**MEASURES** **ENERGY** **CONSUMPTION**

**USING** **PYTHON**

**TEAM MEMBER:**

**AU411521104032 –K.ELAKIYA**

**Projet: Measures Energy Consumption**

**Abstract:**

Measuring energy consumption is an important step in reducing energy waste and improving energy efficiency. Python can be used to measure energy consumption in a variety of ways, including:

* Using the Intel "Running Average Power Limit" (RAPL) technology to estimate the power consumption of a CPU.
* Using the Nvidia "Nvidia Management Library" technology to measure the energy consumption of Nvidia devices.
* Using the pyJoules library to measure the energy consumption of a host machine along the execution of a piece of Python code.

Once the energy consumption data has been collected, it can be pre-processed using Python to clean, transform, and prepare the data for analysis. This may involve handling missing values, converting data types, resampling the data, creating new features, removing outliers, and splitting the data into training and testing sets.Once the data is pre-processed, it can be used for machine learning or other analysis tasks. For example, machine learning models can be trained to predict energy consumption based on historical data and other factors. This information can then be used to develop strategies for reducing energy consumption.

Here is an example of how to use Python to measure the energy consumption of a piece of code:

Python

import pyJoules

# Create an EnergyContext object

ctx = pyJoules.EnergyContext()

# Start recording energy consumption

ctx.start()

# Execute the code to be measured

# ...

# Stop recording energy consumption

ctx.stop()

# Get the total energy consumed

total\_energy\_consumed = ctx.result.total\_energy\_joules

# Print the total energy consumed

print('Total energy consumed:', total\_energy\_consumed)

Use code with caution

content\_copy

Python is a powerful tool for measuring and analyzing energy consumption data. It can be used to develop custom solutions for a variety of needs.

tuneshare

more\_vert

add\_photo\_alternate

mic

send\_spark

**Objectives:**

* Quantify energy usage in order to identify areas for improvement.
* Track energy consumption over time to identify trends and patterns.
* Evaluate the effectiveness of energy efficiency measures.
* Develop predictive models to forecast future energy consumption.
* Support informed decision-making about energy management.

Python can be used to measure energy consumption at the following levels:

* Device level: Measuring the energy consumption of individual devices, such as computers, servers, and network equipment.
* Facility level: Measuring the energy consumption of entire buildings or campuses.
* Grid level: Measuring the energy consumption of an entire region or country.

**PHASE 1: DATA PREPOROCESSING AND FEATURE EMGINEERING**

**1.DATA SOURCE:**

* Smart meters: Smart meters are electronic devices that measure and record energy consumption in real time. Data from smart meters can be collected using Python libraries such as smartmeterpy.
* Building management systems (BMS): BMS are computer systems that monitor and control the operation of building systems, such as HVAC, lighting, and security. Data from BMS can be collected using Python libraries such as bacnetpy and modbus-tk.
* Energy information systems (EIS): EIS are software systems that collect, analyze, and report on energy consumption data. Data from EIS can be collected using Python libraries such as PowerAPI and OpenADR.
* Public datasets: There are a number of public datasets that contain energy consumption data. Some examples include:
  + The US Energy Information Administration (EIA) Open Data Hub
  + The World Resources Institute Global Power Plant Database
  + The EU Open Data Portal

**2.DATA PREPROESSING:**

To pre-process energy consumption data in Python, we can use the following steps:

1. Import necessary libraries.

Python

import pandas as pd

import numpy as np

1. Load the data into a Pandas DataFrame.

Python

df = pd.read\_csv('energy\_consumption.csv', index\_col='Date')

1. Check for and handle missing values.

Python

# Check for missing values

print(df.isnull().sum())

# Handle missing values

# Replace missing values with mean or median

df = df.fillna(df.mean())

1. Convert data types to appropriate formats.

For example, if the Date column is not in a datetime format, we can convert it using the following code:

Python

df['Date'] = pd.to\_datetime(df['Date'])

1. Resample the data to a desired frequency.

For example, if we want to resample the data to a daily frequency, we can use the following code:

Python

df = df.resample('D').mean()

1. Create new features or transformations.

For example, we can create a new feature called Energy Consumption Difference by calculating the difference between the current energy consumption and the previous day's energy consumption.

Python

df['Energy Consumption Difference'] = df['Energy Consumption'].diff()

1. Remove outliers.

We can use various methods to remove outliers, such as IQR filtering, Z-score filtering, or clustering.

Python

# Remove outliers using IQR filtering

Q1 = df['Energy Consumption'].quantile(0.25)

Q3 = df['Energy Consumption'].quantile(0.75)

IQR = Q3 - Q1

df = df[(df['Energy Consumption'] >= Q1 - 1.5 \* IQR) & (df['Energy Consumption'] <= Q3 + 1.5 \* IQR)]

1. Split the data into training and testing sets.

We can use the following code to split the data into training and testing sets:

Python

from sklearn.model\_selection import train\_test\_split

X = df.drop('Energy Consumption', axis=1)

y = df['Energy Consumption']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42)

Once the data is pre-processed, it is ready to be used for machine learning or other analysis tasks.

**DATASET LINK:** [**https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption**](https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption)

**Feature** **extraction**:

Feature extraction is the process of transforming raw data into features that are more informative and useful for machine learning or other analysis tasks. In the context of energy consumption measurement, feature extraction can be used to create features that represent the following:

* Overall energy consumption: Features such as total energy consumption per day, week, month, or year.
* Patterns in energy consumption: Features such as peak energy consumption, average energy consumption, and energy consumption variance.
* Relationship between energy consumption and other variables: Features such as energy consumption per square foot of building space, energy consumption per employee, and energy consumption per unit of production.

Here are some examples of feature extraction techniques that can be used to measure energy consumption using Python:

* Calculate the difference between the current energy consumption and the previous day's energy consumption. This feature can be used to identify sudden changes in energy consumption, which may be indicative of problems or inefficiencies.
* Create a histogram of energy consumption over a period of time. This feature can be used to identify patterns in energy consumption, such as peak and off-peak periods.
* Calculate the correlation between energy consumption and other variables, such as temperature, humidity, and occupancy. This information can be used to develop predictive models of future energy consumption.

Once the features have been extracted, they can be used to train machine learning models to predict energy consumption, identify anomalies, or develop other insights into energy usage

**MODEL DEVELOPMENT:**

Once the energy consumption data has been pre-processed and features have been extracted, it can be used to develop machine learning models to predict energy consumption.

There are a variety of machine learning algorithms that can be used to predict energy consumption. Some of the most popular algorithms include:

* Linear regression
* Support vector regression (SVR)
* Random forest
* Gradient boosting
* Neural networks

The best algorithm to use will depend on the specific data set and the desired outcomes.Here is an example of how to use Python to develop a machine learning model to predict energy consumption using the linear regression algorithm:

Python

from sklearn.linear\_model import LinearRegression

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42)

# Create the linear regression model

model = LinearRegression()

# Train the model on the training data

model.fit(X\_train, y\_train)

# Evaluate the model on the testing data

y\_pred = model.predict(X\_test)

# Calculate the mean squared error

mse = mean\_squared\_error(y\_test, y\_pred)

# Print the mean squared error

print('Mean squared error:', mse)

**VISUALIZATION:**

There are a variety of Python libraries that can be used to visualize energy consumption data. Some of the most popular libraries include:

* Matplotlib: Matplotlib is a low-level library that provides a wide range of plotting and visualization tools.
* Seaborn: Seaborn is a high-level library that builds on top of Matplotlib to provide more concise and informative visualizations.
* Plotly: Plotly is a library that can be used to create interactive visualizations.

Here is an example of how to use Matplotlib to create a line chart of energy consumption over time:

Python

import matplotlib.pyplot as plt

import pandas as pd

# Load the energy consumption data

df = pd.read\_csv('energy\_consumption.csv', index\_col='Date')

# Create a line chart of energy consumption over time

plt.plot(df['Energy Consumption'])

plt.xlabel('Date')

plt.ylabel('Energy Consumption (kWh)')

plt.title('Energy Consumption Over Time')

plt.show()

To automate the collection, analysis, and visualization of energy consumption data in Python, we can use the following steps:

1. Collect the energy consumption data. This can be done using a variety of methods, such as:
   * Reading data from smart meters
   * Querying building management systems (BMS)
   * Downloading data from energy information systems (EIS)
   * Scraping data from public websites
2. Store the energy consumption data. The data can be stored in a variety of formats, such as:
   * A relational database
   * A NoSQL database
   * A CSV file
   * A JSON file
3. Analyze the energy consumption data. We can use Python libraries such as Pandas, NumPy, and scikit-learn to analyze the data. This may involve:
   * Calculating aggregate statistics, such as total energy consumption, average energy consumption, and peak energy consumption
   * Identifying patterns in energy consumption, such as peak and off-peak periods
   * Developing predictive models of future energy consumption
4. Visualize the energy consumption data. We can use Python libraries such as Matplotlib, Seaborn, and Plotly to visualize the data. This may involve creating line charts, bar charts, histograms, and other visualizations.
5. Automate the process. We can use Python libraries such as Airflow and Prefect to automate the entire process of collecting, analyzing, and visualizing energy consumption data.

**AUTOMATION**:

Here is an example of a simple Python script that automates the collection, analysis, and visualization of energy consumption data:

import pandas as pd

import matplotlib.pyplot as plt

# Collect the energy consumption data

df = pd.read\_csv('energy\_consumption.csv', index\_col='Date')

# Calculate the total energy consumption for each day

total\_energy\_consumption = df['Energy Consumption'].sum()

# Create a line chart of total energy consumption over time

plt.plot(total\_energy\_consumption)

plt.xlabel('Date')

plt.ylabel('Total Energy Consumption (kWh)')

plt.title('Total Energy Consumption Over Time')

plt.show()

# Calculate the average energy consumption for each day

average\_energy\_consumption = df['Energy Consumption'].mean()

# Create a bar chart of average energy consumption by day

plt.bar(df.index, average\_energy\_consumption)

plt.xlabel('Date')

plt.ylabel('Average Energy Consumption (kWh)')

plt.title('Average Energy Consumption by Day')

plt.show()

Use code with caution.

content\_copy

This script will collect the energy consumption data from a CSV file, calculate the total and average energy consumption for each day, and create two visualizations: a line chart of total energy consumption over time and a bar chart of average energy consumption by day.