Forecasting of Household Electrical Load using Machine Learning Algorithm

Sanskar R., Roopam Kr. Raj, Kunal R. Singh, Piyush K., Soham V., and Amit Kr. Choudhary

***Abstract*— The aim of the proposed work is to identify & predict the future large-scale electrical energy consumption to reduce energy costs with minimum load on the grid. Machine learning (ML) methods recently contributed very well in the advancement of the prediction models used for energy consumption. Such models highly improve the accuracy, robustness, precision and the generalization ability of the conventional time series forecasting tools. This work reviews the state of the art of ML models used in the general application of energy consumption. This proposed work makes a conclusion on the trend and the effectiveness of the ML models. As the result, this work reports a satisfactory rise in the accuracy and an ever-increasing performance of the prediction technologies using the novel hybrid and ensemble prediction models.**

***Index Terms*— Artificial Intelligence, Decision Trees, Load Forecasting, Linear Regression, Machine Learning, Random Forest, Support Vector Machine.**

I. INTRODUCTION

THERE are different types of load forecasting namely Short-term forecasting (STLF), Mid-term forecasting (MTLF) and Long-term forecasting (LTLF). Generally, load forecasting can be influenced by many factors such as time, weather and the type of customer especially for STLF problems. However, MTLF and LTLF forecasting are functions of historical data for power consumption plus weather, number of customers, number of appliances and demographic data. STLF mostly have been studied in the literature for a time interval ranging from an hour to couple of weeks. STLF is used in different applications such as real time control, energy transfer scheduling, economy dispatch and demand response MTLF can be used for the range of interval from a month to five years in order to plan for near future power plants and show the dynamics of the power system within the time interval. LTLF has been used for an average interval of 5 to 20 years. Usually, LTLF is used to plan the generation power plants by the size and the type in order to satisfies the future requirement and cost efficient. In the series of various load forecasting artificial intelligence has gathered an eminent designation which is upgraded in this proposed work.

Machine Learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. ML algorithms use historical data as input to predict new output values. Its goal is to make machines like computers think and understand as human thinks by mimicking the grid of the human brain connection. AI has been investigated in many industries for automation processes such as automated labor, picture and audio detection, decision maker in critical fields and scientific research assistants. There is numerous literature available for artificial neural network being used for load forecasting [1-6]. Recently, with the rise of ML algorithms load forecasting is performed through it [7]. ML is the essential algorithm of AI which extract patterns from raw data to make subjective decisions. One of the simplest ML algorithms is logistic regression that can recommend the use of caesarean delivery decision for patients. In addition to simple ML algorithms, nave Bayes is a tool for splitting spam emails from legitimate email. ML algorithms are subdivided into supervised learning and unsupervised learning. A supervised learning algorithm is applied to a dataset that has features and each of those features associated with a label [8, 9]. However, an unsupervised learning algorithm is applied to a dataset which has many features in order to learn useful properties from the structure of the dataset. The dataset is always split into training and testing sub-datasets in the learning algorithms. The training dataset helps the model to learn from the raw dataset and the test dataset helps to validate the output of the model. A limitation of ML algorithms is the insufficient learning models for high dimensional datasets. Essentially load forecasting is one of the applications that benefited from ML algorithms in many papers in the literature. This paper shows an overview of the ML algorithms and methods that are used in grid load forecasting.

Further, paper is divided into four sections. Section I, being an introduction is followed by section II that offers an insight to the various ML methods. Review and analysis of the load forecasting through methodology using various ML algorithms is done in section III. Last section IV, presents a satisfactory and acceptable result among the various ML algorithms.

II. MACHINE LEARNING METHODS

Various ML methods include linear regression (*LR*), Support Vector Machine (*SVM*), Decision Trees (*DT*), Random Forest (*RF*) and XG_Boost. All these methods are brief here for quick review.

A. Linear Regression (LR)

The regression analysis is graphing a line on a set of data

The authors are associated with the Department of Electrical Engineering at BIT Sindri, Dhanbad (828123) Jharkhand, India. (*Corresponding author email: amit.ee@bitsindri.ac.in*)

points that most closely fits the overall shape of the data. Regression shows the changes in a dependent variable on the y-axis to the changes in the explanatory variable on the x-axis. Advantages of LR include (i) classification and regression capabilities for predicting the continuous variable, (ii) removes the one missing data point that could optimize the regression and (iii) it's not always computationally expensive than the decision tree or the clustering algorithm. Figure 1 represents the linear regression.

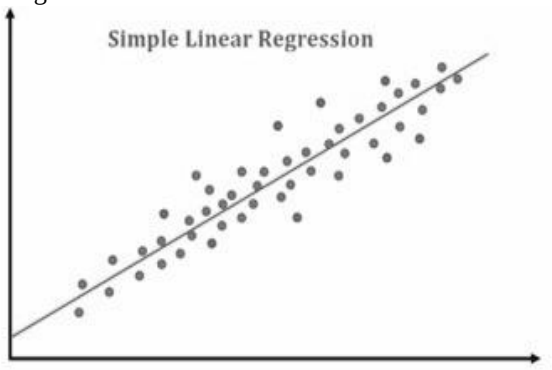

Fig. 1. Linear regression algorithm

B. Support Vector Machine (SVM)

It is a supervised machine learning algorithm that can be used for both classification or regression challenges. Its pictorial depiction is in figure 2. It is mostly used in classification problems. In the SVM algorithm, plotting of each data item as a point in *n*-dimensional space (where *n* is the number of features accessible) with the value of each feature being the value of a particular coordinate. Then, classification is performed by finding the hyper-plane that differentiates the two classes very well.

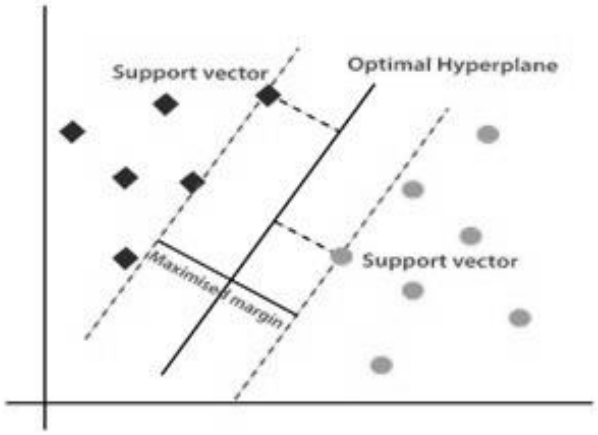

Fig. 2. Support vector machine algorithm

C. Decision Trees (DT)

These are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation. As shown in figure 3, decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules and the fitter the model.

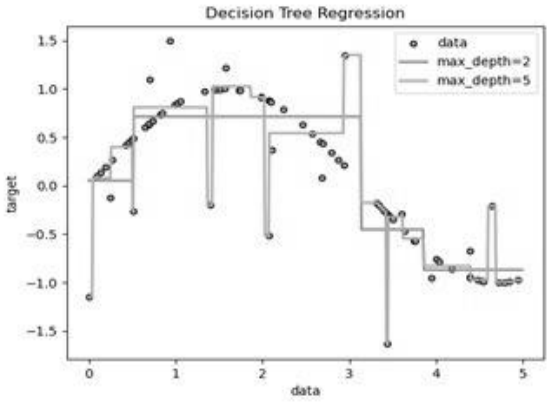

Fig 3. The architectural unit of decision tree

D. Random Forest (RF)

Random forest, just as the name suggests, consists of a large number of individual decision trees that operate as an ensemble that combines multiple models. Thus, a collection of models is used to make predictions rather than an individual model. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes the model's prediction as depicted in figure 4. The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds.

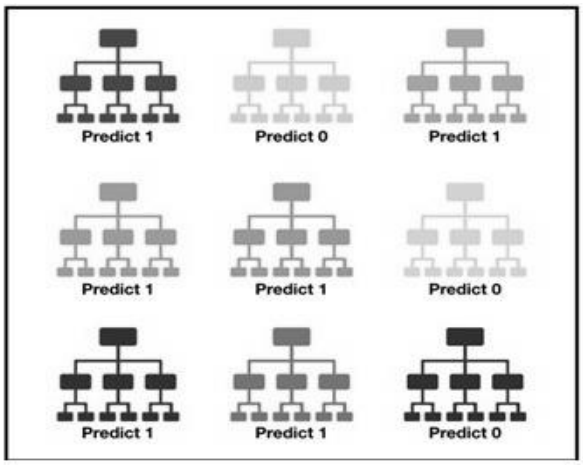

Fig 4. Visualization of a Random Forest Model

One of the most important features of the RF algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems.

E. Extreme Gradient Boosting (XG_Boost)

It is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning library. It provides parallel tree boosting and is the leading machine learning library for regression, classification, and ranking problems. It's vital to an understanding of XG_Boost to first grasp the machine learning concepts and algorithms that XG_Boost builds upon: supervised machine learning, decision trees, ensemble learning, and gradient boosting. Supervised machine learning uses

algorithms to train a model to find patterns in a dataset with labels and features and then uses the trained model to predict the labels on a new dataset's features.

The benefits and attributes of XG_Boost is extensive, and includes (i) large and growing list of data scientists globally that are actively contributing to XG_Boost open-source development, (ii) usage on a wide range of applications, including solving problems in regression, classification, ranking, and user-defined prediction challenges, (iii) library that's highly portable and currently runs on OS X, Windows, and Linux platforms and many others.

III. Methodology And Review of Load Forecasting

The goal of the proposed work is to retrieve a robust, reliable, efficient machine learning model to forecast load and optimize the traditional power system to lower the cost of production and maintenance. The use of the machine learning is gaining the advancement of the dramatically increase of the usage of technology and communication by leveraging the power of data.

Machine learning relies on three major roles:

- The previous consumption.
- Selection of appropriate model
- Proper hyperparameter tuning of the model.

Specifically, load forecasting is one of the most important factors in planning and operating the power system. Electrical Load forecasting can be influenced by many factors that can change the patterns of load consumptions. The flow chart of the load forecasting through ML is shown in figure 5.

Stepwise methodology as mentioned in the flowchart in figure 5 is elaborated below.

A. Gathering Of the Data

The data set is collected from various sources such as a file, database and many other such sources and repositories. Figure 6 showcase the data used.

B. Data Analysis and Processing

Data pre-processing is most important step that helps in building machine learning models more accurately.

Various statistical techniques were used to address data consistency problem.

1) Splitting dataset into training dataset and test dataset:

With the help of scikit-learn library for machine learning, the available dataset was split into training dataset (90%) and testing dataset (10%) as shown in figure 7.

2) Training Dataset

The training dataset depicted in figure 8 is used for training the machine learning model.

3) Testing Dataset

The testing dataset shown in figure 9 is used for testing the performance of the machine learning model on some predefined scikit-learn library performance metrics.

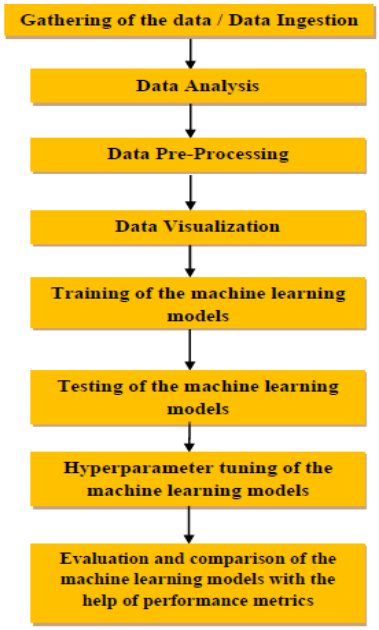

Fig. 5. Flowchart of load forecasting methodology

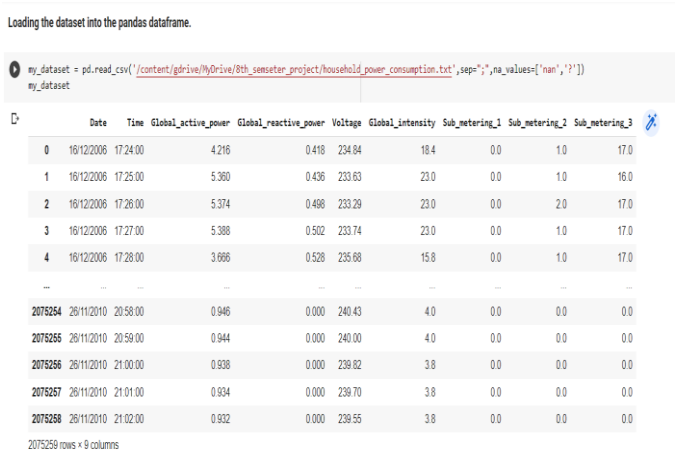

Fig. 6: Desired dataset

Fig 7. Splitting of data into training and testing datasets

training_dataset

	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3	active_energy_per_minute_in_wh
1636907	1.944	0.210	238.85	8.2	0.0	0.0	21.0	11.400000
1594571	0.352	0.098	244.90	1.6	0.0	0.0	0.0	5.866667
1344048	0.300	0.224	239.76	1.4	0.0	0.0	0.0	5.000000
617464	1.278	0.000	239.72	5.2	0.0	0.0	18.0	3.300000
454944	0.302	0.108	240.76	1.2	0.0	0.0	0.0	5.033333
...	...	...	...	...	...	...	...	...
130365	0.370	0.174	238.89	1.6	0.0	1.0	0.0	5.166667
616591	1.800	0.140	239.09	7.6	0.0	1.0	0.0	29.000000
1055194	1.302	0.000	242.54	5.2	0.0	0.0	18.0	3.700000
1139006	1.598	0.208	245.18	6.6	0.0	1.0	19.0	6.533333
29828	1.280	0.000	237.94	5.4	0.0	0.0	0.0	21.333333

1867733 rows × 8 columns

Fig 8. Training Dataset

testing_dataset

	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3	active_energy_per_minute_in_wh
1132415	0.226000	0.000000	248.390000	1.000000	0.000000	0.00000	0.000000	3.766667
1502573	0.448000	0.070000	243.400000	1.800000	0.000000	0.00000	1.000000	6.466667
192435	1.091615	0.123714	240.839658	4.627759	1.121923	1.29652	6.458447	9.314693
279938	0.314000	0.226000	240.120000	1.600000	0.000000	1.00000	0.000000	4.233333
261803	1.674000	0.336000	238.740000	7.200000	0.000000	1.00000	17.000000	9.900000
...	...	...	...	...	...	...	...	...
51733	1.500000	0.148000	240.270000	7.600000	14.000000	0.00000	0.000000	11.000000
1857001	1.288000	0.090000	238.890000	5.400000	0.000000	2.00000	19.000000	0.466667
395704	0.272000	0.182000	236.810000	1.400000	0.000000	2.00000	0.000000	2.533333
783486	0.796000	0.220000	241.330000	3.600000	0.000000	0.00000	0.000000	13.266667
1384970	0.316000	0.238000	241.210000	1.600000	0.000000	0.00000	1.000000	4.266667

207526 rows × 8 columns

Fig 9. Testing Dataset

C. Data Visualization

The process of finding and correlations in data by representing it pictorially is called data visualization. To perform data visualization, various python data visualization modules were used. Python modules used for data visualization are Matplotlib, Seaborn and Plotly. Data Visualization done on the data using various graphs as shown in figure 10.

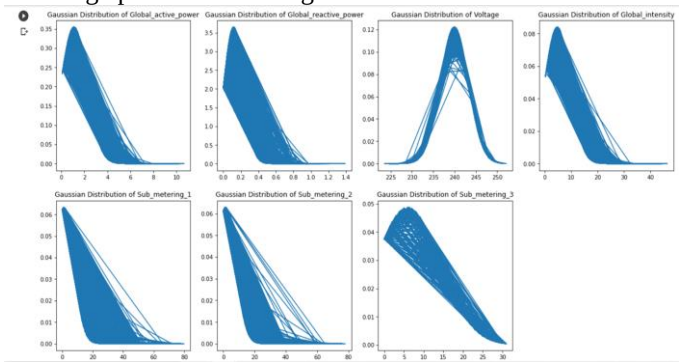

Fig 10. Data Visualization

D. Training Of the Machine Learning Models

The below figures represent the training of the dataset visualization in the earlier section through various ML methods. Figures 11 to 16 represent the training through Linear Regression, Support Vector Machine (SVM), Decision Tree, Random Forest and XG_Boost.

Among all the models trained, XG_Boost model pose the highest r2_score and is best suited for performing electrical load forecasting as tabulated in table-I.

Linear Regression Model

```
#Fitting the model on training set
from sklearn.linear_model import LinearRegression

reg = LinearRegression()
model1 = reg.fit(X_train,Y_train)

print(f'Regression score : {model1.score(X_train,Y_train)}')
print(f'Regression coefficient : {model1.coef_}')
```

```
Regression score : 1.0
Regression coefficient : [[ 1.60300000e+02  0.00000000e+00 -1.77635684e-15  1.06581410e-14
  -8.00000000e+01 -7.70000000e+01 -3.10000000e+01]]
```

```
target1 = Y_test.to_numpy().flatten()
target1
```

```
array([ 2.86666667,  2.76666667,  3.53333333, ...,  5.76666667,
        1.36666667, 13.23333333])
```

```
predicted1 = model1.predict(X_test).flatten()
predicted1
```

```
array([ 3.30811644,  6.22101252,  3.91818698, ...,  6.70050272,
        1.3836455 , 18.45215223])
```

```
predicted1 - target1
```

```
array([0.44144977, 3.45434585, 0.38485364, ..., 0.93383605, 0.01697884,
       5.2188189 ])
```

```
from sklearn.metrics import mean_squared_error,r2_score


mse1 = mean_squared_error(target1,predicted1)
r2_scr1 = r2_score(target1,predicted1)

print(f"value of mse : {mse1}")
print(f"value of r2_score : {r2_scr1}")
```

```
value of mse : 15.142193612381762
value of r2_score : 0.8156753007770728
```

Fig 11. Linear Regression Model Code.

Support Vector Machine Model

```
from sklearn import svm

svr1 = svm.SVR(kernel='linear')

svr1.fit(X_train,Y_train.to_numpy().flatten())

predicted2 = svr1.predict(X_test)
target2 = Y_test.to_numpy().flatten()


mse2 = mean_squared_error(target2,predicted2)
r2_scr2 = r2_score(target2,predicted2)

print(f"value of mse : {mse2}")
print(f"value of r2_score : {r2_scr2}")

filename = f'svm_svr1_linear_model_{r2_scr2}_{mse2}.sav'
pickle.dump(svr1, open(filename, 'wb'))
```

```
value of mse : 15.360562178574126
value of r2_score : 0.8130171178668877
```

Fig 12. Support Vector Machine Linear Kernel Model Code

```
from sklearn import svm

svr2 = svm.SVR(kernel='poly')

svr2.fit(X_train,Y_train.to_numpy().flatten())

predicted3 = svr2.predict(X_test)
target3 = Y_test.to_numpy().flatten()


mse3 = mean_squared_error(target3,predicted3)
r2_scr3 = r2_score(target3,predicted3)

print(f"value of mse : {mse3}")
print(f"value of r2_score : {r2_scr3}")

filename = f'svm_svr2_poly_model_{r2_scr3}_{mse3}.sav'
pickle.dump(svr2, open(filename, 'wb'))
```

```
value of mse : 32.22943014152453
value of r2_score : 0.6076737513047541
```

Fig 13. Support Vector Machine Poly kernel Model Code

Decision Tree Regression Model

```
from sklearn.tree import DecisionTreeRegressor


regressor1 = DecisionTreeRegressor(random_state = 0)
regressor1.fit(X_train,Y_train.to_numpy().flatten())


predicted4 = regressor1.predict(X_test)
target4 = Y_test.to_numpy().flatten()


mse4 = mean_squared_error(target4,predicted4)
r2_scr4 = r2_score(target4,predicted4)

print(f"value of mse : {mse4}")
print(f"value of r2_score : {r2_scr4}")

filename = f'decision_tree_model_{r2_scr4}_{mse4}.sav'
pickle.dump(regressor1, open(filename, 'wb'))
```

```
value of mse : 16.743579167842157
value of r2_score : 0.7961817638162978
```

Fig 14. Decision Tree Model Code

Random Forest Model

```
from sklearn.ensemble import RandomForestRegressor

regressor2 = RandomForestRegressor(n_estimators = 100, random_state = 0)
regressor2.fit(X_train,Y_train.to_numpy().flatten())

predicted5 = regressor2.predict(X_test)
target5 = Y_test.to_numpy().flatten()


mse5 = mean_squared_error(target5,predicted5)
r2_scr5 = r2_score(target5,predicted5)

print(f"value of mse : {mse5}")
print(f"value of r2_score : {r2_scr5}")

filename = f'random_forest_model_{r2_scr5}_{mse5}.sav'
pickle.dump(regressor2, open(filename, 'wb'))
```

```
value of mse : 15.662564533880547
value of r2_score : 0.8093408676001526
```

Fig 15. Random Forest Model Code

Xgboost Regression Model

```
import xgboost as xg
from xgboost import XGBRegressor


regressor3 = XGBRegressor()
regressor3.fit(X_train,Y_train)

predicted6 = regressor3.predict(X_test)
target6 = Y_test.to_numpy().flatten()


mse6 = mean_squared_error(target6,predicted6)
r2_scr6 = r2_score(target6,predicted6)

print(f"value of mse : {mse6}")
print(f"value of r2_score : {r2_scr6}")

filename = f'xgboost_model_{r2_scr6}_{mse6}.sav'
pickle.dump(regressor3, open(filename, 'wb'))
```

```
[06:34:16] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
value of mse : 12.686506695482702
value of r2_score : 0.8455681791756802
```

Fig 16. XG_Boost Model Code.

The figure 17 and figure 18 obtained by the training in the present section depict the acceptable result from the XG_Boost model for mean squared error and r2_score of all the models.

TABLE I
COMPARISON OF R2_SCORES AND MEAN-SQUARED-ERRORS OF ALL TRAINED MODELS

Model	r2_score	mean-squared_error
LR	0.8156	15.1421
SVM with linear kernel	0.8130	15.3605
SVM with poly kernel	0.6076	32.2294
DT	0.7961	16.7435
RF	0.8093	15.6625
XG_boost	0.8455	12.6865

The proposed discussion offers that the ML technology can be used to forecast future electrical load of household. The justification for the acceptance of the proposed work is done by the use of electrical household consumption dataset. Based on the available data, it is observed that the available data has 1.25% missing values. To address the data inconsistency problem various statistical techniques are used. Different graphs and plots are plotted with the help of the available training data. After plotting it is observed that data has lot of outliers which makes data inconsistent, so proper outliers removal method need to be used in order to address the outlier problem. After the pre-processing of the available dataset, the different machine learning models are trained. The models which were trained are linear regression model, support vector machine model with linear kernel, support vector machine model with poly kernel, decision tree model, random forest model, XG_Boost model.

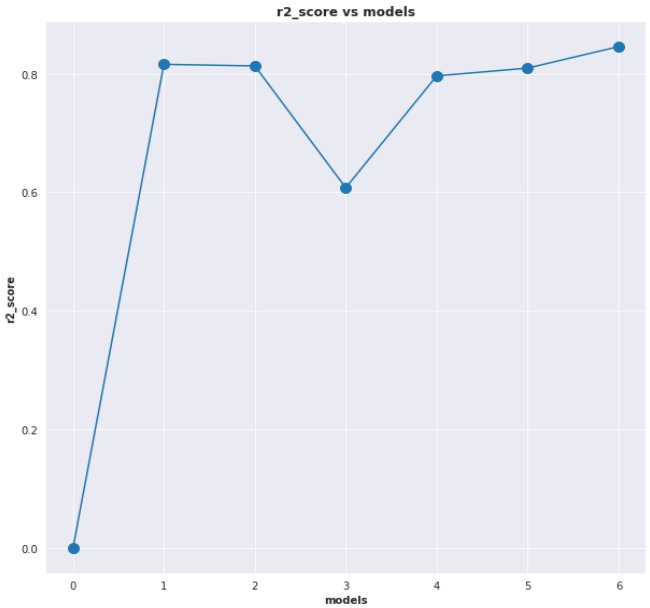

Fig 17. Plot of r2_score of models.

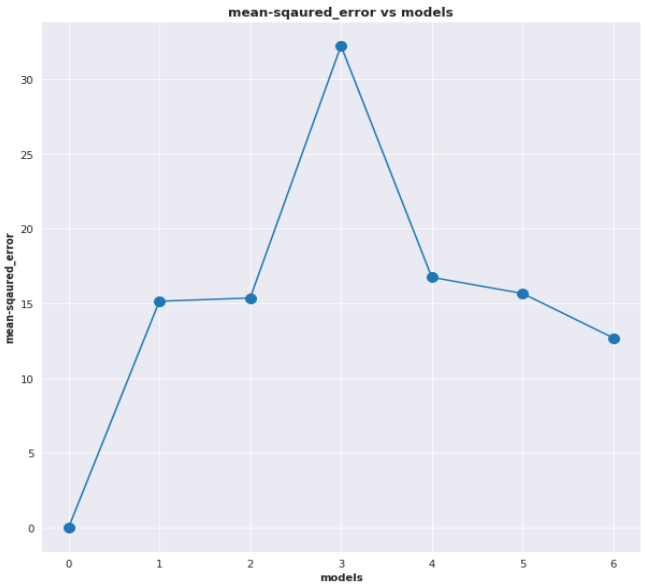

Fig 18. Plot of mse of models.

IV. Conclusion

It is observed that the highest r2_score is obtained for XG_Boost model among all the trained models. The obtained score is satisfactorily acceptable for the final inference purpose. This proposed work may be helpful for future household electrical load forecasting using machine learning algorithms precisely. As a future scope of this work, the performance of the trained models can be improved by performing hyperparameter tuning.

References

[1] M. Hayati and Y. Shirvany, "Artificial neural network approach for short term load forecasting for illam region," *World Academy of Science, Engineering and Technology*, vol. 28, pp. 280–284, 2007.
[2] N. Kandil, R. Wamkeue, M. Saad, and S. Georges, "An efficient approach for short term load forecasting using artificial neural networks," *International Journal of Electrical Power & Energy Systems*, vol. 28, no. 8, pp. 525–530, 2006.
[3] C. Park, M. El-Sharkawi, R. Marks, L. Atlas, and M. Damborg, "Electric load forecasting using an artificial neural network*," IEEE transactions on Power Systems*, vol. 6, no. 2, pp. 442–449, 1991.
[4] Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks:: The state of the art," *International journal of forecasting*, vol. 14, no. 1, pp. 35–62, 1998.
[5] S. Hippert, C. E. Pedreira, and R. C. Souza, "Neural networks for short-term load forecasting: A review and evaluation," *IEEE Transactions on power systems*, vol. 16, no. 1, pp. 44–55, 2001.
[6] Ahmad A, et al. A review on applications of ANN and SVM for building electrical energy consumption forecasting. *Renew Sustain Energy Rev* 2014;33:102–9.
[7] Yildiz B, Bilbao JI, Sproul AB. A review and analysis of regression and machine learning models on commercial building electricity load forecasting. *Renew Sustain Energy Rev* 2017;73:1104–22.
[8] Machine Learning Course, Andrew Ng, Deeplearning.ai, Stand-ford University.
[9] Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.

Sanskar Ram was born in Dhanbad in the Jharkhand, on October 12, 1998. He graduated from the BIT , Sindri, and studied at the Jharkhand University of Technology, Ranchi with a specialization in electrical engineering. He has done internship at IQVIA as Software Developer Intern in the big data engineering team. His special fields of interest included BigData Analytics, Machine Learning and Deep Learning.

Roopam Kr. Raj born in Bihar, India on April 30, 1999. Currently pursuing B.Tech in Electrical Engineering from BIT SINDRI, Jharkhand University of Technology, Ranchi.

Piyush Kumar Singh born in Bihar,India on August 25 ,1999. Currently pursuing B.tech in Electrical Engineering from BIT SINDRI, Jharkhand University of Technology.

Kunal Ranjan Singh born in Bihar (Now Jharkhand), India on December 01,1999. Currently pursuing B Tech in Electrical Engineering from BIT SINDRI, Jharkhand University of Technology.

Soham Vivek born in Jharkhand, India on March 13, 2000. Currently pursuing B.Tech in Electrical Engineering from BIT SINDRI, Jharkhand University of Technology.