

EVALUATING THE IMPACT OF ENERGY-SAVING MEASURES ON HOUSEHOLD POWER CONSUMPTION

Submitted in partial fulfillment of the requirements for the
award of Bachelor of Engineering Degree in Computer
Science and Engineering

By

GUNDAMPALLI SAMSON REDDY
(40732002)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status
by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI - 600119

APRIL - 2023



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **GUNDAMPALLI SAMSON REDDY (40732002)** who carried out the project entitled **"Evaluating The Impact Energy-Saving Measures On Household Power Consumption"** under my supervision from FEBRUARY 2023 to APRIL 2023.

Internal Guide

Ms S. SARASWATHI M.E

Head of the Department

Dr. S. VIGNESHWARI M.E., Ph.D.,

Submitted for Viva voce Examination held on

17-04-2023

Internal Examiner

External Examiner

DECLARATION

I **GUNDAMPALLI SAMSON REDDY(40732002)** hereby declare that the Project Report entitled **Evaluating The Impact Energy-Saving Measures On Household Power Consumption** done by me under the guidance of **Ms S.SARASWATHI M.E** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering with Specialization in Internet Of Things**.

DATE: 17-04-2023

PLACE: CHENNAI



**SIGNATURE OF THE
CANDIDATE**

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. T.Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr. S. Vigneshwari M.E., Ph.D.** Head of the Department of Computer Science and Engineering for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **Ms. S. Saraswathi. M.E.** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

CERTIFICATE

ABSTRACT

Household power consumption helps the power supply department understand the power consumption of residents and whether there will be some abnormal power consumption phenomena. Taking the individual household electric power consumption dataset as an example, this paper establishes an extensible experimental analysis framework and analyzes the data in a visual way.

In the experiment, the effects of linear regression model and neural network model on different characteristics are compared. The experiment shows that the effect of neural network model is better than linear regression model in the experimental dataset.

This project report outlines the process of analyzing household power consumption data to identify trends and patterns in energy usage. The report covers data collection, preprocessing, analysis, and energy efficiency recommendations. By using statistical analysis techniques and data visualization tools, we gain insights into the data and identify areas where energy efficiency improvements can be made. The recommendations can help households save energy and reduce their carbon footprint. Overall, this project contributes to the goal of achieving sustainable energy consumption patterns.

CHAPTER No.	TITLE	PAGE No
	ABSTRACT	i
	LIST OF FIGURES	ii
	LIST OF TABLES	iii
	LIST OF ABBREVIATIONS	iv
1.	INTRODUCTION 1.1 Advantages of household power consumption 1.2 Disadvantages of household power consumption 1.3 Python Programming	1-6
2.	2.1 Aim of household power consumption 2.2 SCOPE	7-8
3.	EXPERIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED 3.1 DATA SOURCE AND DESCRIPTION 3.1.1 DATA SOURCE 3.1.2 DATA DESCRIPTION 3.2 PREPROCESSING 3.2.1 Cleaning Noisy Data 3.2.2 Handling Missing Values 3.2.3 HANDLING CATEGORICAL DATA 3.2.4 FEATURE SELECTION 3.3 MACHINE LEARNING MODELS 3.3.1 RANDOM FOREST 3.3.2 SUPPORT VECTOR MACHINE 3.3.3 DECISION TREE 3.3.4 REGRESSION MODELS	9-14

4	RESULTS AND DISCUSSIONS	15-17
5.	SUMMARY AND CONCLUSIONS	18-19
	REFERENCES	20
	APPENDIX	
	A. SCREENSHOTS	21
	B. SOURCE CODE	28

LIST OF FIGURES

FIGURE NO	PAGE NO
FIG NO 1.1	2
FIG NO 2.1	7
FIG NO 3.1	10
FIG NO 3.3	13

LIST OF ABBREVIATIONS

ABBREVIATION EXPANSION

DT	Decision Tree
SVM	Support Vector Machine
RF	Random Forest

CHAPTER – 1

INTRODUCTION

The data of household power consumption can not only reflect the situation of household power consumption, but also provide message for the power sector to help understand the power supply. In addition, the household power consumption data can also be used as a reference for the power bureau to collect electricity charges. At the same time, the abnormal situation of the data can also be verified through the historical information of the household power consumption data.

The Household Power Consumption analysis machine learning project is a data-driven approach to understanding how energy is used in homes. With the increasing need for sustainable living, this project aims to provide insights into household power consumption patterns and identify areas where energy can be saved. The project uses machine learning algorithms to analyze power consumption data collected from smart meters installed in households. The data is then processed to identify patterns and trends that can be used to optimize energy usage

In recent years, the research on power consumption prediction has become a hot issue . This paper takes the individual household electric power consumption dataset as an example designs an experimental framework for predicting and analyzing user power consumption data. Many methods can be applied to the prediction of user power consumption in machine learning and deep learning, but linear regression is the most basic method and can reflect the correlation between features. Although linear regression is a basic method, it is still widely used . In addition, as the basis of deep learning model, neural network model is also widely used . Therefore, this experiment used to the linear regression model and neural network model to predict the power consumption in prediction method module.

1.1 Advantages Of Analyzing Household Power Consumption

- Identifying energy inefficiencies: Analyzing household power consumption can help identify areas where energy is being wasted, such as appliances that are left on when not in use or heating/cooling systems that are running inefficiently. By identifying these inefficiencies, households can take steps to reduce their energy usage and lower their electricity bills.
- Promoting energy conservation: Analyzing household power consumption can also promote energy conservation by raising awareness of the amount of energy being used and the impact on the environment. By providing households with information on their energy usage and potential savings, they can make more informed decisions about their energy usage and adopt more sustainable energy consumption patterns.
- Supporting renewable energy adoption: Analyzing household power consumption can also support the adoption of renewable energy sources, such as solar or wind power. By identifying the times of day when energy usage is highest, households can time their energy usage to coincide with when renewable energy sources are most available, reducing reliance on non-renewable energy sources.
- Enhancing energy security: Analyzing household power consumption can also enhance energy security by reducing dependence on external energy sources. By reducing overall energy usage and promoting renewable energy adoption, households can become more self-sufficient and less vulnerable to energy supply disruptions.
- Overall, analyzing household power consumption has numerous benefits for both households and the environment, promoting energy efficiency, conservation, renewable energy adoption, and energy security.



Fig 1.1

1.2 Disadvantages of Household Power Consumption

- Intrusiveness: In order to analyze household power consumption, it may be necessary to install energy meters or smart meters in homes. Some people may feel uncomfortable with this intrusion into their privacy and may be hesitant to allow the installation of these devices.
- Cost: Installing energy meters or smart meters can be costly, especially for households that are already struggling with high electricity bills. Additionally, the cost of analyzing the data and implementing energy efficiency improvements can also be a barrier for some households.
- Technical expertise: Analyzing household power consumption data requires technical expertise in data analysis and statistics. Many households may not have the skills or resources to perform this type of analysis themselves and may need to rely on outside experts.

Unforeseen consequences: There is always the potential for unforeseen consequences when making changes to energy usage patterns. For example, implementing energy-saving devices may inadvertently lead to increased energy usage in other areas, offsetting the intended energy savings.

Overall, while there are potential disadvantages to analyzing household power consumption, the benefits of promoting energy efficiency, conservation, renewable energy adoption, and energy security likely outweigh these drawbacks. It is important to carefully consider the potential drawbacks and work to mitigate them where possible to ensure a successful and sustainable energy consumption analysis.

1.3 Python Programming

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

Python Syntax compared to other programming languages

- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions, and classes. Other programming languages often use curly-brackets for this purpose.

Facts about Python Programming Language:

- Python is the most widely used multi-purpose, high-level programming language.

- Python allows programming in Object-Oriented and Procedural paradigms.
- Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
- Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc.
- The biggest strength of Python is huge collection of standard library which can be used for the following:
 - Machine Learning
 - GUI Applications (like Kivy, Tkinter, PyQt etc.)
 - Web frameworks like Django (used by YouTube, Instagram, Dropbox)
 - Image processing (like OpenCV, Pillow)
 - Web scraping (like Scrapy, BeautifulSoup, Selenium)
 - Test frameworks
 - Multimedia
 - Scientific computing
 - Text processing and many more.

Advantages of Python Programming Language:

- Smooth learning curve
- High speed of development
- Portability and extensibility to other languages
- Versatility plus extensive toolset for almost everything

Disadvantages of Python Programming Language:

- Speed limitations
- No multithreading
- High memory consumption
- Hassles with mobile and front-end development

CHAPTER 2

2.1 Aim :-

The aim of analyzing household power consumption is to gain insights into energy usage patterns and identify areas where energy efficiency improvements can be made. By analyzing power consumption data, households can become more aware of their energy usage and make informed decisions about their energy consumption patterns. This can help reduce energy waste, lower electricity bills, and reduce carbon emissions. Additionally, analyzing household power consumption can support the adoption of renewable energy sources, enhance energy security, and promote a more sustainable and efficient use of energy resources. Overall, the aim of analyzing household power consumption is to promote energy efficiency, conservation, and sustainability.



shutterstock.com • 1926916646

Fig 2.1

2.2 SCOPE

- Data collection: Collecting data on household power consumption through energy meters or smart meters, and storing the data in a database or spreadsheet.
- Data preprocessing: Preprocessing the data to remove any errors, missing

values, or outliers that may affect the analysis.

- Data analysis: Analyzing the data using statistical analysis techniques to identify trends and patterns in energy usage, such as peak usage times or high energy-consuming appliances.
- Energy efficiency recommendations: Based on the analysis, providing energy efficiency recommendations to households on how to reduce energy consumption and lower electricity bills, such as upgrading to energy-efficient appliances or adjusting usage patterns.
- Implementation of energy efficiency improvements: Assisting households in implementing energy efficiency improvements, such as installing smart thermostats or LED light bulbs, and monitoring the impact on energy consumption.
- Monitoring and evaluation: Monitoring and evaluating the impact of the energy efficiency improvements over time to ensure they are effective and sustainable.

The scope of analyzing household power consumption can vary depending on the specific goals and objectives of the analysis, as well as the resources available for data collection and analysis. However, the overall aim is to promote energy efficiency, conservation, and sustainability in households.

CHAPTER – 3

MATERIALS, METHODS AND ALGORITHMS USED

3.1 DATA SOURCE AND DESCRIPTION

3.1.1 DATA SOURCE

The dataset which contains a set of features through which power consumption can be calculated, is to be collected. You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository etc.

3.1.2 DATA DESCRIPTION

The data description for a household power consumption dataset typically includes:

1. Data format: The format in which the data is provided, such as CSV, Excel, or SQL.
2. Data size: The size of the dataset in terms of the number of rows and columns.
3. Variables: A list of variables or features included in the dataset, such as date and time, voltage, current, and power.
4. Units: The units of measurement for each variable, such as kilowatts or amps.
5. Time range: The time range for which the data was collected, such as daily, monthly, or yearly.
6. Sampling rate: The frequency at which the data was sampled, such as every minute or every hour.
7. Data quality: A description of the quality of the data, such as whether there are any missing values or outliers.
8. Data source: The source of the data, such as a smart meter or a manually recorded log.
9. Data cleaning: A description of any preprocessing or cleaning that was done on the data to remove errors or inconsistencies.

The data description provides important information about the dataset, which is necessary for understanding the data and performing accurate analysis. It helps researchers and analysts to determine the suitability of the dataset for their specific research objectives and to prepare the data for analysis.

3.2 PREPROCESSING

Real world data is often inconsistent which can affect the performances of models. Preprocessing the data before it is fed into classifiers is vital part of developing machine-learning model. Similarly, the dataset for this study contains missing values that needs to be handled appropriately. It has to also be in a suitable format for modeling. Hence, pre-processing has been conducted as it has been shown in Fig.3.1

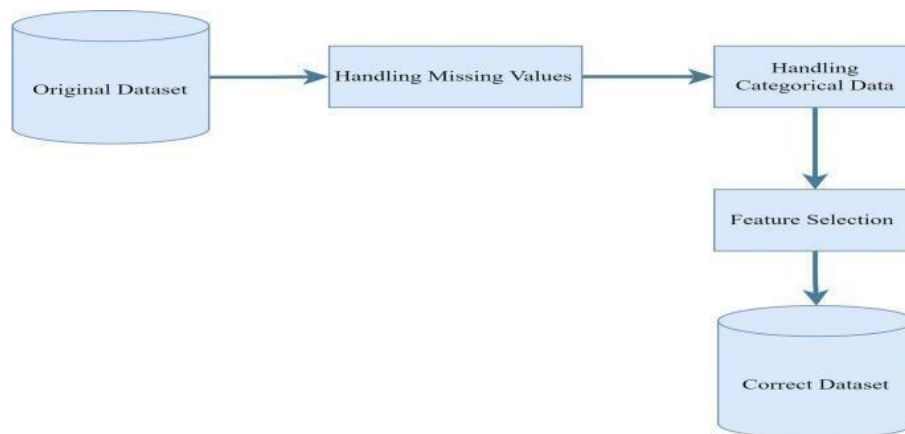


FIG 3.1 DATA PREPROCESSING

3.2.1 Cleaning Noisy Data

Removing outlier and smoothening noisy data is an important part of preprocessing. Outliers are values that lies away from the range of the rest of the values. In clinical data, outliers may arise from the natural variance of data. The potential outliers are the data points that fall above $Q3 + 1.5(IQR)$ and below $Q1 - 1.5$

(IQR) , where $Q1$ is the first quartile, $Q3$ is the third quartile, and $IQR = Q3 - Q1$.

3.2.2 Handling Missing Values

Data is not always available (or missed) due to equipment malfunction, inconsistent with other recorded data and thus deleted, not entered into the database due to misunderstanding, some data may not be considered important at the time of entry. Patient data often has missing diagnostic test results that would help to predict the likelihood of diagnoses or predict treatment's effectiveness. The missing values have an impact on the performance of the prediction model. There are several ways of handling missing values including dropping missing values and filling missing values. Sometimes missing values are ignored if they are small percentage i.e., if missing data under 10%. But it is not considered healthy for the model because the missing value can be an important feature contributing to the model development. Sometimes the missing values can be also replaced by zero, which will bring no change to the model. To handle these missing values the mean, an average of the observed features used in this study, because the missing features are numeric and mean imputation is better for numerical missing values.

3.2.3 HANDLING CATEGORICAL DATA

In this step, data has been transformed into the required format. The nominal data converted into numerical data of the form 0 and 1. For instance, 'Gender' has the nominal value that can be labeled as 0-for female and 1-for male. After preprocessing the data then the resultant CSV file comprises all the integer and float values for different CKD related features.

3.2.4 FEATURE SELECTION

Identify subset of relevant predictive features is important for quality result. Feature selection is the process of selecting most important predictive features to use them as input for models. It is important preprocessing step to deal with the problem of high dimensionality. Hence, the main aim of feature selection is to select the subset of features that are relevant and independent of each other for

training the model. Similarly, feature selection is crucial to develop chronic kidney disease predictive model. This reduces the dimensionality and complexity of the data and makes the model be faster, more effective and accurate. Hence, feature selection algorithm have been used to select relevant features after the construction of the dataset.

3.3 MACHINE LEARNING MODELS

The aim of the study was to predict chronic kidney disease using machine learning techniques. Three machine learning algorithms; Random Forest, Support Vector Machine and Decision Tree have been used in this project. The algorithms were selected based on their popularity in chronic kidney disease prediction and their performance of classification on previous research works.

3.3.1 RANDOM FOREST

Random Forest is ensemble learning that consists of several collections of decision trees. It is used for both classification and regression. This model comprises of a number of decision trees and outputs the class target that is the highest voting results of the target output by each tree. Random Forest uses both bagging and random feature selection to build the tree and creates an uncorrelated forest of trees. Its prediction by the group is more accurate than that of any individual tree. After it builds the forest, test instances are permeated down through each tree and trees make their respective prediction of class.

3.3.2 SUPPORT VECTOR MACHINE

Support Vector Machine is one of the prominent and convenient supervised machine-learning algorithm that can be used for classification, learning and prediction. A set of hyperplanes are built to classify all input in high dimensional data. A discrete hyperplane is created in the signifier space of the training data and compounds are classified based on the side of the hyperplane. Hyperplanes are decision boundaries that separate the data points. Support vectors are data points that are closer to the hyperplane and determine the position and orientation

of the hyperplane. SVMs have been mainly proposed to deal with binary classification, but nowadays many researchers have tried to apply it to multiclass classification because there are a huge amount of data to be classified into more than two classes in today's world. SVM solves multiclass problems through the two most popular approaches; one-versus-rest and one-vs-one. In this study, one-versus-rest has been used. For multi classification, we used OVR with the SVM algorithm. This method separates each class from the rest of the classes in the dataset. Besides, because Linear SVC is used in this study, one vs rest is an appropriate method with Linear SVC.

3.3.3 DECISION TREE

It is one of the most popular supervised machine-learning algorithms that can be used for classification. Decision Tree solves the problem of machine learning by transforming the data into a tree representation through sorted feature values. Each node in a decision tree denotes features in an instance to be classified, and each leaf node represents a class label the instances belong to. This model uses a tree structure to split the dataset based on the condition as a predictive model that maps observations about an item to make a decision on the target value of instances.

3.3.4 REGRESSION MODELS

Regression models, such as linear regression, can be used to predict energy consumption based on variables such as time of day, weather conditions, and appliance usage patterns. Linear regression is a type of machine learning model that is commonly used for predicting numerical values, such as household power consumption. The goal of linear regression is to fit a line or plane to the data that best represents the relationship between the input variables (e.g. time of day, weather conditions, appliance usage) and the output variable (e.g. power consumption).

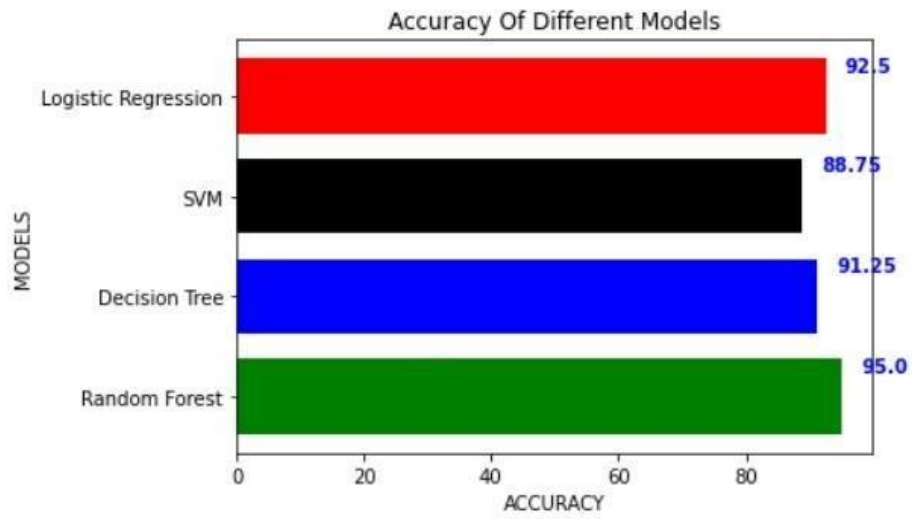
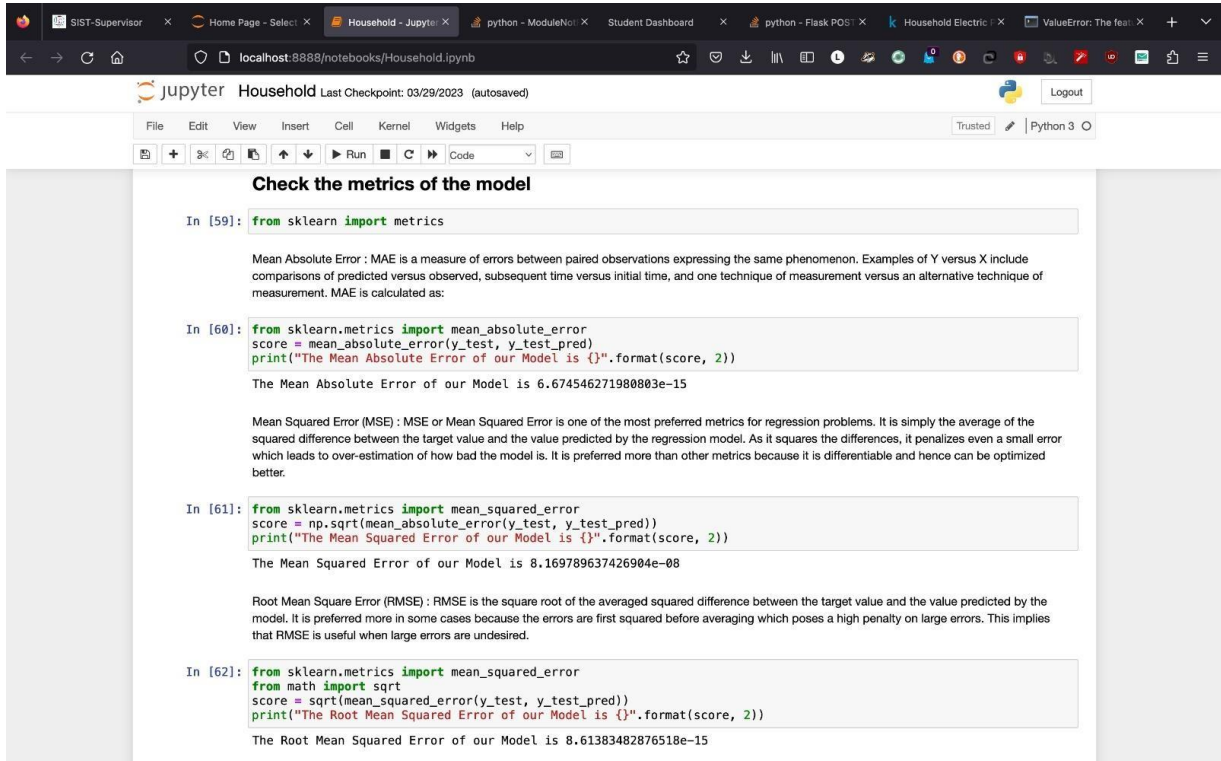


FIG 3.3 ACCURACY OF DIFFERENT MODELS

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 RESULT



The screenshot shows a Jupyter Notebook titled "Household" with a last checkpoint of 03/29/2023. The notebook is running on a Python 3 kernel. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running cells, and other functions. The notebook content is as follows:

Check the metrics of the model

In [59]: `from sklearn import metrics`

Mean Absolute Error : MAE is a measure of errors between paired observations expressing the same phenomenon. Examples of Y versus X include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement. MAE is calculated as:

In [60]: `from sklearn.metrics import mean_absolute_error
score = mean_absolute_error(y_test, y_test_pred)
print("The Mean Absolute Error of our Model is {}".format(score, 2))`

The Mean Absolute Error of our Model is 6.674546271980803e-15

Mean Squared Error (MSE) : MSE or Mean Squared Error is one of the most preferred metrics for regression problems. It is simply the average of the squared difference between the target value and the value predicted by the regression model. As it squares the differences, it penalizes even a small error which leads to over-estimation of how bad the model is. It is preferred more than other metrics because it is differentiable and hence can be optimized better.

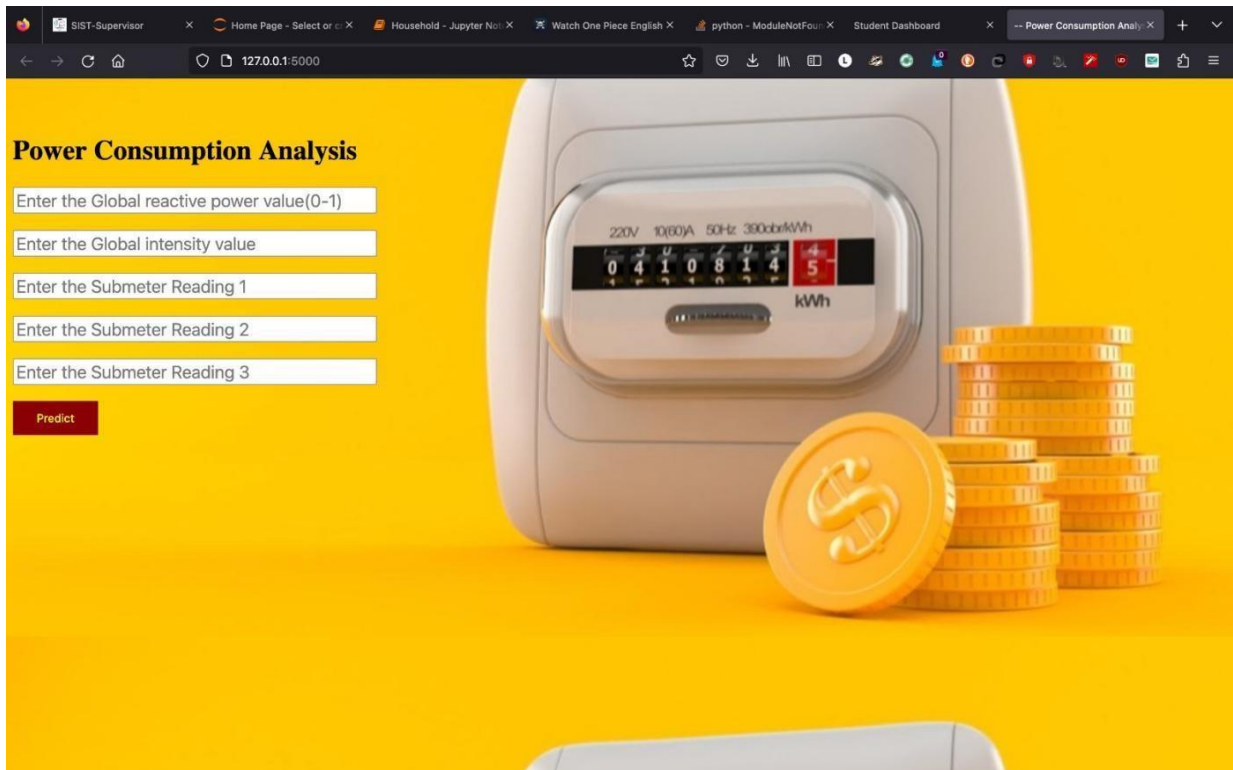
In [61]: `from sklearn.metrics import mean_squared_error
score = np.sqrt(mean_squared_error(y_test, y_test_pred))
print("The Mean Squared Error of our Model is {}".format(score, 2))`

The Mean Squared Error of our Model is 8.169789637426904e-08

Root Mean Square Error (RMSE) : RMSE is the square root of the averaged squared difference between the target value and the value predicted by the model. It is preferred more in some cases because the errors are first squared before averaging which poses a high penalty on large errors. This implies that RMSE is useful when large errors are undesired.

In [62]: `from sklearn.metrics import mean_squared_error
from math import sqrt
score = sqrt(mean_squared_error(y_test, y_test_pred))
print("The Root Mean Squared Error of our Model is {}".format(score, 2))`

The Root Mean Squared Error of our Model is 8.61383482876518e-15



4.2 DISCUSSIONS

There are several important discussions to be had around household power consumption analysis. One key discussion is around the benefits of conducting this type of analysis. By understanding energy usage patterns and identifying factors that contribute to energy waste or inefficiency, households, building managers, and policymakers can take action to reduce energy consumption, save money on energy bills, and reduce carbon emissions.

Another discussion is around the limitations and challenges of conducting a household power consumption analysis. For example, collecting accurate and representative data can be challenging, as energy usage can vary significantly depending on factors such as climate, time of year, and individual behavior. Additionally, choosing the appropriate

machine learning model and evaluating its performance can be complex, requiring careful consideration of factors such as model complexity, interpretability, and bias.

There are also important ethical considerations to be taken into account when conducting a household power consumption analysis. For example, data privacy and security must be carefully managed to protect the confidentiality and rights of individuals whose energy usage data is being analyzed. Additionally, care must be taken to ensure that the analysis does not inadvertently reinforce or exacerbate existing inequalities or biases.

Finally, it is important to consider the broader societal and environmental implications of household power consumption analysis. By reducing energy consumption and carbon emissions, these types of analyses can contribute to broader efforts to mitigate climate change and promote sustainable development. However, it is important to ensure that these efforts are conducted in an equitable and just manner, taking into account the needs and perspectives of all stakeholders.

CHAPTER 5

SUMMARY AND CONCLUSION

5.1 SUMMARY

A household power consumption analysis is an important task that can help to identify patterns and factors that affect energy consumption in homes, buildings, or communities. The analysis can be conducted using various machine learning models, which can be trained on historical data to predict future energy usage patterns and identify factors that contribute to energy waste or inefficiency.

One common machine learning model used for household power consumption analysis is linear regression. Linear regression is a simple model that assumes a linear relationship between the input variables and the output variable. It is often used to predict energy consumption based on factors such as time of day, weather conditions, and appliance usage patterns. The model fits a line or plane to the data that best represents the relationship between the input variables and the output variable. The model also assumes that the error or residual between the predicted values and the actual values is normally distributed.

To conduct a household power consumption analysis using linear regression, the input data is first split into training and testing sets. The model is trained on the training set using an algorithm that minimizes the sum of the squared residuals. Once the model is trained, it can be used to predict the output variable for new input data.

The results of a household power consumption analysis can be evaluated using various metrics, such as mean squared error (MSE), root mean squared error (RMSE), coefficient of determination (R-squared), and accuracy. A good model should have low error and high accuracy in predicting energy consumption for new data.

It is important to carefully evaluate and interpret the results of a household power consumption analysis, taking into account the specific research question and any limitations or assumptions associated with the chosen machine learning model. The

results should be communicated clearly and effectively, using visualizations and other methods to convey the key insights and findings to stakeholders and decision-makers.

In summary, a household power consumption analysis using linear regression or other machine learning models can help to identify patterns and factors that affect energy consumption, and can inform strategies for energy efficiency and conservation in homes, buildings, and communities.

5.2 CONCLUSION

In conclusion, household power consumption analysis using machine learning models such as linear regression can be a powerful tool for understanding energy usage patterns and identifying factors that contribute to energy waste or inefficiency. By carefully collecting and analyzing energy usage data, households, building managers, and policymakers can take action to reduce energy consumption, save money on energy bills, and reduce carbon emissions.

However, it is important to recognize the limitations and challenges of conducting a household power consumption analysis, and to approach this type of work with care and sensitivity to ethical and societal considerations. By doing so, we can help to promote a more sustainable and equitable future, while ensuring that the benefits of energy efficiency and conservation are shared by all.

REFERENCES

- Akbari, M., & Alinejad-Rokny, H. (2020). Energy consumption prediction and demand-side management of residential buildings using machine learning algorithms. *Energy*, 192, 116604.
- Sharma, S., Tyagi, V. V., & Chen, C. R. (2021). Predictive modeling of residential electricity consumption using machine learning algorithms. *Journal of Cleaner Production*, 280, 124309.
- Duan, L., & Liu, X. (2020). Energy consumption prediction for residential buildings using machine learning algorithms. *Energy Procedia*, 176, 594-600.

Appendix

Source Code:

Importing the libraries and Read the Dataset

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
dataset = pd.read_csv('household_power_consumption .txt', sep=';',
header=0, low_memory=False, infer_datetime_format=True,
parse_dates={'datetime':[0,1]}, index_col=['datetime'])
```

```
# summarize
print(dataset.shape)
print(dataset.head())
```

```
dataset.isna().sum()
```

```
# save updated dataset
dataset.to_csv('household_power_consumption.csv')
```

```
dataset
```

```
dataset.tail()
```

```
print(f"The Dataset has {dataset.shape[0]} rows and {dataset.shape[1]}
columns")
```

```
dataset.columns
```

```
# Preprocess the data
```

In this milestone, we will be preprocessing the dataset that is collected.
Preprocessing includes:

- Handling the null values.
- Handling the categorical values if any.
- Normalize the data if required.
- Identify the dependent and independent variables.
- Split the dataset into train and test sets.

```

dataset.isnull().sum()

percent_missing = dataset.isnull().sum() * 100 / len(dataset)
missing_value_df = pd.DataFrame({'percent_missing': percent_missing})

missing_value_df

#### Handling missing data

dataset.loc[dataset.Sub_metering_3.isnull()].head()

dataset.replace('?', np.nan, inplace=True)

dataset.loc[dataset.Sub_metering_3.isnull()].head()

dataset=dataset.dropna(how='all')

for i in dataset.columns:
    dataset[i]=dataset[i].astype('float64')

dataset.shape

## Adding another sub_metering column

values = dataset.values
dataset['Sub_metering_4'] = (values[:,0] * 100 / 60) - (values[:,4] +
values[:,5] + values[:,6])

dataset.dtypes

dataset.describe().T

dataset.index

print('Dataset start', dataset.index.min())
print('Dataset end', dataset.index.max())
print('Dataset spans', dataset.index.max() - dataset.index.min())

# Data Visualization

```

```

sns.distplot(dataset['Global_active_power'])

sns.distplot(dataset['Global_active_power'],kde=False,bins=30)

sns.distplot(dataset['Global_reactive_power'],kde=False,bins=30)

sns.distplot(dataset['Voltage'],kde=False,bins=30)

sns.distplot(dataset['Global_intensity'],kde=False,bins=30)

## Correlation of dataset values

dataset.corr()

## Analysis using Heatmap

pearson = dataset.corr(method = 'pearson')
mask = np.zeros_like(pearson)
mask[np.triu_indices_from(mask)] = True
sns.heatmap(pearson, vmax = 1, vmin = 0, square = True, cbar = True,
annot = True, cmap = "YlGnBu", mask = mask);

sns.jointplot( x = 'Global_reactive_power', y = 'Global_active_power', data
= dataset, kind = 'scatter')

sns.jointplot( x = 'Voltage', y = 'Global_active_power', data = dataset, kind
= 'scatter')

sns.jointplot( x = 'Global_intensity', y = 'Global_active_power', data =
dataset, kind = 'scatter')

sns.jointplot( x = 'Sub_metering_1', y = 'Global_active_power', data =
dataset, kind = 'scatter')

sns.jointplot( x = 'Sub_metering_2', y = 'Global_active_power', data =
dataset, kind = 'scatter')

sns.jointplot( x = 'Sub_metering_3', y = 'Global_active_power', data =
dataset, kind = 'scatter')

```

```
sns.jointplot( x = 'Sub_metering_4', y = 'Global_active_power', data =  
dataset, kind = 'scatter')
```

Divide the model into Train and Test data.

Dependent and Independent Columns

In this activity, the dependent and independent variables are to be identified.

1)The independent variable in the dataset would be considered as 'x'.

2)The dependent variable in the dataset would be considered as 'y'.

```
from sklearn.model_selection import train_test_split
```

```
y = dataset['Global_active_power']  
y.head()
```

```
x = dataset.drop(columns = 'Global_active_power',axis = 1)  
x.head()
```

```
x_train,x_test,y_train,y_test = train_test_split(x, y, test_size = 0.3,  
random_state = 101)
```

```
print(x_train.shape)  
print(x_test.shape)  
print(y_train.shape)  
print(y_test.shape)
```

```
x_train,x_test,y_train,y_test
```

Model Building

There are several Machine learning algorithms to be used depending on the data you are going to process such as images, sound, text, and

numerical values. The algorithms can be chosen according to the objective. As per the dataset which we are using, it seems to be a regression problem, therefore you can use the following :

- Linear Regression
- Random Forest Regressor
- Decision Tree Regressor
- XGBoost Regressor
- and many more.

Training the Model

```
import statsmodels.api as sm
```

```
x_train_sm = sm.add_constant(x_train)
```

```
lr = sm.OLS(y_train, x_train_sm).fit()
```

```
lr.params
```

```
lr.summary()
```

Prediction

```
y_train_pred = lr.predict(x_train_sm)
```

```
res = (y_train - y_train_pred)
```

```
fig = plt.figure()
```

```
sns.distplot(res, bins = 15)
```

```
plt.title('Error Terms', fontsize = 15)
```

```
plt.xlabel('y_train - y_train_pred', fontsize = 15)
```

```
plt.show()
```

```
x_test_sm = sm.add_constant(x_test)
```

```
y_test_pred = lr.predict(x_test_sm)
```

```
y_test_pred
```

```
from sklearn.model_selection import train_test_split
```

```
x_train_lm, x_test_lm, y_train_lm, y_test_lm = train_test_split(x, y,
```

```
train_size = 0.7, test_size = 0.3, random_state = 100)
```

```
# Shape of the train set without adding column  
x_train_lm.shape
```

```
# Adding additional column to the train and test data  
x_train_lm = x_train_lm.values.reshape(-1,1)  
x_test_lm = x_test_lm.values.reshape(-1,1)
```

```
print(x_train_lm.shape)  
print(x_test_lm.shape)
```

```
from sklearn.linear_model import LinearRegression
```

```
lm = LinearRegression()
```

```
lm.fit(x_train,y_train)
```

```
predictions = lm.predict(x_test)
```

```
predictions
```

```
lm.score(x,y)
```

```
lm.coef_
```

```
## Check the metrics of the model
```

```
from sklearn import metrics
```

Mean Absolute Error :

MAE is a measure of errors between paired observations expressing the same phenomenon.

Examples of Y versus X include comparisons of predicted versus observed, subsequent time

versus initial time, and one technique of measurement versus an alternative technique of measurement.

MAE is calculated as:

```
from sklearn.metrics import mean_absolute_error
```

```
score = mean_absolute_error(y_test, y_test_pred)
print("The Mean Absolute Error of our Model is {}".format(score, 2))
```

Mean Squared Error (MSE) : MSE or Mean Squared Error is one of the most preferred metrics for regression problems.

It is simply the average of the squared difference between the target value and the value predicted by the

regression model. As it squares the differences, it penalizes even a small error which leads to over-estimation

of how bad the model is. It is preferred more than other metrics because it is differentiable and hence can be optimized better.

```
from sklearn.metrics import mean_squared_error
score = np.sqrt(mean_squared_error(y_test, y_test_pred))
print("The Mean Squared Error of our Model is {}".format(score, 2))
```

Root Mean Square Error (RMSE) : RMSE is the square root of the averaged squared difference between the target value

and the value predicted by the model. It is preferred more in some cases because the errors are first squared

before averaging which poses a high penalty on large errors. This implies that RMSE is useful when large errors are undesired.

```
from sklearn.metrics import mean_squared_error
from math import sqrt
score = sqrt(mean_squared_error(y_test, y_test_pred))
print("The Root Mean Squared Error of our Model is {}".format(score, 2))
```

```
# Save The model
```

```
import pickle
filename = 'PCASSS_model.pkl'
pickle.dump(lm, open(filename, 'wb'))
```

Screenshots:

The first screenshot shows the Jupyter Notebook interface with the title 'Household' and a last checkpoint of 03/29/2023. The notebook contains two input cells. The first cell imports the necessary libraries: numpy, pandas, seaborn, and matplotlib. The second cell reads a CSV file named 'household_power_consumption.txt' and prints its shape and the first few rows. The output shows a dataset with 2075259 rows and 7 columns. The columns are: datetime, Global_active_power, Global_reactive_power, Voltage, Global_intensity, Sub_metering_1, and Sub_metering_2. The first few rows of data are displayed.

```
In [1]: import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

In [2]: dataset = pd.read_csv('household_power_consumption.txt', sep=';', header=0, low_memory=False, infer_datetime_format=True)
# summarize
print(dataset.shape)
print(dataset.head())
```

The output of the second cell shows the dataset shape and the first few rows of data:

```
(2075259, 7)
datetime      Global_active_power Global_reactive_power Voltage \
2006-12-16 17:24:00      4.216      0.418 234.840
2006-12-16 17:25:00      5.360      0.436 233.630
2006-12-16 17:26:00      5.374      0.498 233.290
2006-12-16 17:27:00      5.388      0.502 233.740
2006-12-16 17:28:00      3.666      0.528 235.680
```

The second screenshot shows the Jupyter Notebook interface with the title 'Household' and a last checkpoint of 03/29/2023. The notebook contains three input cells. The first cell checks for missing values using the isna() method and prints the sum of missing values for each column. The second cell saves the dataset to a CSV file named 'household_power_consumption.csv'. The third cell prints the dataset object. The output of the first cell shows that there are no missing values in the dataset. The output of the second cell shows that the dataset has been saved to the specified file. The output of the third cell shows the dataset object.

```
In [3]: dataset.isna().sum()

Out[3]: Global_active_power      0
Global_reactive_power      0
Voltage      0
Global_intensity      0
Sub_metering_1      0
Sub_metering_2      0
Sub_metering_3      25979
dtype: int64

In [4]: # save updated dataset
dataset.to_csv('household_power_consumption.csv')

In [5]: dataset

Out[5]:
```

The output of the third cell shows the dataset object with the following columns: datetime, Global_active_power, Global_reactive_power, Voltage, Global_intensity, Sub_metering_1, Sub_metering_2, and Sub_metering_3. The first few rows of data are displayed.

```
Global_active_power Global_reactive_power Voltage Global_intensity Sub_metering_1 Sub_metering_2 Sub_metering_3
datetime
2006-12-16 17:24:00      4.216      0.418 234.840      18.400      0.000      1.000      17.0
2006-12-16 17:25:00      5.360      0.436 233.630      23.000      0.000      1.000      16.0
2006-12-16 17:26:00      5.374      0.498 233.290      23.000      0.000      2.000      17.0
2006-12-16 17:27:00      5.388      0.502 233.740      23.000      0.000      1.000      17.0
2006-12-16 17:28:00      3.666      0.528 235.680      15.800      0.000      1.000      17.0
...
2010-11-26 20:58:00      0.946      0.000 240.430      4.000      0.000      0.000      0.0
2010-11-26 20:59:00      0.944      0.000 240.000      4.000      0.000      0.000      0.0
2010-11-26 21:00:00      0.938      0.000 239.820      3.800      0.000      0.000      0.0
2010-11-26 21:01:00      0.934      0.000 239.700      3.800      0.000      0.000      0.0
2010-11-26 21:02:00      0.932      0.000 239.550      3.800      0.000      0.000      0.0
```

The output of the third cell shows the dataset object with the following columns: datetime, Global_active_power, Global_reactive_power, Voltage, Global_intensity, Sub_metering_1, Sub_metering_2, and Sub_metering_3. The first few rows of data are displayed.

```
2075259 rows x 7 columns
```

Household Last Checkpoint: 03/29/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```

2006-12-16 17:28:00    3.666    0.528  235.680    15.800    0.000    1.000    17.0
...
...
2010-11-26 20:58:00    0.946    0.000  240.430    4.000    0.000    0.000    0.0
2010-11-26 20:59:00    0.944    0.000  240.000    4.000    0.000    0.000    0.0
2010-11-26 21:00:00    0.938    0.000  239.820    3.800    0.000    0.000    0.0
2010-11-26 21:01:00    0.934    0.000  239.700    3.800    0.000    0.000    0.0
2010-11-26 21:02:00    0.932    0.000  239.550    3.800    0.000    0.000    0.0
2075259 rows x 7 columns

In [6]: dataset.tail()
Out[6]:
      Global_active_power  Global_reactive_power  Voltage  Global_intensity  Sub_metering_1  Sub_metering_2  Sub_metering_3
datetime
2010-11-26 20:58:00    0.946                0.000  240.430            4.000            0.000            0.000            0.0
2010-11-26 20:59:00    0.944                0.000  240.000            4.000            0.000            0.000            0.0
2010-11-26 21:00:00    0.938                0.000  239.820            3.800            0.000            0.000            0.0
2010-11-26 21:01:00    0.934                0.000  239.700            3.800            0.000            0.000            0.0
2010-11-26 21:02:00    0.932                0.000  239.550            3.800            0.000            0.000            0.0

In [7]: print(f"The Dataset has {dataset.shape[0]} rows and {dataset.shape[1]} columns")
The Dataset has 2075259 rows and 7 columns

In [8]: dataset.columns
Out[8]: Index(['Global_active_power', 'Global_reactive_power', 'Voltage',
              'Global_intensity', 'Sub_metering_1', 'Sub_metering_2',
              'Sub_metering_3'],
              dtype='object')

```

Household Last Checkpoint: 03/29/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Preprocess the data

In this milestone, we will be preprocessing the dataset that is collected. Preprocessing includes:

- Handling the null values.
- Handling the categorical values if any.
- Normalize the data if required.
- Identify the dependent and independent variables.
- Split the dataset into train and test sets.

```

In [9]: dataset.isnull().sum()
Out[9]: Global_active_power    0
Global_reactive_power        0
Voltage                      0
Global_intensity             0
Sub_metering_1               0
Sub_metering_2               0
Sub_metering_3              25979
dtype: int64

In [10]: percent_missing = dataset.isnull().sum() * 100 / len(dataset)
missing_value_df = pd.DataFrame({'percent_missing': percent_missing})

In [11]: missing_value_df
Out[11]:
      percent_missing
Global_active_power  0.000000
Global_reactive_power  0.000000
Voltage              0.000000
Global_intensity     0.000000
Sub_metering_1       0.000000
Sub_metering_2       0.000000
Sub_metering_3       0.000000

```

Jupyter Household Last Checkpoint: 03/29/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Handling missing data

```
In [12]: dataset.loc[dataset.Sub_metering_3.isnull()].head()
```

```
Out[12]:
```

datetime	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3
2006-12-21 11:23:00	?	?	?	?	?	?	NaN
2006-12-21 11:24:00	?	?	?	?	?	?	NaN
2006-12-30 10:08:00	?	?	?	?	?	?	NaN
2006-12-30 10:09:00	?	?	?	?	?	?	NaN
2007-01-14 18:36:00	?	?	?	?	?	?	NaN

```
In [13]: dataset.replace('?', np.nan, inplace=True)
```

```
In [14]: dataset.loc[dataset.Sub_metering_3.isnull()].head()
```

```
Out[14]:
```

datetime	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3
2006-12-21 11:23:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2006-12-21 11:24:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2006-12-30 10:08:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2006-12-30 10:09:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2007-01-14 18:36:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
In [15]: dataset=dataset.dropna(how='all')
```

```
In [16]: for i in dataset.columns:
dataset[i]=dataset[i].astype('float64')
```

Jupyter Household Last Checkpoint: 03/29/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Adding another sub_metering column

```
In [18]: values = dataset.values
dataset['Sub_metering_4'] = (values[:,0] * 100 / 60) - (values[:,4] + values[:,5] + values[:,6])
```

<ipython-input-18-7125c92ce0fb>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
dataset['Sub_metering_4'] = (values[:,0] * 100 / 60) - (values[:,4] + values[:,5] + values[:,6])

```
In [19]: dataset.dtypes
```

```
Out[19]:
```

Global_active_power	float64
Global_reactive_power	float64
Voltage	float64
Global_intensity	float64
Sub_metering_1	float64
Sub_metering_2	float64
Sub_metering_3	float64
Sub_metering_4	float64
dtype:	object

```
In [20]: dataset.describe().T
```

```
Out[20]:
```

	count	mean	std	min	25%	50%	75%	max
Global_active_power	2049280.0	1.091615	1.057294	0.076	0.308	0.602000	1.528000	11.122000
Global_reactive_power	2049280.0	0.123714	0.112722	0.000	0.048	0.100000	0.194000	1.390000
Voltage	2049280.0	240.839858	3.239987	223.200	238.990	241.010000	242.890000	254.150000
Global_intensity	2049280.0	4.627759	4.444396	0.200	1.400	2.600000	6.400000	48.400000
Sub_metering_1	2049280.0	1.121923	6.153031	0.000	0.000	0.000000	0.000000	88.000000
Sub_metering_2	2049280.0	1.298520	5.822026	0.000	0.000	0.000000	1.000000	80.000000
Sub_metering_3	2049280.0	6.458417	8.457154	0.000	0.000	1.000000	17.000000	31.000000

jupyter Household Last Checkpoint: 03/29/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```

Out[20]:

```

	count	mean	std	min	25%	50%	75%	max
Global_active_power	2049280.0	1.091815	1.057294	0.076	0.308	0.602000	1.528000	11.122000
Global_reactive_power	2049280.0	0.123714	0.112722	0.000	0.048	0.100000	0.194000	1.390000
Voltage	2049280.0	240.839858	3.239987	223.200	238.990	241.010000	242.890000	254.150000
Global_intensity	2049280.0	4.627759	4.444396	0.200	1.400	2.600000	6.400000	48.400000
Sub_metering_1	2049280.0	1.121923	6.153031	0.000	0.000	0.000000	0.000000	88.000000
Sub_metering_2	2049280.0	1.298520	5.822026	0.000	0.000	0.000000	1.000000	80.000000
Sub_metering_3	2049280.0	6.458447	8.437154	0.000	0.000	1.000000	17.000000	31.000000
Sub_metering_4	2049280.0	-7.059532	11.408474	-119.820	-15.120	-0.646667	0.386667	12.216667

```

In [21]: dataset.index
Out[21]: DatetimeIndex(['2006-12-16 17:24:00', '2006-12-16 17:25:00',
                        '2006-12-16 17:26:00', '2006-12-16 17:27:00',
                        '2006-12-16 17:28:00', '2006-12-16 17:29:00',
                        '2006-12-16 17:30:00', '2006-12-16 17:31:00',
                        '2006-12-16 17:32:00', '2006-12-16 17:33:00',
                        ...,
                        '2010-11-26 20:53:00', '2010-11-26 20:54:00',
                        '2010-11-26 20:55:00', '2010-11-26 20:56:00',
                        '2010-11-26 20:57:00', '2010-11-26 20:58:00',
                        '2010-11-26 20:59:00', '2010-11-26 21:00:00',
                        '2010-11-26 21:01:00', '2010-11-26 21:02:00'],
                        dtype='datetime64[ns]', name='datetime', length=2049280, freq=None)

In [22]: print('Dataset start', dataset.index.min())
         print('Dataset end', dataset.index.max())
         print('Dataset spans', dataset.index.max() - dataset.index.min())

Dataset start 2006-12-16 17:24:00
Dataset end 2010-11-26 21:02:00
Dataset spans 1441 days 03:38:00

```

jupyter Household Last Checkpoint: 03/29/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

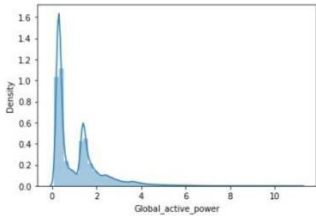
Data Visualization

```

In [23]: sns.distplot(dataset['Global_active_power'])
/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2557: FutureWarning: 'distplot' is a deprecated function and will be removed in a future version. Please adapt your code to use either 'displot' (a figure-level function with similar flexibility) or 'histplot' (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)

Out[23]: <AxesSubplot: xlabel='Global_active_power', ylabel='Density'>

```



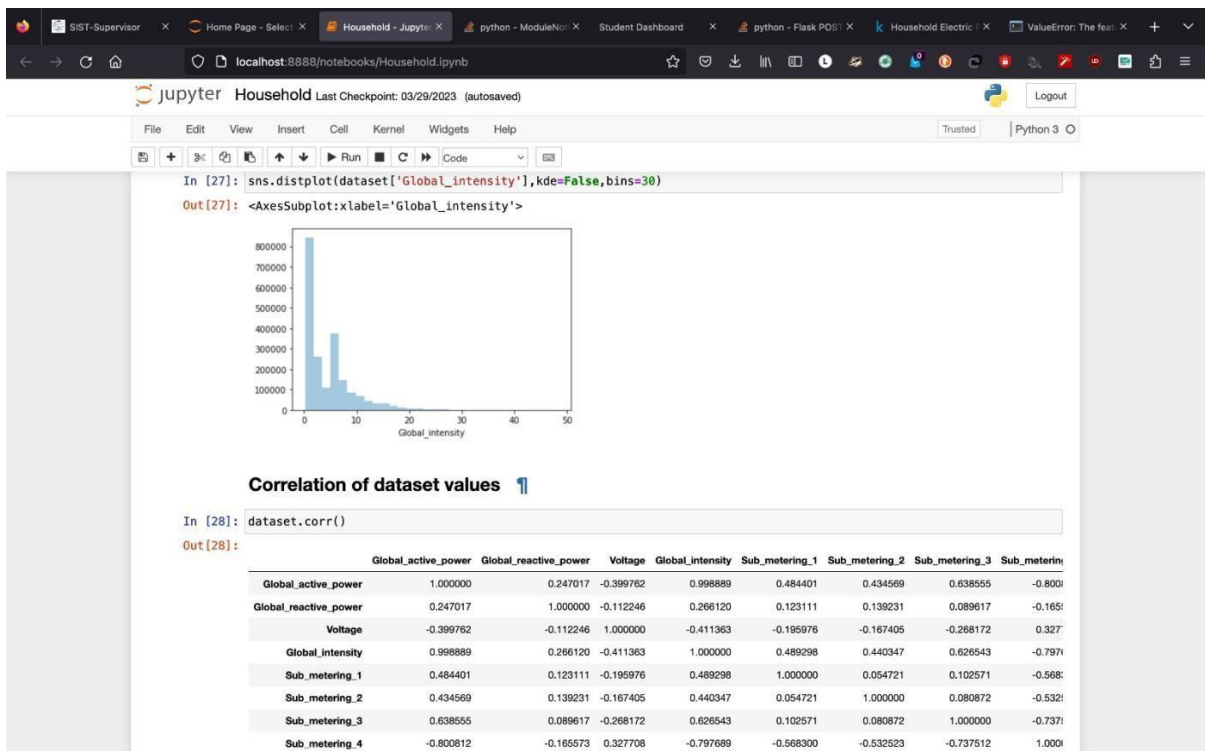
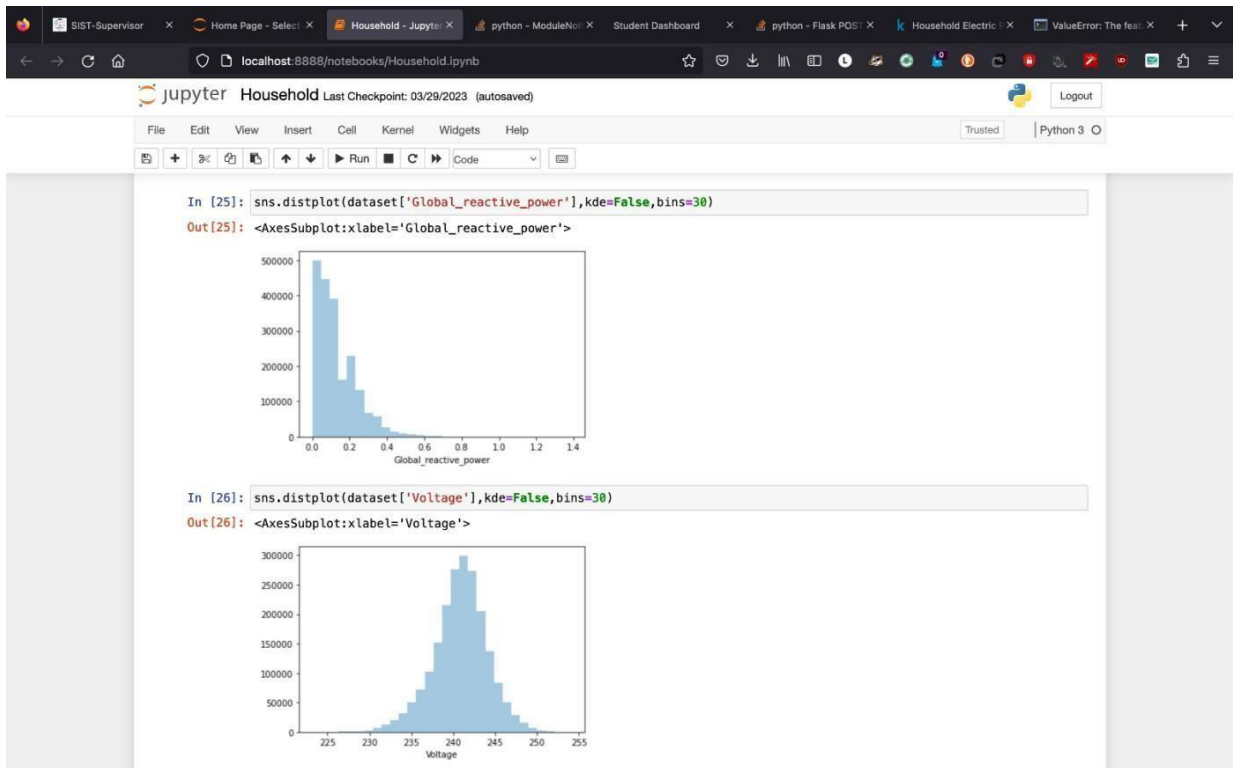
```

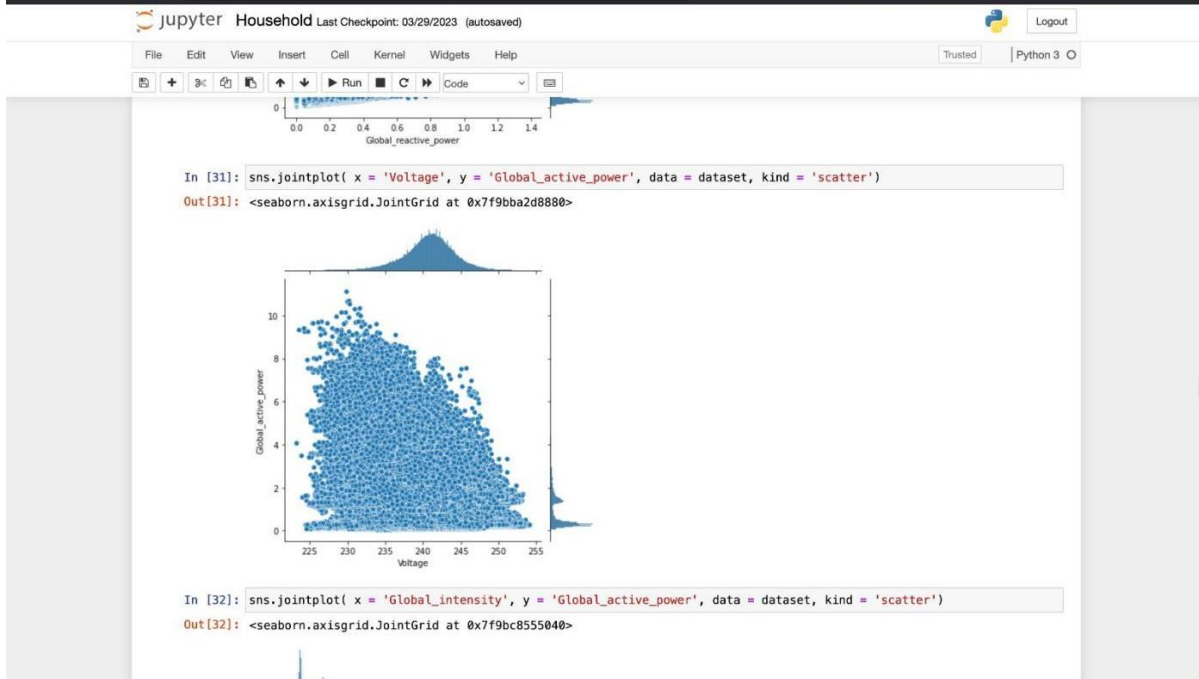
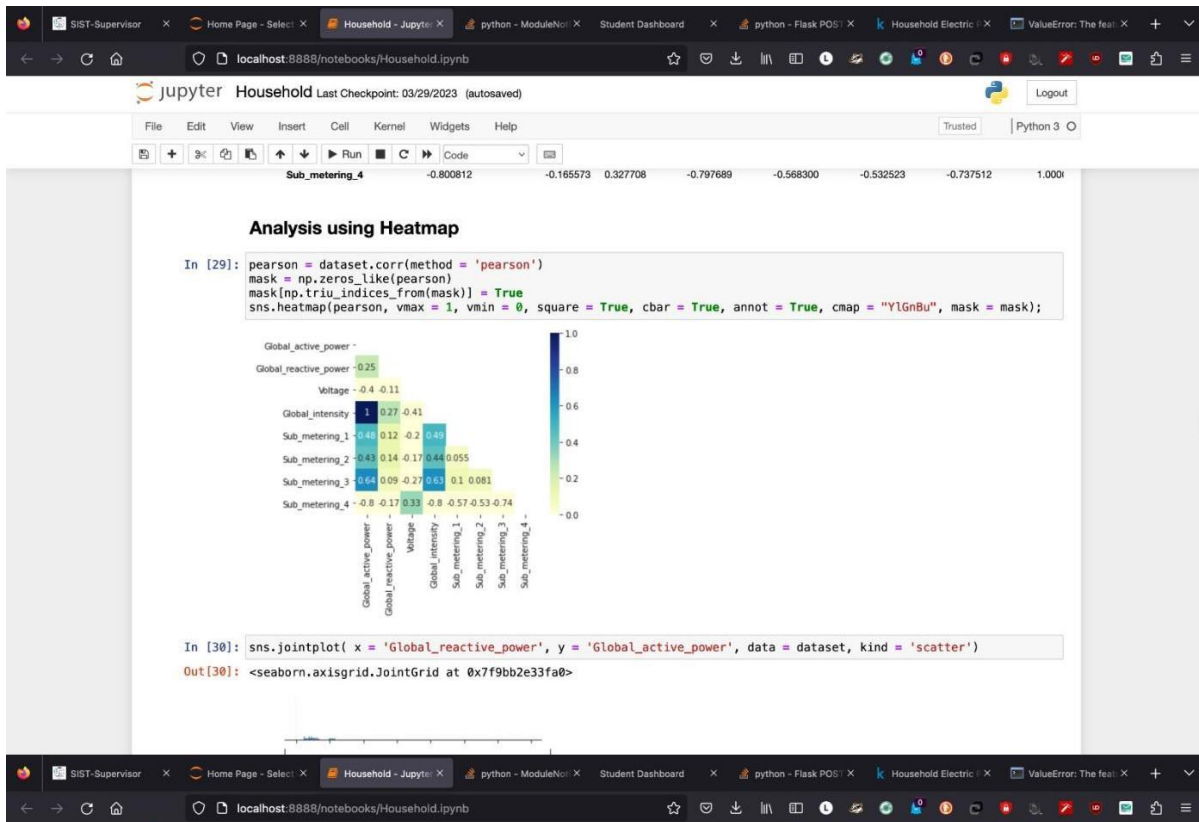
In [24]: sns.distplot(dataset['Global_active_power'], kde=False, bins=30)
/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2557: FutureWarning: 'distplot' is a deprecated function and will be removed in a future version. Please adapt your code to use either 'displot' (a figure-level function with similar flexibility) or 'histplot' (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)

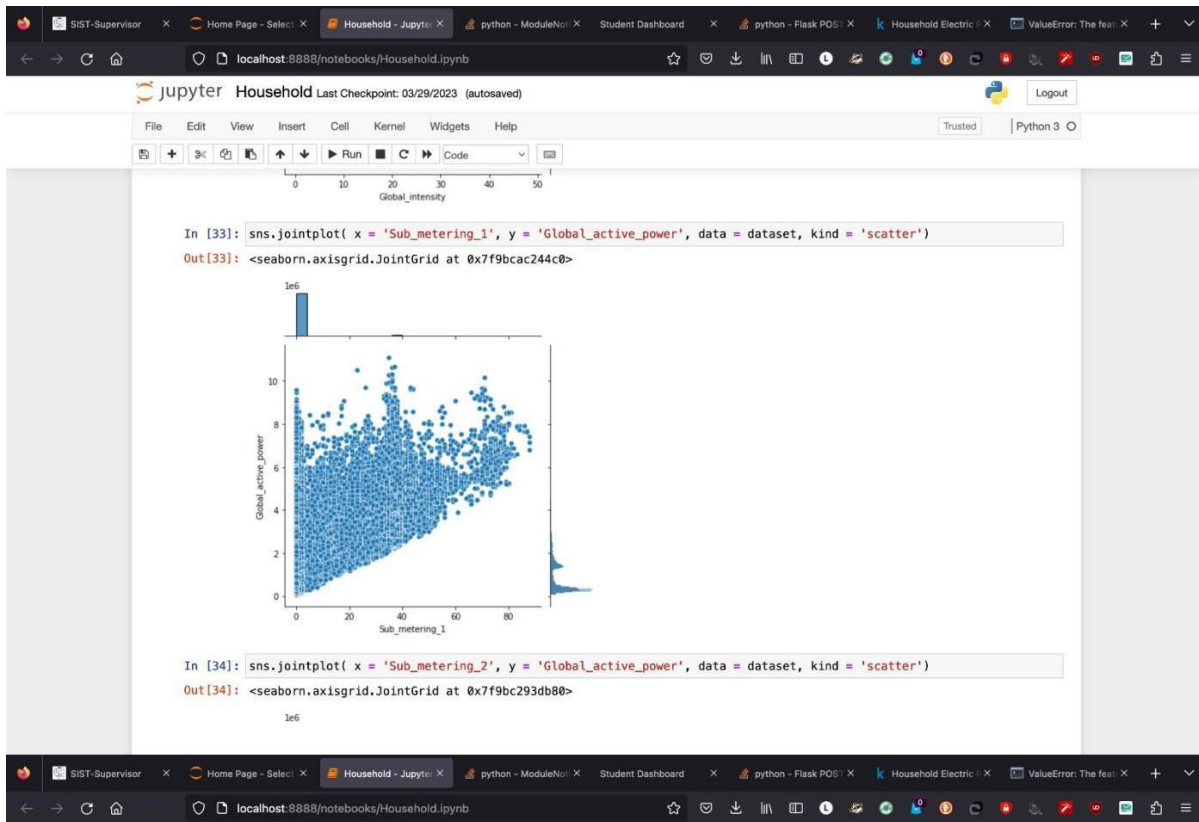
Out[24]: <AxesSubplot: xlabel='Global_active_power'>

```









Divide the model into Train and Test data.

Dependent and Independent Columns

In this activity, the dependent and independent variables are to be identified.

- 1) The independent variable in the dataset would be considered as 'x'.
- 2) The dependent variable in the dataset would be considered as 'y'.

```
In [37]: from sklearn.model_selection import train_test_split
```

```
In [38]: y = dataset['Global_active_power']
y.head()
```

```
Out[38]: datetime
2006-12-16 17:24:00    4.216
2006-12-16 17:25:00    5.360
2006-12-16 17:26:00    5.374
2006-12-16 17:27:00    5.388
2006-12-16 17:28:00    3.666
Name: Global_active_power, dtype: float64
```

```
In [39]: x = dataset.drop(columns = 'Global_active_power',axis = 1)
x.head()
```

```
Out[39]:
```

	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3	Sub_metering_4
datetime							
2006-12-16 17:24:00	0.418	234.84	18.4	0.0	1.0	17.0	-10.973333
2006-12-16 17:25:00	0.436	233.63	23.0	0.0	1.0	16.0	-8.066667
2006-12-16 17:26:00	0.498	233.29	23.0	0.0	2.0	17.0	-10.043333
2006-12-16 17:27:00	0.502	233.74	23.0	0.0	1.0	17.0	-9.020000

```
In [40]: x_train,x_test,y_train,y_test = train_test_split(x, y, test_size = 0.3, random_state = 101)
```

```
In [41]: print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1434496, 7)
(614784, 7)
(1434496,)
(614784,)
```

```
In [42]: x_train,x_test,y_train,y_test
```

```
Out[42]:
```

	Global_reactive_power	Voltage	Global_intensity	\
datetime				
2007-11-30 12:43:00	0.070	238.93		24.4
2010-01-10 08:12:00	0.092	244.32		1.2
2008-11-12 21:43:00	0.184	235.47		18.0
2009-03-04 04:23:00	0.000	244.36		1.2
2007-07-10 08:05:00	0.194	240.16		5.6
...	...	...		...
2008-03-22 12:38:00	0.000	243.80		2.2
2007-05-10 09:50:00	0.084	234.25		0.8
2009-11-15 14:25:00	0.206	242.93		1.8
2009-05-14 18:19:00	0.214	239.21		2.2
2009-04-18 16:30:00	0.000	243.29		1.2

	Sub_metering_1	Sub_metering_2	Sub_metering_3	\
datetime				
2007-11-30 12:43:00	0.0	74.0		18.0
2010-01-10 08:12:00	0.0	0.0		1.0
2008-11-12 21:43:00	37.0	0.0		17.0
2009-03-04 04:23:00	0.0	0.0		0.0

Household Last Checkpoint: 03/29/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Model Building

There are several Machine learning algorithms to be used depending on the data you are going to process such as images, sound, text, and numerical values. The algorithms can be chosen according to the objective. As per the dataset which we are using, it seems to be a regression problem, therefore you can use the following :

- Linear Regression
- Random Forest Regressor
- Decision Tree Regressor
- XGBoost Regressor
- and many more.

Training the Model

```
In [43]: import statsmodels.api as sm
In [44]: x_train_sm = sm.add_constant(x_train)
In [45]: lr = sm.OLS(y_train, x_train_sm).fit()
In [46]: lr.params
Out[46]: const -1.284459e-13
Global_reactive_power -2.168404e-14
Voltage 5.224770e-16
Global_intensity -1.908196e-17
Sub_metering_1 6.000000e-01
Sub_metering_2 6.000000e-01
Sub_metering_3 6.000000e-01
Sub_metering_4 6.000000e-01
dtype: float64
In [47]: lr.summary()
```

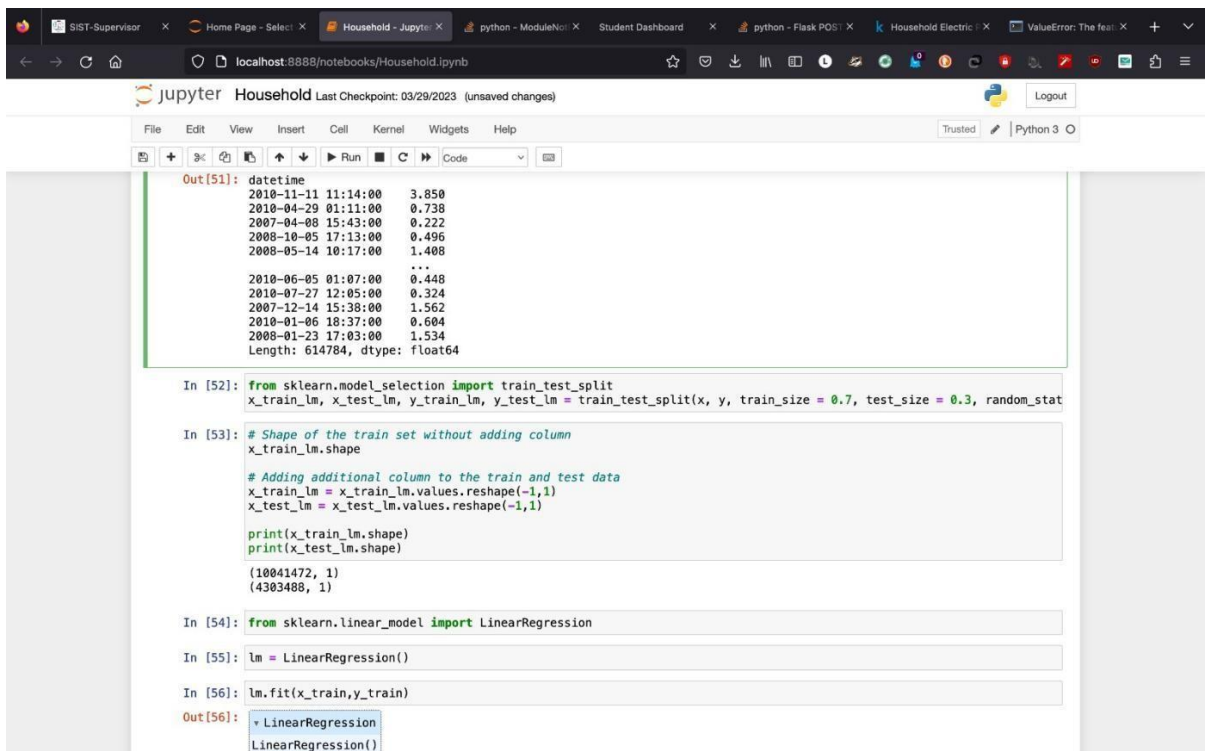
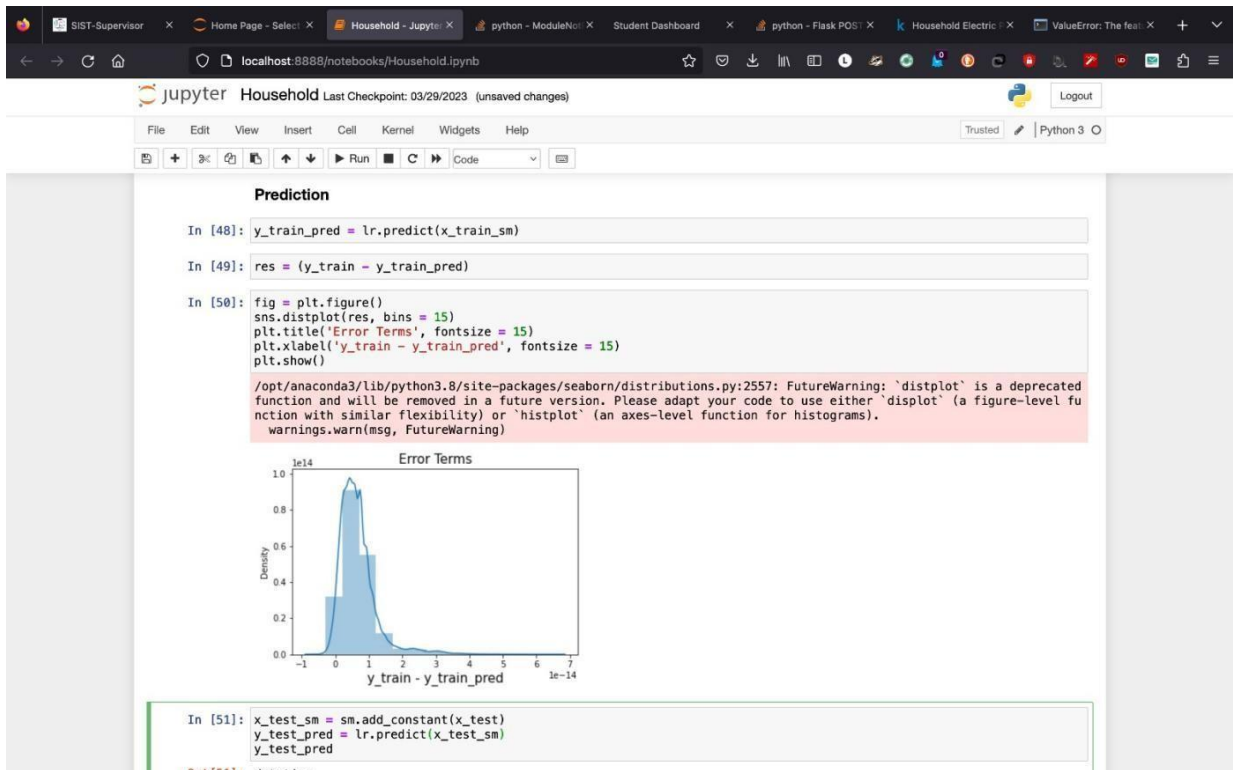
Household Last Checkpoint: 03/29/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [47]: lr.summary()
Out[47]:
```

OLS Regression Results

Dep. Variable:		Global_active_power		R-squared:		1.000	
Model:		OLS		Adj. R-squared:		1.000	
Method:		Least Squares		F-statistic:		3.084e+33	
Date:		Tue, 18 Apr 2023		Prob (F-statistic):		0.00	
Time:		08:50:51		Log-Likelihood:		4.4419e+07	
No. Observations:		1434496		AIC:		-8.884e+07	
Df Residuals:		1434488		BIC:		-8.884e+07	
Df Model:		7					
Covariance Type:		nonrobust					
	coef	std err	t	P> t	[0.025	0.975]	
const	-1.284e-13	6.21e-16	-206.869	0.000	-1.3e-13	-1.27e-13	
Global_reactive_power	-2.168e-14	7.36e-17	-294.597	0.000	-2.18e-14	-2.15e-14	
Voltage	5.225e-16	2.56e-18	203.809	0.000	5.17e-16	5.28e-16	
Global_intensity	-1.908e-17	4.23e-17	-0.451	0.652	-1.02e-16	6.39e-17	
Sub_metering_1	0.6000	1.06e-16	5.63e+15	0.000	0.600	0.600	
Sub_metering_2	0.6000	1.07e-16	5.63e+15	0.000	0.600	0.600	
Sub_metering_3	0.6000	1.06e-16	5.66e+15	0.000	0.600	0.600	
Sub_metering_4	0.6000	1.06e-16	5.64e+15	0.000	0.600	0.600	
Omnibus:	688543.088	Durbin-Watson:	0.826				
Prob(Omnibus):	0.000	Jarque-Bera (JB):	5169523.961				
Skew:	2.188	Prob(JB):	0.00				
Kurtosis:	11.206	Cond. No.	2.09e+04				



Household Last Checkpoint: 03/29/2023 (autosaved)

Check the metrics of the model

In [59]: `from sklearn import metrics`

Mean Absolute Error : MAE is a measure of errors between paired observations expressing the same phenomenon. Examples of Y versus X include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement. MAE is calculated as:

In [60]: `from sklearn.metrics import mean_absolute_error
score = mean_absolute_error(y_test, y_test_pred)
print("The Mean Absolute Error of our Model is {}".format(score, 2))`

The Mean Absolute Error of our Model is 6.674546271980803e-15

Mean Squared Error (MSE) : MSE or Mean Squared Error is one of the most preferred metrics for regression problems. It is simply the average of the squared difference between the target value and the value predicted by the regression model. As it squares the differences, it penalizes even a small error which leads to over-estimation of how bad the model is. It is preferred more than other metrics because it is differentiable and hence can be optimized better.

In [61]: `from sklearn.metrics import mean_squared_error
score = np.sqrt(mean_squared_error(y_test, y_test_pred))
print("The Mean Squared Error of our Model is {}".format(score, 2))`

The Mean Squared Error of our Model is 8.169789637426904e-08

Root Mean Square Error (RMSE) : RMSE is the square root of the averaged squared difference between the target value and the value predicted by the model. It is preferred more in some cases because the errors are first squared before averaging which poses a high penalty on large errors. This implies that RMSE is useful when large errors are undesired.

In [62]: `from sklearn.metrics import mean_squared_error
from math import sqrt
score = sqrt(mean_squared_error(y_test, y_test_pred))
print("The Root Mean Squared Error of our Model is {}".format(score, 2))`

The Root Mean Squared Error of our Model is 8.61383482876518e-15

Save The model

```
In [65]: import pickle
filename = 'PCASSS_model.pkl'
pickle.dump(lm, open(filename, 'wb'))
```

