PROJECT REPORT

Team id:- NM2023TMID11742

TOPIC :- A Reliable Energy Consumption Analysis System for Energy-Efficient Appliances.

Table of Contents

Completed By :-
Team Lead :- RANA PRATAP RAO
Team member 1 :- PRATIK KUMAR JHA
Team member 2 :- VISHAL KUMAR
Team member 3 :- VIVEK KUMAR

Introduction

The Energy Consumption Analysis System is a project aimed for analyzing and predicting energy consumption for energy-efficient appliances in households. This system utilizes Machine Learning(ML) algorithms to analyze the historical energy consumption data and make accurate predictions for future usage. The goal of the project is to assist house owners or utility companies to manage their energy usage, reducing waste, and lowering costs.

This project is useful for the house owners or utility companies to manage their electricity consumptions. Manage the record of the electric consumptions and finding the way of using less energy by maintaining data of the particular house or utility companies which they can use to increase energy-efficiency at their places.

Overview of the Complete Project

Home Page:- The home page provides an overview of the project and displays the project name, "Energy Consumption."

About Page:- The about page provides detailed information about the project, including its purpose, methodology, and potential impact on energy usage and sustainability. It also includes a visually appealing image and accompanying content.

Contact Page:- The contact page allows users to get in touch with the project team. It includes contact details such as location, email address, and contact number. Users can fill in their name, email, and message in the provided input fields and submit the form to send a message to the project team.

Prediction Page:- The prediction page enables users to make energy consumption predictions by providing input values for various parameters such as global reactive power, global intensity, sub metering 1, metering 2, and metering 3. Users can then click the "Predict" button to submit the form and receive the predicted result.

Prediction Result Page:- The prediction result page displays the predicted result obtained from the prediction page. Users can view the result in a visually appealing format.

TOOLS AND TECHNOLOGIES USED

HTML:- Used for creating the structure and layout of the web pages.

CSS:- Used for styling the web pages and enhancing the visual appearance.

Flask:- Used as the backend framework to handle the routing and logic of the web application.

Python:- Used for implementing the backend functionality, data processing, and prediction algorithms.

Data Visualization:- The system incorporates data visualization techniques to present energy consumption trends and patterns in a visually appealing and intuitive manner. Graphs, charts, and interactive visual elements provide users with a clear understanding of their energy usage.

PROBLEM STATEMENT &ARCHITECTURE

Problem Statement :- I am a house owner who is trying to monitor and optimize the energy usage of my energy-efficient appliances. But, I am facing challenges in accurately tracking individual appliance energy consumption, which makes it difficult for me to identify energy-saving opportunities and make informed decisions to reduce my energy consumption. This lack of visibility and analysis capabilities leaves me feeling frustated and unable to effectively manage and optimize my energy usage, leading to potential wastage and higher energy bills.

Architecture :- The Energy Consumption Analysis System follows a client-server architecture. The client-side consists of HTML, CSS, and JavaScript, while the server-side is implemented using the Flask framework and Python. The server handles the routing, data processing, and prediction functionalities, while the client-side renders the web pages and interacts with the server through HTTP requests.

Data Collection & Preparation

Dataset link :-
https://www.kaggle.com/datasets/uciml/electric-power-consumption-data-set

Data Collection :- Data collection is a crucial step in the Energy Consumption Analysis System. The system collects energy consumption data from households through smart meters or other monitoring devices. Additional data sources, such as weather data or occupancy information, may also be integrated into the system. This diverse dataset provides comprehensive information for accurate analysis and prediction.

```
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression
from xgboost import XGBRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import Ridge
```

```
[2]  dataset=pd.read_csv("Household_power_consumption.csv")
```

```
[3]  dataset.head()
```

	datetime	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3	sub_metering_4
0	2006-12-16 17:24:00	4.216	0.418	234.84	18.4	0.0	1.0	17.0	52.266670
1	2006-12-16 17:25:00	5.360	0.436	233.63	23.0	0.0	1.0	16.0	72.333336
2	2006-12-16 17:26:00	5.374	0.498	233.29	23.0	0.0	2.0	17.0	70.566666
3	2006-12-16 17:27:00	5.388	0.502	233.74	23.0	0.0	1.0	17.0	71.800000
4	2006-12-16 17:28:00	3.666	0.528	235.68	15.8	0.0	1.0	17.0	43.100000

```
[4]  dataset.tail()
```

	datetime	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3	sub_metering_4
2075254	2010-11-26 20:58:00	0.946	0.0	240.43	4.0	0.0	0.0	0.0	15.766666
2075255	2010-11-26 20:59:00	0.944	0.0	240.00	4.0	0.0	0.0	0.0	15.733334
2075256	2010-11-26 21:00:00	0.938	0.0	239.82	3.8	0.0	0.0	0.0	15.633333
2075257	2010-11-26 21:01:00	0.934	0.0	239.70	3.8	0.0	0.0	0.0	15.566667
2075258	2010-11-26 21:02:00	0.932	0.0	239.55	3.8	0.0	0.0	0.0	15.533334

Data Processing :- Once the data is collected, it undergoes pre-processing and cleaning. This involves handling missing values, normalizing the data, and transforming it into a suitable format for machine learning algorithms. Data processing techniques ensure the quality and reliability of the input data.

Complete information of dataset

```
In [7]: dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2075259 entries, 2006-12-16 17:24:00 to 2010-11-26 21:02:00
Data columns (total 7 columns):
 #   Column                 Dtype
---  ------                 -----
 0   Global_active_power    object
 1   Global_reactive_power  object
 2   Voltage                object
 3   Global_intensity       object
 4   Sub_metering_1         object
 5   Sub_metering_2         object
 6   Sub_metering_3         float64
dtypes: float64(1), object(6)
memory usage: 126.7+ MB
```

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
```

Handling missing values

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

datetime	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_mete
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)
```

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

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

```
(2049280, 7)
```

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
```


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
```

Perform the Analysis of your Project

Descriptive Statistical :-

```
# Descriptive statistics
print(df.describe())
```

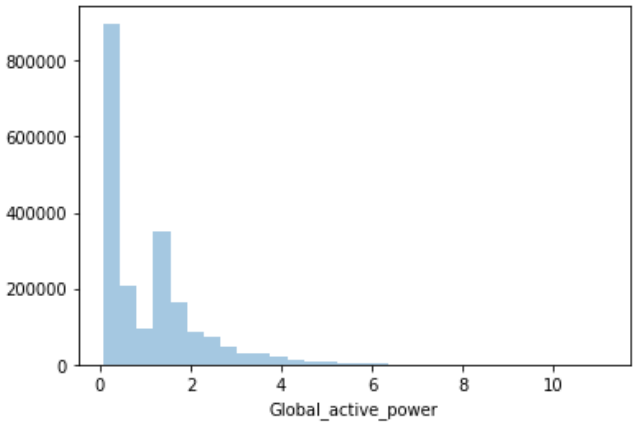
```
       Global_active_power  Global_reactive_power       Voltage  \
count         2.049280e+06           2.049280e+06  2.049280e+06
mean          1.091615e+00           1.237145e-01  2.408399e+02
std           1.057294e+00           1.127220e-01  3.239987e+00
min           7.600000e-02           0.000000e+00  2.232000e+02
25%           3.080000e-01           4.800000e-02  2.389900e+02
50%           6.020000e-01           1.000000e-01  2.410100e+02
75%           1.528000e+00           1.940000e-01  2.428900e+02
max           1.112200e+01           1.390000e+00  2.541500e+02

       Global_intensity  Sub_metering_1  Sub_metering_2  Sub_metering_3
count      2.049280e+06    2.049280e+06    2.049280e+06    2.049280e+06
mean       4.627759e+00    1.121923e+00    1.298520e+00    6.458447e+00
std        4.444396e+00    6.153031e+00    5.822026e+00    8.437154e+00
min        2.000000e-01    0.000000e+00    0.000000e+00    0.000000e+00
25%        1.400000e+00    0.000000e+00    0.000000e+00    0.000000e+00
50%        2.600000e+00    0.000000e+00    0.000000e+00    1.000000e+00
75%        6.400000e+00    0.000000e+00    1.000000e+00    1.700000e+01
max        4.840000e+01    8.800000e+01    8.000000e+01    3.100000e+01
```

Visual Analysis :-

Univariate

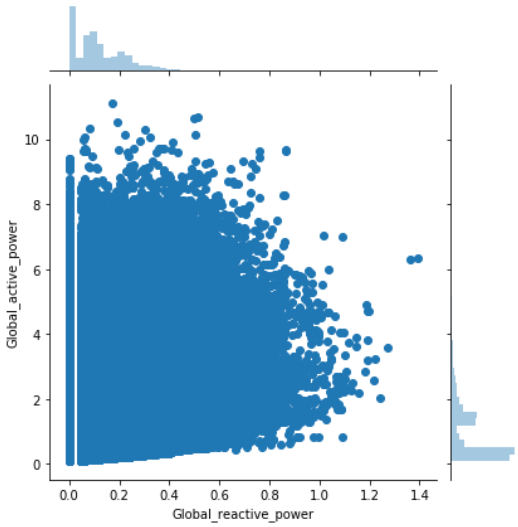
```
In [23]: sns.distplot(dataset['Global_active_power'],kde=False,bins=30)
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x236dd9e2448>
```



Bivariate

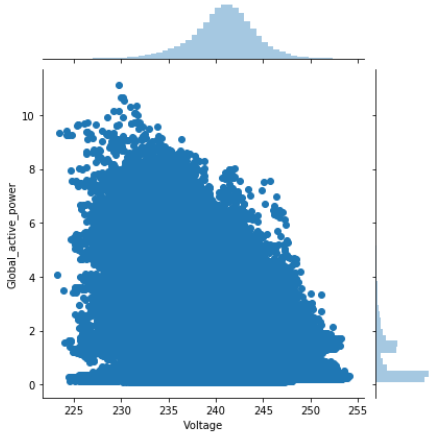
Data Visualization

```
In [29]: sns.jointplot( x = 'Global_reactive_power' , y = 'Global_active_power' , data = dataset , kind = 'scatter')
Out[29]: <seaborn.axisgrid.JointGrid at 0x23687af36c8>
```



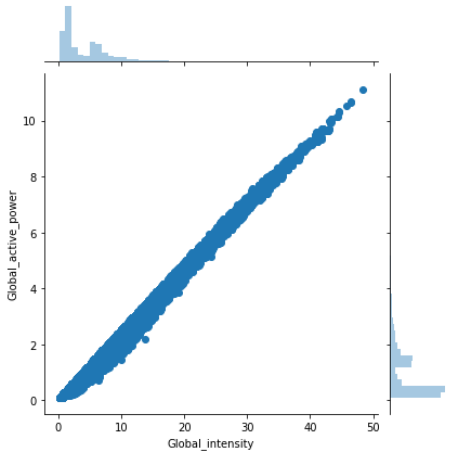
In [30]:
```
sns.jointplot( x = 'Voltage' , y = 'Global_active_power' , data = dataset , kind = 'scatter')
```

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



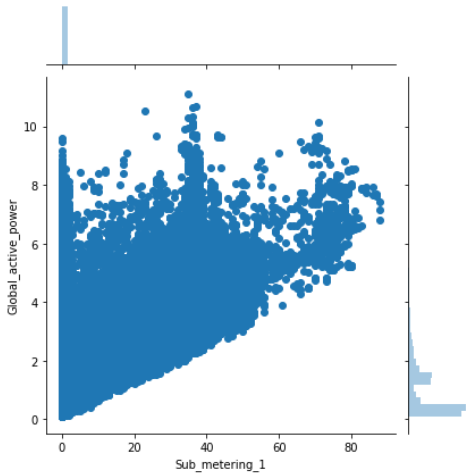
In [31]:
```
sns.jointplot( x = 'Global_intensity' , y = 'Global_active_power' , data = dataset , kind = 'scatter')
```

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



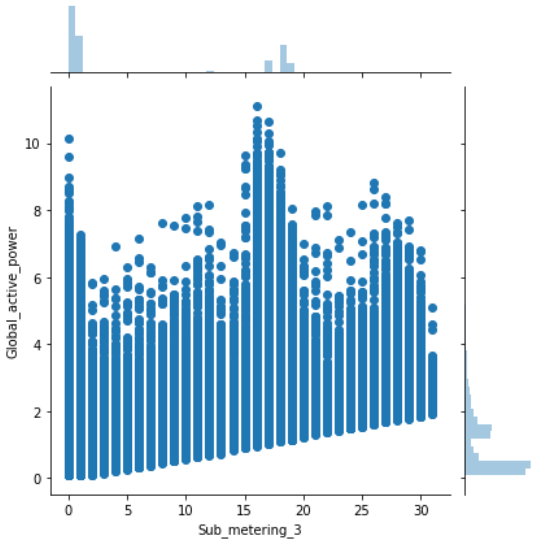
In [32]:
```
sns.jointplot( x = 'Sub_metering_1' , y = 'Global_active_power' , data = dataset , kind = 'scatter')
```

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



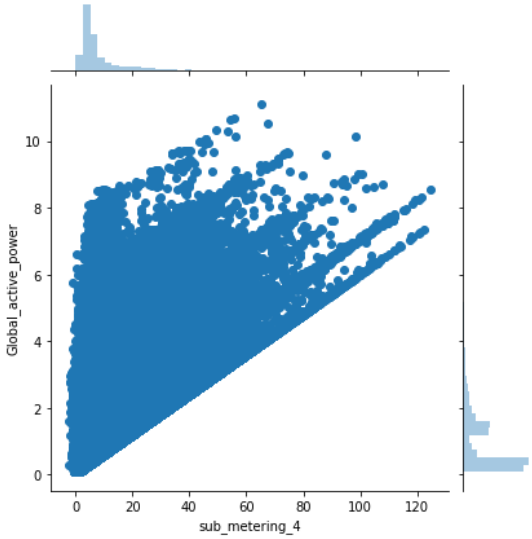
In [34]:
```
sns.jointplot( x = 'Sub_metering_3' , y = 'Global_active_power' , data = dataset , kind = 'scatter')
```

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



In [35]:
```
sns.jointplot( x = 'sub_metering_4' , y = 'Global_active_power' , data = dataset , kind = 'scatter')
```

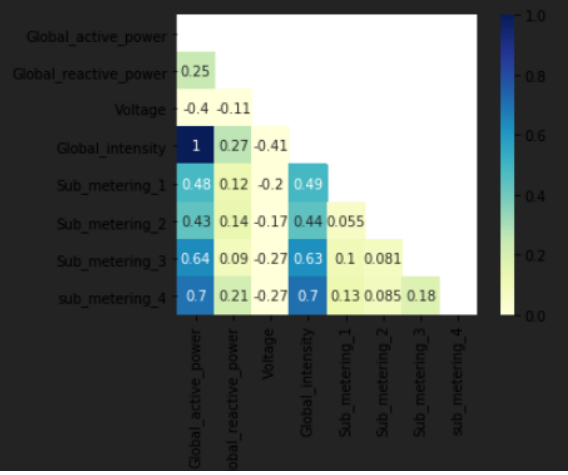
Out[35]: <seaborn.axisgrid.JointGrid at 0x2369fa4d548>



Multivariate

Analysis using heatmap

```
[28]:
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);
```



Machine Learning Models in Project

```python
# Train a linear regression model
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
```

Python

```
LinearRegression
LinearRegression()
```

```python
# Make predictions on the test set
linear_pred = linear_model.predict(X_test)
```

Python

```python
# Train an XGB Regressor model
xgb_model = XGBRegressor()
xgb_model.fit(X_train, y_train)
```

Python

```
XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...)
```

```python
from sklearn.model_selection import cross_val_score
cv=cross_val_score(linear_model,X,y,cv=5)
```

```python
np.mean(cv)
```

```
0.9982971156884088
```

```python
import pickle
filename='PCASS_model.pkl'
pickle.dump(linear_model,open(filename,'wb'))
```

```python
model=pickle.load(open('PCASS_model.pkl', 'rb'))
print(model.predict([[.481, 18.4, 0.0, 1.0, 17.0]]))
```

```
[4.31179297]
C:\Users\LENOVO\AppData\Local\Packages\PythonSoftwareFoundation.Pyth
  warnings.warn(
```

```python
# Train a Random Forest Regressor model
rf_model = RandomForestRegressor()
rf_model.fit(X_train, y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor()
```

```python
# Make predictions on the test set
rf_pred = rf_model.predict(X_test)
```

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

# Evaluate linear regression model
linear_mse = mean_squared_error(y_test, linear_pred)
linear_rmse = mean_squared_error(y_test, linear_pred, squared=False)
linear_mae = mean_absolute_error(y_test, linear_pred)
linear_r2 = r2_score(y_test, linear_pred)

print("Linear Regression Metrics:")
print("Mean Squared Error (MSE):", linear_mse)
print("Root Mean Squared Error (RMSE):", linear_rmse)
print("Mean Absolute Error (MAE):", linear_mae)
print("R-squared (R2) Score:", linear_r2)
```

```
Linear Regression Metrics:
Mean Squared Error (MSE): 0.0018048510362474114
Root Mean Squared Error (RMSE): 0.04248353841486619
Mean Absolute Error (MAE): 0.027138585467910953
R-squared (R2) Score: 0.998365740219586
```

```python
# Evaluate XGB Regressor model
xgb_mse = mean_squared_error(y_test, xgb_pred)
xgb_rmse = mean_squared_error(y_test, xgb_pred, squared=False)
xgb_mae = mean_absolute_error(y_test, xgb_pred)
xgb_r2 = r2_score(y_test, xgb_pred)

print("XGB Regressor Metrics:")
print("Mean Squared Error (MSE):", xgb_mse)
print("Root Mean Squared Error (RMSE):", xgb_rmse)
print("Mean Absolute Error (MAE):", xgb_mae)
print("R-squared (R2) Score:", xgb_r2)
```

```
XGB Regressor Metrics:
Mean Squared Error (MSE): 3.994307915145391e-06
Root Mean Squared Error (RMSE): 0.001998576472178483
Mean Absolute Error (MAE): 0.0013712860352856151
R-squared (R2) Score: 0.9999963832268453
```

```python
# Evaluate Random Forest Regressor model
rf_mse = mean_squared_error(y_test, rf_pred)
rf_rmse = mean_squared_error(y_test, rf_pred, squared=False)
rf_mae = mean_absolute_error(y_test, rf_pred)
rf_r2 = r2_score(y_test, rf_pred)

print("Random Forest Regressor Metrics:")
print("Mean Squared Error (MSE):", rf_mse)
print("Root Mean Squared Error (RMSE):", rf_rmse)
print("Mean Absolute Error (MAE):", rf_mae)
print("R-squared (R2) Score:", rf_r2)
```

```
Random Forest Regressor Metrics:
Mean Squared Error (MSE): 1.1044615132566399e-08
Root Mean Squared Error (RMSE): 0.00010509336388453078
Mean Absolute Error (MAE): 1.0241608358432137e-06
R-squared (R2) Score: 0.999999989999302
```

Prediction & Future Enhancements

Prediction:-

The prediction phase involves taking user input for the parameters such as global reactive power, global intensity, sub metering 1, metering 2, and metering 3. These input values are then passed through the trained machine learning models to generate predictions for future energy consumption. The predicted results are displayed to the user, providing valuable insights into their energy usage patterns and potential areas for improvement.

Future Enhancements :-

Integration with smart home systems: Enhance the system by integrating it with smart home systems to automatically collect real-time energy consumption data and control appliances based on energy-saving recommendations.

Energy efficiency recommendations: Provide personalized energy efficiency recommendations to users based on their consumption patterns, helping them optimize their energy usage and reduce waste.

Integration with utility companies: Collaborate with utility companies to incorporate real-time energy pricing and incentivize energy-efficient behaviour among users.

Mobile application: Develop a mobile application version of the Energy Consumption Analysis System, allowing users to access and monitor their energy consumption data on the go.

Advanced analytics: Expand the system's analytics capabilities by incorporating advanced techniques such as anomaly detection, clustering, or time-series forecasting to provide more in-depth insights into energy consumption patterns.

Social engagement: Implement social sharing features to encourage users to share their energy-saving achievements and engage in friendly competitions, fostering a sense of community and awareness around energy conservation.

Integrate With Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server-side script
- Run the web application

Build HTML Code:-

In this HTML page, we will create three pages namely

- index.html
- inspect.html
- output.html

and save them in the templates folder

Build Python Code:-

Import the libraries

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

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.

```
model=pickle.load(open('PCASSS_model.pkl','rb'))
app=Flask(__name__)
```

```
@app.route("/")
def f():
    return render_template("index.html")

@app.route("/inspect")
def inspect():
    return render_template("inspect.html")
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI: -

```
@app.route("/home", methods=["GET", "POST"])
def home():
    GlobalReactivePower = float(request.form['GlobalReactivePower'])
    Global_intensity = float(request.form['Global_intensity'])
    Sub_metering_1 = float(request.form['Sub_metering_1'])
    Sub_metering_2 = float(request.form['Sub_metering_2'])
    Sub_metering_3 = float(request.form['Sub_metering_3'])
    X = [[GlobalReactivePower,Global_intensity, Sub_metering_1, Sub_metering_2, Sub_metering_3]]

    output = round(model.predict(X)[0],3)
    return render_template('output.html',output=output)
```

Here we are routing our app to home() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the output.html page earlier.

Main Function:-

```
if __name__ == "__main__" :
    app.run(debug=True)
```

Run the Web Application

Open anaconda prompt from the start menu

Navigate to the folder where your python script is.

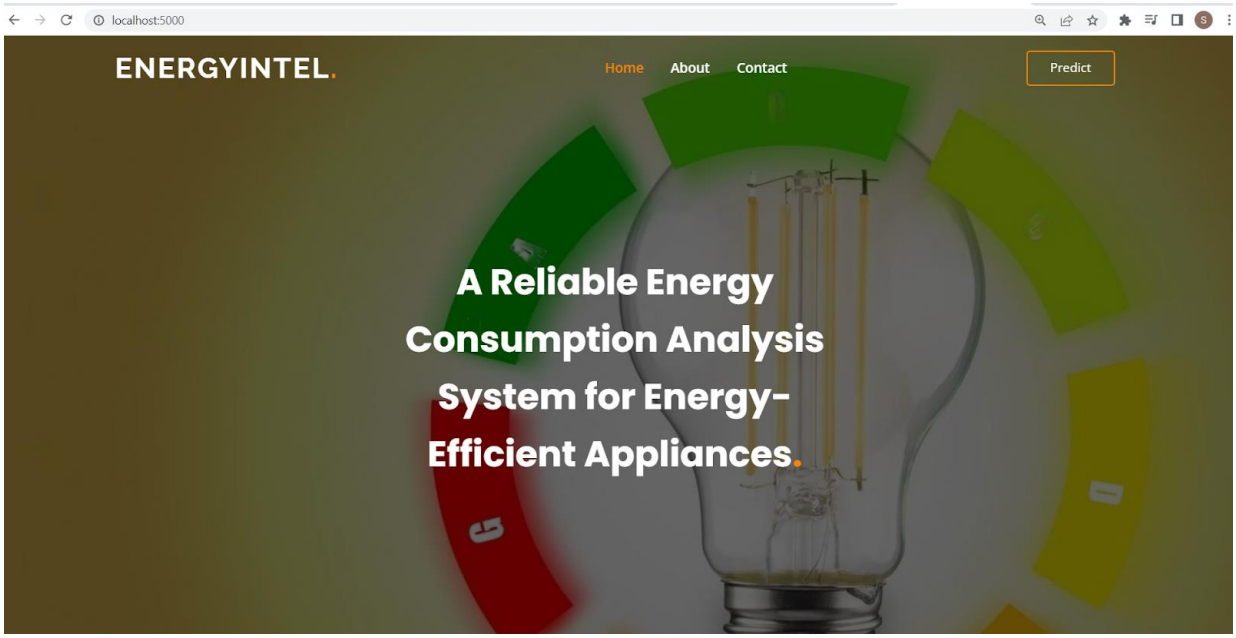
Now type “python app.py” command

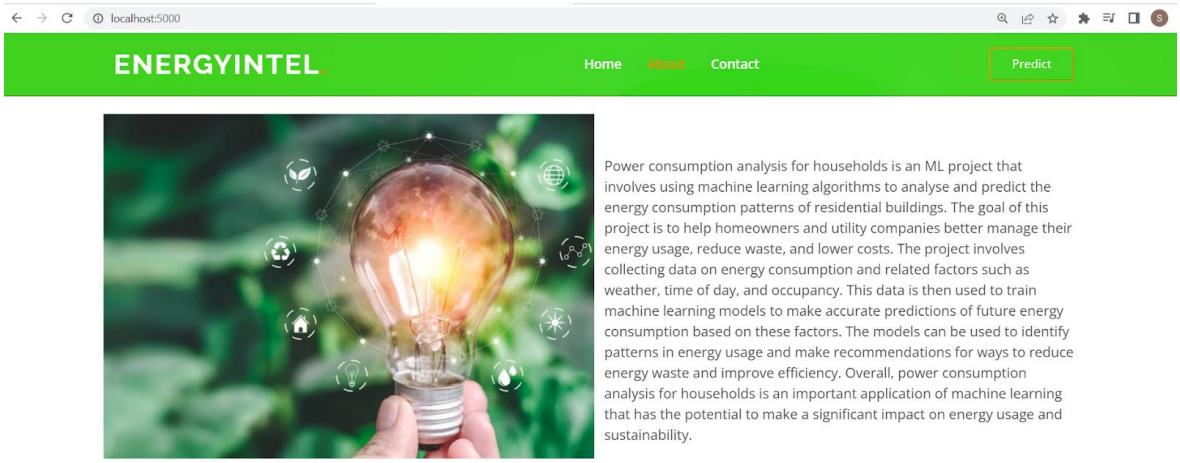
Navigate to the localhost where you can view your web page.

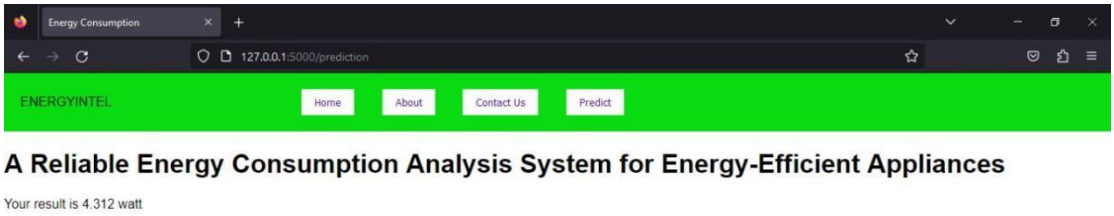
Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

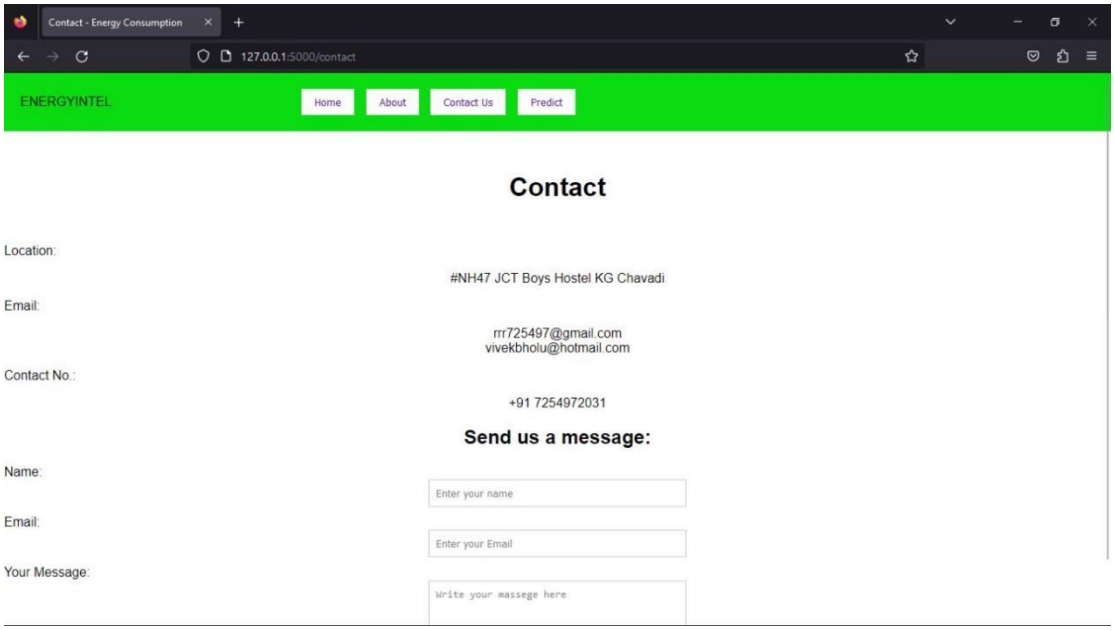
```
 warnings.warn(
* Serving Flask app 'app'
* Debug mode: on
ARNING: This is a development server. Do not use it in a
* Running on http://127.0.0.1:5000
ress CTRL+C to quit
```

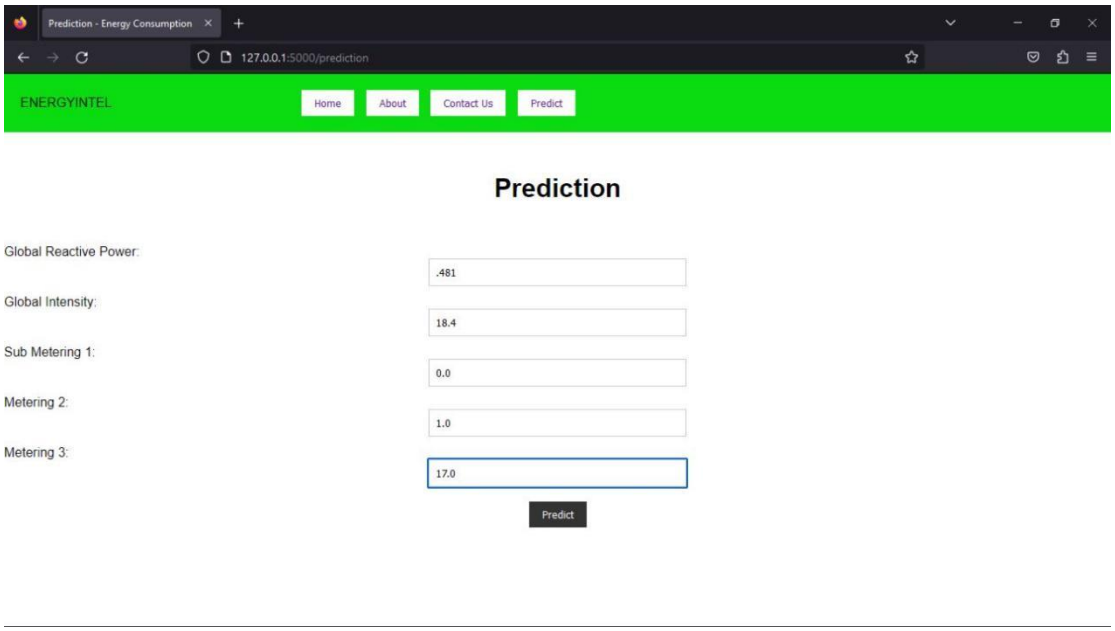
Now, Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result

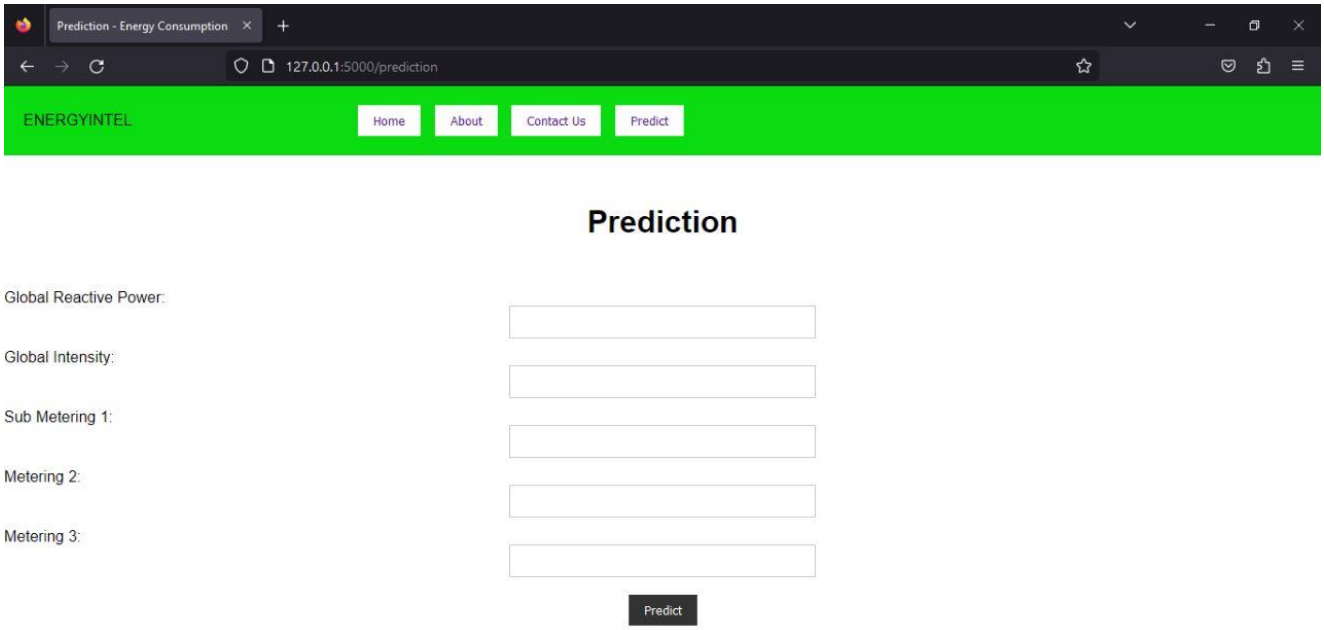



localhost:5000
ENERGYINTEL.
Home
About
Contact
Predict
Power consumption analysis for households is an ML project that involves using machine learning algorithms to analyse and predict the energy consumption patterns of residential buildings. The goal of this project is to help homeowners and utility companies better manage their energy usage, reduce waste, and lower costs. The project involves collecting data on energy consumption and related factors such as weather, time of day, and occupancy. This data is then used to train machine learning models to make accurate predictions of future energy consumption based on these factors. The models can be used to identify patterns in energy usage and make recommendations for ways to reduce energy waste and improve efficiency. Overall, power consumption analysis for households is an important application of machine learning that has the potential to make a significant impact on energy usage and sustainability.


Energy Consumption
127.0.0.1:5000/prediction
ENERGYINTEL
Home
About
Contact Us
Predict
A Reliable Energy Consumption Analysis System for Energy-Efficient Appliances
Your result is 4.312 watt


Contact - Energy Consumption
127.0.0.1:5000/contact
ENERGYINTEL
Home
About
Contact Us
Predict
Contact
Location:
#NH47 JCT Boys Hostel KG Chavadi
Email:
rrr725497@gmail.com
vivekbholu@hotmail.com
Contact No.:
+91 7254972031
Send us a message:
Name:
Enter your name
Email:
Enter your Email
Your Message:
Write your massege here


Prediction - Energy Consumption
127.0.0.1:5000/prediction
ENERGYINTEL
Home
About
Contact Us
Predict
Prediction
Global Reactive Power:
.481
Global Intensity:
18.4
Sub Metering 1:
0.0
Metering 2:
1.0
Metering 3:
17.0
Predict


Prediction - Energy Consumption
127.0.0.1:5000/prediction
ENERGYINTEL
Home
About
Contact Us
Predict
Prediction
Global Reactive Power:
Global Intensity:
Sub Metering 1:
Metering 2:
Metering 3:
Predict

Outro & Conclusion

Hence, the above shared project report shows the whole & sole of this project which is named as a "*A Reliable Energy Consumption Analysis System for Energy-Efficient Appliances*".

Refrences

https://aws.amazon.com/blogs/industries/voice-applications-in-clinical-research-powered-by-ai-on-aws-part-1-architecture-and-design-considerations/