

POWER CONSUMPTION ANALYSIS FOR HOUSE HOLDS USING IBM WATSON

A UG Project Phase – I report submitted to
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of
BACHELOR OF TECHNOLOGY

In
COMPUTER SCIENCE AND ENGINEERING

Submitted By

G. INDIRA

20UK5A0507

K. SAINATH

20UK5A0511

A. ASHWINI

19UK1A05N0

MD. MUSEEF ANWAR

18UK1A05F9

Under the guidance of

Ms. S. ANOOSHA

Assistant Professor



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VAAGDEVI ENGINEERING COLLEGE**

Affiliated to JNTUH, HYDERABAD
BOLLIKUNTA, WARANGAL (T.S) – 506005

2019-2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VAAGDEVI ENGINEERING COLLEGE
WARANGAL



CERTIFICATE

This is to certify that the UG phase-I Report entitled “**POWER CONSUMPTION ANALYSIS FOR HOUSE HOLDS USING IBM WATSON**” is being submitted by **G.INDIRA(H.NO:20UK5A0507),K.SAINATH(H.NO:20UK5A0511),A.ASHWINI(H.NO: 19UK1A05N0),MD.MUSEEF ANWAR(H.NO:18UK1A05F9)** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** to **Jawaharlal Nehru Technological University Hyderabad** during the academic year **2019-2023**.

Project Guide
Ms. S. ANOOSHA
(Assistant Professor)

Head of the Department
Mr. Dr. R. NaveenKumar
(Professor)

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. P. PRASAD RAO**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG Project Phase – I in the institute.

We extend our heartfelt thanks to **Dr. R. NAVEEN KUMAR**, Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG Project Phase – I.

We express heartfelt thanks to Smart Bridge Educational Services Private Limited, for **their** constant supervision as well as for providing necessary information regarding the mini project and for their support in completing the mini project.

We express heartfelt thanks to the guide, **Ms. S. ANOOSHA**, Assistant Professor, Department of CSE for her constant support and giving necessary guidance for completion of this mini project.

Finally, we express our sincere thanks and gratitude to our family members, friends for their encouragement and outpouring their knowledge and experiencing throughout thesis.

TEAM MEMBERS:

G.INDIRA	(20UK5A0507)
K.SAINATH	(20UK5A0511)
A.ASHWINI	(19UK1A05N0)
MD. MUSEEF ANWAR	(18UK1A05F9)

ABSTRACT

rational consumption of electricity at home becomes of a great importance. The first step In this paper, we analyze the average usage and priority of home appliances based on the characteristics of power data in household. In the case of electric power used in household, the control ability of individual appliance is determined according to the characteristics, and the analysis. Depending on the result of this kind of analysis, it is possible to suggest more efficient energy operation from the viewpoint of the energy consumer. Therefore, by analyzing the characteristics of each power source based on the time horizon, we set the priority of individual power resources according to the user preference. Through the application of this method, it is proposed to minimize the inconvenience of users' power operation and power operation fee. As a result, peak power was reduced by 15%, and the total power operation fee was reduced by 4%. Nowadays practically all European countries while providing energy effective policy are concerned about reducing the total demand of energy consumption with maintaining the high level of development. Since the sector of individual consumption is one of the largest consumers of electric energy, the of rational consumption of electric energy is analyzing the level of electricity demand in order to predict future demand of electricity with a high degree of accuracy. Forecasting of the future demand can help making decisions for improving energy efficiency at home, choosing the right class of the household equipment. Forecasting is also significant instrument for effective and rational planning of the budget.

KEYWORDS: Power consumption; Household electric; Linear model; Neural network model.

TABLE OF CONTENTS

LIST OF CHAPTERS	PAGE NUMBER
1.INTRODUCTION	
1.1 OVERVIEW.....	7
1.2 PURPOSE.....	8
2.EXPERIMENTAL ANALYSIS	
2.1 DATASET.....	9
3.THEORETICAL ANALYSIS	
3.1 BLOCK DIAGRAM.....	10
3.2 HARDWARE AND SOFTWARE REQUIREMENTS..	11
4.PROJECT FLOW	
4.1 PROJECT STRUCTURE.....	14
5.EXPERIMENTAL INVESTIGATION	
5.1 ANALYSIS OF THE PROJECT.....	15
5.1.1 COLLECTION OF DATASET.....	15
5.1.2 IMPORTING LIBRARIES READ THE DATASET..	16
6.PREPROCESS THE DATA	
6.1 HANDLING MISSING VALUES.....	19
6.1.1 REMOVING NULL VALUES.....	20
6.2 DATA VISUALIZATION.....	21
6.2.1 PLOTTING INDIVIDUAL VARIABLES.....	21

6.3 DIVIDE THE	8. MODEL INTO TRAIN AND TEST DATA.	31
6.3.1 SPLITTING THE DATA INTO TRAIN AND TEST.....		32
7.MODEL BUILDING		
7.1 TRAINING THE MODEL.....		33
7.1.1 NOTE THE PREDICTION OUTPUT.....		34
7.2 CHECK THE METRICS OF THE MODEL.....		34
7.2.1 REGRESSION EVALUATION METRICS.....		34
7.3 SAVE THE MODEL.....		35
8.APPLICATION BULIDING		
8.1 BUILD HTML CODE.....		36
8.2 BUILD PYTHON CODE.....		36
9.RUN THE APP.....		38
10.RESULT.....		39
11.APPLICATIONS.....		41
12.CONCLUSION.....		42
13.REFERENCES.....		44

1.INTRODUCTION

1.1. OVERVIEW

Electricity sector in India. India is the world's third largest producer and third largest consumer of electricity. The gross electricity consumption in 2018-19 was 1,181 kWh per capita. Energy use can be viewed as a function of total GDP, structure of the economy and technology. The increase in household energy consumption is more significant than that in the industrial sector. To achieve reduction in electricity consumption, it is vital to have current information about household electricity use. This Guided Project mainly focuses on applying a machine-learning algorithm to calculate the power consumed by all appliances. This will help you track the power consumed on regular intervals for all kinds of appliances which use heavy loads such as Air Conditioners, Oven or a washing machine etc.

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. 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 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 .

1.2 PURPOSE

In this project, Electricity is an essential part of modern life . People use electricity for lighting, heating, cooling, and refrigeration and for operating appliances, computers, electronics, machinery, and public transportation systems.

This Guided Project mainly focuses on applying a machine-learning algorithm to calculate the power consumed by all appliances. This will help you track the power consumed on regular intervals for all kinds of appliances which use heavy loads such as Air Conditioners, Oven etc.

2. EXPERIMENTAL ANALYSIS

This section introduces the dataset and framework, and reports experimental results.

2.1 Dataset

This experiment uses the individual household electric power consumption dataset, which contains 2075259 pieces of data. The power consumption data of a household for 4 years are recorded in the dataset, and sub metering 1 , sub metering 2 and sub metering 3 to represent the power consumption of different appliances.

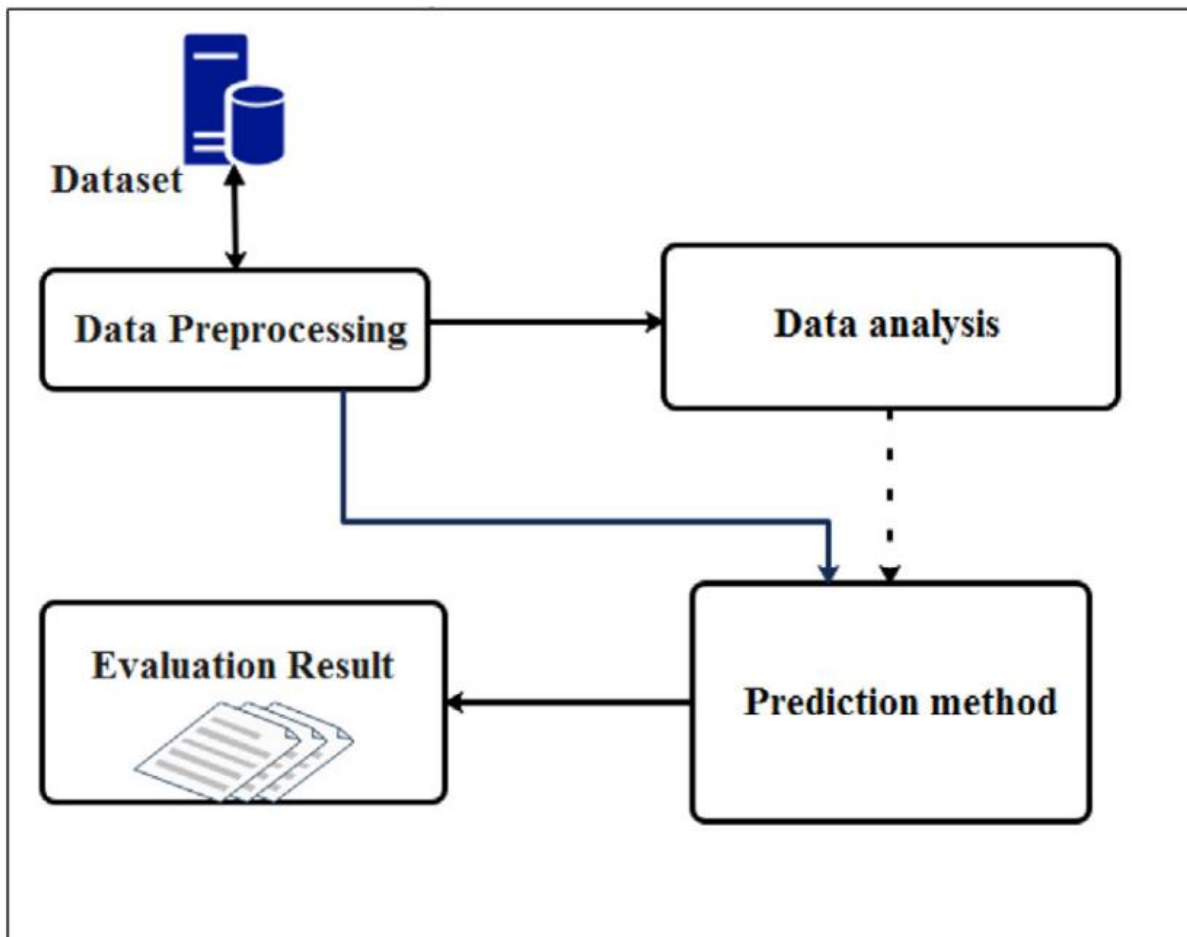


Fig.1:General framework.

3.THEORETICAL ANALYSIS

3.1 Block Diagram

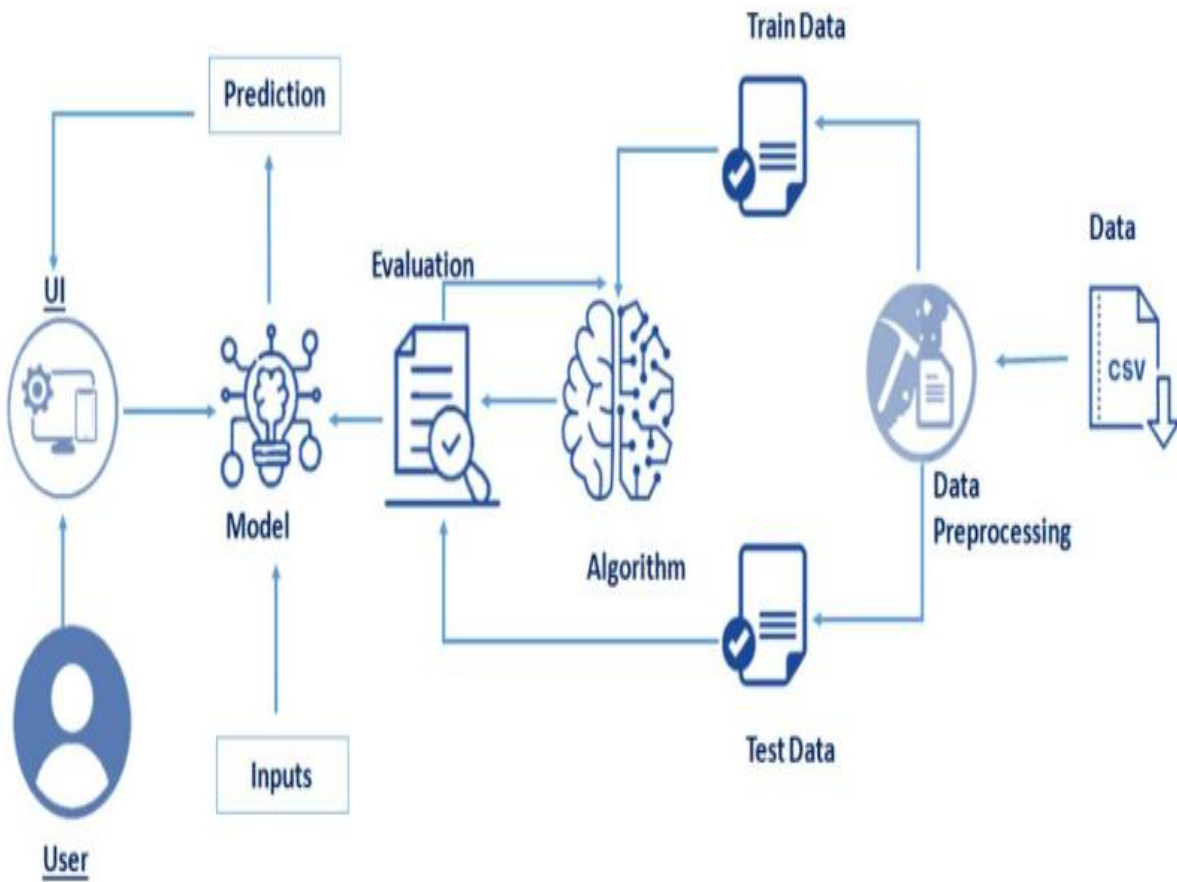


Fig 2: Block Diagram

3.2 HARDWARE AND SOFTWARE DESIGNING REQUIREMENTS OF THE PROJECT

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and mac OS Conda is an open-source, cross-platform, package management system. Anaconda comes with great tools like **Jupyter Lab**, **Jupyter Notebook**, Qt Console, Spyder, Orange, R studio, Visual Studio Code.

For this project, we will be using **Jupyter** notebook and **Spyder**

The artefact created utilizes the Python programming language, specifically version 3.9.2, and has its functionality extended with the assistance of 8 packages (libraries). While the latest Python version is recommended, it is possible to run the artefact with a minimum of version 3.7. However, it is critical to ensure that the libraries installed utilise the same versions described in this section. The 8 libraries used include Flask, NumPy, gevent, Tensor flow, Werkzeug, Pandas, Scikit learn, and Scikit-plot.

Another core piece of software used for creating the artefact is virtual environments. When developing software, it is common for developers to have packages and libraries already set up with the latest package versions. Unfortunately, it can be time consuming and troublesome to switch between package versions when working on multiple projects. However, virtual

environments help to mitigate this issue by preventing conflicting package versions. Anaconda provides a platform to manage Python environments and enables accessibility through a Python kernel, integrated seamlessly with Jupyter Notebooks.

Name	Description	Version
Flask	Flask is a micro-framework in Python which is extensively used to deploy ML models on the web	2.0.2
NumPy	A package used for scientific computing, providing multidimensional array objects and the ability to perform mathematical computations on them.	1.18.5
Gevent	Gevent 1.3 is an important update for performance, debugging and monitoring, and platform support. It introduces an (optional) <u>libuv</u> loop implementation and supports PyPy on Windows.	21.8.0
Tensorflow	TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.	2.3.0
Werkzeug	Werkzeug is a comprehensive <u>WSGI</u> web application library. It began as a simple collection of various utilities for WSGI applications and has become one of the most advanced WSGI utility libraries.	2.0.2
Pandas	A package dedicated to data analysis that uses dataframe objects to view and manipulate data within a tabular format. Required for presenting the model results in a tabular format.	1.2.2
Scikitlearn	A package built on top of NumPy, SciPy, and Matplotlib that provides functionality for efficiently creating ML models and calculating their performance. Required for creating confusion matrices.	0.2.4
Scikit-plot	An extension of Scikit-learn specific to plotting performance metrics. Required for creating the confusion matrix and ROC curve plots.	0.3.7

Table 1: Software required in the project

CNNs can take a long time to train and tune when only using a CPU. To increase this, NVIDIA has a parallel computing toolkit that allows the transfer of data onto GPUs, called CUDA. Fortunately, PY torch has CUDA functionality built into its library that automatically keeps track of the available GPUs on a system and the tensors allocated to them. When experimenting with different CUDA versions, CUDA toolkit version 10.1 proved fundamental and is required to run the artefact within this project to minimize the training, tuning a computation speed.

The system hardware used to create the artefact is as follows:

- Intel Core i7-4790k CPU 4.00GHz**
- 16GB of RAM**
- NVIDIA GeForce GTX 970 (4GB) GPU**
- Windows 11 Operating System**

It is recommended, but not required, to have similar hardware specifications to run the artefact to maximize its efficiency. If a GPU is not available, it is possible to run the artefact on a typical CPU. However, the tuning time taken to complete the 36 model variants on a GPU took approximately two days. Using a CPU, the time taken will increase by a minimum of 4x as long.

4. Project Flow

- Download the dataset.
- Pre process or clean the data.
- Analyse the pre-processed data.
- Train the machine with pre processed data using an appropriate machine learning algorithm.
- Save the model and its dependencies.
- Build a Web application using flask that integrates with the model built.

4.1 Project Structure

Create a Project folder which contains files as shown below

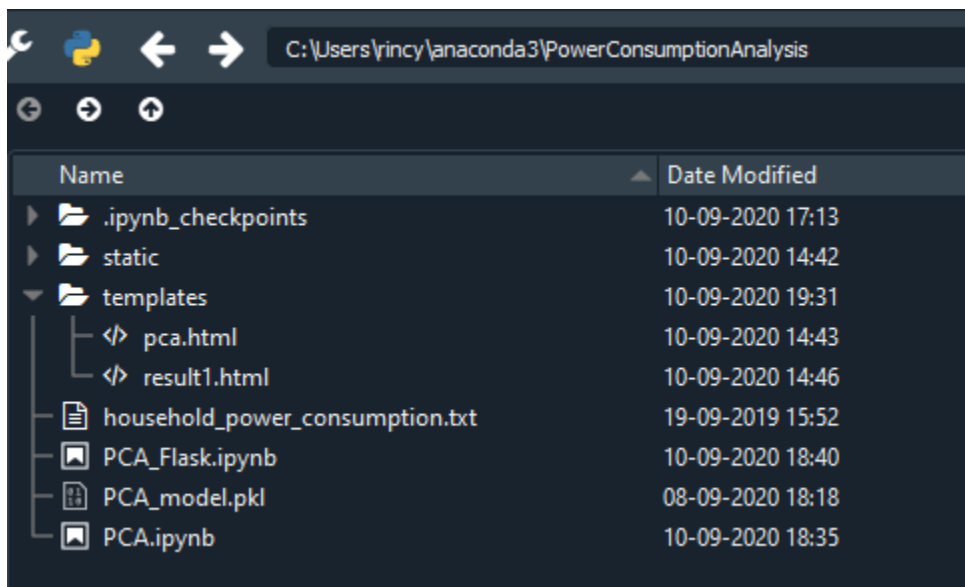


Fig.4 project structure

- household_power_consumption.txt is the dataset used.
- PCA.ipynb is the python file where algorithm is applied to dataset for testing and training. Finally, the model is saved for future use.
- PCA_model.pkl is the saved model used in flask to predict the output instantly for the given inputs.
- To build a Flask Application, save HTML pages in the templates folder and a python script PCA_Flask.ipynb for server-side scripting.

5. EXPIREMENTAL INVESTIGATION

5.1 Analysis of the project

5.1.1 Collection Of Dataset

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.

	Global_act	Global_rex	Voltage	Global_int	Sub_meter1	Sub_meter2	Sub_meter3	Sub_metering_4
1	4.216	0.418	234.84	18.4	0	1	17	52.26667
2	5.36	0.436	233.63	23	0	1	16	72.33334
3	5.374	0.498	233.29	23	0	2	17	70.56667
4	5.388	0.502	233.74	23	0	1	17	71.8
5	3.666	0.528	235.68	15.8	0	1	17	43.1
6	3.52	0.522	235.02	15	0	2	17	39.66667
7	3.702	0.52	235.09	15.8	0	1	17	43.7
8	3.7	0.52	235.22	15.8	0	1	17	43.66667
9	3.668	0.51	233.99	15.8	0	1	17	43.13334
10	3.662	0.51	233.86	15.8	0	2	16	43.03333
11	4.448	0.498	232.86	19.6	0	1	17	56.13333
12	5.412	0.47	232.78	23.2	0	1	17	72.2
13	5.224	0.478	232.99	22.4	0	1	16	70.06667
14	5.268	0.398	232.91	22.6	0	2	17	68.8
15	4.054	0.422	235.24	17.6	0	1	17	49.56667
16	3.384	0.282	237.14	14.2	0	0	17	39.4
17	3.27	0.152	236.73	13.8	0	0	17	37.5
18	3.43	0.156	237.06	14.4	0	0	17	40.16667
19	3.266	0	237.13	13.8	0	0	18	36.43333
20	3.728	0	235.84	16.4	0	0	17	45.13334
21	5.894	0	232.69	25.4	0	0	16	82.23333
22	7.706	0	230.98	33.2	0	0	17	111.4333
23	7.026	0	232.21	30.6	0	0	16	101.1
24	5.174	0	234.19	22	0	0	17	69.23333
25	4.474	0	234.96	19.4	0	0	17	57.56667
26	3.248	0	236.66	13.6	0	0	17	37.13334
27	3.236	0	235.84	13.6	0	0	17	36.93333

Fig.5 :Dataset

5.1.2 Importing the Libraries And Read The Dataset

The libraries can be imported using the import keyword

```
Importing necessary libraries

In [1]: import numpy as np # Linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
```

Fig.6: importing libraries

Numpy and Pandas : Open source data analysis and manipulation tool, built on top of the Python programming language.

Matplotlib and Seaborn : Used for visualization with python.

```
Importing dataset

In [2]: dataset = pd.read_csv("C:/Users/rincy/OneDrive/Desktop/PowerConsumptionAnalysis
    /household_power_consumption.txt"
    , sep=';', header=0, infer_datetime_format=True,
    parse_dates={'datetime':[0,1]}, index_col=['datetime'])

C:\Users\rincy\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3063: DtypeWarning: Columns (2,3,4,5,6,7) have mixed types.Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Fig.7: importing dataset

Here, we are reading the dataset(“.txt” file) from the system using pandas (pd is the alias name given to pandas package) and storing it in a variable ‘dataset’ head() method is used to return top n (5 by default) rows of a DataFrame or series.

Understand the dataset							
In [3]:	dataset.head()						
	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_mete
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

Fig.8:understand the dataset

tail() method is used to return last n (5 by default) rows of a Data Frame or series..

In [4]:	dataset.tail()						
	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_mete
datetime							
2010-11-26 20:58:00	0.946	0	240.43	4	0	0	0.0
2010-11-26 20:59:00	0.944	0	240	4	0	0	0.0
2010-11-26 21:00:00	0.938	0	239.82	3.8	0	0	0.0
2010-11-26 21:01:00	0.934	0	239.7	3.8	0	0	0.0
2010-11-26 21:02:00	0.932	0	239.55	3.8	0	0	0.0

Fig.9 :row of series

Displaying the number of rows and columns of dataset.

```
In [5]: print(f"The Dataset has {dataset.shape[0]} rows and {dataset.shape[1]} columns")

The Dataset has 2075259 rows and 7 columns
```

Fig.10: Displaying rows and columns

Displaying column names of dataset.

```
In [6]: dataset.columns

Index(['Global_active_power', 'Global_reactive_power', 'Voltage',
      'Global_intensity', 'Sub_metering_1', 'Sub_metering_2',
      'Sub_metering_3'],
      dtype='object')
```

Fig.11: Displaying columns

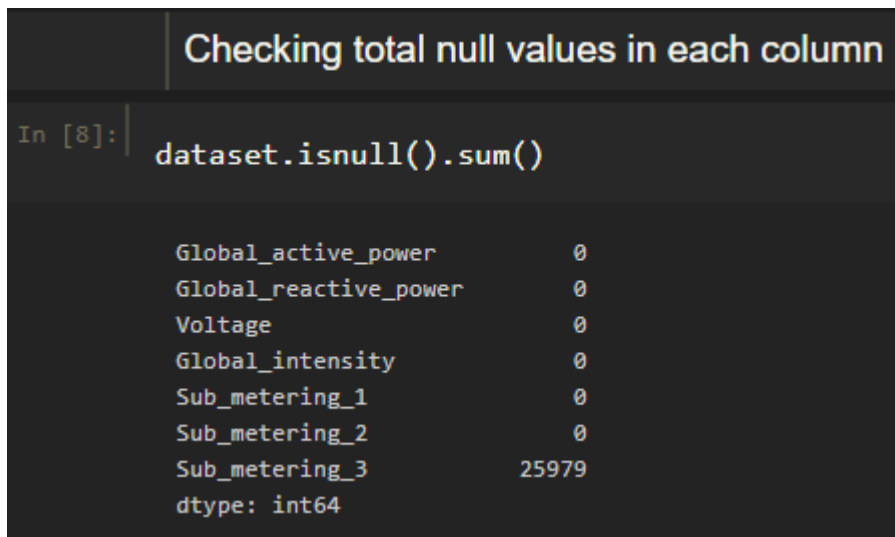
6. Preprocess The Data

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.

6.1 Handling The Missing Values

Now that we read the dataset, let's find if there are any missing values. `isnull()` checks if there are null values. `sum()` prints the sum of values. `isnull().sum()` displays total number of null values in each column.



```
Checking total null values in each column

In [8]: dataset.isnull().sum()

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
```

Fig.12: Checking Null values

6.1.1 Removing Null Values

Displaying percentage of data missing

```
Understanding percent of data missing

In [9]: percent_missing = dataset.isnull().sum() * 100 / len(dataset)
        missing_value_df = pd.DataFrame({'percent_missing': percent_missing})

In [10]: missing_value_df
```

	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	1.251844

Fig.13: Displaying percentage of data missing

Displaying first five null values of sub metering 3.

`replace(x,y)` is used to replace `x` with `y`. Here, we are replacing '?' with 'NaN' values.

```
Handling missing values

In [13]: dataset.loc[dataset.Sub_metering_3.isnull()].head()
```

	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_mete
datetime							
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 [14]: dataset.replace('?', np.nan, inplace=True)
```

Fig .14: Handling missing values

Displaying null values after replacing '?' with 'NaN'.

`dropna(how='all')` is used to drop all null values. Now, there are no null values in the dataset.

```
In [15]: dataset.loc[dataset.Sub_metering_3.isnull()].head()
```

	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_mete
datetime							
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 [16]: dataset = dataset.dropna(how = 'all')
```

Fig.15:Dropping null values

`dataset[i].astype('float64')` converts all type of data to float.

`shape()` is used to display number of rows and columns.

```
In [17]: for i in dataset.columns:
          dataset[i] = dataset[i].astype('float64')
          #dataset = dataset.astype('float32')
```

```
In [18]: dataset.shape
```

(2049280, 7)

Fig.16: Displaying rows and columns

Adding sub_metering_4 value to the dataset. We created another submeter value by performing operations on Global_active_power, Sub_metering_1, Sub_metering_2, Sub_metering_3 columns.

```

Adding another sub_metering_4 column

In [19]: values = dataset.values
         dataset['sub_metering_4'] = (values[:,0] * 1000 / 60) - (values[:,4] + values[:,5] + values[:,6])

In [20]: dataset.dtypes

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

```

Fig.17: Adding another column

describe() functions are used to compute values like count, mean, standard deviation and IQR(Inter Quartile Ranges) and give a summary of numeric type data.

```

In [21]: dataset.describe()

```

	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3
count	2.049280e+06	2.049280e+06	2.049280e+06	2.049280e+06	2.049280e+06	2.049280e+06	2.049280e+06
mean	1.091615e+00	1.237145e-01	2.408399e+02	4.627759e+00	1.121923e+00	1.298520e+00	6.458447e-01
std	1.057294e+00	1.127220e-01	3.239987e+00	4.444396e+00	6.153031e+00	5.822026e+00	8.437154e-01
min	7.600000e-02	0.000000e+00	2.232000e+02	2.000000e-01	0.000000e+00	0.000000e+00	0.000000e+00
25%	3.080000e-01	4.800000e-02	2.389900e+02	1.400000e+00	0.000000e+00	0.000000e+00	0.000000e+00
50%	6.020000e-01	1.000000e-01	2.410100e+02	2.600000e+00	0.000000e+00	0.000000e+00	1.000000e-01
75%	1.528000e+00	1.940000e-01	2.428900e+02	6.400000e+00	0.000000e+00	1.000000e+00	1.700000e-01
max	1.112200e+01	1.390000e+00	2.541500e+02	4.840000e+01	8.800000e+01	8.000000e+01	3.100000e-01

Fig.18: Adding describe()function

6.2 Data Visualization

Data visualization is where a given data set is presented in a graphical format. It helps the detection of patterns, trends and correlations that might go undetected in text-based data. Understanding your data and the relationship present within it is just as important as any algorithm used to train your machine learning model. In fact, even the most sophisticated machine learning models will perform poorly on data that was not visualized and understood properly. Libraries like matplotlib, seaborn, pyplot etc.,

Let's visualize our data using matplotlib and seaborn library.

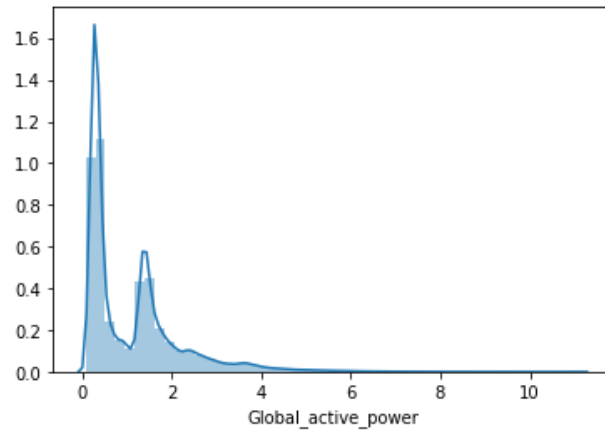
6.2.1 Plotting Individual variables

Seaborn distplot lets you show a histogram with a line on it. This can be shown in all kinds of variations. A distplot plots a univariate distribution of observations.

Data Visualization

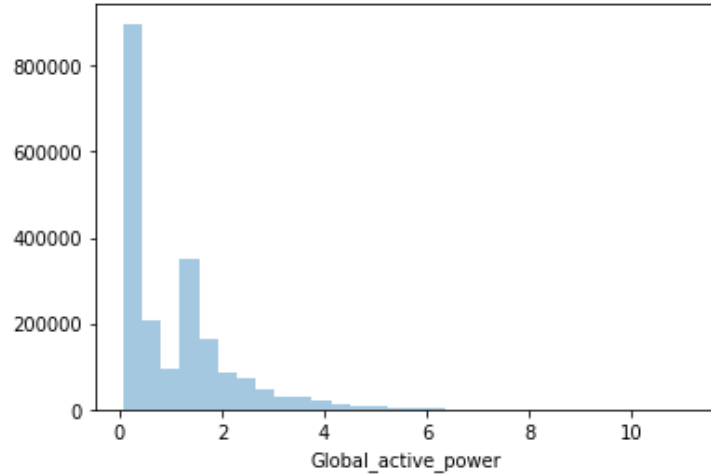
```
In [22]: sns.distplot(dataset['Global_active_power'])
```

```
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x236ddd3cd88>
```



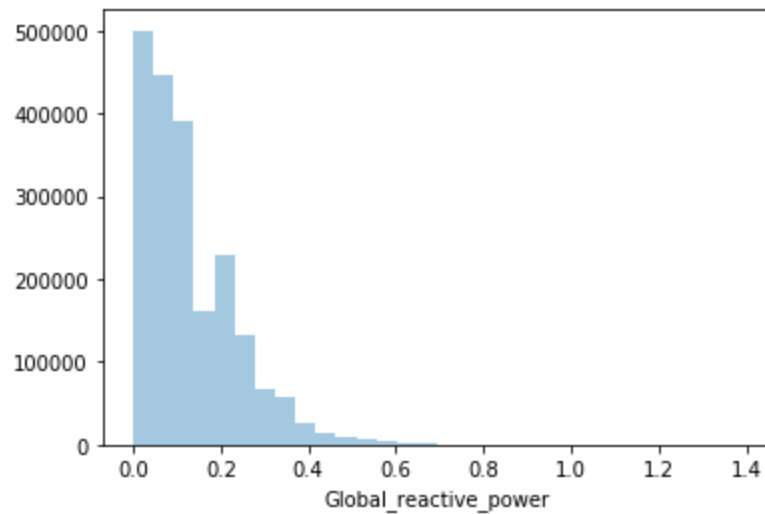
```
In [23]: sns.distplot(dataset['Global_active_power'],kde=False,bins=30)
```

```
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x236dd9e2448>
```



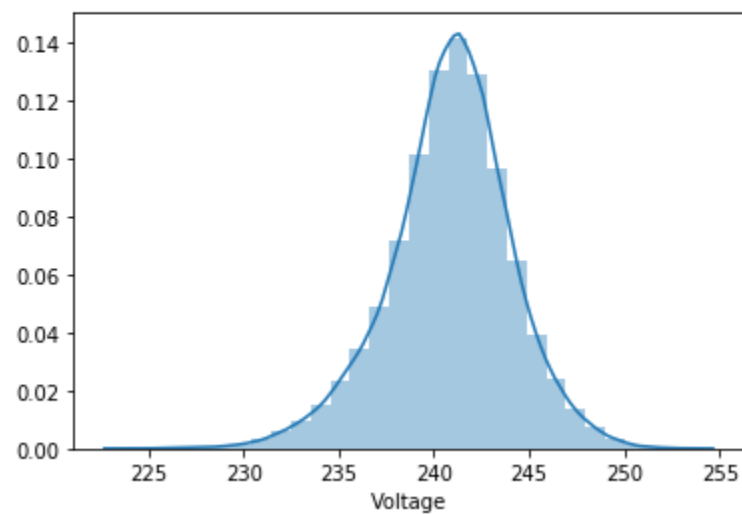
```
In [24]: sns.distplot(dataset['Global_reactive_power'],kde=False,bins=30)
```

```
Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x236df31fd48>
```



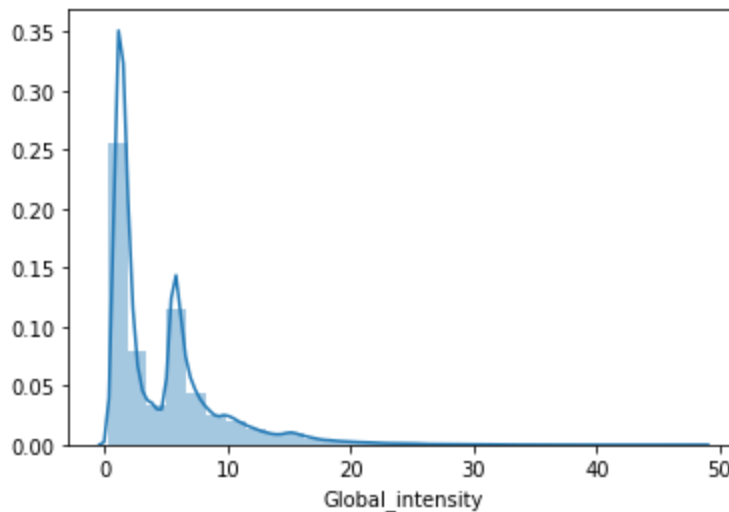
```
In [25]: sns.distplot(dataset['Voltage'],kde=True,bins=30)
```

```
Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x236df307688>
```



```
In [26]: sns.distplot(dataset['Global_intensity'],kde=True,bins=30)
```

```
Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x2368791aa88>
```



Corr() is used to find the pairwise correlation of all columns in the data frame. Any na values are automatically excluded. For any non-numeric data type columns in the data frame it is ignored.

Correlation of dataset values						
In [27]: dataset.corr()						
	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2
Global_active_power	1.000000	0.247017	-0.399762	0.998889	0.484401	0.434569
Global_reactive_power	0.247017	1.000000	-0.112246	0.266120	0.123111	0.139231
Voltage	-0.399762	-0.112246	1.000000	-0.411363	-0.195976	-0.167405
Global_intensity	0.998889	0.266120	-0.411363	1.000000	0.489298	0.440347
Sub_metering_1	0.484401	0.123111	-0.195976	0.489298	1.000000	0.054721
Sub_metering_2	0.434569	0.139231	-0.167405	0.440347	0.054721	1.000000
Sub_metering_3	0.638555	0.089617	-0.268172	0.626543	0.102571	0.080872
sub_metering_4	0.701380	0.211624	-0.271371	0.703258	0.125067	0.085201

Fig.19: Correlation of dataset values

A heat map (or heatmap) is a graphical representation of data where values are depicted by color. Heat maps make it easy to visualize complex data and understand it at a glance. Here lighter colour means that the columns are highly correlated with our target data.

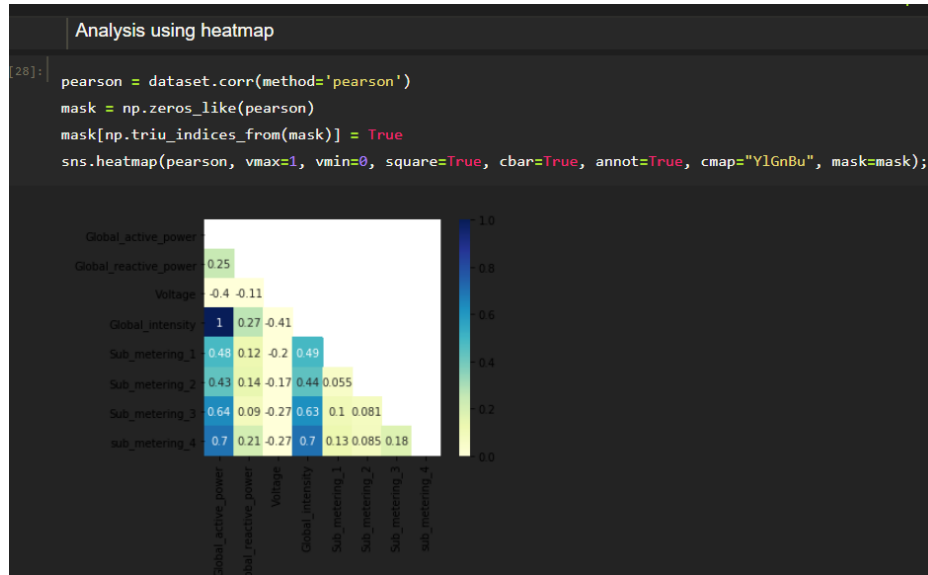
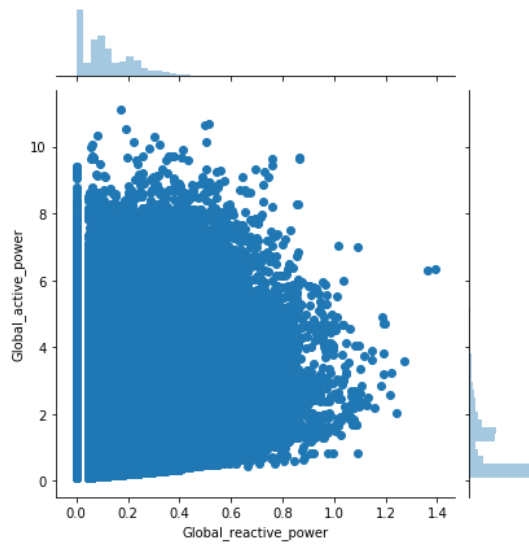


Fig.20: Analysing using heatmap

Seaborn jointplot lets you show a plot of two variables with bivariate and univariate graphs.

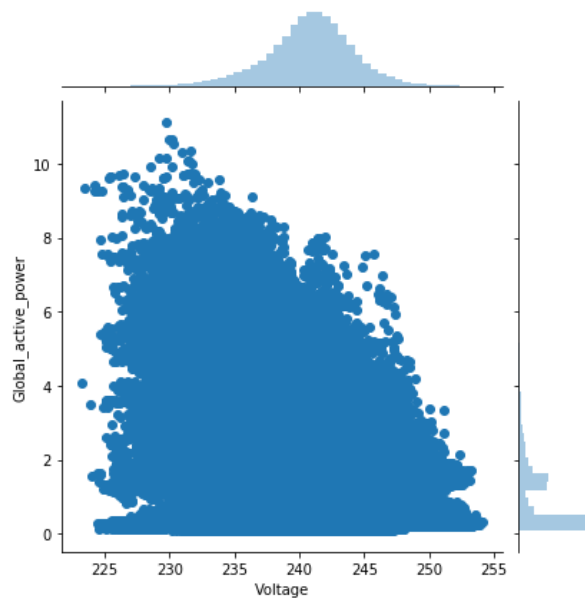
```
In [29]: sns.jointplot( x = 'Global_reactive_power' , y = 'Global_active_power' , data = dataset , kind = 'scatter')
```

```
Out[29]: <seaborn.axisgrid.JointGrid at 0x23687af36c8>
```



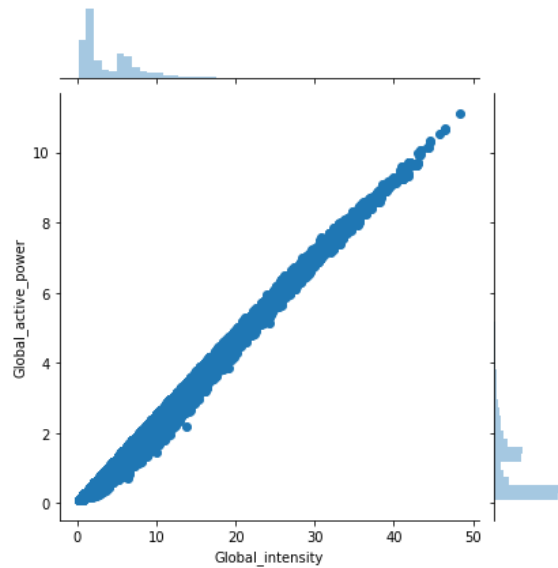
```
In [30]: sns.jointplot( x = 'Voltage' , y = 'Global_active_power' , data = dataset , kind = 'scatter')
```

```
Out[30]: <seaborn.axisgrid.JointGrid at 0x23687a75488>
```



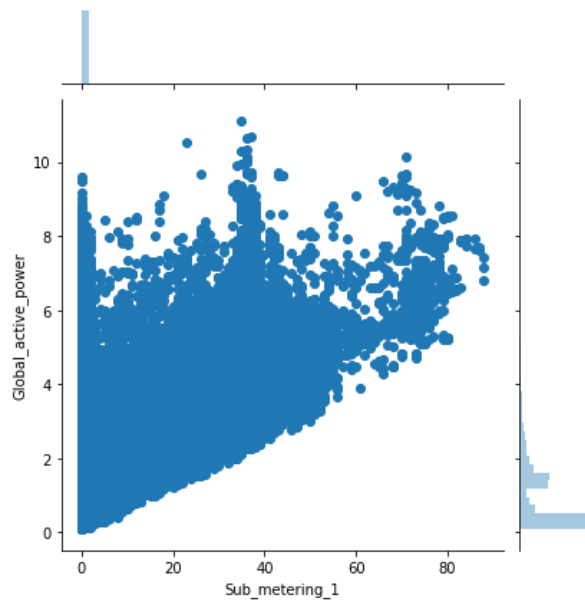
```
In [31]: sns.jointplot( x = 'Global_intensity' , y = 'Global_active_power' , data = dataset , kind = 'scatter')
```

```
Out[31]: <seaborn.axisgrid.JointGrid at 0x23687dfb848>
```



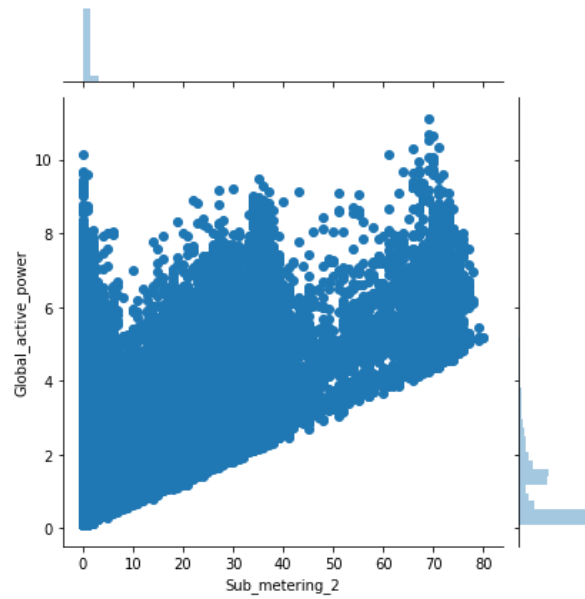
```
In [32]: sns.jointplot( x = 'Sub_metering_1' , y = 'Global_active_power' , data = dataset , kind = 'scatter')
```

```
Out[32]: <seaborn.axisgrid.JointGrid at 0x23688060548>
```



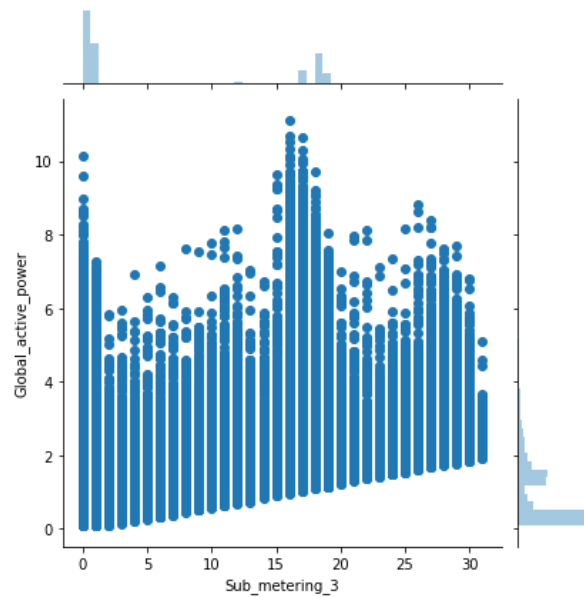
```
In [33]: sns.jointplot( x = 'Sub_metering_2' , y = 'Global_active_power' , data = dataset , kind = 'scatter')
```

```
Out[33]: <seaborn.axisgrid.JointGrid at 0x23688050748>
```



```
In [34]: sns.jointplot( x = 'Sub_metering_3' , y = 'Global_active_power' , data = dataset , kind = 'scatter')
```

```
Out[34]: <seaborn.axisgrid.JointGrid at 0x23689c180c8>
```



6.3 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'.

Now we will split the data of independent and dependent variables,

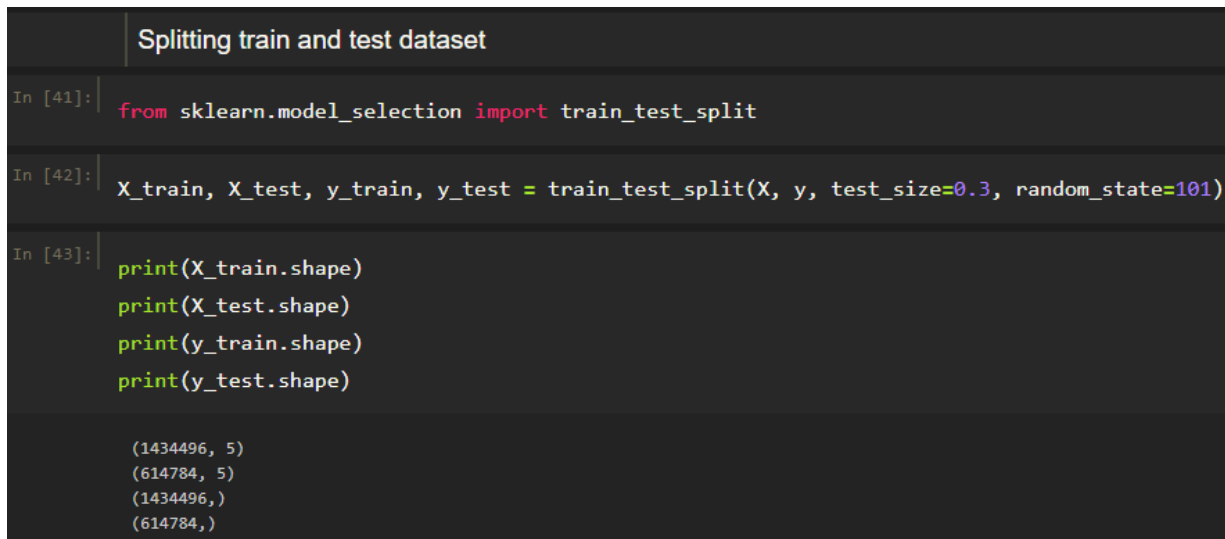
```
In [38]: y.head()

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
```

Fig.21: Split the data

6.3.1 Splitting the data into Train and Test

After identifying the dependent and independent variables, the dataset now has to be split into two sets, one set is used for training the model and the second set is used for testing how good the model is built. The split ratio we consider is 80% for training and 20% for testing.



```
Splitting train and test dataset

In [41]: from sklearn.model_selection import train_test_split

In [42]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)

In [43]: print(X_train.shape)
          print(X_test.shape)
          print(y_train.shape)
          print(y_test.shape)

          (1434496, 5)
          (614784, 5)
          (1434496,)
          (614784,)
```

Fig.22: splitting data into train and test

7. 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 :

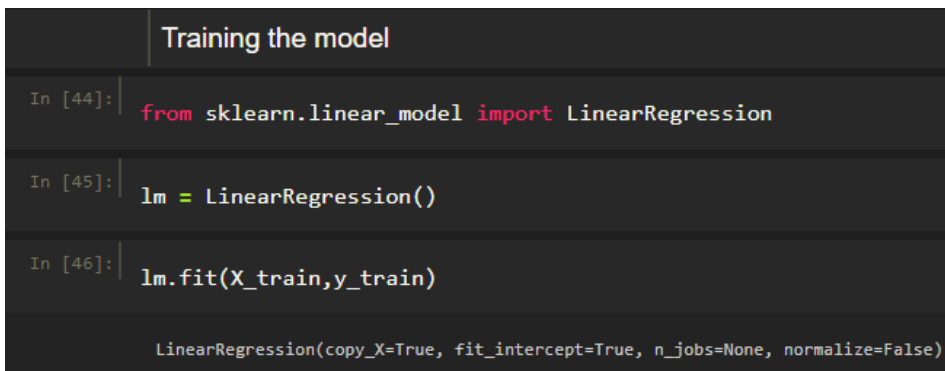
- Linear Regression
- Random Forest Regressor
- Decision Tree Regressor
- XGBoost Regressor

and many more.

7.1 Training The Model

Once after splitting the data into train and test, the data should be fed to an algorithm to build a model.

The machine learning algorithms used for this model here is Linear Regression. Linear regression is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). We are getting good accuracy using Linear Regression. We are using the algorithm from Scikit learn library to build the model as shown below :



```
Training the model

In [44]: from sklearn.linear_model import LinearRegression

In [45]: lm = LinearRegression()

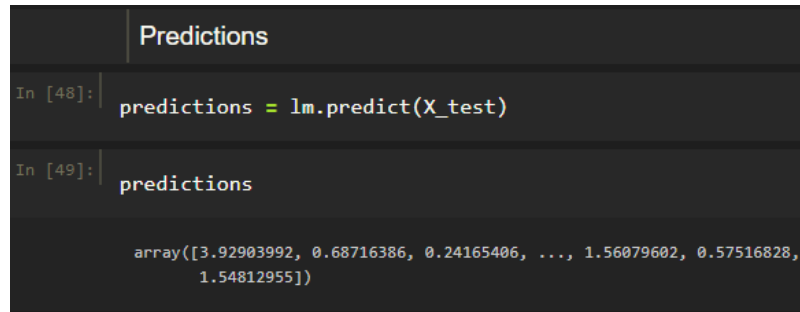
In [46]: lm.fit(X_train,y_train)

LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Fig.23: Training the Model

Once the model is trained, it's ready to make predictions. We can use the predict method on the model and pass x_test as a parameter to get the output as y_pred.

7.1.1 Note that the prediction output is an array of real numbers corresponding to the input array.



```
Predictions
In [48]: predictions = lm.predict(X_test)
In [49]: predictions

array([3.92903992, 0.68716386, 0.24165406, ..., 1.56079602, 0.57516828,
       1.54812955])
```

Fig.24: prediction

7.2 Check The Metrics Of The Model

After training the model, the model should be tested by using the test data which is been separated while splitting the data for checking the functionality of the model.

7.2.1 Regression Evaluation 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:

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.

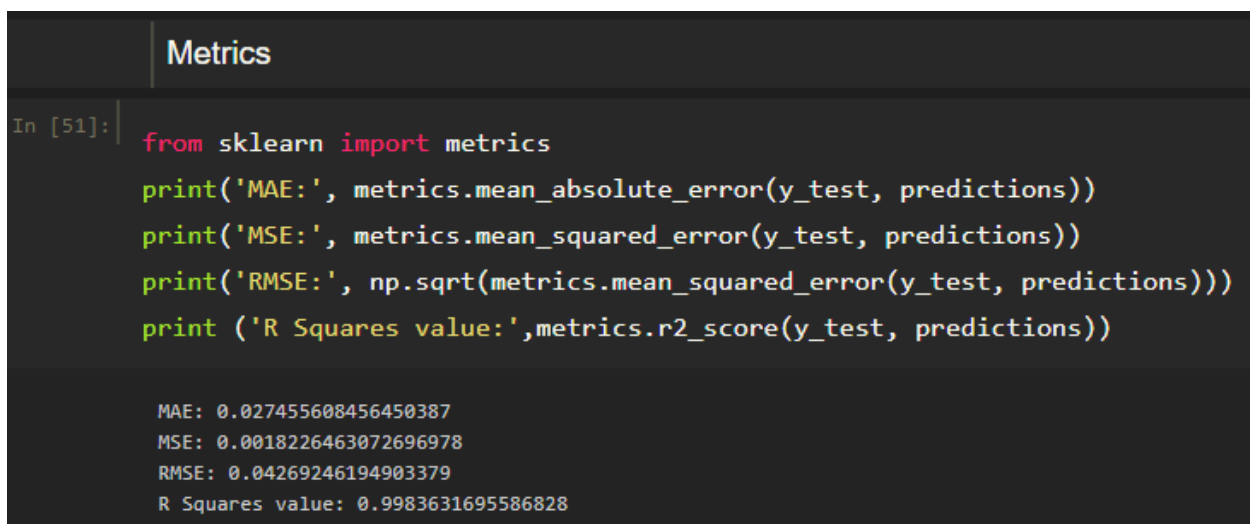
Root Mean Square Error (RMSE) : is the square root of the averaged squared difference between the target value and the value predicted by the model. It is preferred more in some cases because the errors are first squared before averaging which poses a high penalty on large errors. This implies that RMSE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

R² Score :

Coefficient of Determination or R² is another metric used for evaluating the performance of a regression model. The metric helps us to compare our current model with a constant baseline and tells us how much our model is better.

The constant baseline is chosen by taking the mean of the data and drawing a line at the mean. R² is a scale-free score that implies it doesn't matter whether the values are too large or too small, the R² will always be less than or equal to 1.

For testing the model we use the below method.



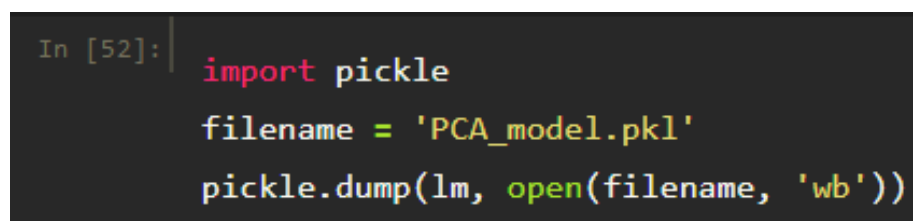
```
In [51]: from sklearn import metrics
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
print('R Squares value:', metrics.r2_score(y_test, predictions))

MAE: 0.027455608456450387
MSE: 0.0018226463072696978
RMSE: 0.04269246194903379
R Squares value: 0.9983631695586828
```

Fig.25: Metrics

7.3 Save The Model

The finalised model is now to be saved. We will be saving the model as a pickle or pkl file.



```
In [52]: import pickle
filename = 'PCA_model.pkl'
pickle.dump(lm, open(filename, 'wb'))
```

Fig.26: Saving the model

8. Application Building

After the model is built, we will be integrating it to a web application so that normal users can also use this trained model. In the application, the user provides the required values and a prediction for total consumed power is then shown on the screen.

8.1 Build HTML Code

To build this flask application you should have basic knowledge of “HTML, CSS, Bootstrap, flask framework and python”

We Build an HTML page to take the user values and save them into variables upon clicking of the “PREDICT” button. The output is to be then displayed on the page. The HTML pages are put under templates folder and any style sheets if present are put in the static folder.

HTML pages “pca.html” for our home page and “result1.html” which comes to use when we print out the final predictions made, both of these are stored in the templates folder .

Let’s see how our pca.html page looks like:

Let’s see how our output page looks like:

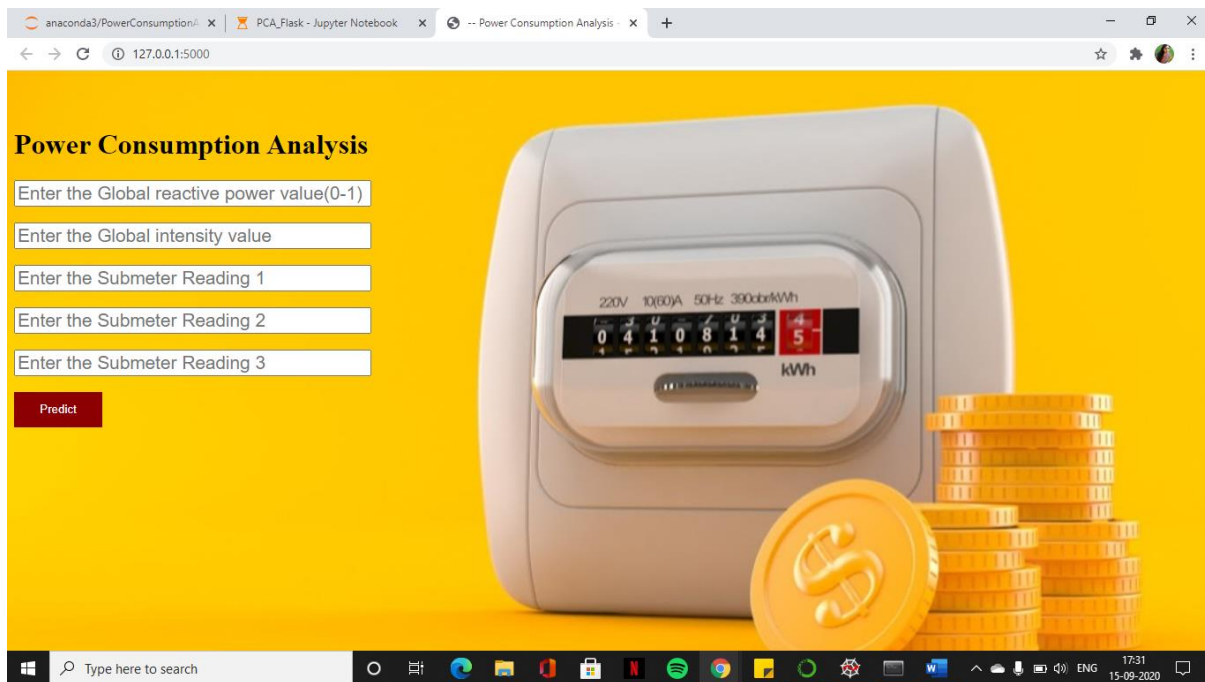


Fig.27: Output page

Enter the values and click on Predict button to view the result on “result1.html”.

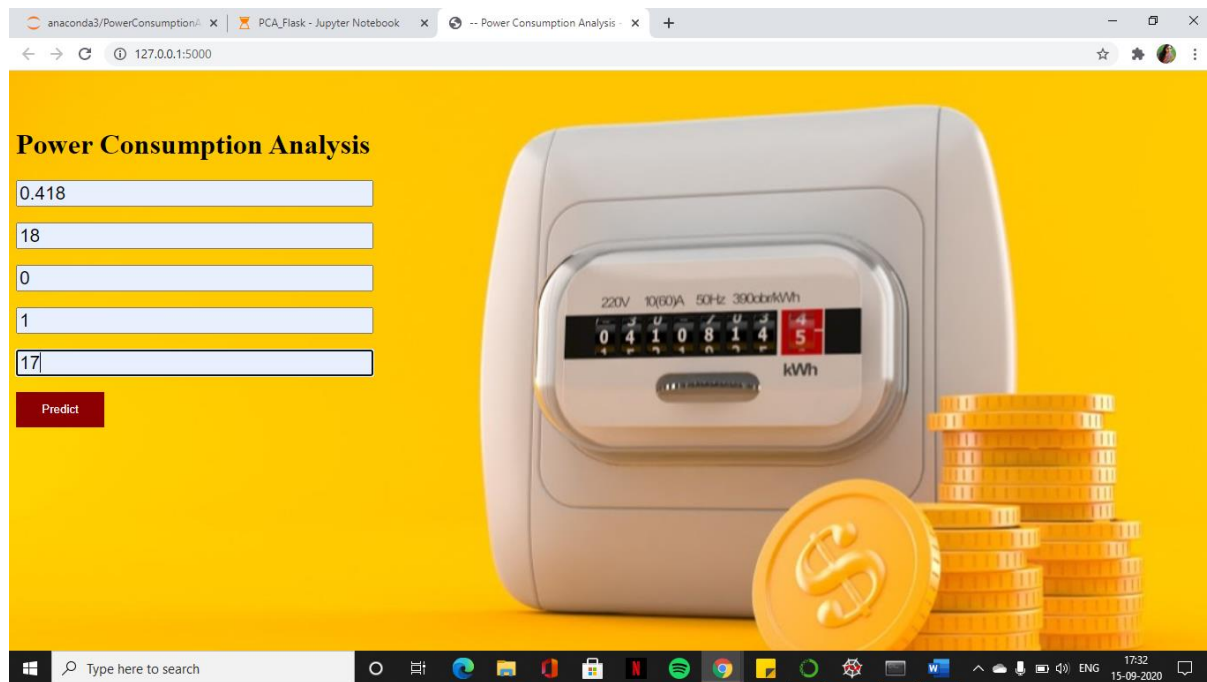


Fig.28: Entering Values

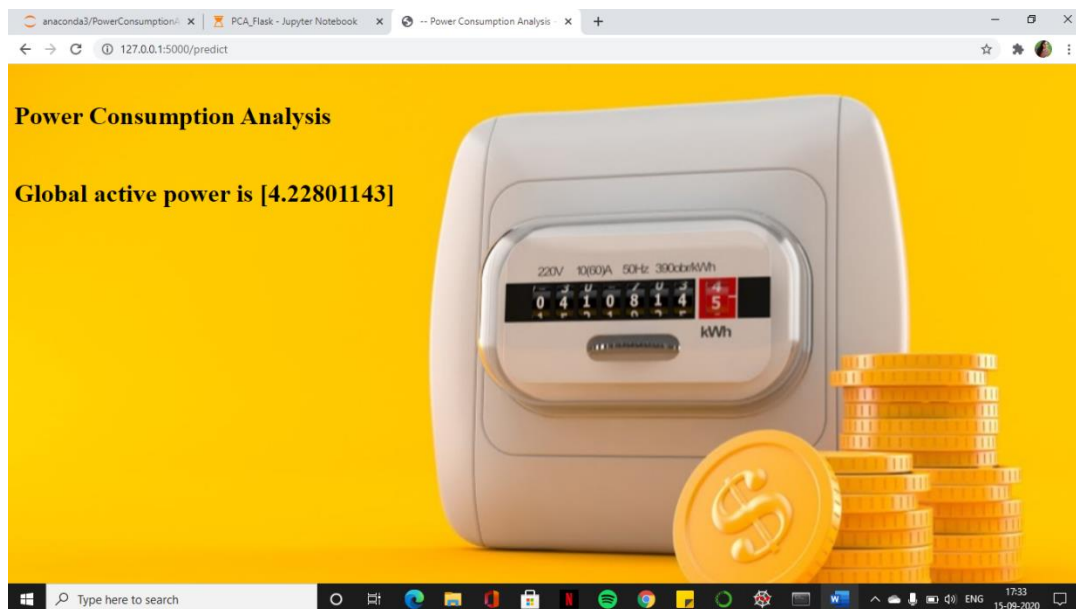


Fig.29: Total power consumption

Finally, the total power consumption by all the appliances is calculated and displayed.

8.2 Build Python Code

Let us build app.py flask file which is a web framework written in python for server-side scripting. Let's see step by step procedure for building the backend application import required libraries

```
from flask import Flask,request,render_template
import numpy as np
import pandas as pd
import pickle
import os
```

Fig.30: importing required libraries

Configure app.py to fetch the user inputs from the UI, process the values, and return the prediction.

```
@app.route('/')
def home():
    return render_template("pca.html")
@app.route('/predict',methods=["POST","GET"])

def predict():
    input_features = [float(x) for x in request.form.values()]
    features_value = [np.array(input_features)]

    features_name = ['Global_reactive_power', 'Global_intensity', 'Sub_metering_1',
                    'Sub_metering_2', 'Sub_metering_3']

    df = pd.DataFrame(features_value, columns=features_name)
    output = model.predict(df)

    return render_template('result1.html', prediction_text=output)
```

Fig.31:Configure app.py

```
if __name__=="__main__":
    #port = int(os.getenv('PORT', 8080))
    #app.run(host='0.0.0.0', port=port, debug=False)
    app.run(debug=False)

* Serving Flask app "__main__" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off

* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Fig.32:Runcode

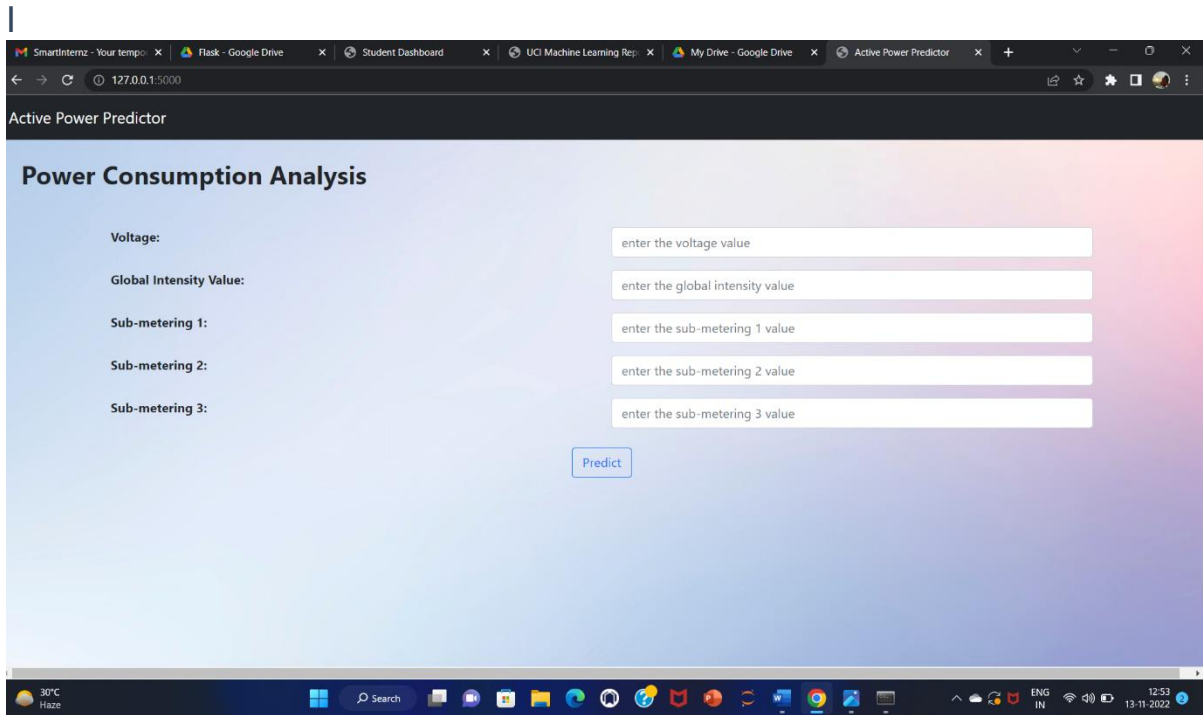
9.Run the App

Execute the python code and after the module is running, open index.html page and scroll down to find the buttons to test with.

- Open the anaconda prompt from the start menu.
- Navigate to the folder where your app.py resides.
- Now type “python app.py” command.
- It will show the local host where your app is running on `http://127.0.0.1.5000/`
- Copy that local host URL and open that URL in the browser. It does navigate me to where you can view your web page.
- Enter the values, click on the predict button and see the result/prediction on the web page.

10. Result

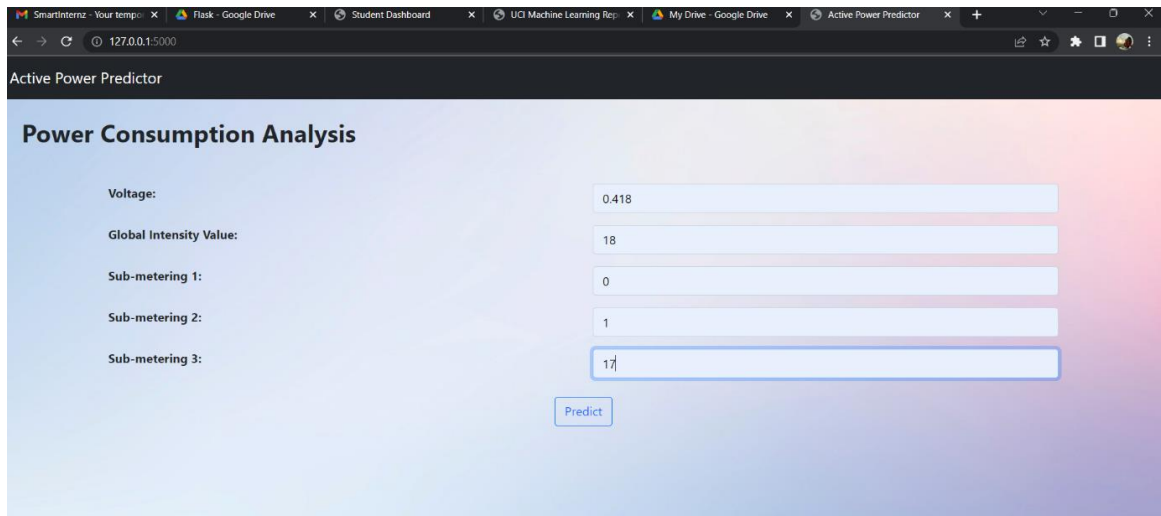
Let's see how our output page looks like:



The screenshot shows a web browser window with the title "Active Power Predictor". The page has a dark header bar with the same title. Below the header, the main content area has a light blue and purple gradient background. The title "Power Consumption Analysis" is displayed in bold. On the left side, there are five labels: "Voltage:", "Global Intensity Value:", "Sub-metering 1:", "Sub-metering 2:", and "Sub-metering 3:". On the right side, there are five corresponding input fields with placeholder text: "enter the voltage value", "enter the global intensity value", "enter the sub-metering 1 value", "enter the sub-metering 2 value", and "enter the sub-metering 3 value". Below these input fields is a blue "Predict" button. The browser's address bar shows the URL "127.0.0.1:5000". The Windows taskbar is visible at the bottom, showing the time as 12:53 on 13-11-2022.

Fig.33:ouput page

Enter the values and click on Predict button to view the result on “result1.html”.



The screenshot shows the same web application as Fig.33, but with values entered in the input fields. The values are: "0.418" for Voltage, "18" for Global Intensity Value, "0" for Sub-metering 1, "1" for Sub-metering 2, and "17" for Sub-metering 3. The "Predict" button is still visible below the input fields. The browser's address bar shows the URL "127.0.0.1:5000". The Windows taskbar is visible at the bottom, showing the time as 12:53 on 13-11-2022.

Fig.34: Entering values

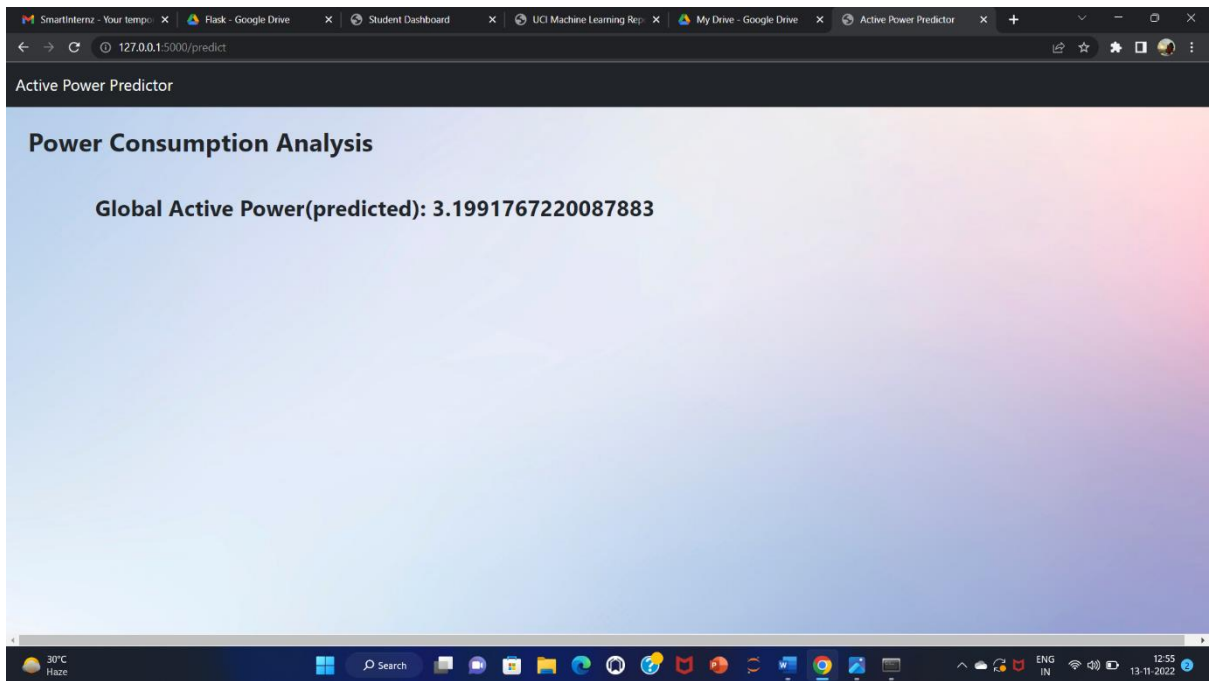


Fig.35:predicting output

Finally, total power consumption by all the appliances is calculated and displayed.

11.Applications

- You'll be able to understand the problem to classify if it is a regression or a classification kind of problem.
- You will be able to know how to pre-process/clean the data using different data pre-processing techniques.
- Applying different algorithms according to the dataset
- You will be able to know how to find the accuracy of the model.
- You will be able to build web applications using the Flask framework

12. Conclusion

This paper realizes the experiment on the individual household electric power consumption dataset. An extensible framework is established in the experiment. The correlation between the features obtained in the analysis module in the framework is consistent with the prediction results, global active Power feature is relatively important to the results. In addition, the experimental results show that the prediction effect of neural network model is better than that of linear model.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

13.References

- [1] Shi H, Xu M, Li R. Deep learning for household load forecasting—A novel pooling deep RNN. *IEEE Trans Smart Grid* 2018;9(5):5271–80.
- [2] Kim TY, Cho SB. Predicting the household power consumption using CNN-LSTM hybrid networks. In: *Intelligent data engineering and automated learning – IDEAL 2018. Lecture notes in computer science*, vol. 11314, Cham: Springer; 2018.
- [3] Loginov A, Heywood MI, Wilson G. Benchmarking a coevolutionary streaming classifier under the individual household electric power consumption dataset. In: *2016 international joint conference on neural networks*. 2016, p. 2834–41.
- [4] Hajjaji I, Alami HE, El-Fenni MR, Dahmouni H. Evaluation of artificial intelligence algorithms for predicting power consumption in university campus microgrid. In: *2021 international wireless communications and mobile computing*. 2021, p. 2121–6.
- [5] Pinto Tiago, Praça Isabel, Vale Zita, Silva Jose. Ensemble learning for electricity consumption forecasting in office buildings. *Neurocomputing* 2021;423:747–55.
- [6] Lim C, Choi H. Deep learning-based analysis on monthly household consumption for different electricity contracts. In: *2020 IEEE international conference on big data and smart computing*. 2020, p. 545–7.
- [7] Hyeon J, Lee H, Ko B, Choi H-J. Building energy consumption forecasting: Enhanced deep learning approach. In: *2nd international workshop on big data analysis for smart energy*. p. 22–5.
- [8] Liang K, Liu F, Zhang Y. Household power consumption prediction method based on selective ensemble learning. *IEEE Access* 2020;8:95657–66.

- [9] Pandiaraj K, Sivakumar P, Jeya Prakash K. Machine learning based effective linear regression model for TSV layer assignment in 3DIC. *Microprocess Microsyst* 2021;83.
- [10] Barbhuiya Sakil, Kilpatrick Peter, Nikolopoulos Dimitrios S. Linear regression based DDoS attack detection. In: 13th international conference on machine learning and computing. p. 568–74.
- [11] Zekic-Sušac M, Knežević M, Scitovski R. Modeling the cost of energy in public sector buildings by linear regression and deep learning. *CEJOR Cent Eur J Oper Res* 2021;29:307–22.
- [12] Miri D, Khedher A, BenOthman K. Tracking of trajectory and fault estimation of MIABOT robot using an artificial neural network. In: 2021 18th International multi-conference on systems, signals & devices. 2021 p. 1296–301.
- [13] Abdallah_Qasaimeh Bashar Mohammad, Abdallah Ammar, Ratte Sylvie. Detecting depression in Alzheimer and MCI using artificial neural networks (ANN). In: International conference on data science, E-learning and information systems. 2021, p. 250–3.
- [14] El-Khalek AAA, Khalil AT, El-Soud MAA, Yasser I. Classification of galaxy images using computer vision and artificial neural network techniques: A survey. In: Proceedings of the international conference on artificial intelligence and computer vision, vol. 1377, 2021.