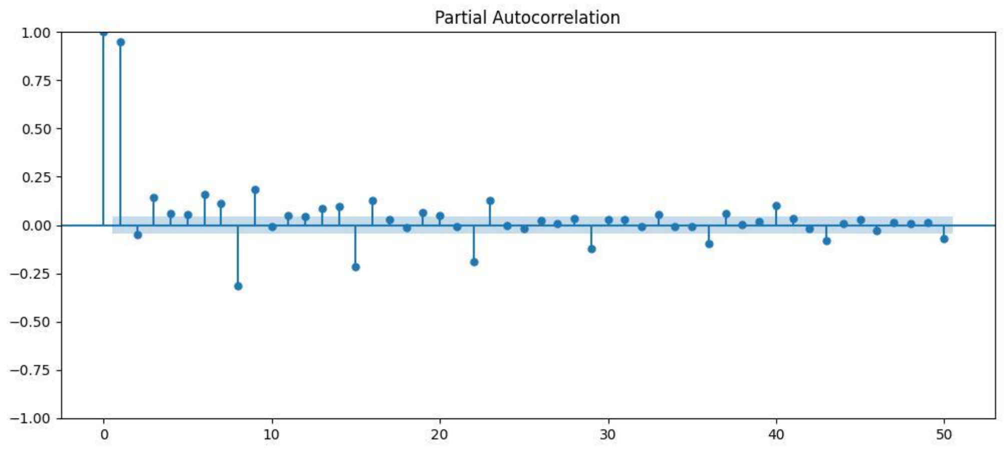


Fig 3.4: Partial correlation Plot of UP Data set

The day 1 is showing strong correlation with the data. Hence, the adding one lag column with 1 day shift respectively will make the model more reliable.

Chapter-4
Model Evaluation

1. Metrics used for Evaluation

1.1. R2 Score

R-squared measures goodness of fit of the regression line to actual data. It measures the fraction of variation in the dependent variable accounted for by the independent variable(s) included within the regression model. The R-squared values range between 0 and 1, or 0% and 100%. The higher value signifies a good fit of the regression line to the data. The higher the R-squared value, the more the variation of data is explained by the model, and therefore, it makes it more precise at the end of prediction.[8]

Formula

R square is the comparison made between the sum of squares of residuals (SSR) and total sum of squares (SST). The total sum of squares would be calculated by summation of squares of perpendicular distance between data points and the average line.

The formula to calculate R squared is:

$$R - Squared = 1 - \left(\frac{SSR}{SST}\right) = 1 - \frac{\sum_{i=1}^{n}(yi - y\hat{\imath})^2}{\sum_{i=1}^{n}(yi - y\bar{\imath})^2}$$

Where,

SSR is the sum of squared residuals (i.e., the sum of squared errors)

SST is the total sum of squared (i.e., the sum of squared deviations from the mean)

Limitations

1. Misleading R-squared: R-squared may inflate by noise in the data and provides a false sense of how accurate the model is.
2. Limitations with categorical variables: In case of categorical variables, R-squared is less useful. Other metrics will be more useful in such cases.

1.2. Mean Absolute Error

The Mean Absolute Error (MAE) is the average of all absolute errors. The formula is:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - x|$$

n = the number of errors,

Σ = summation symbol (which means “add them all up”),

$|x_i - x|$ = the absolute errors.

1.3. Mean Squared Error

MSE is the average of squared differences between estimated and true values. MSE is sometimes called a risk function since it represents the expected squared error loss. It measures the expected value of squared error loss. Actually, MSE is always nonnegative; the smaller its value, the better. As can be seen above, MSE incorporates both variance

and bias of the estimator, but remains the most often used measure of how well a model performs.
The formula for calculation is given in the following equation.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \widehat{Y_i})^2$$

1.4. Root Mean Squared Error [10]

RSME is a square root of the average squared difference between the predicted and actual value of the variable/feature. Let's see the following formula.

$$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - p_i)^2}$$

Σ - It represents the "sum".

d_i- It represents the predicted value for the ith

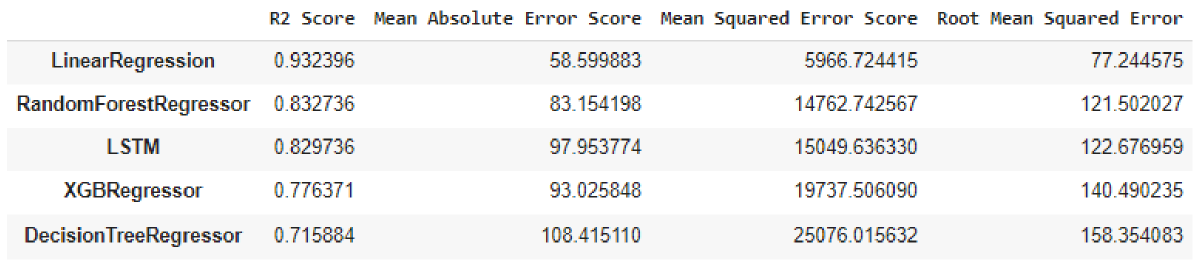
p_i- It represents the predicted value for the ith

n - It represents the sample size.

2. Model Evaluation

2.1. All India Dataset

The Fig 4.1 shows summary of multiple trained models that were on the All India Data Set.



	R2 Score	Mean Absolute Error Score	Mean Squared Error Score	Root Mean Squared Error
LinearRegression	0.932396	58.599883	5966.724415	77.244575
RandomForestRegressor	0.832736	83.154198	14762.742567	121.502027
LSTM	0.829736	97.953774	15049.636330	122.676959
XGBRegressor	0.776371	93.025848	19737.506090	140.490235
DecisionTreeRegressor	0.715884	108.415110	25076.015632	158.354083

Fig 4.1: Models Performance Summary on All India Data Set

The All India Data was best performed by Linear Regression while it will be Decision Tree that gives us the worst result. The Fig 4.2 shows the graph of predicted vs actual values for Linear Regression.

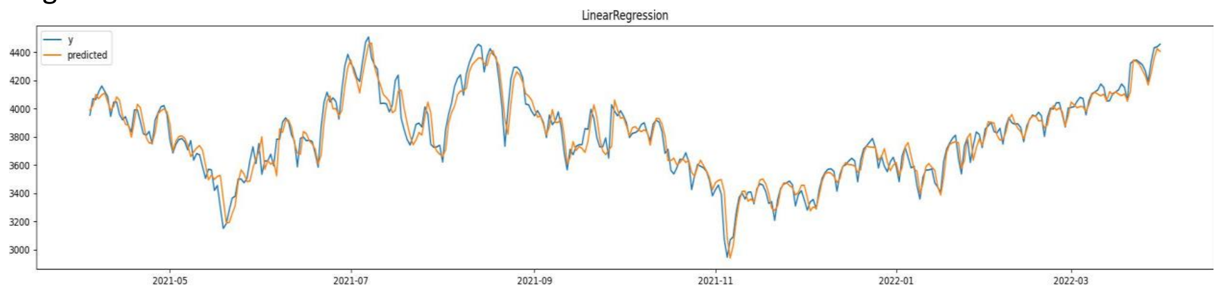


Fig 4.2: Predicted vs actual values for Linear Regression

The Fig 4.3 displays the graph of predicted vs actual values for Random Forest Regression.

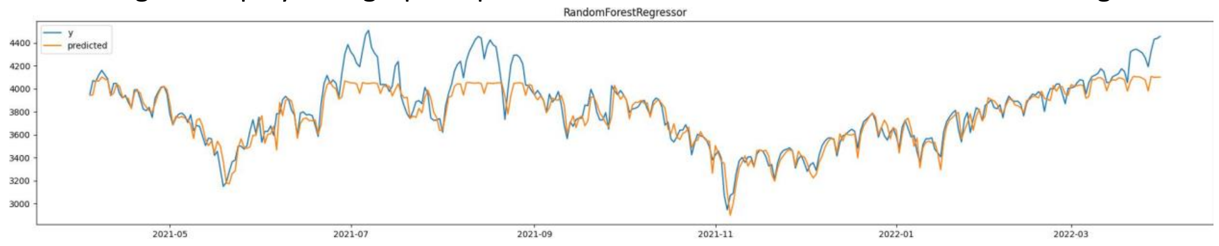


Fig 4.3: Predicted vs actual values for Random Forest Regression

The Fig 4.4 displays the graph of predicted vs actual values for LSTM Deep Model.

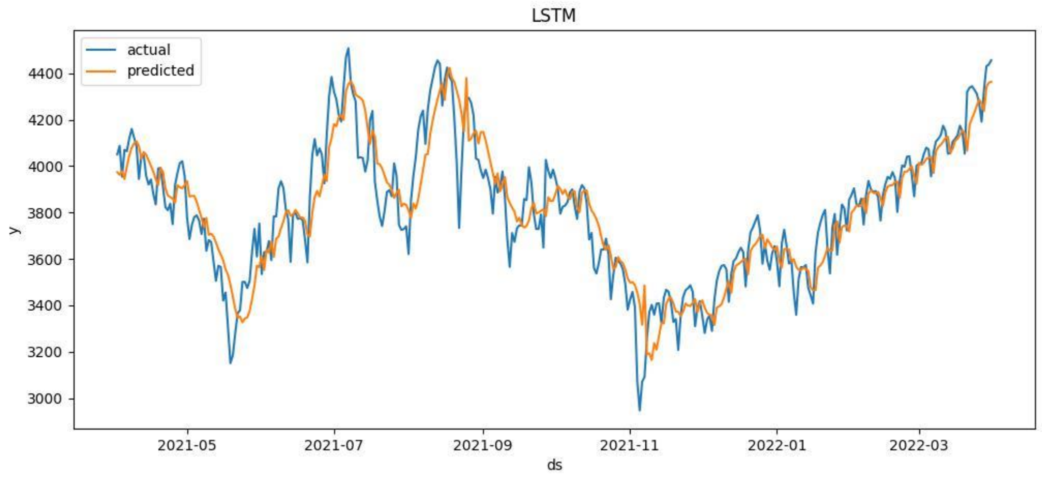


Fig 4.4: Predicted vs actual values for LSTM Deep Model

The Fig 4.5 displays the graph of predicted vs actual values for XGB Regression.

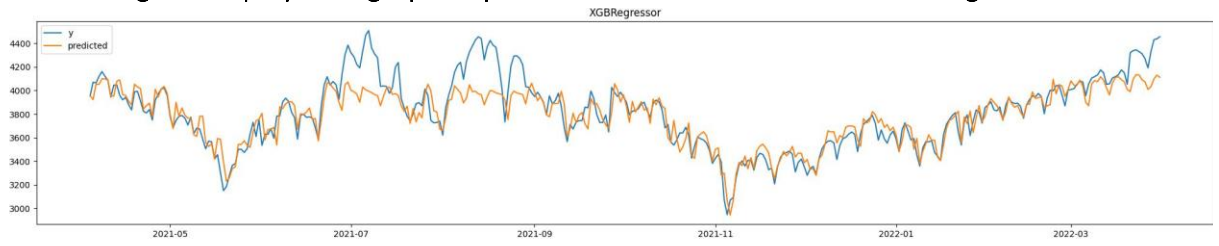


Fig 4.5: Predicted vs actual values for XGB Regression

The Fig 4.6 displays the graph of predicted vs actual values for Decision Tree Regression.

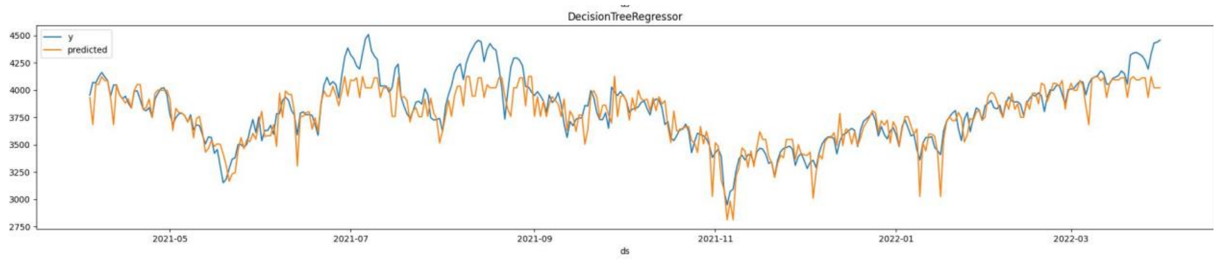
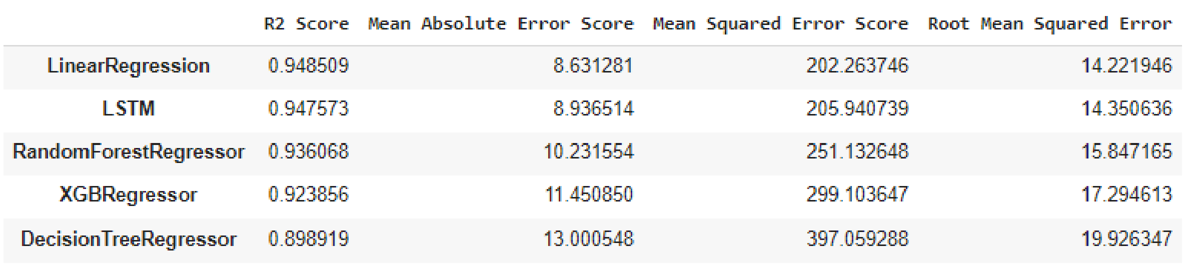


Fig 4.6: Predicted vs actual values for Decision Tree Regression

2.2. Punjab Dataset

The Fig 4.7 shows the summary of multiple models that were trained and tested on the Punjab Data Set.



	R2 Score	Mean Absolute Error Score	Mean Squared Error Score	Root Mean Squared Error
LinearRegression	0.948509	8.631281	202.263746	14.221946
LSTM	0.947573	8.936514	205.940739	14.350636
RandomForestRegressor	0.936068	10.231554	251.132648	15.847165
XGBRegressor	0.923856	11.450850	299.103647	17.294613
DecisionTreeRegressor	0.898919	13.000548	397.059288	19.926347

Fig 4.7: Models Performance Summary on Punjab Data Set

The Linear Regression performs the best on the Punjab Data with LSTM just behind it, while once more, Decision Tree gives us the worst result.

The graph of predicted vs actual values for Linear Regression is depicted in Fig 4.8.

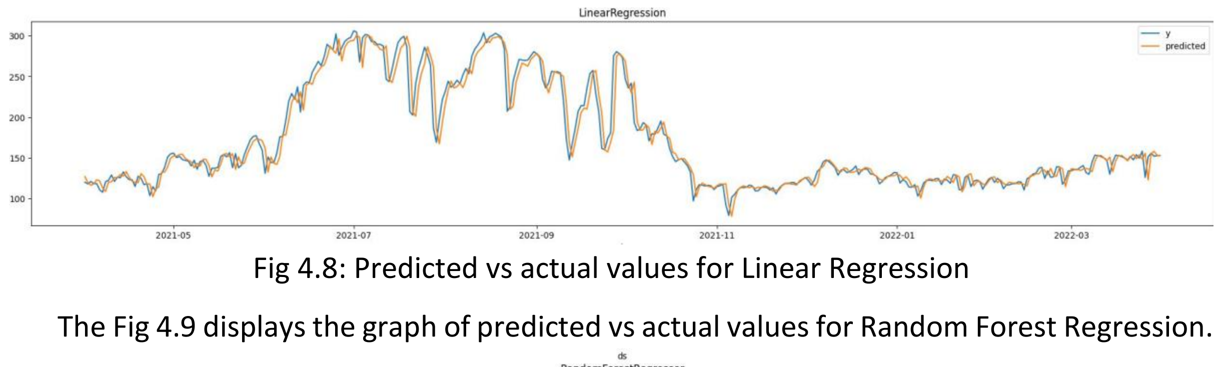


Fig 4.8: Predicted vs actual values for Linear Regression

The Fig 4.9 displays the graph of predicted vs actual values for Random Forest Regression.

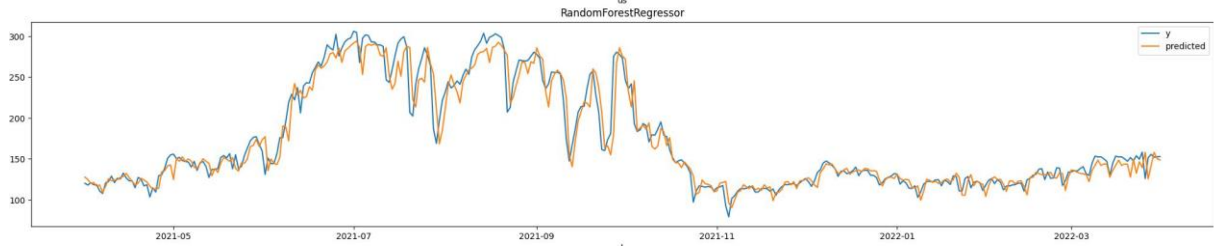


Fig 4.9: Predicted vs actual values for Random Forest Regression

The Fig 4.10 displays the graph of predicted vs actual values for LSTM Deep Model.

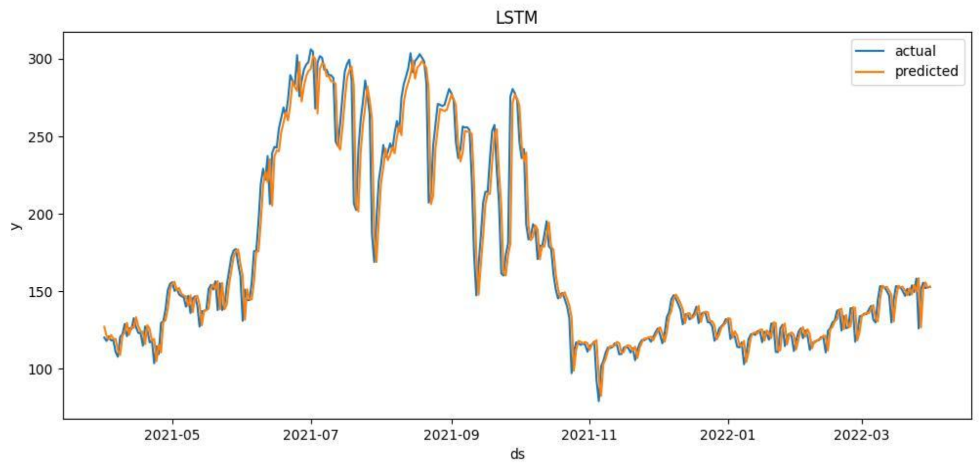


Fig 4.10: Predicted vs actual values for LSTM Deep Model

The Fig 4.11 displays the graph of predicted vs actual values for XGB Regression.

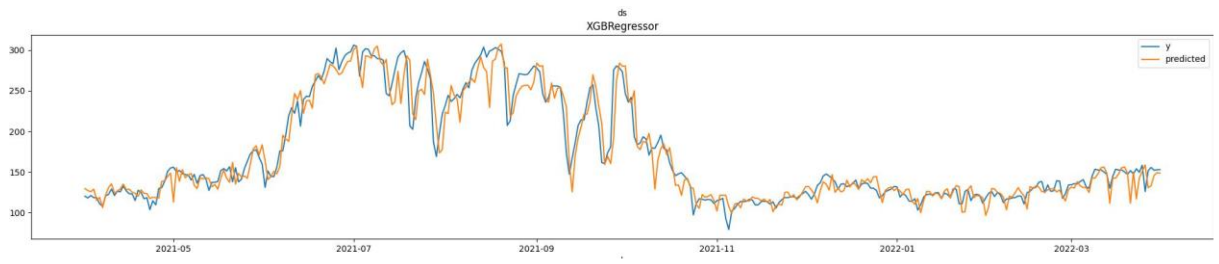


Fig 4.11: Predicted vs actual values for XGB Regression

The Fig 4.12 displays the graph of predicted vs actual values for Decision Tree Regression.

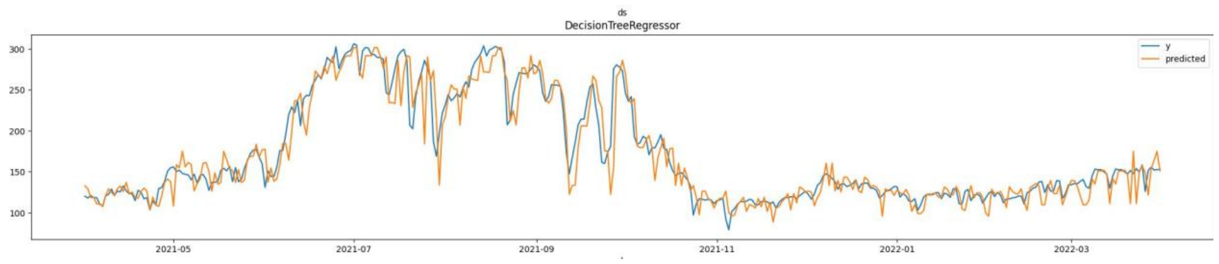


Fig 4.12: Predicted vs actual values for Decision Tree Regression

2.3. <u>Haryana Dataset</u>

The Fig 4.13 shows the summary of multiple models that were trained and tested on the Haryana Set.

	R2 Score	Mean Absolute Error Score	Mean Squared Error Score	Root Mean Squared Error
LinearRegression	0.909746	7.660772	135.433806	11.637603
LSTM	0.893376	8.651432	159.998630	12.649057
RandomForestRegressor	0.873715	8.696744	189.502000	13.765973
XGBRegressor	0.848321	9.319586	227.607353	15.086661
DecisionTreeRegressor	0.782135	11.204658	326.925233	18.081074

Fig 4.13: Models Performance Summary on All India Data Set

The Linear Regression performed the best on the Haryana Data will Decision Tree giving us the worst result.

The graph of predicted vs actual values for Linear Regression is displayed in Fig 4.14.

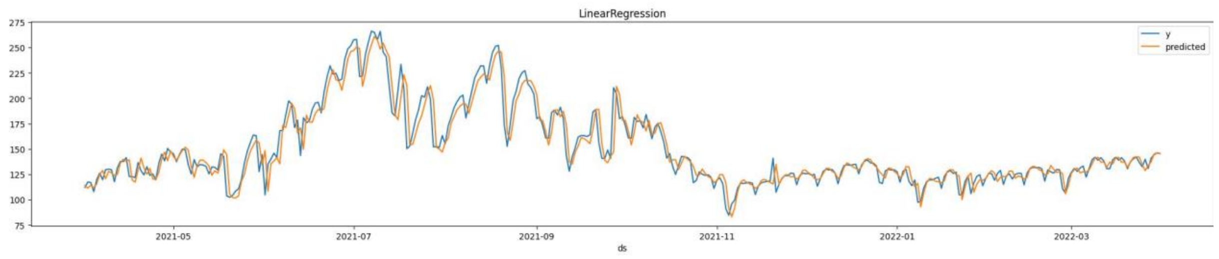


Fig 4.14: Predicted vs actual values for Linear Regression

The Fig 4.15 displays the graph of predicted vs actual values for Random Forest Regression.

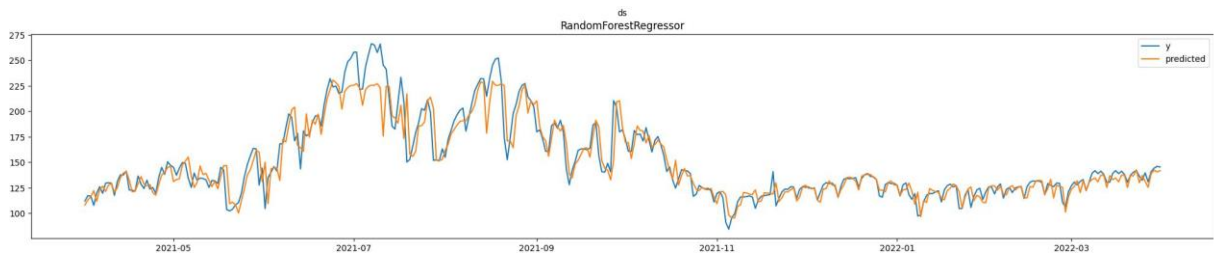


Fig 4.15: Predicted vs actual values for Random Forest Regression

The Fig 4.16 displays the graph of predicted vs actual values for LSTM Deep Model.

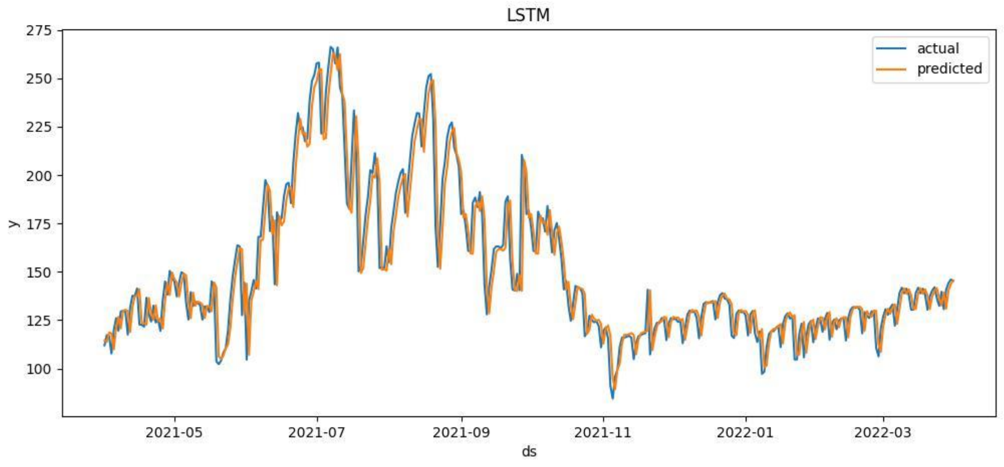


Fig 4.16: Predicted vs actual values for LSTM Deep Model

The Fig 4.17 displays the graph of predicted vs actual values for XGB Regression.

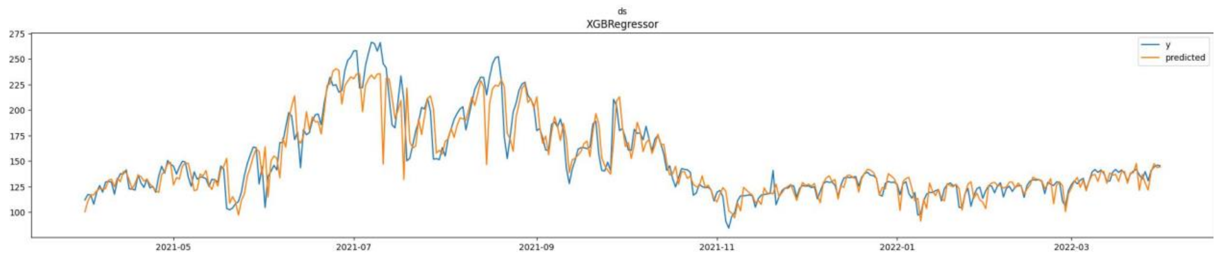


Fig 4.17: Predicted vs actual values for XGB Regression

The Fig 4.18 displays the graph of predicted vs actual values for Decision Tree Regression.

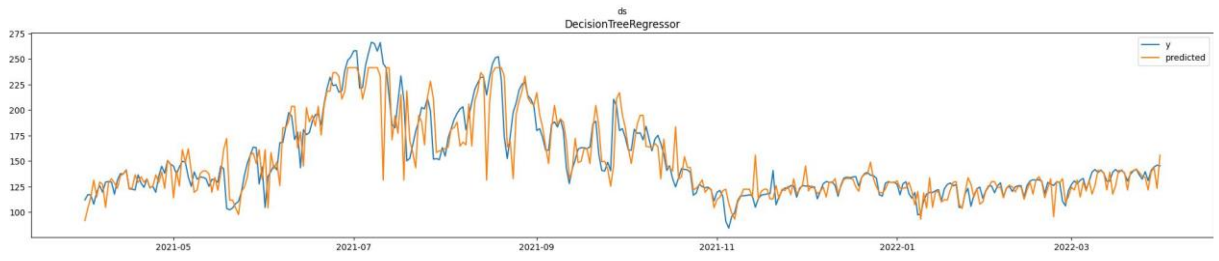
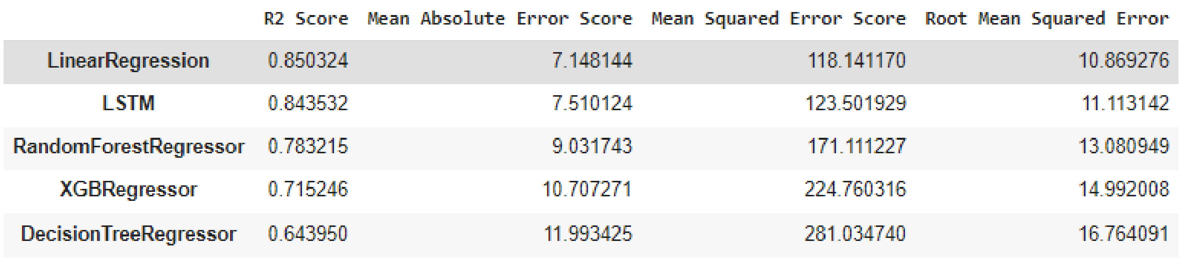


Fig 4.18: Predicted vs actual values for Decision Tree Regression

2.4. Rajasthan Dataset

The Fig 4.19 shows the summary of multiple models that were trained and tested on the Rajasthan Data Set.



	R2 Score	Mean Absolute Error Score	Mean Squared Error Score	Root Mean Squared Error
LinearRegression	0.850324	7.148144	118.141170	10.869276
LSTM	0.843532	7.510124	123.501929	11.113142
RandomForestRegressor	0.783215	9.031743	171.111227	13.080949
XGBRegressor	0.715246	10.707271	224.760316	14.992008
DecisionTreeRegressor	0.643950	11.993425	281.034740	16.764091

Fig 4.10: Models Performance Summary on Rajasthan Data Set

The best performance came from Linear Regression on the Rajasthan Data and the worst result by Decision Tree.

The Fig 4.20 is the graph of predicted vs actual values for Linear Regression.

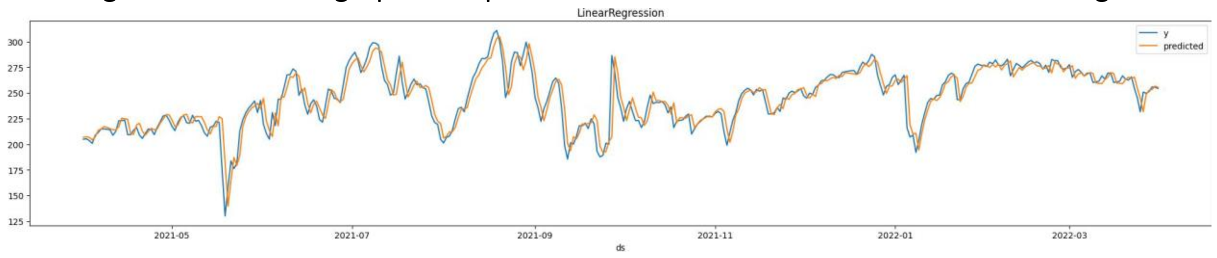


Fig 4.20: Predicted vs actual values for Linear Regression

The Fig 4.21 displays the graph of predicted vs actual values for Random Forest Regression.

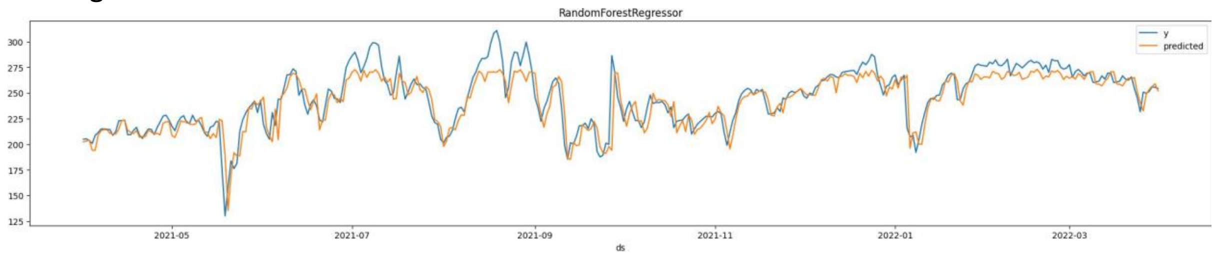


Fig 4.21: Predicted vs actual values for Random Forest Regression

The Fig 4.22 displays the graph of predicted vs actual values for LSTM Deep Model.

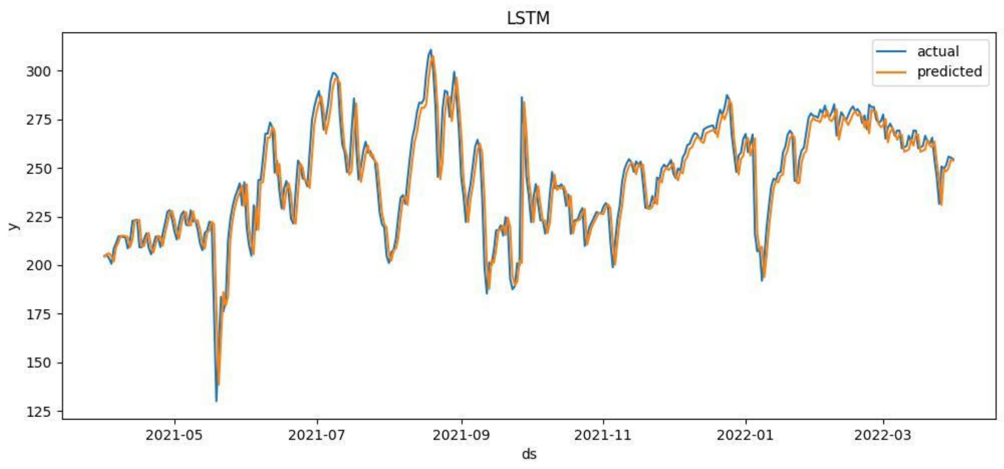


Fig 4.22: Predicted vs actual values for LSTM Deep Model

The Fig 4.23 displays the graph of predicted vs actual values for XGB Regression.

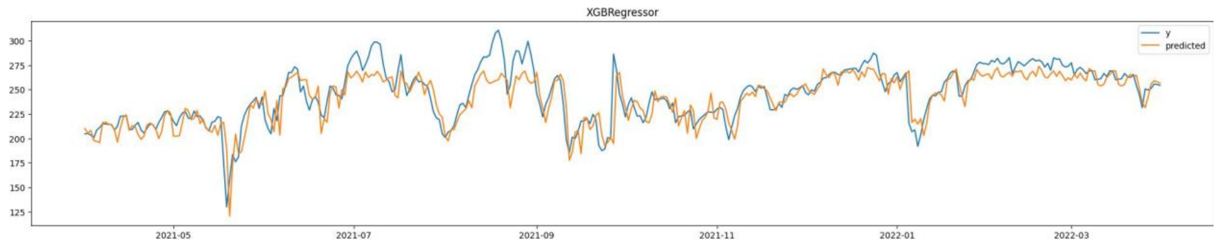


Fig 4.23: Predicted vs actual values for XGB Regression

The Fig 4.24 displays the graph of predicted vs actual values for Decision Tree Regression.

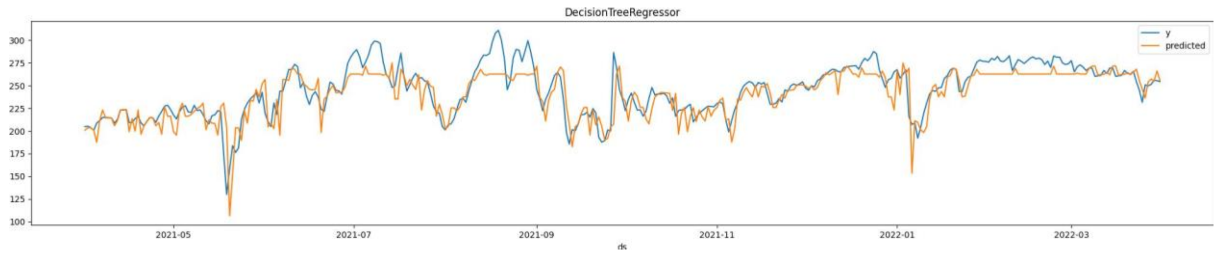
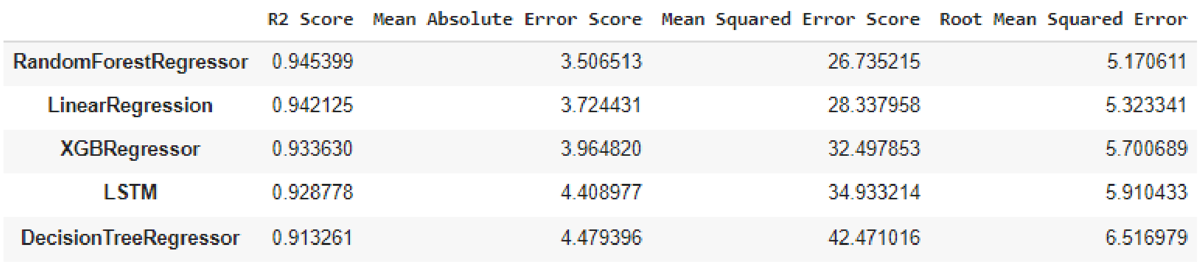


Fig 4.24: Predicted vs actual values for Decision Tree Regression

2.5. <u>Delhi Dataset</u>

The Fig 4.25 shows the summary of multiple models that were trained and tested on the Delhi Data Set.



	R2 Score	Mean Absolute Error Score	Mean Squared Error Score	Root Mean Squared Error
RandomForestRegressor	0.945399	3.506513	26.735215	5.170611
LinearRegression	0.942125	3.724431	28.337958	5.323341
XGBRegressor	0.933630	3.964820	32.497853	5.700689
LSTM	0.928778	4.408977	34.933214	5.910433
DecisionTreeRegressor	0.913261	4.479396	42.471016	6.516979

Fig 4.25: Models Performance Summary on Delhi Data Set

The Random Forest Regression gave the best performance on the Delhi Data will Decision Tree, giving us the worst result.

Fig 4.26: Graph of Predicted vs actual values for Linear Regression.

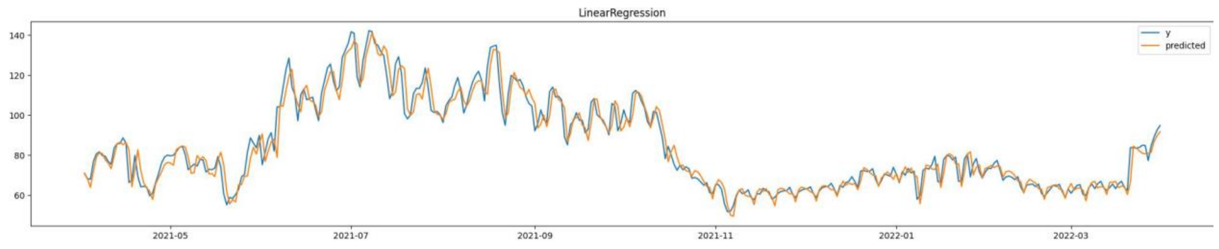


Fig 4.26: Predicted vs actual values for Linear Regression

The Fig 4.27 displays the graph of predicted vs actual values for Random Forest Regression.

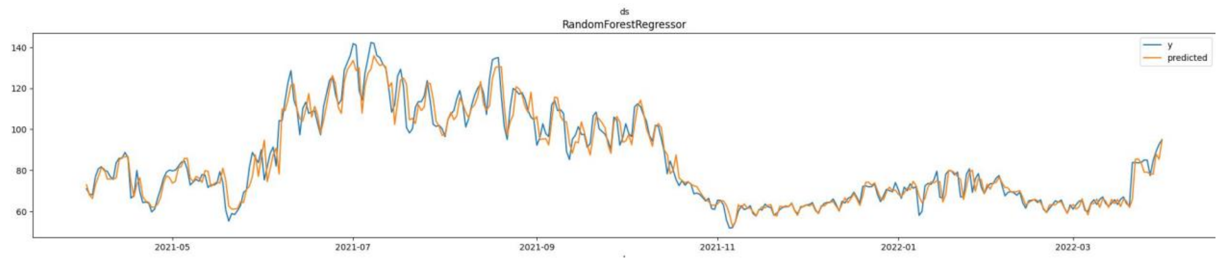


Fig 4.27: Predicted vs actual values for Random Forest Regression

The Fig 4.28 displays the graph of predicted vs actual values for LSTM Deep Model.

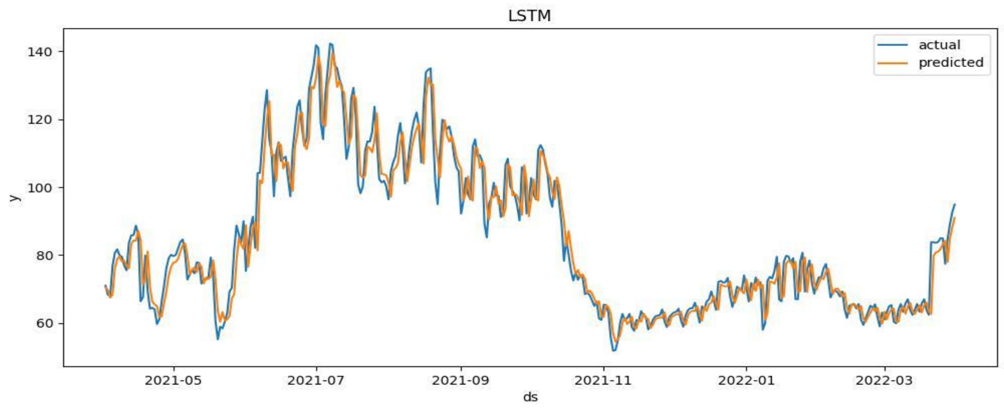


Fig 4.28: Predicted vs actual values for LSTM Deep Model

The Fig 4.29 displays the graph of predicted vs actual values for XGB Regression.

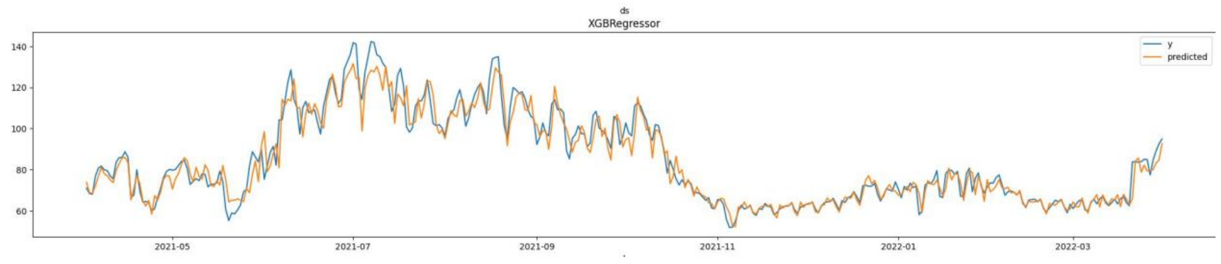


Fig 4.29: Predicted vs actual values for XGB Regression.

The Fig 4.30 displays the graph of predicted vs actual values for Decision Tree Regression.

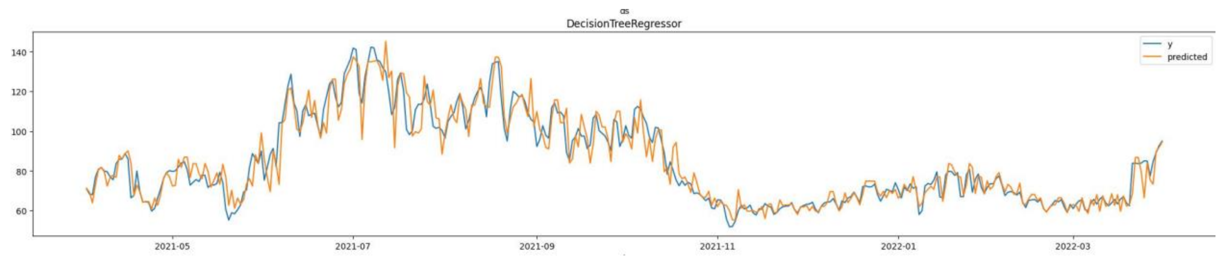
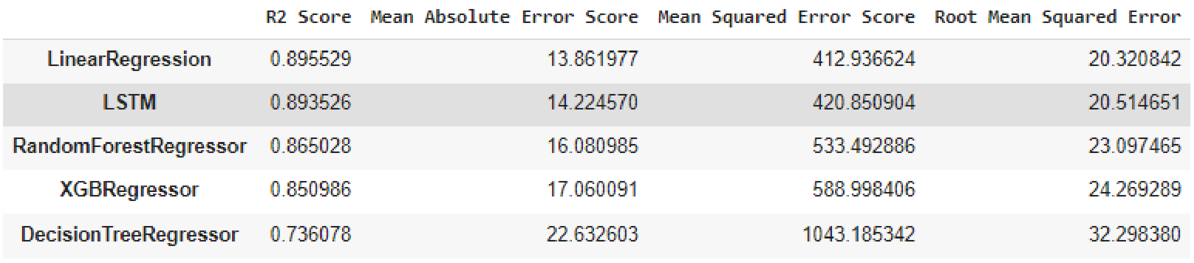


Fig 4.30: Predicted vs actual values for Decision Tree Regression

2.6. UP Dataset

The Fig 4.31 shows the summary of multiple models that were trained and tested on the UP Data Set.



	R2 Score	Mean Absolute Error Score	Mean Squared Error Score	Root Mean Squared Error
LinearRegression	0.895529	13.861977	412.936624	20.320842
LSTM	0.893526	14.224570	420.850904	20.514651
RandomForestRegressor	0.865028	16.080985	533.492886	23.097465
XGBRegressor	0.850986	17.060091	588.998406	24.269289
DecisionTreeRegressor	0.736078	22.632603	1043.185342	32.298380

Fig 4.31: Models Performance Summary on UP Data Set

We tried the Linear Regression at its best on the UP Data, but decision tree gave us our worst result.

Fig 4.32 depicts the plot of predicted vs actual values for Linear Regression.

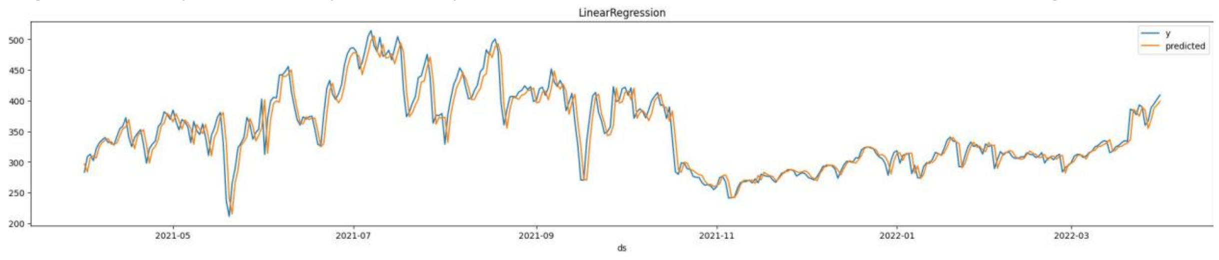


Fig 4.32: Predicted vs actual values for Linear Regression

The Fig 4.33 displays the graph of predicted vs actual values for Random Forest Regression.

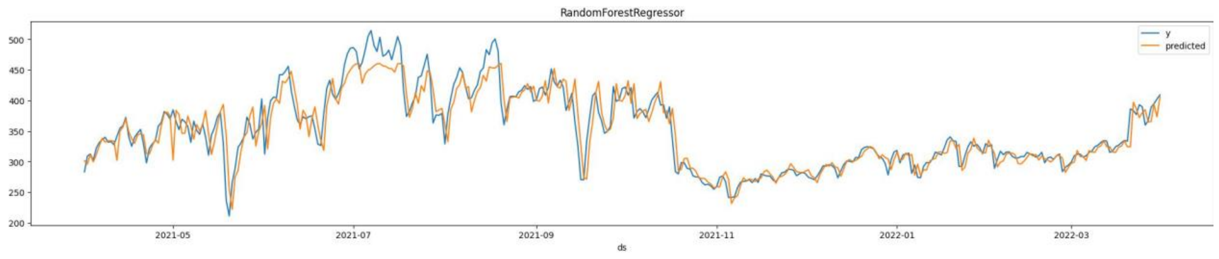


Fig 4.33: Predicted vs actual values for Random Forest Regression

The Fig 4.34 displays the graph of predicted vs actual values for LSTM Deep Model.

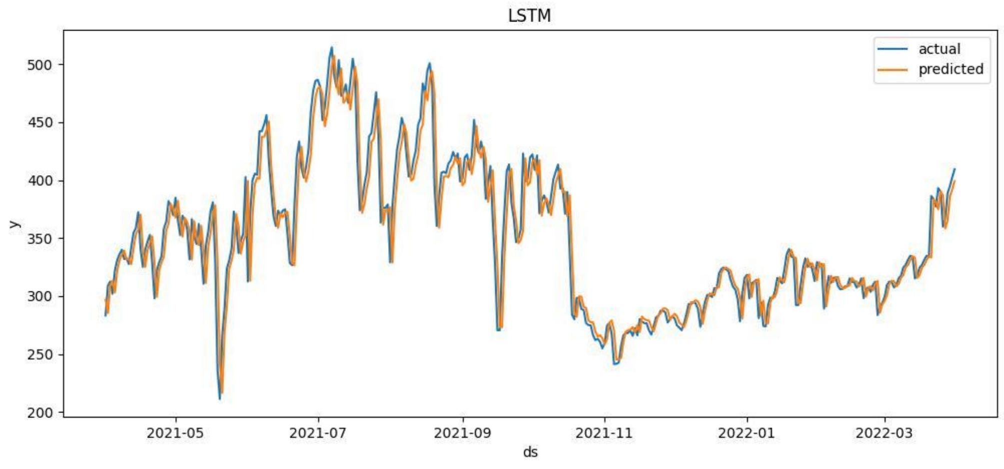


Fig 4.34: Predicted vs actual values for LSTM Deep Model

The Fig 4.35 displays the graph of predicted vs actual values for XGB Regression.

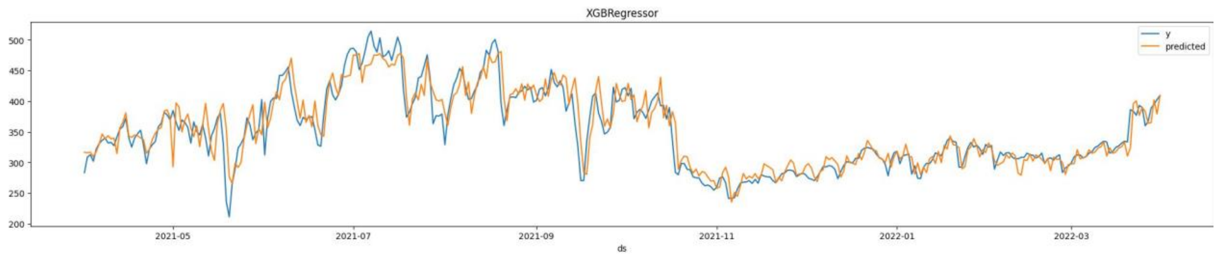


Fig 4.35: Predicted vs actual values for XGB Regression

The Fig 4.36 displays the graph of predicted vs actual values for Decision Tree Regression.

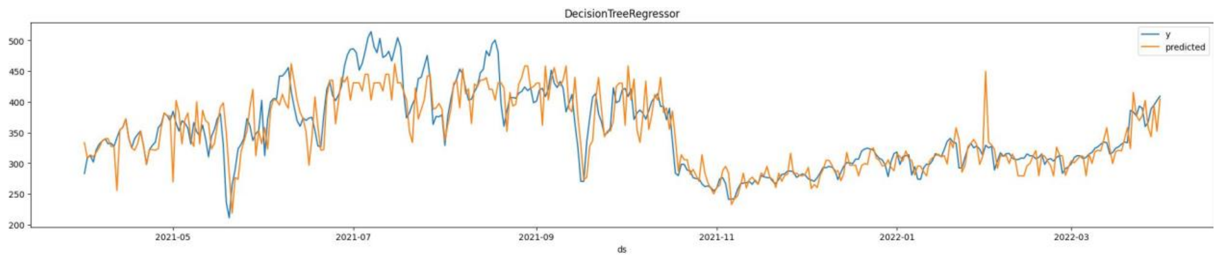
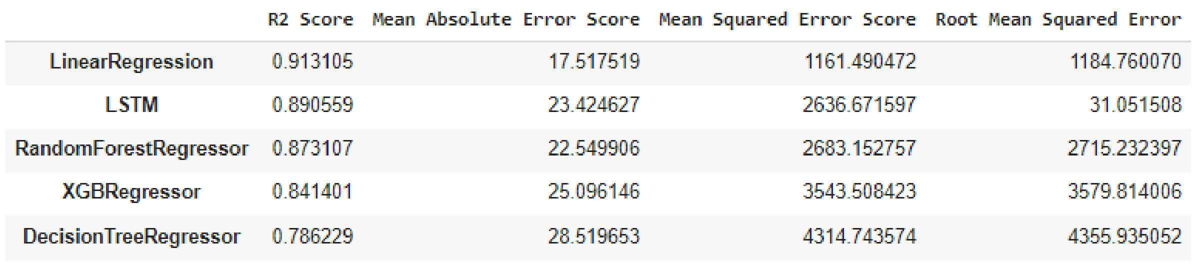


Fig 4.36: Predicted vs actual values for Decision Tree Regression

Chapter-5
Conclusions and Future Scope

Taking the average of all four metrics for all six models, the results are displayed as a table. The result is shorted on the base of R2 score.



	R2 Score	Mean Absolute Error Score	Mean Squared Error Score	Root Mean Squared Error
LinearRegression	0.913105	17.517519	1161.490472	1184.760070
LSTM	0.890559	23.424627	2636.671597	31.051508
RandomForestRegressor	0.873107	22.549906	2683.152757	2715.232397
XGBRegressor	0.841401	25.096146	3543.508423	3579.814006
DecisionTreeRegressor	0.786229	28.519653	4314.743574	4355.935052

Fig 5.1: Average result of every Model

Based on the Fig. 5.1, the Linear Regression performed on the best on average followed by LSTM deep learning model and Random Forest Regressor and Decision tree fairing the worst. Based on the above summary the linear regression can be used for load forecasting. Some to the future scopes of load forecasting are discussed below:

1. Advanced machine learning: RNN and CNN techniques will be developed, enhancing the load forecasting model in terms of accuracy and efficiency level.
2. Renewable Energy Integration: There will be heavy reliance on load forecasting in the integration of renewable energy sources into the grid in such a way that there is proper supply versus demand.
3. Smarter Grid Technologies: With accurate load forecasting, the different smart grid applications shall optimize the operations, scheduling, maintenance planning, and system efficiency of the grid.
4. Big Data Analytics: More accurate predictions that will enhance the load forecasting model with data from smart meters, weather sensors, and social media.
5. Electric Vehicle Integration: Introduction of EVs may pose problems for the load forecasting as there would be a permanent demand of charging from those vehicles not predicted appropriately, thus may overloads the grid.
6. Uncertainty Quantification: The uncertainty arising from weather, market condition and consumer behaviour would further make the load forecasting models more robust and reliable.

References

[1] H. Hippert, C. Pedreira and R. Souza, “Neural Networks for Short-Term Load Forecasting: A Review and Evaluation,” *IEEE Transactions on Power Systems,* vol. 16, pp. 44-55, 2001.

[2] R. Engle, C. Mustafa and J. Rice, “Modeling Peak Electricity Demand,” *Journal of Forecasting,* vol. 11, pp. 241-251, 1992..

[3] S. Mill, “Electric load forecasting: advantages and challenges,” electrical-equipment.org, [Online]. Available: https://engineering.electrical-equipment.org/electrical-distribution/electric-load-forecasting-advantages-challenges.html#:~:text=Electrical%20load%20forecasting%20is%20an,consumers%20with%20the%20required%20energy.

[4] J. H. Chow, W. F. Felix and J. A. Momoh, Applied Mathematics for Restructured Electric Power Systems, Springer, 2005.

[5] “Decision Tree Classification Algorithm,” javatpoint, [Online]. Available: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm.

[6] G. M. K, “Machine Learning Basics: Decision Tree Regression,” Towards Data Science, 2020 July 2020. [Online]. Available: https://towardsdatascience.com/machine-learning-basics-decision-tree-regression-1d73ea003fda.

[7] M. R. Islam, A. A. Mamun and M. Sohel, “LSTM-Based Electrical Load Forecasting for Chattogram City of Bangladesh,” in *International Conference on Emerging Smart Computing and Informatics (ESCI)*, AISSMS Institute of Information Technology, Pune, India. Mar 12-14, 2020, 2020.

[8] M. Narang, “How to Calculate R squared in Linear Regression,” 16 November 2022. [Online]. Available: https://www.shiksha.com/online-courses/articles/how-to-calculate-r-squared-in-linear-regression/#Limitations-of-Using-R-Squared.

Appendix:

- Code link :

  https://colab.research.google.com/drive/1PCnKsBjX7oeOcJpWeM-58jgf30G9n6UY?usp=sharing