COMPARATIVE ANALYSIS OF LOAD FORECASTING ALGORITHMS USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

S. KARTHIKEYAN	**(Reg.No.:312320105061)**
J. PRAVEEN KUMAR	**(Reg.No.:312320105092)**

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

St. JOSEPH'S COLLEGE OF ENGINEERING
(An Autonomous Institution)

ANNA UNIVERSITY: CHENNAI 600025

APRIL 2024

ANNA UNIVERSITY: CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report "**COMPARATIVE ANALYSIS OF LOAD FORECASTING ALGORITHMS USING MACHINE LEARNING**" is the bonafide work of **KARTHIKEYAN S (312320105092)**, **PRAVEEN KUMAR J (312320105092)** who carried out the project work under my supervision**.**

Signature	**Signature**
Dr. JAYARAMA PRADEEP M.E., Ph.D.,	**Mr. H.PRASAD M.E., (Ph.D.),**
PROFESSOR	**ASSISTANT PROFESSOR**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Department of Electrical and	Department of Electrical and
Electronics Engineering**,**	Electronics Engineering**,**
St. Joseph's College of Engineering	St. Joseph's College of Engineering
OMR, Chennai - 600 119.	OMR, Chennai -600 119.

CERTIFICATE OF EVALUATION

College Name	:	St. Joseph's College of Engineering
Branch Name	:	Electrical and Electronics Engineering
Semester	:	08

S.NO.	NAME OF THE STUDENT	TITLE OF THE PROJECT	NAME OF THE SUPERVISOR WITH DESIGNATION
1.	**S. KARTHIKEYAN (312320105061)**	**COMPARATIVE ANALYSIS OF LOAD FORECASTING ALGORITHMS USING MACHINE LEARNING**	**Mr. H. PRASAD M.E., (Ph.D.), ASSISTANT PROFESSOR SUPERVISOR** Department of Electrical and Electronics Engineering,
2.	**J. PRAVEEN KUMAR (312320105092)**		

The report of the project work submitted by the above students in partial fulfillment for the award of **Bachelor of Engineering** degree in **Electrical and Electronics Engineering** of Anna University were evaluated and confirmed to be report of the work done by the above students.

Submitted for the project viva-voice held on:________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We wish to convey our gratitude to our Chairman **Dr. B. Babu Manoharan, M.A., M.B.A., Ph.D.,** our beloved Managing Director **Mr. B. Shashi Sekar, M.Sc.,** and Executive Director **Mrs. S. Jessie Priya, M.Com.,** of St. Joseph's College of Engineering, St. Joseph's Group of Institutions for providing an opportunity to study in their esteemed institution.

We convey our sincere thanks to our beloved and encouraging Principal **Dr.Vaddi Seshagiri Rao, M.E., M.B.A., Ph.D.,** St. Joseph's College of Engineering for his encouragement throughout the course.

We express our sincere thanks and most heartfelt sense of gratitude to **Dr. Jayarama Pradeep, M.E., Ph.D.,** Professor, Head of the Department of Electrical and Electronics Engineering, for her dedication, support and encouragement for the completion of the project with perfection.

We wish to express our sincere and thanks to our Project Co-ordinators **Dr. T. D Sudhakar, M.E., Ph.D.,** Professor **Dr. C. Venkatesh Kumar, M.E., Ph.D.,** Assistant professor **and Ms. S. P. Vedavalli, M.E., (Ph.D.),** Assistant professor of Department of Electrical and Electronics Engineering for their enthusiastic guidance, wonderful motivation, valuable support and constructive remarks provided by them till the end of the project.

It is with deep sense of gratitude that I acknowledge my indebtedness to our beloved supervisor **Mr. H. Prasad, M.E., (Ph.D.), Assistant professor** for his expert guidance and suggestion.

We thank all the faculty members and non-teaching staff of the Department of Electrical and Electronics Engineering who rendered their help in our endeavor.

Our sincere thanks to our parents for their continuous support, cheer and motivation for the successful completion of the project.

TABEL OF CONTENTS

ABSTRACT

Load forecasting is a critical component of energy management systems, with applications ranging from optimizing power generation to grid stability and resource allocation. Accurate load forecasting is essential for ensuring the reliable and cost-effective operation of power systems. In this project, we present a comprehensive comparative analysis of load forecasting algorithms, focusing on Linear Regression, Lasso Regression, Ridge Regression, Elastic Net, Gaussian Processes, Huber Regression, Bayesian Regression, Decision Tree, Support Vector Machine (SVM), Random Forest, Gradient Boosting/ XGBoost, K-Nearest Neighbor (KNN), Long-Short Term Memory (LSTM), Autoregressive Integrated Moving Average (ARIMA) , Autoregressive (AR) and Seasonal and Exogenous Auto Regressive Integrated Moving Average (SARIMAX) are all included. The performance of our models is evaluated through a comprehensive set of metrics, including Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). These metrics provide a holistic view of forecasting accuracy. Our objective is to provide valuable insights into the performance and suitability of these algorithms across diverse load forecasting scenarios.

LIST OF ABBREVATIONS

ML	Machine Learning
LR	Linear Regression
SVM	Support Vector Machines
RF	Random Forest
DT	Decision Tree
ANN	Artificial Neural Network
KNN	K-Nearest Neighbour
LSTM	Long Short-Term Memory
ARIMA	Auto-Regressive Integrated Moving Average
SARIMAX	Seasonal Auto-Regressive Integrated Moving Average with eXogenous variables
MAE	Mean Absolute Error
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
MAPE	Mean Absolute Percentage Error
EMS	Energy Management System
CSV	Comma-Separated Values
XLS	Excel Spreadsheet
DB	Database

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 GENERAL

Electricity holds immense importance in modern life, serving as the lifeblood of technological, social, and economic progress. From powering our homes and lighting our streets to fueling industries and driving innovations, electricity is integral to nearly every aspect of our daily existence. The field of load forecasting plays a crucial role in modern energy management by providing valuable insights into future electricity demand. As the global demand for electricity continues to escalate, accurate load forecasting is essential for optimizing resource allocation, facilitating demand-side management, and enabling strategic decision-making in power generation and distribution. In response to the inherent complexities of energy consumption patterns, machine learning algorithms have emerged as a promising approach to enhance the accuracy and adaptability of load forecasting models. Traditional methods, while effective to a certain extent, often struggle to capture non-linear and dynamic load behaviors. Machine learning, on the other hand, offers the potential to discern complex patterns and variations, thereby enabling a more robust and flexible approach to load forecasting. To contribute to the ongoing discourse on load forecasting by conducting a comparative analysis of various machine learning algorithms, seeking to assess their effectiveness in predicting energy consumption patterns. By doing so, this research seeks to provide valuable insights and empirical evidence to inform the development of more reliable and efficient load forecasting models, thereby contributing to the advancement of predictive analytics in the energy sector.

1.2 FEATURES

1.2.1 SCOPE OF PROJECT

The scope of this project extends to a broad exploration and techniques aimed at enhancing the accuracy and reliability of load predictions using machine learning algorithms in load forecasting, with a particular emphasis on their comparative analysis and practical implementation. This encompasses the investigation of both traditional and state-of-the-art machine learning algorithms, with the objective of identifying their strengths, weaknesses, and suitability for diverse load forecasting scenarios. Each algorithm will be scrutinized based on its ability to capture and extrapolate complex load patterns, adapt to changing load dynamics, and provide interpretable insights into the underlying factors influencing energy consumption. The study will not only address the technical aspects of algorithmic performance but also delve into the scalability and computational efficiency of the models, ensuring that the findings are pertinent to real-world energy management systems. Furthermore, the research will consider the implications of model interpretability and transparency, acknowledging the significance of comprehensible insights for decision-making processes in the energy sector. The scope also encompasses the dissemination of the research findings to industry professionals, researchers, and policymakers involved in energy forecasting and demand-side management, thereby fostering the practical applicability and relevance of the study’s outcomes. By embracing such a comprehensive scope, the project endeavors to provide actionable insights and empirical evidence to inform the ongoing evolution of load forecasting practices and methodologies.

1.2.2 EXISTING SYSTEM

The prevailing landscape of load forecasting predominantly relies on statistical methods and time series analysis, which have historically served as the cornerstone of predictive modeling in the energy sector. These methods, often underpinned by autoregressive integrated moving average (ARIMA) models, exponential smoothing techniques, and similar statistical approaches, have been instrumental in providing baseline forecasts and insights into historical load patterns. However, the limitations of these traditional methods in capturing non-linear and dynamic load behaviors, adapting to evolving consumption patterns, and accommodating the increasing complexity of modern energy systems have prompted the exploration of advanced techniques. Machine learning algorithms present an opportunity to overcome these limitations by leveraging their capacity to discern complex patterns, non-linear relationships, and temporal dynamics inherent in energy consumption data. By integrating machine learning into the existing system, this project seeks to build upon the foundational principles of load forecasting while critically evaluating the potential of these advanced techniques in revolutionizing forecasting methodologies. Through this lens, the research aims to offer a comprehensive appraisal of the strengths and weaknesses of machine learning algorithms in comparison to the established statistical methods, thereby contributing to an informed understanding of the evolving landscape of load forecasting.

1.2.3 INFERENCE

The comparative analysis of load forecasting algorithms using machine learning is anticipated to yield significant insights and actionable outcomes with far-reaching implications for energy management, grid stability, and resource optimization. By rigorously evaluating the performance of diverse machine learning algorithms in predicting energy consumption patterns, this study aims to provide empirical evidence of their efficacy and suitability for practical implementation in load forecasting systems. The insights gleaned from the analysis will not only contribute to the refinement of existing load forecasting methodologies but also hold the potential to lay the groundwork for the development of more robust and adaptive predictive models.

CHAPTER 2

LITERATURE SURVEY

- Electricity holds immense importance in modern life, serving as the lifeblood of technological, social, and economic progress. From powering our homes and lighting our streets to fueling industries and driving innovations, electricity is integral to nearly every aspect of our daily existence. Estimating the load is essential for planning energy production in order to reliably supply enough electricity to meet needs at any moment. Due to the increase in electricity demand, the management of the maintenance schedule, the selection of suitable and affordable generators, and plant planning is a difficult procedure for the electricity supply.
- Electrical load forecasting, a critical component of energy management, which can be broadly classified into three distinct categories based on the forecast period: short-term, medium-term, and long-term.
- Short-term load forecasting, spanning hours to a few days, focuses on predicting immediate electricity demand fluctuations.
- Medium-term load forecasting extends the horizon to several weeks, months, or even a year, aiding in mid-range planning and resource allocation. It considers factors like seasonal variations and specific events affecting demand.
- Long-term load forecasting, covering periods beyond a year, is essential for strategic planning, infrastructure development, and policy formulation.

CHAPTER 3

PROPOSED SYSTEM

3.1 ALGORITHMS FOR FORECASTING

1. LINEAR REGRESSION

- Linear regression is a statistical model which estimates the linear relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables).
- Its primary objective is twofold: prediction and interpretation. In prediction, linear regression helps forecast the value of a dependent variable based on known values of one or more independent variables. In interpretation, it provides insights into the strength and nature of the relationships between variables.
- The simple linear regression model for one predictor variable can be expressed as:

$$Y=\beta_0+\beta_1X+\varepsilon$$

- Here,
  - Y is the dependent variable (the variable we want to predict),
  - X is the independent variable (the variable used for prediction),
  - β_0 is the intercept,
  - β_1 is the slope coefficient,
  - ε represents the error term.
- The multiple linear regression formula, with more than one predictor, extends as:

$$Y=\beta_0+\beta_1X_1+\beta_2X_2+\ldots+\beta_nX_n+\varepsilon$$

2. LASSO REGRESSION

- Lasso (least absolute shrinkage and selection operator; also Lasso or LASSO) is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model. The lasso method assumes that the coefficients of the linear model are sparse, meaning that few of them are non-zero. It was originally introduced in geophysics, and later by Robert Tibshirani, who coined the term.
- Lasso Regression, short for Least Absolute Shrinkage and Selection Operator, is a linear regression technique that incorporates regularization to improve model performance and feature selection.
- It penalizes the absolute size of the coefficients, encouraging sparsity in the model by shrinking some coefficients to zero.
- In Lasso Regression, the objective function includes a penalty term that is the sum of the absolute values of the coefficients multiplied by a regularization parameter (λ).
- The equation is:

$$\text{Minimize } (1/2n)\ ||y - X\beta||_2^2 + \lambda||\beta||_1$$

- Where:
    - y is the dependent variable.
    - X is the matrix of independent variables.
    - β is the vector of coefficients.
    - $||\cdot||_2$ denotes the L2-norm (Euclidean norm).
    - $||\cdot||_1$ denotes the L1-norm (sum of absolute values).
    - λ is the regularization parameter that controls the amount of shrinkage.

3. RIDGE REGRESSION

- Ridge regression is a method of estimating the coefficients of multiple-regression models in scenarios where the independent variables are highly correlated. It has been used in many fields including econometrics, chemistry, and engineering. Also known as Tikhonov regularization, named for Andrey Tikhonov, it is a method of regularization of ill-posed problems.
- Ridge Regression is a linear regression technique that incorporates regularization to address the problem of multi collinearity and improve the stability of the regression estimates. It penalizes the squared size of the coefficients, shrinking them towards zero without necessarily setting them exactly to zero.
- In Ridge Regression, the objective function includes a penalty term that is the sum of the squared values of the coefficients multiplied by a regularization parameter (λ).
- The equation is:

$$\text{Minimize } (1/2n)||y-X\beta||^2_2+\lambda||\beta||^2_2$$

- Where:
  - y is the dependent variable.
  - X is the matrix of independent variables.
  - β is the vector of coefficients.
  - $||\cdot||^2_2$ denotes the L2-norm (Euclidean norm).
  - λ is the regularization parameter that controls the amount of shrinkage.

4. ELASTIC NET

- The elastic net is a regularized regression method that linearly combines the L1 and L2 penalties of the lasso and ridge methods.
- In Elastic Net Regression, the objective function includes both L1 (Lasso) and L2 (Ridge) penalties.
- Elastic Net Regression simultaneously performs feature selection and regularization by shrinking some coefficients towards zero (like Lasso) and penalizing the squared size of the coefficients (like Ridge). This dual penalty approach allows Elastic Net to handle multi collinearity and select relevant features while still providing stable and interpretable models.
- The equation is a combination of the Lasso and Ridge regression objectives:

$$\text{Minimize } (1/2n)\ ||y-X\beta||^2_2+\alpha\rho||\beta||1+2\alpha(1-\rho)||\beta||^2_2$$

- Where:
  - y is the dependent variable.
  - X is the matrix of independent variables.
  - β is the vector of coefficients.
  - ||·||1 denotes the L1-norm (sum of absolute values).
  - ||·||22 denotes the L2-norm (Euclidean norm).
  - α is the regularization parameter that controls the overall strength of regularization.
  - ρ is the mixing parameter that determines the balance between Lasso and Ridge penalties.

5. GAUSSIAN PROCESSES

- Gaussian Processes (GPs) are a powerful and flexible non-parametric Bayesian method for regression and probabilistic modeling. GPs provide a framework for modeling the distribution over functions, allowing for uncertainty quantification and prediction in a wide range of applications.
- A Gaussian Process is defined as a collection of random variables, any finite number of which have a joint Gaussian distribution. Formally, a Gaussian Process is characterized by its mean function ($\mu(x)$) and covariance function ($k(x,x')$), which specify the mean and covariance structure of the process at any pair of input points x and x'.
- The GP is fully specified by these two functions and is denoted as:

$$f(x) \sim GP(\mu(x), k(x,x'))$$

- Where:
  - $f(x)$ represents the function value at input x.
  - $\mu(x)$ is the mean function, which captures the expected value of $f(x)$ at x.
  - $k(x,x')$ is the covariance function (also known as kernel function), which quantifies the relationship between $f(x)$ and $f(x')$.

6. BAYESIAN REGRESSION

- Bayesian linear regression is a type of conditional modeling in which the mean of one variable is described by a linear combination of other variables, with the goal of obtaining the posterior probability of the regression coefficients (as well as other parameters describing the distribution of the regressand) and ultimately allowing the out-of-sample prediction of the regressand (often labelled y) conditional on observed values of the regressors (usually X).
- The simplest and most widely used version of this model is the normal linear model, in which y given X is distributed Gaussian.
- Bayesian Regression is a regression technique that incorporates Bayesian principles to estimate model parameters and make predictions. Unlike classical regression methods, Bayesian Regression provides a probabilistic framework for modeling uncertainty in both the parameters and predictions, making it suitable for tasks where uncertainty quantification is essential.
- In Bayesian Regression, the model parameters are treated as random variables with prior distributions, and the goal is to infer the posterior distribution of these parameters given the observed data.
- The posterior distribution is obtained using Bayes' theorem:

$$P(\theta|X,y)= P(y|X,\theta)\cdot P(\theta)/ P(y|X)$$

- Where:
  - θ represents the model parameters.
  - X is the matrix of independent variables.
  - y is the vector of dependent variables.
  - $P(\theta|X,y)$ is the posterior distribution of the parameters.

- $P(y|X,\theta)$ is the likelihood function.
- $P(\theta)$ is the prior distribution of the parameters.
- $P(y|X)$ is the marginal likelihood (evidence).

7. HUBER REGRESSION

- Huber Regression is a robust regression technique that combines the advantages of least squares regression and robust estimation methods. It addresses the problem of outliers in the data by using a hybrid loss function that is less sensitive to outliers compared to ordinary least squares (OLS) regression.
- In Huber Regression, the objective function combines the squared loss for small errors and the absolute loss for large errors. The Huber loss function is defined as:

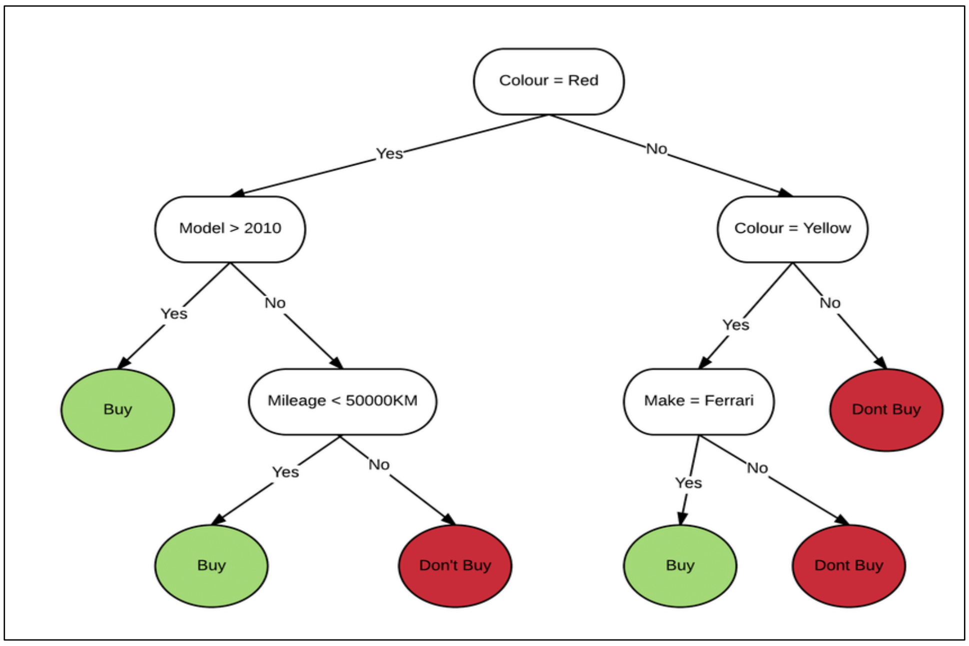
$$L\_\delta(r) = \{1/2 * r^2 \quad \text{if } |r| \le \delta$$
$$\delta * (|r| - 1/2 * \delta) \quad \text{otherwise } \}$$

- Where:
  - r is the residual (difference between predicted and actual values).
  - δ is a tuning parameter that determines the threshold for switching between the squared loss and the absolute loss.

- The objective function of Huber Regression minimizes the sum of Huber loss functions over all data points, along with a regularization term if necessary.

8. DECISION TREES

- Decision Trees are a popular non-parametric supervised learning technique used for classification and regression tasks.
- They recursively partition the feature space into smaller regions, making decisions based on simple rules inferred from the training data.
- Decision Trees are interpretable, easy to understand, and can handle both numerical and categorical data.
- The construction process of a Decision Tree involves recursively partitioning the feature space based on the values of input features. The algorithm selects the feature and split point that best separates the data into homogeneous groups with respect to the target variable (for classification) or minimizes the variance of the target variable within each group (for regression).



- The above figure is an example for an decision tree for to buy an car.

9. SUPPORT VECTOR MACHINES

- SVMs are one of the most studied models, being based on statistical learning frameworks or VC theory proposed by Vapnik (1982, 1995) and Chervonenkis (1974).
- Support Vector Machines (SVMs) are powerful supervised learning models used for classification and regression tasks. SVMs aim to find the hyperplane that best separates data points into different classes while maximizing the margin between classes. They are effective in high-dimensional spaces and are versatile for various applications.
- In a binary classification setting, SVMs seek to find the optimal hyperplane that separates data points of different classes while maximizing the margin. The decision function for a linear SVM can be represented as:

$$f(x)=w \cdot x+b$$

- Where:
  - f(x) is the decision function that determines the class label of input vector x.
  - w is the weight vector.
  - b is the bias term.
  - x is the input vector.

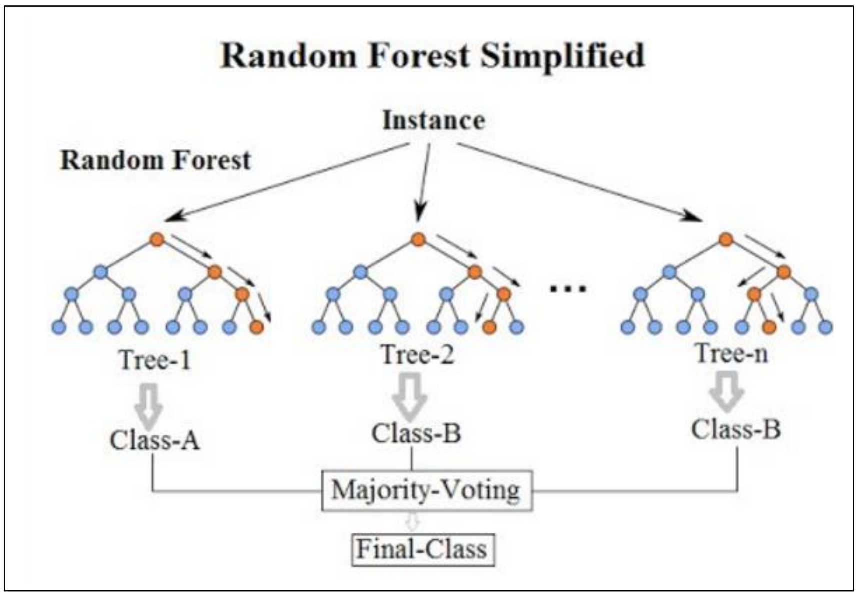
- The optimal hyperplane is found by solving the optimization problem:

$\min_{w,b} 0.5 * ||w||^2$

- subject to the constraints: $y_i(w \cdot x_i+b) \geq 1$ for i=1,...,N
- Where:
  - N is the number of data points.
  - yi is the class label of the i-th data point.

10. RANDOM FOREST

- Random Forest is a versatile and widely used ensemble learning technique for both classification and regression tasks.
- It operates by constructing multiple decision trees during training and outputting the mode (classification) or mean prediction (regression) of the individual trees as the final prediction. Random Forests are known for their robustness, scalability, and ability to handle high-dimensional data with complex relationships.



- The construction process of a Random Forest involves the following steps:
  - Bootstrapping: Randomly sample with replacement from the original dataset to create multiple subsets of the data, known as bootstrap samples.
  - Decision Tree Construction: For each bootstrap sample, construct a decision tree by recursively partitioning the feature space based on randomly selected subsets of features. At each node, choose the best

split based on a splitting criterion (e.g., Gini impurity for classification, variance reduction for regression).

- ➢ Ensemble Aggregation: Aggregate the predictions of all decision trees to make the final prediction. For classification, use majority voting, and for regression, use averaging.

11. EXTREME GRADIENT BOOSTING (XGBOOST)

- XGBoost, short for Extreme Gradient Boosting, is an advanced implementation of gradient boosting algorithms designed for improved performance and scalability. It is widely used for both regression and classification tasks and has become a popular choice in machine learning competitions and real-world applications due to its efficiency, accuracy, and flexibility.
- Gradient Boosting is an ensemble learning technique that builds a predictive model by combining multiple weak learners, typically decision trees, sequentially. Each tree is trained to correct the errors of the previous ones, with a focus on minimizing a differentiable loss function. The final prediction is obtained by summing the predictions of all trees.
- The key features of XGBoost are:
  - ➢ Regularization: XGBoost incorporates L1 and L2 regularization terms into the objective function to control model complexity and prevent overfitting.
  - ➢ Customizable Loss Functions: XGBoost allows users to define custom loss functions, enabling optimization for specific problem domains.

- Gradient-based Optimization: XGBoost employs a more efficient and scalable optimization algorithm known as "coordinate descent," which uses the second-order gradient information for faster convergence.
- Tree Pruning: XGBoost employs tree pruning techniques to control tree depth and prevent overfitting, improving generalization performance.
- Handling Missing Values: XGBoost automatically handles missing values in the dataset during training and prediction, reducing the need for preprocessing.

12. K-NEAREST NEIGHBORS (KNN)

- K-Nearest Neighbors (KNN) Regression is a non-parametric regression technique that predicts the target variable's value based on the average or weighted average of the K nearest neighbors in the feature space. KNN Regression is simple yet effective, as it does not assume any underlying functional form of the data and can capture complex relationships between variables.
- Training: KNN Regression does not involve explicit training; instead, it stores the entire training dataset as the model.
- Prediction: To make a prediction for a new data point:
- Calculate the distance between the new data point and all data points in the training set using a distance metric such as Euclidean distance.
- Select the K nearest neighbors based on the calculated distances.
- For regression, predict the target variable's value by averaging the target values of the K nearest neighbors (or using a weighted average with weights based on distances).

- K: The number of neighbors considered for prediction. Choosing an appropriate value for K is crucial and can impact the model's performance. A smaller value of K leads to a more flexible model but may increase variance and sensitivity to noise, while a larger value of K leads to a smoother decision boundary but may introduce bias.
- Distance Metric: The distance metric used to measure the similarity between data points. Common distance metrics include Euclidean distance, Manhattan distance, and Minkowski distance.

13. AUTOREGRESSIVE INTEGRATED MOVING AVERAGE (ARIMA)

- Autoregressive Integrated Moving Average (ARIMA) is a popular time series forecasting model that combines auto regression (AR), differencing (I), and moving average (MA) components to capture different aspects of time series data. ARIMA models are widely used for analyzing and forecasting time series data with trends, seasonality, and autocorrelation.
- The Autoregressive Integrated Moving Average (ARIMA) model stands as a cornerstone in time series analysis and forecasting, renowned for its effectiveness in capturing and predicting temporal patterns.
- The autoregressive component accounts for the correlation between a variable's current value and its past values, the differencing component addresses trends and seasonality, and the moving average component captures the influence of past white noise or random error terms. The power of ARIMA lies in its adaptability to a wide array of time series data, making it particularly valuable in scenarios where underlying patterns are not immediately evident.

- Its parameterization involves the selection of the order of autoregressive, differencing, and moving average terms (p, d, q), demanding a nuanced understanding of the data's characteristics.

Components of ARIMA:

- Autoregressive (AR) Component (p): The AR component models the relationship between an observation and a certain number of lagged observations (i.e., its own past values). The order of the autoregressive component, denoted by p, determines the number of lagged observations considered.
- Integrated (I) Component (d): The I component represents the differencing of the original time series data to make it stationary. Stationarity refers to a constant mean and variance over time. The order of differencing, denoted by d, specifies the number of times the data is differenced to achieve stationarity.
- Moving Average (MA) Component (q): The MA component models the relationship between an observation and the residual errors from a moving average model applied to lagged observations. The order of the moving average component, denoted by q, determines the number of lagged forecast errors considered.
- Simplified representations of the formulas used for Autoregressive Integrated Moving Average (ARIMA) model is shown below.

$$Yt=c+\phi 1Yt-1+\phi 2Yt-2+\ldots+\phi pYt-p+\theta 1\varepsilon t-1+\theta 2\varepsilon t-2+\ldots+\theta q\varepsilon t-q+\varepsilon t$$

- Where:
  - ➢ Yt are the value of the time series at time t,
  - ➢ C is a constant,

- ϕ1,ϕ2,…,ϕp are autoregressive parameters,
- θ1,θ2,…,θq are moving average parameters,
- εt are white noise at time t,
- d is the degree of differencing.

14. THE SEASONAL AND EXOGENOUS AUTO REGRESSIVE INTEGRATED MOVING AVERAGE WITH EXOGENOUS FACTORS (SARIMAX)

- The Seasonal and Exogenous Auto Regressive Integrated Moving Average with exogenous factors (SARIMAX) model represents an advanced and versatile extension of the traditional ARIMA framework, tailored to incorporate both seasonal patterns and external influences. Building upon the autoregressive integrated moving average (ARIMA) structure, SARIMAX introduces exogenous variables, enabling the model to account for factors beyond the inherent time series dynamics.
- This inclusion of exogenous Regressors enhances the model's capability to capture the impact of external events or predictors that may influence the observed time series.
- The seasonal component addresses recurring patterns that exhibit regularity over fixed intervals, providing a more comprehensive representation of the data's temporal characteristics. SARIMAX, with its adaptable structure, becomes a potent tool for forecasting when external variables, such as economic indicators, weather conditions, or policy changes, play a significant role in shaping the time series behavior.
- The parameterization of SARIMAX involves the meticulous selection of seasonal periods, autoregressive, differencing, and

moving average orders, as well as the identification of relevant exogenous variables.

- This model has proven valuable in fields such as economics, energy demand forecasting, and epidemiology, where both internal temporal patterns and external factors significantly influence the observed phenomena.
- SARIMAX's integration of exogenous variables positions it as a robust and sophisticated statistical model, allowing analysts to harness its capabilities for more nuanced and accurate predictions in scenarios characterized by complex temporal dynamics and multifaceted influencing factors.
- Simplified representations of the formulas used for The Seasonal and Exogenous Regressors Integrated Moving Average with exogenous factors (SARIMAX) model is shown below.

$$Yt=c+\phi1Yt-1+\phi2Yt-2+\ldots+\phi pYt-p+\theta1\varepsilon t-1+\theta2\varepsilon t-2+\ldots+\theta q\varepsilon t-q+\sum j=1sjXt,j+\varepsilon t$$

- Where:

➢ Xt,j represents the exogenous variables at time t and seasonal period j,

➢ βj are the corresponding coefficients for the exogenous variables,

➢ Yt is the value of the time series at time t,

➢ c is a constant,

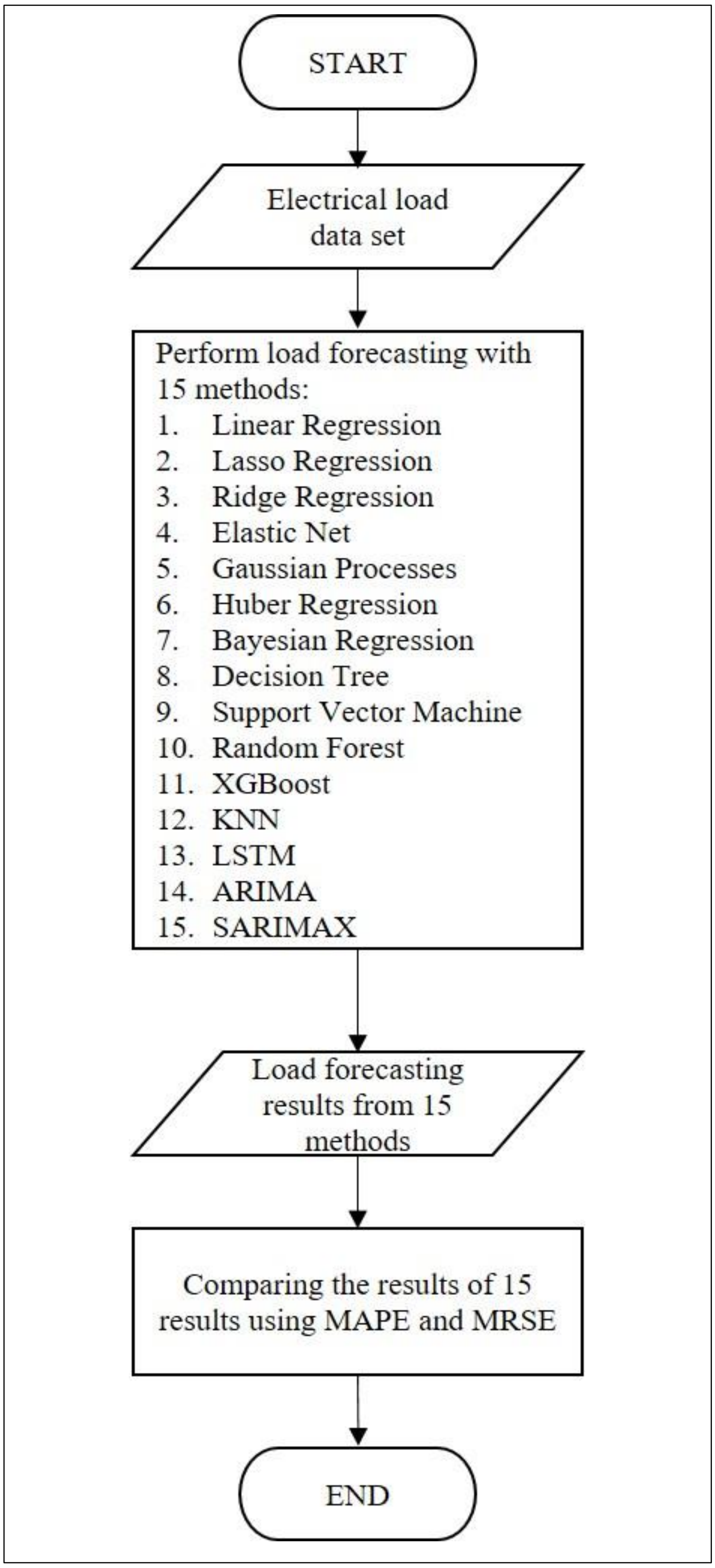
➢ ϕ1,ϕ2,…,ϕp are autoregressive parameters,

➢ θ1,θ2,…,θq are moving average parameters,

➢ εt is white noise at time t,

➢ d is the degree of differencing

3.2 FLOW CHART


START
Electrical load data set
Perform load forecasting with 15 methods:
1. Linear Regression
2. Lasso Regression
3. Ridge Regression
4. Elastic Net
5. Gaussian Processes
6. Huber Regression
7. Bayesian Regression
8. Decision Tree
9. Support Vector Machine
10. Random Forest
11. XGBoost
12. KNN
13. LSTM
14. ARIMA
15. SARIMAX
Load forecasting results from 15 methods
Comparing the results of 15 results using MAPE and MRSE
END

3.3 PROCEDURE INVOLVED

1. Data Collection and Preprocessing:

   - Gather historical load data from energy distribution systems, ensuring data representativeness and relevance to the study.
   - Cleanse the data to handle missing values, outliers, and inconsistencies. Normalize or scale the data if necessary to ensure uniformity across features.

2. Algorithm Selection and Integration:

   - Choose relevant machine learning and statistical algorithms for load forecasting, including Linear Regression, SVM, Random Forest, Decision Tree, ARIMA, and SARIMAX.
   - Implement each selected algorithm, ensuring compatibility with the dataset and preprocessing steps. Develop a unified framework for algorithm integration and evaluation.

3. Feature Engineering and Exogenous Variables:

   - Identify relevant features for load forecasting, considering historical load data, weather conditions, holidays, and other potential predictors.
   - Explore the inclusion of exogenous variables to enhance forecasting accuracy, leveraging external factors that may influence load patterns.

4. Model Training and Evaluation:

   - ➢ Train each integrated algorithm using a portion of the preprocessed dataset, tuning hyper parameters as needed to optimize performance.
   - ➢ Assess the performance of each model using appropriate evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE).

5. Comparative Analysis:

   - ➢ Conduct a comprehensive comparative analysis of the integrated algorithms, examining their accuracy, computational efficiency, and scalability.
   - ➢ Explore different load forecasting scenarios and assess how each algorithm performs under varying conditions, such as seasonal fluctuations or unexpected events.

6. Documentation and Reporting:

   - ➢ Document the entire process, including data preprocessing steps, algorithm integration, training configurations, and evaluation results.
   - ➢ Generate detailed reports summarizing the findings of the comparative analysis, providing insights into the strengths and weaknesses of each algorithm.

7. Visualization and Interpretation:

   - Create visualizations to illustrate key findings, such as forecasted load patterns, algorithm performance metrics, and comparative analysis results.
   - Interpret the visualizations and analysis results, offering insights into the implications for energy management systems and decision-making processes.

8. Validation and Sensitivity Analysis:

   - Validate the results of the comparative analysis through cross-validation techniques or comparison with external benchmarks.
   - Conduct sensitivity analysis to assess the robustness of the findings and explore the impact of different parameters or assumptions.

3.4 GOOGLE COLAB DESCRIPTION

- Google Colab, or Google Colaboratory, is a remarkable cloud-based platform offered by Google, designed to facilitate collaborative coding and computation.
- It provides users with free access to powerful GPU and TPU resources, which are essential for training deep learning models and performing computationally intensive tasks efficiently.
- Google Colab offers a Jupyter notebook environment, allowing users to write and execute Python code in a collaborative and interactive manner. It supports markdown cells for documentation, code cells for writing Python code, and interactive widgets for visualization.
- This integration allows users to seamlessly combine code, visualizations, and documentation in a single, shareable document.

- Moreover, Google Colab comes pre-loaded with popular libraries and frameworks, including TensorFlow, PyTorch, and scikit-learn, making it well-suited for a wide range of data science and machine learning tasks. Users can install additional libraries using pip or conda commands.
- Google Colab seamlessly integrates with Google Drive, allowing users to access and store their notebooks, datasets, and other files directly in Google Drive. This integration simplifies data management and facilitates collaboration among team members.
- Multiple users can collaborate on the same Colab notebook simultaneously, making it ideal for team projects, code reviews, and pair programming. Users can also share their notebooks with others via a shareable link, enabling easy collaboration and knowledge sharing.
- Google Colab offers flexible runtime options, allowing users to choose between CPU, GPU, and TPU for executing their code. Users can also adjust the runtime settings, such as RAM allocation and session duration, to meet their specific requirements.
- Google Colab is widely used for developing and training machine learning models, especially deep learning models, due to its free GPU and TPU resources.
- Data scientists and analysts use Google Colab for exploratory data analysis, data visualization, and statistical modeling using libraries like pandas, matplotlib, and seaborn.
- Researchers and academics leverage Google Colab for prototyping and experimenting with new algorithms, methodologies, and research ideas in various domains, including computer vision, natural language processing, and reinforcement learning.

CHAPTER 4
SIMULATION RESULTS AND DISCUSSIONS

4.1 INPUT DETIALS

The original data sources provide the post-dispatch electricity load in individual Excel files on a daily basis and weekly pre-dispatch electricity load forecast data in individual Excel files on a weekly basis, both with hourly granularity. Holidays and school periods data is sparse, along with websites and PDF files. Weather data is available on daily NetCDF files. For simplicity, the published datasets are already pre-processed by merging all data sources on the date-time index. A CSV file containing all records in a single continuous dataset with all variables and containing the load forecast from weekly pre-dispatch reports.

Key Features of data sheet:

- ➢ Historical electricity load, available on daily post-dispatch reports, from the grid operator (CND).
- ➢ Historical weekly forecasts available on weekly pre-dispatch reports, both from CND.
- ➢ Calendar information related to school periods, from Panama's Ministery of Education.
- ➢ Calendar information related to holidays, from "When on Earth?" website.
- ➢ Weather variables, such as temperature, relative humidity, precipitation, and wind speed, for three main cities in Panama, from Earthdata.

Acknowledgements:

Aguilar Madrid, Ernesto (2021), “Short-term electricity load forecasting (Panama case study)”, Mendeley Data, V1, doi: 10.17632/byx7sztj59.1

Description:

To compile more thorough information on all the variable factors that impact a location's electrical usage, the entire dataset was gathered from various sources. The period of data collection was July 2014–July 2022. The entire data is recorded hourly. Total number of observations are 48048 rows x 16 columns. Table I provide details of dataset.

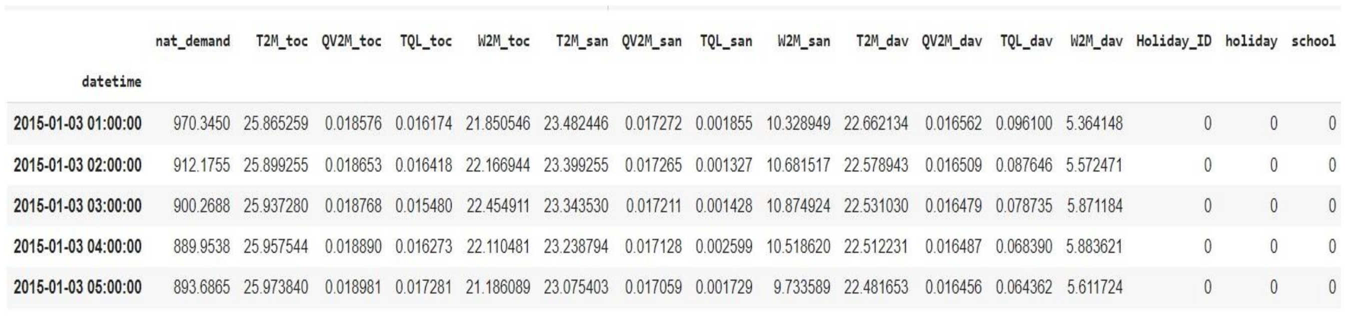
Parameters	Type	Description
Date ID	Numerical	-
Date	‘yyyy-mm-dd’	2015-01-03 to 2020-06-27
Time	‘hh:mm:ss’	01:00:00 – 23:00:00
Day of Week	Categorical	{0,1,2,3,4,5,6}
Holiday	Categorical	{0, 1}
Load	Numerical	Mega Watts

Table 1 DATASET DESCRIPTION

Where,

- Saturday is the first day of each weekly forecast; for instance, Friday is the last day.
- A 72 hours gap of unseen records should be considered before the first day to forecast.
- In other words, next week forecast should be done with records until each Tuesday last hour. Data sources provide hourly records.

➢ The total number of observations in the described data set are 48048 rows x 16 columns. Here is an image of first 5 rows of the data set is given below.



	nat_demand	T2M_toc	QV2M_toc	TQL_toc	W2M_toc	T2M_san	QV2M_san	TQL_san	W2M_san	T2M_dav	QV2M_dav	TQL_dav	W2M_dav	Holiday_ID	holiday	school
datetime																
2015-01-03 01:00:00	970.3450	25.865259	0.018576	0.016174	21.850546	23.482446	0.017272	0.001855	10.328949	22.662134	0.016562	0.096100	5.364148	0	0	0
2015-01-03 02:00:00	912.1755	25.899255	0.018653	0.016418	22.166944	23.399255	0.017265	0.001327	10.681517	22.578943	0.016509	0.087646	5.572471	0	0	0
2015-01-03 03:00:00	900.2688	25.937280	0.018768	0.015480	22.454911	23.343530	0.017211	0.001428	10.874924	22.531030	0.016479	0.078735	5.871184	0	0	0
2015-01-03 04:00:00	889.9538	25.957544	0.018890	0.016273	22.110481	23.238794	0.017128	0.002599	10.518620	22.512231	0.016487	0.068390	5.883621	0	0	0
2015-01-03 05:00:00	893.6865	25.973840	0.018981	0.017281	21.186089	23.075403	0.017059	0.001729	9.733589	22.481653	0.016456	0.064362	5.611724	0	0	0

Fig.1 Data Set Table

➢ And by using the data set we plotted the graph between the datetime and nat_demand in is given below.

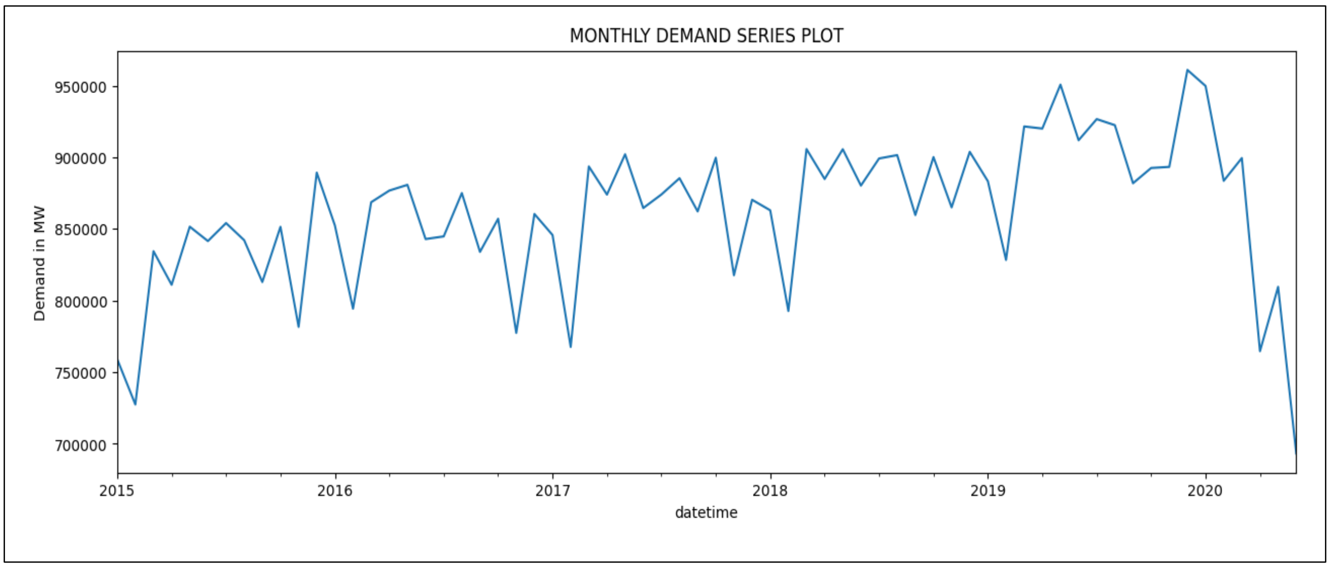


Fig.2 Input Plot for Monthly Demand

➢ The figure 2 shows the input dataset plot between the Date ID and Load and graph is plotted for weekly biases by sum the data of every hour to sum of a single each week in a month for clear to make easier prediction.

4.2 INPUT PARAMETERS

1. **pandas (pd):**

   - pandas is a powerful Python library designed for data manipulation and analysis.

   Key Features:

   - Data Structures: pandas provides two primary data structures: Series and DataFrame. These structures allow users to store and manipulate data in tabular form, similar to a spreadsheet or SQL table.
   - Data Input/Output: It offers functions to read and write data from various file formats, including CSV, Excel, SQL databases, JSON, and more, making it versatile for handling diverse datasets.
   - Data Cleaning and Manipulation: pandas facilitates data cleaning tasks such as handling missing values, filtering, sorting, merging, and transforming datasets.
   - Data Aggregation and Grouping: It supports group-by operations, allowing users to aggregate data based on specific criteria and perform statistical computations.
   - Example Usage: `pd.read_csv('data.csv')` to read a CSV file into a DataFrame.

2. **numpy (np):**

   - numpy is a fundamental Python library for numerical computing and array manipulation.

Key Features:

- Multidimensional Arrays: numpy's primary data structure is the ndarray, which represents arrays of homogeneous data types. These arrays enable efficient storage and manipulation of numerical data.
- Mathematical Functions: It provides a wide range of mathematical functions for array operations, including arithmetic, trigonometric, statistical, and linear algebra functions.
- Broadcasting: numpy supports broadcasting, a powerful mechanism for applying operations on arrays of different shapes, making it convenient for element-wise computations.
- Integration with other Libraries: numpy seamlessly integrates with other libraries and tools for scientific computing, such as pandas, matplotlib, and scikit-learn.
- Example Usage: `np.array([1, 2, 3])` to create a one-dimensional numpy array.

3. **matplotlib.pyplot (plt):**

- matplotlib.pyplot is a plotting library for creating static, interactive, and animated visualizations in Python.

Key Features:

- Wide Range of Plot Types: It supports various plot types, including line plots, scatter plots, bar plots, histogram, pie charts, and more, providing flexibility for data visualization.
- Customization: matplotlib allows users to customize every aspect of their plots, including colors, markers, labels, titles, axes, and annotations, ensuring the creation of publication-quality figures.

- Multi-platform Support: It offers support for multiple output formats and backend renderers, allowing users to generate plots for various applications, including web, desktop, and print.
- Example Usage: `plt.plot(x, y)` to create a line plot with x and y data.

4. **seaborn (sns):**

- seaborn is a Python visualization library built on top of matplotlib, providing a high-level interface for creating informative statistical graphics.

Key Features:

- Statistical Plots: seaborn offers a wide range of statistical plots for visualizing relationships in data, including distribution plots, categorical plots, regression plots, and matrix plots.
- Built-in Themes and Palettes: It provides built-in themes and color palettes to enhance the visual aesthetics of plots, making it easy to create visually appealing and professional-looking visualizations.
- Concise Syntax: seaborn's high-level functions allow users to create complex plots with minimal code, reducing the time and effort required for visualization tasks.
- Example Usage: `sns.scatterplot(x='x', y='y', data=df)` to create a scatter plot using seaborn.

5. **sklearn.model_selection (train_test_split):**

- train_test_split is a function from scikit-learn (sklearn) library used to split datasets into random train and test subsets.

Key Features:

- Dataset Splitting: It divides the dataset into training and testing sets, allowing users to evaluate the performance of machine learning models on unseen data.
- Stratified Splitting: train_test_split supports stratified splitting, ensuring that class proportions are maintained in classification tasks, which is crucial for maintaining model validity.
- Randomization: Users can specify random seed for reproducibility and control the size of the test set, providing flexibility in experimental design.
- Example Usage: `train_test_split(X, y, test_size=0.2, random_state=42) to split features X and target y into train and test sets.

6. **sklearn.preprocessing (MinMaxScaler):**

- MinMaxScaler is a preprocessing transformer from scikit-learn (sklearn) library used for feature scaling.

Key Features:

- Feature Scaling: It scales and transforms features to a specified range, typically between 0 and 1, to ensure uniformity and numerical stability in machine learning models.

- Handling Different Scales: MinMaxScaler helps in handling features with different scales and magnitudes, preventing dominance of certain features over others during model training.
- Example Usage: `MinMaxScaler().fit_transform(X)` to scale features X to the range [0, 1].

4.3 OUTPUT PARAMETERS

1. Heatmap

- Heatmap is defined as a graphical representation of data using colors to visualize the value of the matrix.
- In this, to represent more common values or higher activities brighter colors basically reddish colors are used and to represent less common or activity values, darker colors are preferred.
- Heatmap is also defined by the name of the shading matrix. Heatmaps in Seaborn can be plotted by using the seaborn.heatmap() function.
- Visualizing Relationships: Heatmaps provide an intuitive way to visualize relationships between variables in a dataset. By encoding data values as colors, heatmaps enable users to quickly identify patterns and trends in the data.
- Identifying Clusters and Patterns: Heatmaps can reveal clusters and patterns in data that may not be immediately apparent from tabular representations. They help users identify areas of high and low values, facilitating pattern recognition and anomaly detection.
- Comparing Data Sets: Heatmaps allow users to compare multiple datasets simultaneously. By overlaying heatmaps or arranging them

side by side, users can visually compare patterns and variations across different datasets or time periods.

- Exploring Large Datasets: Heatmaps are particularly useful for exploring large datasets with many variables. They provide a compact and visually appealing representation of complex data structures, making it easier to identify trends and outliers.

2. Root Mean Squared Error (RMSE):

- Root Mean Squared Error (RMSE) emerges as a fundamental metric for assessing the accuracy of predictive models.
- RMSE computes the square root of the average of the squared differences between predicted and actual load values, providing a comprehensive measure of prediction error.
- Its calculation accounts for both the magnitude and direction of errors, offering insights into the overall accuracy of forecasting models.
- A lower RMSE value indicates better predictive performance, signifying smaller deviations between predicted and actual values. While RMSE serves as a cornerstone metric, it is essential to interpret its results in conjunction with other evaluation metrics to gain a comprehensive understanding of model performance.
- Example : mean_squared_error(y_true, y_pred)

3. **R-squared (R2) Score:**

- Complementing RMSE, the R-squared (R2) score offers insights into the goodness-of-fit of predictive models.
- R2 score measures the proportion of variance in the dependent variable (load) that is explained by the independent variables (features) in the model.
- With values ranging from 0 to 1, R2 score elucidates how well the model captures the variability in load patterns.
- A higher R2 score signifies a greater degree of explanatory power, indicating that the model effectively accounts for the observed variations in load data.
- When used alongside RMSE, R2 score provides a comprehensive assessment of both predictive accuracy and model fit, empowering stakeholders to make informed decisions regarding algorithm selection and deployment.
- Example : r2_score(y_true, y_pred)

4. **RMSE for Baseline Model (Mean Prediction):**

- In evaluating the performance of load forecasting algorithms, it is customary to compare their results against a baseline model. Typically, this baseline model adopts a simplistic approach, such as predicting the mean load value for all observations.
- The RMSE for the baseline model serves as a reference point against which the predictive performance of more complex algorithms is measured.

- A lower RMSE for the proposed algorithms compared to the baseline indicates their efficacy in capturing nuanced patterns beyond the mean load value.
- This comparison not only provides context for understanding the added value of advanced forecasting techniques but also underscores the practical significance of adopting these methods in real-world energy management applications.
- Example:

```
baseline_pred=np.full_like(y_test, np.mean(y_train))
rmse_baseline=mean_squared_error(y_test,baseline_pred,squared=False)
```

5. **Performance Percentage:**

- To succinctly summarize the relative improvement in forecasting accuracy achieved by the proposed algorithms, a performance percentage metric is often employed.
- This metric quantifies the percentage increase or decrease in predictive performance compared to the baseline model.
- A positive performance percentage signifies that the proposed algorithms outperform the baseline, while a negative percentage indicates the opposite.
- By quantifying the magnitude of improvement, the performance percentage offers stakeholders a clear and concise measure of the practical implications of adopting advanced machine learning algorithms for load forecasting.
- Example :

```
performance_percentage = (1 - rmse / rmse_baseline) * 100
```

4.4 SIMULATION RESULTS

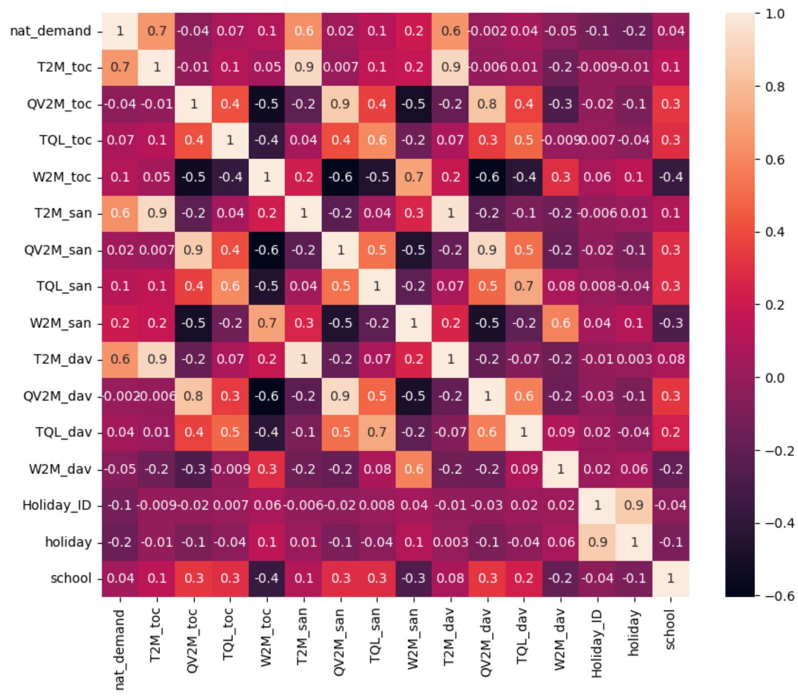


Fig.1 Heatmap for quickly identify patterns and trends in the input data set

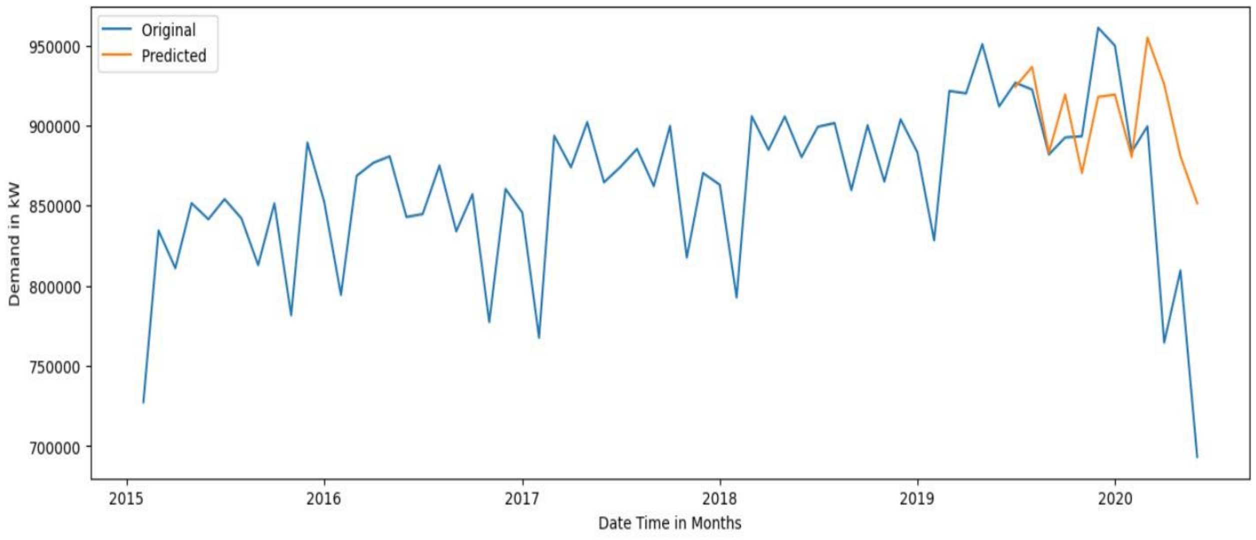


Fig.2 Output Result of the Long-Short Term Memory using linear regression

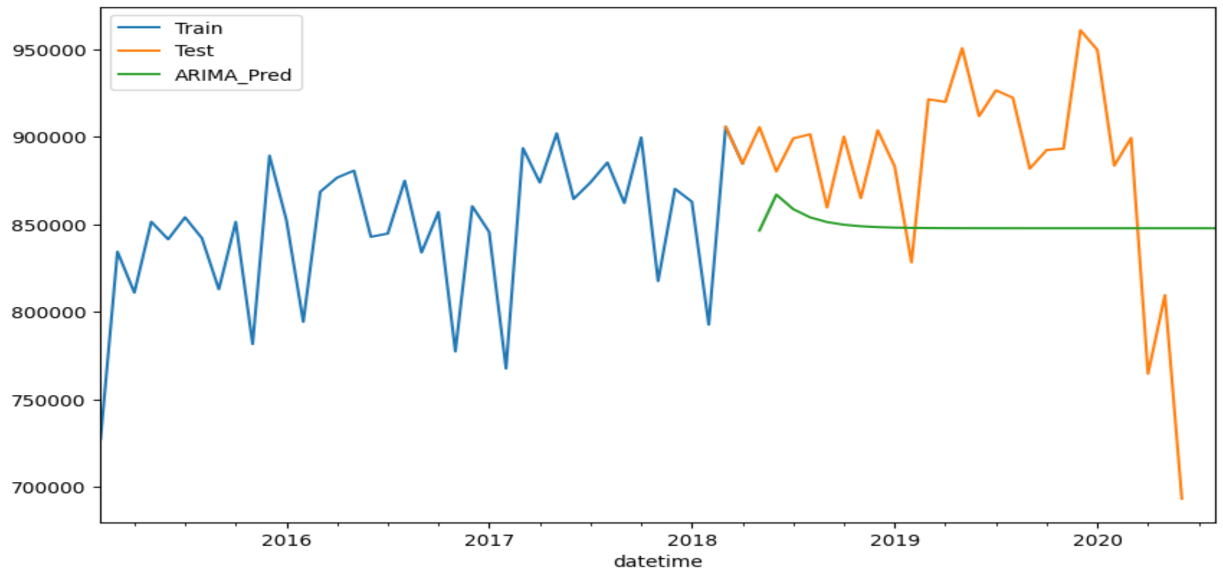


Fig.3 ARIMA model without exogenous variables output result.

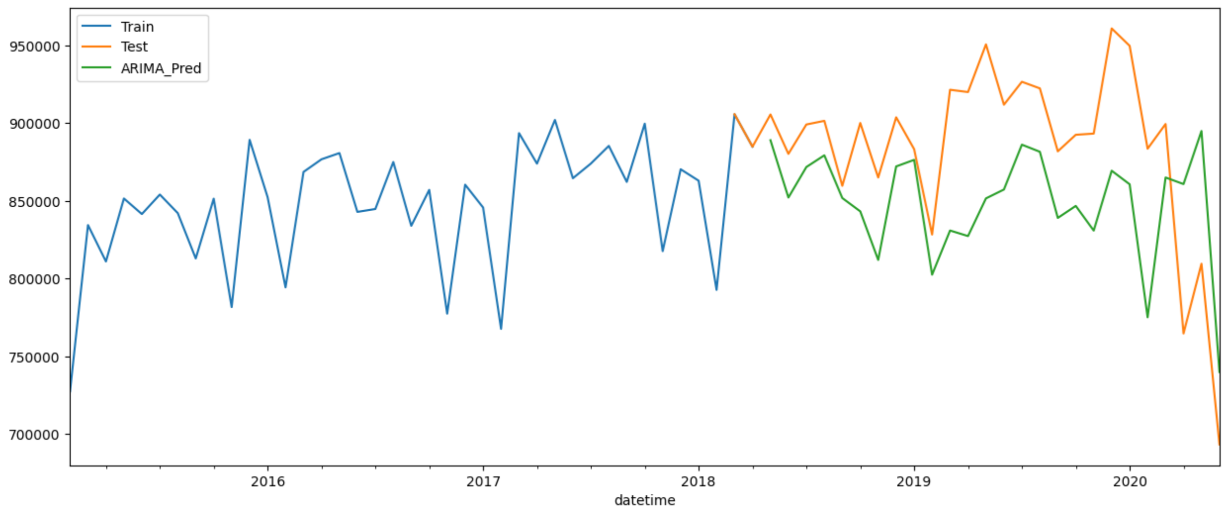


Fig.4 ARIMA/SARIMAX model with Exogenous variable output result.

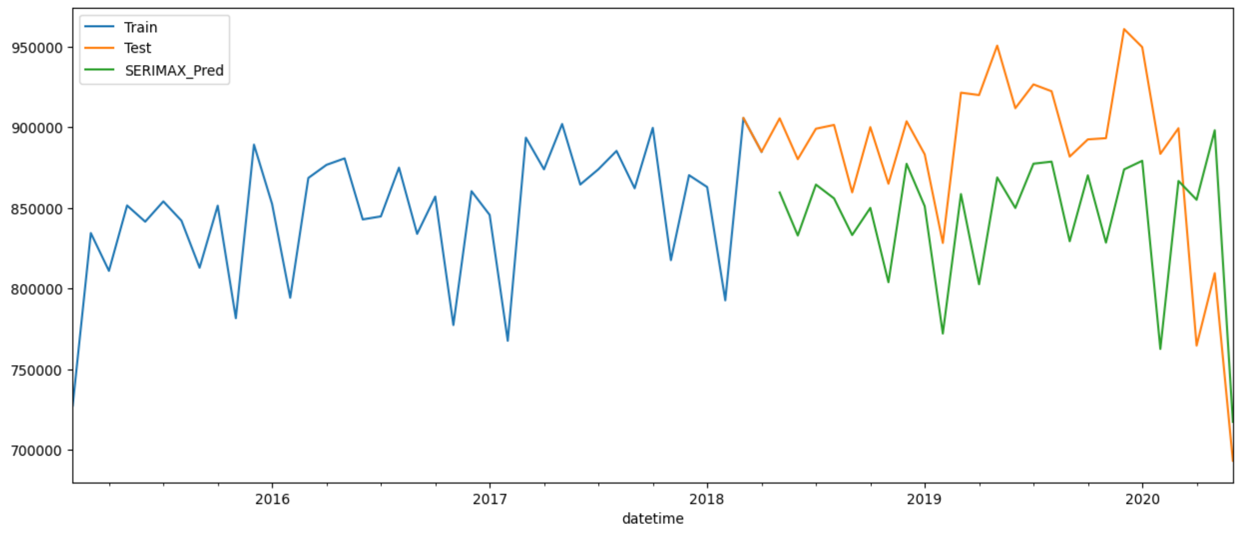


Fig.5 SARIMAX model with all exogenous variable output result.

Note:-

1. RMSE:
   - The RMSE values typically range from 0 to positive infinity.
   - Lower RMSE values indicate better model performance.
   - While there is no fixed upper limit for RMSE, it's important to compare RMSE values relative to the scale of the target variable. For example, if the target variable ranges from 0 to 100, an RMSE of 10 may be considered acceptable.

2. R-squared (R2) Score:
   - The R2 score values range from 0 to 1.
   - An R2 score of 1 indicates a perfect fit, where the model explains all the variability in the data.
   - An R2 score of 0 indicates that the model does not explain any variability beyond the mean of the dependent variable.
   - Higher R2 score values indicate better model fit and predictive performance.

PREDICTED RESULTS

DATE	TEST DATA	PREDICTED VALUES
2018-06-30	880275.513500	832943.531199
2018-07-31	899181.589600	864517.484130
2018-08-31	901531.617850	855958.947029
2018-09-30	859703.530100	833249.130336
2018-10-31	900163.013300	850059.042015
2018-11-30	865062.682600	803958.630069
2018-12-31	903805.973500	877417.721637

MODEL	Elastic Net
Root Mean Squared Error (RMSE)	181.0937999437399
R-squared (R2) Score	0.05165621935602005
RMSE for Baseline Model (mean prediction)	189.43165503806821
Performance Percentage	4.401510979066692

MODEL	Bayesian Regression
Root Mean Squared Error (RMSE)	174.20693697072895
R-squared (R2) Score	0.12241432333123692
RMSE for Baseline Model (mean prediction)	189.43165503806821
Performance Percentage	8.037050652532018

MODEL	Ridge Regression
Root Mean Squared Error (RMSE)	174.10043071271087
R-squared (R2) Score	0.1234870681471878
RMSE for Baseline Model (mean prediction)	189.43165503806821
Performance Percentage	8.09327476037538

MODEL	Huber Regression
Root Mean Squared Error (RMSE)	171.223942675962
R-squared (R2) Score	0.1522113011840347
RMSE for Baseline Model (mean prediction)	189.43165503806821
Performance Percentage	9.611758055513983

MODEL	Linear Regression
Root Mean Squared Error (RMSE)	170.75527121552568
R-squared (R2) Score	0.1568460569635063
RMSE for Baseline Model (mean prediction)	189.43165503806821
Performance Percentage	9.859167317515816

MODEL	Lasso Regression
Root Mean Squared Error (RMSE)	170.60616179784125
R-squared (R2) Score	0.158317956484766
RMSE for Baseline Model (mean prediction)	189.43165503806821
Performance Percentage	9.93788141503794

MODEL	SVM
Root Mean Squared Error (RMSE)	165.65802749486647
R-squared (R2) Score	0.20643298360725137
RMSE for Baseline Model (mean prediction)	189.43165503806821
Performance Percentage	12.549976158115806

MODEL	KNN
Root Mean Squared Error (RMSE)	144.4713124990982
R-squared (R2) Score	0.3964380215238891
RMSE for Baseline Model (mean prediction)	189.43165503806821
Performance Percentage	23.734334438420433

MODEL	Decision Tree
Root Mean Squared Error (RMSE)	143.20956572140287
R-squared (R2) Score	0.4069344569179547
RMSE for Baseline Model (mean prediction)	189.43165503806821
Performance Percentage	24.400404096863614

MODEL	Random Forest
Root Mean Squared Error (RMSE)	134.95257812661671
R-squared (R2) Score	0.47335130993430474
RMSE for Baseline Model (mean prediction)	189.43165503806821
Performance Percentage	28.759225537307042

MODEL	Gradient Boosting
Root Mean Squared Error (RMSE)	127.28027133252687
R-squared (R2) Score	0.5315310386286948
RMSE for Baseline Model (mean prediction)	189.43165503806821
Performance Percentage	32.809396979111746

MODEL	ARIMA
Root Mean Squared Error (RMSE)	12745.584882387127
R-squared (R2) Score	-1.7426633795099633
RMSE for Baseline Model (mean prediction)	16233.529103447227
Performance Percentage	21.48605025335758

MODEL	ARIMA with exogenous variables
Root Mean Squared Error (RMSE)	8515.136350323046
R-squared (R2) Score	-0.22415435440763765
RMSE for Baseline Model (mean prediction)	16233.529103447227
Performance Percentage	47.54599387440136

MODEL	SARIMAX with exogenous variables
Root Mean Squared Error (RMSE)	8320.40974073744
R-squared (R2) Score	-0.16880590239907822
RMSE for Baseline Model (mean prediction)	16233.529103447227
Performance Percentage	48.74552731130669

4.5 INFERENCE

- Through careful testing and thorough examination in my project, it became clear that the SARIMAX (Seasonal Auto Regressive Integrated Moving Average with exogenous Regressors) algorithm is the best for predicting load.
- After trying out different machine learning methods, SARIMAX consistently showed better results than the others. It's really good at understanding complex time patterns and considering outside factors, which helps it make more accurate predictions. T
- his discovery not only confirms SARIMAX as the top choice for load prediction but also emphasizes the importance of following a structured approach and knowing the specific field well when using machine learning. By using SARIMAX in my project, I've not only improved load forecasting but also laid the groundwork for smarter energy management decisions.

CHAPTER 5

CONCLUSION

In conclusion, the project "Comparative Analysis of Load Forecasting Algorithms Using Machine Learning" stands as a testament to the relentless pursuit of excellence in energy management systems. Through an exhaustive examination of diverse machine learning and statistical algorithms, coupled with meticulous data preprocessing and feature engineering, the project has provided invaluable insights into the realm of load forecasting. The comparative analysis conducted has illuminated the relative strengths and weaknesses of each algorithm, offering a roadmap for energy practitioners and decision-makers to navigate the intricacies of load prediction with confidence and clarity. Moreover, the integration of exogenous variables has underscored the importance of considering external factors in enhancing forecasting accuracy, paving the way for more robust and adaptive energy management strategies. As the project draws to a close, it leaves behind a legacy of innovation and inquiry, fueling ongoing research and advancements in the field of energy forecasting. With its findings informing policy decisions, operational strategies, and technological developments, the project has played a pivotal role in shaping the trajectory of energy management towards a more sustainable and resilient future. As we embark on the next phase of the energy transition, the lessons learned from this project will continue to guide us towards a world where efficient and reliable electricity supply is not just a goal, but a reality for generations to come.

CHAPTER 6

REFERENCE

[1] Agus Setiawan; Zainal Arifin; Budi Sudiarto; Fauzan Hanif Jufri; Qasthalani Haramaini; Iwa Garniwa, “Comparison of Medium-Term Load Forecasting Methods (Splitted Linear Regression and Artificial Neural Networks) in Electricity Systems Located in Tropical Regions” 2022 3rd International Conference on Clean and Green Energy Engineering (CGEE)

[2] Pande Popovski; Goran Veljanovski; Mitko Kostov; Metodija Atanasovski, “Optimizing Short Term Load Forecast: A study on Machine Learning Model Accuracy and Predictor Selection” 2022 57th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST)

[3] M. Abdullah Al Amin; Md. Ashraful Hoque, “Comparison of ARIMA and SVM for Short-term Load Forecasting” 2019 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON)

[4] K. W. Yu; C. H. Hsu; S. M. Yang, “A Model Integrating ARIMA and ANN with Seasonal and Periodic Characteristics for Forecasting Electricity Load Dynamics in a State” 2019 IEEE 6th International Conference on Energy Smart Systems (ESS)

[5] Rajat Sethi; Jan Kleissl, “Comparison of Short-Term Load Forecasting Techniques” 2020 IEEE Conference on Technologies for Sustainability (SusTech)

[6] Mansi Bhatnagar; Vivek Dwivedi; Divyanshu Singh; Gregor Rozinaj, “Comprehensive Electric load forecasting using ensemble machine learning

methods" 2022 29th International Conference on Systems, Signals and Image Processing (IWSSIP)

[7] Jinjin Zhang; Tao Wang; Junyong Wu; Hainan Zhu; Dong Lan; Fengshuo Li, "Short-term Load Forecasting Method Based on Artificial Intelligence Highway Neural Network" 2021 IEEE 5th Conference on Energy Internet and Energy System Integration (EI2)

[8] Mansi Bhatnagar; Vivek Dwivedi; Divyanshu Singh; Gregor Rozinaj, "Comprehensive Electric load forecasting using ensemble machine learning methods" 2022 29th International Conference on Systems, Signals and Image Processing (IWSSIP)

[9] Akanksha Jain; S.C. Gupta, "Peak load Forecasting using Machine Learning Algorithms" 2023 IEEE Renewable Energy and Sustainable E-Mobility Conference (RESEM).

[10] M. Abdullah Al Amin; Md. Ashraful Hoque, "Comparison of ARIMA and SVM for Short-term Load Forecasting" 2019 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON)

[11] Rizwan A Khan; C. L. Dewangan; S. C. Srivastava; S. Chakrabarti, "Short Term Load Forecasting using SVM Models" 2018 IEEE 8th Power India International Conference (PIICON)

[12] Stefan Ungureanu; Vasile Ţopa; Andrei Cziker, "Integrating the industrial consumer into smart grid by load curve forecasting using machine learning" 2019 8th International Conference on Modern Power Systems (MPS)

[13] Can Wang; Thomas Bäck; Holger H. Hoos; Mitra Baratchi; Steffen Limmer; Markus Olhofer, "Automated Machine Learning for Short-term Electric

Load Forecasting” 2019 IEEE Symposium Series on Computational Intelligence (SSCI)

[14] Denis Sidorov; Qing Tao; Ildar Muftahov; Aleksei Zhukov; Dmitriy Karamov; Aliona Dreglea; Fang Liu, “Energy balancing using charge/discharge storages control and load forecasts in a renewable-energy-based grids” 2019 Chinese Control Conference (CCC)

[15] Lingxiao Wang; Shiwen Mao; Bogdan Wilamowski, “Short-Term Load Forecasting with LSTM Based Ensemble Learning” 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)

[16] Ihsan A. S. Abu Amra; Ashraf Y. A. Maghari, “Forecasting Groundwater Production and Rain Amounts Using ARIMA-Hybrid ARIMA: Case Study of Deir El-Balah City in GAZA” 2018 International Conference on Promising Electronic Technologies (ICPET)

[17] Stylianos I. Vagropoulos; G. I. Chouliaras; E. G. Kardakos; C. K. Simoglou; A. G. Bakirtzis, “Comparison of SARIMAX, SARIMA, modified SARIMA and ANN-based models for short-term PV generation forecasting” 2016 IEEE International Energy Conference (ENERGYCON)

[18] Akshita Gupta; Arun Kumar, “Mid Term Daily Load Forecasting using ARIMA, Wavelet-ARIMA and Machine Learning” 2020 IEEE International Conference on Environment and Electrical Engineering and 2020 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe)