FORECASTING OF FUTURE HOUSEHOLD ELECTRICAL LOAD USING MACHINE LEARNING

A

Project Report

Submitted to

JHARKHAND UNIVERSITY OF TECHNOLOGY, RANCHI

In Partial Fulfillment of the Requirement for the Award of Degree of

BACHELOR OF TECHNOLOGY
IN
ELECTRICAL ENGINEERING

Submitted by

Kunal Ranjan Singh	**18030450044**
Piyush Kumar	**18030450058**
Roopam Kr. Raj	**18030450072**
Sanskar Ram	**18030450081**
Soham Vivek	**18030455015**

Under the guidance of

Dr. Amit Kumar Choudhary
Assistant Professor

DEPARTMENT OF ELECTRICAL ENGINEERING
BIT SINDRI, DHANBAD-828123 (JHARKHAND)

June, 2022

BIT SINDRI, DHANBAD-828123

Department of Higher & Technical Education, Government of Jharkhand

CANDIDATE'S DECLARATION

We hereby certify that the project entitled "**FORECASTING OF FUTURE HOUSEHOLD ELECTRICAL LOAD USING MACHINE LEARNING"** which is being submitted in the partial fulfillment of the requirement for the award of **Bachelor of Technology in the Department of Electrical Engineering**, is a record of our own work carried out under the supervision and guidance of Dr. Amit Kumar Choudhary, Assistant Professor, Department of Electrical Engineering, BIT Sindri.

All information in this document has been obtained and presented as per academic rules and ethical conduct.

To the best of our knowledge, the contents presented in this project report have not been submitted elsewhere for the award of any other degree/diploma.

Date:

Place:

Signature:

Kunal Ranjan Singh (**18030450044**)

Signature:

Piyush Kumar (**18030450058)**

Signature:

Roopam Kr. Raj (**18030450072)**

Signature:

Sanskar Ram (**18030450081)**

Signature:

Soham Vivek (**18030455015)**

BIT SINDRI, DHANBAD-828123

Department of Higher & Technical Education, Government of Jharkhand

CERTIFICATE OF DECLARATION FROM THE SUPERVISOR

This is to certify that the work presented in the project report entitled "**FORECASTING OF FUTURE HOUSEHOLD ELECTRICAL LOAD USING MACHINE LEARNING**" in partial fulfillment of the requirement for the award of Degree of **Bachelor of Technology** in the Department of Electrical Engineering at BIT Sindri is an authentic work carried out under my supervision and guidance.

To the best of my knowledge, the content of this report does not form a basis for the award of any previous Degree to anyone else.

Date:	Signature of supervisor
Place:	Name: Dr. Amit Kumar Choudhary
	Designation: Assistant Professor
	Department Name: Electrical Engineering

BIT SINDRI, DHANBAD-828123

Department of Higher & Technical Education, Government of Jharkhand

CERTIFICATE OF APPROVAL FROM THE EXAMINERS

The foregoing project report entitled “**FORECASTING OF FUTURE HOUSEHOLD ELECTRICAL LOAD USING MACHINE LEARNING**”, is hereby approved as a creditable study of the research topic and has been presented satisfactorily to warrant its acceptance as a prerequisite to the degree for **Bachelor of Technology** in the Department of Electrical Engineering at BIT Sindri.

It is understood that by this approval, the undersigned do not necessarily endorse any conclusion drawn or opinion expressed therein, but approve the report for the purpose for which it is submitted.

Signature of Internal Examiner	Signature of External Examiner
Name:	Name :
Designation:	Designation:
Name of Department:	Name of Department:
Name of the Institute:	Name of the Institute:

Signature of HOD

Name: Dr. Md Abul Kalam

Head, Department of Electrical Engineering, BIT Sindri

ACKNOWLEDGMENT

We would like to articulate our deep gratitude to our project guide Dr. Amit Kumar Choudhary, Assistant Professor, Department of Electrical Engineering who have always been source of motivation and firm support for carrying out the project. We also express our gratitude to him for his invaluable suggestion and constant encouragement all through the project work.

We would also like to convey our sincerest gratitude and indebtedness to the Head of the Department and all other faculty members and staff of Department of Electrical Engineering, BIT Sindri, who bestowed their great effort and guidance at appropriate times without it, would have been very difficult on our work.

An assemblage of this nature could have been attempted with our reference to and inspiration from the works of others whose details are mentioned in references section. We acknowledge our indebtedness to all of them. Further, we would like to express our feeling towards our parents and god who directly or indirectly encouraged and motivated us during this dissertate.

Date:
Place:

Signature:
Kunal Ranjan Singh (**18030450044**)

Signature:
Piyush Kumar (**18030450058)**

Signature:
Roopam Kr. Raj (**18030450072)**

Signature:
Sanskar Ram (**18030450081)**

Signature:
Soham Vivek (**18030455015)**

INDEX

TITLE	**PAGE NO.**
Declaration	1
Certificate of Declaration from the Supervisor	2
Certificate of Approval from the Examiners	3
Acknowledgement	4
Index	5
Abstract	7
List of Figures	8
List of Tables	8
1. INTRODUCTION	**9**
1.1 Overview of Machine Learning	9
2. OBJECTIVE	**10**
2.1 Objective of the work	10
3. METHODOLOGY OF THE WORK	**11**
3.1 Flowchart of the work	11
3.2 Gathering of data	11
3.3 Data analysis and Pre-processing	13
3.3.1 Data Cleaning	14
3.3.2 Data Integration	14
3.3.3 Data Reduction	15
3.3.4 Data Transformation	15
3.4 Data Visualization	15
3.4.1 Line Charts	16
3.4.2 Bar Graphs	17
3.4.3 Histograms	17
3.4.4 Scatter Plots	18
3.4.5 Violin Plots	18
3.4.6 Box Plots	19
3.5 Training, Testing and Evaluating Machine Learning Models	19
3.5.1 Splitting the dataset	19

3.5.2 Model training and testing 21

3.5.3 Hyper parameter tuning 24

3.5.4 Model Evaluation 25

4. RESULTS AND DISCUSSION 27

5. CONCLUSIONS AND FUTURE SCOPES 29

5.1 Conclusions 29

5.2 Future Scopes 29

REFRENCES 30

LIST OF PUBLICATION 31

ABSTRACT

The objective of the project is to identify & predict future large-scale electrical energy consumption, a more flexible power consumption so as to reduce energy costs and the impact on the environment as well as utilize periods with large environmental power generation and minimum load on the grid.

Machine learning (ML) methods has recently contributed very well in the advancement of the prediction models used for energy consumption. Such models highly improve the accuracy, robustness, and precision and the generalization ability of the conventional time series forecasting tools. This project reviews the state of the art of machine learning models used in the general application of energy consumption. Through a novel search and taxonomy, the most relevant literature in the field is classified according to the ML modelling technique, energy type, perdition type, and the application area. A comprehensive review of the literature identifies the major ML methods, their application and a discussion on the evaluation of their effectiveness in energy consumption prediction. This proposed work further makes a conclusion on the trend and the effectiveness of the ML models. As the result, this work reports an outstanding rise in the accuracy and an ever-increasing performance of the prediction technologies using the novel hybrid and ensemble prediction models.

LIST OF FIGURES

Fig No.	Title	Page No.
3.1	Flowchart of the work	11
3.2	Code Snippet showing the location of dataset in the drive.	12
3.3	Code Snippet for loading of the dataset in the dataframe.	12
3.4	Data Pre-processing of raw data.	13
3.5	Line Chart	16
3.6	Bar Graph	17
3.7	Histogram	17
3.8	Scatter Plot	18
3.9	Violin Plot	18
3.10	Box Plot	19
3.11	Distribution of dataset into different segment	19
3.12	Code snippet for splitting of the dataset	20
3.13	Training dataset after splitting of the dataset	20
3.14	Testing dataset after splitting of the dataset	21
3.15	Code Snippet for training & testing of the linear regression model	22
3.16	Code Snippet for training & testing of the svm regression (linear kernel) model	22
3.17	Code Snippet for training & testing of the svm regression (poly kernel) model	22
3.18	Code Snippet for training & testing of the decision tree regression model	23
3.19	Code Snippet for training & testing of the random forest regression model	23
3.20	Code Snippet for training & testing of the xgboost regression model	24
3.21	Hyperparameter tuning	24
3.22	Hyperparameter tuning technique	24
3.23	Models and their respective mse and r2_score values	25
3.24	Plot of r2_score of models	26
3.25	Plot of mse of models	26
4.1	Plot of r2_score of models	28
4.2	Plot of mse of models	28

LIST OF TABLES

Table No.	Title	Page No.
4.1	Comparison Between Mean Squared Error & R2_Score	27

CHAPTER 1
INTRODUCTION

1.1 Overview of machine learning

Machine learning (ML) methods has recently contributed very well in the advancement of the prediction models used for energy consumption. Such models highly improve the accuracy, robustness, and precision and the generalization ability of the conventional time series forecasting tools [1].

Types of Machine Learning Algorithms:

- *Supervised Machine Learning*: In supervised learning, both the input and desired output are provided and the machine must learn how to map the former to the latter. To accomplish this, the machine is trained on a statistically representative set of example inputs and corresponding outputs. Supervised Learning deals with two main tasks Regression and Classification.
- *Unsupervised Machine Learning*: In unsupervised learning, the machine is not provided labelled examples or previous patterns on which to base the analysis of the data inputs. The machine must uncover patterns and draw inferences by itself, without having the correct answers. It will classify or cluster data by discovering the similarity of features on its own. Unsupervised Learning deals with clustering and associative rule mining problems.
- *Reinforcement Learning*: Reinforcement learning continuously improves its model based on feedback from experiences. It learns through trial and error – from the consequences of its action and by new choices. As an action is taken, the success of the outcome is graded and receives either a positive or negative score. Reinforcement Learning deals with exploitation or exploration, Markov's decision processes, Policy Learning, Deep Learning and value learning [2].

CHAPTER 2
OBJECTIVE

2.1 Objective of the Work

This project reviews the state of the art of machine learning models used in the general application of energy consumption. Through a novel search the most relevant literature in the field is classified according to the ML modelling technique, energy type, perdition type, and the application area.

A comprehensive review of the literature identifies the major ML methods, their application and a discussion on the evaluation of their effectiveness in energy consumption prediction. This literature reviews the training of the different machine learning models and their performance score on the standard comparison metrics.

This project further makes a conclusion on the trend and the effectiveness of the data analysis & pre-processing, data visualization, outlier removal and data splitting prior to start building ML models. The different state of the art machine learning will be trained with the help of the available dataset with hyper parameter tuning to improve their accuracy and appropriate model will be selected among all the trained models. As the result, this project reports an outstanding rise in the accuracy and an ever-increasing performance of the prediction technologies using the novel hybrid and ensemble prediction models [3] [4].

With the use of machine learning the below optimization are produced in the power system**:**

1. The decrease in the maintenance cost of the power system.
2. Sufficient supply of electrical energy to household as per the requirement
3. To predict upcoming future load.
4. Improved energy efficiency, reliability, and security
5. Reducing power generation costs [5] [6] [7].

CHAPTER 3

METHODOLOGY OF THE WORK

3.1 Flowchart of the work

The below flowchart that is represented by Fig. 3.1 shows the various steps involved in the project.

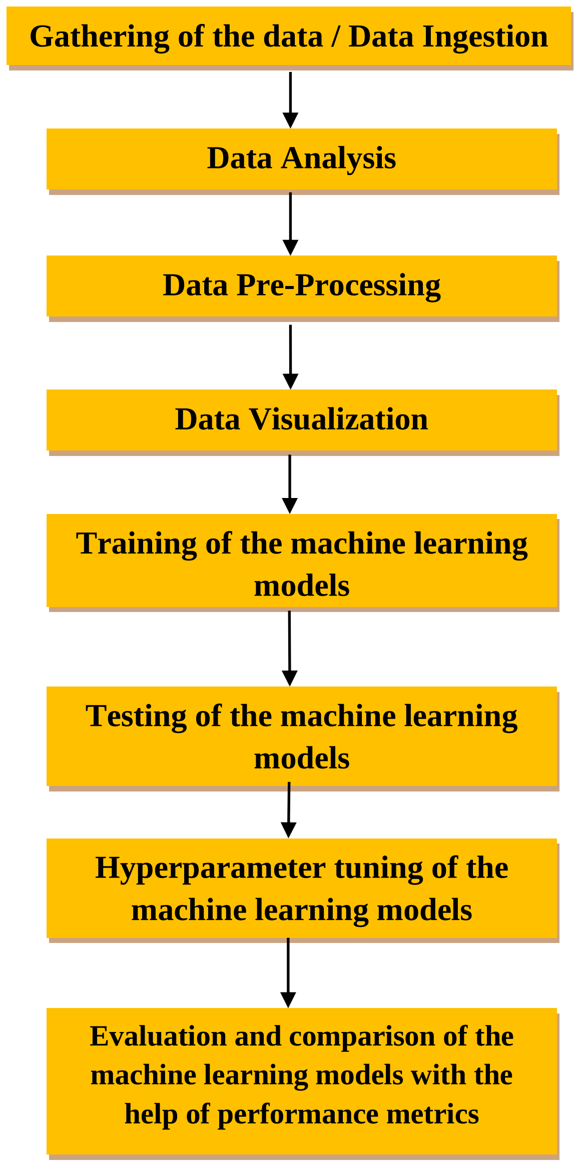
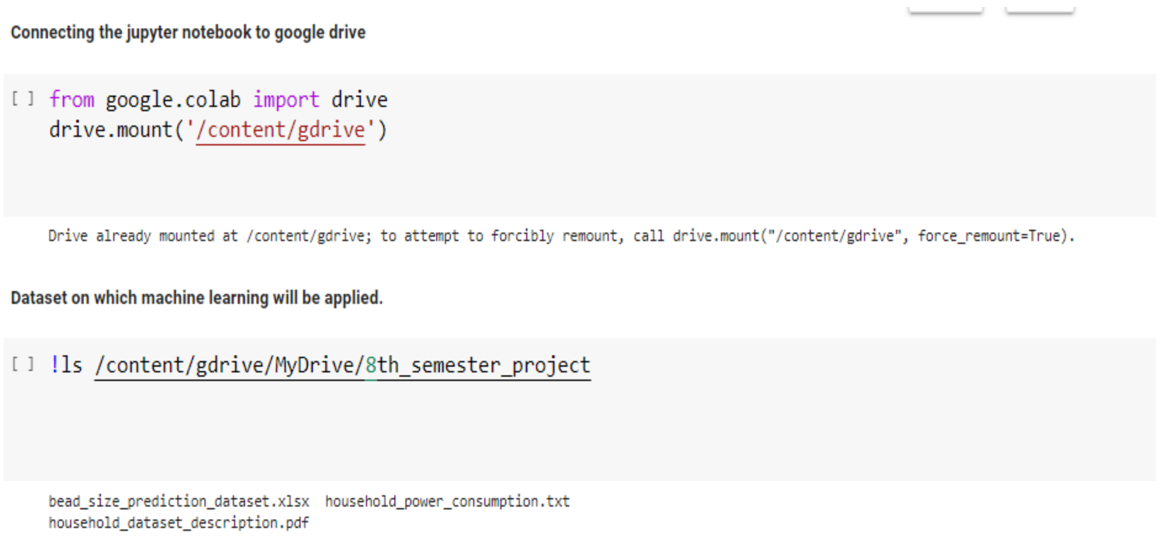

Gathering of the data / Data Ingestion
Data Analysis
Data Pre-Processing
Data Visualization
Training of the machine learning models
Testing of the machine learning models
Hyperparameter tuning of the machine learning models
Evaluation and comparison of the machine learning models with the help of performance metrics

Fig. 3.1 Flowchart of the work**.**

3.2 Gathering of the data

The process of gathering data depends on the type of project one desires to make, if anyone want to make an ML project that uses real-time data. The data set can be collected from various sources such as a file, database, sensor and many other such sources but the collected data cannot be used directly for performing the analysis process as there might be a lot of missing data, extremely large values, unorganized text data or noisy data. Data pre-processing is one of the most important steps in machine learning [1] [3] [4]. It is the most important step that helps in building machine learning models more accurately. In machine learning, there is an 80/20 rule.

In this project the data for training the machine learning model is collected from the UC Irvine Machine learning repository, which is an open-source machine learning dataset platform for performing machine learning. The dataset on which machine learning will be performed is stored in the hierarchical directory and it is further loaded in pandas data frame with the help of python library for performing various kind of exploratory data analysis and data visualization tasks [3]. Fig 3.2 and fig 3.3 depicts the location of dataset in drive and loading of dataset in data frame respectively



Connecting the jupyter notebook to google drive

```
[ ] from google.colab import drive
    drive.mount('/content/gdrive')
```

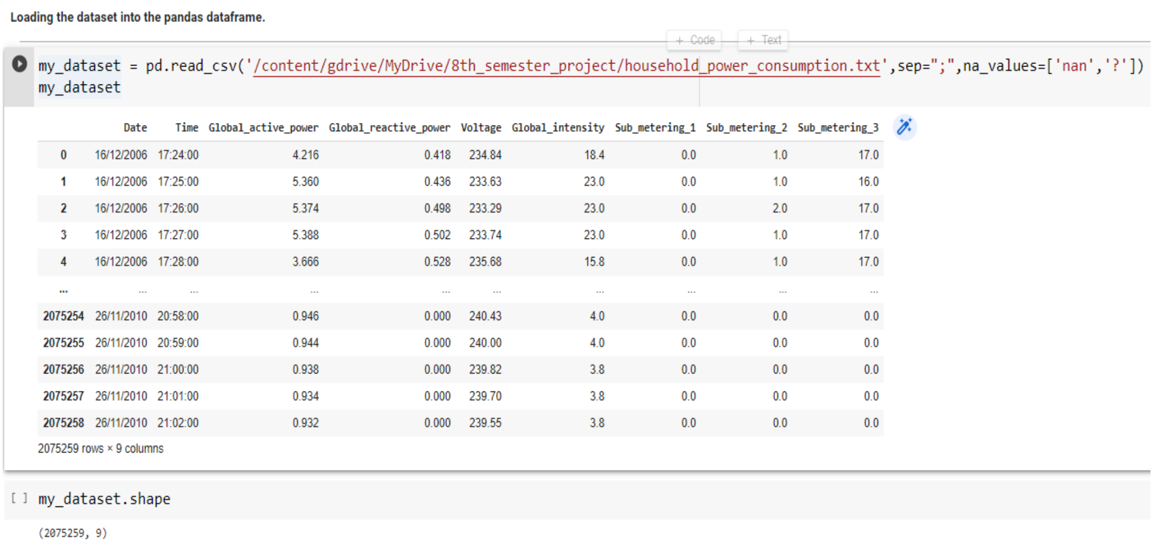
```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

Dataset on which machine learning will be applied.

```
[ ] !ls /content/gdrive/MyDrive/8th_semester_project
```

```
bead_size_prediction_dataset.xlsx   household_power_consumption.txt
household_dataset_description.pdf
```

Fig 3.2 Code Snippet showing the location of dataset in the drive.



Loading the dataset into the pandas dataframe.

```
my_dataset = pd.read_csv('/content/gdrive/MyDrive/8th_semester_project/household_power_consumption.txt',sep=";",na_values=['nan','?'])
my_dataset
```

```
               Date      Time  Global_active_power  Global_reactive_power  Voltage  Global_intensity  Sub_metering_1  Sub_metering_2  Sub_metering_3
0        16/12/2006  17:24:00                4.216                  0.418   234.84              18.4             0.0             1.0            17.0
1        16/12/2006  17:25:00                5.360                  0.436   233.63              23.0             0.0             1.0            16.0
2        16/12/2006  17:26:00                5.374                  0.498   233.29              23.0             0.0             2.0            17.0
3        16/12/2006  17:27:00                5.388                  0.502   233.74              23.0             0.0             1.0            17.0
4        16/12/2006  17:28:00                3.666                  0.528   235.68              15.8             0.0             1.0            17.0
...             ...       ...                  ...                    ...      ...               ...             ...             ...             ...
2075254  26/11/2010  20:58:00                0.946                  0.000   240.43               4.0             0.0             0.0             0.0
2075255  26/11/2010  20:59:00                0.944                  0.000   240.00               4.0             0.0             0.0             0.0
2075256  26/11/2010  21:00:00                0.938                  0.000   239.82               3.8             0.0             0.0             0.0
2075257  26/11/2010  21:01:00                0.934                  0.000   239.70               3.8             0.0             0.0             0.0
2075258  26/11/2010  21:02:00                0.932                  0.000   239.55               3.8             0.0             0.0             0.0
2075259 rows × 9 columns
```

```
[ ] my_dataset.shape
```

```
(2075259, 9)
```

Fig 3.3 Code Snippet for loading of the dataset in the data frame.

3.3 Data Analysis & Pre-processing

Data pre-processing is the process of transforming raw data into an understandable format. It is also an important step in data mining while working with raw data. The quality of the data should be checked before applying machine learning or data mining algorithms [1].

Importance of Data pre-processing

Pre-processing of data is mainly to check the data quality and consistency. The quality can be checked by the following:

1. **Accuracy**: To check whether the data entered is correct or not.
2. **Completeness**: To check whether the data is available or not recorded.
3. **Consistency:** To check whether the same data is kept in all the places that do or do not match.
4. **Timeliness**: The data should be updated correctly.
5. **Believability**: The data should be trustable.
6. **Interpretability**: The understandability of the data.

Major Tasks in Data Pre-processing is shown in fig. 3.4.

1. Data cleaning
2. Data integration
3. Data reduction
4. Data transformation

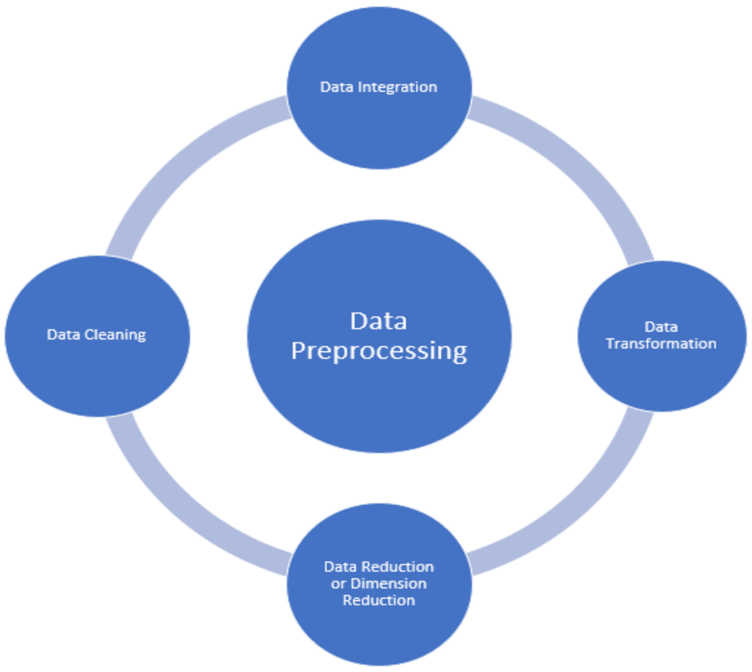


Fig 3.4 Data Pre-processing of raw data.

3.3.1 Data Cleaning

It is the process to remove incorrect data, incomplete data and inaccurate data from the datasets, and it also replaces the missing values. There are some techniques in data cleaning [3] [4] [5] [6].

Handling missing values:

1. Standard values like “*Not Available” or “NA*” can be used to replace the missing values.
2. Missing values can also be filled manually but it is not recommended when that dataset is big.
3. The attribute’s mean value can be used to replace the missing value when the data is normally distributed wherein in the case of non-normal distribution median value of the attribute can be used.
4. While using regression or decision tree algorithms the missing value can be replaced by the most probable value.

Noisy:

Noisy generally means random error or containing unnecessary data points. *Here are some of the methods to handle noisy data.*

1. Regression: This is used to smooth the data and will help to handle data when unnecessary data is present. For the analysis, purpose regression helps to decide the variable which is suitable for our analysis.
2. Clustering: This is used for finding the outliers and also in grouping the data. Clustering is generally used in *unsupervised learning* [1].

3.3.2 Data Integration

The process of combining multiple sources into a single dataset. The Data integration process is one of the main components in data management. There are some problems to be considered during data integration [1] [4]:

- *Schema integration*: Integrates metadata (a set of data that describes other data) from different sources.
- *Entity identification problem*: Identifying entities from multiple databases. For example, the system or the use should know student _id of one database and student name of another database belongs to the same entity.
- *Detecting and resolving data value concepts*: The data taken from different databases while merging may differ. Like the attribute values from one database may differ from another database [1].

3.3.3 Data Reduction

This process helps in the reduction of the volume of the data which makes the analysis easier yet produces the same or almost the same result. This reduction also helps to reduce storage space. There are some of the techniques in data reduction are Dimensionality reduction, Numerosity reduction, Data compression [1] [4].

- **Dimensionality reduction:** This process is necessary for real-world applications as the data size is big. In this process, the reduction of random variables or attributes is done so that the dimensionality of the data set can be reduced. Combining and merging the attributes of the data without losing its original characteristics. This also helps in the reduction of storage space and computation time is reduced. When the data is highly dimensional the problem called "*Curse of Dimensionality*" occurs.
- **Numerosity Reduction**: In this method, the representation of the data is made smaller by reducing the volume. There will not be any loss of data in this reduction.
- **Data compression:** The compressed form of data is called data compression. This compression can be lossless or lossy. When there is no loss of information during compression it is called lossless compression. Whereas lossy compression reduces information but it removes only the unnecessary information.

3.3.4 Data Transformation

The change made in the format or the structure of the data is called data transformation. This step can be simple or complex based on the requirements. There are some methods in data transformation [4].

- **Smoothing**: With the help of algorithms, noises are removed from the dataset and this helps in knowing the important features of the dataset. By smoothing we can find even a simple change.
- **Aggregation**: In this method, the data is stored and presented in the form of a summary. The data set which is from multiple sources is integrated into with data analysis description.
- **Discretization**: The continuous data here is split into intervals. Discretization reduces the data size.
- **Normalization**: It is the method of scaling the data so that it can be represented in a smaller range. Example ranging from -1.0 to 1.0.

3.4 Data Visualization

The process of finding trends and correlations in the data by representing it pictorially is called Data Visualization. To perform data visualization in python, we will use the various python data visualization modules such as Matplotlib, Seaborn and Plotly are used. Data visualization is a field in data analysis that deals with visual representation of data. It graphically plots data and is an effective way to communicate inferences from data [1] [3] [4] [5].

Using data visualization, a visual summary of the dataset can be obtained. With pictures, maps and graphs, it becomes very easy to get the insight about the data. Data visualization plays a significant role in the representation of both small and large data sets, but it is especially useful when the dataset is large, in which it is impossible to see all of the available data. Python offers several plotting libraries, namely Matplotlib, Seaborn and many other such data visualization packages with different features for creating informative, customized, and appealing plots to present data in the most simple and effective way [3] [5].

Matplotlib and Seaborn are two python libraries used for data visualization. They have inbuilt modules for plotting different graphs. While Matplotlib is used to embed graphs into applications, Seaborn is primarily used for statistical graphs [5].

Other different data visualization techniques are *Line Charts*, *Bar Graphs*, *Histograms*, *Scatter Plots*, *Violin-plot and Box-plot*. These techniques are discussed herewith for brief understanding.

3.4.1 Line Charts

A line chart as shown in fig. 3.5 is a graph that represents information as a series of data points connected by a straight line. In line charts, each data point or marker is plotted and connected with a line or curve.

```
years = range(2000, 2012)
apples = [0.895, 0.91, 0.919, 0.926, 0.929, 0.931, 0.934, 0.936, 0.937, 0.9375, 0.9372, 0.939]
oranges = [0.962, 0.941, 0.930, 0.923, 0.918, 0.908, 0.907, 0.904, 0.901, 0.898, 0.9, 0.896, ]
```

```
plt.plot(years, apples)
plt.plot(years, oranges)
plt.xlabel('Year')
plt.ylabel('Yield (tons per hectare)');
```

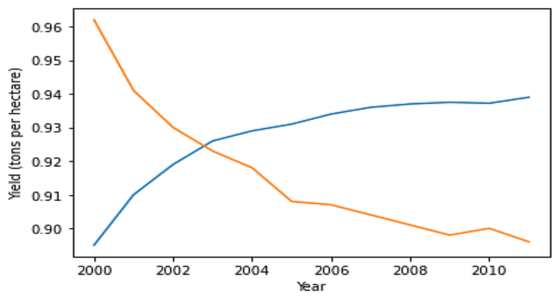


Fig 3.5 Line Chart.

3.4.2 Bar Graphs

When the data is categorical data, it can be represented with a bar graph. A bar graph plots data with the help of bars, which represent value on the y-axis and category on the x-axis. Bar graphs use bars with varying heights to show the data which belongs to a specific category. The fig. 3.6 represents information as bar graph of datapoints with help of vertical bars.

```
years = range(2000, 2006)
apples = [0.35, 0.6, 0.9, 0.8, 0.65, 0.8]
oranges = [0.4, 0.8, 0.9, 0.7, 0.6, 0.8]
```

```
plt.bar(years, oranges)

plt.xlabel('Year')
plt.ylabel('Yield (tons per hectare)')

plt.title("Crop Yields in Kanto")
```

```
<BarContainer object of 6 artists>
```

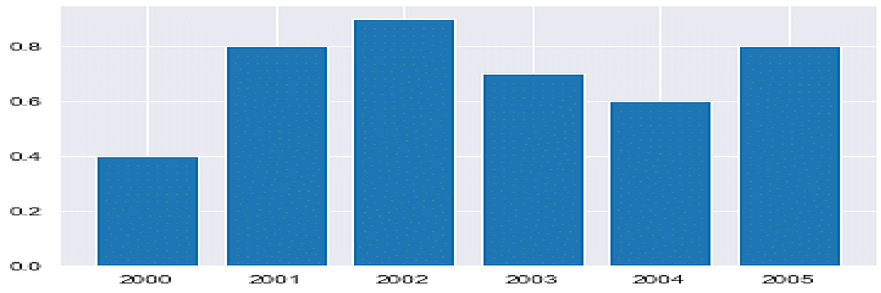


Fig 3.6 Bar Graphs.

3.4.3 Histogram

A Histogram is a bar representation of data that varies over a range. It plots the height of the data belonging to a range along the y-axis and the range along the x-axis. Histograms are used to plot data over a range of values. It uses a bar representation to show the data belonging to each range. A histogram is shown in fig. 3.7.

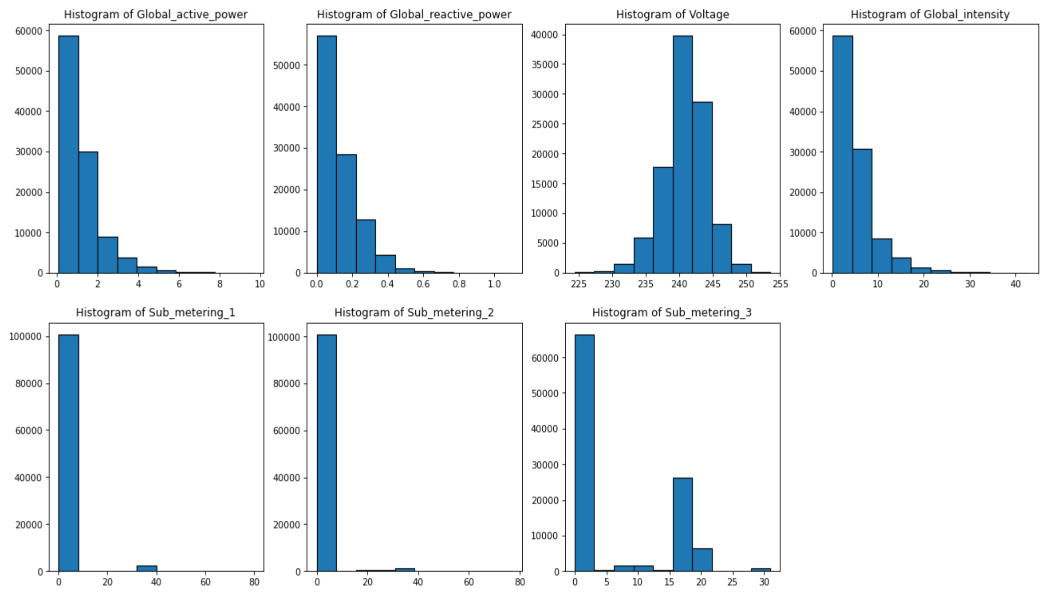


Fig 3.7 Histogram.

3.4.4 Scatter Plots

Scatter plots shown in fig. 3.8 are used to plot two or more variables present at different coordinates. The data is scattered all over the graph and is not confined to a range. Two or more variables are plotted in a Scatter Plot, with each variable being represented by a different colour.

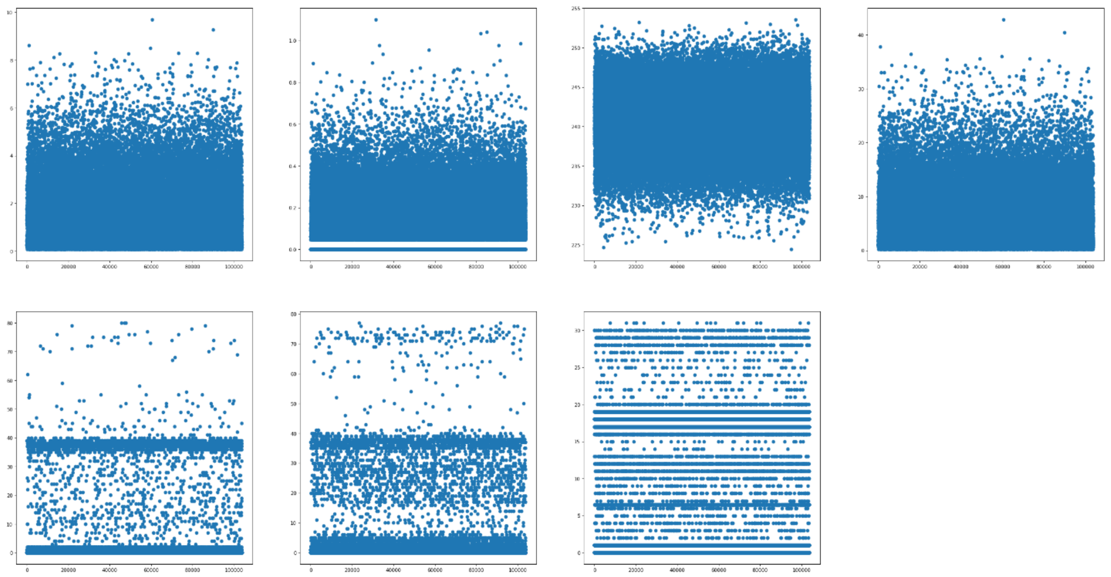


Fig 3.8 Scatter Plot

3.4.5 Violin-Plot

A violin plot shown in fig.3.9 is a method of plotting numeric data. It is similar to a box plot, with the addition of a rotated kernel density plot on each side. Violin plots are similar to box plots, except that it also shows the probability density of the data at different values, usually smoothed by a kernel density estimator.

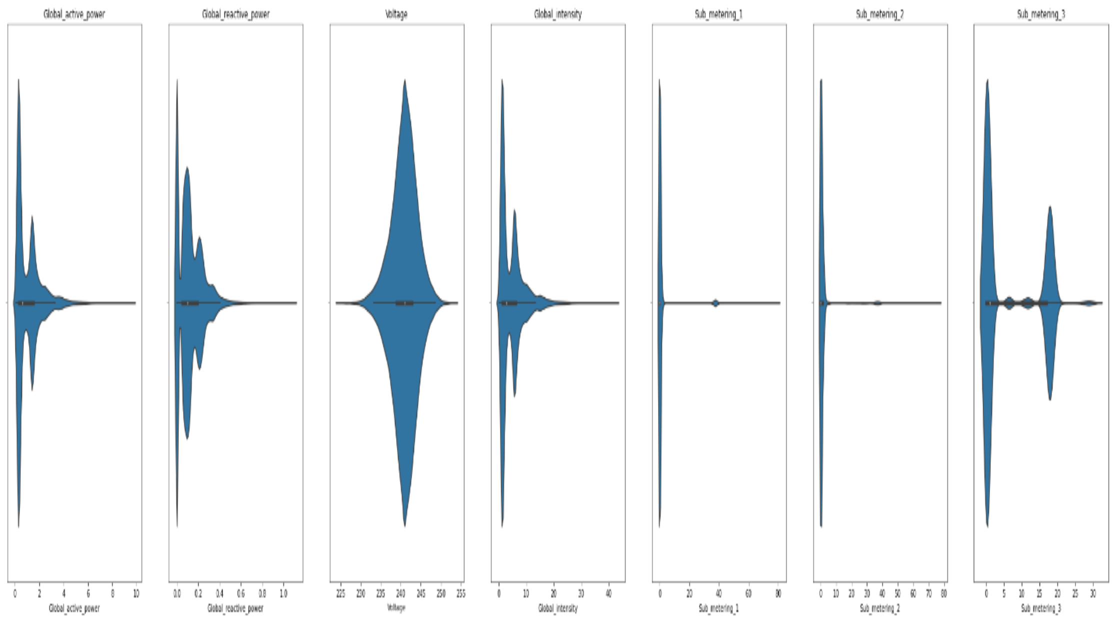


Fig 3.9 Violin Plot.

3.4.6 Box-Plot

A box plot shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable. The box shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution, except for points that are determined to be "outliers" using a method that is a function of the inter-quartile range. The box plot is represented in fig 3.10.

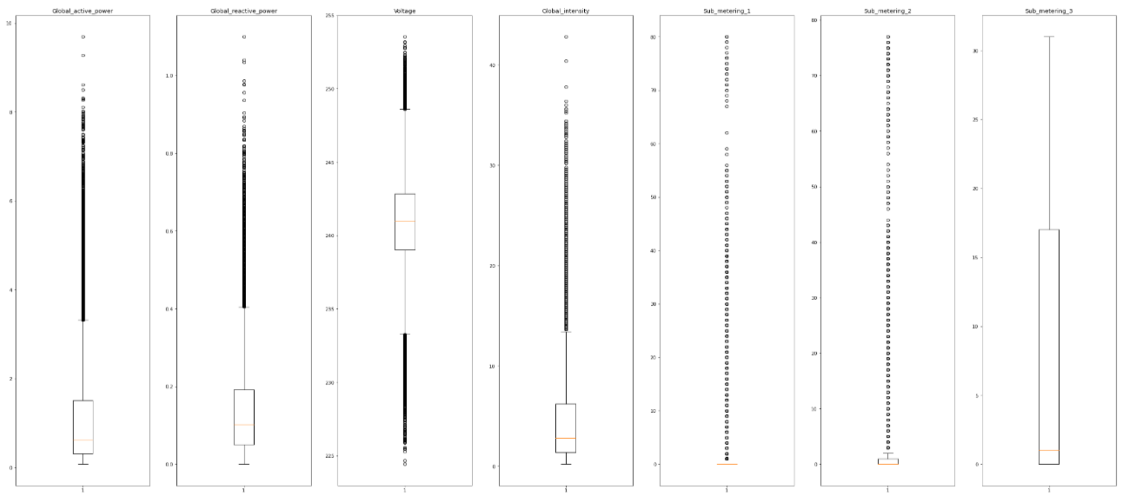


Fig 3.10 Box Plot

3.5 Training, Testing and Evaluating Machine Learning Models

Training a machine learning model is a process in which a machine learning algorithm is fed with training data from which it can learn. ML models can be trained to benefit businesses in numerous ways, by quickly processing huge volumes of data, identifying patterns, finding anomalies or testing correlations that would be difficult for a human to do unaided [1] [3] [4] [7].

3.5.1 Splitting the dataset

The data collected for training needs to be split into three different sets: training, validation and test as shown in fig 3.11. The code snippet for splitting the dataset is shown in fig 3.12.

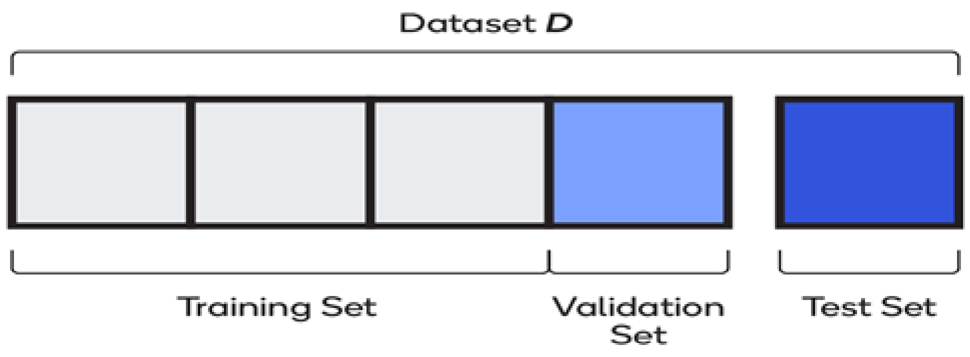
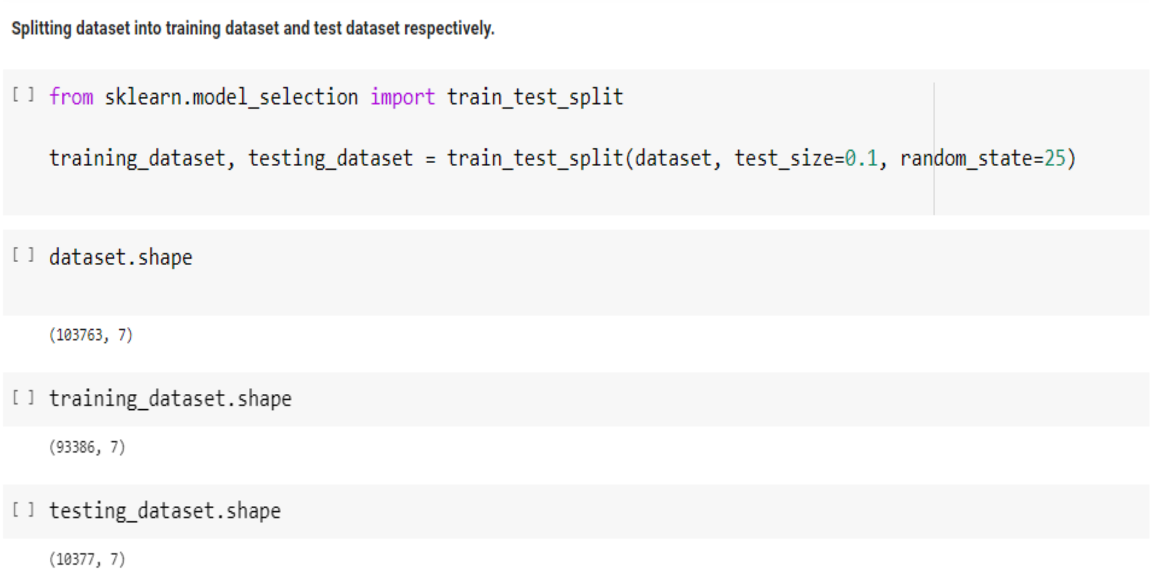


Fig 3.11 Distribution of dataset into different segment

Training Dataset: Up to 75 percent of the total dataset is used for training. The model learns on the training set; in other words, the set is used to assign the weights and biases that go into the model. Training dataset is depicted in fig 3.13.

Validation Dataset: Between 15 and 20 percent of the data is used while the model is being trained, for evaluating initial accuracy, seeing how the model learns and fine-tuning hyperparameters. The model sees validation data but does not use it to learn weights and biases.

Testing Dataset: Between five and 10 percent of the data is used for final evaluation. Having never seen this dataset, the model is free of any of its bias. Testing dataset is depicted in fig 3.14.



Splitting dataset into training dataset and test dataset respectively.

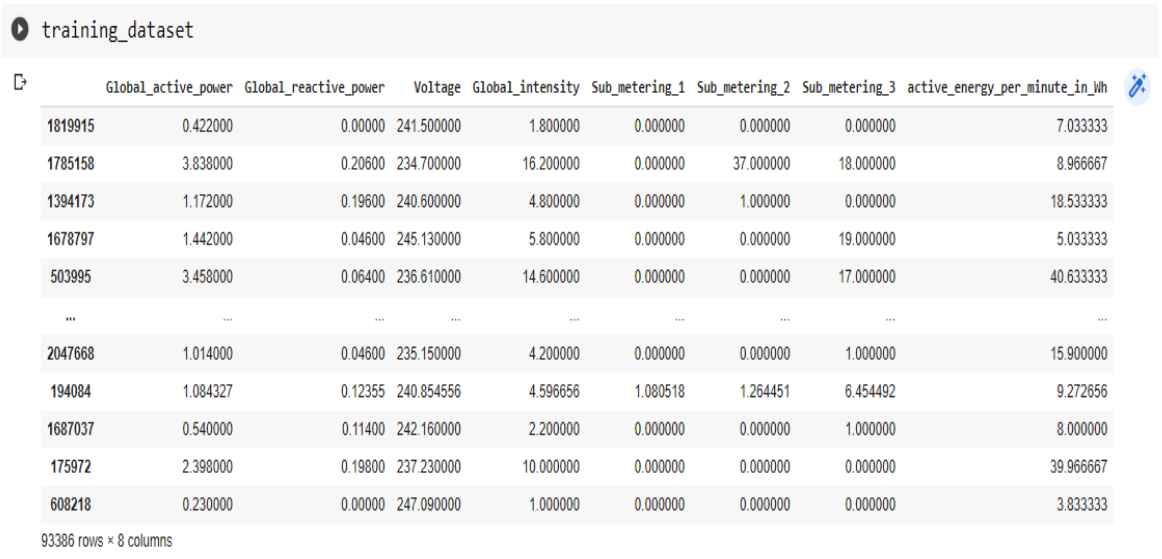
```
[ ] from sklearn.model_selection import train_test_split

    training_dataset, testing_dataset = train_test_split(dataset, test_size=0.1, random_state=25)
```

```
[ ] dataset.shape

    (103763, 7)
```

```
[ ] training_dataset.shape

    (93386, 7)
```

```
[ ] testing_dataset.shape

    (10377, 7)
```

Fig 3.12 Code snippet for splitting of the dataset.

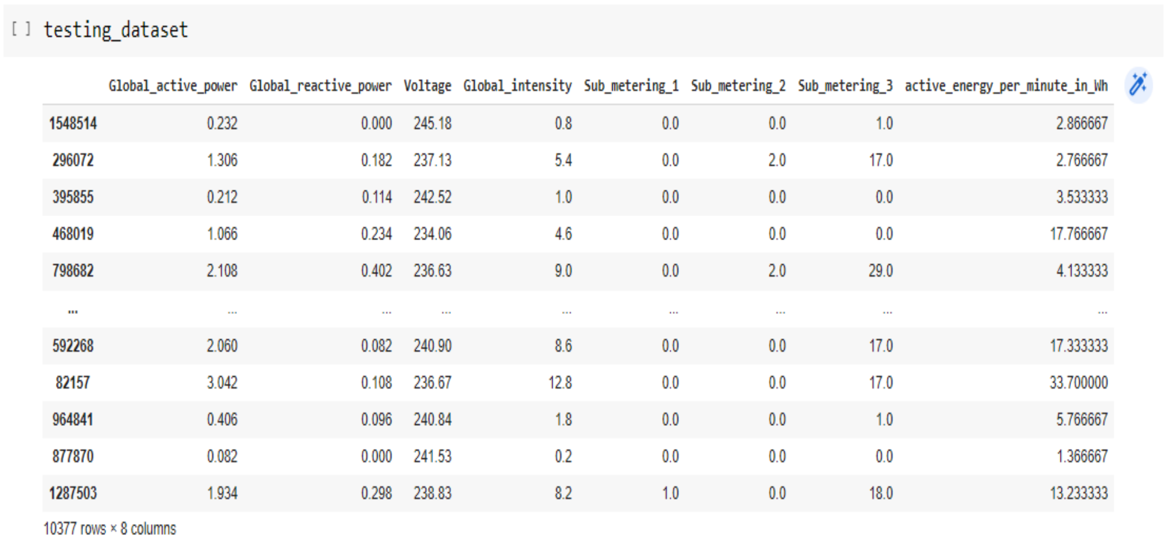


```
training_dataset
```

	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3	active_energy_per_minute_in_Wh
1819915	0.422000	0.00000	241.500000	1.800000	0.000000	0.000000	0.000000	7.033333
1785158	3.838000	0.20600	234.700000	16.200000	0.000000	37.000000	18.000000	8.966667
1394173	1.172000	0.19600	240.600000	4.800000	0.000000	1.000000	0.000000	18.533333
1678797	1.442000	0.04600	245.130000	5.800000	0.000000	0.000000	19.000000	5.033333
503995	3.458000	0.06400	236.610000	14.600000	0.000000	0.000000	17.000000	40.633333
...	...	...	...	...	...	...	...	...
2047668	1.014000	0.04600	235.150000	4.200000	0.000000	0.000000	1.000000	15.900000
194084	1.084327	0.12355	240.854556	4.596656	1.080518	1.264451	6.454492	9.272656
1687037	0.540000	0.11400	242.160000	2.200000	0.000000	0.000000	1.000000	8.000000
175972	2.398000	0.19800	237.230000	10.000000	0.000000	0.000000	0.000000	39.966667
608218	0.230000	0.00000	247.090000	1.000000	0.000000	0.000000	0.000000	3.833333

93386 rows × 8 columns

Fig 3.13 Training dataset after splitting of the 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
1548514	0.232	0.000	245.18	0.8	0.0	0.0	1.0	2.866667
296072	1.306	0.182	237.13	5.4	0.0	2.0	17.0	2.766667
395855	0.212	0.114	242.52	1.0	0.0	0.0	0.0	3.533333
468019	1.066	0.234	234.06	4.6	0.0	0.0	0.0	17.766667
798682	2.108	0.402	236.63	9.0	0.0	2.0	29.0	4.133333
...	...	...	...	...	...	...	...	...
592268	2.060	0.082	240.90	8.6	0.0	0.0	17.0	17.333333
82157	3.042	0.108	236.67	12.8	0.0	0.0	17.0	33.700000
964841	0.406	0.096	240.84	1.8	0.0	0.0	1.0	5.766667
877870	0.082	0.000	241.53	0.2	0.0	0.0	0.0	1.366667
1287503	1.934	0.298	238.83	8.2	1.0	0.0	18.0	13.233333

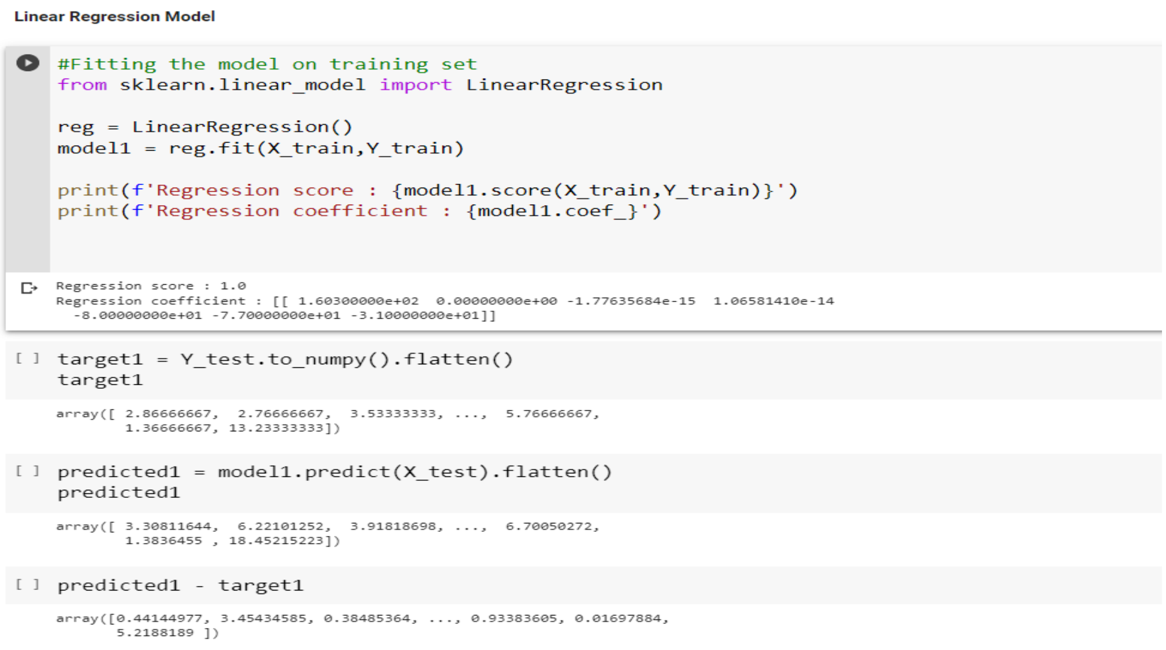
10377 rows × 8 columns

Fig 3.14 Testing dataset after splitting of the dataset.

3.5.2 Model training and testing

Model training for machine learning includes splitting the dataset, training different machine learning models, tuning hyperparameters and performing batch normalization. In this work with the available dataset the below mentioned machine learning models were trained [1] [2]:

- Linear Regression Model (Fig 3.15)
- Support Vector Machine Regression Model (linear kernel) (Fig 3.16)
- Support Vector Machine Regression Model (poly kernel) (Fig 3.17)
- Decision Tree Regression Model (Fig 3.18)
- Random Forest Regression Model (Fig 3.19)
- XgBoost Regression Model (Fig 3.20)



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
```

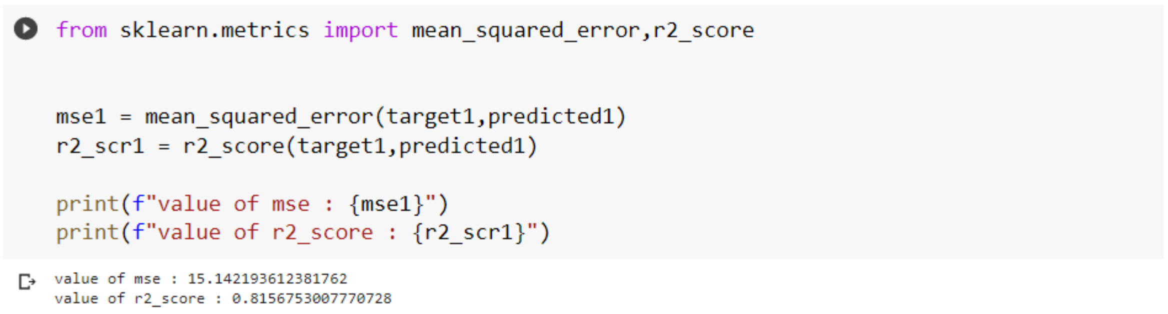
```
array([ 2.86666667,  2.76666667,  3.53333333, ...,  5.76666667,
        1.36666667, 13.23333333])
```

```
[ ] 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 ])
```

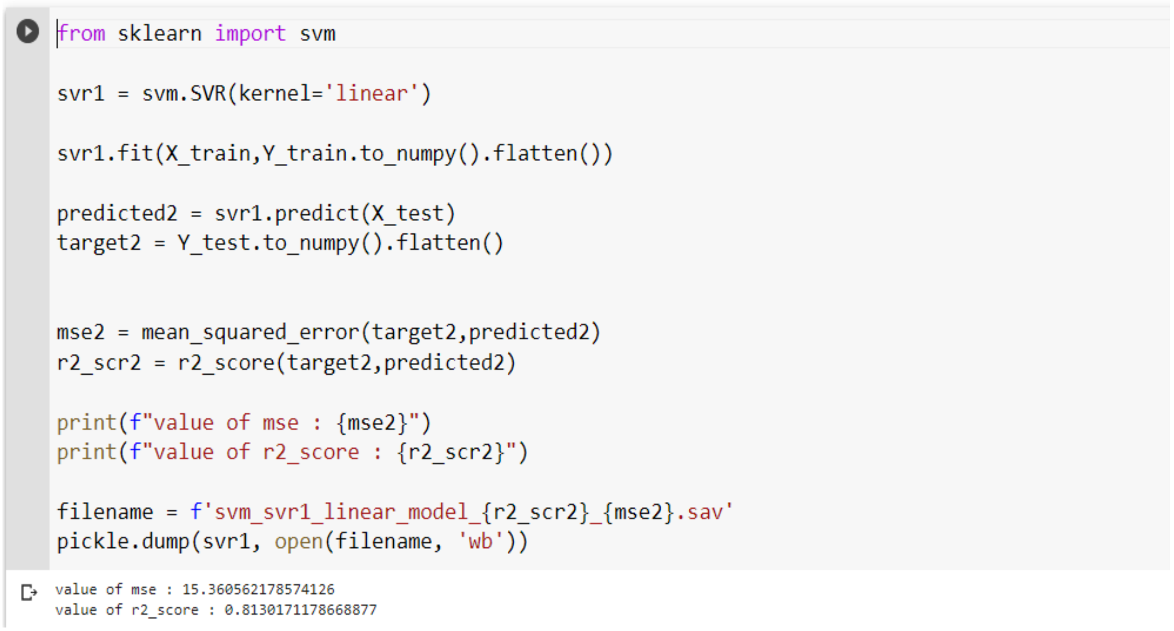


```python
from sklearn.metrics import mean_squared_error,r2_score


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 3.15 Code Snippet for training & testing of the linear regression model.

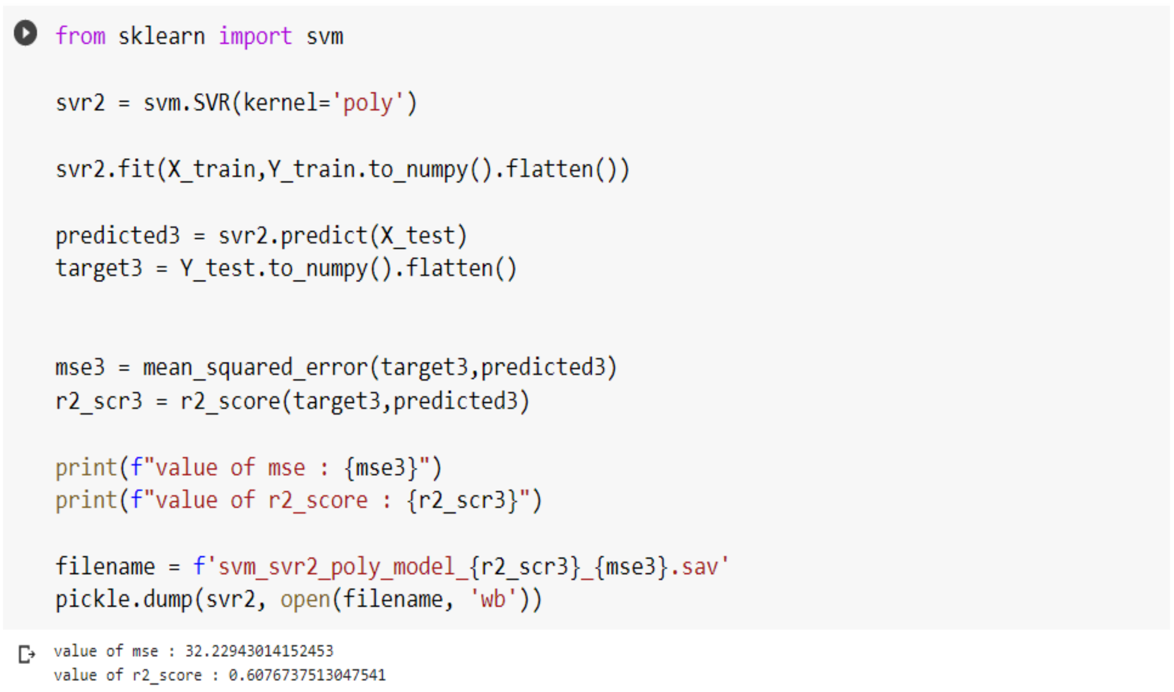
Support Vector Machine Model



```python
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
```

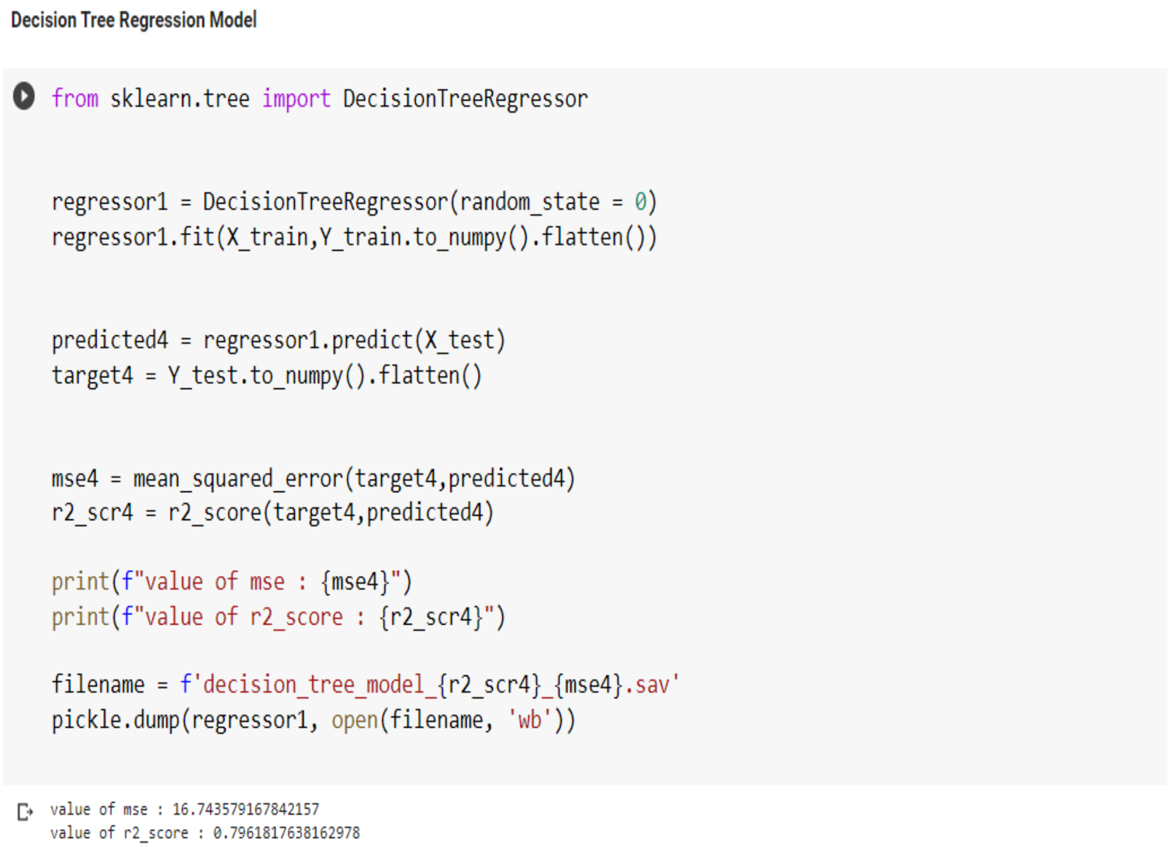
Fig 3.16 Code Snippet for training & testing of the svm regression (linear kernel) model



```python
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 3.17 Code Snippet for training & testing of the svm regression (poly kernel) model

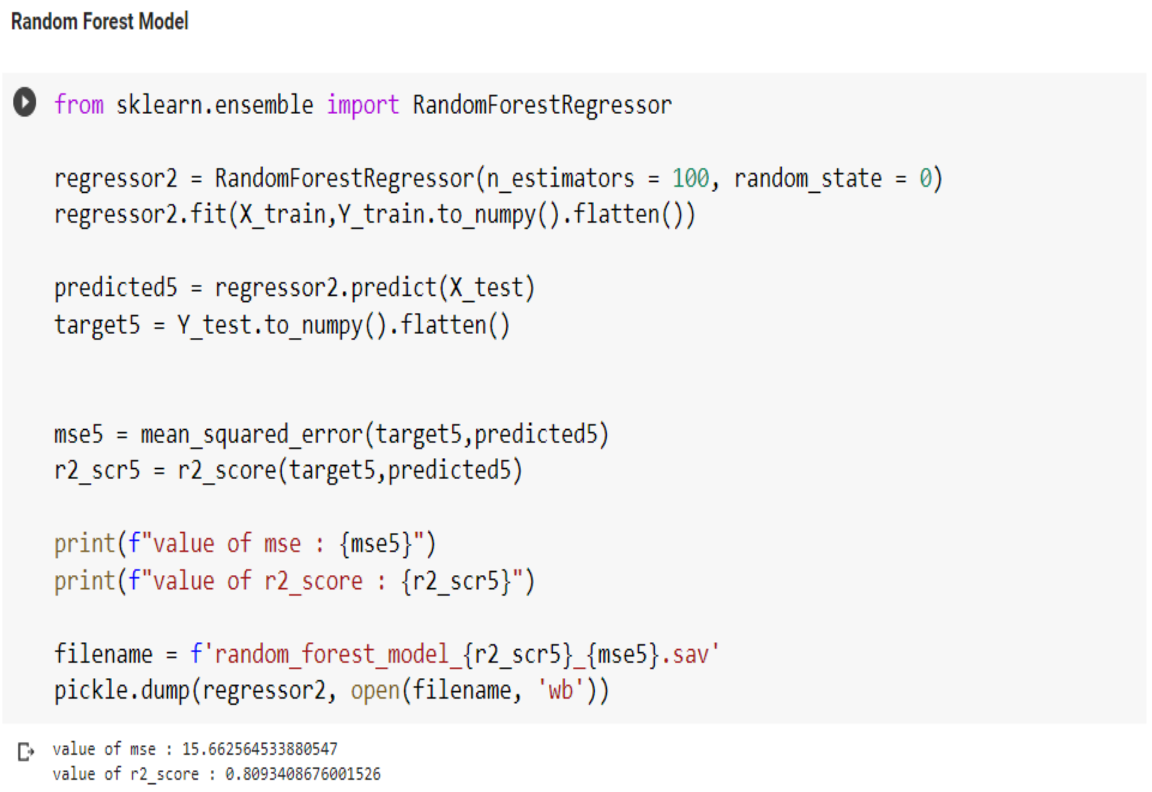


Decision Tree Regression Model

```python
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 3.18 Code Snippet for training & testing of the decision tree regression model

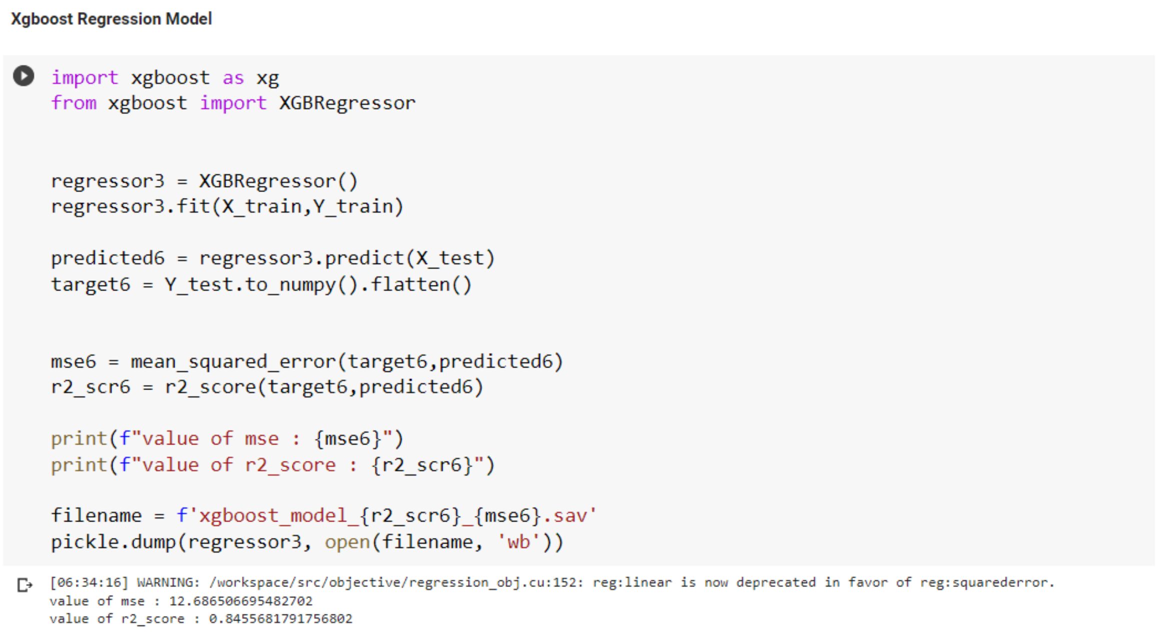


Random Forest Model

```python
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 3.19 Code Snippet for training & testing of the random forest regression model



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 3.20 Code Snippet for training & testing of the xgboost regression model

3.5.3 Hyperparameter tuning

Hyperparameters can be imagined as settings for controlling the behaviour of a training algorithm, as shown below, in fig 3.21 [9] [10] [11].

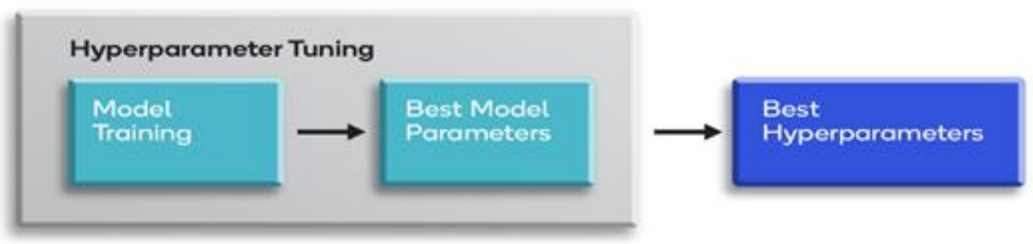


Fig 3.21 Hyperparamater tuning

There are two common approaches to tuning hyper parameters, as depicted in the fig 3.22.

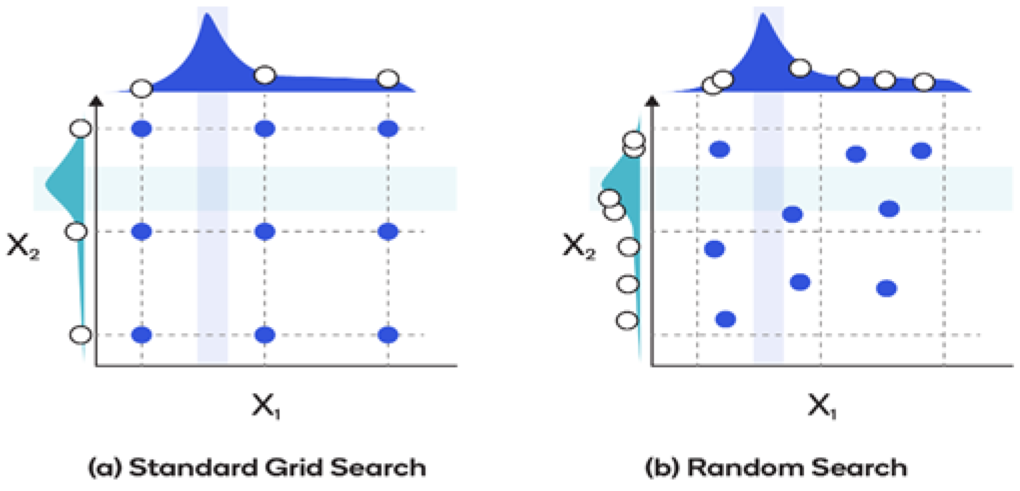


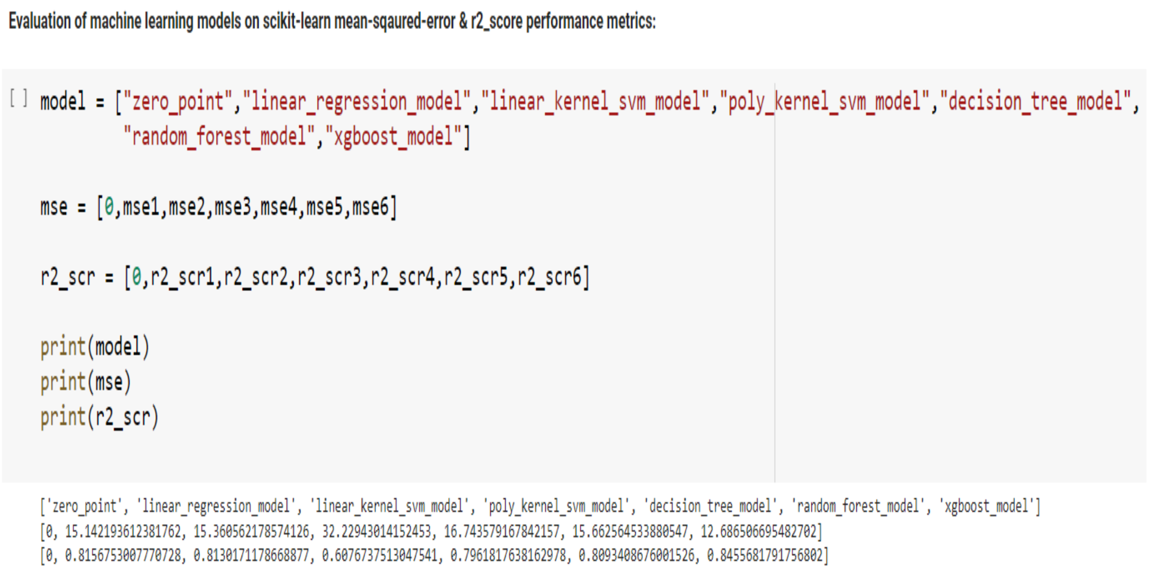
Fig 3.22 Hyperparamater tuning technique.

The first, standard grid search optimization, is a brute-force approach through a predetermined list of combinations of hyperparameters. All possible values of hyperparameters are listed and looped through in an iterative fashion to obtain the best values. Grid search optimization takes relatively little time to program and works well if the dimensionality in the feature-vector is low. But as the number of dimensions increases, tuning takes longer and longer [1] [7].

The other common approach, random search optimization, involves randomly sampling values instead of exhaustively searching through every combination of hyperparameters. Generally, it produces better results in less time than grid search optimization [7].

3.5.4 Model Evaluation

Evaluation of the machine learning models on a performance metrics is an essential step after the training of the models. It is very useful step for the selection of the appropriate model for the current need of application. There are several evaluation metrics, like r2_score, mean squared error, confusion matrix, cross-validation, AUC-ROC curve etc [7].

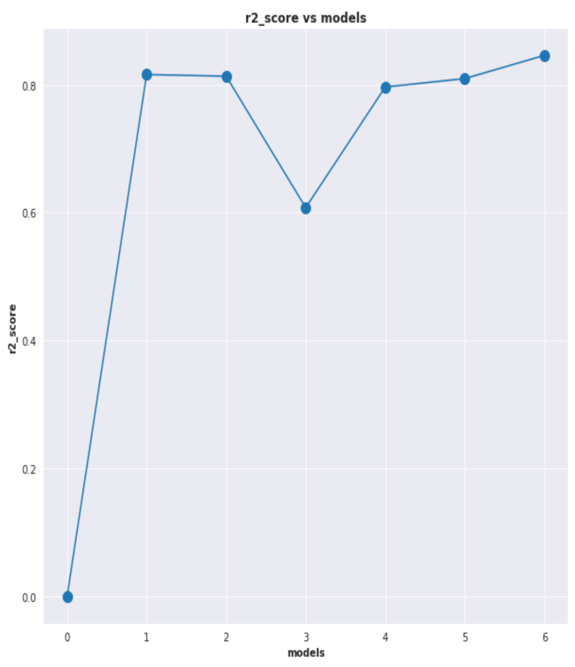

Evaluation of machine learning models on scikit-learn mean-sqaured-error & r2_score performance metrics:

```
[ ] model = ["zero_point","linear_regression_model","linear_kernel_svm_model","poly_kernel_svm_model","decision_tree_model",
             "random_forest_model","xgboost_model"]

    mse = [0,mse1,mse2,mse3,mse4,mse5,mse6]

    r2_scr = [0,r2_scr1,r2_scr2,r2_scr3,r2_scr4,r2_scr5,r2_scr6]

    print(model)
    print(mse)
    print(r2_scr)
```

```
['zero_point', 'linear_regression_model', 'linear_kernel_svm_model', 'poly_kernel_svm_model', 'decision_tree_model', 'random_forest_model', 'xgboost_model']
[0, 15.142193612381762, 15.360562178574126, 32.22943014152453, 16.743579167842157, 15.662564533880547, 12.686506695482702]
[0, 0.8156753007770728, 0.813017117866877, 0.6076737513047541, 0.7961817638162978, 0.8093408676001526, 0.8455681791756802]
```

Fig 3.23 Models and their respective mse and r2_score values.

Different evaluation metrics are used for different kinds of problems. In this work r2_score and mean squared error metrics of the scikit-learn library will be used to compare the performance of different models as shown in fig 3.23, fig 3.24 and fig 3.25.



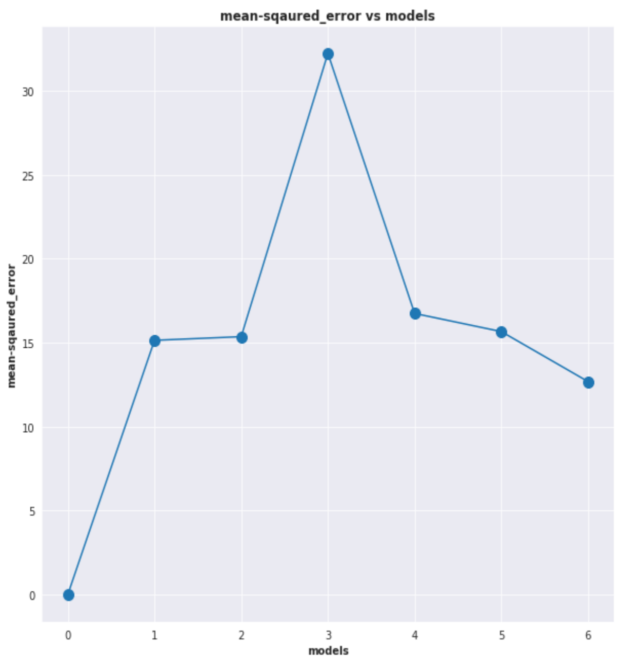


Fig 3.24 Plot of r2_score of models. Fig 3.25 Plot of mse of models.

The next chapter will discuss about the results obtained in this proposed work.

CHAPTER 4
RESULTS AND DISCUSSION

The various machine learning models were trained along with hyperparameter tuning with the help of the training dataset obtained after the splitting of the dataset. After the training of the various machine learning models with the help of testing dataset, r2_score metric of scikit-learn and mean squared error metric of scikit-learn their performance were checked.

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, Xgboost model as tabulated in table 4.1

From the table, it can be clearly observed that, the Xgboost model has the highest r2_score. The obtained score is satisfactorily acceptable for the final inference purpose.

TABLE 4.1: COMPARISON BETWEEN MEAN SQUARED ERROR & R2_SCORE

Model	**r2_score**	**mean-squared_error**
Linear regression model	0.8156	15.1421
Support vector machine model with linear kernel	0.8130	15.3605
Support vector machine model with poly kernel	0.6076	32.2294
Decision tree model	0.7961	16.7435
Random forest model	0.8093	15.6625
Xgboost model	0.8455	12.6865

The fig 4.1 and fig 4.2 obtained by the training in the earlier chapter depict the acceptable result from the Xgboost model for mean squared error and r2_score of all the models.

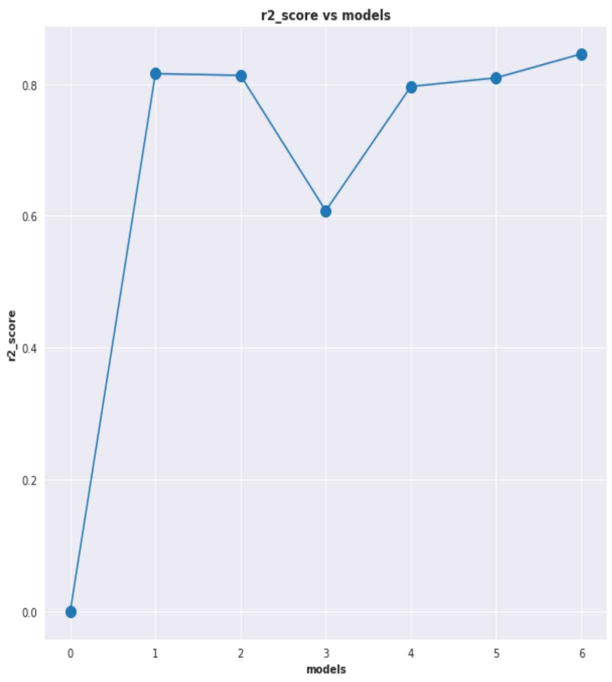


Fig 4.1 Plot of r2_score of models.

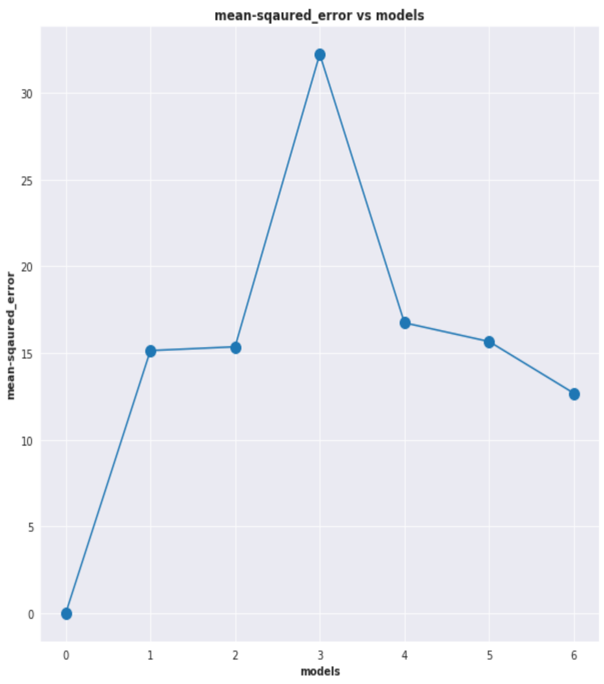


Fig 4.2 Plot of mse of models.

CHAPTER 5
CONCLUSIONS AND FUTURE SCOPES

5.1 Conclusions

Despite the increasing requirement of electrical energy, there must be a proper technology and system that can predict the upcoming requirement of the household electrical load. Based on that prediction the energy can be fulfilled as per the requirement. To address this problem, there is a need of infrastructure that can make the traditional power system more efficient and optimized.

This project reviews the machine learning technology that can be used to forecast future electrical load of household. The implementation of the project is done by the use 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 inorder to address the outlier problem.

After the pre-processing of the available dataset, the different machine learning models are trained. The different machine learning models are trained successfully by using the training dataset. It is observed that the highest r2_score is obtained for XG_boost model among all the trained models. The r2_scores of various models are shown in the table 4.1. The performance of the trained models can be improved by performing hyperparameter tuning. By this work with the help of machine learning algorithms the future household electrical load can be predicted precisely.

5.2 Future Scopes

This system provides precise prediction of future household electrical load with the help of parameters of a power system. This system can make the traditional power system more intelligent. This system can form the basis for the implementation of smart grid in future. This system can become more adaptable than the current system, allowing it to quickly predict energy demand of an area.

REFERENCES

[1] Machine Learning Course,Andrew Ng,Deeplearning.ai, Stand-ford University.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*.MIT Press, 2016.

[3] Python, https://docs.python.org/3/

[4] Pandas, https://pandas.pydata.org/docs/

[5] Matplotlib, https://matplotlib.org/stable/

[6] Numpy, https://numpy.org/doc/

[7] Scikit, https://scikit-learn.org/stable/

[8] 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.

[9] N. Kandil, R. Wamkeue, M. Saad, and S. Georges, "An effi-cient 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.

[10] D. 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.

[11] G. 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.

[12] H. 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.

LIST OF PUBLICATION

01. Sanskar R., Kunal R. Singh, Piyush K., Roopam Kr. Raj, Soham V., Amit K. Choudhary, “***Forecasting of future household electrical load using machine learning algorithm***” ***IEEE transactions on Power Systems***. ***(Communicated)***