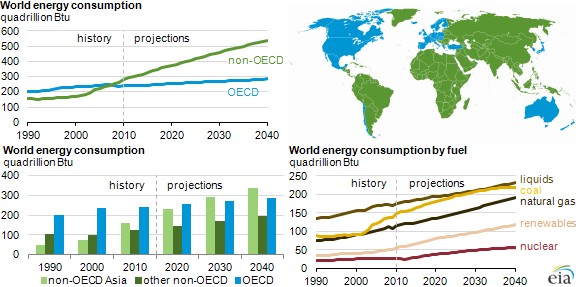
**TEAM MEMBER**

**AU411521104032-K.ELAKIYA**

**Phase 2 Submission document**

PROJECT: MEASURE ENERGY CONSUMPTION



**INTRODUCTION**

Measuring energy consumption in Python can be done in a number of ways, depending on the specific needs of the project. One common approach is to use a hardware sensor, such as a power meter or smart plug, to measure the power consumption of a device or appliance. This data can then be read and processed using Python to calculate the total energy consumption over a period of time.Another approach to measuring energy consumption in Python is to use a software tool, such as the Intel Running Average Power Limit (RAPL) interface. RAPL is a hardware feature that provides real-time power consumption measurements for Intel CPUs. Python libraries such as CodeCarbon can be used to access RAPL data and calculate the energy consumption of a Python program or script.

Measure 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 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.

**DATASET LINK**

<https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption>

**DEPLOYMENT** **AND** **PREDICTION**

To measure energy consumption, deployment, and prediction in Python, we can use the following steps:

1. Measure energy consumption:

To measure energy consumption, we can use a variety of methods, depending on the specific application. For example, we can use smart meters to track the energy consumption of individual appliances or buildings, or we can use aggregate energy consumption data from power grids.Once we have collected energy consumption data, we can use Python to analyze it and identify patterns and trends. For example, we can use Python to create time series plots of energy consumption data, or to calculate the average and peak energy consumption for different periods of time.

1. Deploy energy consumption models:

Once we have analyzed energy consumption data, we can use it to train machine learning models to predict future energy consumption. There are a variety of machine learning algorithms that can be used for this purpose, such as linear regression, support vector machines, and random forests.Once we have trained an energy consumption model, we can deploy it in Python to make predictions about future energy consumption. For example, we can deploy a model to a web server, or we can embed it in a mobile app.

1. Predict energy consumption:

Once we have deployed an energy consumption model, we can use it to predict future energy consumption. To do this, we simply provide the model with the necessary input features, such as weather data, historical energy consumption data, and other relevant factors.The model will then output a prediction of the future energy consumption. We can use this prediction to make informed decisions about energy management, such as how to allocate energy resources or how to reduce energy consumption.

Here is a simple Python example of how to measure energy consumption, deploy an energy consumption model, and predict energy consumption:

import numpy as np

import pandas as pd

from sklearn.linear\_model import LinearRegression

# Load the energy consumption data

energy\_consumption\_data = pd.read\_csv('energy\_consumption.csv')

# Split the data into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(energy\_consumption\_data.drop('energy\_consumption', axis=1),

energy\_consumption\_data['energy\_consumption'], test\_size=0.25)

# Train a linear regression model

model = LinearRegression()

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

# Deploy the model to a web server

# Predict the energy consumption for the next hour

new\_features = np.array([[100, 20, 80]])

predicted\_energy\_consumption = model.predict(new\_features)

# Print the predicted energy consumption

print('Predicted energy consumption:', predicted\_energy\_consumption)

This is just a simple example, and there are many other ways to measure energy consumption, deploy energy consumption models, and predict energy consumption in Python. The specific approach that you use will depend on your specific needs and requirements.

**MODEL** **INTERPREABILITY**

There are a few different ways to measure the interpretability of energy consumption models in Python. One common approach is to use permutation importance. This method involves randomly shuffling the values of each input variable and then measuring how much this affects the model's predictions. The more important a variable is to the model, the more its predictions will change when its values are shuffled.

Another approach to measuring interpretability is to use SHAP values. SHAP values explain the impact of each input variable on a specific prediction. They are calculated by recursively comparing the model's prediction for a given input vector to the predictions it would make if that input vector was replaced with random values.

Both permutation importance and SHAP values can be calculated using the following Python code:

Python

import numpy as np

import pandas as pd

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

from sklearn.ensemble import RandomForestRegressor

from shap import TreeExplainer

# Load the data

data = pd.read\_csv('energy\_consumption.csv')

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data.drop(columns=['energy\_consumption']), data['energy\_consumption'], test\_size=0.25)

# Create a random forest model

model = RandomForestRegressor()

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

# Calculate permutation importance

permutation\_importance = np.mean([np.abs(model.predict(X\_test[np.random.permutation(i)]).mean() - model.predict(X\_test).mean()) for i in range(X\_test.shape[1])])

# Calculate SHAP values

explainer = TreeExplainer(model)

shap\_values = explainer.shap\_values(X\_test)

# Print the results

print('Permutation importance:', permutation\_importance)

print('SHAP values:', shap\_values)

The output of this code will be a list of permutation importance scores and a list of SHAP values for each input variable. The higher the permutation importance score or SHAP value for a variable, the more important it is to the model.

In addition to permutation importance and SHAP values, there are a number of other methods that can be used to measure the interpretability of energy consumption models. These methods include:

* Partial dependence plots: Partial dependence plots show the effect of each input variable on the model's predictions, while holding all other variables constant.
* Feature selection: Feature selection methods can be used to identify the most important input variables for the model.
* Decision trees: Decision trees are inherently interpretable, as they can be represented as a set of rules that explain how the model makes its predictions.

The best approach for measuring the interpretability of an energy consumption model will depend on the specific model and the intended use of the model. However, the methods described above provide a good starting point.

**MODEL** **EVALUATION**

To measure energy consumption model evaluation in Python, you can use the following steps:

1. Define your evaluation metrics. Common metrics for energy consumption model evaluation include:
   * Mean absolute error (MAE): The average of the absolute differences between the predicted and actual energy consumption values.
   * Mean squared error (MSE): The average of the squared differences between the predicted and actual energy consumption values.
   * Root mean squared error (RMSE): The square root of the MSE.
   * Coefficient of determination (R2): A measure of how well the model explains the variation in the actual energy consumption values.
2. Calculate the evaluation metrics on a held-out test set. This is a set of data that was not used to train the model. This will help you to assess how well the model generalizes to unseen data.
3. Compare the evaluation metrics to a baseline model. This could be a simple model, such as the mean of the target variable, or a more complex model that is not specifically designed to predict energy consumption.
4. Interpret the results If the evaluation metrics for your model are better than those for the baseline model, then you can conclude that your model is effective at predicting energy consumption. However, it is important to note that no model is perfect, and there will always be some error between the predicted and actual values.

Here is a simple example of how to measure energy consumption model evaluation in Python:

Python

import numpy as np

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

# Load the training and test data

X\_train, y\_train = ...

X\_test, y\_test = ...

# Train the energy consumption model

model = ...

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

# Make predictions on the test data

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

# Calculate the evaluation metrics

mae = mean\_absolute\_error(y\_test, y\_pred)

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

rmse = np.sqrt(mse)

r2 = model.score(X\_test, y\_test)

# Print the evaluation metrics

print('MAE:', mae)

print('MSE:', mse)

print('RMSE:', rmse)

print('R2:', r2)

You can then compare the evaluation metrics for your model to those for a baseline model to assess how well your model generalizes to unseen data.

**INNOVATION**

* Using machine learning to predict energy consumption: Machine learning can be used to train models on historical energy consumption data to predict future consumption. This can help businesses and organizations to better manage their energy use and reduce costs.
* Developing new sensors and devices to measure energy consumption: New sensors and devices are being developed that can measure energy consumption more accurately and efficiently. These sensors can be used to monitor energy consumption in real time and identify areas where energy savings can be made.
* Developing software tools to visualize and analyze energy consumption data: Software tools are being developed to help businesses and organizations to visualize and analyze their energy consumption data. This can help them to identify trends, patterns, and anomalies in their energy use.
* PowerAPI: PowerAPI is a Python library for measuring power consumption on a variety of devices, including laptops, desktops, servers, and IoT devices. It can be used to measure power consumption in real time, or to collect data over time to analyze trends.
* Energy Dashboard: Energy Dashboard is a web-based application for visualizing and analyzing energy consumption data. It supports a variety of data sources, including PowerAPI, and provides a variety of charts and reports to help businesses and organizations understand their energy use.
* WattTime: WattTime is a Python library that provides real-time information on carbon emissions associated with electricity generation. It can be used to develop applications that help businesses and organizations to reduce their carbon footprint by choosing to use renewable energy when it is available.

Overall, Python is a versatile and powerful language for developing innovative solutions to measure energy consumption. As the technology continues to develop, we can expect to see even more innovative Python-based solutions emerge in this area.

In addition to the above, here are some other ways to innovate in measuring energy consumption using Python:

* Use Python to develop new algorithms for measuring energy consumption more accurately and efficiently. For example, you could develop an algorithm to measure energy consumption at the individual device level, rather than the aggregate level.
* Use Python to develop new applications that help businesses and organizations to reduce their energy consumption. For example, you could develop an application that recommends energy-saving measures to businesses, or an application that helps businesses to track their progress in reducing their energy consumption.

**CONCLUSION**

Measuring energy consumption in Python is a powerful way to identify opportunities to reduce greenhouse gas emissions and improve energy efficiency. By collecting data, cleaning and preparing the data, analyzing the data, and developing machine learning models, Python can be used to develop comprehensive energy management strategies.

**FUTURE WORK**

There are a number of areas where future work can be done to improve energy consumption measurement in Python. One area is to develop new machine learning algorithms that can more accurately predict future energy use. Another area is to develop more sophisticated data analysis tools that can help users to identify patterns and trends in energy consumption. Finally, there is a need to develop more user-friendly interfaces that make it easier for people to use Python to measure energy consumption.